IoTDisco: Strong yet Lightweight End-to-End Security for the Internet of Constrained Things

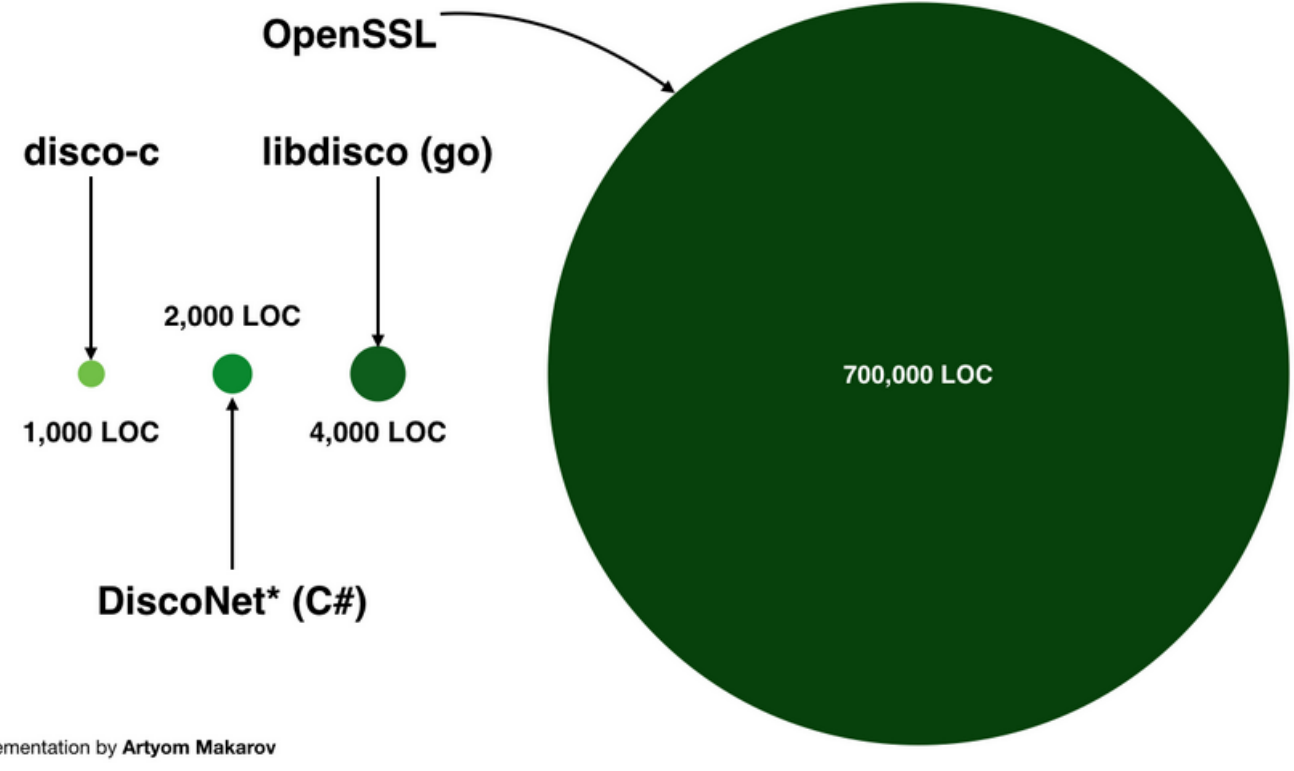
Johann Großschädl
SnT and DCS, University of Luxembourg
Johann.groszschaedl@uni.lu

The research described in this presentation was supported by the Fonds National de la Recherche (FNR) Luxembourg under CORE grant C19/IS/13641232

Outline

- Disco = Noise + Strobe
  - Framework for designing custom protocols
  - Clean slate approach to E2E security
- Efficient End-to-End security for the IoT
  - Simplicity, low run-time complexity
  - Only two low-level cryptographic primitives
- Prototype implementation (Noise-NK)
  - 16-bit MSP430 microcontroller, 8kB RAM
  - Curve25519 and Xoodoo permutation in Assembly
- Results and comparison

Disco vs TLS (OpenSSL)



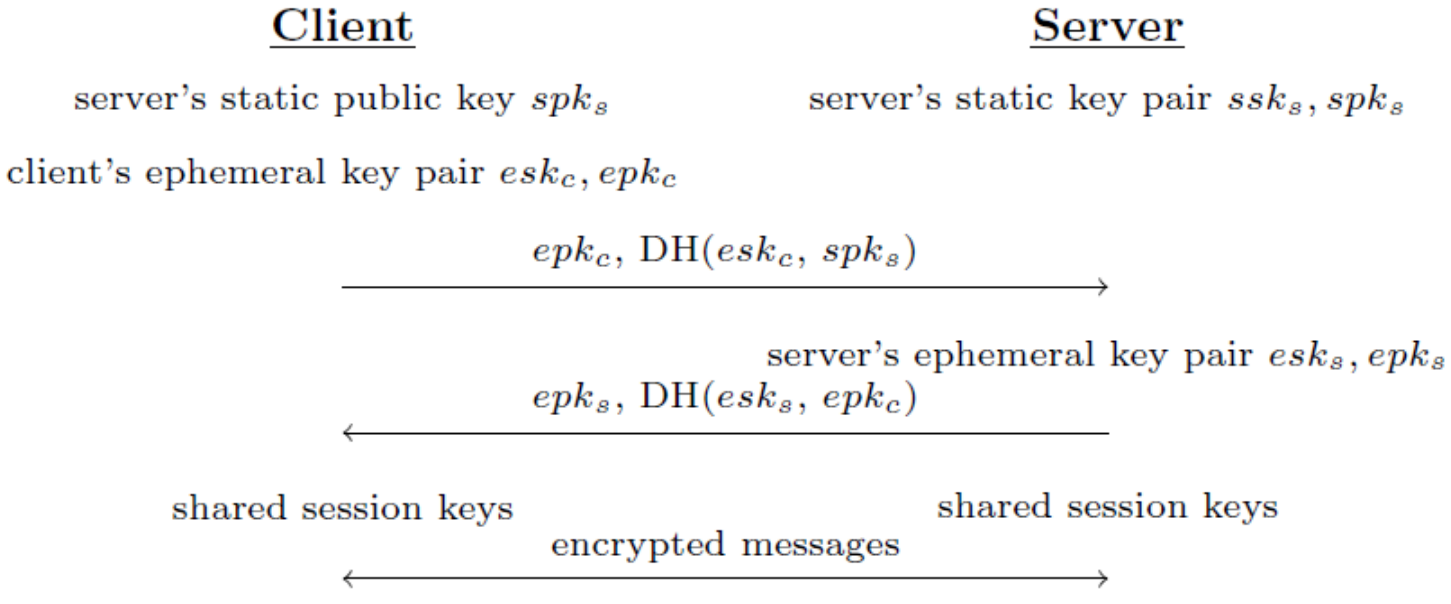
E2E Security with 1000 LoC?

- Noise: Authenticated Key Exchange (AKE)
  - Avoids run-time decisions, i.e., negotiations
  - Instead: design-time decisions (who is authenticated to whom, which cryptographic algorithms are used, etc.)
  - Simple negotiation phase at the beginning, thereafter straight sequence of messages/operations, no branches
- Strobe: Secure Transport (Symmetric Crypto)
  - Based on one single primitive: a permutation
  - Authenticated encryption for transport of application data
  - Simplifies symmetric crypto used during handshake

Some Properties of Noise

- Authentication through DH operations
  - Long-term keys are ECDH keys (like in NAXOS, ECMQV)
  - Signatures may still be needed for binding key to identity
- Pre-defined handshake patters
  - Well-analyzed and formally-proven security properties
  - Patterns named after status of long-term keys: N, K, X, I
  - Client and server follow a linear path that never branches
- Early payloads (during handshake)
  - Encrypted if at least one DH operation has been carried out
  - Less secure than transport payloads after handshake

Example: Noise-NK Handshake



N = No static key for initiator

K = Static key for responder Known to initiator (e.g., pre-deployed)

https://noiseexplorer.com/patterns/NK/

Some Properties of Strobe

- Symmetric crypto during handshake
  - Authenticated encryption of early payloads (generation of keys, nonces, etc.), hashing of protocol transcripts, etc.
  - Noise uses e.g. AES-GCM and SHA-256
  - Much simpler with a permutation (sponge, duplex)
  - Stateful object: cryptographic output of any step depends not only on direct input, but also on all preceding inputs
- Symmetric crypto after handshake
  - Two cipher states (one for each direction)
  - Key derivation, authenticated encryption (AEAD)

MSP430 Microcontroller

- 16-bit architecture for IoT
  - von-Neumann memory model
  - CISC-like memory-to-memory instructions
  - 16 registers (12 usable for programmer)
  - Only 27 core instructions
  - 7 different addressing modes
  - Ultra low power consumption
- Zolertia Z1 device
  - 16-bit MSP430F2617 @ 8 MHz
  - 8 kB of RAM, 92 kB flash memory

Implementation and Evaluation

- Starting point: EmbeddedDisco
  - Developed by David Wong: https://embeddeddisco.com
  - Roughly 2000 lines of C code (Noise-NK, including crypto)
  - Uses TweetNaCl for Curve25519 (slow!)
- Curve25519 and Xoodoo
  - Field-arithmetic implemented in MSP430 Assembly
  - Optimized fixed-base and variable-base scalar multiplication
  - Xoodoo permutation is Assembly, duplex mode in C
- Evaluation of computational part of protocol
  - Without sending messages between devices

IoTDisco Primitives for NK Handshake

Table 4. Execution time (in clock cycles) of Disco primitives in an Noise NK handshake and the subsequent symmetric processing (i.e., Strobe) of EmbeddedDisco and IoTDisco.

Primitives	EmbeddedDisco (C impl.)	IoTDisco (ASM impl.)
Initialization		
`disco_Initialize()`	583,052	55,699
client → server handshake (Noise)		
`disco_WriteMessage()`	443,604,514	15,218,721
`disco_ReadMessage()`	222,379,982	10,825,192
server → client handshake (Noise)		
`disco_WriteMessage()`	444,768,307	15,299,704
`disco_ReadMessage()`	223,543,472	10,903,793
client ↔ server symmetric processing (Strobe)		
`disco_EncryptInPlace()`†	1,158,706	75,774
`disco_DecryptInPlace()`†	1,158,677	75,745

Results for Full IoTDisco Protocol

Table 5. Execution time, RAM usage, and code size of an Noise NK handshake computation of EmbeddedDisco and IoTDisco on MSP430X microcontrollers (at 8 MHz).

Reference	Impl.	Side†	Execution time (cycles)	RAM‡ (bytes)	Size (bytes)
EmbeddedDisco	C	client	667,731,038	3366	8911
		server	667,731,341	3366	8911
IoTDisco	ASM	client	26,178,213	1382	11,602
		server	26,180,595	1382	11,602

† We here consider the key-establishment process using Noise NK pattern. Each of client and server side is performing a disco_Initialize, a disco_WriteMessage, and a disco_ReadMessage operations to obtain a shared session key.

‡ The RAM usage includes two 128-byte buffers (used for respectively sending and receiving the handshake messages) for each of client and server side.

Comparison with Related Work

Table 6. The comparison of handshake computation of different end-to-end security protocols for IoT devices.

Protocols	Sec. (bits)	Devices	Side	Execution time (cycles)	RAM (bytes)	Size (bytes)
HIP DEX [23]	112	32-bit ARM9 @ 180 MHz	client	192,960,000	n/a	n/a
			server	192,960,000	n/a	n/a
μEDHOC [18]	128	32-bit Cortex-M0 @ 16 MHz	client	274,816,000	2381	18,950
			server	274,832,000	2624	18,950
IoTDisco (This work)	128	16-bit MSP430X @ 8 MHz	client	26,178,213	1382	11,602
			server	26,180,595	1382	11,602

18. S. Hristozov, M. Huber, L. Xu, J. Fietz, M. Liess, and G. Sigl. The cost of OSCORE and EDHOC for constrained devices. In *ACM Conference on Data and Application Security and Privacy 2021*, pages 245–250. ACM, 2021.
23. P. Nie, J. Vähä-Herttua, T. Aura, and A. V. Gurtov. Performance analysis of HIP diet exchange for WSN security establishment. In *7th ACM Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet 2011)*, pages 51–56. ACM, 2011.

Conclusions

- IoTDisco prototype
  - Computational part of Noise-NK in 26M cycles on MSP430
  - E2E security with 1.4 kB RAM and 11.6 kB code size
- Ongoing and future work
  - Full implementation (including communication)
  - Post-quantum version using PQNoise (KEMs instead of DH)
- Resources
  - Disco protocol: https://www.discocrypto.com
  - IoTDisco: https://orbilu.uni.lu/handle/10993/59880
  - Post-quantum Noise: https://eprint.iacr.org/2022/539