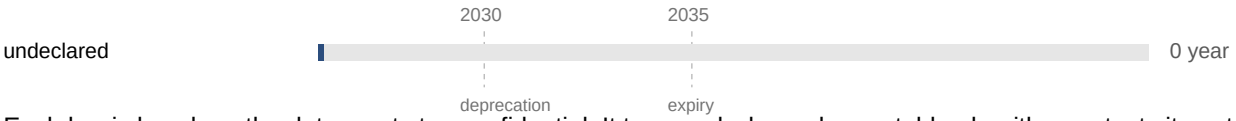
Sablier — BlackMesa Labs

No data whose confidentiality lifetime outlasts the expiry of the algorithms protecting it.

2 findings kept across everything analysed. Expiry date used: 2035 — that is the regulatory deadline, not a prediction of when the maths breaks. Deadline last checked against its sources on 30/09/2026.

How long data must stay confidential, against the expiry of the algorithms



Each bar is how long the data must stay confidential. It turns red when a harvestable algorithm protects it past the expiry line — crossing the line without turning red means the data lives long but is protected by cryptography that will hold.

What the server actually negotiates

mesa.black

service	HTTPS
negotiated protocol	TLSv1.3
cipher suite	TLS_AES_128_GCM_SHA256 (128 bits)
negotiated group	X25519MLKEM768
certificate signature	ecdsa-with-SHA384
certificate key	elliptic curve 256 bits
chain	4 certificates
certificate valid until	12/01/2027
accepted versions	TLSv1.3

A handshake, not a file read: it is the only way to know what really protects the traffic, and it can contradict what the repository declares.

WATCH 1

ECDSA undeclared

Signature: not harvestable. Migrate before 2030 for compliance, with no confidentiality urgency. Server certificate. Its lifetime is short: replacing it with a post-quantum signature happens at renewal, with no data migration.

Accept this finding: sablier accept 4d2a0227 --reason="…" --until=2027-05-20

The tool is wrong? Open a pre-filled report

CLEAR 1

- tls://mesa.black:443:0

What this report did not look at

- the cryptography of your managed services — database, object storage, TLS terminated by an intermediary — appears in no file of this repository
- what is actually negotiated at runtime, for any host not declared in the probe section
- keys held in an HSM or by a provider
- the real lifetime of the data: it comes from your declaration, not from the code — an undeclared domain is

computed with a default lifetime of 0 years

- The system's service horizon is not declared: the calculation assumes it stops writing data today. That is the optimistic reading — a service still in production in ten years will produce ten more years of data to protect. Add `service_until` to the declaration.
- Expiry used: 2035, after NIST IR 8547. Another regime — institutional, defence — gives another date: see `regime` in the declaration.
- Checksums (CRC32, Adler-32) are reported only beside a token, a link or a secret. A Luhn check digit or a modulo written by hand never is: nothing tells it apart from ordinary arithmetic, and code that accepts a number on the strength of its check digit does not appear here.
- Binary files (images, fonts, archives) are read in full up to 8 MiB (8,388,608 bytes) — anything larger is not read at all — and only for three verifiable signs: a key block, bytes after the end of the image, an extension that lies about the content. A message hidden in the bits of an image is not detected here, and does not claim to be.
- YAML configuration files are read as text, not interpreted: an algorithm set through a parameter, an imported file or an environment variable is not seen here.

An inventory that does not say what it failed to look at is not an inventory.

What it looked for and did not find

Every line here is a result rather than a silence: a detector opened these files and got nothing out of them. The count is there so you can judge how far the sentence reaches instead of taking it.

- 6 PHP file(s) read: no cryptographic call.
- 401 file(s) read: no private key, no certificate, no keystore in the versioned tree.
- 112 binary file(s) examined: none carries a key block, bytes past the end of the image, or an extension that lies about its content.
- 2 dependency file(s) read: no catalogued cryptographic library.
- 2 web server configuration(s) read: no cipher and no protocol version declared.
- 7 script(s) and Dockerfile(s) read: no cryptographic command.

What there is to change

1 place(s) to revisit, across 1 file(s). The rows below count places, not weeks.

Algorithm	Call sites	Files	Domains	What that changes
ECDSA	1	1	1	

The EU roadmap counts the migration effort among the three factors of quantum risk. This tool counts it in places, because that is what a repository can tell; it does not turn that into a duration. Eight call sites in one file behind one function are an afternoon; eight across six services with a protocol between them are a quarter, and nothing in the files distinguishes the two. That estimate is yours.

What to do, in order

An inventory that concludes nothing gets filed away. Here is the arbitration — and the order matters as much as the list.

- Nothing urgent, and that is a resultNo long-lived data is exposed to harvesting, and nothing is broken today. Doing nothing is the right decision here: replacing cryptography that holds costs time, introduces risk and improves nothing.
- Put this report back on the calendarThe window closes by itself: the expiry date does not move (2035), but every year that passes brings your data closer to that line. A domain that is clear today with a nine-year lifetime will not be next year. Re-running the analysis once a year is enough — and checking, while you are there, that the regulatory deadline has not moved.

Send the summary on Threema

Opens Threema on a phone where the app is installed. On a desktop the scheme is usually not registered and the browser refuses the link: copy the text below instead. Nothing is sent to a third party.

False positive?

Two different things get called a false positive, and they do not go to the same place.

1. The tool is right, but this finding is accepted here. That is a project decision: add this block to your declaration. It is versioned, so the review happens in code review. A reason and an expiry date are both required.

On the command line: sablier accept <fingerprint> --reason="…" --until=2027-05-20

2. The tool is wrong. That is a rule to fix, and it belongs in Sablier's own repository: github.com/mesa-black/sablier

The fingerprint printed next to each finding identifies it, and it is what you quote in either case. It is computed from the evidence rather than the line number: it survives the code moving, and it changes the day the line materially changes.

Digest and signature

findings digest	8ecb829ccc278ef7ab3138f56e07ee13319cbb72ce1dabbfc58ee7b4d6b670b5
signed on	Ed25519 + ML-DSA-65 · 16/10/2026 22:20
public key	zEMKv+R/IT+XlaqW…
page signature	report.html.page.sig
date attested	06/10/2026 21:58 UTC · Asseco Data Systems S.A.
Bitcoin anchoring	block 976238 · 17/11/2026 06:11 UTC

The digest covers the findings, not this file: two renderings of the same inventory, in two languages, give the same value.

This report carries two signatures: Ed25519, verifiable anywhere, and ML-DSA-65, which resists the known quantum algorithms. Neither replaces the other — it is the hybridation ANSSI asks for, applied to the document itself.

What the timestamp establishes: the digest above existed on that date, according to a third party with no stake in these conclusions. No signature on this report can prove that, because the date it carries comes from the clock of the machine that signed. Its limit: the token is signed with RSA-4096, which this report itself classes as quantum vulnerable — it is evidence for a dispute in the next few years, not for 2040.

The date is checked without this tool, with the token filed beside the report:

Three steps rather than one line, and each repairs a false assumption. The certificate store is not at the same path from one system to the next — the command everybody publishes, `-CAfile /etc/ssl/certs/ca-certificates.crt`, fails on Fedora, Rocky and openSUSE, where that file does not exist. And the token carries its full chain, but LibreSSL — the openssl shipped with macOS — does not read it and answers `unable to get local issuer certificate`: handing it over with `-untrusted` changes nothing elsewhere and settles that case. Checked as printed on Debian, Ubuntu, Alpine, Fedora, Rocky, openSUSE, and on macOS with both openssl builds.

The same imprint was presented to other authorities, in other jurisdictions. Each signed its own token knowing nothing of the others, and each sets an independent bound: the date defensible without trusting any single operator is the latest in this list, so forging the earliest no longer detaches this document from a date.

- 08/10/2026 02:19 UTC, by GlobalSign nv-sa — token in report.html.2.tsr, verifiable with the same recipe.
- 08/10/2026 02:19 UTC, by DigiCert — token in report.html.3.tsr, verifiable with the same recipe.

What the anchoring establishes: the digest above existed no later than that block of the Bitcoin chain, which thousands of independent machines hold. Nobody has to be believed: the proof is a chain of SHA-256 operations from the digest to the block's merkle root, checked without this tool and without any certificate. Its limits: it proves when, not who — that is the signature's job. Against a quantum computer, the proof itself is hashes only (SHA-256, sometimes RIPEMD-160), which Grover's algorithm weakens without breaking, so it does not expire like the tokens' RSA; whether the Bitcoin chain keeps its consensus through that transition is outside what the proof can say.

Checked without any authority, with the proof filed beside the report:

The official client (pip install opentimestamps-client) takes the block header from a Bitcoin node of your own. sablier verify takes it from two public explorers that must agree, and checks the header's proof of work itself rather than believing them; on a machine with no network, the header is passed by hand with --block-header.

This date is earlier than the report's (20/11/2026 09:46 UTC), and that is not an inconsistency: the findings have not moved since, so the original signature and the original token are both kept. Only the rendering changed. The earliest attestation is the one that carries weight — re-signing and re-attesting on every run would destroy the antecedence it establishes, for no new information at all.

What the signature guarantees: the list of what was found — each algorithm, the file it was found in, the verdict it was given. It does not depend on how this page is laid out or on the language it is written in: the same analysis checks out in French, English or Spanish. The page itself has its own signature, in the file report.html.page.sig filed beside it. That one is made from a fingerprint of the file — change one character and the fingerprint changes completely — so if a single character of this page is changed, a verdict, a date, a comma, checking it fails. What neither signature can stop: whoever holds the signing key can sign another page. Which key is the right one is the subject of the next paragraph. One command checks both: sablier verify report.html.sig --declare=sablier.json

What the signatures establish: the list and the page were signed by the key the declaration designates. What ties that key to somebody is not this page — it is the history of the file that declares it, where any change of key is a dated commit anybody can read.