# Hardware Secured, Password-based Authentication for Smart Sensors for the Industrial Internet of Things

**Institute of Technical Informatics**

22.08.17

**Thomas Wolfgang Pieber**, Thomas Ulz, Christian Steger, Rainer Matischek

# Outline

- Introduction
- Design
- Evaluation
- Results
- Conclusion
- Future Work

# Outline

- Introduction
- Design
- Evaluation
- Results
- Conclusion
- Future Work

# Introduction

- Sensors are a key component for the Internet of Things.
- Decisions depend on the information sensors provide.
- Smart sensors can manipulate the gathered data and make decisions.
- To update the sensors settings, the operator needs to be trusted.
- →Authentication on sensors is vital to trust the sensor data and the connected system.

# Improvement

- Implementing fully functional versions of SPAKE2.
- Deploying SPAKE2 on a Hardware Security Module, and a PC.
- Evaluate the performance of SPAKE2 compared to current authentication mechanisms.
- Evaluate the time performance in absolute numbers.

# Outline

- Introduction
- Design
- Evaluation
- Results
- Conclusion
- Future Work

# Design

$$\text{public information: } G, H(\cdot), u_A, u_B$$

$$\text{private shared information: } password$$

| User A | User B |
|---|---|
| $x = rand()$ | $y = rand()$ |
| $X = xG;\ M = H(u_A)G$ | $Y = yG;\ N = H(u_B)G$ |
| $X^* = X + (password)M$ | $Y^* = Y + (password)N$ |

$$X^* \rightarrow$$

$$\leftarrow Y^*$$

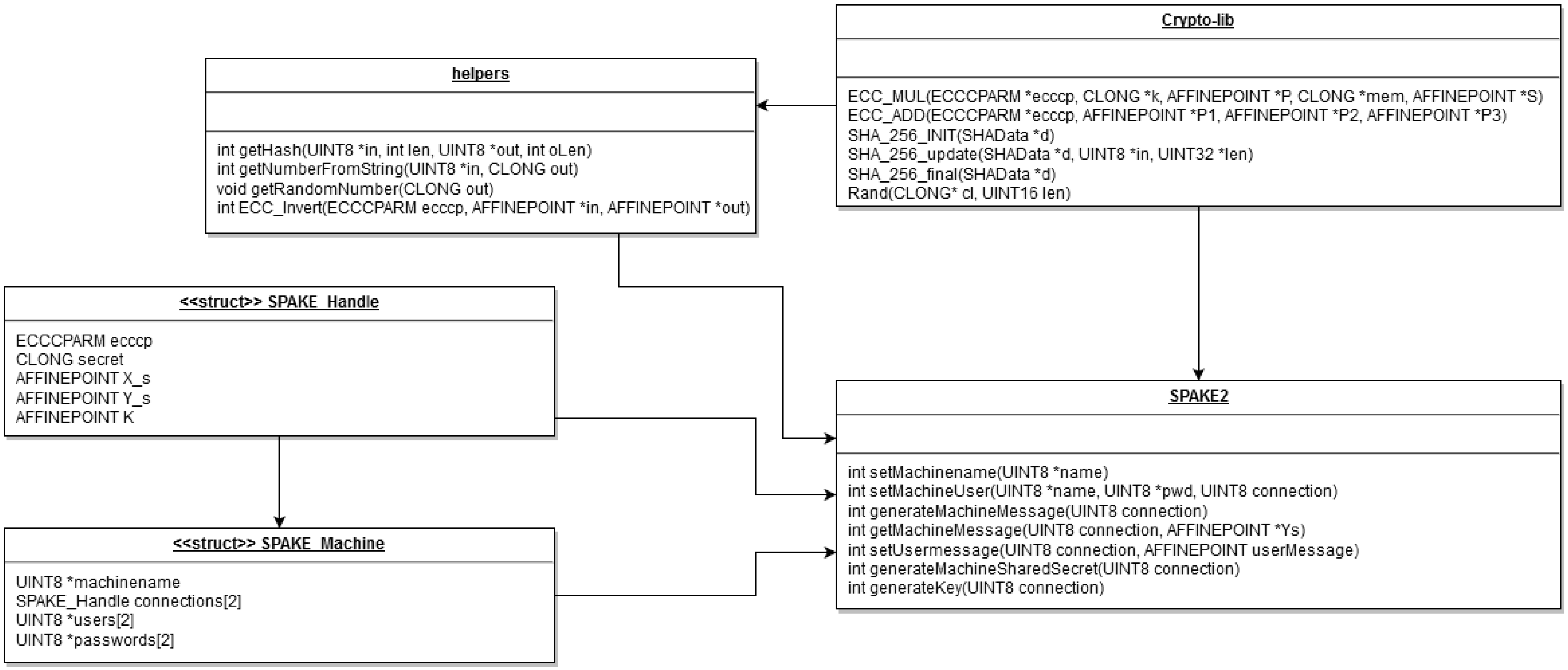
| | |
|---|---|
| $N = H(u_B)G$ | $M = H(u_A)G$ |
| $K_A = x(Y^* + Inv((password)N))$ | $K_B = y(X^* + Inv((password)M))$ |

$$sk_A = H(H(u_A), H(u_B), X^*, Y^*, password, K_A)$$

$$sk_B = H(H(u_A), H(u_B), X^*, Y^*, password, K_B)$$

# Design for the HSM

- Cryptographic primitives performed by Hardware



# Low Memory vs. Low Coputation Time

- Calculate masks when changing credentials
  →only compute once, but use memory to save it

Memory saving Method

| **SPAKE_Credentials** |
| --- |
| UINT8 *machinename<br>UINT8 *username<br>UINT8 *password |

Computation time saving Method

| **SPAKE_Credentials** |
| --- |
| UINT8 H_machinename [32]<br>UINT8 Mx [32]<br>UINT8 My [32]<br>UINT8 Nx [32]<br>UINT8 Ny [32]<br>UINT8 H_password [32]<br>UINT8 H_username [32] |

# Outline

- Introduction
- Design
- Evaluation
- Results
- Conclusion
- Future Work

# Evaluation

- Current SPAKE implementation of the uWeave project [1] does not allow for testing against it.
  - It uses fixed M and N and does not correctly compute the final key. Additionally only a 224 bit implementation is available.
- →Evaluating performance of SPAKE2 on a HSM

[1] https://github.com/mahmed8003/esp8266-weave-test/blob/master/src/libuweave/src/crypto_spake.c

# Time and Memory Performance

- The communication speed is measured by sending packets with different sized payloads that get returned.
- The different implementations get compared in terms of time and needed cryptographic operations.
- The different implementations get compared in terms of required permanent memory.
- As baseline a password authentication scheme in combination with ECDH is chosen.

# Outline

- Introduction
- Design
- Evaluation
- Results
- Conclusion
- Future Work

# Results

- The implementation with low computation time is approx. 309 ms faster in normal operation.
- The time savings are generated by performing less operations and performing them early.

| Operation (low-mem) | Mean [$\mu s$] | Sigma [$\mu s$] |
|---|---|---|
| unauthSend(64b) | 8023 | 45 |
| sendKey | 8295 | 61 |
| generatePubKey | 218813 | 205 |
| calculateSharedSecret | 203548 | 142 |
| calculateKey | 42381 | 68 |

| Operation (low-comp) | Mean [$\mu s$] | Sigma [$\mu s$] |
|---|---|---|
| generatePubKey | 74077 | 67 |
| calculateSharedSecret | 67858 | 125 |
| calculateKey | 39554 | 49 |
| initUser | 91002 | 28 |
| initMachine | 12453 + 43542/User | 18 |
| changeMachine | 273919 | 170 |

# Operations

- The Low Computation Time implementation performs 4 of 6 multiplications at initialization and only 1 of 5 Hash computations during the authentication.
- Furthermore, 2 Hash computations are dropped. These are the recomputation of Hashes wehn calculating the inverse mask.

| | SPAKE-low-mem | SPAKE-low-comp |
|---|---|---|
| Crypto-operations ($*; +; H(\ldots); rand()$) | 6; 2; 5; 1 | 2; 2; 1; 1 |
| initialization crypto-operations | 0; 0; 0; 0 | 4; 0; 2; 0 |
| Time authentication [ms] | 426 | 145 |
| Time initialization [ms] | 0 | ~100 |

# Memory

- The Low Memory implementation needs permanent memory for storing the credentials of users (username and password) and HSM (name).
- In the other implementation the credentials are saved hashed and additional 2 EC-Points (masks) have to be saved per user and the name of the HSM is saved hashed.

| | SPAKE-low-mem | SPAKE-low-comp |
|---|---|---|
| Permanent Memory / User | credentials | 2 Hashes + 2 Points |
| Permanent Memory SE | credentials | 1 Hash |

# Comparison

- For comparison, the authentication mechanism of Lee and Hwang [2] in combination with ECDH needs one EC-multiplication less and one Hash and random number more. This is comparable to the Low Memory implementation.

| | ECDH | [2] | SPAKE-low-mem | SPAKE-low-comp |
|---|---|---|---|---|
| Crypto-operations $(*;+;H(\ldots);rand())$ | 2; 0; 1; 1 | 3; 2; 6; 1 | 6; 2; 5; 1 | 2; 2; 1; 1 |
| initialization crypto-operations | 0; 0; 0; 0 | 3; 2; 0; 0 | 0; 0; 0; 0 | 4; 0; 2; 0 |
| Time authentication [ms] | 120 | 279 + ECDH | 426 | 145 |
| Time initialization [ms] | 0 | 203 | 0 | ~100 |
| Permanent Memory / User | 0 | credentials | credentials | 2 Hashes + 2 Points |
| Permanent Memory SE | 0 | credentials | credentials | 1 Hash |

[2] Liao, I. E., Lee, C. C., & Hwang, M. S. (2006). A password authentication scheme over insecure networks. *Journal of Computer and System Sciences*, *72*(4), 727-740.

# Outline

- Introduction
- Design
- Evaluation
- Results
- Conclusion
- Future Work

# Cunclusion

- The SPAKE2 protocol can be implemented very efficiently to fit on a very restricted device.
- It is advantageous compared to standard solutions that perform key exchange and authentication separately.

# Outline

- Introduction
- Design
- Evaluation
- Results
- Conclusion
- Future Work

# Future Work

- Restructure the communication between devices to minimize communication overhead.
- Change protocol to enable authentication between machines.

# Acknowledgement

This project has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 692480. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Netherlands, Spain, Austria, Belgium, Slovakia.

IoSense is funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program "ICT of the Future" between May 2016 and May 2019. More information https://iktderzukunft.at/en/

# Thank you for your attention!