Pipelining of modular multiplication operations for efficient trust computation in decentralized cyber-physical environments

V.I. Petrenko, M.P. Sutormin

***Abstract* — Modern decentralized cyber-physical environments (DCPEs) are dynamic environments where ensuring trust between autonomous agents is a critical task, requiring efficient methods for trust computation between agents under resource constraints. This study aims to reduce the computational costs in trust evaluation, caused by the quadratic complexity of traditional methods, limiting their applicability in resource-constrained devices. To address this issue, the paper proposes pipelining modular multiplication operations to accelerate computations. This approach enables parallel data processing, reduces energy consumption, and ensures deterministic operation execution times, which are crucial for dynamic environments. The integration of pipelined modular multiplication operations, implemented at the hardware level, into a trusted interaction model, as well as their application in combination with blockchain technologies for decentralized updating of the trust matrix, is considered. Experimental results demonstrate a six-fold acceleration of modular multiplication operations compared to classical methods, as well as enhanced system resilience against attacks. The implementation of the proposed method opens up new possibilities for creating scalable and energy-efficient DCPEs capable of operating in highly dynamic and uncertain conditions.**

***Keywords* — decentralized cyber-physical systems, pipelined multipliers, modular operations, trust matrix, hardware acceleration, blockchain, energy efficiency.**

I. INTRODUCTION

Modern decentralized cyber-physical environments (DCPEs) [1] are complex dynamic systems where multiple autonomous entities - both physical (robots, sensors) and virtual (software agents) - interact. A key challenge in such systems is ensuring trusted interactions among agents, particularly in the absence of centralized control. Trust here serves as a quantitative measure of confidence in the reliability, competence, and predictability of other system participants, computed based on multitude of factors such as interaction history and current context. However, as DCPEs scale and the number of interactions grows, traditional trust assessment methods face high computational costs, which are especially critical for resource-constrained devices [2], such as mobile robots or embedded systems.

High computational costs are due to the need for frequent recalculation of trust levels in real time, which requires performing numerous modular operations, including multiplication and addition with weighting coefficients. These computational operations are the basis for weighted trust metrics, but their software implementation on general-purpose processors often proves to be too slow and energy-intensive. This paper proposes solving these problems through pipelined modular multiplication operations implemented at the hardware level.

Pipelined modular multiplication operations are highly efficient computational methods optimized for sequential modular multiplications with minimal delays and resource usage. Their key advantage lies in parallel computation organization [3, 4], significantly accelerating data processing compared to sequential algorithms. Integrating such methods into the computational systems of robotic agents optimizes critical operations without increasing the load on the central processor.

The scientific novelty of this work lies in the application of pipelined multiplication algorithms in a trusted interaction model [5], which has not been previously explored in the context of decentralized cyber-physical systems. The proposed approach not only speeds up computations but also reduces energy consumption, which is particularly important for autonomous agents operating under resource constraints. Additionally, hardware implementation enhances resilience against potential attacks on system software components.

The subsequent sections of the paper will detail the architecture of the proposed solution, including the formalization of trust computation using pipelined multipliers, methods for their integration into agents, and experimental results validating the approach's effectiveness.

II. CURRENT STATE OF THE ART

Decentralized cyber-physical environments (DCPEs) consist of interacting physical and virtual entities - robots, sensors, software agents, and computational nodes - united to achieve common goals. Their defining feature is the absence of centralized control: interactions among entities are governed by dynamic connections, and the system structure adapts to environmental changes and participant actions. Each entity has unique attributes (identifier, location, resource) that determine its role, behavior, and

Article received on May 21, 2025.

Petrenko Vyacheslav Ivanovich, Head of the Department of Information protection arrangement and technologies, North-Caucasus Federal University, Stavropol, Russia (email: vipetrenko@ncfu.ru).

Sutormin Matvey Pavlovich, Student of Information Security, North-Caucasus Federal University, Stavropol, Russia (email: sutorminp@gmail.com).

resource access rights. Active subjects (agents) initiate actions and make decisions, while passive entities (data, devices) serve as operation targets.

Trust is the basis of interaction in decentralized architectures. It reflects confidence in the reliability, competence, and predictability of participants, formed through analyzing their actions, interaction history, and context. For example, in multi-agent robotic systems (MARS), the introduction of rogue robots or misinformation dissemination necessitates algorithms capable of dynamically assessing agents' usefulness to the collective. Trust models like the Buddy Security Model (BSM) [6] and reputation systems offer solutions through iterative metric updates. In the works of Zikratov [7–9] trust is defined as the willingness to interact based on an agent's actions, while reputation is a stable assessment of its qualities, distinguishing legitimate participants from malicious ones.

Let's consider the main computational challenges that arise when calculating trust. Algorithmic complexity in this context characterizes the mathematically expressed dependence of the required computational resources on the number of input data. The main complexity arises from the high load of real-time trust matrix recalculations. This complexity is expressed as $O(n^2)$, indicating quadratic growth in required computations as the number of agents n increases. In practice, this leads to exponential computational load growth even with a small increase in participants, as each agent's trust relationships with all others must be calculated. Dynamic environments necessitate continuous execution of these $O(n^2)$ operations, including multiplying weight coefficients by reliability, competence, and honesty metrics [10]. Traditional software implementations on general-purpose processors lack the necessary speed and energy efficiency, especially for resource-constrained devices. Asymmetric trust evaluations (e.g., agent A trusts B more than B trusts A) further complicate computations, requiring modular operation optimization.

Modular multiplication and addition operations underpin cryptographic methods (RSA) [11–13], blockchain technologies [14], and weighted trust metrics. For example, in robot target allocation algorithms, they are used to calculate action efficiency and verify transaction integrity [15]. However, software implementations of such operations often become bottlenecks due to delays and high energy consumption, highlighting the need for hardware solutions.

The proposed approach leverages pipelined modular multiplication, which ensures high performance through parallel data processing at different computation stages. Unlike classical multiplication, pipelined implementation breaks the operation into sequential stages executed in parallel for different data streams. Applying this method in robotic agents enhances system resilience against attacks and ensures deterministic execution times for critical operations.

Thus, combining reputation-based trust models with hardware-accelerated modular operations paves the way for resilient DCPEs capable of operating under uncertainty and dynamically adapting to threats. Further research focuses on optimizing multiplier architectures and their integration into heterogeneous environments where performance and security requirements vary by context.

III. Decentralized Trust Matrix Update Using Blockchain Technology

Decentralized trust matrix updates in cyber-physical environments can be organized through blockchain technology, ensuring transparency, immutability, and distributed data storage. Each agent in the environment acts as a blockchain node, recording interactions, verifying transactions, and locally computing trust based on consensus rules. Interactions among agents - such as task execution, data exchange, or detecting malicious behavior - are recorded as transactions. These transactions are grouped into blocks, validated by the network via a chosen consensus mechanism (e.g., Practical Byzantine Fault Tolerance for resource-constrained devices [16]), and added to the distributed ledger. Each block contains the hash of the previous block, ensuring interaction history integrity.

Practical Byzantine Fault Tolerance (PBFT) is a consensus algorithm designed to ensure distributed system resilience against Byzantine failures, where some nodes may malfunction or act maliciously. PBFT achieves agreement among nodes through a multi-stage message exchange: a client sends a request to a primary node, which initiates a pre-prepare phase by broadcasting a proposal to other participants. Validator nodes verify the proposal, exchange confirmations, and finalize the result if a majority (at least 2/3+1) supports its correctness. This allows the system to remain operational even with up to f malicious nodes, where the total number of participants is $N \geq 3f+1$. PBFT is particularly effective in closed networks with a known, limited number of participants, such as consortium blockchains or decentralized cyber-physical systems, where low latency and energy efficiency are critical [18]. Unlike resource-intensive Proof of Work algorithms, PBFT avoids complex computations, making it suitable for low-power devices. However, its scalability diminishes as the number of nodes increases due to quadratic growth in communication volume, limiting its applicability in global networks. In decentralized trust matrices, PBFT ensures reliable update coordination, maintaining system integrity and data authenticity even with malicious agents.

Let's consider the Proof of Stake (PoS) consensus mechanism [19], an alternative to energy-intensive Proof of Work. Unlike classical algorithms, PoS eliminates the need for resource-heavy computations by probabilistically selecting validators based on their stake (ownership) of cryptographic assets. This significantly reduces system energy consumption, which is critical for autonomous DCPE agents operating under resource constraints. Combined with hardware-accelerated modular operations via pipelined multipliers, PoS balances security and efficiency: validators with larger stakes gain increased block creation rights, but their actions are automatically verified by other participants through cryptographic signatures. This minimizes Sybil attack risks and reduces trust matrix update delays, as validators are incentivized to maintain their reputation to preserve their stake. However, PoS adoption

requires addressing initial token distribution and power centralization challenges, especially in heterogeneous environments with uneven resource distribution among agents.

Further fault tolerance and consensus speed improvements can be achieved via the Tendermint [20] algorithm, combining PBFT with PoS elements. In Tendermint, validators participate in a multi-round voting process, where block finalization requires approval from at least 2/3 of participants, ensuring Byzantine fault resilience even in dynamically changing networks. Tendermint integration into DCPEs enables deterministic, rapid trust matrix updates: each agent, acting as a network node, participates in transaction verification, while pipelined modular multiplication accelerates digital signature and hash checks. This reduces block formation time to seconds, critical for real-time systems. Additionally, Tendermint's "slashing" (penalizing malicious behavior) naturally complements reputation-based trust models, automatically downgrading agents attempting to disrupt consensus.

Pipelined modular multiplication accelerates operations required for RSA digital signature generation and verification, critical for frequent validator changes and high transaction volumes. Parallel hardware-level data processing reduces staking share verification and block formation delays, ensuring compliance with dynamic environment time constraints. For Tendermint, where multi-round consensus requires mass signature verification, pipelined modular multiplication minimizes computation delays at each voting stage, enabling block finalization in fractions of a second. This is especially important for high-frequency trust matrix updates, where each agent must promptly confirm its reputation via cryptographically secured transactions. Moreover, pipelined multipliers' energy efficiency reduces load on resource-constrained devices, allowing them to participate in consensus without compromising autonomy.

IV. Computation Challenges

Cryptographic operations like RSA digital signature verification underpin decentralized cyber-physical environment (DCPE) security but face challenges due to the computational complexity of modular operations. In RSA algorithms [21] signatures are generated via modular exponentiation, which reduces to a sequence of modular multiplications. For large numbers (2048 bits or more), each multiplication requires processing hundreds of digits, and their sequential execution on general-purpose processors leads to critical delays. For example, RSA-2048 signature verification involves up to $O(n^2)$ elementary operations, where n is the modulus bit length, creating a quadratic computation time dependency on key size. In dynamic DCPEs, where thousands of agents simultaneously update the trust matrix via signed transactions, this becomes a bottleneck, limiting system throughput and increasing energy consumption.

Beyond computation volume, determinism poses a challenge. Software-based modular multiplication on general-purpose processors suffers from execution time variations due to instruction pipeline and caching, making systems vulnerable to timing attacks. Moreover, resource-constrained devices like autonomous robots or sensors cannot sustain high-frequency real-time trust recalculations using traditional methods. This directly impacts system resilience: signature verification delays may lead to accepting outdated or compromised data, undermining trust matrix integrity.

Pipelined modular multiplication addresses these issues. Classical multiplication of two n-bit numbers requires $O(n^2)$ operations, as each digit of one number is sequentially multiplied by all digits of the other. Pipelined architectures split the operation into n independent stages executed in parallel for data streams. While the first stage processes the least significant digit of the current number pair, the next stage begins working on the previous result for a new pair. Thus, for k number pairs, execution time is $n + k - 1$ cycles, corresponding to linear complexity $O(n)$.

Energy efficiency is achieved through hardware implementation: each pipeline stage is optimized at the logic circuit level, eliminating software overhead (e.g., context switching or caching). This enables devices to perform cryptographic operations (RSA signature verification) faster and with lower energy consumption.

V. Methodology

The RSA algorithm [22] exemplifies modular multiplication applications in blockchain cryptography and decentralized cyber-physical systems [23]. RSA operations involve large prime numbers: two primes p and q are selected, their product $n = p \times q$ is computed, which defines the modulus of the system. The most important step is calculating the Euler function $\varphi(n) = (p\text{-}1) \times (q\text{-}1)$, which together with the public exponent e forms the basis for generating the private key d through the modular inversion operation $d \equiv e^{-1} \bmod \varphi(n)$. RSA cryptographic transformations rely entirely on modular arithmetic: message m encryption is $c \equiv m^e \bmod n$, and decryption is $m \equiv c^d \bmod n$, where exponentiation is a sequence of modular multiplications. In DCPEs, these mathematical mechanisms are applied in several key areas. RSA-based digital signatures authenticate transactions and messages among agents, similar to their role in blockchain networks. Data and participant authenticity verification also relies on modular arithmetic, particularly crucial in decentralized settings. For confidential information like trust parameters or critical commands, public-key encryption protects against unauthorized access.

These operations' efficiency directly impacts system performance. For example, frequent trust matrix recalculations or intensive signed message exchanges benefit from hardware-implemented modular multiplication [24, 25] via pipelined multipliers or specialized cryptographic coprocessors, significantly reducing computational costs. This is critical for resource-constrained devices: autonomous robots, sensors, or embedded systems requiring a balance between processing speed, energy consumption, and security. Thus, modular multiplication optimization is not merely a technical task but a key factor

determining DCPE scalability and reliability.

Pipelined modular multiplication methods combine high performance with efficient resource use. Their defining feature is splitting multiplication into sequential stages executed in parallel for different operands, enabling minimal-delay data stream processing. In DCPEs requiring frequent inter-agent trust recalculations, such devices are indispensable, providing necessary computation speeds under strict energy constraints.

The authors have developed a pipelined modular multiplication method as follows. Let

$$C \equiv (A \cdot B) \bmod P, \quad (1)$$

where A and B are positive integers, $0 \le A$, $B < P$, called the multiplicand and the multiplier, respectively;

P is a positive integer, called the modulus;

C is a positive integer, the product of A and B, reduced modulo P.

Moreover:

$$A = a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \ldots + a_1 \cdot 2 + a_0, \quad (2)$$

$$B = b_{n-1} \cdot 2^{n-1} + b_{n-2} \cdot 2^{n-2} + \ldots + b_1 \cdot 2 + b_0, \quad (3)$$

$$P = p_{n-1} \cdot 2^{n-1} + p_{n-2} \cdot 2^{n-2} + \ldots + p_1 \cdot 2 + p_0, \quad (4)$$

$$C = c_{n-1} \cdot 2^{n-1} + c_{n-2} \cdot 2^{n-2} + \ldots + c_1 \cdot 2 + c_0, \quad (5)$$

where a_i, $i = \overline{0, n-1}$ are coefficients taking the value 0 or 1 depending on the value of the number A;

b_i, $i = \overline{0, n-1}$ are coefficients taking the value 0 or 1 depending on the value of the number B;

p_i, $i = \overline{0, n-1}$ are coefficients taking the value 0 or 1 depending on the value of the modulus P;

c_i, $i = \overline{0, n-1}$ are coefficients taking the value 0 or 1 depending on the value of the product C;

n is the number of digits in the representation of numbers.

The task is to find product C modulo P given A and B. The product of A and B modulo P can be expressed as follows:

$$(A \cdot B) \bmod P = (a_{n-1} \cdot 2^{n-1} B + a_{n-2} \cdot 2^{n-2} B + \ldots + a_1 \cdot 2B + a_0 \cdot B) \bmod P \quad (6)$$

Let's introduce the following notation:

$$B'_i = (2^i \cdot B) \bmod P, \text{ where } i = (0, \ldots, n\text{-}1). \quad (7)$$

Then the expression can be written as:

$$(a_{n-1} \cdot B'_{n-1} + a_{n-2} \cdot B'_{n-2} + \ldots + a_1 \cdot B'_1 + a_0 \cdot B'_0) \bmod P. \quad (8)$$

Obviously, that

$$B'_0 = B. \quad (9)$$

Thus, computing expression (6) in pipelined mode reduces to the following steps.

At pipeline stage 1, compute:

$$B'_1 = (2 \cdot B) \bmod P, \quad (10)$$

$$t_0 = a_0 \cdot B. \quad (11)$$

At pipeline stage 2, compute:

$$B'_2 = (2 \cdot B'_1) \bmod P, \quad (12)$$

$$t_1 = (a_1 \cdot B'_1 + t_0) \bmod P. \quad (13)$$

And so on. At pipeline stage n, compute:

$$t_{n-1} = (a_{n-1} \cdot B'_{n-1} + t_{n-2}) \bmod P. \quad (14)$$

The value t_{n-1} ~ is the desired product.

For a stream of numbers A_j and B_j modulo P_j, input clock-wise to the pipelined multiplier, j=1, 2, 3, compute:

$$C_j \equiv (A_j \cdot B_j) \bmod P_j. \quad (15)$$

Let $t_{i,j}$ denote the i-th partial product of the j-th pair of numbers A_j and B_j in j-th modulo P_j at pipeline stage i, where i =1, 2, … , n is the pipeline stage number, and j=1, 2, 3, …, is the device clock cycle. Then:

$$t_{i,j} = (a_{i\text{-}1,j} \cdot B'_{i-1,\,j} + t_{i\text{-}1,j}) \bmod P_j \quad (16)$$

where $a_{i\text{-}1,j}$ are coefficients in expression (2) for A_j,

$$B'_{i,\,j} = (2^i \cdot B_j) \bmod P_j, \quad (17)$$

$$t_{0,j} = a_{0,j} \cdot B_j. \quad (18)$$

Obviously, that

$$B'_{0,\,j} = B_j. \quad (19)$$

At the n-*th* clock cycle of the device, at the n-th pipeline stage as per (16), for the pair of numbers A_1 and B_1 and modulus P_1 compute $a_{n-1,1} \cdot B'_{n-1,1}$ and $t_{1,1}$:

$$t_{n-1,1} = (a_{n-1,1} \cdot B'_{n-1,1} + t_{n-2,1}) \bmod P_1. \quad (20)$$

The developed method is technically implemented in [26] and [27]. Consider the hardware implementation of pipelined modular multiplication.

The pipelined modular multiplier is a device implementing the above method (see Fig. 1). Key components include:

- n parallel registers (1.1÷1.n) store intermediate values during computation.
- n keys (2.1÷2.n) control data flow based on multiplicand bits.
- (n−1) modulo-2 multipliers (3.1÷3.(n−1)) – perform number doubling with modulo reduction.
- (n-−1) modulo adders (4.1÷4.(n−1)) sum intermediate results with modulo reduction.

Input 5 receives multiplicand A, input 6 receives multiplier B, and input 7 receives the modulus P's inverse code. Clock signal is input at 9, and the multiplication result $C = (A \cdot B) \bmod P$ is output at 8.

The device operates in pipelined mode: each clock cycle inputs a new number pair A and B, and after n cycles, the result appears at the output. Simultaneously, n number pairs are processed, each at its pipeline stage. This approach significantly boosts performance compared to sequential computation.

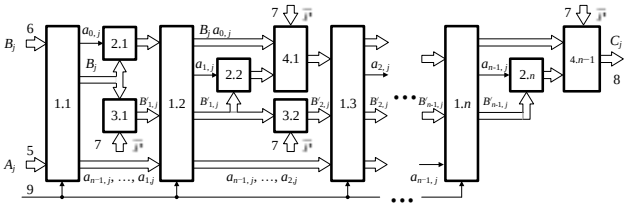

Fig. 1. Pipelined modular multiplier

To evaluate the device's efficiency versus the prototype [28], consider the multiplication time for 100 pairs of 16-bit numbers. The prototype device, handling 16-bit operands, includes 8 sequential transformation stages. Let $T_{\text{пр}}$ be the time for one stage. When subsequent computations can only begin after the current one finishes, total processing time for

100 number pairs is $800 \cdot T_{пр}$.

The proposed device implements pipelined data processing with 16 sequential stages. Let $T_{из}$ be the time for one pipeline stage. Under similar load, total computation time for 100 number pairs is $132 \cdot T_{из}$.

Assuming $T_{пр} \approx T_{из}$ (and in practice, $T_{пр} > T_{из}$), the speedup B is estimated as the time ratio:

$$B = \frac{800 \cdot T_{пр}}{132 \cdot T_{из}} \approx 6.$$

Thus, the proposed device achieves a sixfold speedup versus the prototype for 100 pairs of 16-bit numbers. As number size and computation volume increase, the speedup grows.

VI. Performance Evaluation

The classical multiplication approach involves sequential operations. For one number pair, the number of stages (operations) is m. For 16-bit numbers, $m = 8$. Stage processing time is $t_{пр}$, and total time for N number pairs is calculated by the formula:

$$T_{\text{classic}} = N \cdot m \cdot t_{пр}.$$

The pipelined approach uses parallel data processing. Here, the number of stages (digits) is n (e.g., $n = 16$). Stage processing time is $t_{из}$, and total time for N number pairs is is determined as:

$$T_{\text{pipeline}} = (N + n - 1) \cdot t_{из}.$$

The pipelined method's speedup over the classical method is calculated by the formula :

$$S = \frac{T_{\text{classic}}}{T_{\text{pipeline}}} = \frac{N \cdot m \cdot t_{пр}}{(N + n - 1) \cdot t_{из}}.$$

If we assume that $t_{пр} \approx t_{из}$, the formula is simplified:

$$S = \frac{N \cdot m}{N + n - 1}.$$

In [256] an example with parameters $N = 100$, $m = 8$, $n = 16$ is given. The speedup calculation yields:

$$S = \frac{100 \cdot 8}{100 + 16 - 1} = \frac{800}{115} \approx 6.96 \approx 7 \text{(с учётом округления)}.$$

Analyzing speedup dependency on the number of pairs N. For fixed $n = 16$ and $m = 8$, the trend is observed in Fig. 2:

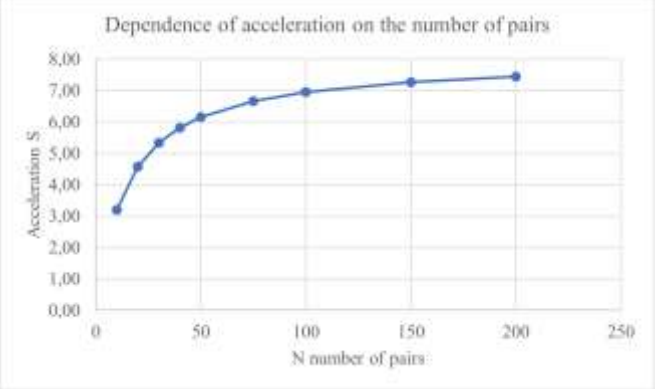

Fig. 2. Speedup vs. Number of pairs

As N increases, speedup approaches $m = 8$, demonstrating asymptotic dependency.

Analyzing speedup dependency on digit count n is presented in Fig.3. With $N = 100$ and $m = \frac{n}{2}$:

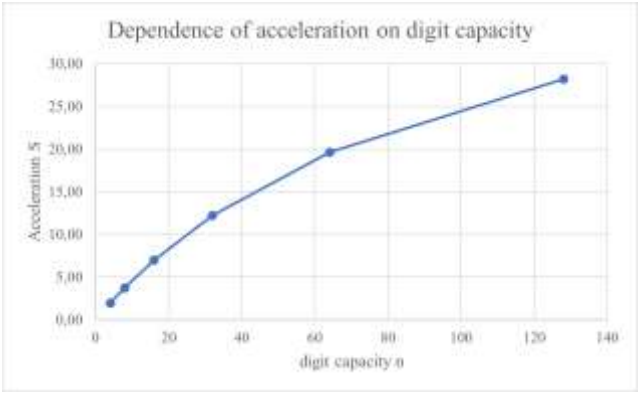

Fig. 3. Speedup vs. Digit count

Increasing n boosts speedup, but growth slows due to the denominator $N + n - 1$.

The formula for calculations is:

$$S = \frac{N \cdot m}{N + n - 1},$$

where $m = \frac{n}{2}$ (for digit count analysis). For example, for $n = 24$, $N = 50$:

$$S = \frac{50 \cdot 12}{50 + 24 - 1} = \frac{600}{73} \approx 8.22.$$

Pipelined architecture achieves significant speedup via parallel data processing. For 16-bit numbers and 100 pairs, speedup reaches ~6x. The formula $S = \frac{N \cdot m}{N + n - 1}$ reflects speedup dependency on pair count N and digit count n. Graphs show speedup grows with N and n but is constrained by pipeline structure. For example, as $N \to \infty$, speedup approaches m, while increasing n slows growth due to the denominator.

VII. Discussion

Despite its advantages, deploying pipelined multipliers for trust computation in decentralized cyber-physical environments has limitations. The primary technical constraint is reliance on hardware implementation accuracy - even minor modular operation errors can distort final trust metrics, critical in systems where trust influences key decisions. Additionally, integrating specialized hardware into existing systems incurs significant overhead, including agent architecture modifications, new interface development, and compatibility assurance with existing protocols. These factors may substantially increase deployment time and costs, especially in large-scale heterogeneous environments with diverse agent platforms and computational capabilities.

Future development prospects extend beyond trust computation. One promising direction is applying pipelined multipliers in other DCPE components, particularly cryptographic algorithms. In systems using homomorphic

encryption, digital signatures, or zero-knowledge protocols, hardware-accelerated modular operations could significantly boost performance while maintaining security. Moreover, pipelined architectures could benefit distributed machine learning algorithms requiring efficient matrix operations over finite fields or IoT systems [29] prioritizing energy efficiency and high-speed data processing. Further research could focus on developing universal hardware solutions dynamically adaptable to various moduli and algorithms, broadening applicability and easing integration into heterogeneous computing environments.

VIII. Conclusion

Implementing pipelined multiplication methods in decentralized cyber-physical environments demonstrates significant potential for optimizing inter-agent trust computations. The proposed approach enhances performance via parallel multiplication processing, reduces trust matrix update delays, and lowers system energy consumption - critical for resource-constrained autonomous agents. Hardware-implemented pipelined multipliers accelerate critical computations (e.g., weighted sums in trust formulas) and improve security by standardizing operation execution times, hindering timing attacks.

However, successful integration faces technical challenges, including hardware accuracy dependencies and high adaptation costs for existing systems. These limitations necessitate careful balancing of performance and reliability, alongside universal interface development for cross-platform compatibility.

Future research could expand pipelined multiplication applications beyond trust computation, such as cryptographic protocols (e.g., finite field operation acceleration), distributed machine learning, or real-time systems. Reconfigurable hardware solutions adaptable to dynamic moduli and contexts could enhance system flexibility and scalability.

In summary, the proposed methodology confirms that combining algorithmic optimizations with hardware acceleration is key to building efficient, secure decentralized systems. This paves the way for intelligent systems operating in dynamic, uncertain environments while maintaining resilience and energy efficiency.

References

1. Petrenko V. I. et al. Method of trusted agent interaction in a decentralized cyber-physical environment based on distributed ledger technology // Caspian Journal: Management and High Technologies. — 2023. — No. 3 (63). — P. 115.
2. Roig P. J. [et al.]. Modeling an Edge Computing Arithmetic Framework for IoT Environments // Sensors 2022, Vol. 22, Page 1084. 2022. No. 3 (22). C. 1084.
3. Samofalov K.G., Lutskiy G.M. Fundamentals of the theory of multilevel pipelined computing systems. — M.: Radio i svyaz, 1989. — 272p.
4. Orton G., Peppard L., Tavares S. A design of a fast pipelined modular multiplier based on a diminished-radix algorithm // Journal of Cryptology. 1993. No. 4 (6). P. 183–208.
5. Ma C. [et al.]. Trusted AI in Multiagent Systems: An Overview of Privacy and Security for Distributed Learning // Proceedings of the IEEE. 2023. No. 9 (111). P. 1097–1132.
6. Page J., Zaslavsky A., Indrawan M. A buddy model of security for mobile agent communities operating in pervasive scenarios // Proceedings of the Second Workshop on Australasian Information Security, Data Mining and Web Intelligence, and Software Internationalisation. 2004. (32). P. 17–25.
7. Zikratov I. A., Zikratova T. V., Lebedev I. S. Trust model of information security of multi-agent robotic systems with decentralized control // Scientific and technical bulletin of information technologies, mechanics and optics. - 2014. - No. 2 (90). - P. 47-52.
8. Zikratov I. A. [et al.]. Construction of a model of trust and reputation for objects of multi-agent robotic systems with decentralized control // Scientific and Technical Bulletin of Information Technologies, Mechanics and Optics. - 2014. - No. 3 (91). - P. 30-38
9. Zikratov I. A. [et al.]. Security model of mobile multi-agent robotic systems with collective control // Scientific and technical bulletin of information technologies, mechanics and optics. - 2017. - Vol. 17, No. 3. - P. 439–449. - DOI: 10.17586/2226-1494-2017-17-3-439-449.
10. Amini M. R., Baidas M. W. Availability-Reliability-Stability Trade-Offs in Ultra-Reliable Energy-Harvesting Cognitive Radio IoT Networks // IEEE Access. 2020. (8). C. 82890–82916.
11. Ullah S. [et al.]. Elliptic Curve Cryptography; Applications, challenges, recent advances, and future trends: A comprehensive survey // Computer Science Review. 2023. (47). C. 100530.
12. Singh S., Maakar S. K., Kumar S. A Performance Analysis of DES and RSA Cryptography 2013.
13. Lysyanskaya A. Security analysis of RSA-BSSA 2023. P. 251–280.
14. Naser S. M. Cryptography: From the ancient history to now, it's applications and a new complete numerical model // International journal of mathematics and statistics studies. 2021. No. 3 (9). P. 11–30.
15. Fryer D. [et al.]. Checking the integrity of transactional mechanisms // ACM Transactions on Storage. 2014. No. 4 (10).
16. Bohm H., Distler T., Wagemann P. TinyBFT: Byzantine Fault-Tolerant Replication for Highly Resource-Constrained Embedded Systems // 2024 IEEE 30th Real-Time and Embedded Technology and Applications Symposium (RTAS). 2024. P. 225–238.
17. Wang Y., Zhong M., Cheng T. Research on PBFT consensus algorithm for grouping based on feature trust // Scientific Reports 2022 12:1. 2022. No. 1 (12). P. 1–12.
18. Ali A. [et al.]. Securing Secrets in Cyber-Physical Systems: A Cutting-Edge Privacy Approach with Consortium Blockchain // Sensors 2023, Vol. 23, Page 7162. 2023. No. 16 (23). P. 7162.
19. Chinnam R. K. [et al.]. Enhancing IoT Security and Efficiency with DPOS Enabled Blockchain and IPFS Integration // 2024 2nd International Conference Computational and Characterization Techniques in Engineering and Sciences, IC3TES 2024. 2024.
20. Amoussou-Guenou Y. [et al.]. Dissecting Tendermint // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2019. (11704 LNCS). P. 166–182.
21. Gennaro R., Krawczyk H., Rabin T. RSA-based undeniable signatures // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 1997. (1294). P. 132–149.
22. Gupta N., Jain A. K. RSA Based Consensus Algorithm for Lightweight Private Blockchain Network // ITM Web of Conferences. 2023. (54). P. 03003.
23. Milanov E. [et al.]. The RSA algorithm // RSA laboratories. 2009. No. 2 (42). P. 1–11.
24. Pavan Kumar C. H., Sivani K. Implementation of efficient parallel prefix adders for residue number system // International Journal of Computing and Digital Systems. 2015. No. 4 (4). P. 295–300.
25. Ma C. [et al.]. Trusted AI in Multiagent Systems: An Overview of Privacy and Security for Distributed Learning // Proceedings of the IEEE. 2023. No. 9 (111). P. 1097–1132.
26. Petrenko V. I. Pipelined modulo multiplier // Patent for invention 2797164 C1, Published 07/19/2021 Bulletin No. 20.
27. Petrenko V. I., Sutormin M. P., Puiko D. D. Pipelined modulo multiplier // Patent for invention 2797164 C1, Published 07/19/2021 Bulletin No. 20.
28. Petrenko V. I. Modulo multiplier // Patent for invention 2751802 C1, Published 07/19/2021 Bulletin No. 20.
29. Alzubi J. A. Blockchain-based Lamport Merkle digital signature: authentication tool in IoT healthcare // Computer Communications. 2021. (170). P. 200–208.