CRYPTOLOGY MONSTER

Cryptographic utilities for people who want control, clarity, and a better everyday workflow.



SECURITY TOOLKIT

Passwords. Encryption. Keys. Signatures. Encodings. One browser workspace.

GENERATE

Random and deterministic credentials

PROTECT

Text and files with AES-GCM

VERIFY

Hashes, HMAC, JWT, PGP

Explore Cryptology Monster

Documentation

April 16, 2026

OVERVIEW

One toolkit. Many cryptographic tasks.

A focused interface for common cryptography, encoding, and verification workflows.

Cryptology Monster groups practical security utilities into a browser-based workspace. Select a tool, provide the relevant text, file, or key material, and receive the result through an on-screen output or a local download.

Password tools

Generate random passwords or reproducible seeded passwords; estimate text entropy and review basic password composition.

Encryption

Encrypt and decrypt text or files with AES-GCM; choose a generated key or a password-derived key.

Keys and OpenPGP

Create standalone RSA and Ed25519 keys; generate OpenPGP identities and use armored messages.

Verification utilities

Hash text/files; create or verify HMAC; inspect JWTs and verify supported HMAC signatures.

BUILT FOR HANDS-ON WORK

Additional tools cover Base32/Base64 encoding, string comparison, QR generation and decoding, plus a local-data purge action. The site states that it never saves or sells user data; operational limits are discussed on page 8.

Make stronger passwords. Keep control.

Random generation and reproducible generation serve different needs.

Random generation

Choose password length and generate using the browser cryptographic randomness API. Suitable for unique credentials that you store in a trusted manager.

Deterministic mode

Supply a seed phrase, site tag, and counter. The tool derives a repeatable password from those inputs; changing an input changes the result.

REPRODUCIBLE PASSWORD FLOW

SEED PHRASE	SITE TAG	COUNTER
Secret input	Account context	Version / rotation

Seeded mode is useful when you need reproducibility without storing each derived password individually. It is only as resilient as the secrecy and unpredictability of the master seed.

ENTROPY AND STRENGTH CHECKS

The entropy tool estimates information content in supplied text. The password-strength checker applies a simple composition score based on length, uppercase letters, digits, and symbols. These are guidance tools, not formal password-cracking simulations.

BEST PRACTICE **Use a unique strong seed; never publish it or reuse it casually.**

Protect messages and files.

AES-GCM authenticated encryption with practical key handling.

01

CHOOSE INPUT

Enter plaintext or select a file.

02

SELECT KEY MODE

— Generated raw key or password-derived key.

03

SAVE OUTPUT

— Download ciphertext and retain key material.

Text encryption

Encrypt a plaintext entry into a downloadable binary file, then decrypt it back to text using the matching key file or password.

File encryption

Encrypt an uploaded file into a downloadable .enc output. The corresponding decryption workflow restores the original file bytes.

KEY HANDLING MATTERS

- Raw-key mode exports a key file; losing that key can make decryption impossible.
- Password mode derives a key from the supplied password using PBKDF2.
- Never share the secret through the same channel as sensitive encrypted content.

Keys, messages, and signatures.

Standalone key generation and an OpenPGP message workflow.

STANDALONE KEY GENERATION

RSA keypairs

Export RSA-OAEP public and private PEM keys.

Ed25519 keypairs

Export Ed25519 signing and verification PEM keys.

OPENPGP OPERATIONS

Create identities

- User ID with name and email
- Ed25519 or RSA PGP keys
- Optional private-key passphrase
- Public/private ASCII armor and possible revocation certificate

Work with messages

- Encrypt with a recipient public key
- Decrypt with your private key
- Create cleartext signatures
- Verify a signed message with the signer public key

PRACTICAL TRUST MODEL

Distribute public keys deliberately. Protect private keys and passphrases. When exchanging OpenPGP keys, verify fingerprints through a trusted channel before assuming that a key belongs to the intended person.

From hashes to message authenticity.

Related concepts, different security guarantees.

Hashes

SHA-256, SHA-384, and SHA-512 for text or file digests. Use them to compare content fingerprints; hashes alone do not authenticate the source.

HMAC

Create and verify keyed message authentication codes with SHA-256, SHA-384, or SHA-512. Both sides need the same secret key.

JWT INSPECTION & SELECTIVE VERIFICATION

Decode a JSON Web Token to view its header, payload, and signature field. The optional signature check supports HS256, HS384, and HS512 when supplied with the shared secret. Decoding a token is not the same as verifying its authenticity.

DECODE != VERIFY

PICK THE RIGHT PRIMITIVE

HASH	Does this content match the expected digest?
HMAC	Did someone with the shared key authenticate this message?
JWT VERIFY	Does this supported HMAC signature match the supplied secret?

The small tools that save time.

Encoding, QR, comparison, and operational cross-checks.

Base64 & Base32

Encode or decode text for representation and transport. These encodings are reversible and do not provide secrecy.

QR workflows

Generate a QR code from text; download it as PNG or copy the image when supported. Read QR codes from uploaded images.

String comparison

Compare two input strings through a focused utility, useful for quickly spotting mismatches.

File hashing

Produce digest values from a chosen file for manual comparison with an expected checksum.

A QUICK DECISION GUIDE

Need a digest?	Use SHA hashing.
Need a shared-secret check?	Use HMAC.
Need confidentiality?	Use AES-GCM or PGP.
Need a QR or transfer encoding?	Use QR, Base32, or Base64.

Private workflows. Clear boundaries.

Use the toolkit carefully, especially when handling secret material.

PRIVACY AND OPERATIONAL CONTEXT

Cryptology Monster presents a privacy commitment that it does not save or sell user data. Core operations use browser APIs and downloaded results. Some features dynamically load third-party JavaScript libraries, so do not equate browser-side processing with a promise of zero network activity or independently audited security.

GOOD HABITS

- Back up keys and recovery material securely.
- Choose high-entropy passphrases.
- Verify public-key fingerprints.
- Double-check output before deleting originals.

KNOW THE LIMITS

- Encoding is not encryption.
- Password scores are heuristic.
- JWT decoding is not validation.
- Browser and dependency behavior varies.

EXPLORE CRYPTOLOGY MONSTER

Open the toolkit or explore the QVLx documentation hub for product help and additional guides.

Open Application | Documentation | Contact QVLx Labs