Package: attestix (via r-universe)

July 5, 2026

Type Package

Title Offline Verifier for Attestix Ed25519 Credentials and UCAN Delegations

Version 0.4.1

Description An offline verifier for verifiable credentials and delegation chains issued by the Attestix Python core. Verifies Ed25519 (RFC 8032) signatures over W3C Verifiable Credentials, decodes Ed25519 'did:key' identifiers, and verifies UCAN delegation chains (EdDSA 'JWT's) including capability attenuation, with no Python runtime required. Reproduces the Attestix JCS-style JSON canonical form (a practical subset of RFC 8785 that additionally applies 'NFC' Unicode normalization) byte-for-byte so that signatures produced by the reference implementation verify here. Useful for compliance, research and biostatistics users who work in R and need to check AI-agent compliance credentials. See <https://attestix.io> for the project and <https://attestix.io/spec/bundle/v1> for the bundle wire format.

License Apache License (>= 2)

URL https://github.com/VibeTensor/attestix-r, https://attestix.io

BugReports https://github.com/VibeTensor/attestix-r/issues

Encoding UTF-8

Depends R (>= 4.0.0)

Imports sodium, openssl, stringi

Suggests jsonlite, testthat (>= 3.0.0)

Config/testthat/edition 3

RoxygenNote 7.3.2

NeedsCompilation no

Config/pak/sysreqs libicu-dev libsodium-dev libssl-dev

Repository https://vibetensor.r-universe.dev

Date/Publication 2026-06-24 16:10:49 UTC

RemoteUrl https://github.com/vibetensor/attestix-r

RemoteRef HEAD

RemoteSha 732dd364a29c7971ab389fe855ee2c07a87ca0ac

Contents

`atx_canonicalize` *Attestix JCS-style canonical JSON*

Description

Reproduces `attestix/auth/crypto.py::canonicalize_json` byte-for-byte.

Usage

```
atx_canonicalize(obj)
```

Arguments

`obj`	A JSON string, raw UTF-8 bytes, or a value tree produced by `atx_json_parse`.

Details

This is **JCS-style, NOT strict RFC 8785**. The two load-bearing divergences from RFC 8785 are:

1. NFC Unicode normalization is applied to every string value and every object key (RFC 8785 does *not* normalize).
2. Whole-number floats collapse to integers (`1.0` -> `1`); non-whole floats use Python's `repr` (the vectors only use integers and `1.5`, on which every port agrees).

Keys are sorted by Unicode code point, separators are `","` and `":"` with no whitespace, output is raw UTF-8 (no `\uXXXX` escapes), and large integers (`> 2^53`) are preserved exactly.

Value

A raw vector of the canonical UTF-8 bytes.

Examples

```
bytes <- atx_canonicalize('{"b":2,"a":1}')
rawToChar(bytes) # {"a":1,"b":2}
```

---

atx_decode_did_key *Decode an Ed25519* did:key *to its raw 32-byte public key*

Description

Reproduces attestix/auth/crypto.py::did_key_to_public_key: strip the did:key:z prefix, base58btc-decode the multibase payload, assert the first two bytes are the 0xed 0x01 multicodec prefix, and return the remaining 32 raw bytes.

Usage

```
atx_decode_did_key(did)
```

Arguments

did	A did:key:z... string.

Value

A raw vector of length 32 (the Ed25519 public key).

Examples

```
did <- "did:key:z6Mko5TBPGKHkCxSgmf3aC6p6SGj2auwCfRmBydXJFEwL4ev"
length(atx_decode_did_key(did)) # 32
```

---

atx_did_key_fragment *Return the multibase fragment portion of a* did:key

---

Description

The verification method is <did>#<multibase> where <multibase> is the z... portion of the did:key. This returns that fragment with a leading #.

Usage

```
atx_did_key_fragment(did)
```

Arguments

did	A did:key:z... string.

Value

The #z... fragment.

---

atx_ed25519_verify *Verify an Ed25519 (RFC 8032) signature*

Description

Thin wrapper over sodium::sig_verify (libsodium). Returns a logical rather than signalling, so callers can fold it into a structured result.

Usage

```
atx_ed25519_verify(message, signature, public_key)
```

Arguments

message	Raw vector of the signed message bytes.
signature	Raw vector, 64 bytes.
public_key	Raw vector, 32 bytes.

Value

TRUE if the signature is valid, else FALSE.

Examples

```
## Not run:
atx_ed25519_verify(msg, sig, pubkey)

## End(Not run)
```

atx_parse_json *Parse JSON into the canonicalisation value tree*

Description

Convenience alias for `atx_json_parse`.

Usage

```
atx_parse_json(txt)
```

Arguments

`txt`	JSON text or raw UTF-8 bytes.

Value

The parsed value tree.

atx_public_key_to_did_key *Encode a raw 32-byte Ed25519 public key to a* `did:key`

Description

Inverse of `atx_decode_did_key`.

Usage

```
atx_public_key_to_did_key(pubkey)
```

Arguments

`pubkey`	A raw vector of length 32.

Value

A `did:key:z...` string.

`atx_verification_method` *Full verification method for a* `did:key` *(*`<did>#<multibase>`*).*

Description

Full verification method for a `did:key` (`<did>#<multibase>`).

Usage

```
atx_verification_method(did)
```

Arguments

`did`	A `did:key:z...` string.

Value

The verification-method string.

`atx_verify_credential` *Verify an Attestix W3C Verifiable Credential offline*

Description

Mirrors `attestix/services/credential_service.py::verify_credential`. The signing payload is the credential with the `proof` and `credentialStatus` top-level keys removed; that payload is JCS-style canonicalized (see `atx_canonicalize`) and the Ed25519 signature in `proof.proofValue` is verified against the issuer public key.

Usage

```
atx_verify_credential(vc, public_key = NULL, now = Sys.time())
```

Arguments

`vc`	The credential as a JSON string, raw UTF-8 bytes, or a value tree from `atx_json_parse`.
`public_key`	The issuer Ed25519 public key. Accepts a raw 32-byte vector, a 64-char hex string, or `NULL` to derive it from the credential's `issuer.id/verificationMethod` did:key.
`now`	The reference time for the expiry check, as a `POSIXct` or ISO-8601 string. Defaults to `Sys.time()`.

Details

Three independent checks are ANDed:

- `signature_valid` - Ed25519 verification of the canonical bytes.
- `not_expired` - `now` < `expirationDate` (tz-aware ISO-8601).
- `not_revoked` - `credentialStatus$revoked` is falsy.

Value

A list with logical fields `signature_valid`, `not_expired`, `not_revoked`, `structure_valid`, and the overall `verify`.

Examples

```
## Not run:
res <- atx_verify_credential(vc_json)
isTRUE(res$verify)

## End(Not run)
```

---

atx_verify_delegation_chain

Verify a UCAN delegation chain (Ed25519 per link + attenuation)

Description

Mirrors the recursive `prf`-chain verification in `attestix/services/delegation_service.py`. Each token in the chain is a PyJWT EdDSA JWT; the signed message is the compact `header.payload` form (base64url **unpadded**), NOT the JCS canonical form.

Usage

```
atx_verify_delegation_chain(
  chain,
  public_key,
  now = Sys.time(),
  revoked_jti = character(0)
)
```

Arguments

`chain`	A list describing the chain. Accepts the conformance-vector shape: `parent_token`, `token` (child), and optionally `parent_att` / `child_att`. Alternatively a list of compact JWT strings ordered root-first.
`public_key`	The Ed25519 server public key (raw 32 bytes or hex).
`now`	Reference time. Defaults to `Sys.time()`.
`revoked_jti`	Optional revoked `jti` character vector.

Details

The chain verifies iff every token's signature is valid AND every token is unexpired and unrevoked AND each child's `att` is a **subset** of its parent's `att` (capability attenuation; escalation is rejected). Cycles (a repeated `jti`) are rejected.

Value

A list with `parent_signature_valid`, `child_signature_valid`, `attenuation_is_subset`, and the overall `verify`.

Examples

```
## Not run:
atx_verify_delegation_chain(list(parent_token = pt, token = ct), pk)

## End(Not run)
```

Index