FreelanceShield

Cryptographic Shield Certificate

This document certifies that Alex Designer possessed and submitted the following artifact at 2026-07-10 20:05:42 UTC:

File: branding_concept_v3.fig
SHA-256:
1ff73bb6c992e8577496680c6ac9d65a02ea70e9909babdf48091115daf90a71
Timestamp: 2026-07-10 20:05:42 UTC
Shield ID: adf251ba
Binding: Bilateral (v0x04) - proof of authorship

Scan QR code to verify instantly.

Or visit: verify.aotrust.link/s/adf251ba

Or use the offline verification method on page 2.

How to Verify This Certificate

Option 1: Scan QR code (page 1) with your phone camera.
Option 2: Short link — visit the URL shown on page 1:
`verify.aotrust.link/s/adf251ba`
Option 3: Offline — no internet needed. See algorithm below.

Technical Data

PDR (base64 — copy without spaces or line breaks):


```
BAEFllBRagAAAABub3Rhcnktbm9kZS5uZWFyAAAAAAAAAAAAAAAAAAAAAAAAAABByppIVpjEqEVldeZwXl/6Nxg2hSM+biWlxfChE5Ra
6B/3O7bJkuhXdJZoDGrJ1loC6nDpkJur30gJERXa+QpxY8GwwFeq1zHla5UzfOAUPTAIWx9004CziDZUdDQ7tJit8lG6AAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAESIDph0DGiQR3//F8thpMWlDvghyU93/dMhxw+DWxiXzsHSSC7HBz/re5Jziw5FSzPHKt9kEZWBxgEQ
ZVNqcwM=
```


Binding Hash (payload_hash in PDR):


```
1ff73bb6c992e8577496680c6ac9d65a02ea70e9909babdf48091115daf90a71
```


Notary:

```
notary-node.near
```

Notary public key (Ed25519, hex):


```
490f51f23b993eacaff54fc977d9a7689ab7d4ae91504dc6cbdeadb2dbf1f462
```


Offline Verification Algorithm

This certificate is self-contained. AOTrust is not required for verification. You need Python 3 with PyNaCl (or any Ed25519 library) and hashlib.

Step 1: Hash your file

```
work_hash = hashlib.sha256(file_bytes).hexdigest()
```

Step 2: Decode PDR

```
pdr = base64.b64decode(PDR_BASE64) # 239 bytes
version = pdr[0] # should be 0x04 for bilateral
timestamp = int.from_bytes(pdr[3:11], 'little') # Unix UTC
agent_pubkey_from_pdr = pdr[44:76].hex() # subject_hash
binding_hash_from_pdr = pdr[76:108].hex() # payload_hash
notary_sig = pdr[175:239] # 64 bytes Ed25519
```

Step 3: Verify agent signature (NEP-413)

```
import struct
TAG = 2147484061 # u32 little-endian
envelope = struct.pack('<II', TAG, 32) + bytes.fromhex(work_hash)
# Verify: ed25519.verify(agent_pubkey, envelope, sig_A)
# sig_A is the agent signature from this page
# If valid, the agent signed this specific file hash
```

Step 4: Verify binding hash

```
import hashlib
computed = hashlib.sha256(
bytes.fromhex(work_hash) + base64.b64decode(sig_A) + bytes.fromhex(agent_pubkey)
).hexdigest()
# Must equal binding_hash_from_pdr (payload_hash in PDR)
```

This proves: file hash + agent signature + agent pubkey are bound together

Step 5: Verify notary signature

message = pdr[0:175] # everything except signature

Verify: ed25519.verify(NOTARY_PUBKEY, message, notary_sig)

NOTARY_PUBKEY is shown above on this page

If valid, AOTrust cryptographically signed this PDR

All three checks must pass:

[OK] Agent signature authentic — agent signed this file hash

[OK] Binding hash matches — file, signature, and pubkey are linked

[OK] Notary signature authentic — AOTrust signed this record

=> This file existed at {timestamp}, signed by {agent_pubkey}

Note: Line breaks in the PDR string above are for display only. Use the continuous base64 string without spaces for verification.

This certificate remains valid even if AOTrust, shield.aotrust.link, or verify.aotrust.link cease to exist. The cryptographic proof is permanent.