Static-Static Pattern Modifiers for Noise

Trevor Perrin (noise@trevp.net)
Justin Cormack (justin@specialbusservice.com)

Revision 1, 2018-11-18, unofficial/unstable

Contents

1. Introduction

This document introduces the `"ss"` and `"noss"` pattern modifiers which add and remove the `"ss"` token from Noise handshake patterns.

2. Overview

Noise handshakes can use a static-static DH (as represented by the `"ss"` token) for a couple purposes:

- To authenticate the sender of a message to the recipient when the sender hasn't yet received the recipient's ephemeral public key.
- To add additional security in cases where one or both parties have their ephemeral private keys compromised, but their static private keys remain secure.

The fundamental Noise handshake patterns `IK` and `XK` use `"ss"` for the former purpose. The `"noss"` modifier can be applied to these patterns to remove the static-static DH if sender authentication in the first message is not desired.

The `"ss"` modifier can be applied to other patterns for the second purpose (increased security against ephemeral key compromise).

3. The "ss" and "noss" modifiers

The `"ss"` modifier adds the `"ss"` token into a handshake immediately after both `"se"` and `"es"` have been performed. If the pattern has an existing `"ss"` token in a different place it is deleted.

The `"noss"` modifier removes the `"ss"` token.

These modifiers can only be applied to interactive handshake patterns (one-way patterns necessarily rely on `"ss"` more than interactive patterns).

4. Fundamental patterns with "noss"

The following table demonstrates the `"noss"` modifier applied to the `KK` and `IK` fundamental patterns. The original pattern is on the left, and the modified version is on the right.

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


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

Omitting the "ss" substantially changes the security properties for the initial two handshake payloads. The initiator's first payload not longer has any authentication. Less obviously, the responder's first payload has *weak forward secrecy* since anyone could forge the initial ephemeral, then attempt to compromise the initiator's static key at a later date.

Using the notation from the Noise specification, the new security properties are:

```
KKnoss
  -> s
  <- s
  ...
  -> e, es                0               2
  <- e, ee, se            2               3
  ->                      2               5
  <-                      2               5


IKnoss
  <- s
  ...
  -> e, es, s             0               2
  <- e, ee, se            2               3
  ->                      2               5
  <-                      2               5
```

5. Fundamental patterns with “ss”

The below table demonstrates the `"ss"` modifier applied to the relevant fundamental handshake patterns.

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

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

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

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

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

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

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

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

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

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

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

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

---

4. Deferred patterns with “ss”

The below table demonstrates the `"ss"` modifier applied to the relevant deferred handshake patterns.

```
X1K:
  <- s
  ...
  -> e, es
  <- e, ee
  -> s
  <- se
```

```
X1Kss:
  <- s
  ...
  -> e, es
  <- e, ee
  -> s
  <- se, ss
```

```
XK1:
  <- s
  ...
  -> e
  <- e, ee, es
  -> s, se
```

```
XK1ss:
  <- s
  ...
  -> e
  <- e, ee, es
  -> s, se, ss
```

```
X1K1:
  <- s
  ...
  -> e
  <- e, ee, es
  -> s
  <- se
```

```
X1K1ss:
  <- s
  ...
  -> e
  <- e, ee, es
  -> s
  <- se, ss
```

```
X1X:
  -> e
  <- e, ee, s, es
  -> s
  <- se
```

```
X1Xss:
  -> e
  <- e, ee, s, es
  -> s
  <- se, ss
```

```
XX1:
  -> e
  <- e, ee, s
  -> es, s, se
```

```
XX1ss:
  -> e
  <- e, ee, s
  -> es, s, se, ss
```

```
X1X1:
  -> e
  <- e, ee, s
  -> es, s
  <- se
```

```
X1X1ss:
  -> e
  <- e, ee, s
  -> es, s
  <- se, ss
```

```
K1K:
  -> s
  <- s
  ...
  -> e, es
  <- e, ee
  -> se
```

```
K1Kss:
  -> s
  <- s
  ...
  -> e, es
  <- e, ee
  -> se, ss
```

```
KK1:
  -> s
  <- s
  ...
  -> e
  <- e, ee, se, es
```

```
KK1ss:
  -> s
  <- s
  ...
  -> e
  <- e, ee, se, es, ss
```

```
K1K1:
  -> s
  <- s
  ...
  -> e
  <- e, ee, es
  -> se
```

```
K1K1ss:
  -> s
  <- s
  ...
  -> e
  <- e, ee, es
  -> se, ss
```

```
K1X:
  -> s
  ...
  -> e
  <- e, ee, s, es
  -> se
```

```
K1Xss:
  -> s
  ...
  -> e
  <- e, ee, s, es
  -> se, ss
```

```
KX1:
  -> s
  ...
  -> e
  <- e, ee, se, s
  -> es
```

```
KX1ss:
  -> s
  ...
  -> e
  <- e, ee, se, s
  -> es, ss
```

```
K1X1:
  -> s
  ...
  -> e
  <- e, ee, s
  -> se, es
```

```
K1X1ss:
  -> s
  ...
  -> e
  <- e, ee, s
  -> se, es, ss
```

```
I1K:
  <- s
  ...
  -> e, es, s
  <- e, ee
  -> se
```

```
I1Kss:
  <- s
  ...
  -> e, es, s
  <- e, ee
  -> se, ss
```

```
IK1:
  <- s
  ...
  -> e, s
  <- e, ee, se, es
```

```
IK1ss:
  <- s
  ...
  -> e, s
  <- e, ee, se, es,ss
```

```
I1K1:
  <- s
  ...
  -> e, s
  <- e, ee, es
  -> se
```

```
I1K1ss:
  <- s
  ...
  -> e, s
  <- e, ee, es
  -> se, ss
```

```
I1X:
  -> e, s
  <- e, ee, s, es
  -> se
```

```
I1Xss:
  -> e, s
  <- e, ee, s, es
  -> se, ss
```

```
IX1:
  -> e, s
  <- e, ee, se, s
  -> es
```

```
IX1ss:
  -> e, s
  <- e, ee, se, s
  -> es, ss
```

```
I1X1:
  -> e, s
  <- e, ee, s
  -> se, es
```

```
I1X1ss:
  -> e, s
  <- e, ee, s
  -> se, es, ss
```

---

5. Security considerations

The `"noss"` modifier substantially changes the security properties of any encrypted data in the first round-trip between initiator and responder (i.e. the handshake payloads). The `"noss"` modifier should not be used unless you have a clear undersanding of how the security properties change, and why this is acceptable.

The `"ss"` modifier doesn't reduce any security properties of the handshake.

However, it only helps in the case that one or both parties' ephemeral private keys are compromised, while the corresponding static keys remain secure. This is an unusual attack scenario. This scenario might apply in some cases of RNG failure, though note that a severe enough RNG failure might cause repeating ephemerals, which the `"ss"` modifier would not protect against.

6. Rationales

In some of the deferred patterns it would be valid to move the `"ss"` earlier than a deferred authentication DH like `"se"` or `"es"`. However, such an `"ss"` would end up providing a weaker authentication than the DH which was intentionally deferred.

Since the goal of deferred patterns is to defer authentication, it seemed better to preserve that property and add `"ss"` after the authentication DHs.

7. IPR