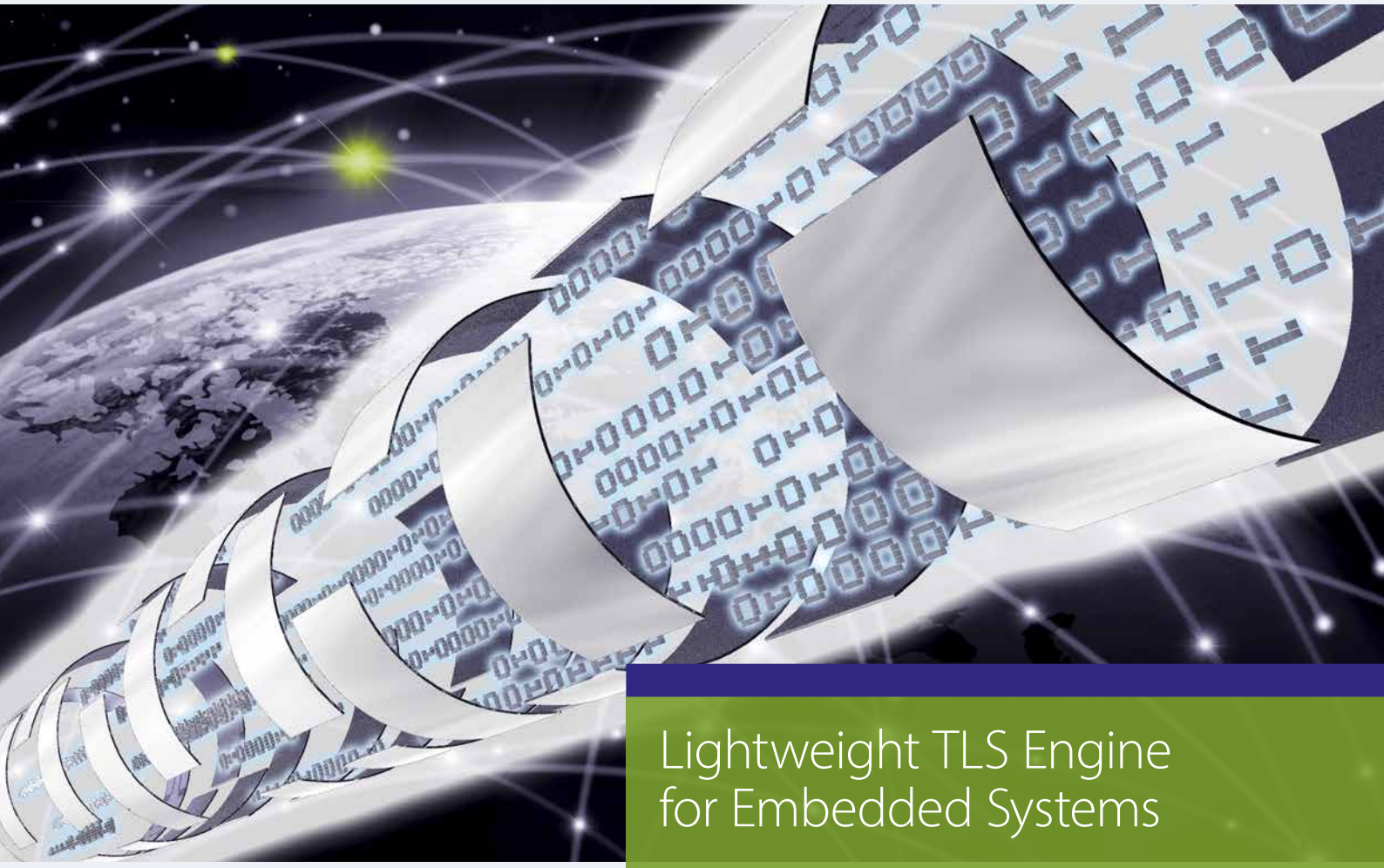
escrypt
SECURITY. TRUST. SUCCESS.

CycurTLS



Lightweight TLS Engine for Embedded Systems

Overview

Transmission of confidential data becomes ever more important in times of growing networks. A well-established solution to secure a connection is the Transport Layer Security (TLS, formerly known as SSL) protocol which provides authentication, integrity and confidentiality. While being widely used to protect web connections via HTTPS, TLS finds its way more and more into the embedded world.

CycurTLS is an efficient implementation of the TLS and DTLS protocols. It is designed particularly for being used in embedded systems which typically have only a low amount of memory available.

CycurTLS is based on ESCRYPT's successful cryptographic library CycurLIB which makes CycurTLS very fast and small. The protocol implementation is very lightweight and does not provide unnecessary optional features such as compression.

CycurTLS is fully compliant to the latest TLS 1.2 and DTLS 1.2 standards and fulfills all security requirements of the IETF.

CycurTLS is highly flexible and provides support for any kind of transportation method. This way CycurTLS is not limited to TCP and UDP, but can also be used with other transport layers.

CycurTLS is compatible to other TLS implementations making it flexible for a multipurpose usage.

CycurTLS is highly configurable to the customer's needs, meaning that only the necessary algorithms are used, while providing the demanded security of your product.

ESCRYPT 2018/05

escrypt
SECURITY. TRUST. SUCCESS.

CycurTLS

Details

General

- Implemented according to IETF and ANSI-C standard
- Compliant to MISRA-C:2012
- Optimized for code size while satisfying stringent performance constraints
- Client and server side
- Easy to integrate in your product
- Intuitive API
- Modular
- Well-documented

Available Cipher Suites

- Elliptic Curve Diffie-Hellman:
  - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
  - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
  - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
  - TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8
  - TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8
  - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
  - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
  - TLS_ECDHE_ECDSA_WITH_ CHACHA20_POLY1305_SHA256
  - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
  - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- RSA key transport:
  - TLS_RSA_WITH_AES_128_CBC_SHA
  - TLS_RSA_WITH_AES_256_CBC_SHA
  - TLS_RSA_WITH_AES_128_CBC_SHA256
  - TLS_RSA_WITH_AES_256_CBC_SHA256
  - TLS_RSA_WITH_AES_128_CCM_8
  - TLS_RSA_WITH_AES_256_CCM_8
  - TLS_RSA_WITH_AES_128_GCM_SHA256
  - TLS_RSA_WITH_AES_256_GCM_SHA384
  - TLS_RSA_WITH_NULL_SHA256
- Pre-shared keys:
  - TLS_PSK_WITH_AES_128_CBC_SHA
  - TLS_PSK_WITH_AES_256_CBC_SHA
  - TLS_PSK_WITH_AES_128_CBC_SHA256
  - TLS_PSK_WITH_AES_128_CCM_8
  - TLS_PSK_WITH_AES_256_CCM_8
  - TLS_PSK_WITH_AES_128_GCM_SHA256
  - TLS_PSK_WITH_AES_256_GCM_SHA384
  - TLS_PSK_WITH_CHACHA20_POLY1305_SHA256
  - TLS_PSK_WITH_NULL_SHA256
  - TLS_PSK_WITH_NULL_SHA384
- Pre-shared keys with Diffie-Hellman:
  - TLS_DHE_PSK_WITH_AES_128_CBC_SHA
  - TLS_DHE_PSK_WITH_AES_128_GCM_SHA256
  - TLS_DHE_PSK_WITH_AES_256_GCM_SHA384
- Callbacks:
  - to externalize PSK computations
  - to externalize private key RSA/ECDSA operations

Supported Protocols

- TLS 1.2
- DTLS 1.2

Supported Extensions

- Server Name Indication (SNI)

Supported Platforms

- Any platform providing an ANSI-C conform compiler - from 8 bit to 64 bit

Features & Benefits

- Seamless integration in existing products
- Implemented to account for highest quality standards
- Low footprint
- Modularity
- Runs on all platforms
- Different licensing models available