A04:2025 — Cryptographic Failures

January 6, 2026

Document Summary

This document consolidates the provided content for *A04:2025 — Cryptographic Failures* into a structured, print-ready reference, including background context, scoring metrics, cryptographic failure indicators (transport, storage, key management, PRNG, and protocol misuse), prevention guidance (TLS, encryption at rest, hashing, authenticated encryption, and post-quantum readiness), example attack scenarios, references, and the mapped CWE list.

Contents

1 Background

Moving down two positions to #4, this weakness focuses on failures related to lack of cryptography, insufficiently strong cryptography, leaking of cryptographic keys, and related errors. Three of the most common Common Weakness Enumerations (CWEs) in this risk involved use of weak pseudo-randomness and/or risky cryptographic choices, including:

- CWE-327: Use of a Broken or Risky Cryptographic Algorithm
- CWE-331: Insufficient Entropy
- CWE-1241: Use of Predictable Algorithm in Random Number Generator
- CWE-338: Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG)

2 Score Table

Metric	Value
CWEs Mapped	32
Max Incidence Rate	13.77%
Avg Incidence Rate	3.80%
Max Coverage	100.00%
Avg Coverage	47.74%
Avg Weighted Exploit	7.23
Avg Weighted Impact	3.90
Total Occurrences	1,665,348
Total CVEs	2,185

Table 1: Provided scoring summary for Cryptographic Failures.

3 Description

3.1 Transport Layer Encryption Expectations

Generally speaking, all data in transit should be encrypted at the transport layer (OSI Layer 4). Previous hurdles such as CPU performance and private key/certificate management are now addressed by:

- CPUs offering encryption acceleration instructions (e.g., AES support), and
- simplified private key and certificate management via services such as `https://letsencrypt.org`, with major cloud vendors providing integrated certificate management services.

3.2 Encryption at Rest and Additional Application-Layer Protection

Beyond securing the transport layer, it is important to determine:

- what data requires encryption **at rest**, and

- what data needs extra encryption **in transit at the application layer** (OSI Layer 7).

For example, passwords, credit card numbers, health records, personal information, and business secrets require additional protection, especially when subject to privacy laws (e.g., GDPR) or industry regulations (e.g., PCI DSS).

3.3 Common Cryptographic Failure Questions

For sensitive data, assess the following:

1. Are old or weak cryptographic algorithms or protocols used either by default or in older code?
2. Are default crypto keys in use, are weak keys generated, are keys re-used, or is proper key management and rotation missing?
3. Are crypto keys checked into source code repositories?
4. Is encryption not enforced (e.g., missing HTTP headers / browser security directives)?
5. Is the received server certificate and trust chain properly validated?
6. Are initialization vectors (IVs) ignored, reused, or generated insecurely? Is an insecure mode of operation (e.g., ECB) in use? Is encryption used when authenticated encryption is more appropriate?
7. Are passwords being used as cryptographic keys without a password-based key derivation function?
8. Is randomness used that was not designed to meet cryptographic requirements? Even if a strong API is chosen, has it been seeded predictably or with insufficient entropy?
9. Are deprecated hash functions such as MD5 or SHA1 in use, or are non-cryptographic hash functions used when cryptographic hashing is required?
10. Are cryptographic error messages or side-channel outputs exploitable (e.g., padding oracle attacks)?
11. Can the cryptographic algorithm be downgraded or bypassed?

See references ASVS: Cryptography (V11), Secure Communication (V12) and Data Protection (V14).

4 How to Prevent

Do the following, at a minimum, and consult the references:

1. **Classify data:** Classify and label data processed, stored, or transmitted by an application. Identify which data is sensitive according to privacy laws, regulatory requirements, or business needs.
2. **Protect keys with HSMs:** Store the most sensitive keys in a hardware or cloud-based HSM.
3. **Use trusted crypto implementations:** Use well-trusted implementations of cryptographic algorithms whenever possible.

4. **Minimize sensitive data retention:** Do not store sensitive data unnecessarily. Discard it as soon as possible, or use PCI DSS-compliant tokenization or truncation. Data that is not retained cannot be stolen.
5. **Encrypt at rest:** Ensure all sensitive data is encrypted at rest.
6. **Use strong modern standards:** Ensure up-to-date and strong standard algorithms, protocols, and keys are in place; apply proper key management.
7. **Encrypt in transit:** Encrypt all data in transit with protocols $\geq$ TLS 1.2 only, with forward secrecy (FS) ciphers; drop support for cipher block chaining (CBC) ciphers; support quantum key change algorithms. For HTTPS, enforce encryption using HTTP Strict Transport Security (HSTS). Validate configuration using dedicated tools.
8. **Disable caching for sensitive responses:** Disable caching in CDN, web server, and application caches (e.g., Redis) for responses containing sensitive data.
9. **Apply controls per classification:** Apply required security controls based on data classification.
10. **Avoid insecure protocols:** Do not use unencrypted protocols such as FTP and STARTTLS. Avoid using SMTP for transmitting confidential data.
11. **Strong password storage:** Store passwords using strong adaptive and salted hashing functions with a work factor (delay factor), such as Argon2, yescrypt, scrypt, or PBKDF2-HMAC-SHA-512. For legacy systems using bcrypt, consult the OWASP Cheat Sheet: Password Storage.
12. **IV/nonce correctness:** Choose initialization vectors appropriate for the mode of operation. This may require a CSPRNG. For modes requiring a nonce, the IV does not necessarily need a CSPRNG; in all cases, the IV must never be used twice for a fixed key.
13. **Prefer authenticated encryption:** Always use authenticated encryption instead of encryption alone.
14. **Key material handling:** Keys should be generated with cryptographic randomness and stored in memory as byte arrays. If a password is used, convert it to a key via an appropriate password-based key derivation function.
15. **Cryptographic randomness:** Ensure cryptographic randomness is used where appropriate and is not seeded predictably or with low entropy. Most modern APIs do not require manual seeding to be secure.
16. **Avoid deprecated constructions:** Avoid deprecated cryptographic functions and padding schemes, such as MD5, SHA1, CBC, and PKCS #1 v1.5.
17. **Configuration assurance:** Ensure settings and configurations meet security requirements by review from security specialists, automated tools, or both.
18. **Post-quantum readiness:** Prepare now for post-quantum cryptography (PQC), with high-risk systems safe no later than the end of 2030 (see ENISA reference).

5 Example Attack Scenarios

5.1 Scenario #1: Missing TLS Enforcement / Downgrade Attack

A site does not use or enforce TLS for all pages or supports weak encryption. An attacker monitors network traffic (e.g., on an insecure wireless network), downgrades connections from HTTPS to HTTP, intercepts requests, and steals a session cookie. The attacker then replays the cookie and hijacks the authenticated session, accessing or modifying the user's private data. Alternatively, the attacker could alter transported data, such as changing the recipient of a money transfer.

5.2 Scenario #2: Weak Password Hashing / Rainbow Table Exposure

A password database uses unsalted or simple hashes to store user passwords. A file upload flaw allows an attacker to retrieve the password database. Unsalted hashes can be exposed with rainbow tables of pre-calculated hashes. Hashes generated by fast hash functions may be cracked by GPUs, even if they were salted.

6 References

- OWASP Proactive Controls: C2: Use Cryptography to Protect Data
- OWASP Application Security Verification Standard (ASVS): V11, V12, V14
- OWASP Cheat Sheet: Transport Layer Protection
- OWASP Cheat Sheet: User Privacy Protection
- OWASP Cheat Sheet: Password Storage
- OWASP Cheat Sheet: Cryptographic Storage
- OWASP Cheat Sheet: HSTS
- OWASP Testing Guide: Testing for weak cryptography
- ENISA: A Coordinated Implementation Roadmap for the Transition to Post-Quantum Cryptography
- NIST Releases First 3 Finalized Post-Quantum Encryption Standards

7 List of Mapped CWEs

CWE	Title
CWE-261	Weak Encoding for Password
CWE-296	Improper Following of a Certificate's Chain of Trust
CWE-319	Cleartext Transmission of Sensitive Information
CWE-320	Key Management Errors (Prohibited)
CWE-321	Use of Hard-coded Cryptographic Key
CWE-322	Key Exchange without Entity Authentication
CWE-323	Reusing a Nonce, Key Pair in Encryption
CWE-324	Use of a Key Past its Expiration Date
CWE-325	Missing Required Cryptographic Step
CWE-326	Inadequate Encryption Strength
CWE-327	Use of a Broken or Risky Cryptographic Algorithm
CWE-328	Reversible One-Way Hash
CWE-329	Not Using a Random IV with CBC Mode
CWE-330	Use of Insufficiently Random Values
CWE-331	Insufficient Entropy
CWE-332	Insufficient Entropy in PRNG
CWE-334	Small Space of Random Values
CWE-335	Incorrect Usage of Seeds in Pseudo-Random Number Generator (PRNG)
CWE-336	Same Seed in Pseudo-Random Number Generator (PRNG)
CWE-337	Predictable Seed in Pseudo-Random Number Generator (PRNG)
CWE-338	Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG)
CWE-340	Generation of Predictable Numbers or Identifiers
CWE-342	Predictable Exact Value from Previous Values
CWE-347	Improper Verification of Cryptographic Signature
CWE-523	Unprotected Transport of Credentials
CWE-757	Selection of Less-Secure Algorithm During Negotiation (*Algorithm Downgrade*)
CWE-759	Use of a One-Way Hash without a Salt
CWE-760	Use of a One-Way Hash with a Predictable Salt
CWE-780	Use of RSA Algorithm without OAEP
CWE-916	Use of Password Hash With Insufficient Computational Effort
CWE-1240	Use of a Cryptographic Primitive with a Risky Implementation
CWE-1241	Use of Predictable Algorithm in Random Number Generator