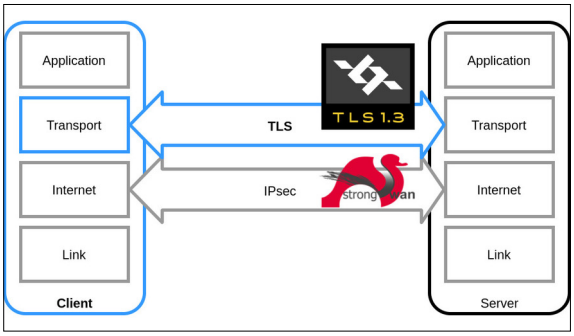

Pascal Knecht


Méline Sieber

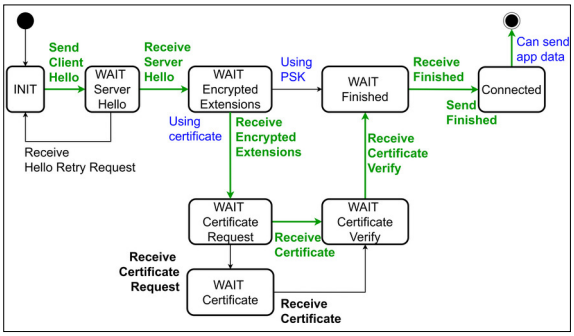
Students	Pascal Knecht, Méline Sieber
Examiner	Prof. Dr. Andreas Steffen
Subject Area	Security

TLS 1.3 for strongSwan

A Client-Side Prototype



In the the context of the OSI model, the TLS library is a layer above strongSwan's main operation mode.
Logos by the IETF TLS Working Group / strongSwan



The state machine of a TLS 1.3 handshake. Green colour indicates our implementation.
Own illustration

Introduction: The Transport Layer Security protocol (TLS) secures network connections between a client and a server. It encrypts and authenticates data from higher-level protocols such as the Hypertext Transfer Protocol (HTTP), and guarantees that the information transmitted remains confidential and unmodified. The most widely used TLS version today is still version 1.2 from 2008 (RFC 5246), even though 1.3 has been released in 2018 (RFC 8446). The strongSwan project, which is maintained by the University of Applied Sciences Rapperswil (HSR), is an open-source IPsec implementation written in C. StrongSwan features its own TLS stack in the library libtls. It enables client-authentication via various EAP authentication methods (TLS, TTLS, PEAP), which is used to establish an IKEv2 connection. However, libtls supports only TLS up to version 1.2.

Objective: This project aims to implement the client-side of TLS 1.3 in libtls and to update the TLS stack. The concrete goal is to successfully perform a minimal handshake and then exchange application data between a strongSwan client and a server running TLS version 1.3. As part of this minimal handshake, it is necessary to integrate new or adapt existing messages that are exchanged between client and server. In addition, TLS 1.3 requires fundamental changes to the cryptographic mechanisms that enable a secure and authenticated encryption.
Until a connection is established, the handshake passes through various states in a state machine. This has considerably changed in the new version, which also implies that the handshake flow and state machine must be adapted too. Our scope includes three optional features: The server-side in a TLS handshake, client-side authentication and remaining non-mandatory extensions.

Result: The updated client implementation in libtls can successfully perform a minimal handshake with a server running TLS 1.3. It has been successfully and extensively tested with external TLS 1.3 servers such as those from Google or Facebook, but also with a local OpenSSL server. During implementation the new cryptographic mechanisms proved to be more difficult that originally anticipated. Especially the HKDF (HMAC-based Extract-and-Expand Key Derivation Function, RFC 5869) was an unexpected major challenge, especially since it is only marginally described in the TLS specification. Due to these difficulties, only the minimal handshake was implemented, the client-side authentication and the server-side was omitted.

For future work, the HKDF implementation needs attention: It is functional, but the code needs to be refactored. The features defined as optional, i.e. the server-side and client-side authentication, were not implemented due to time constraints. Still, they are relevant to the strongSwan project and will hopefully be completed as a Bachelor thesis.