PUBLICATION EDITION — PRIVACY REDACTIONS APPLIED

From an Orphaned Encrypted Vault to a Confirmed Recovery

An extreme-case data recovery study involving obsolete SanDisk SecureAccess software, a missing original device, layered wallet encryption, and a legacy Bitcoin derivation path

Publication edition — privacy redactions applied

The client's original flash drive no longer existed. The software that created the encrypted vault had been discontinued for years. What remained was a collection of copied folders on old hard drives, an unreliable legacy launcher, seventeen encrypted payload records, and no remembered working password.

The recovery succeeded.

The copied vault was authenticated without altering the preserved evidence. Its password-verification behavior was reconstructed from the original signed 32-bit engine, accelerated on an AMD Radeon RX 5700 XT, and independently confirmed through the vendor engine. The decrypted export contained a 2015 MultiBit HD Bitcoin wallet. Its layered backups were recovered offline, 266 public addresses were checked, three funded legacy paths were identified, and approximately 0.0033 BTC was transferred to an owner-controlled destination and confirmed on the Bitcoin blockchain.

> Privacy note: wallet identifiers, passwords, recovery words, addresses, transaction identifiers, exact block/time combinations, personal paths, QR data, and private cryptographic material are intentionally hidden. Opaque redactions are part of the published images. The confidential evidence set retains the exact values and complete hashes.

Case at a glance

Challenge	Verified result
Original flash drive	Missing
Supported recovery software	No longer available
Surviving evidence	Three complete vault trees plus one matching archive copy
Legacy vault contents	20 files, including 17 encrypted payload records
Encrypted data	Approximately 474 MB
Source integrity	Matching structural manifests; SQLite integrity check passed
Password recovery	Two-stage GPU recovery plus original-engine confirmation
GPU	AMD Radeon RX 5700 XT
Export	14 files, including 12 dated MultiBit HD backups
Public wallet scan	266 addresses
Funded legacy paths	3
Recovered balance	Approximately 0.0033 BTC
Final state	Confirmed on-chain in July 2026

1. Starting with copies—not the missing device

The initial assumption was that the recovery might be impossible without the original SanDisk drive. That assumption changed once the surviving data was classified correctly.

Three complete SecureAccess vault trees were found in different backup locations, along with one archive copy. Each tree had the same 20-file structure and the same approximately 474 MB total size. Seventeen files were encrypted payload records. A normalized manifest showed no divergence among the surviving folder copies, and all relevant members inside the archive matched.

The vault database identified as SQLite and passed an integrity check. That mattered: a password is useless if the underlying encrypted records or database are corrupt.

The original copies were frozen as evidence. Every experiment was performed against a separate disposable working copy.

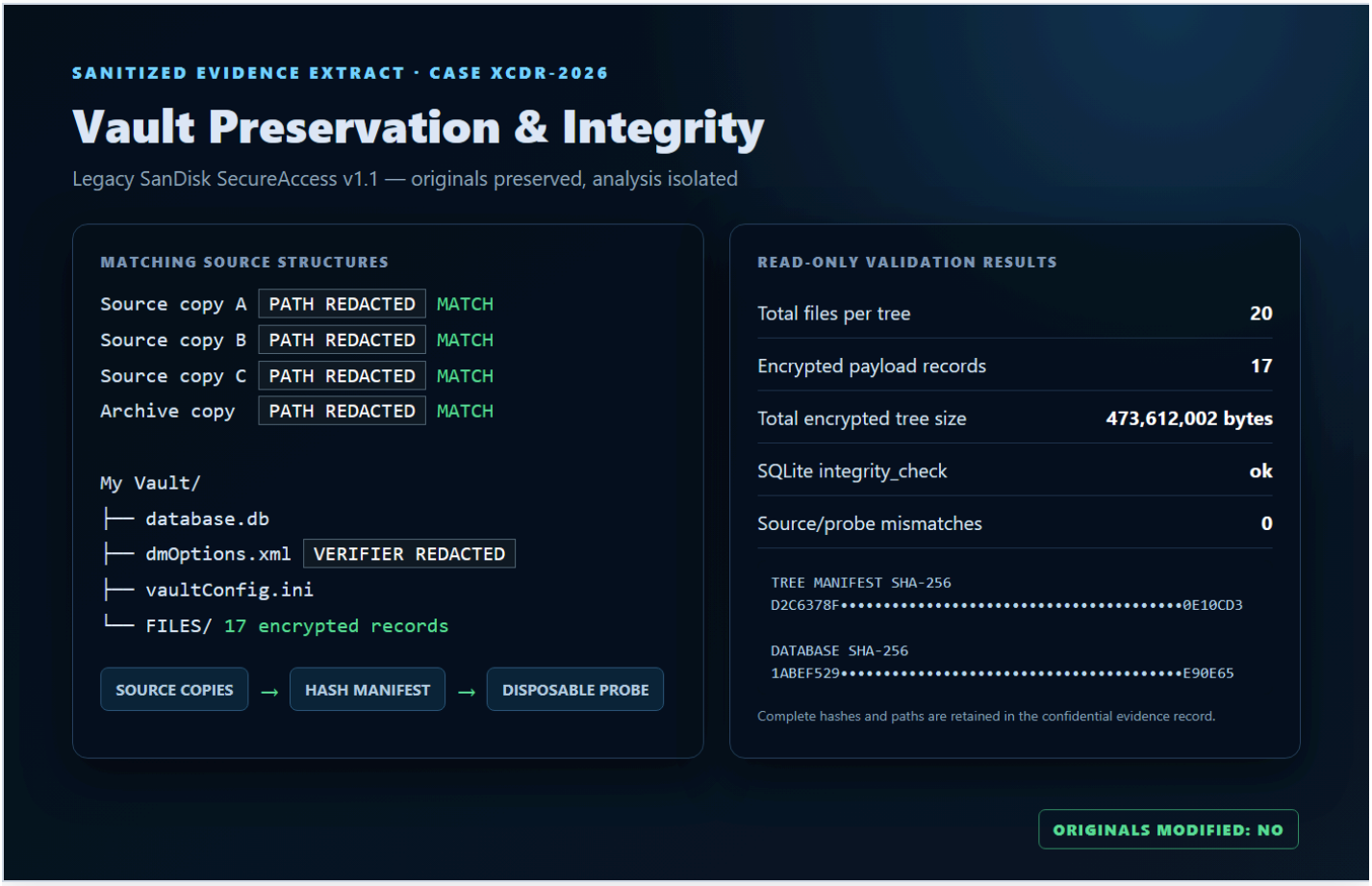

Figure 1. Publication-safe integrity summary. Full paths and complete hashes are retained only in the confidential evidence package.

2. Identifying the correct legacy format

The evidence did not match the newer DataVault layouts covered by most public recovery scripts. Instead, it contained the older SecureAccess v1.1 structure:

- A signed 32-bit vendor engine
- A SQLite vault database
- Legacy XML options and verifier state
- A numbered encrypted FILES hierarchy

This distinction prevented a destructive or unproductive tool mismatch. The discontinued launcher was also unreliable on modern Windows and expected a compatible SanDisk removable-device

environment. A newer SanDisk device was used only to satisfy that host check; it was not treated as source evidence.

3. Reconstructing an offline password check

The signed 32-bit vendor DLL exposed profile and event functions that could still be driven from a 32-bit Windows host. Testing against disposable vaults established two reliable outcomes:

- Incorrect candidate: wrong_password, normally in about 25 milliseconds
- Valid candidate: profile_loaded after the encrypted profile was opened

That native engine path provided a trustworthy final oracle, but roughly 40 candidates per second was too slow for a large personalized list.

Dynamic tracing of disposable test profiles revealed enough of the legacy MD5/AES verification behavior to implement a private OpenCL checker. The live verifier bytes and password are not published.

4. Building a case-specific candidate set

Instead of beginning with a blind keyspace, the recovery process mined old filenames, folder names, and a limited set of small user-authored documents for historically relevant clue tokens. Program files, unrelated data, and the encrypted payloads themselves were excluded.

The resulting statistics were:

- 71,485 filenames/documents examined
- 18,369 clue tokens retained
- 100,000 high-probability candidates
- 2,000,000 expanded candidates

The first 100,000 candidates produced no match. The expanded list reached a successful match at position 1,720,813.

The GPU result was then submitted to the original signed vendor engine. The engine returned profile_loaded, independently confirming the password against the complete legacy implementation.

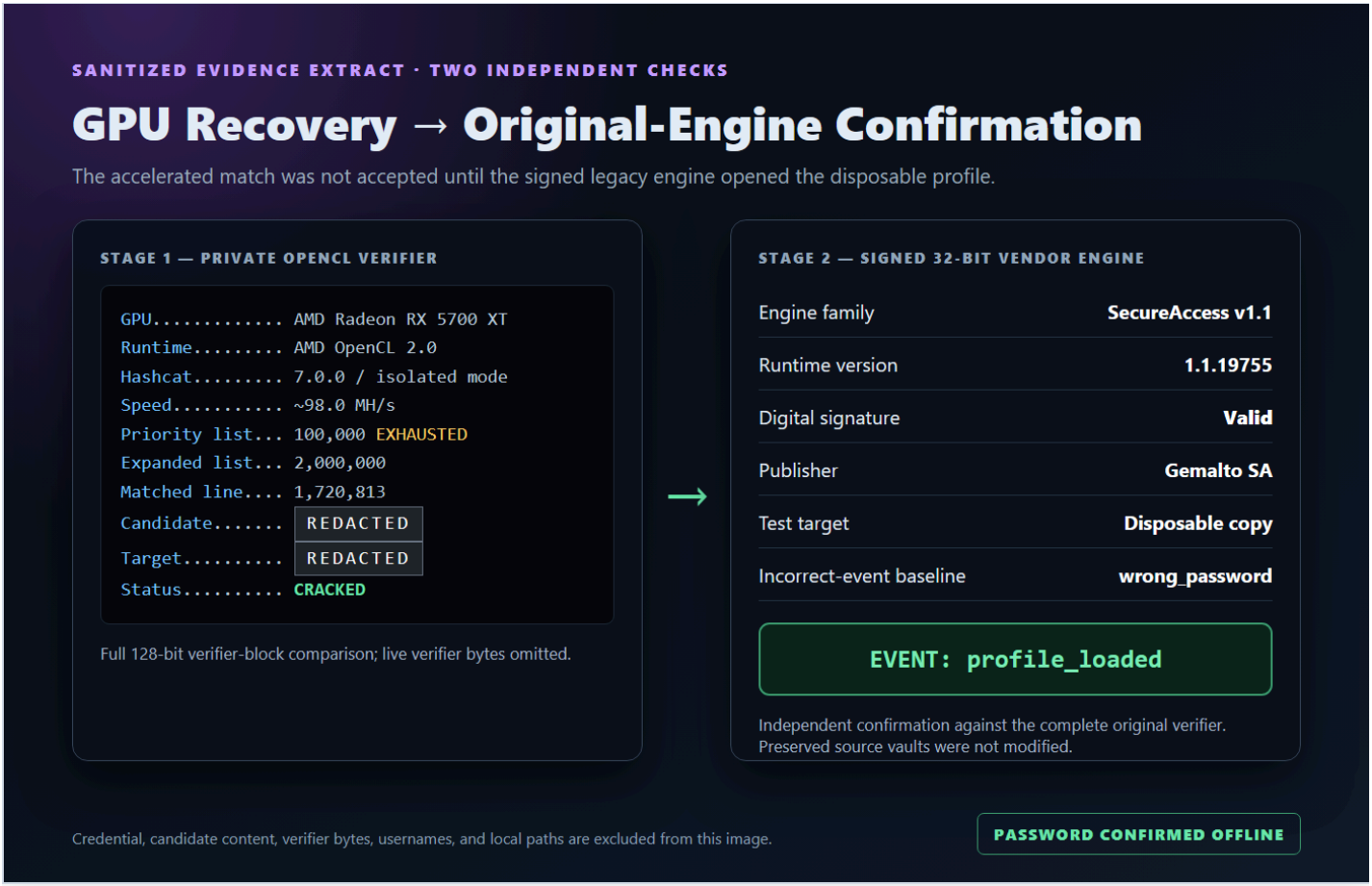

Figure 2. The candidate and verifier are fully blocked. The proof retains the list sizes, GPU result, candidate position, and independent native event.

5. Exporting and preserving the recovered files

After successful authentication, the relevant folder was exported from the disposable recovery environment. It contained:

- 12 encrypted MultiBit HD backups dated September 1–8, 2015
- 2 small recovery-related text artifacts
- 14 files total
- 2,508,353 bytes total

A second verified analysis copy produced the same normalized relative-path, size, and SHA-256 manifest. The exported source set was not edited during analysis.

6. Recovering the MultiBit HD wallet

The export was a nested recovery problem rather than a collection of ordinary documents.

First, twelve valid English BIP39 recovery words were separated from one trailing note token that was not part of the mnemonic. The twelve words passed checksum validation.

Next, the original MultiBit derivation behavior was reproduced offline. The derived wallet identifier matched the identifier encoded in every encrypted backup filename. That was a strong internal consistency check before decryption.

The newest outer backup was decrypted into a ZIP archive and passed CRC validation. An encrypted credential stored inside the backup was then recovered offline and used to decrypt the inner MultiBit wallet protobuf. The mnemonic, wallet credential, decrypted wallet, and private keys never entered a website or the publication package.

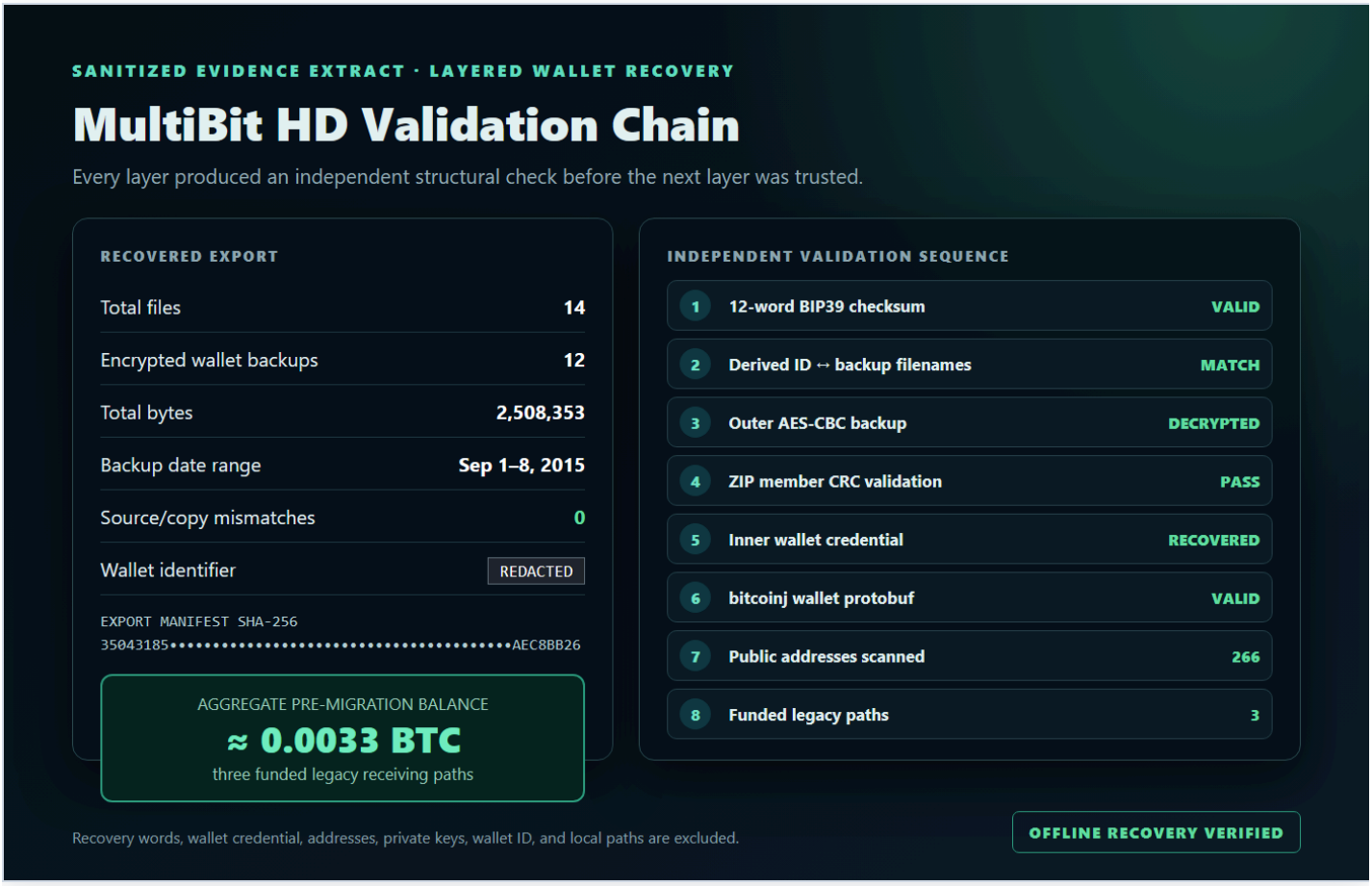

Figure 3. Aggregate validation results from the recovered backup set. Wallet IDs, addresses, and credential material are excluded.

7. The legacy derivation trap

The recovered wallet used receiving paths under m/0'/0. Many current wallet applications default to BIP44 or BIP84 paths. Restoring the correct recovery words into an application that scans only modern defaults could therefore show a false zero balance.

The recovery instead parsed the public keys stored in both MultiBit branches and checked all 266 derived addresses. Three legacy receiving paths had confirmed unspent outputs.

Aggregate balance before migration:

Approximately 0.0033 BTC across three funded addresses

This step converted a successful file decryption into a verified financial recovery.

8. Authenticating the destination

The owner supplied a PayPal “Your Bitcoin address” screen. The destination was checked four ways:

1. Visible text
2. QR payload decoded locally
3. Bitcoin-mainnet Bech32 checksum
4. Final transaction output script

The QR payload and visible text matched exactly, and the destination was not one of the source addresses.

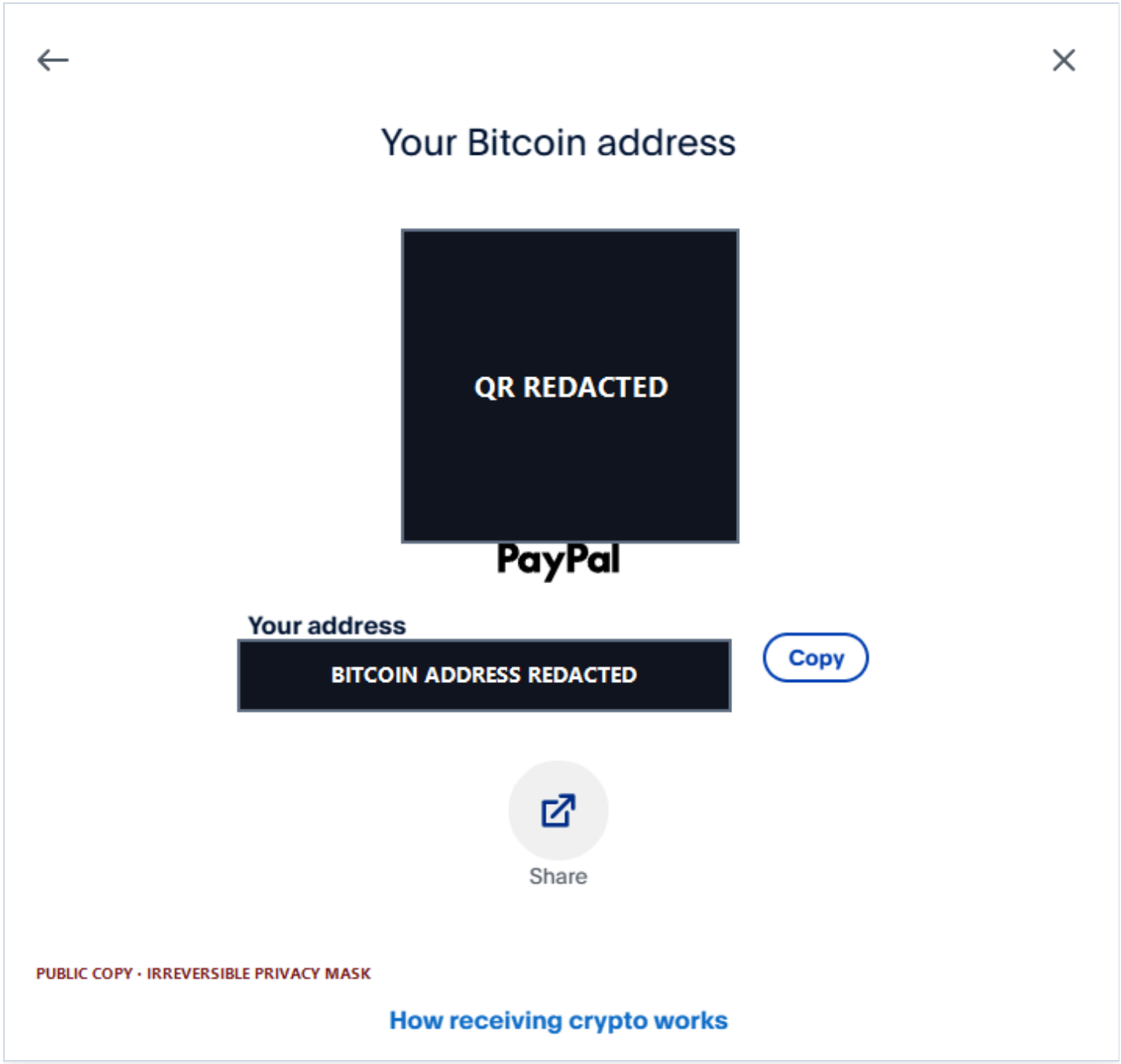

Figure 4. Original owner-provided screenshot, flattened and re-encoded after opaque QR and address redaction. The published image contains no decodable QR payload.

9. Offline signing and independent verification

The migration transaction was constructed as a one-output sweep with no change:

- Three legacy P2PKH inputs
- One native-SegWit destination output

- Version 2
- Replace-by-fee enabled
- A miner fee close to the prevailing network estimate

Only the three funded keys were derived, and signing occurred offline. The recovery phrase was read in memory and was not added to the report.

A separate verifier then:

- Parsed the raw transaction independently
- Confirmed every source outpoint and input value
- Confirmed the sole output script and value
- Rebuilt all three legacy SIGHASH_ALL preimages
- Verified three secp256k1 ECDSA signatures
- Checked low-S encoding
- Recomputed the transaction ID, fee, size, version, sequence, and locktime

The verifier returned PASS before broadcast.

10. One broadcast, independent confirmation

After explicit owner authorization, the transaction was submitted exactly once. Two independent public explorers reported the same transaction and confirmation state.

The published explorer image hides the transaction ID, block hash, exact block/time combination, and repeated identifiers. It retains the explorer identity, confirmed status, transaction size, fee-rate class, version, locktime, and RBF state.

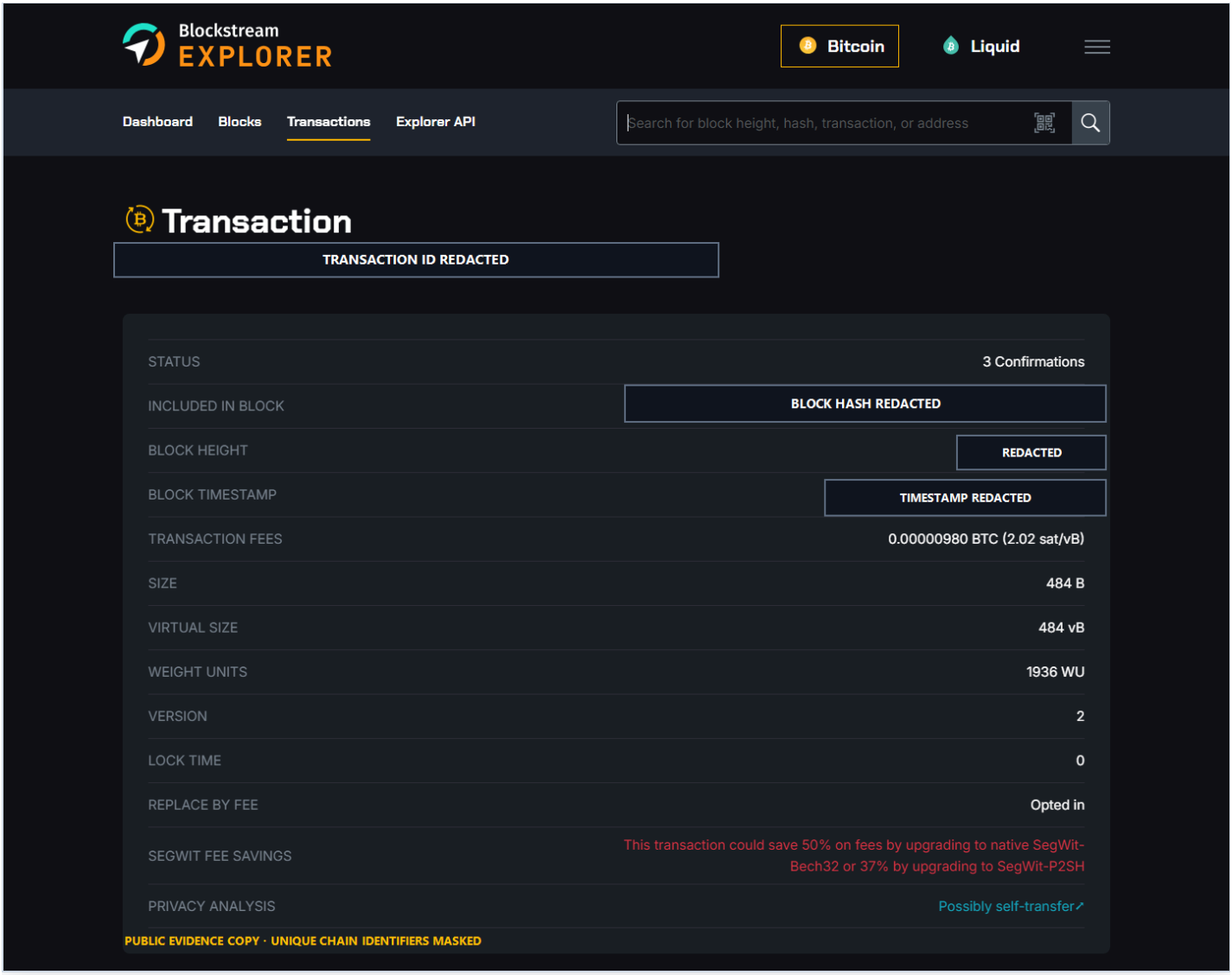

Figure 5. Actual Blockstream explorer capture after confirmation. Unique chain identifiers are irreversibly covered in the publication copy.

11. Why this was an extreme case

No single recovery application solved this case. It required several disciplines to work together:

- Filesystem search and duplicate classification
- Hash-based preservation and manifest comparison
- SQLite integrity validation
- 32-bit Windows interoperability
- Native API analysis and event handling
- Dynamic tracing of a discontinued cryptographic verifier

- Case-derived password modeling
- GPU/OpenCL implementation and validation
- BIP39 and MultiBit-specific key derivation
- AES-CBC and scrypt compatibility work
- Protobuf validation
- Legacy Bitcoin derivation-path analysis
- Public-chain verification
- Offline transaction signing and independent signature auditing

The original device was gone, but enough redundant copied evidence survived to reproduce the application's behavior safely.

12. What made the result defensible

Preservation before experimentation

The source trees were never used as test targets. Disposable copies absorbed every compatibility and password-recovery experiment.

Two independent password checks

The OpenCL result was verified by the original signed engine. A GPU match alone was not treated as final proof.

Multiple internal consistency checks

The recovered words passed BIP39 checksum validation, derived the filename wallet identifier, decrypted a valid ZIP, and unlocked a structurally valid inner wallet.

Exhaustive public-key analysis

All 266 stored public addresses were checked rather than assuming a modern default path.

Separate transaction verifier

The code that signed the transaction was not the only code trusted to validate it.

Public confirmation

The final transfer was independently visible on more than one Bitcoin explorer.

13. Privacy and evidentiary handling

The public package contains no:

- Password or password hash suitable for candidate confirmation
- Seed or recovery words
- Wallet password
- Private keys or WIFs
- Raw signed transaction
- Full wallet ID
- Bitcoin addresses or QR payload
- Transaction ID
- User or host identity
- Full local paths
- Live vault verifier
- Unredacted JSON analysis files

Opaque masking—not blur or pixelation—was used for the two source-derived screenshots. The public images were flattened, re-encoded, stripped of source metadata, and assigned new SHA-256 hashes.

14. Lessons for legacy encrypted-data recovery

1. A missing original device is not always fatal if authenticated copies survive.
2. Identify the exact on-disk generation before choosing tools.
3. Database integrity and payload completeness should be proved before password work.
4. Native vendor behavior can provide an independent oracle even when the GUI fails.
5. Personalized, evidence-derived candidates can outperform enormous generic lists.
6. GPU acceleration is useful only after the verifier is understood and validated.

7. Nested encrypted formats require a validation checkpoint at every layer.
8. Legacy cryptocurrency wallets may use derivation paths that modern defaults miss.
9. A recovered balance should be migrated only after destination, fee, signatures, and serialization are independently checked.
10. Publication evidence should be created from sanitized source material, never by hiding live secrets inside an editable document.

Conclusion

This case began with copied fragments from a missing flash drive and software that no longer functioned reliably. It ended with:

- A byte-consistent legacy vault
- A password recovered and independently verified offline
- A preserved decrypted export
- A structurally validated 2015 MultiBit HD wallet
- Three funded legacy Bitcoin paths
- An offline-signed and independently audited migration
- A confirmed destination transaction

The result demonstrates what extreme-case recovery looks like when preservation, reverse engineering, cryptography, and transaction safety are treated as one continuous evidence process.

No recovery outcome is guaranteed. This is one documented, owner-authorized case.

Public evidence fingerprints

The complete post-redaction SHA-256 manifest is supplied beside this paper as PUBLIC-EVIDENCE-MANIFEST.txt. Displayed hashes in the figures are shortened intentionally; the published files themselves are covered by full hashes in that manifest.