JUMBO BLOCK CHECK

Checksums and integrity of files and folders



Manual · Version 1.0.0

October 2026

ZANNI GROUP · Andreas Zanni e.K. · Brauhof 12 · 44866 Bochum · jumboblock.org

Contents

1. Overview
2. Basic terms
3. Workflow
4. Checking a file
5. Creating a manifest
6. Comparing
7. Message to a situation log
8. Verifying without Check
9. Files and formats
10. Security and technology
11. Troubleshooting

1. Overview

CHECK is the verification tool in the KRITIS area of JUMBO BLOCK. It calculates checksums of files and entire folders and thereby proves whether data is unchanged – for example a downloaded installation image, a backup before it is restored or a server's data before restarting after an attack.

CHECK works read-only. It never changes, moves or deletes a file. Results are only created as new files that you save yourself: manifests, checksum lists, check reports and messages to a situation log.

Fully offline: CHECK runs in the browser, also directly from a USB stick (`file://`). There is no server connection, no account and no cloud – nothing leaves your device.

Also for large files: files of several gigabytes are processed step by step. A progress window shows the file, amount and remaining time; the calculation can be cancelled at any time.

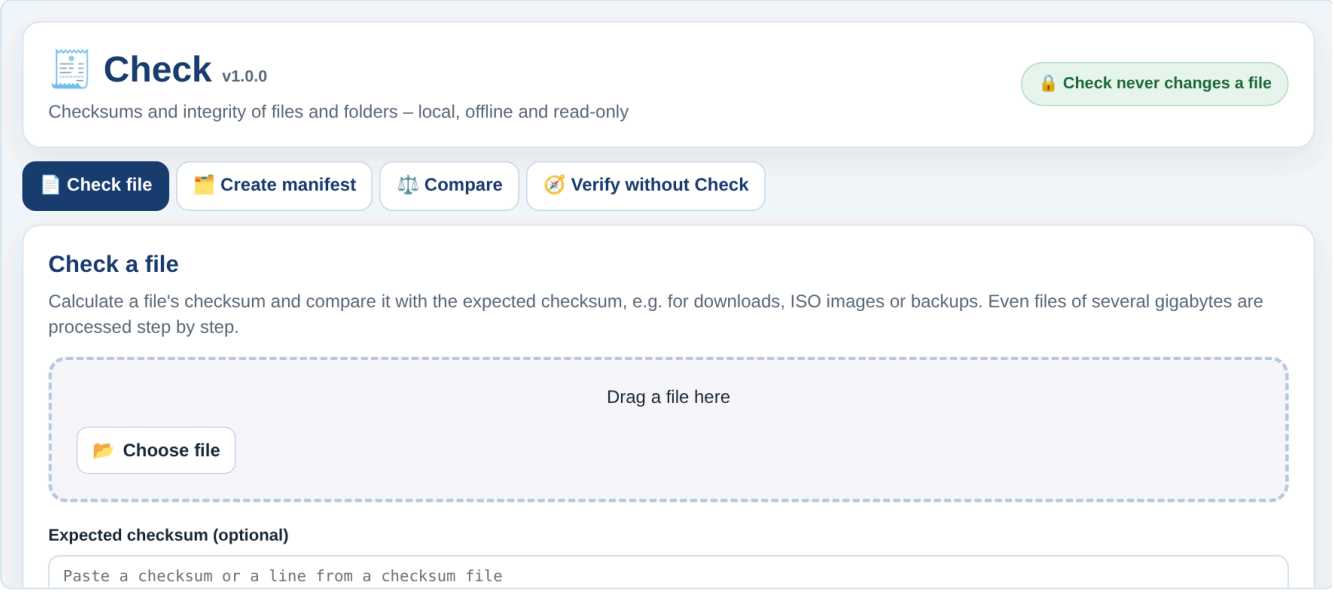


Fig. 1: The four areas of CHECK

Area	Purpose
Check file	Calculate a file's checksum and compare it with the manufacturer's expected checksum or a checksum file
Create manifest	Record checksum, size and modification date of all files of a folder including subfolders
Compare	Compare a folder or a second manifest with an earlier state (reference state) and show all deviations
Verify without Check	Instructions for verifying checksum lists independently with the built-in tools of Linux, macOS and Windows

2. Basic terms

Term	Meaning
Checksum (hash value)	Short, unique fingerprint of a file's content. If even a single bit changes, a completely different checksum results. File name and location are not part of the checksum.
Algorithm	Calculation method of the checksum. CHECK supports SHA-256, SHA-384, SHA-512, SHA-1 and MD5.
Expected checksum	Checksum published by the manufacturer or a trusted party, against which the calculated one is compared.
Manifest	CHECK file (.dbm) with path, size, modification date and checksum of every file in a folder. The manifest describes a fixed state.
Manifest checksum	SHA-256 checksum over the manifest's content. Kept separately, it later proves that the manifest itself was not changed.
Checksum list	Text file in the usual format of `sha256sum` (e.g. SHA256SUMS) that can also be verified without CHECK.
Reference state / current state	The reference state is the recorded state known to be correct (manifest or list). The current state is the folder found today or a second manifest.
Check report	Printable report on a check with result, checksums and fields for place, date and signature.

Which algorithm?

- **SHA-256** is the default and right for almost every purpose. SHA-384 and SHA-512 are equally secure.
- **MD5 and SHA-1** detect accidental errors such as transmission or storage errors, but not deliberate manipulation. CHECK marks them with ⚠. Only use them if a manufacturer publishes nothing else.

3. Workflow

CHECK shows its value when checksums are recorded **before** an incident. The following procedure has proven itself:

1. **Preparation:** regularly create a manifest for important folders – file servers, configurations, installation sources, backups (chapter 5).
2. **Keep the manifest checksum separately:** print it, pass it to the situation log as a message (chapter 7) or store it on a write-protected medium. Also keep the manifest itself in the emergency folder or on the USB stick.
3. **In an emergency:** before restarting, compare the affected folder with the manifest (chapter 6). Enter the noted checksum – this proves that the manifest is unchanged as well.
4. **Assess:** check changed and new files, restore missing files from a verified backup.
5. **Document:** print the check report and pass the result to the crisis team's situation log as a message.
6. **New reference state:** once clean-up is complete, create a new manifest from the current state.

> For software and installation images (e.g. JUMBO BLOCK Rescue), **Check file** is sufficient: choose the file, paste the published checksum, calculate.

4. Checking a file

In the *Check file* area CHECK calculates the checksum of a single file and compares it with the expected checksum.

1. Drag the file into the field or select it via *Choose file*. Name, size and modification date are displayed.
2. Paste the expected checksum (optional). It is sufficient to paste the entire line from a checksum file – CHECK recognises the checksum and, from its length, the algorithm. Alternatively, *Load checksum file* loads a file such as `.sha256` or `SHA256SUMS` and takes over the line for the chosen file.
3. Choose the algorithms. If an expected checksum is entered, CHECK selects the matching algorithm automatically; further algorithms can be ticked.
4. Click *Calculate checksum*.

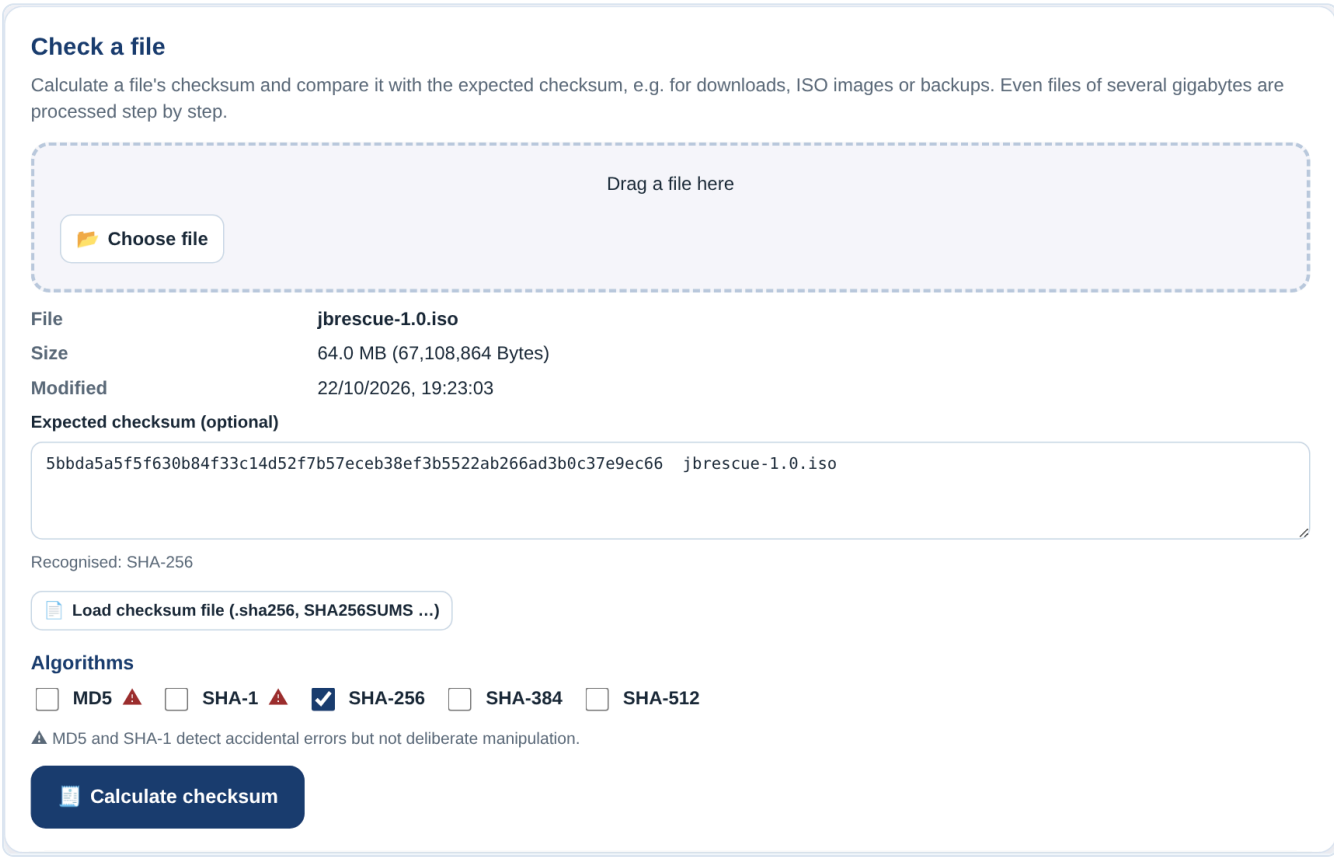


Fig. 2: Choosing the file, pasting the expected checksum and setting the algorithm

The result shows the calculated checksums and – if an expected checksum is available – an unambiguous verdict.

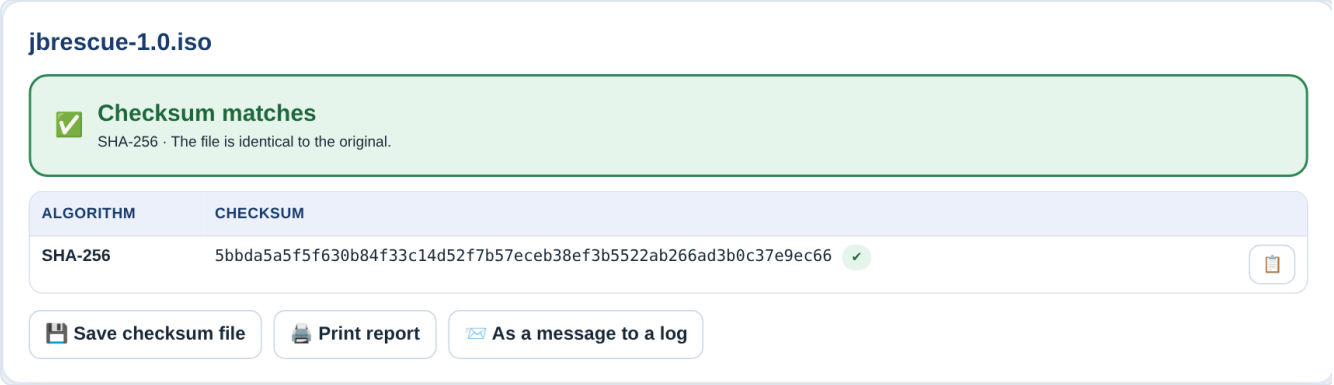


Fig. 3: Checksum matches – the file is identical to the original



Fig. 4: Checksum does not match – do not use the file

If the expected checksum is changed afterwards, CHECK updates the verdict without calculating again.

Button	Effect
Save checksum file	Saves the checksum as a file (e.g. `file.iso.sha256`) in the format of `sha256sum`
Print report	Check report with file, checksums, expected checksum and result – can also be saved as PDF
As a message to a log	Creates an encrypted message for JUMBO BLOCK LOGBOOK (chapter 7)

If the checksum does not match, the file is damaged, incomplete or altered. Do not use it; obtain it again from the original source.

5. Creating a manifest

A manifest records the state of an entire folder: for every file – in all subfolders as well – the relative path, size, modification date and checksum are stored.

1. Drag the folder into the field or select it via *Choose folder*. The browser asks once whether the files may be read. CHECK shows the number of files and total size.
2. Give a *Label*, e.g. “Building authority file server – state before restart”. The label appears in the manifest, in the check report and in the file name.
3. Choose the algorithm (recommended: SHA-256).
4. If required, tick *Skip system files*. Files such as `Thumbs.db`, `desktop.ini`, `.DS_Store` and recycle bin folders, which operating systems create and change on their own, are then ignored.
5. Click *Create manifest*.

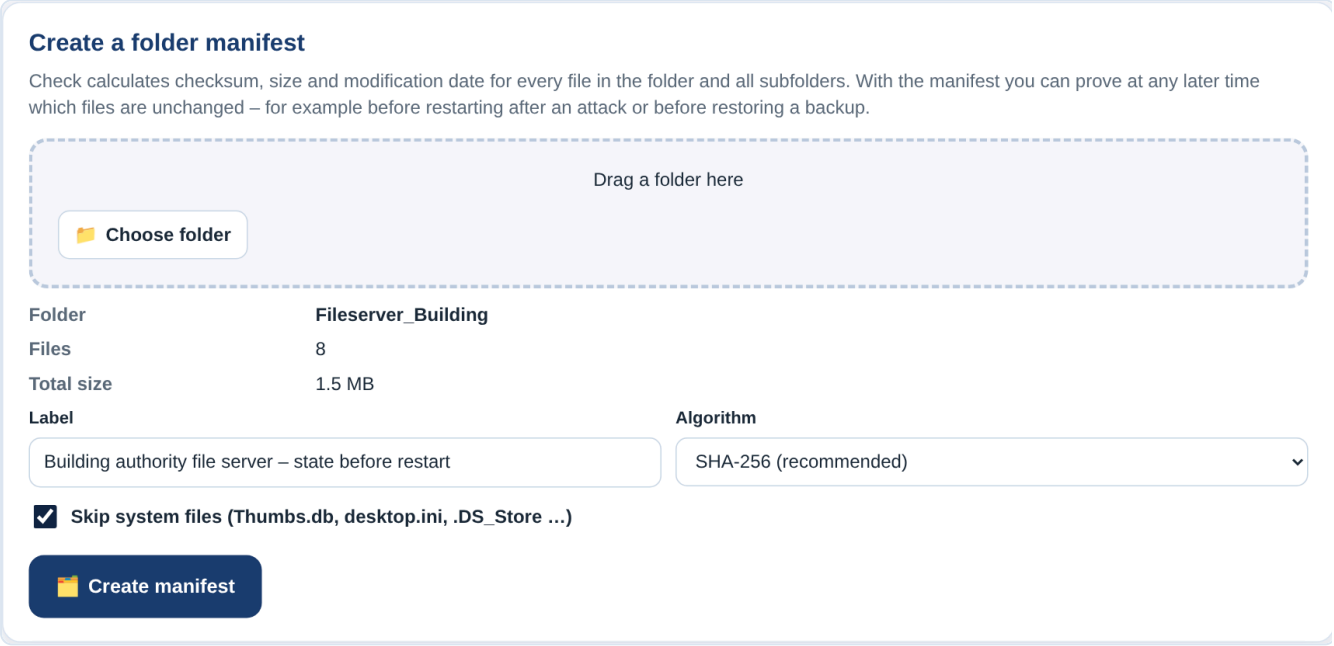


Fig. 5: Choosing the folder and creating the manifest

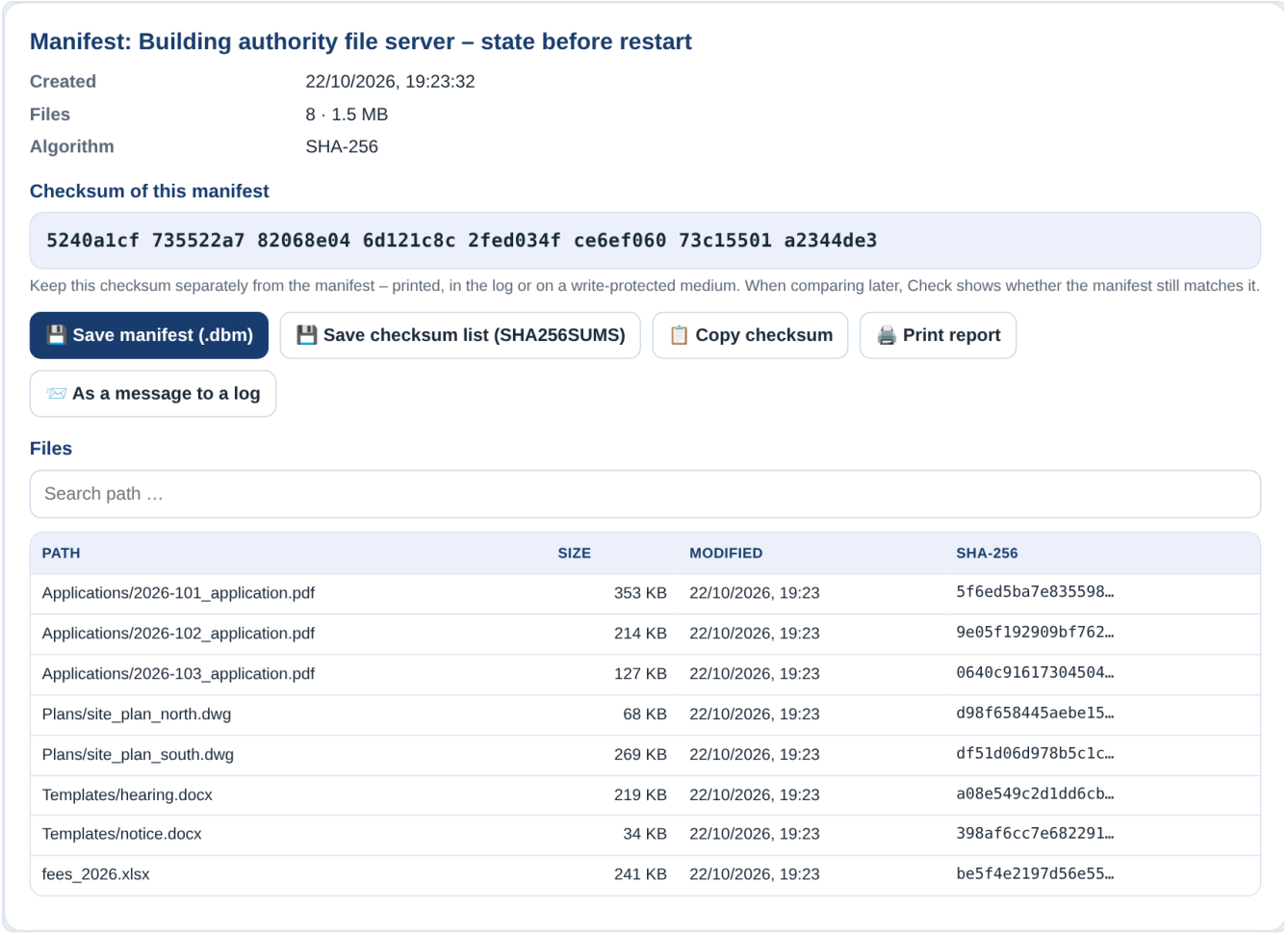


Fig. 6: Result with the manifest checksum and the file list

The file list can be filtered with *Search path*; with very many files further entries are loaded step by step. Files that could not be read are listed separately.

Button	Effect
Save manifest (.dbm)	Saves the manifest. This file is the reference state for later comparisons.
Save checksum list	Additionally saves the checksums as SHA256SUMS (or SHA512SUMS, MD5SUMS …) for verification without CHECK (chapter 8)
Copy checksum	Copies the manifest checksum to the clipboard
Print report	Check report with label, folder, manifest checksum and all files
As a message to a log	Secures the manifest checksum in the crisis team's situation log; the manifest is attached

Keep the manifest checksum separately. Anyone who manipulates a folder can also recalculate a manifest stored there. Only a checksum kept separately – printed, in the log or on a write-protected medium – gives the later comparison evidential value.

6. Comparing

In the *Compare* area CHECK sets a reference state against a current state.

1. Load the reference state

Load manifest or list opens a CHECK manifest (.dbm) or a checksum list from another source: SHA256SUMS, `*.sha256`, MD5SUMS etc. in the format of `sha256sum` as well as the BSD format (`SHA256 (file) = …`). For a manifest, CHECK immediately verifies its own checksum and shows *manifest unchanged* or *manifest altered*.

Under *Separately noted manifest checksum* enter the secured checksum – at least the first 16 characters. CHECK immediately shows whether the loaded manifest matches it.

2. Choose the current state

Choose the folder to be checked – or, via *Load second manifest*, another recorded state to compare two states without access to the files. *Skip system files* works as when creating the manifest.

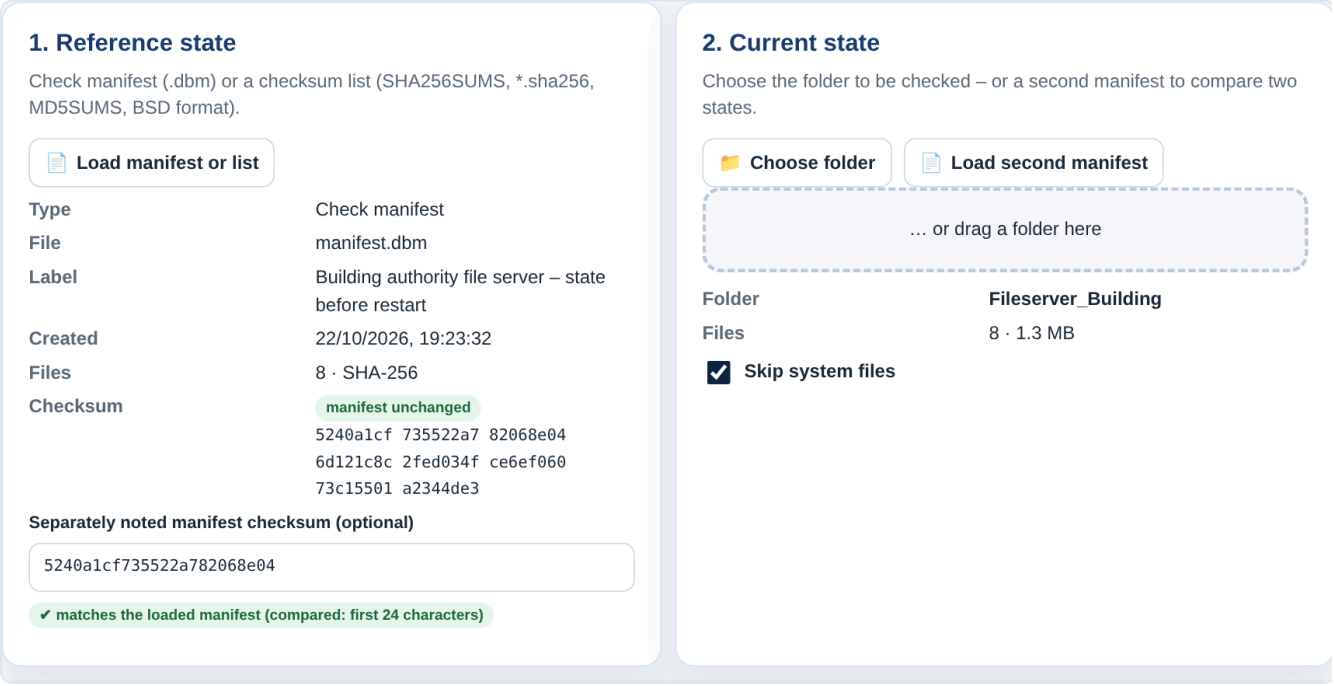


Fig. 7: Reference state with noted checksum and current state

3. Compare

Compare calculates the checksums of the current state with the same algorithm as in the reference state and assigns every file a status.

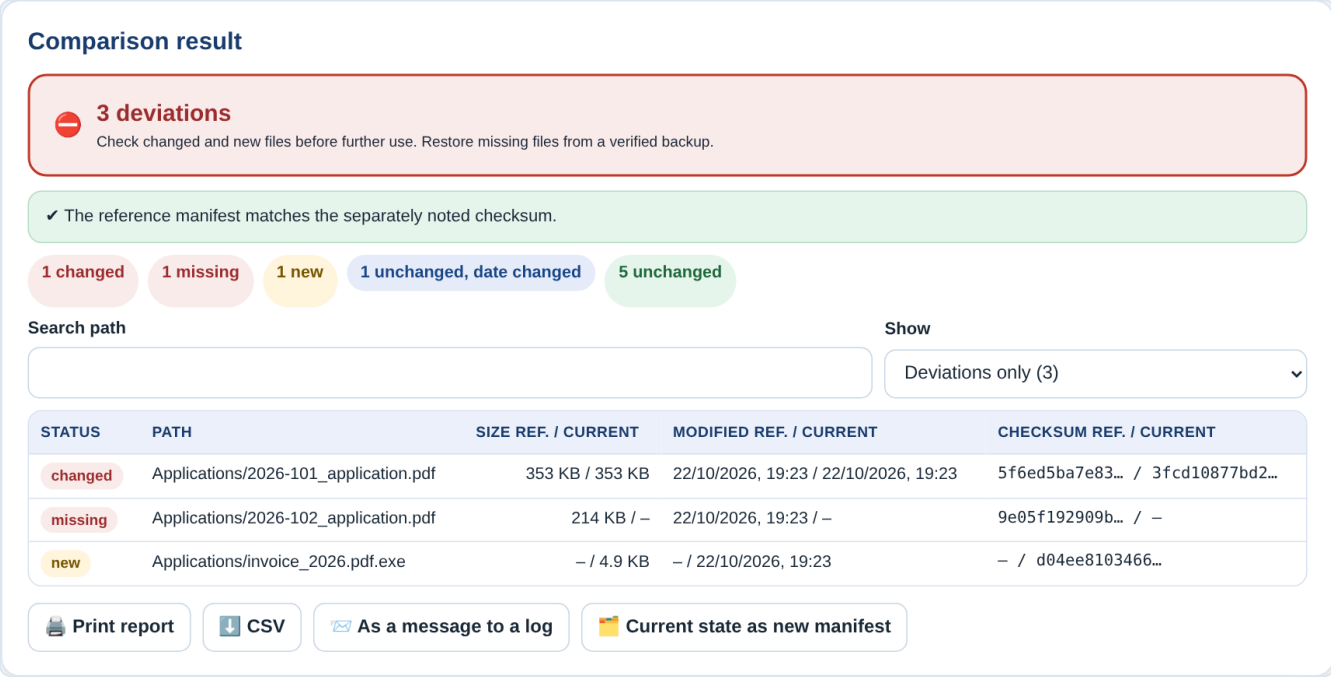


Fig. 8: Result with summary and list of deviations

Status	Meaning and action
unchanged	Content identical to the reference state
unchanged, date changed	Content identical, only the modification date differs – typical after copying or restoring
changed	Content differs – check the file, do not use it unverified
new	File did not exist in the reference state – pay particular attention after an attack (e.g. malware, ransom notes)
missing	File no longer exists in the current state – restore it from a verified backup
not in list	File in the folder that a third-party checksum list does not include; not assessed

The list initially shows deviations only; *Show* displays all files or individual statuses, *Search path* filters by name. For each file, size, modification date and checksum of the reference and current state are shown side by side.

Button	Effect
Print report	Comparison report with reference and current state, check against the noted checksum, summary and deviations
CSV	Result list for further processing, e.g. in a spreadsheet
As a message to a log	Result with an excerpt of the deviations to the crisis team
Current state as new manifest	Takes over the verified current state as a new manifest, e.g. after clean-up is complete

If the reference manifest is altered or does not match the noted checksum, CHECK points this out clearly: such a comparison has no evidential value.

7. Message to a situation log

Check results belong in the crisis team's documentation. CHECK creates a message file (.jbm) for JUMBO BLOCK LOGBOOK – from *Check file*, *Create manifest* and *Compare*, each via *As a message to a log*.

1. *Load message key (.jbk)*: you receive the message key from the crisis team. CHECK shows the log, organisation and fingerprint – compare the fingerprint by phone when possible. The message key is remembered on this device for the next message.
2. Enter the sender (name, unit). Subject and content are pre-filled and can be supplemented.
3. Click *Encrypt and save message* and hand the file over to the crisis team (USB stick, network drive, e-mail).

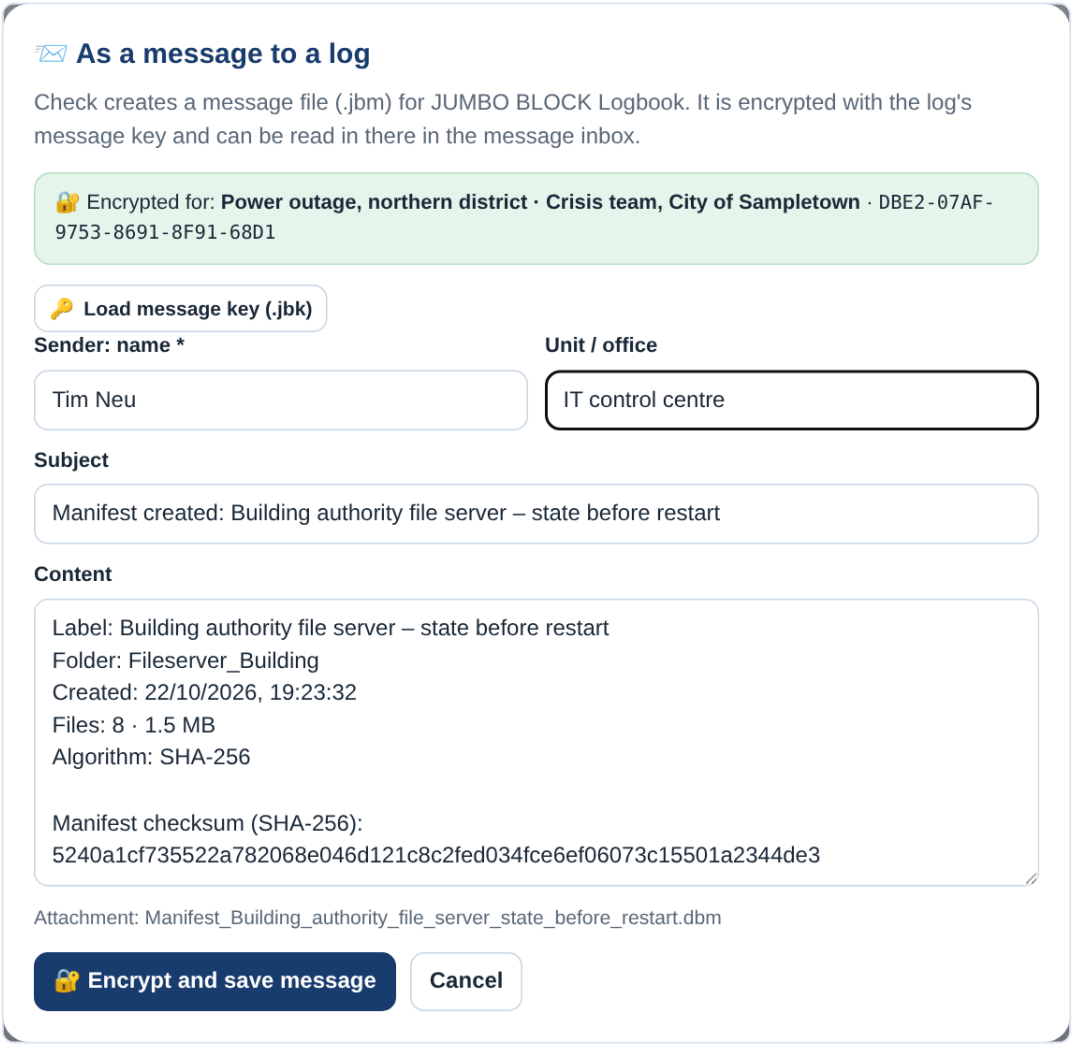


Fig. 9: Message with the manifest checksum; the manifest is attached

For a manifest, the message contains its checksum and the manifest as an attachment. This secures the checksum tamper-proof in the situation log: log entries are linked by a hash chain and cannot be changed afterwards.

> Without a message key a message can only be created explicitly unencrypted. This is not recommended because file and folder names may contain confidential information.

8. Verifying without Check

CHECK's checksum lists use the standard format of `sha256sum`. Every result can therefore be verified independently with the operating systems' built-in tools – important when a second proof is required. The *Verify without Check* area summarises the commands.

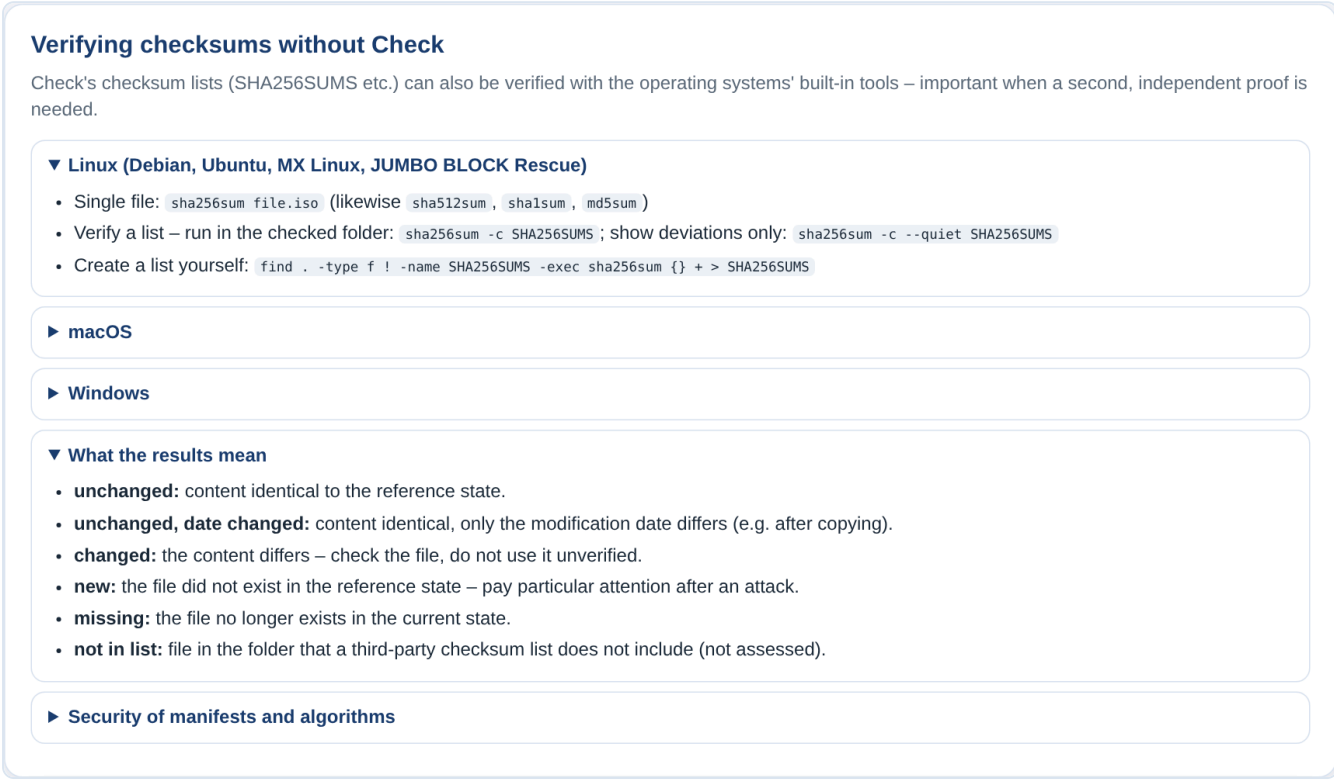


Fig. 10: Instructions in the program

System	Command
Linux (also JUMBO BLOCK Rescue)	Single file: `sha256sum file.iso` Verify a list (in the checked folder): `sha256sum -c SHA256SUMS` Deviations only: `sha256sum -c --quiet SHA256SUMS`
macOS	Single file: `shasum -a 256 file.iso` Verify a list: `shasum -a 256 -c SHA256SUMS`
Windows	PowerShell: `Get-FileHash file.iso -Algorithm SHA256` Command prompt: `certutil -hashfile file.iso SHA256` Windows does not verify a whole list itself – use CHECK for this.

For other algorithms use `sha512sum`, `sha1sum` and `md5sum` or `-Algorithm SHA512` etc. accordingly.

9. Files and formats

File	Content	Use
Manifest (.dbm)	Label, folder, time, algorithm and for each file path, size, modification date and checksum; with its own SHA-256 checksum	Reference state for *Compare*
SHA256SUMS, SHA512SUMS, MD5SUMS …	One line per file with checksum and relative path	Verification without CHECK, exchange with other tools
file.sha256 etc.	Checksum of a single file	Store next to the file or pass on
CSV	Comparison result with status, sizes, dates and checksums (semicolon, UTF-8)	Evaluation in a spreadsheet
Message (.jbm)	Encrypted message for LOGBOOK, possibly with the manifest as attachment	Message inbox of the situation log
Check report	A4 print view with fields for place, date and “Checked by”	Printout or PDF via the print dialog

A manifest is a JSON file in the suite's file format (.dbm). The manifest checksum is calculated over the canonically ordered content and can therefore be reproduced independently of formatting and field order.

10. Security and technology

Area	Implementation
Access	Read-only. Files and folders are read via the browser's selection or drag-and-drop function; CHECK cannot change any file.
Algorithms	SHA-256, SHA-384, SHA-512, SHA-1 and MD5 as original ZANNI GROUP code without third-party libraries; for SHA algorithms on smaller files additionally the browser's Web Crypto API
Large files	Step-by-step processing in segments of 4 MiB; memory use stays low regardless of file size.
Messages	Encryption for the log's message key with ECDH P-256, HKDF-SHA-256 and AES-256-GCM – compatible with JUMBO BLOCK LOGBOOK
Privacy	No network connection, no telemetry. Only the last loaded message key and the sender are remembered in the browser.

Limits

- A checksum proves *that* a file has changed – not *who* changed it. For origin, use a signature with JUMBO BLOCK CRYPT.
- Empty folders and file permissions are not recorded, only files with content, size and modification date.
- The modification date is supplied by the file system. It is a useful hint but proves nothing – the checksum is decisive.
- Browsers may slow down when reading very large folders. With very many files, record subfolders individually.

11. Troubleshooting

Problem	Cause and solution
“No valid checksum recognised”	The pasted string does not have 32, 40, 64, 96 or 128 hexadecimal characters. Paste it completely and without line break.
Checksum does not match, file just downloaded	Download incomplete or wrong version. Download again and check that the checksum belongs to the same version.
Many files “unchanged, date changed”	Normal after copying or restoring a backup. The content is decisive.
Many “new” files such as Thumbs.db or .DS_Store	Set *Skip system files* the same way when creating and when comparing.
“manifest altered”	The manifest was edited or damaged after creation. Use a separately secured copy; the comparison with this manifest has no evidential value.
Noted checksum does not match	Wrong manifest loaded or typing error. Compare at least 16 characters.
Folder cannot be chosen	Some browsers only allow folder selection after confirmation. Alternatively drag the folder into the field.
File “unreadable”	File is locked (e.g. opened by a running application) or lacks read permission. Close the application and check again.
Message key is rejected	The file is not a LOGBOOK message key or was altered. Request it again from the crisis team.