MPSE - SS2022

Implementing eID protocols

Chiara-Marie Zok
Daniel Christopher Römer
Martyna Orszula
Patrick Moos
Sebastian Schuch

Agenda

1. Introduction
2. Where we began
3. Goals
4. Our Work
5. Future Work
6. Conclusion

4.1. Theory

4.2. Code

4.3. Board

4.4. Problems

1 Introduction

- PIN based authentication might be threatened by quantum computers
- Find a scheme without any connection between PIN and encryption key
- Only use quantum safe components
- Implement POC

2 Where we began

Decisions:

- CRYSTALS - Kyber as PQC scheme
- Which boards to use

Implementations:

- PACE and Kyber on Linux (Mint)
- Kyber on board

Proposals:

- Idea to incorporate nonce

3 Goals

First goal:

- Develop a working prototype

Second goal:

- Make it run on a development board

Side goals:

- Fix issues left over from the previous semester

4 Our Work

4.1 Theory

Exchange of PACE protocol

- New Kyber-Ding-PACE based on Ding PAKE
- Solves nonce problem

Additionally

- Human readable run on paper
- Look into attacks

4.1 Kyber Ding PACE

Alice		**Bob**
password π		password π
	Exchange nonce	
$K_\pi = \mathcal{H}(\pi\|\|0)$ choose $n \leftarrow \mathbb{Z}_q$ $z = \mathcal{C}(K_\pi, n)$ generate $\mathbf{A} \in \mathbb{R}^{k\times k}_{q=3329}$		$K_\pi = \mathcal{H}(\pi\|\|0)$
	$\xrightarrow{\mathbf{A}, z}$	
		$n = C^{-1}(K_\pi, z)$
	Mapping	
generate $\mathbf{s}_a, \mathbf{e}_a \in \mathbb{R}^k_{\eta=3}$ $\mathbf{t}_a = \mathbf{A}\mathbf{s}_a + \mathbf{e}_a$		generate $\mathbf{s}_b, \mathbf{e}_b \in \mathbb{R}^k_{\eta=3}$ $\mathbf{t}_b = \mathbf{A}\mathbf{s}_b + \mathbf{e}_b$
	$\xleftarrow{\mathbf{t}_b}$ with hash $\mathbf{H}$: $\{0,1\}^{256} \to \mathbb{R}^k_{q=3329}$	
$\mathbf{p}_a = \mathbf{t}_a + \mathbf{H}(n)$	$\xrightarrow{\mathbf{p}_a}$	$\mathbf{t}_a = \mathbf{p}_a - \mathbf{H}(n)$
	two-path preKeys	
generate $m_a \leftarrow \{0,1\}^{256}$ $(\hat{K}_a, (\mathbf{r}_a, \mathbf{e}_{a1}, e_{a2})) = \mathbf{G}(\mathbf{H}(\mathbf{t}_b), m_a)$		generate $m_b \leftarrow \{0,1\}^{256}$ $(\hat{K}_b, (\mathbf{r}_b, \mathbf{e}_{b1}, e_{b2})) = \mathbf{G}(\mathbf{H}(\mathbf{t}_a), m_b)$
	with hash $\mathbf{G}$: $\mathbb{R}^k_{q=3329}, \{0,1\}^{256} \to$ $\{0,1\}^{256}, (\mathbb{R}^k_{\eta=3}, \mathbb{R}^k_{\eta=3}, \mathbb{R}^k_{\eta=3})$	
$c_a = \begin{cases} \mathbf{u}_a = (\mathbf{A}^T\mathbf{r}_a + \mathbf{e}_{a1}) \\ v_a = \mathbf{t}_b^T\mathbf{r}_a + e_{a2} + \lceil \frac{q}{2} \rfloor \cdot m_a \end{cases}$		$c_b = \begin{cases} \mathbf{u}_b = (\mathbf{A}^T\mathbf{r}_b + \mathbf{e}_{b1}) \\ v_b = \mathbf{t}_a^T\mathbf{r}_b + e_{b2} + \lceil \frac{q}{2} \rfloor \cdot m_b \end{cases}$
	$\xrightarrow{c_a=(\mathbf{u}_a, v_a)}$ $\xleftarrow{c_b=(\mathbf{u}_b, v_b)}$	
$m_b^* = (v_b - \mathbf{s}_a^T\mathbf{u}_b)$ $\hat{K}_b^*, (\mathbf{r}_b^*, \mathbf{e}_{b1}^*, e_{b2}^*)) = \mathbf{G}(\mathbf{H}(\mathbf{t}_a), m_b^*)$ $c_b^* = \begin{cases} \mathbf{u}_b^* = (\mathbf{A}^T\mathbf{r}_b^* + \mathbf{e}_{b1}^*) \\ v_b^* = \mathbf{t}_a^T\mathbf{r}_b^* + e_{b2}^* + \lceil \frac{q}{2} \rfloor \cdot m_b^* \end{cases}$ generate $z_a \leftarrow \{0,1\}^{256}$ $\hat{K}_b = \begin{cases} \hat{K}_b^* \text{ if } c_b = c_b^* \\ z_a \text{ if } c_b \neq c_b^* \end{cases}$		$m_a^* = (v_a - \mathbf{s}_b^T\mathbf{u}_a)$ $\hat{K}_a^*, (\mathbf{r}_a^*, \mathbf{e}_{a1}^*, e_{a2}^*)) = \mathbf{G}(\mathbf{H}(\mathbf{t}_b), m_a^*)$ $c_a^* = \begin{cases} \mathbf{u}_a^* = (\mathbf{A}^T\mathbf{r}_a^* + \mathbf{e}_{a1}^*) \\ v_a^* = \mathbf{t}_b^T\mathbf{r}_a^* + e_{a2}^* + \lceil \frac{q}{2} \rfloor \cdot m_a^* \end{cases}$ generate $z_b \leftarrow \{0,1\}^{256}$ $\hat{K}_a = \begin{cases} \hat{K}_a^* \text{ if } c_a = c_a^* \\ z_b \text{ if } c_a \neq c_a^* \end{cases}$
	Authentication	
$K = \mathbf{KDF}(\hat{K}_a, \hat{K}_b)$ $K_{enc} = \mathcal{H}(K\|\|1)$ $K_{mac} = \mathcal{H}(K\|\|2)$ $K'_{mac} = \mathcal{H}(K\|\|3)$ $T_A \leftarrow \mathcal{M}(K'_{mac}, (\mathbf{t}_b, \mathbf{A}))$		$K = \mathbf{KDF}(\hat{K}_a, \hat{K}_b)$ $K_{enc} = \mathcal{H}(K\|\|1)$ $K_{mac} = \mathcal{H}(K\|\|2)$ $K'_{mac} = \mathcal{H}(K\|\|3)$ $T_B \leftarrow \mathcal{M}(K'_{mac}, (\mathbf{t}_a, \mathbf{A}))$
	$\xrightarrow{T_A}$ $\xleftarrow{T_B}$	
abort if T_B invalid		abort if T_A invalid
	Establish Session	
key $= (K_{enc}, K_{mac})$ sid $= (\mathbf{t}_a, \mathbf{t}_b, \mathbf{A})$ pid $= \epsilon$		key $= (K_{enc}, K_{mac})$ sid $= (\mathbf{t}_a, \mathbf{t}_b, \mathbf{A})$ pid $= \epsilon$

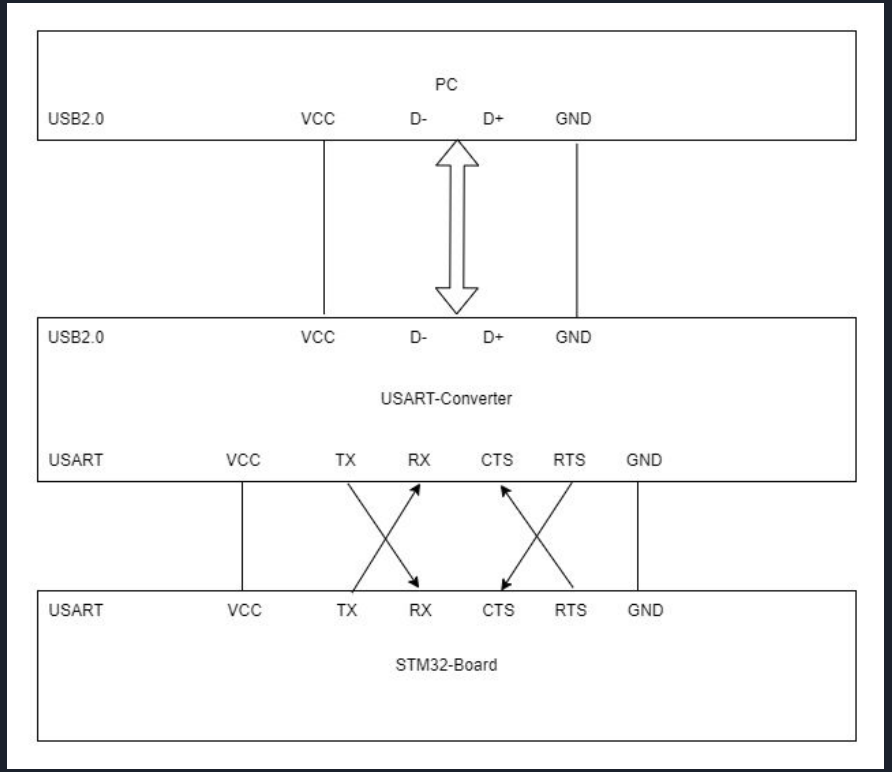
4.2 Code

- New class structure
  - Easier to use
  - Create session key object
  - Use this to send and receive encrypted messages
- Abstraction layer for the communication types
- Socket / USART
- TODO: (Bluetooth / NFC)

4.2 Code - Demo

4.3 Board

- Runs on STM32L4R5ZI
- Connected to PC via USART-Converter
- pqm4 project implements optimized Kyber for Cortex-M4



4.4 Problems

- USART communication
- Board -> PC
  - works
- PC -> board
  - does not receive any data or only incomplete data
- Possible cause:
  - PC delivers data too fast
- Possible solutions:
  - Usage of flow control
  - Reduce baudrate

5 Future Work

Formal proof of security:

- Security of the protocol has been examined, but a formal proof is still missing

Kyber implementations:

- Only the reference implementation is used, there are more optimized implementations available

Board:

- Fix technical issues so the program can be tested
- Run a benchmark

6 Conclusion

- First goal: Develop a working prototype
  - ✓ Achieved
  - ✓ In a development environment

- Second goal: Make it run on a development board
  - ✓ The program can be flashed onto the board and runs.
  - ✖ Communication doesn't work properly

- Side goals: Fix issues left over from the previous semester
  - ✓ Nonce no longer part of the keying material
  - ✓ Protocol has been adapted
  - ✓ Dependency issues are resolved

Agenda	Martyna
Introduction	Martyna
Where we began	Martyna
Goals	Martyna
-Theory	Chiara
-Protocol	Chiara
-Code	Daniel
-Code Demo	Daniel
-Board	Sebastian
-Problems	Sebastian
Future Work	Patrick
Conclusion	Patrick