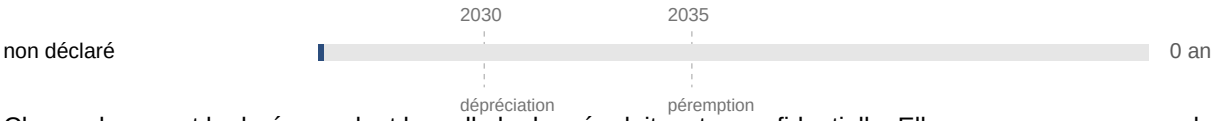
Sablier — BlackMesa Labs

Aucune donnée dont la durée de confidentialité dépasse la péremption des algorithmes qui la protègent.

2 constats retenus sur l'ensemble analysé. Date de péremption retenue : 2035 — c'est l'échéance réglementaire, pas une prédiction de rupture cryptographique. Échéance vérifiée auprès de ses sources le 30/09/2026.

Durée de confidentialité des données, face à la péremption des algorithmes



Chaque barre est la durée pendant laquelle la donnée doit rester confidentielle. Elle passe en rouge quand un algorithme récoltable la protège au-delà du trait de péremption — dépasser le trait sans être rouge signifie que la donnée dure longtemps, mais qu'elle est protégée par de la cryptographie qui tiendra.

Ce que le serveur négocie réellement

mesa.black

service	HTTPS
protocole négocié	TLSv1.3
suite	TLS_AES_128_GCM_SHA256 (128 bits)
groupe négocié	X25519MLKEM768
signature du certificat	ecdsa-with-SHA384
clé du certificat	courbe elliptique 256 bits
chaîne	4 certificats
certificat valide jusqu'au	12/01/2027
versions acceptées	TLSv1.3

Un handshake, pas une lecture de fichier : c'est le seul moyen de savoir ce qui protège vraiment le trafic, et cela peut contredire ce que le dépôt déclare.

SURVEILLER 1

ECDSA non déclaré

Signature : pas de récolte possible. À migrer avant 2030 pour la conformité, sans urgence de confidentialité. Certificat serveur. Sa durée de vie est courte : le remplacer par une signature post-quantique se fera au renouvellement, sans migration de données.

Accepter ce constat : sablier accept 4d2a0227 --reason="…" --until=2027-05-20

L'outil a tort ? Ouvrir un signalement pré-rempli

CONFORME 1

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

Ce que ce rapport n'a pas regardé

- la cryptographie de vos services gérés — base de données, stockage objet, terminaison TLS chez un intermédiaire — n'apparaît dans aucun fichier de ce dépôt
- ce qui est réellement négocié à l'exécution, pour tout hôte non déclaré dans la section « sonde »
- les clés détenues dans un HSM ou chez un fournisseur

- la durée de vie réelle des données : elle vient de votre déclaration, pas du code — un domaine non déclaré est calculé avec une durée par défaut de 0 ans
- L'horizon d'exploitation du système n'est pas déclaré : le calcul suppose qu'il cesse d'écrire des données aujourd'hui. C'est l'hypothèse optimiste — un service encore en production dans dix ans produira dix ans de données de plus à protéger. Ajoutez `service_until` à la déclaration.
- Échéance retenue : 2035, d'après NIST IR 8547. Un autre cadre — institutionnel, défense — donne une autre date : voir `regime` dans la déclaration.
- Les sommes de contrôle (CRC32, Adler-32) ne sont signalées qu'à côté d'un jeton, d'un lien ou d'un secret. Une clé de Luhn ou un modulo écrits à la main ne le sont jamais : rien ne les distingue d'un calcul ordinaire, et un code qui accepte un numéro sur la foi de son chiffre de contrôle n'apparaît pas ici.
- Les fichiers binaires (images, polices, archives) sont lus en entier jusqu'à 8 Mio (8 388 608 octets) — au-delà, ils ne sont pas lus du tout —, et seulement pour trois signes vérifiables : un bloc de clé, des octets après la fin de l'image, une extension qui ment sur le contenu. Un message dissimulé dans les bits d'une image n'est pas détecté ici, et ne prétend pas l'être.
- Les fichiers YAML de configuration sont lus comme du texte, pas interprétés : un algorithme défini par un paramètre, par un fichier importé ou par une variable d'environnement n'est pas vu ici.

Un inventaire qui ne dit pas ce qu'il n'a pas vu n'est pas un inventaire.

Ce qu'il a cherché et n'a pas trouvé

Chaque ligne est un résultat, pas un silence : un détecteur a ouvert ces fichiers et n'en a rien tiré. Le nombre est là pour que vous puissiez juger la portée de la phrase plutôt que la croire.

- 6 fichier(s) PHP lu(s) : aucun appel cryptographique.
- 401 fichier(s) lu(s) : aucune clé privée, aucun certificat, aucun magasin de clés dans l'arborescence versionnée.
- 112 fichier(s) binaire(s) examiné(s) : aucun ne porte de bloc de clé, d'octets après la fin de l'image, ni d'extension qui mente sur son contenu.
- 2 fichier(s) de dépendances lu(s) : aucune bibliothèque cryptographique du catalogue.
- 2 configuration(s) de serveur web lue(s) : aucun chiffre ni version de protocole déclarés.
- 7 script(s) et Dockerfile lu(s) : aucune commande cryptographique.

Ce qu'il y a à changer

1 emplacement(s) à reprendre, dans 1 fichier(s). Les lignes ci-dessous comptent des endroits, pas des semaines.

Algorithme	Appels	Fichiers	Domaines	Ce que cela change
ECDSA	1	1	1	

La feuille de route européenne compte l'effort de migration parmi les trois facteurs du risque quantique. Cet outil le compte en emplacements parce que c'est ce qu'un dépôt sait dire ; il n'en fait pas une durée. Huit appels dans un fichier derrière une même fonction sont une après-midi ; huit répartis sur six services avec un protocole entre eux sont un trimestre, et rien dans les fichiers ne distingue les deux. Cette estimation vous appartient.

Ce qu'il faut faire, dans l'ordre

Un inventaire qui ne conclut rien se range dans un dossier. Voici l'arbitrage — et l'ordre compte autant que la liste.

- Rien d'urgent, et c'est un résultatAucune donnée à longue durée n'est exposée à la récolte, et rien n'est cassé aujourd'hui. Ne rien faire est ici la bonne décision : remplacer de la cryptographie qui tient coûte du temps, introduit du risque et n'améliore rien.
- Remettre ce rapport au calendrierLa fenêtre se referme d'elle-même : la date de péremption ne bouge pas (2035), mais chaque année qui passe rapproche vos données de cette ligne. Un domaine conforme aujourd'hui avec une durée de neuf ans ne le sera plus l'an prochain. Rejouer l'analyse une fois par an suffit — et vérifier à cette occasion que l'échéance réglementaire n'a pas bougé.

Envoyer le résumé sur Threema

Ouvre Threema sur un téléphone où l'application est installée. Sur un ordinateur, le schéma n'est en général pas enregistré et le navigateur refuse le lien : copiez le texte ci-dessous. Rien n'est transmis à un serveur tiers.

Faux positif ?

Deux choses s'appellent « faux positif », et elles ne vont pas au même endroit.

1. L'outil a raison, mais ce constat est accepté ici. C'est une décision de projet : ajoutez ce bloc à votre déclaration, elle est versionnée, donc la relecture se fait en revue de code. La raison et la date d'expiration sont obligatoires.

En ligne de commande : sablier accept <empreinte> --reason="…" --until=2027-05-20

2. L'outil a tort. C'est une règle à corriger, et sa place est dans le dépôt de Sablier : github.com/mesa-black/sablier

L'empreinte imprimée à côté de chaque constat le désigne, et c'est elle qu'on cite dans les deux cas. Elle est calculée sur la preuve, pas sur le numéro de ligne : elle survit à un déplacement du code, et change le jour où la ligne change vraiment.

Empreinte et signature

empreinte des constats	8ecb829ccc278ef7ab3138f56e07ee13319cbb72ce1dabbfc58ee7b4d6b670b5
signé le	Ed25519 + ML-DSA-65 · 16/10/2026 22:20
clé publique	zEMKv+R/IT+XlaqW…
signature de la page	report.html.page.sig
date attestée	06/10/2026 21:58 UTC · Asseco Data Systems S.A.
ancrage Bitcoin	bloc 976238 · 17/11/2026 06:11 UTC

L'empreinte porte sur les constats, pas sur ce fichier : deux rendus du même inventaire, dans deux langues, donnent la même valeur.

Ce rapport porte deux signatures : Ed25519, vérifiable partout, et ML-DSA-65, qui résiste aux algorithmes quantiques connus. Aucune ne remplace l'autre — c'est l'hybridation que l'ANSSI demande, appliquée ici au document lui-même.

Ce que l'horodatage établit : l'empreinte ci-dessus existait à cette date, selon un tiers qui n'a aucun intérêt dans ces conclusions. Aucune signature de ce rapport ne peut le prouver, puisque la date qu'elle porte sort de l'horloge de la machine qui a signé. Sa limite : le jeton est signé en RSA-4096, que ce rapport classe lui-même comme vulnérable au quantique — c'est une preuve pour un litige dans les prochaines années, pas pour 2040.

La date se vérifie sans cet outil, avec le jeton déposé à côté du rapport :

Trois étapes plutôt qu'une ligne, et chacune répare une hypothèse fausse. Le chemin du magasin de certificats n'est pas le même d'un système à l'autre — la commande qu'on lit partout, `-CAfile /etc/ssl/certs/ca-certificates.crt`, échoue sur Fedora, Rocky et openSUSE, où ce fichier n'existe pas. Et le jeton porte sa chaîne complète, mais LibreSSL — l'openssl livré avec macOS — ne la lit pas et répond `unable to get local issuer certificate` : la lui donner avec `-untrusted` ne change rien ailleurs et règle ce cas. Vérifié tel quel sur Debian, Ubuntu, Alpine, Fedora, Rocky, openSUSE, et sur macOS avec les deux openssl.

La même empreinte a été présentée à d'autres autorités, dans d'autres juridictions. Chacune a signé son propre jeton sans rien savoir des autres, et chacune pose une borne indépendante : la date qu'on peut défendre sans accorder sa confiance à un seul opérateur est la plus tardive de cette liste, et falsifier la plus ancienne ne suffit donc plus à détacher ce document de sa date.

- 08/10/2026 02:19 UTC, par GlobalSign nv-sa — jeton dans report.html.2.tsr, vérifiable avec la même recette.
- 08/10/2026 02:19 UTC, par DigiCert — jeton dans report.html.3.tsr, vérifiable avec la même recette.

Ce que l'ancrage établit : l'empreinte ci-dessus existait au plus tard à ce bloc de la chaîne Bitcoin, que des milliers de machines indépendantes détiennent. Personne n'a à être cru : la preuve est une suite d'opérations SHA-256 qui mène de l'empreinte à la racine de Merkle du bloc, vérifiée sans cet outil et sans aucun certificat. Ses limites : elle prouve quand, pas qui — c'est le rôle de la signature. Face à un ordinateur quantique, la preuve elle-même n'est faite que d'empreintes (SHA-256, parfois RIPEMD-160), que l'algorithme de Grover affaiblit sans les casser :

elle n'expire pas comme le RSA des jetons ; que la chaîne Bitcoin garde son consensus à travers cette transition est hors de ce que la preuve peut dire.

Se vérifie sans aucune autorité, avec la preuve déposée à côté du rapport :

Le client officiel (pip install opentimestamps-client) prend l'en-tête du bloc auprès d'un nœud Bitcoin à vous. sablier verify le prend auprès de deux explorateurs publics qui doivent concorder, et vérifie lui-même la preuve de travail de l'en-tête plutôt que de les croire ; sur une machine sans réseau, l'en-tête se passe à la main avec --block-header.

Cette date est antérieure à celle du rapport (20/11/2026 09:46 UTC), et ce n'est pas une incohérence : les constats n'ont pas bougé depuis, donc la signature et le jeton d'origine sont tous deux conservés. Seul le rendu a changé. C'est l'attestation la plus ancienne qui a une valeur — en resignant et en réattestant à chaque passage, on détruirait l'antériorité qu'elle établit, pour n'y gagner aucune information nouvelle.

Ce que la signature garantit : la liste de ce qui a été trouvé — chaque algorithme, le fichier où il a été trouvé, le verdict qui lui a été donné. Elle ne dépend ni de la mise en page ni de la langue : la même analyse se vérifie en français, en anglais ou en espagnol. La page elle-même a sa propre signature, dans le fichier report.html.page.sig déposé à côté d'elle. Celle-là est calculée sur une empreinte du fichier — changez un seul caractère et l'empreinte change complètement —, si bien que si un seul caractère de cette page est modifié, un verdict, une date, une virgule, la vérification échoue. Ce qu'aucune des deux ne peut empêcher : celui qui détient la clé de signature peut signer une autre page. Savoir quelle clé est la bonne, c'est l'objet du paragraphe suivant. Une seule commande vérifie les deux : sablier verify report.html.sig --declare=sablier.json

Ce que les signatures établissent : la liste et la page ont été signées par la clé que la déclaration désigne. Ce qui rattache cette clé à quelqu'un n'est pas cette page — c'est l'historique du fichier qui la déclare, où tout changement de clé est un commit daté que n'importe qui peut relire.