BPSE SS23: IMPLEMENTING AND BENCHMARKING THE PAKEM PROTOCOL

PATRICK FENDER
YANNICK LECHLER
YAHYA EL HADJ AHMED
DAVID HAAS
RAFAEL CABRAL VOGT
TOBIAS DEPUYDT-WIEDEMANN
GISANE GASPARAYAN-JUNG
FELIX MAXIMILIEN EHONDJE NDOUMBE

SUPERVISOR: NOURI ALNAHAWI, PROF. ALEXANDER WIESMAIER

Agenda

- Project and Goals
- The PAKEM Protocol
- Implementation
- Benchmarks
- Challenges / Future Work
- Live Demo

Motivation & Goals

- Quantum-resistant version of current eID and eMRTD protocols
- Currently used PACE (Password Authenticated Connection Establishment) is based on Diffie-Hellman
- Diffie-Hellman is not quantum-resistant!

➢ PQC KEM Kyber [BDK+18] to replace DH key agreement

Motivation & Goals

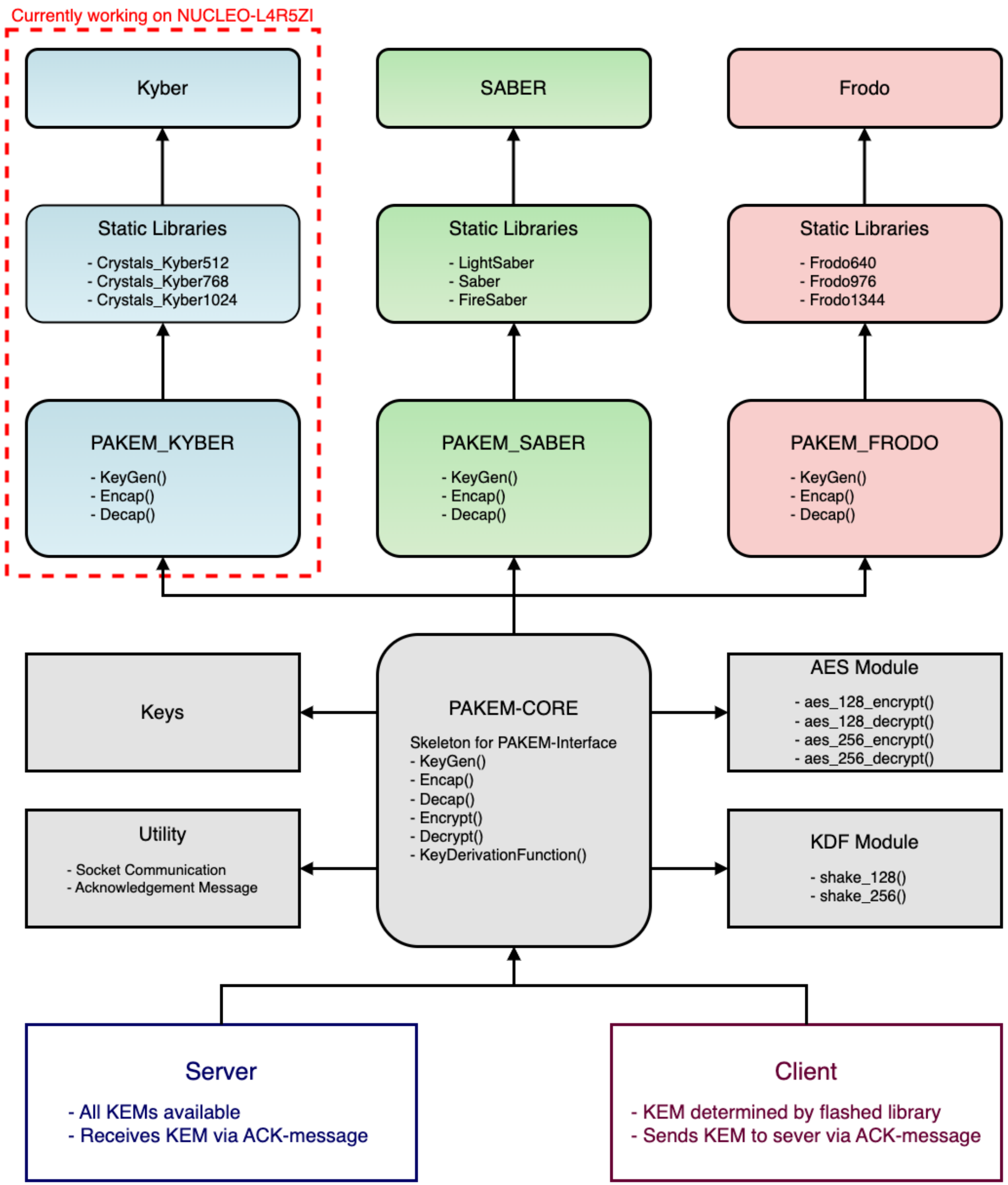
- Implementing PAKEM on NUCLEO STM32 board based on previous student projects
- Integrate libOpenCM3 for STM32 boards
- Integrating optimized CRYSTALS-Kyber-KEM PQM4 [KPR+]
- Implementing thorough performance benchmarks

Alice		**Bob**
$Password\ \pi$		$Password\ \pi'$
$K_\pi = \mathcal{KDF}(\pi)$		$K_{\pi'} = \mathcal{KDF}(\pi')$
$sk_a, pk_a \xleftarrow{\$} KeyGen$		
$apk_a \xleftarrow{\$} C_{K_\pi}(pk)$		
	$\xrightarrow{apk_a}$	
		$pk'_a = \mathcal{C}^{-1}_{K_{\pi'}}(apk_a)$
		$(c_b, \overline{K}) = Encap(pk'_a)$
	$\xleftarrow{c_b}$	
$\overline{K}^* = Decap(sk_a, c_b)$		
$K = \mathcal{KDF}(\overline{K}^*)$		$K = \mathcal{KDF}(\overline{K})$

Password Authenticated Key Encapsulation Mechanism

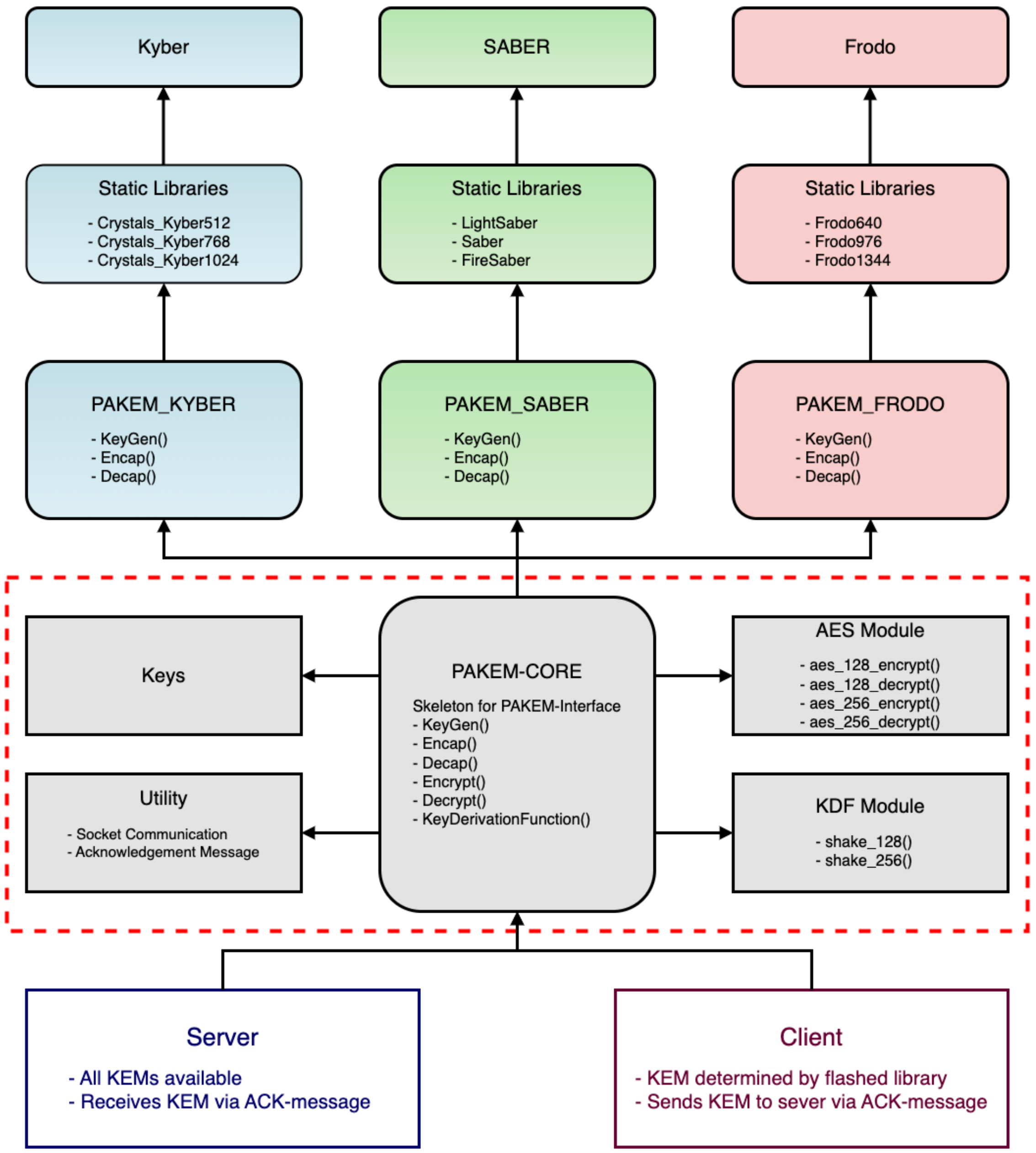
PAKEM Architecture

- Flexibility
  - Interchange components quickly
- Usability
  - Easy to use interface
- Efficiency
  - Improved memory usage
- Extensibility
  - Adding KEMs in the future



Used Libraries

- TinyAES [1]
- PKCS7 Padding [2]
- Shake128/256 [3]



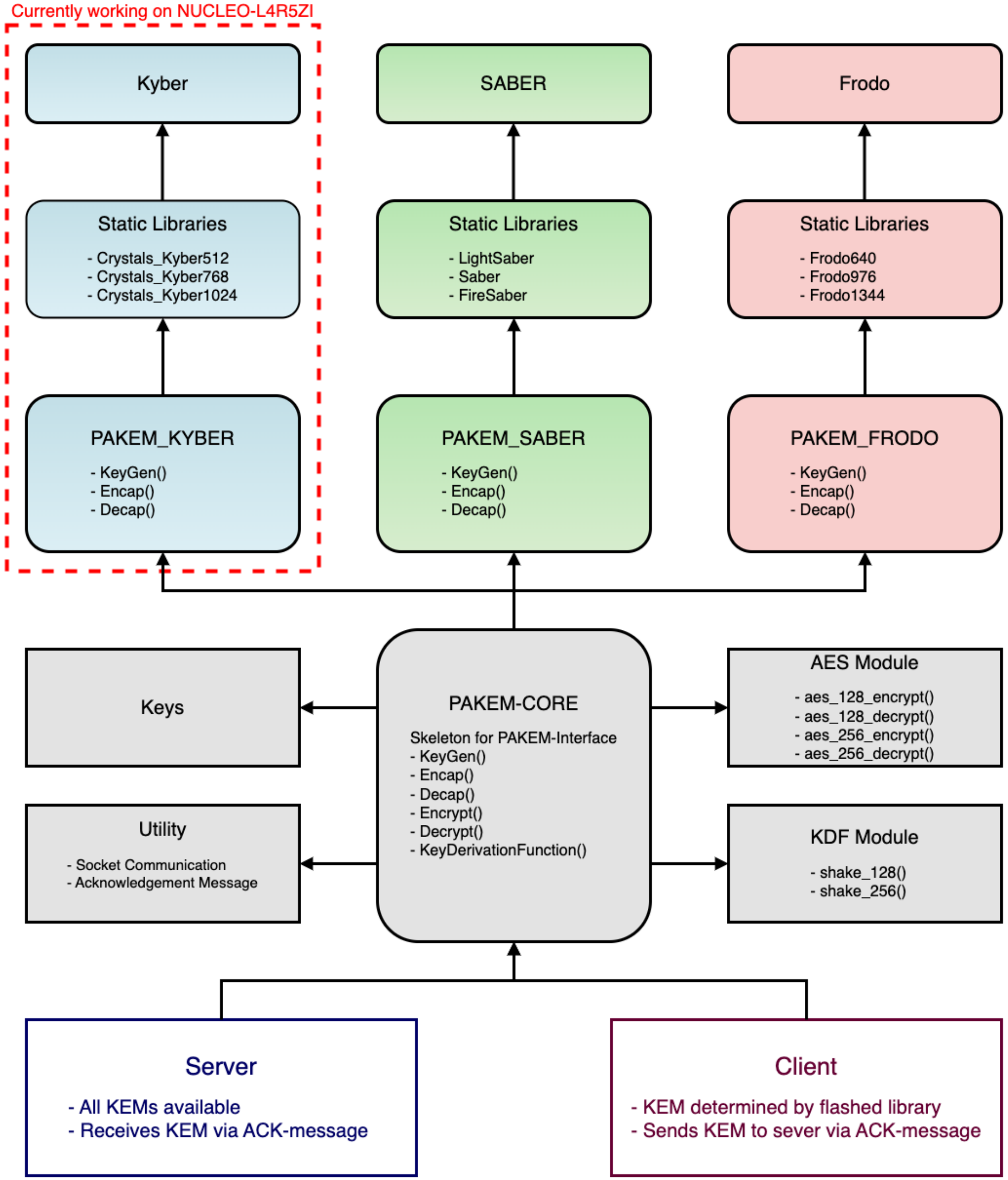
[1] Tiny aes in c: https://github.com/kokke/tiny-AES-c

[2] Pkcs7-padding: https://github.com/GRISHNOV/PKCS7-Padding

[3] Pqcrypto-lweke: https://github.com/microsoft/PQCrypto-LWEKE

Secondary Achievements

- Communicate KEM with ACK-Message
- Memory-Efficiency
- Usability



LibOpenCM3 Integration

- Handle USART communication with libOpenCM3 [1]
- Keeping STM32 HAL for other setups (GPIO, clocks, …)

- ✓ Linking and compiling with libOpenCM3
- ✗ Initializing hardware, sending correct strings

- ⚠ Interrupts for USART never reached
- ⚠ Debugging is difficult, tools by STM32CubeIDE limited and unreliable
- ⚠ Conflicts between STM32 HAL and libOpenCM3

[1] libopencm3: https://github.com/libopencm3/libopencm3

Optimized CRYSTALS-Kyber-KEM

- Post-Quantum Crypto Library for the ARM Cortex-M4 (PQM4 [KPR+])
- Speed- and stack-optimized variants
- Benchmarks were run with speed-optimized variant

Memory Benchmarks

- Thorough speed and memory benchmarks on different system configurations and subroutines
- Compare PAKEM to modified Kyber-Ding-PACE
- Identify memory-intensive subroutines

- Intercepting memory allocation and deallocation
- Tracking total memory usage, peak memory usage and memory usage in subroutines
- Transmitting memory profile for analysis

		Previous Work	Our Work		
		Kyber-Ding-PACE	PAKEM Kyber512	PAKEM Kyber768	PAKEM Kyber1024
RAM and FLASH	RAM used (kB)	4.2	3.6	3.6	3.6
	FLASH used (kB)	61.3	91.6	91.9	93.5
Heap	total usage (kB)	48.4	8.8	13.8	19.9
	global peak (kB)	47.3	5.4	8.9	13.3
	before protocol (kB)	0.1	0.2	0.2	0.2
	after protocol (kB)	47.3	2.8	5.2	8.4
	keygen peak (kB)	5.9	3.4	5.6	8.1
	keygen delta (kB)	5.9	3.2	5.4	7.9
	apk encrypt peak (kB)	2.3	1.8	2.5	3.4
	apk encypt delta (kB)	2.3	1.0	1.3	1.8
	decap peak (kB)	0	1.0	2.0	3.4
	decap delta (kB)	0	-1.6	-1.7	-1.5

Timing Benchmarks

- Timer to measure individual methods
- Timer to measure total runtime of the protocol

		Previous Work	Our Work		
		Kyber-Ding-PACE	PAKEM Kyber512	PAKEM Kyber768	PAKEM Kyber1024
Function Calls	sendClientPakemSetup (ms)	504	502	502	502
	generateKeyPair (ms)	67	32	52	82
	sendClientPublicKey (ms)	501	518	525	533
	receiveDecapsulatedSecret (ms)	86	34	56	337
	deriveSessionKey (ms)	45	1	1	1
	total time (ms)	3669	1095	1146	1469

Challenges

LibOpenCM3 [1] integration ⚠

OpenSSL compilation for ARM ⚠

Missing hardware implementations for SABER [DKRV18] and Frodo [BCD+16] ⚠

Infrastructure (VM, STM32 Cube IDE) ⚠

Multiple redesigns of system architecture ⚠

[1] libopencm3: https://github.com/libopencm3/libopencm3

Future Work

- USART with libOpenCM3 [1]
- Expand code for real-world scenario
- Implement SABER [DKRV18] and Frodo [BCD+16] on STM32 client
- Further benchmarking to compare KEMs

[1] libopencm3: https://github.com/libopencm3/libopencm3

Demonstration

Any Questions?

REFERENCES

- [BDK+18] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber: a cca-secure module-lattice-based kem. In 2018 IEEE European Symposium on Security and Privacy (EuroS&P), pages 353–367. IEEE, 2018.
- [KPR+] Matthias J. Kannwischer, Richard Petri, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. PQM4: Post-quantum crypto library for the ARM Cortex-M4. https://github.com/mupq/pqm4.
- [DKRV18] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure kem. In Progress in Cryptology – AFRICACRYPT 2018, 2018.
- [BCD+16] Joppe Bos, Craig Costello, Leo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! practical,quantum-secure key exchange from lwe. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16, page 1006–1018, New York, NY, USA, 2016. Association for Computing Machinery.