Ilmiy tadqiqotlar va ularning yechimlari jurnali
Aoka guvohnoma № 323759 ISSN: 3060-4818

METHODS FOR ENSURING THE INTEGRITY AND AUTHENTICITY OF MILITARY ELECTRONIC DOCUMENTS

Giyosjon Jumaev Abdivaxobovich
Artificial intellegance and digital technologies, Tashkent International University of Financial Management and Technologies, Tashkent, Uzbekistan
giyosjonjumaev@gmail.com
Turapov Sherzod Nurmatovich
Head of Department of Act and Military Communication Institute, Tashkent, Uzbekistan
https://doi.org/10.5281/zenodo.23324540

Abstract

Background. Orders, intelligence reports and logistics records in military information systems must remain unaltered, be attributable to a known originator, and stay verifiable for years, often over degraded or disconnected networks. Checksums, stored hashes, shared-key message authentication codes and digital signatures are all used for this purpose, but they protect against different threats. **Objective.** To establish experimentally which threats each mechanism detects, to propose a layered scheme that covers all of them, and to measure its computational cost. **Methods.** Five schemes (CRC-32, stored SHA-256, HMAC-SHA256 with a group key, Ed25519 signatures, and Ed25519 signatures with a trusted timestamp and a hash-chained ledger) were implemented and tested with 2,000 trials per attack against six attack classes: random modification, modification with tag recomputation, forged authorship by an insider, rollback to an older authentic version, backdating by an author, and deletion. Hashing, signing and ledger verification costs were benchmarked. **Results.** All schemes detected random modification. CRC-32 and stored SHA-256 detected nothing else; HMAC-SHA256 additionally detected recomputation by an outsider; Ed25519 signatures additionally detected forged authorship; only the layered scheme detected rollback, backdating and deletion (6 of 6 attack classes, with no false alarms on genuine records). Hashing dominated the cost: 130.8 ms for a 50 MB document, while one signature verification took about 0.20 ms. A Merkle inclusion proof for a 262,144-entry ledger verified in about 17 µs, against about 262 ms for a linear chain walk. **Conclusions.** Integrity alone is not authenticity, and authenticity alone is not freshness or completeness. A signature, a trusted timestamp and an append-only ledger together close the gaps at a modest cost. The attack results follow from the design of each mechanism and confirm the implementation, not field performance; timings are from one machine.

Keywords: document integrity; authenticity; digital signature; trusted timestamp; tamper-evident log; Merkle tree; military information systems; post-quantum transition

1. Introduction

A military order that has been altered by one word, a report that appears to come from the wrong unit, or an older instruction replayed after it was superseded can each cause a failure as serious as a leak.

Confidentiality controls have received most of the attention in classified systems, but integrity and authenticity are the properties on which command decisions rest. Integrity models such as Biba's [16] and the commercial integrity policy of Clark and Wilson [15] have long argued that data should be changed only through controlled, accountable transformations; the cryptographic mechanisms that implement this for documents are the subject of this article.

Four families of mechanism are in common use. Checksums and cyclic redundancy codes detect accidental corruption. Cryptographic hash functions such as SHA-256 [10] give a fixed-length fingerprint, and their security has been tested in practice: collisions were found for MD5 [19] and later for SHA-1 [20], which is why those functions have been withdrawn from integrity use. Message authentication codes, notably HMAC [4], add a secret key so that only key holders can produce a valid tag. Public-key digital signatures [1,2,11] make the signing key private to one party, so that anyone can verify the origin and the originator cannot easily deny it. Each has a place, yet in practice they are often described interchangeably, and it is rarely stated what attacks a given choice leaves open.

Three further properties matter for military records. **Freshness and order:** a valid, authentic, but superseded version must not be accepted as current. **Time:** the claimed creation time must be trustworthy, which requires a third-party timestamp [5,6]. **Completeness:** the removal of a record must be detectable, which requires an independent, append-only register [17,18]. Merkle trees [3] make such registers efficient to verify at scale. Added to these are operational demands specific to defence: keys must be protected in hardware and managed under strict policy [14], certificates must be validated and revoked [7], verification must often work offline, and signature schemes must be replaceable as quantum-resistant standards [12,13] take effect.

This article asks: **which of six realistic attacks on stored military documents does each integrity or authenticity mechanism detect, what does a layered scheme add, and what does it cost?** Its contributions are (i) a layered scheme combining signatures, trusted timestamps and a tamper-evident ledger, (ii) an executable attack-and-detection experiment, and (iii) measurements of hashing, signing and ledger-verification costs that inform deployment choices.

2. Methods

2.1 Requirements and threat model

The protected asset is an electronic document in a versioned archive. The adversary can read and write the storage layer, including document content and metadata, but cannot obtain the private key of another user or of the timestamp authority and cannot rewrite a trusted reference copy of the ledger head. Some adversaries are insiders: they may hold a legitimate key of their own, and for symmetric schemes the shared group key. Table 1 lists the six attack classes that were implemented.

Table 1. Attack classes used in the experiment.

ID	Attack	What the adversary does
A1	Random modification	Flips one random bit of the content; leaves tags and metadata unchanged
A2	Modification with tag recomputation	Replaces the content, then recomputes any integrity tag that needs no secret (CRC or hash); cannot compute keyed tags

ID	Attack	What the adversary does
A3	Insider forgery of authorship	Creates a new document claiming that another user is the author; holds the group key but not the other user's private key
A4	Rollback	Substitutes an older, fully authentic version for the current one
A5	Backdating by an author	A legitimate author signs a new document that claims an earlier creation time
A6	Deletion	Removes a record from storage; detection requires an independent register of what should exist

2.2 Mechanisms compared

Table 2 summarises the properties of the five schemes studied. They form a ladder in which each scheme adds one capability to the previous one.

Table 2. Mechanisms compared and their theoretical properties.

Scheme	Accidental error	Deliberate change	Origin authentication	Non-repudiation	Freshness, time, completeness
CRC-32	Yes	No	No	No	No
Stored SHA-256	Yes	No (tag can be replaced)	No	No	No
HMAC-SHA256, group key	Yes	Yes (outsiders)	Group only	No	No
Ed25519 signature	Yes	Yes	Yes (per user)	Yes	No
Signature + timestamp + ledger	Yes	Yes	Yes	Yes	Yes

2.3 The layered scheme

The scheme (Figure 1) has four creation steps and three verification checks. The author's device hashes the content together with its metadata, signs the digest with a certified private key held in a smart card or hardware security module, and sends the signature to a timestamp authority (TSA), which returns a signed token that binds the signature to a time [5,6]. The record is then appended to an append-only ledger and stored.

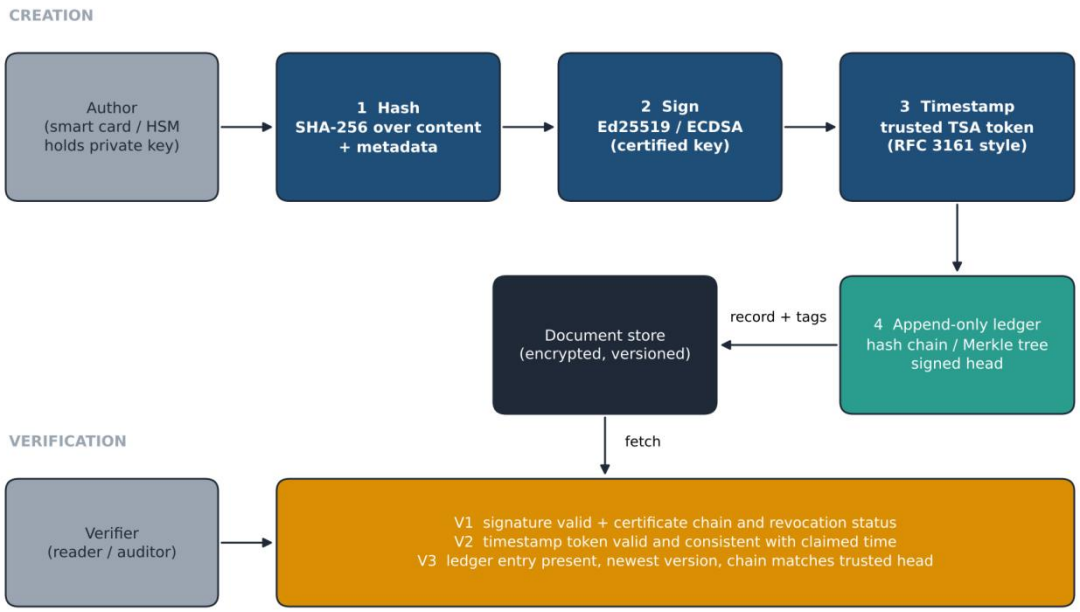

Figure 1. Creation and verification flow of the layered scheme.

Formally, for document content c, metadata m (identifier, version v, originator, claimed time t), originator signing key sk and timestamp-authority key sk(TSA):

$\sigma = \mathrm{Sign}[sk](\ \mathrm{SHA\text{-}256}(c) \parallel m\ ) \qquad \tau = \mathrm{Sign}[sk(TSA)](\ \mathrm{SHA\text{-}256}(\sigma) \parallel t'\ )$

$e(i) = (\ i,\ doc,\ v,\ \mathrm{SHA\text{-}256}(m \parallel \sigma),\ t',\ H(e(i-1))\ )$

Each ledger entry e(i) is chained to its predecessor by hash, and the head of the chain is signed by the custodian and distributed as a trusted anchor. A verifier accepts a record only if (V1) σ verifies under the public key of the claimed originator and the originator's certificate chain and revocation status are valid [7]; (V2) τ verifies and t′ is within a tolerance (here 60 s) of the claimed time; and (V3) the ledger entry for that document and version exists, matches the record, is the newest version, and the chain agrees with the trusted head.

2.4 Experimental design

The experiment was implemented in Python using the cryptography library (OpenSSL backend) and the standard library. An archive of 40 documents, each with three versions of 4 KB random content, was created for each scheme with 20 originators, each with its own Ed25519 key, and one group key for the HMAC scheme. Each attack was applied 2,000 times to randomly chosen documents and the verifier's decision recorded; the detection rate is the share of attacks flagged. A control run verified all genuine current records, and a separate test altered a random ledger entry 500 times. Because the cryptographic primitives behave deterministically under these attacks, rates of 0% and 100% are expected outcomes; the experiment confirms that each implementation behaves as the theory predicts, which is its purpose.

Cost was measured separately. Hashing and keyed hashing were timed on random documents of 0.1, 1, 10 and 50 MB (median of 15 runs, 5 for 50 MB). Signature algorithms (RSA-3072 with PSS, ECDSA P-256 and Ed25519) were timed over 32-byte digests (median of 300 runs), which is how signing is done for large documents. Ledger verification was timed for 1,024 to 262,144 entries,

comparing a linear walk of the hash chain with a Merkle inclusion proof. All timings come from a single cloud virtual machine and should be read as relative, not absolute, values.

3. Results

3.1 Detection of attacks

Table 3 and Figure 2 show the outcome. All five schemes detected random modification (A1). CRC-32 and stored SHA-256 missed every deliberate attack, because an adversary who can change the content can also replace the tag. HMAC-SHA256 detected modification by an outsider (A2) but not forgery by an insider holding the group key (A3), since the same key both creates and checks tags. Ed25519 signatures detected A3, because an insider cannot sign as another user. None of the first four schemes detected rollback (A4), because the older record is entirely authentic, nor backdating by an author (A5), since the signature is valid, nor deletion (A6), since nothing remains to be checked. The layered scheme detected all six. No genuine record was rejected in the control run, and all 500 ledger alterations were detected.

Table 3. Detection rate by scheme and attack (2,000 trials per cell).

Scheme	A1	A2	A3	A4	A5	A6
CRC-32	100%	0%	0%	0%	0%	0%
Stored SHA-256	100%	0%	0%	0%	0%	0%
HMAC-SHA256	100%	100%	0%	0%	0%	0%
Ed25519	100%	100%	100%	0%	0%	0%
Signature + TS + ledger	100%	100%	100%	100%	100%	100%

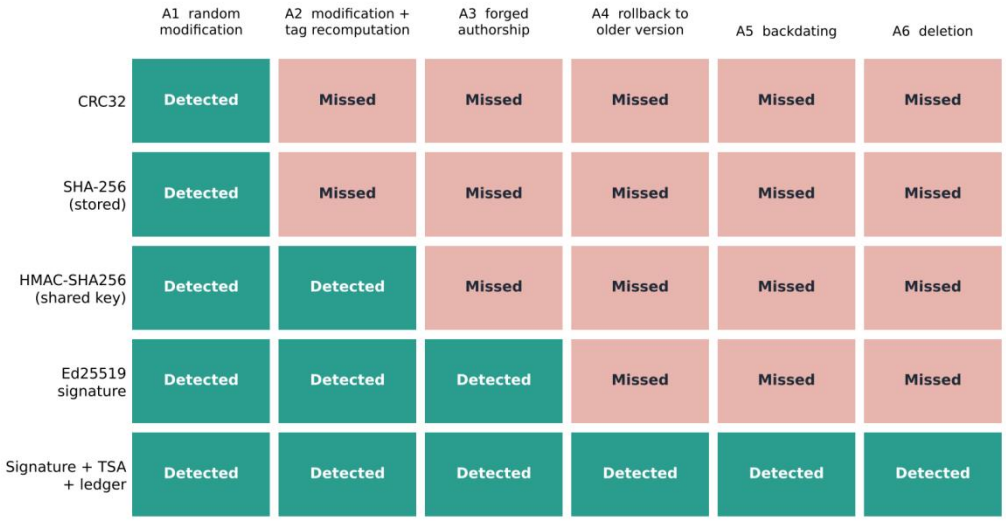

Figure 2. Which attacks each scheme detects.

3.2 Cost of hashing

Hashing time grew linearly with document size (Table 4, Figure 3). SHA-256 processed 50 MB in about 130.8 ms, roughly 380 MB/s on this machine, SHA-512 was faster than SHA-256 for large inputs, SHA3-256 was somewhat slower, and HMAC-SHA256 cost slightly more than plain SHA-256. CRC-

32 was about eight times faster than SHA-256 but, as Table 3 shows, offers no protection against deliberate change.

Table 4. Median time to process a document (ms).

Function	0.1 MB	1 MB	10 MB	50 MB
CRC-32	0.03	0.30	3.27	16.4
SHA-256	0.23	2.41	27.0	130.8
SHA-512	0.15	1.62	16.6	85.5
SHA3-256	0.29	2.91	30.1	174.9
HMAC-SHA256	0.24	2.42	26.5	144.7

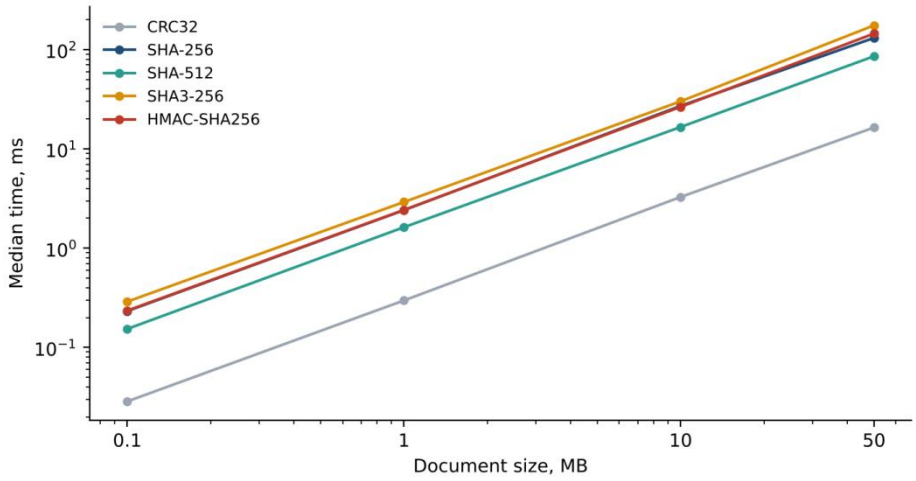

Figure 3. Hashing time against document size (log–log axes; the SHA-256 and HMAC-SHA256 lines almost coincide).

3.3 Cost of signatures

Because large documents are hashed first, signing and verification operate on a short digest and their cost does not depend on document size (Table 5). Ed25519 and ECDSA P-256 signed in well under 0.1 ms. RSA-3072 signing was about 30 to 45 times slower but its verification was the fastest of the three. Ed25519 gave the smallest keys and signatures, which matters for storage in a ledger and for transmission over narrow tactical links.

Table 5. Signature algorithms over a 32-byte digest (median of 300 runs).

Algorithm	Sign, ms	Verify, ms	Signature, bytes	Public key, bytes
RSA-3072 (PSS)	2.053	0.119	384	387
ECDSA P-256	0.044	0.146	71	64
Ed25519	0.063	0.199	64	32

3.4 Scaling of ledger verification

Verifying a ledger by walking the whole hash chain grows linearly with the number of entries and reached about 262 ms at 262,144 entries (Table 6, Figure 4). A Merkle inclusion proof [3] needs only a

logarithmic number of hashes: about 17 µs and 576 bytes of proof at the same size, an improvement of about four orders of magnitude, and it can be checked by a verifier that holds only the signed root.

Table 6. Verifying one document against the ledger.

Ledger entries	Linear chain walk, ms	Merkle proof, µs	Merkle proof size, bytes
1,024	1.8	15.5	320
4,096	6.6	18.6	384
16,384	24.8	20.8	448
65,536	82.6	22.7	512
262,144	262.0	17.1	576

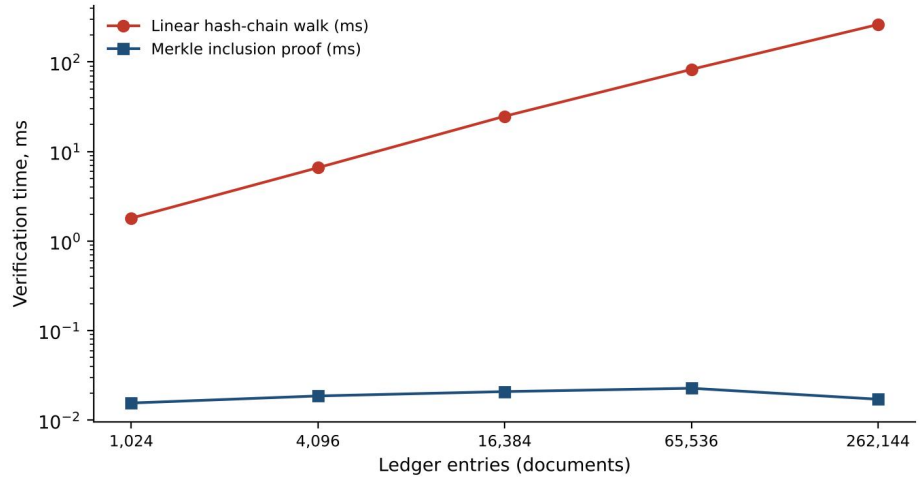

Figure 4. Ledger verification cost: linear chain walk versus Merkle inclusion proof.

4. Discussion

4.1 Principal findings

Three conclusions follow. First, integrity, authenticity and freshness are different properties, and each simple mechanism supplies only part of them. A stored hash or CRC is an error check, not a security control, because whoever can alter a document can alter the tag. Second, a shared-key MAC [4] is a poor fit for multi-user military archives: any holder of the key can forge any author's document, so it can show that a document came from the group but not from whom. Signatures [1,2,11] resolve this, in line with the accountability principle behind controlled-transformation integrity policies [15]. Third, even a correct signature leaves three gaps that real adversaries use: replay of a superseded version, backdating by an authorised author, and quiet deletion. A trusted timestamp [5,6] closes the time gap, and an append-only ledger with version counters closes the order and completeness gaps, as the secure-log literature anticipated [17,18].

4.2 Cost and deployment

The measurements show that the cryptographic cost of the layered scheme is small compared with moving and hashing the document. For a 10 MB document, hashing takes about 27 ms while signature verification is about 0.2 ms and a Merkle proof is a few hundredths of a millisecond, so cost is governed

by document size, not by the added layers. A TSA round trip adds network latency but only one small request per document. Linear verification of a ledger does not scale, so large archives should use Merkle structures with a signed root, and verifiers should keep a trusted copy of that root. Elliptic-curve signatures such as Ed25519 [8,9] and ECDSA [11] are preferable to RSA where signing frequency or bandwidth is constrained, whereas RSA verification remains very fast. Table 7 gives a practical selection guide.

Table 7. Suggested controls by document class.

Document class	Minimum controls	Rationale
Routine, low-impact	SHA-256 plus individual digital signature	Authenticity and non-repudiation at low cost
Operational orders, intelligence reports	Signature, trusted timestamp, version counter	Prevents replay of superseded orders and false time claims
Records of legal or archival value	Full scheme with Merkle ledger and signed root	Completeness, auditability and long-term verifiability
Tactical, disconnected use	Signature plus cached revocation data and last trusted root	Offline verification with a bounded staleness window

4.3 Operational and long-term considerations

The strength of the scheme rests on key management. Private keys should be generated and kept in tamper-resistant hardware, with defined lifetimes, rollover and revocation procedures [14,7]; the experiment assumes that these keys are not compromised. Archives that must remain verifiable for decades face two further problems. Hash functions can weaken, as MD5 and SHA-1 did [19,20], so records need a renewal process that re-timestamps them with a stronger function before the older one is considered unsafe. Quantum computers would break the RSA and elliptic-curve signatures used today, and national-security guidance already sets a transition timetable toward quantum-resistant algorithms [13] such as the lattice-based signature standard [12]. Systems should therefore be designed with algorithm agility, for example by recording the algorithm identifier in each signed record and by allowing dual signatures during the transition. Larger post-quantum signatures will also increase ledger and bandwidth costs, which the present benchmarks do not measure.

4.4 Limitations

The detection results are deterministic consequences of how each mechanism works, so they confirm a correct implementation but are not statistical evidence about real adversaries. The six attack classes were chosen by the author and are not exhaustive; collusion between a custodian and an attacker, compromise of the TSA or of an author's key, side-channel attacks and denial-of-service attacks were not modelled. The timestamp authority and ledger custodian are treated as trusted, although in practice they should be distributed or independently audited. Timings were taken on one virtual machine with one software library, without hardware security modules, and without network latency, and post-quantum algorithms were not benchmarked. Offline and degraded-network operation was discussed but not tested. Field trials on operational networks and red-team evaluation are the necessary next steps.

5. Conclusion

Checksums and stored hashes detect only accidental damage, and shared-key authentication codes cannot attribute a document to an individual. Digital signatures provide attribution but still accept rolled-back, backdated and silently deleted records. Adding a trusted timestamp and an append-only ledger with version counters detected all six implemented attacks without false alarms, and the measured overhead is small compared with the cost of hashing the document. Military organisations should adopt the layered scheme for operationally significant documents, use Merkle structures for large archives, protect keys in hardware, and build in algorithm agility for the post-quantum transition.

Declarations

Data and code availability. The experiment was implemented in Python with a fixed random seed, so the attack matrix is reproducible from the stated parameters; timings will vary with hardware. **Conflicts of interest.** None declared. **Funding.** None declared. [Edit these statements as required by the target journal.]

References

1. Diffie W, Hellman ME. New directions in cryptography. IEEE Transactions on Information Theory. 1976;22(6):644–654.
2. Rivest RL, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM. 1978;21(2):120–126.
3. Merkle RC. A certified digital signature. In: Advances in Cryptology – CRYPTO '89. Lecture Notes in Computer Science, vol. 435. Berlin: Springer; 1990. p. 218–238.
4. Bellare M, Canetti R, Krawczyk H. Keying hash functions for message authentication. In: Advances in Cryptology – CRYPTO '96. Lecture Notes in Computer Science, vol. 1109. Berlin: Springer; 1996. p. 1–15.
5. Haber S, Stornetta WS. How to time-stamp a digital document. Journal of Cryptology. 1991;3(2):99–111.
6. Adams C, Cain P, Pinkas D, Zuccherato R. Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). RFC 3161. Internet Engineering Task Force; 2001.
7. Cooper D, Santesson S, Farrell S, Boeyen S, Housley R, Polk W. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280. Internet Engineering Task Force; 2008.
8. Josefsson S, Liusvaara I. Edwards-Curve Digital Signature Algorithm (EdDSA). RFC 8032. Internet Engineering Task Force; 2017.
9. Bernstein DJ, Duif N, Lange T, Schwabe P, Yang B-Y. High-speed high-security signatures. Journal of Cryptographic Engineering. 2012;2(2):77–89.
10. National Institute of Standards and Technology. Secure Hash Standard (SHS). FIPS PUB 180-4. Gaithersburg (MD): NIST; 2015.
11. National Institute of Standards and Technology. Digital Signature Standard (DSS). FIPS PUB 186-5. Gaithersburg (MD): NIST; 2023.
12. National Institute of Standards and Technology. Module-Lattice-Based Digital Signature Standard. FIPS PUB 204. Gaithersburg (MD): NIST; 2024.

13. National Security Agency. Commercial National Security Algorithm Suite 2.0 (CNSA 2.0). Cybersecurity advisory. Fort Meade (MD): NSA; 2022.
14. Barker E. Recommendation for key management: Part 1 – General. NIST Special Publication 800-57 Part 1, Revision 5. Gaithersburg (MD): NIST; 2020.
15. Clark DD, Wilson DR. A comparison of commercial and military computer security policies. In: Proceedings of the IEEE Symposium on Security and Privacy; 1987. p. 184–194.
16. Biba KJ. Integrity considerations for secure computer systems. Technical Report MTR-3153. Bedford (MA): The MITRE Corporation; 1977.
17. Schneier B, Kelsey J. Secure audit logs to support computer forensics. ACM Transactions on Information and System Security. 1999;2(2):159–176.
18. Crosby SA, Wallach DS. Efficient data structures for tamper-evident logging. In: Proceedings of the 18th USENIX Security Symposium; 2009. p. 317–332.
19. Wang X, Yu H. How to break MD5 and other hash functions. In: Advances in Cryptology – EUROCRYPT 2005. Lecture Notes in Computer Science, vol. 3494. Berlin: Springer; 2005. p. 19–35.
20. Stevens M, Bursztein E, Karpman P, Albertini A, Markov Y. The first collision for full SHA-1. In: Advances in Cryptology – CRYPTO 2017. Lecture Notes in Computer Science, vol. 10401. Cham: Springer; 2017. p. 570–596.