# BIKE

**3rd NIST PQC Standardization Workshop** 

June 8th, 2021



Nicolas Aragon, University of Limoges, France Paulo S. L. M. Barreto, University of Washington Tacoma, USA Slim Bettaieb, Worldline, France Loïc Bidoux, Worldline, France Olivier Blazy, University of Limoges, France Jean-Christophe Deneuville, ENAC, Federal University of Toulouse, France Philippe Gaborit, University of Limoges, France Santosh Ghosh, Intel, USA Shay Gueron, University of Haifa, and Amazon Web Services, Israel & USA

Tim Güneysu, Ruhr-Universität Bochum & DFKI, Germany Carlos Aguilar Melchor, University of Toulouse, France <u>Rafael Misoczki</u>, Google, USA Edoardo Persichetti, Florida Atlantic University, USA Jan Richter-Brockmann, Ruhr-Universität Bochum, Germany Nicolas Sendrier, INRIA, France Jean-Pierre Tillich, INRIA, France Valentin Vasseur, INRIA, France Gilles Zémor, IMB, University of Bordeaux, France

https://bikesuite.org

# Agenda

- BIKE recap
- A hardware-friendly tweak
- BIKE adoption
- New team member Jan Richter-Brockmann



## **BIKE Recap**

- Niederreiter-based KEM instantiated with QC-MDPC codes
- Leverage Fujisaki-Okamoto CCA Transform<sup>1</sup>
- State-of-the-art QC-MDPC Decoding Failure Rate analysis<sup>2</sup>
- Black-Gray-Flip Decoder implemented in constant time<sup>3</sup>
- Following NIST suggestion, BIKE has converged to a single variant

For an updated analysis of the FO transform applied to BIKE, see: Drucker, N., Gueron, S., Kostic, D., & Persichetti, E. (2021). On the applicability of the Fujisaki-Okamoto transformation to the BIKE KEM. Intl. Journal of Computer Mathematics: Computer Systems Theory.
For a comprehensive discussion on Decoding Failure Rate of BIKE decoders, see: Valentin Vasseur's PhD thesis "Post-quantum cryptography: study on the decoding of QC-MDPC codes", 2021, available at: <a href="https://who.rocq.inria.fr/Valentin.Vasseur/phd-defence/3">https://who.rocq.inria.fr/Valentin.Vasseur/phd-defence/3</a>: See BIKE's Additional Implementation available at: <a href="https://github.com/awslabs/bike-kem">https://github.com/awslabs/bike-kem</a> and paper by N. Drucker, S, Gueron, D. Kostic "QC-MDPC Decoders with Several Shades of Gray". PQCrypto 2020: 35-50



# BIKE Recap - Spec

KeyGen : () $\mapsto$ $(h_0, h_1, \sigma), h$ Output: $(h_0, h_1, \sigma) \in \mathcal{H}_w \times \mathcal{M}, h \in \mathcal{R}$ 1: $(h_0, h_1) \stackrel{\$}{\leftarrow} \mathcal{H}_w$ 2: $h \leftarrow h_1 h_0^{-1}$ 3: $\sigma \stackrel{\$}{\leftarrow} \mathcal{M}$	Encaps : $h \mapsto K, c$ Input: $h \in \mathcal{R}$ Output: $K \in \mathcal{K}, c \in \mathcal{R} \times \mathcal{M}$ 1: $m \stackrel{\$}{\leftarrow} \mathcal{M}$ 2: $(e_0, e_1) \leftarrow \mathbf{H}(m)$ 3: $c \leftarrow (e_0 + e_1h, m \oplus \mathbf{L}(e_0, e_1))$ 4: $K \leftarrow \mathbf{K}(m, c)$	Parameters $r$ : block length $w$ : row weight $t$ : error weight $\ell$ : shared secret size $\mathcal{M}$ : message space in $\{0,1\}^{\ell}$ $\mathcal{K}$ : key space in $\{0,1\}^{\ell}$
$\begin{array}{c} \textbf{Decaps}: (h_0, h_1, \sigma), c \mapsto K\\ \text{Input: } ((h_0, h_1), \sigma) \in \mathcal{H}_w \times \mathcal{M}, \ c = (c_0, c_0, c_0)\\ \text{Output: } K \in \mathcal{K}\\ 1: \ e' \leftarrow \texttt{decoder}(c_0, h_0, h_0, h_1)\\ 2: \ m' \leftarrow c_1 \oplus \mathbf{L}(e')\\ 3: \ \textbf{if} \ e' = \mathbf{H}(m') \ \textbf{then} \ K \leftarrow \mathbf{K}(m', c) \end{array}$	$\triangleright e' \in \mathcal{R}^2 \cup \{\bot\}$ $\triangleright$ with the convention $\bot = (0, 0)$	

NOTATIO	N	Functions
$\mathbb{F}_2$ :	Binary finite field.	
$\mathcal{R}$ :	Cyclic polynomial ring $\mathbb{F}_2[X]/(X^r-1)$ .	• $\mathbf{H}: \mathcal{M} \to \mathcal{E}_t$ .
$\mathcal{H}_w$ :	Private key space $\{(h_0, h_1) \in \mathbb{R}^2 \mid  h_0  =  h_1  = w/2\}$	
$\mathcal{E}_t$ :	Error space $\{(e_0, e_1) \in \mathcal{R}^2 \mid  e_0  +  e_1  = t\}$	• $\mathbf{K}: \mathcal{M} \times \mathcal{R} \times \mathcal{M} \to \mathcal{K}.$
g :	Hamming weight of a binary polynomial $g \in \mathcal{R}$ .	• $\mathbf{L}: \mathcal{R}^2 \to \mathcal{M}$
$u \stackrel{\hspace{0.1em}\scriptscriptstyle\$}{\leftarrow} U$ :	Variable $u$ is sampled uniformly at random from the set $U$ .	1999 MALES 1977 (2019) (2019)
⊕:	exclusive or of two bits, componentwise with vectors	



### **BIKE Recap - Performance**

	AVX2	AVX512	VPCLMUL
KeyGen	600	585	470
Encaps	220	205	195
Decaps	2220	1356	1280

Latency cost for BIKE Level 1 in kilocycles (Additional Implementation)

Message Flow	Message	Size	Level 1	Level 3	Level 5
Init. $\rightarrow$ Resp.	h	r	12,323	24,659	40,973
Resp. $\rightarrow$ Init.	C	$r + \ell$	12,579	24,915	41,229

Communication cost in bits

# A HW-Friendly Tweak

#### Algorithm 1: Encapsulation.

**Input** : Public key h. **Output:** Encapsulated key K and ciphertext  $C = (c_0, c_1)$ .

- 1 Generate  $m \stackrel{\$}{\leftarrow} \{0,1\}^{\ell}$  uniformly at random.
- 2 Compute  $(e_0, e_1) \leftarrow \mathbf{H}(m)$ .
- **s** Compute  $C = (c_0, c_1) \leftarrow (e_0 + e_1h, m \oplus \mathbf{L}(e_0, e_1)).$
- 4 Compute  $K \leftarrow \mathbf{K}(m, C)$
- **5** Return (C, K).

#### Algorithm 1: Decapsulation.

**Input** : Private key  $(h_0, h_1, \sigma)$  and ciphertext  $C = (c_0, c_1)$ . **Output:** Decapsulated key K.

- 1 Compute syndrome  $s \leftarrow c_0 h_0$ .
- 2 Compute  $\{(e'_0, e'_1), \bot\} \leftarrow \texttt{decoder}(s, h_0, h_1).$
- **3** Compute  $m' \leftarrow c_1 \oplus \mathbf{L}(e'_0, e'_1)$ .
- 4 if  $H(m') \neq (e'_0, e'_1)$  then
- 5 | Compute  $K \leftarrow \mathbf{K}(\sigma, C)$ .
- 6 else
- $\tau$  | Compute  $K \leftarrow \mathbf{K}(m', C)$
- s Return K.

#### Random Oracles: H, K, and L

Replaced the underlying symmetric cryptographic primitives

∕ New

### Implementation of our Random Oracles

Function	Old	New
Н	AES-256	SHAKE-256
K	SHA2-384	SHA3-384
L	SHA2-384	SHA3-384
PRNG	AES-256	SHA3-384
		All KECCAK-based

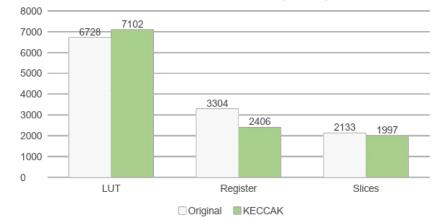
Only one cryptographic primitive is required instead of two



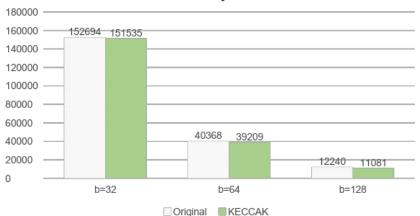
## Hardware

Encapsulation

#### Resource Utilization (b=32)



**Clock Cycles** 



### Software

	Spec v4.1 to v4.2 Slowdown
Key Generation	+1.79%
Encapsulation	+13.54%
Decapsulation	+3.21%

Clock cycles difference for Level 1 on a machine with an Intel Xeon CPU E5-1660 3.2 GHz, 128 GB RAM (Reference Implementation)

#### **Smaller and faster hardware implementation**

#### at the cost of a slightly slower software implementation

Obs: Recall that Encaps is by far the fastest BIKE step (~200 kcycles Additional implementation), thus a ~13% penalty is in practice minor

### **BIKE Adoption - Status Update**

#### **AWS Security Blog**

#### Post-quantum TLS now supported in AWS KMS

by Andrew Hopkins | on 04 NOV 2019 | in Advanced (300), AWS Key Management Service, Security, Identity, & Compliance | Permalink | 🗩 Comments | 🏕 Share

Internet Engineering Task Force Internet-Draft Intended status: Experimental Expires: September 10, 2021 M. Campagna E. Crockett AWS March 9, 2021

Hybrid Post-Quantum Key Encapsulation Methods (PQ KEM) for Transport Layer Security 1.2 (TLS) draft-campagna-tls-bike-sike-hybrid-06

#### Abstract

Hybrid key exchange refers to executing two independent key exchanges and feeding the two resulting shared secrets into a Pseudo Random Function (PRF), with the goal of deriving a secret which is as secure as the stronger of the two key exchanges. This document describes new hybrid key exchange schemes for the Transport Layer Security 1.2 (TLS) protocol. The key exchange schemes are based on combining Elliptic Curve Diffie-Hellman (ECDH) with a post-quantum key encapsulation method (PQ KEM) using the existing TLS PRF.

#### **OPEN QUANTUM SAFE**

software for prototyping quantum-resistant cryptography

The Open Quantum Safe (OQS) project is an open-source project that aims to support the development and prototyping of quantum-resistant cryptography.

### New Team Member

- Jan Richter-Brockmann
  - PhD Candidate Ruhr-Universität Bochum
  - Intern at Intel Labs
  - Area of expertise: efficient Hardware cryptographic implementations



# Thank you

https://bikesuite.org

