НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
май–июнь 2026 Том 26 № 3 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
May–June 2026 Vol. 26 No 3 http://ntv.ifmo.ru/en/
ISSN 2226-1494 (print) ISSN 2500-0373 (online)

НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

doi: 10.17586/2226-1494-2026-26-3-525-531

Blockchain-based logging of data lifecycle events in cloud storage

Vladimir V. Eremuk[1]✉, Alexandr V. Kasyanov[2], Liubov A. Primak[3], Victor A. Romashov[4]

[1,3,4] ITMO University, Saint Petersburg, 197101, Russian Federation
[2] Saint Petersburg Electrotechnical University "LETI", Saint Petersburg, 197022, Russian Federation

[1] polar.vl@yandex.ru✉, https://orcid.org/0000-0003-2337-0015
[2] kasjanov@inbox.ru, https://orcid.org/0009-0000-3319-8357
[3] laprimak@itmo.ru, https://orcid.org/0009-0002-1462-3156
[4] whiviktor@gmail.com, https://orcid.org/0000-0002-1999-5381

Abstract

This study presents a method for registering data lifecycle events in cloud storage under an untrusted cloud provider infrastructure. According to the adopted threat model, if the cloud provider is compromised, an adversary is capable of modifying, reordering, rolling back, and deleting stored data as well as the corresponding lifecycle event records. The novelty of the proposed method lies in binding each data version to a dual-signed lifecycle record anchored in a blockchain which enables detection of unauthorized modifications independently of the provider after transaction confirmation. The method is intended for deployment in systems that include cloud storage operated by an untrusted provider and client devices acting as nodes of a private blockchain network. Each create, modify, or delete operation is accompanied by the formation of a registration record containing the hash of the current data version, the hash of the previous version, the operation type, a timestamp, and a user identifier. The record is signed by both the client and the provider, after which a corresponding transaction is formed and submitted to a blockchain using the Tendermint consensus protocol. To accelerate verification procedures, registration records and transaction confirmation statuses are additionally stored on the provider side. The confirmation status is synchronized during subsequent accesses; data versions without confirmed transactions are available in read-only mode. Integrity verification includes matching the payload hash, verifying both signatures, and confirming the presence of the transaction in the blockchain. An experimental evaluation of the method was conducted in a private blockchain network using the Tendermint consensus protocol with 1,000 clients and 10 validators. The average transaction confirmation time achieved in the experiment was 1.3 s, with a peak value of 1.8 s. Client-side cryptographic operations required by the proposed logging method take less than 3 ms per data operation. Compared to commercial solutions (Amazon Web Services CloudTrail and Google Cloud Audit Logs), the proposed method enables audit execution immediately after transaction confirmation but introduces additional latency in the data operation path. Issues related to availability and prevention of physical data deletion is beyond the scope of this study.

Keywords

data lifecycle, blockchain, cloud storage, integrity checking, auditing

For citation: Eremuk V.V., Kasyanov A.V., Primak L.A., Romashov V.A. Blockchain-based logging of data lifecycle events in cloud storage. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2026, vol. 26, no. 3, pp. 525–531. doi: 10.17586/2226-1494-2026-26-3-525-531

УДК 004.75

Журналирование событий жизненного цикла данных в облачном хранилище на основе технологии блокчейн

Владимир Вадимович Еремук[1]✉, Александр Владимирович Касьянов[2], Любовь Андреевна Примак[3], Виктор Андреевич Ромашов[4]

[1,3,4] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация

[2] Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина), Санкт-Петербург, 197022, Российская Федерация

[1] polar.vl@yandex.ru✉, https://orcid.org/0000-0003-2337-0015

[2] kasjanov@inbox.ru, https://orcid.org/0009-0000-3319-8357

[3] laprimak@itmo.ru, https://orcid.org/0009-0002-1462-3156

[4] whiviktor@gmail.com, https://orcid.org/0000-0002-1999-5381

Аннотация

Введение. Представлен метод регистрации событий жизненного цикла данных в облачном хранилище в условиях недоверенной инфраструктуры облачного провайдера. Согласно построенной модели угроз, при компрометации облачного провайдера, злоумышленник имеет возможность модифицировать, переупорядочивать, откатывать и удалять хранимые данные и соответствующие им записи событий жизненного цикла. Новизна предложенного метода заключается в привязке каждой версии данных к двусторонне подписанной записи жизненного цикла с ее фиксацией в блокчейне, что обеспечивает обнаружение несанкционированных изменений независимо от провайдера после подтверждения транзакции. **Метод.** Метод предназначен для внедрения в системы, включающие облачное хранилище под управлением недоверенного провайдера и клиентские устройства, являющиеся узлами частной блокчейн-сети. Каждая операция создания, изменения или удаления данных сопровождается формированием регистрационной записи, содержащей хеш текущей и предыдущей версий данных, тип операции, метку времени и идентификатор пользователя. Запись подписывается на стороне клиента и провайдера, после чего формируется транзакция и отправляется в блокчейн с протоколом консенсуса Tendermint. Для ускорения проверок регистрационные записи и статус подтверждения транзакций дополнительно хранятся на стороне провайдера. Статус подтверждения синхронизируется при последующих обращениях; версии данных без подтвержденных транзакций доступны только для чтения. Проверка целостности включает сопоставление хеш-суммы, проверку подписей и проверку подтверждения транзакции в блокчейне. **Основные результаты.** Выполнена экспериментальная оценка метода в частной блокчейн-сети с протоколом консенсуса Tendermint, с 1000 клиентами и 10 валидаторами. В эксперименте достигнуто среднее время подтверждения транзакции в 1,3 с при пиковом значении 1,8 с. Выполнение криптографических операций, необходимых для работы изложенного метода журналирования, занимает менее 3 мс на стороне клиента на одну операцию над данными. **Обсуждение.** В сравнении с коммерческими аналогами (Amazon Web Services CloudTrail и Google Cloud Audit Logs), предложенный метод обеспечивает возможность выполнения аудита сразу после подтверждения транзакции, но требует дополнительных временных затрат при выполнении операций над данными. Задачи обеспечения доступности и предотвращения удаления данных не рассматриваются в рамках данного исследования.

Ключевые слова

жизненный цикл данных, блокчейн, облачное хранилище, проверка целостности, аудит, журналирование

Ссылка для цитирования: Еремук В.В., Касьянов А.В., Примак Л.А., Ромашов В.А. Журналирование событий жизненного цикла данных в облачном хранилище на основе технологии блокчейн // Научно-технический вестник информационных технологий, механики и оптики. 2026. Т. 26, № 3. С. 525–531 (на англ. яз.). doi: 10.17586/2226-1494-2026-26-3-525-531

Introduction

Data growth and rising computing costs motivate cloud outsourcing [1–5], but outsourcing introduces leakage and unauthorized-access risks [6–9]. Lifecycle logging is required for create/modify/delete operations [10], yet provider-hosted logs can be altered after compromise [11]. Related Denial-Of-Service (DoS) risks in Internet of Things/cloud environments further stress the need for resilient integrity mechanisms [12–17]. Managed audit services (e.g., Amazon Web Services CloudTrail and Google Cloud Audit Logs) are easy to deploy [18, 19], and retention/Write-Once Read-Many controls improve compliance [20], but evidence remains in the provider trust domain and is not natively bound to exact data versions. Provider-managed ledger services (e.g., Amazon Quantum Ledger Database and Azure Confidential Ledger) provide append-only storage [21, 22], yet are still provider-operated and do not directly enforce lifecycle semantics with dual-party non-repudiation. Existing secure-logging approaches improve tamper resistance [23–25], but practical cloud workflows still require explicit version binding and external verifiability. Thus, the practical trade-off is between minimal operation-path latency with delayed or provider-scoped audit visibility and slightly higher write latency with immediate external verifiability after commitment. We therefore propose a method that anchors dual-signed lifecycle entries in an external append-only blockchain ledger. The method combines version-to-entry digest binding, status synchronization for unconfirmed entries, and independent verification under a strong provider-compromise assumption. Its advantage is architectural trust-boundary separation, not dependence on a specific

national cryptographic standard. Here, data denotes any digital object subject to lifecycle operations.

Threat model

We consider an adversary $\mathcal{A}$ that can fully compromise provider infrastructure (control and data planes), including read/modify/reorder/delete of stored versions, metadata, and provider-local logs. The adversary may attempt payload substitution, rollback to older versions, selective erasure, and forgery/truncation of lifecycle records. We assume private signing keys of users/provider and auditor verification keys remaining uncompromised, and that the permissioned blockchain satisfies its standard security threshold (committed transactions cannot be silently rewritten). DoS and availability attacks are out of scope. Security goals are: (G1) tamper-evident integrity of confirmed lifecycle records, (G2) digest-based binding between delivered payload and registered event, (G3) dual-signature non-repudiation, and (G4) independent auditability via blockchain verification.

Preliminaries

We consider a system with up to $u_{\max}$ users $U = \{u_1, u_2, \ldots, u_{u_{\max}}\}$, a cloud service provider c, a blockchain b, and an auditor a. Let $P \in E(\mathbb{F}_p)$ be a base point on elliptic curve E, and let q be the order of P. Each participant has two key pairs generated according to GOST R 34.10-2012 [26]: session keys (k^{sess}, Q^{sess}) and signature keys (k, Q), where $k \in \mathbb{Z}_q$ and $Q = kP$. Signature generation is denoted by σ_k: $\{0, 1\}^* \to \mathbb{Z}_q \times \mathbb{Z}_q$ and verification by:

$$\mathrm{Verify}_Q(m, \sigma) = \begin{cases} 1, \text{ if the signature is correct} \\ 0, \text{ if the signature is incorrect} \end{cases},$$

where Q is the public verification key; m is a message; σ is a signature. For confidentiality we use the block cipher Kuznyechik [27]; for payload digests we use Streebog-256 [28, 29], denoted $\mathrm{H}(m)$. We denote by h the blockchain-native hash function used to derive transaction identifiers and/or header hashes [30]. Concatenation is denoted by '||'.

Choice of elliptic-curve domain parameters

The GOST R 34.10-2012[1] signature scheme uses domain parameters (p, α, β, q, P) defining curve E over $\mathbb{F}_p$, base point P, and subgroup order q. We instantiate (E, P, q) with standardized 256-bit prime-field parameter set id-tc26-gost-3410-2012-256-paramSetA in short Weierstrass form:

$$Y^2 = X^3 + \alpha X + \beta (\mathrm{mod} p).$$

Any approved equivalent parameter set with equivalent security level can be used without changing method procedures.

[1] GOST R 34.10-2012. Information Technology. Cryptographic Data Security. Processes of Formation and Verification of Electronic Digital Signature. 07.08.2012. Moscow, Standartinform. 20 p.

Registration entry structure

After each operation on version v of data item d with payload x, user u forms entry $e = (d, v, z, z', u, o, t)$, where $z = \mathrm{H}(x)$, z' is the previous-version digest (or 0 for $v = 1$), $o \in \{1, 2, 3\}$ denotes create/modify/delete, and t is UNIX time. The provider stores local records as $\ell_i = (e_i, \sigma_{u,i}, \sigma_{c,i}, r_i)$, where $r_i \in \{0, 1\}$ indicates blockchain confirmation. For each d, the provider exposes $L_d = \{\ell_j\}$; entries with $r_j = 0$ are unconfirmed.

Blockchain

The system uses a private blockchain network with users as nodes. The auditor does not participate in consensus but has read access for verification. A blockchain transaction is $y = (e, \sigma_u, \sigma_c)$, where e is a registration entry; σ_u, σ_c are signatures over $\mathrm{enc}(e)$. Transaction acceptance requires:

$$\mathrm{Verify}_u(\mathrm{enc}(e), \sigma_u) = 1$$
$$\mathrm{Verify}_c(\mathrm{enc}(e), \sigma_c) = 1.$$

Here Verify_u and Verify_c use public keys of u and c. Signature verification follows GOST R 34.10-2012. HotStuff and Tendermint are suitable consensus options [31–35]; implementation details are out of scope.

Method workflow overview

The method is executed through 6 procedures. First, registration R creates a lifecycle entry, obtains dual signatures, stores a local record with pending flag, and submits the corresponding transaction to the blockchain. Next, status synchronization S updates confirmation flags by checking committed transaction identifiers. Verification V then validates digest binding, signatures, and blockchain commitment for the selected version, with optional freshness checks when latest-version semantics is required. Read/modify handling M runs S and V before allowing version update via a new registration. Deletion D also runs S and V, then records and submits a signed deletion entry. Finally, audit A verifies confirmed records and performs bidirectional log-ledger consistency checks in scope.

Procedure R: Registering a lifecycle event

Input: user u, provider c, blockchain b; data id d; version v; payload x; operation $o \in \{1, 2, 3\}$; timestamp t.

Output: transaction y submitted to b; provider local log record ℓ created/updated.

1. Compute $z \leftarrow \mathrm{H}(x)$. Set $z' \leftarrow$ digest of the previous version (or 0 if undefined).
2. Form $e \leftarrow (d, v, z, z', u, o, t)$.
3. Compute user signature σ_u over $\mathrm{enc}(e)$ using the user's private signing key and send (x, e, σ_u) to c.
4. Provider checks $\mathrm{Verify}_u(\mathrm{enc}(e), \sigma_u) = 1$. If not, abort.
5. Provider computes signature σ_c over $\mathrm{enc}(e)$ using the provider's private signing key and returns σ_c to u.
6. User checks $\mathrm{Verify}_c(\mathrm{enc}(e), \sigma_c) = 1$. If not, abort.
7. Provider applies the storage action defined by o and appends local record $\ell \leftarrow (e, \sigma_u, \sigma_c, r = 0)$.
8. Form $y \leftarrow (e, \sigma_u, \sigma_c)$ and submit y to b.

Procedure S: Synchronizing confirmation status

Input: user u, provider c, blockchain b; data id d.

Output: updated confirmation flags r for all provider log records ℓ associated with d; (optional) set I of confirmed record indices returned to c.

1. Provider returns the local set $L_d = \{\ell_j\}$, where $\ell_j = (e_j, \sigma_{u,j}, \sigma_{c,j}, r_j)$.
2. For each $\ell_j \in L_d$:
   2.1. Reconstruct $y_j \leftarrow (e_j, \sigma_{u,j}, \sigma_{c,j})$.
   2.2. Compute $\tau_j \leftarrow \mathrm{h}(\mathrm{enc}(y_j))$.
   2.3. Query b for the commitment status of τ_j.
3. Let $I \leftarrow \{j | \tau_j$ is committed in $b\}$.
4. Send I to c (or directly update if the model allows).
5. Provider sets $r_j \leftarrow 1$ for all $j \in I$; leaves others unchanged ($r_j = 0$).

Procedure V: Data integrity verification

Input: user u, provider c, blockchain b; data id d; version v; payload x; entry e; signatures σ_u, σ_c; confirmation flag $r \in \{0, 1\}$.

Output: decision $f \in \{0, 1\}$ (accept/reject); (optional) report w describing failed checks.

1. If $r = 0$, set $f \leftarrow 0$ and stop.
2. Parse e as (d, v, z, z', u, o, t).
3. Check binding: if $\mathrm{H}(x) \neq z$, set $f \leftarrow 0$ and stop.
4. Check signatures: if $\mathrm{Verify}_u(\mathrm{enc}(e), \sigma_u) = 0$, set $f \leftarrow 0$ and stop; if $\mathrm{Verify}_c(\mathrm{enc}(e), \sigma_c) = 0$, set $f \leftarrow 0$ and stop.
5. Reconstruct $y \leftarrow (e, \sigma_u, \sigma_c)$ and compute $\tau \leftarrow \mathrm{h}(\mathrm{enc}(y))$.
6. Query b for a committed transaction with id τ. If none exists, set $f \leftarrow 0$ and stop.
7. If latest-version semantics is required for the current operation, check that there is no confirmed record for d with version index greater than v; if such a record exists, set $f \leftarrow 0$ and stop.
8. Set $f \leftarrow 1$.
9. (Optional) Populate w with the first failed check and auxiliary values ($\mathrm{H}(x)$, τ).

Procedure M: Data reading and modification

Input: user u, provider c, blockchain b; data id d; version v; payload x; timestamp t.

Output: decision $f \in \{0, 1\}$ (verified/read-only); payload x for read; (optional) transaction y for the modified version.

1. Run status synchronization $S(u, c, b; d)$ and obtain the log record $\ell = (e, \sigma_u, \sigma_c, r)$ for version v.
2. Run integrity verification $f \leftarrow \mathrm{V}(u, c, b; d, v, x, e, \sigma_u, \sigma_c, r)$; for modification requests, enable latest-version freshness checking in step 7 of V.
3. If $f = 0$, treat the version as read-only and stop.
4. If modification is requested, generate new payload x' and invoke registration $\mathrm{R}(u, c, b, d, v + 1, x', o = 2, t)$, which returns a transaction y submitted to b.

Procedure D: Deletion of a data version

Input: user u, provider c, blockchain b; data id d; version v; payload x; timestamp t.

Output: transaction y submitted to b; provider local log record ℓ created.

1. Run status synchronization $\mathrm{S}(u, c, b; d)$ and obtain $\ell = (e, \sigma_u, \sigma_c, r)$ for version v.
2. Run integrity verification $f \leftarrow \mathrm{V}(u, c, b; d, v, x, e, \sigma_u, \sigma_c, r)$. If $f = 0$, abort.
3. Set $z \leftarrow \mathrm{H}(x)$. Set $z' \leftarrow$ digest of the previous version (or 0 if $v = 1$).
4. Form deletion entry $e \leftarrow (d, v, z, z', u, 3, t)$.
5. Compute user signature σ_u over $\mathrm{enc}(e)$ using the user's private signing key and send (e, σ_u) to c.
6. Provider checks $\mathrm{Verify}_u(\mathrm{enc}(e), \sigma_u) = 1$. If not, abort.
7. Provider computes signature σ_c over $\mathrm{enc}(e)$ using the provider's private signing key and returns σ_c to u.
8. User checks $\mathrm{Verify}_c(\mathrm{enc}(e), \sigma_c) = 1$. If not, abort.
9. Provider deletes version (d, v) from storage and appends local record $\ell \leftarrow (e, \sigma_u, \sigma_c, r = 0)$.
10. Form $y \leftarrow (e, \sigma_u, \sigma_c)$ and submit y to b.

Procedure A: Integrity audit of registration entries

Input: auditor a, provider c, blockchain b; audit scope s (e.g., time window and/or a set of data ids).

Output: verdict $g \in \{0, 1\}$; audit report w (mismatches, missing transactions, signature failures).

1. Auditor requests from c the provider log subset $L_s = \{\ell_j\}$ in scope s, where $\ell_j = (e_j, \sigma_{u,j}, \sigma_{c,j}, r_j)$.
2. Initialize report $w \leftarrow \varnothing$.
3. For each $\ell_j \in L_s$:
   3.1. If $r_j = 0$, continue.
   3.2. Extract user id u_j from e_j.
   3.3. If $\mathrm{Verify}_{u_j}(\mathrm{enc}(e_j), \sigma_{u,j}) = 0$, append "invalid user signature $\sigma_{u,j}$" to w.
   3.4. If $\mathrm{Verify}_c(\mathrm{enc}(e_j), \sigma_{c,j}) = 0$, append "invalid provider signature $\sigma_{c,j}$" to w.
   3.5. Reconstruct $y_j \leftarrow (e_j, \sigma_{u,j}, \sigma_{c,j})$, compute $\tau_j \leftarrow \mathrm{h}(\mathrm{enc}(y_j))$.
   3.6. Query b for a committed transaction with id τ_j; if absent, append "missing committed transaction τ_j" to w.
4. Enumerate committed transactions in b within scope and check that each of them has a matching confirmed local record in L_s; if any is missing, append "missing confirmed local record for committed transaction" to w.
5. Set verdict $g \leftarrow 1$ if $w = \varnothing$; otherwise $g \leftarrow 0$.

Security analysis

A logging mechanism based solely on hash values is insufficient under a full provider-compromise model, since an adversary can alter payloads and recompute digests. In the proposed method, $z = \mathrm{H}(x)$ is embedded in a dual-signed entry committed to an append-only blockchain ledger; after commitment, undetected modification requires either signature forgery or ledger rewrite, which is infeasible under the stated assumptions. The method is therefore tamper-evident; availability/DoS and prevention of physical data deletion remain out of scope. Detection of representative attacks follows directly from the verification workflow: substitution is detected by $\mathrm{H}(x^*) \neq z$; rollback is detected by freshness checks against the latest confirmed record for d (monotonic v or t with digest linkage); local-log forgery is detected by signature failure and/or missing committed $\tau = \mathrm{h}(\mathrm{enc}(y))$; truncation is detected by bidirectional cross-checking between provider logs and committed ledger entries in audit scope.

G1 (tamper-evident integrity): confirmed lifecycle evidence cannot be altered, removed, or reordered without detection because entries are dual-signed and ledger-anchored.

Table. Transaction performance under different user loads

Number of users	Average latency, s	95th percentile latency, s	Minimum latency, s	Maximum latency, s	Throughput, TPS
50	0.5	0.7	0.4	0.9	490
100	0.6	0.9	0.5	1.0	480
250	0.8	1.1	0.6	1.3	450
500	1.0	1.3	0.7	1.5	410
750	1.2	1.5	0.8	1.6	370
1,000	1.3	1.8	0.9	1.8	340

G2 (digest binding): a delivered payload is cryptographically bound to its lifecycle entry by $z = H(x)$, and mismatch detection is externally verifiable after commitment.

G3 (non-repudiation): σ_u proves user initiation and σ_c proves provider acceptance, so uncompromised parties cannot be credibly impersonated or repudiated.

G4 (independent auditability): auditors validate signatures and committed transaction presence independently of provider-local state and detect omissions via ledger-to-log reconciliation.

Experimental evaluation

We evaluated the method on a private Tendermint network with 10 validators and up to 1,000 concurrent users issuing create/modify/delete operations at random intervals. Confirmation latency averaged about 1.3 s (max 1.8 s), while client-side cryptography remained below 3 ms per operation; consensus delay dominated. Table summarizes latency and throughput under varying load. Overall, the measurements show a small, predictable synchronous overhead to obtain immediately auditable lifecycle evidence.

Comparative discussion

Provider-managed audit logs (e.g., AWS CloudTrail, Google Cloud Audit Logs) introduce near-zero critical-path overhead but deliver records with a delay, keeping auditability inside the provider's trust domain [36, 37]. Our method adds a bounded confirmation delay yet provides immediate, provider-independent verifiability once a transaction is confirmed. Azure Confidential Ledger offers an immutable managed ledger within the provider's administrative domain. Under our threat model of a potentially compromised provider, we instead anchor dual-signed lifecycle evidence in a blockchain ledger verifiable outside the provider and bind it to versioned data semantics (status synchronization and version digests). For explicit comparison we use five criteria: confirmation latency, operation-path overhead, cryptographic overhead, delay until independent audit visibility, and attack-detection coverage (substitution, rollback, forgery, truncation). In our prototype, confirmation latency is about 1.3 s (max 1.8 s), cryptographic overhead is less than 3 ms per operation, and evidence becomes independently verifiable immediately after commit. CloudTrail-like systems have near-zero operation-path overhead but longer delay before audit visibility and provider-scoped trust; managed provider ledgers improve immutability but remain within the provider administrative boundary.

Conclusion

We presented a method for registering data lifecycle events in cloud storage under a strong provider-compromise assumption by anchoring dual-signed, version-bound entries in an external immutable ledger. Verification ties delivered versions to confirmed records and enables independent audit; a compromised provider can still delete data but cannot make such actions undetectable once verification is performed. Experiments show feasibility with mean confirmation latency about 1.3 s and negligible cryptographic overhead.

References

1. Hao J., Deng Z., Li J. The evolution of data pricing: From economics to computational intelligence. *Heliyon*, 2023, vol. 9, no. 9, pp. e20274. doi: 10.1016/j.heliyon.2023.e20274
2. Juhasz Z. Quantitative cost comparison of on-premise and cloud infrastructure based EEG data processing. *Cluster Computing*, 2021, vol. 24, no. 2, pp. 625–641. doi: 10.1007/s10586-020-03141-y
3. Wang L., Tian Y., Xiong J. Achieving reliable and anti-collusive outsourcing computation and verification based on blockchain in 5G-enabled IoT. *Digital Communications and Networks*, 2022, vol. 8, no. 5, pp. 644–653. doi: 10.1016/j.dcan.2022.05.012
4. Song M., Sang Y. Secure outsourcing of matrix determinant computation under the malicious cloud. *Sensors*, 2021, vol. 21, no. 20, pp. 6821. doi: 10.3390/s21206821
5. Chen Z., Tian Y., Zhou F., Xiong W., Yang Z., Wang S. A rational and reliable model for outsourcing polynomial two-party computation. *Computers and Electrical Engineering*, 2024, vol. 120, part C, pp. 109829. doi: 10.1016/j.compeleceng.2024.109829

Литература

1. Hao J., Deng Z., Li J. The evolution of data pricing: From economics to computational intelligence // Heliyon. 2023. V. 9. N 9. P. e20274. doi: 10.1016/j.heliyon.2023.e20274
2. Juhasz Z. Quantitative cost comparison of on-premise and cloud infrastructure based EEG data processing // Cluster Computing. 2021. V. 24. N 2. P. 625–641. doi: 10.1007/s10586-020-03141-y
3. Wang L., Tian Y., Xiong J. Achieving reliable and anti-collusive outsourcing computation and verification based on blockchain in 5G-enabled IoT // Digital Communications and Networks. 2022. V. 8. N 5. P. 644–653. doi: 10.1016/j.dcan.2022.05.012
4. Song M., Sang Y. Secure outsourcing of matrix determinant computation under the malicious cloud // Sensors. 2021. V. 21. N 20. P. 6821. doi: 10.3390/s21206821
5. Chen Z., Tian Y., Zhou F., Xiong W., Yang Z., Wang S. A rational and reliable model for outsourcing polynomial two-party computation // Computers and Electrical Engineering. 2024. V. 120. Part C. P. 109829. doi: 10.1016/j.compeleceng.2024.109829

6. Dawood M., Tu S., Xiao C., Alasmary H., Waqas M., Rehman S.U. Cyberattacks and security of cloud computing: a complete guideline. *Symmetry*, 2023, vol. 15, no. 11, pp. 1981. doi: 10.3390/sym15111981
7. Ang'udi J.J. Security challenges in cloud computing: A comprehensive analysis. *World Journal of Advanced Engineering Technology and Sciences*, 2023, vol. 10, no. 2, pp. 155–181. doi: 10.30574/wjaets.2023.10.2.0304
8. Mahida A. Secure data outsourcing techniques for cloud storage. *International Journal of Science and Research (IJSR)*, 2024, vol. 13, no. 4, pp. 181–184. doi: 10.21275/sr24402065432
9. Du J., Dong G., Ning J., Xu Z., Yang R. Identity-based controlled delegated outsourcing data integrity auditing scheme. *Scientific Reports*, 2024, vol. 14, no. 1, pp. 7582. doi: 10.1038/s41598-024-58325-y
10. Makhdoom I., Abolhasan M., Lipman J., Piccardi M., Franklin D. PrivySeC: A secure and privacy-compliant distributed framework for personal data sharing in IoT ecosystems. *Blockchain: Research and Applications*, 2024, vol. 5, no. 4, pp. 100220. doi: 10.1016/j.bcra.2024.100220
11. Alwaheidi M.K., Islam S. Data-driven threat analysis for ensuring security in cloud enabled systems. *Sensors*, 2022, vol. 22, no. 15, pp. 5726. doi: 10.3390/s22155726
12. Dikii D., Arustamov S., Grishentsev A. DOS attacks detection in MQTT networks. *Indonesian Journal of Electrical Engineering and Computer Science*, 2021, vol. 21, no. 1, pp. 601–608. doi: 10.11591/ijeecs.v21.i1.pp601-608
13. Kumari P., Jain A.K. A comprehensive study of DDoS attacks over IoT network and their countermeasures. *Computers & Security*, 2023, vol. 127, pp. 103096. doi: 10.1016/j.cose.2023.103096
14. Pakmehr A., Aßmuth A., Taheri N., Ghaffari A. DDoS attack detection techniques in IoT networks: a survey. *Cluster Computing*, 2024, vol. 27, no. 10, pp. 14637–14668. doi: 10.1007/s10586-024-04662-6
15. Bukhowah R., Aljughaiman A., Rahman M.M.H. Detection of DoS attacks for IoT in information-centric networks using machine learning: Opportunities, challenges, and future research directions. *Electronics*, 2024, vol. 13, no. 6, pp. 1031. doi: 10.3390/electronics13061031
16. Sahu S.K., Mazumdar K. Exploring security threats and solutions Techniques for Internet of Things (IoT): from vulnerabilities to vigilance. *Frontiers in Artificial Intelligence*, 2024, vol. 7, pp. 1397480. doi: 10.3389/frai.2024.1397480
17. Almutairi M., Sheldon F.T. IoT-Cloud Integration Security: A Survey of Challenges, Solutions, and Directions. *Electronics*, 2025, vol. 14, no. 7, pp. 1394. doi: 10.3390/electronics14071394
18. Sekar S., Rani P.S., Dhanamathi A., Raju A.R., Pandey P., Bavaneeswari M. Securing the cloud with AWS Shield for comprehensive protection of cloud infrastructure and services. *Proc. of the 11th International Conference on Communication and Signal Processing (ICCSP)*, 2025, pp. 1825–1830. doi: 10.1109/iccsp64183.2025.11088761
19. Roy A., Banerjee A., Bhardwaj N. A study on Google Cloud Platform (GCP) and its security. *Machine Learning Techniques and Analytics for Cloud Security*, 2021, pp. 313–338. doi: 10.1002/9781119764113.ch15
20. Anderson M.W. Enabling the Write-Once Read-Many (WORM) dashboard and web interface for HPC data storage. *Idaho National Laboratory*, 2023.
21. Lupaiescu S., Cioata P., Turcu C.E., Gherman O., Turcu C.O., Paslaru G. Centralized vs. decentralized: Performance comparison between bigchaindb and amazon QLDB. *Applied Sciences*, 2023, vol. 13, no. 1, pp. 499. doi: 10.3390/app13010499
22. Zhao X., Li M., Feng F., Xia Y. Towards a secure joint cloud with confidential computing. *Proc. of the IEEE International Conference on Joint Cloud Computing (JCC)*, 2022, pp. 79–88. doi: 10.1109/jcc56315.2022.00019
23. Ahmad A., Lee S., Peinado M. Hardlog: Practical tamper-proof system auditing using a novel audit device. *Proc. of the IEEE Symposium on Security and Privacy (SP)*, 2022, pp. 1791–1807. doi: 10.1109/sp46214.2022.9833745
24. Javed H., Abaid Z., Akbar S., Ullah K., Ahmad A., Saeed A., et al. Blockchain-based logging to defeat malicious insiders: The case of remote health monitoring systems. *IEEE Access*, 2024, vol. 12, pp. 12062–12079. doi: 10.1109/access.2023.3346432
25. Tong Q., Yin L., Liu Y., Xu J. Append-only authenticated Data Sets based on RSA accumulators for transparent log system. *Computer Standards & Interfaces*, 2025, vol. 93, pp. 103978. doi: 10.1016/j.csi.2025.103978

6. Dawood M., Tu S., Xiao C., Alasmary H., Waqas M., Rehman S.U. Cyberattacks and security of cloud computing: a complete guideline // Symmetry. 2023. V. 15. N 11. P. 1981. doi: 10.3390/sym15111981
7. Ang'udi J.J. Security challenges in cloud computing: A comprehensive analysis // World Journal of Advanced Engineering Technology and Sciences. 2023. V. 10. N 2. P. 155–181. doi: 10.30574/wjaets.2023.10.2.0304
8. Mahida A. Secure data outsourcing techniques for cloud storage // International Journal of Science and Research (IJSR). 2024. V. 13. N 4. P. 181–184. doi: 10.21275/sr24402065432
9. Du J., Dong G., Ning J., Xu Z., Yang R. Identity-based controlled delegated outsourcing data integrity auditing scheme // Scientific Reports. 2024. V. 14. N 1. P. 7582. doi: 10.1038/s41598-024-58325-y
10. Makhdoom I., Abolhasan M., Lipman J., Piccardi M., Franklin D. PrivySeC: A secure and privacy-compliant distributed framework for personal data sharing in IoT ecosystems // Blockchain: Research and Applications. 2024. V. 5. N 4. P. 100220. doi: 10.1016/j.bcra.2024.100220
11. Alwaheidi M.K., Islam S. Data-driven threat analysis for ensuring security in cloud enabled systems // Sensors. 2022. V. 22. N 15. P. 5726. doi: 10.3390/s22155726
12. Dikii D., Arustamov S., Grishentsev A. DOS attacks detection in MQTT networks // Indonesian Journal of Electrical Engineering and Computer Science. 2021. V. 21. N 1. P. 601–608. doi: 10.11591/ijeecs.v21.i1.pp601-608
13. Kumari P., Jain A.K. A comprehensive study of DDoS attacks over IoT network and their countermeasures // Computers & Security. 2023. V. 127. P. 103096. doi: 10.1016/j.cose.2023.103096
14. Pakmehr A., Aßmuth A., Taheri N., Ghaffari A. DDoS attack detection techniques in IoT networks: a survey // Cluster Computing. 2024. V. 27. N 10. P. 14637–14668. doi: 10.1007/s10586-024-04662-6
15. Bukhowah R., Aljughaiman A., Rahman M.M.H. Detection of DoS attacks for IoT in information-centric networks using machine learning: Opportunities, challenges, and future research directions // Electronics. 2024. V. 13. N 6. P. 1031. doi: 10.3390/electronics13061031
16. Sahu S.K., Mazumdar K. Exploring security threats and solutions Techniques for Internet of Things (IoT): from vulnerabilities to vigilance // Frontiers in Artificial Intelligence. 2024. V. 7. P. 1397480. doi: 10.3389/frai.2024.1397480
17. Almutairi M., Sheldon F.T. IoT-Cloud Integration Security: A Survey of Challenges, Solutions, and Directions // Electronics. 2025. V. 14. N 7. P. 1394. doi: 10.3390/electronics14071394
18. Sekar S., Rani P.S., Dhanamathi A., Raju A.R., Pandey P., Bavaneeswari M. Securing the cloud with AWS Shield for comprehensive protection of cloud infrastructure and services // Proc. of the 11th International Conference on Communication and Signal Processing (ICCSP). 2025. P. 1825–1830. doi: 10.1109/iccsp64183.2025.11088761
19. Roy A., Banerjee A., Bhardwaj N. A study on Google Cloud Platform (GCP) and its security // Machine Learning Techniques and Analytics for Cloud Security. 2021. P. 313–338. doi: 10.1002/9781119764113.ch15
20. Anderson M.W. Enabling the Write-Once Read-Many (WORM) dashboard and web interface for HPC data storage // Idaho National Laboratory. 2023.
21. Lupaiescu S., Cioata P., Turcu C.E., Gherman O., Turcu C.O., Paslaru G. Centralized vs. decentralized: Performance comparison between bigchaindb and amazon QLDB // Applied Sciences. 2023. V. 13. N 1. P. 499. doi: 10.3390/app13010499
22. Zhao X., Li M., Feng F., Xia Y. Towards a secure joint cloud with confidential computing // Proc. of the IEEE International Conference on Joint Cloud Computing (JCC). 2022. P. 79–88. doi: 10.1109/jcc56315.2022.00019
23. Ahmad A., Lee S., Peinado M. Hardlog: Practical tamper-proof system auditing using a novel audit device // Proc. of the IEEE Symposium on Security and Privacy (SP). 2022. P. 1791–1807. doi: 10.1109/sp46214.2022.9833745
24. Javed H., Abaid Z., Akbar S., Ullah K., Ahmad A., Saeed A., et al. Blockchain-based logging to defeat malicious insiders: The case of remote health monitoring systems // IEEE Access. 2024. V. 12. P. 12062–12079. doi: 10.1109/access.2023.3346432
25. Tong Q., Yin L., Liu Y., Xu J. Append-only authenticated Data Sets based on RSA accumulators for transparent log system // Computer Standards & Interfaces. 2025. V. 93. P. 103978. doi: 10.1016/j.csi.2025.103978

26. Kim T.N., Ngoc D.H., Moldovyan N.A. Constructing new representative collective signature using the GOST R 34.10-2012. *Journal of Communications*, 2022, vol. 17, no. 6, pp. 478–485. doi: 10.12720/jcm.17.6.478-485
27. Mikhailovich P.P., Victorovich C.A., Nikolayevich Y.N., Yuryevich Z.M. Implementation of GOST 34.12-2018 Encryption for LoRaWAN End-devices. *Proc. of the Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, 2021, pp. 0451–0454. doi: 10.1109/USBEREIT51232.2021.9455037
28. Kazymyrov O., Kazymyrova V. Algebraic aspects of the Russian hash standard GOST R 34.11-2012. *Cryptology ePrint Archive*, 2013.
29. Komarova A., Menshchikov A., Klyaus T., Korobeynikov A., Gatchin Y., Tishukova N. Analysis and comparison of electronic digital signature state standards GOST R 34.10-1994, GOST R 34.10-2001 and GOST R 34.10-2012. *Proc. of the 10th International Conference on Security of Information and Networks*, 2017, pp. 262–267. doi: 10.1145/3136825.3136894
30. Kim S., Kim D. Data-tracking in blockchain utilizing hash chain: a study of structured and adaptive process. *Symmetry*, 2024, vol. 16, no. 1, pp. 62. doi: 10.3390/sym16010062
31. Yin M., Malkhi D., Reiter M.K., Gueta G.G., Abraham I. HotStuff: BFT consensus with linearity and responsiveness. *Proc. of the 2019 ACM Symposium on Principles of Distributed Computing*, 2019, pp. 347–356. doi: 10.1145/3293611.3331591
32. Wang R., Yuan M., Wang Z., Li Y. Improved fast-response consensus algorithm based on HotStuff. *Sensors*, 2024, vol. 24, no. 16, pp. 5417. doi: 10.3390/s24165417
33. Kang D., Gupta S., Malkhi D., Sadoghi M. Hotstuff-1: Linear consensus with one-phase speculation. *Proc. of the ACM on Management of Data*, 2025, vol. 3, no. 3, pp. 1–29. doi: 10.1145/3725308
34. Amoussou-Guenou Y., Beltrando L., Herlihy M., Potop-Butucaru M. Byzantine Reliable Broadcast and Tendermint consensus with trusted components. *Cryptology ePrint Archive*, 2025.
35. Cason D., Fynn E., Milosevic N., Milosevic Z., Buchman E., Pedone F. The design, architecture and performance of the tendermint blockchain network. *Proc. of the 40th International Symposium on Reliable Distributed Systems (SRDS)*, 2021, pp. 23–33. doi: 10.1109/srds53918.2021.00012
36. Makani S.T. Efficient resource utilization with auto tagging using amazon's cloud trail services. *International Journal of Computer Sciences and Engineering*, 2023, vol. 11, no. 9, pp. 11–16. doi: 10.26438/ijcse/v11i9.1116
37. Kesseli T. *Security in Cloud Environments: Using an Open-Source Tool to Secure the Cloud Platform*. Master's Thesis. Oulu University of Applied Sciences, 2025, 71 p.

26. Kim T.N., Ngoc D.H., Moldovyan N.A. Constructing new representative collective signature using the GOST R 34.10-2012 // Journal of Communications. 2022. V. 17. N 6. P. 478–485. doi: 10.12720/jcm.17.6.478-485
27. Mikhailovich P.P., Victorovich C.A., Nikolayevich Y.N., Yuryevich Z.M. Implementation of GOST 34.12-2018 Encryption for LoRaWAN End-devices // Proc. of the Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT). 2021. P. 0451–0454. doi: 10.1109/USBEREIT51232.2021.9455037
28. Kazymyrov O., Kazymyrova V. Algebraic aspects of the Russian hash standard GOST R 34.11-2012 // Cryptology ePrint Archive. 2013.
29. Komarova A., Menshchikov A., Klyaus T., Korobeynikov A., Gatchin Y., Tishukova N. Analysis and comparison of electronic digital signature state standards GOST R 34.10-1994, GOST R 34.10-2001 and GOST R 34.10-2012 // Proc. of the 10th International Conference on Security of Information and Networks. 2017. P. 262–267. doi: 10.1145/3136825.3136894
30. Kim S., Kim D. Data-tracking in blockchain utilizing hash chain: a study of structured and adaptive process // Symmetry. 2024. V. 16. N 1. P. 62. doi: 10.3390/sym16010062
31. Yin M., Malkhi D., Reiter M.K., Gueta G.G., Abraham I. HotStuff: BFT consensus with linearity and responsiveness // Proc. of the 2019 ACM Symposium on Principles of Distributed Computing. 2019. P. 347–356. doi: 10.1145/3293611.3331591
32. Wang R., Yuan M., Wang Z., Li Y. Improved fast-response consensus algorithm based on HotStuff // Sensors. 2024. V. 24. N 16. P. 5417. doi: 10.3390/s24165417
33. Kang D., Gupta S., Malkhi D., Sadoghi M. Hotstuff-1: Linear consensus with one-phase speculation // Proc. of the ACM on Management of Data. 2025. V. 3. N 3. P. 1–29. doi: 10.1145/3725308
34. Amoussou-Guenou Y., Beltrando L., Herlihy M., Potop-Butucaru M. Byzantine Reliable Broadcast and Tendermint consensus with trusted components // Cryptology ePrint Archive. 2025.
35. Cason D., Fynn E., Milosevic N., Milosevic Z., Buchman E., Pedone F. The design, architecture and performance of the tendermint blockchain network // Proc. of the 40th International Symposium on Reliable Distributed Systems (SRDS). 2021. P. 23–33. doi: 10.1109/srds53918.2021.00012
36. Makani S.T. Efficient resource utilization with auto tagging using amazon's cloud trail services // International Journal of Computer Sciences and Engineering. 2023. V. 11. N 9. P. 11–16. doi: 10.26438/ijcse/v11i9.1116
37. Kesseli T. Security in Cloud Environments: Using an Open-Source Tool to Secure the Cloud Platform. Master's Thesis. Oulu University of Applied Sciences. 2025. 71 p.

Authors

Vladimir V. Eremuk — PhD Student, ITMO University, Saint Petersburg, 197101, Russian Federation, https://orcid.org/0000-0003-2337-0015, polar.vl@yandex.ru

Alexandr V. Kasyanov — PhD Student, Saint Petersburg Electrotechnical University "LETI", Saint Petersburg, 197022, Russian Federation, https://orcid.org/0009-0000-3319-8357, kasjanov@inbox.ru

Liubov A. Primak — PhD Student, ITMO University, Saint Petersburg, 197101, Russian Federation, https://orcid.org/0009-0002-1462-3156, laprimak@itmo.ru

Victor A. Romashov — PhD Student, ITMO University, Saint Petersburg, 197101, Russian Federation, https://orcid.org/0000-0002-1999-5381, whiviktor@gmail.com

Авторы

Еремук Владимир Вадимович — аспирант, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, https://orcid.org/0000-0003-2337-0015, polar.vl@yandex.ru

Касьянов Александр Владимирович — аспирант, Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина), Санкт-Петербург, 197022, Российская Федерация, https://orcid.org/0009-0000-3319-8357, kasjanov@inbox.ru

Примак Любовь Андреевна — аспирант, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, https://orcid.org/0009-0002-1462-3156, laprimak@itmo.ru

Ромашов Виктор Андреевич — аспирант, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, https://orcid.org/0000-0002-1999-5381, whiviktor@gmail.com

Received 19.09.2025
Approved after reviewing 28.03.2026
Accepted 21.05.2026

Статья поступила в редакцию 19.09.2025
Одобрена после рецензирования 28.03.2026
Принята к печати 21.05.2026