Noise protocol framework

Martin Stanek

Department of Computer Science
Comenius University
stanek@dcs.fmph.uniba.sk

Cryptology 1 (2023/24)

Content

Introduction

Components, handshake state, and handshake tokens

Handshake patterns

Security

- payload security properties

Introduction

- Trevor Perrin
- handshake protocols for two participants
  - initiator, responder
  - framework: handshake patterns
- based on DH key exchange
  - static and ephemeral keys
- some instances used real applications
  - WireGuard (VPN), WhatsApp, Lightning Network (Bitcoin/blockchain transactions)

Components

- DH keys (public and private part for DH exchange) – each party has (one or both)
    - long-term static key pair (acceptance is left for an application: certificates, pinning, preconfigured list etc.)
    - ephemeral key pair: always new, never reused
    - instantiation: Curve25519 (X25519), Curve448 (X448)
- symmetric cipher:
    - only AEAD ciphers
    - instantiation: AES-GCM, ChaCha20/Poly1305
- hash function
    - instantiation: SHA-256, SHA-512, BLAKE2s, BLAKE2b

Handshake state

- variables maintained by each party:

  s, **e** local static and ephemeral key pairs (may be empty)

  rs, **re** remote static and ephemeral public keys (may be empty)

  h handshake hash (all data sent and received)

  ck chaining key: hashes all previous DH outputs
  transport encryption keys are derived from **ck**

  k, **n** encryption key and nonce (counter)
  computed whenever **ck** is updated (**n** is reset to 0)
  encrypt static public keys and handshake data
  h is always used as associated data in AEAD

Handshake tokens

- handshake message = tokens + payload
  - payload – data chosen by application, e.g. a certificate
  - payload encrypted using **k** (if non-empty) and **h** is updated
- possible tokens:

e	new ephemeral public key sent in cleartext, **h** is updated
s	static public key sent encrypted (if **k** is set), **h** is updated
ee, es, se, ss	DH is performed with ephemeral/static key pair the first/second letter for initiator's/responders's pair result hashed with old **ck** to derive a new **ck** and **k**
psk	pre-shared symmetric key mixed into **h** and encryption keys

Handshake patterns

- prologue – arbitrary data hashed into **h**
- pre-message pattern
  - information about public keys of the other party
- sequence of message patterns
  - 3 one-way handshake patterns
  - 12 fundamental interactive handshake patterns
- unauthenticated DH:
  - no static key for initiator
  - no static key for reponder

```
NN:
  -> e
  <- e, ee
```

Some other patterns

– both static keys trasmitted

```
XX:
    <- e
    -> e, ee, s, se
    <- s, es
```

– no static key for initiator
– reponder's static key known to initiator

```
NK:
    <- s
    ...
    -> e, es
    <- e, ee
```

Naming convention

N no static key for initiator/responder

K static key known to the other party

X static key transmitted to the other party

I static key for initiator immediately transmitted to responder (reduced or absent identity hiding)

Security of Noise protocols

- handshake pattern validity – set of rules for pattern to be valid, e.g.,
  - no more than one occurrence of `ee`, `es`, `se`, or `ss` per handshake
  - after an `ss` token, the initiator must not send a handshake payload or transport payload unless there has also been an `es` token, etc.
- payload security properties
  - source properties: $0, 1, 2$ (authentication)
  - destination properties: $0, \ldots, 5$ (confidentiality and forward secrecy)
- identity hiding properties:
  - for initiator and for responder on scale $0, \ldots, 9$
  - based on static public keys (not addressing other possible identity leaks through IP addresses, payload, etc.)

Example: IK pattern

```
IK:
    <- s
    ...
    -> e, es, s, ss
    <- e, ee, se
```

- used by WireGuard
- I: static key for initiator immediately transmitted to responder
- K: static public key for responder known to initiator

IK pattern – properties (1)

```
IK:
    <- s
    ...
    -> e, es, s, ss, □      source: 1, destination: 2
    <- e, ee, se
```

- payload security properties
- source: 1 – sender authentication vulnerable to key-compromise impersonation (KCI)
  - when a longterm static private key is compromised
- destination: 2 – encryption to a known recipient, forward secrecy for sender compromise only, vulnerable to replay

IK pattern – properties (2)

```
IK:
    <- s
    ...
    -> e, es, s, ss
    <- e, ee, se, □
```

source: 2, destination: 4

- source: 2 – sender authentication resistant to KCI
- destination: 4 – encryption to a known recipient, weak forward secrecy if the sender’s private key has been compromised

IK pattern – properties (3)

```
IK:
    <- s
    ...
    -> e, es, s, ss
    <- e, ee, se
    -> □                   source: 2, destination: 5
    <- □                   source: 2, destination: 5
```

- source: 2 – sender authentication resistant to KCI
- destination: 5 – encryption to a known recipient, strong forward secrecy