Additional Symmetric Keys

Mathias hall-Andersen (mathias@hall-andersen.dk)
Second Author (second@email)

Revision 1, 2018-07-09, unofficial/unstable

Contents

1. Introduction

This extension is intended for applications wishing to use key material derived by Noise for auxiliary purposes or other extensions of Noise. Examples include:

- Resumption PSKs - Deriving a "resumption PSK" from an initial Noise session. The resumption PSK can be used to perform a PSK Noise handshake later. ASK can also be used to derive associated data, such as labels to identify PSKs.
- Used to enable 0-RTT with patterns not normally permitting encryption in the first message (e.g. XX pattern), by combining ASK with a psk modifier.
- Deriving keys used to generate random padding or other length-hiding / traffic-hiding countermeasures, include the derivation of obfuscation keys used to discourage middle boxes from parsing fields inside messages.

- Generate application-layer keys in case the Noise handshake is being used as the handshake / key-establishment component within a larger secure transport layer protocol.

2. Overview

This section should give a brief overview of how your extension works.

Introduce new terms in **bold**. Use internal references such as Section 1. Use bibliographic references such as [1], [2], [3] that refer to bibtex entries in either the `spectools/*.bib` files or the local `my.bib` file.

The Additional Symmetric Keys API is centered around labels and chains: **labels** are arbitrary byte strings, each label allows the construction of a **chain** based on the value of the label and the current SymmetricState value. Each chain can then be **invoked** any number of times to generate a arbitrary amount of key material, the API exposed by this extension consists of 3 methods:

- **EnableASK()**: Enables ASK, by setting a boolean `ask_enable = true`, which causes ASK master keys to be derived.
- **InitializeASK(labels)**: If ASK is enabled and an non-empty ASK master key is available, the function makes a set of labels and initializes a chain for each label. This method can be called multiple times, both during and after the handshake and replaces any previous ASK chains when called.
- **GetASK(label)**: Assuming ASK is enabled and initialized, returns the next key from the appropriate chain, and advances the appropriate ASK chain key, deleting the previous chain key.

3. Implementation

4. Security considerations

The Additional Symmetric Keys extension is design to meet the following goals:

- The ASKs should be independent from the Noise chaining key. Recovering knowledge about the Noise chaining key from any number of ASK outputs must be infeasible.
- The ASKs should be mutual independent; deriving any of the ASK outputs from any other should be infeasible.
- ASKs should be capable of serving as collision-resistant hashes of the session transcript at the security level expected by the employed hash function (256/512-bit).

- The mechanism should not be restricted to using the output key-material for particular purposes (e.g. the use cases listed above).
- When not enabled, the ASK mechanism should incur no additional computational cost. This allows libraries to implement the extension without introducing significant overhead on applications not using the ASK API.

6. Rationales

Not required, but might be a good idea to explain nonobvious design decisions.

Rather than truncating to 32 bytes (size of a symmetric key), full output of HKDF is returned (rathe

7. IPR

This document is hereby placed in the public domain.

8. Acknowledgements

This extension is based on the “Resumption PSKs” discussion between:

- str4d (str4d@i2pmail.org)
- Trevor Perrin (trevp@trevp.net)
- Christopher Wood (christopherwood07@gmail.com)
- David Wong (davidwong.crypto@gmail.com)

And in particular the ASK proposal outlined by Trevor Perrin. From which both terminology and implementation details has been lifted into this document.

9. References

[1] H. Krawczyk, “‘Cryptographic extraction and key derivation: The hkdf scheme’.” Cryptology ePrint Archive, Report 2010/264, 2010. http://eprint.iacr.org/2010/264

[2] C. Kudla and K. G. Paterson, “Modular Security Proofs for Key Agreement Protocols,” in Advances in Cryptology - ASIACRYPT 2005: 11th International

Conference on the Theory and Application of Cryptology and Information Security, 2005. http://www.isg.rhul.ac.uk/~kp/ModularProofs.pdf

[3] H. Krawczyk and P. Eronen, “HMAC-based Extract-and-Expand Key Derivation Function (HKDF).” Internet Engineering Task Force; RFC 5869 (Informational); IETF, May-2010. http://www.ietf.org/rfc/rfc5869.txt