noiseprotocol

Release 0.3.0

Mar 03, 2020

Contents:

CHAPTER 1

Documentation for the Code

class noise.state.**CipherState**(*noise_protocol*)

Implemented as per Noise Protocol specification - paragraph 5.1.

The initialize_key() function takes additional required argument - noise_protocol.

This class holds an instance of Cipher wrapper. It manages initialisation of underlying cipher function with appropriate key in initialize_key() and rekey() methods.

decrypt_with_ad(*ad: bytes*, *ciphertext: bytes*) → bytes

If k is non-empty returns DECRYPT(k, n++, ad, ciphertext). Otherwise returns ciphertext. If an authentication failure occurs in DECRYPT() then n is not incremented and an error is signaled to the caller.

Parameters

- **ad** – bytes sequence
- **ciphertext** – bytes sequence

Returns plaintext bytes sequence

encrypt_with_ad(*ad: bytes*, *plaintext: bytes*) → bytes

If k is non-empty returns ENCRYPT(k, n++, ad, plaintext). Otherwise returns plaintext.

Parameters

- **ad** – bytes sequence
- **plaintext** – bytes sequence

Returns ciphertext bytes sequence

has_key()

Returns True if self.k is not an instance of Empty

initialize_key(*key*)

Parameters **key** – Key to set within CipherState

class noise.state.**HandshakeState**

Implemented as per Noise Protocol specification - paragraph 5.3.

The initialize() function takes different required argument - noise_protocol, which contains handshake_pattern.

classmethod initialize(*noise_protocol: NoiseProtocol*, *initiator: bool*, *prologue: bytes = b''*, *s: _KeyPair = None*, *e: _KeyPair = None*, *rs: _KeyPair = None*, *re: _KeyPair = None*) → HandshakeState

Constructor method. Comments below are mostly copied from specification. Instead of taking handshake_pattern as an argument, we take full NoiseProtocol object, that way we have access to protocol name and crypto functions

Parameters

- **noise_protocol** – a valid NoiseProtocol instance
- **initiator** – boolean indicating the initiator or responder role
- **prologue** – byte sequence which may be zero-length, or which may contain context information that both parties want to confirm is identical
- **s** – local static key pair
- **e** – local ephemeral key pair
- **rs** – remote party's static public key
- **re** – remote party's ephemeral public key

Returns initialized HandshakeState instance

read_message(*message: Union[bytes, bytearray], payload_buffer: bytearray*)

Comments below are mostly copied from specification.

Parameters

- **message** – byte sequence containing a Noise handshake message
- **payload_buffer** – buffer-like object

Returns None or result of SymmetricState.split() - tuple (CipherState, CipherState)

write_message(*payload: Union[bytes, bytearray], message_buffer: bytearray*)

Comments below are mostly copied from specification.

Parameters

- **payload** – byte sequence which may be zero-length
- **message_buffer** – buffer-like object

Returns None or result of SymmetricState.split() - tuple (CipherState, CipherState)

class noise.state.**SymmetricState**

Implemented as per Noise Protocol specification - paragraph 5.2.

The initialize_symmetric function takes different required argument - noise_protocol, which contains protocol_name.

decrypt_and_hash(*ciphertext: bytes*) → bytes

Sets plaintext = DecryptWithAd(h, ciphertext), calls MixHash(ciphertext), and returns plaintext. Note that if k is empty, the DecryptWithAd() call will set plaintext equal to ciphertext.

Parameters **ciphertext** – bytes sequence

Returns plaintext bytes sequence

encrypt_and_hash(*plaintext: bytes*) → bytes
Sets ciphertext = EncryptWithAd(h, plaintext), calls MixHash(ciphertext), and returns ciphertext. Note that if k is empty, the EncryptWithAd() call will set ciphertext equal to plaintext.

Parameters **plaintext** – bytes sequence

Returns ciphertext bytes sequence

classmethod initialize_symmetric(*noise_protocol: NoiseProtocol*) → SymmetricState
Instead of taking protocol_name as an argument, we take full NoiseProtocol object, that way we have access to protocol name and crypto functions

Comments below are mostly copied from specification.

Parameters **noise_protocol** – a valid NoiseProtocol instance

Returns initialised SymmetricState instance

mix_hash(*data: bytes*)
Sets h = HASH(h + data).

Parameters **data** – bytes sequence

mix_key(*input_key_material: bytes*)

Parameters **input_key_material** –

Returns

split()
Returns a pair of CipherState objects for encrypting/decrypting transport messages.

Returns tuple (CipherState, CipherState)

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

n

Index