OpenSSL CONFERENCE

AI-Assisted PQC Integration in Bouncy Castle

Roy Basmacier · Cryptographer, Bouncy Castle team at Keyfactor

15.10.2026

Nine schemes, one release

OpenSSL CONFERENCE

40
Round 1 · 2023

→

14
Round 2 · Oct 2024 · NIST IR 8528

→

9
Round 3 · 14 May 2026

Family	Round 3 schemes	In Bouncy Castle
Multivariate	MAYO · UOV · QR-UOV · SNOVA	1.81 (MAYO, SNOVA) · 1.85 (UOV, QR-UOV)
MPC-in-the-head	MQOM · SDitH	1.85
Symmetric	FAEST	1.85
Isogeny	SQIsign	1.85
Lattice	HAWK (withdrawn from NIST)	1.85, then pqc.legacy in 1.86

Bouncy Castle 1.85, 12 July 2026: all nine.

What adding a scheme takes

OpenSSL CONFERENCE

core	prov
Engine: KeyPairGenerator · Signer · Parameters · key parameters	<Alg>ParameterSpec · <Alg>Key interface
BCObjectIdentifiers: one OID per parameter set	BC<Alg>PublicKey · BC<Alg>PrivateKey (constant-time equals)
Utils: OID ↔️ parameters maps and lookups	KeyFactorySpi · KeyPairGeneratorSpi · SignatureSpi
PublicKeyFactory · PrivateKeyFactory	Mappings → BouncyCastlePQCProvider.ALGORITHMS
SubjectPublicKeyInfoFactory · PrivateKeyInfoFactory	module-info · tests · release notes

loadPQCKeys(): the BC ↔️ BCPQC bridge. Skip it and certificates decode via BCPQC but not via BC.

About 15 files, 16 steps. One of them is invisible from any single file.

The sprint: 15 May 2026

The day after NIST announced Round 3. Six branches in five minutes, one prompt each.

15.10.2026

OpenSSL CONFERENCE

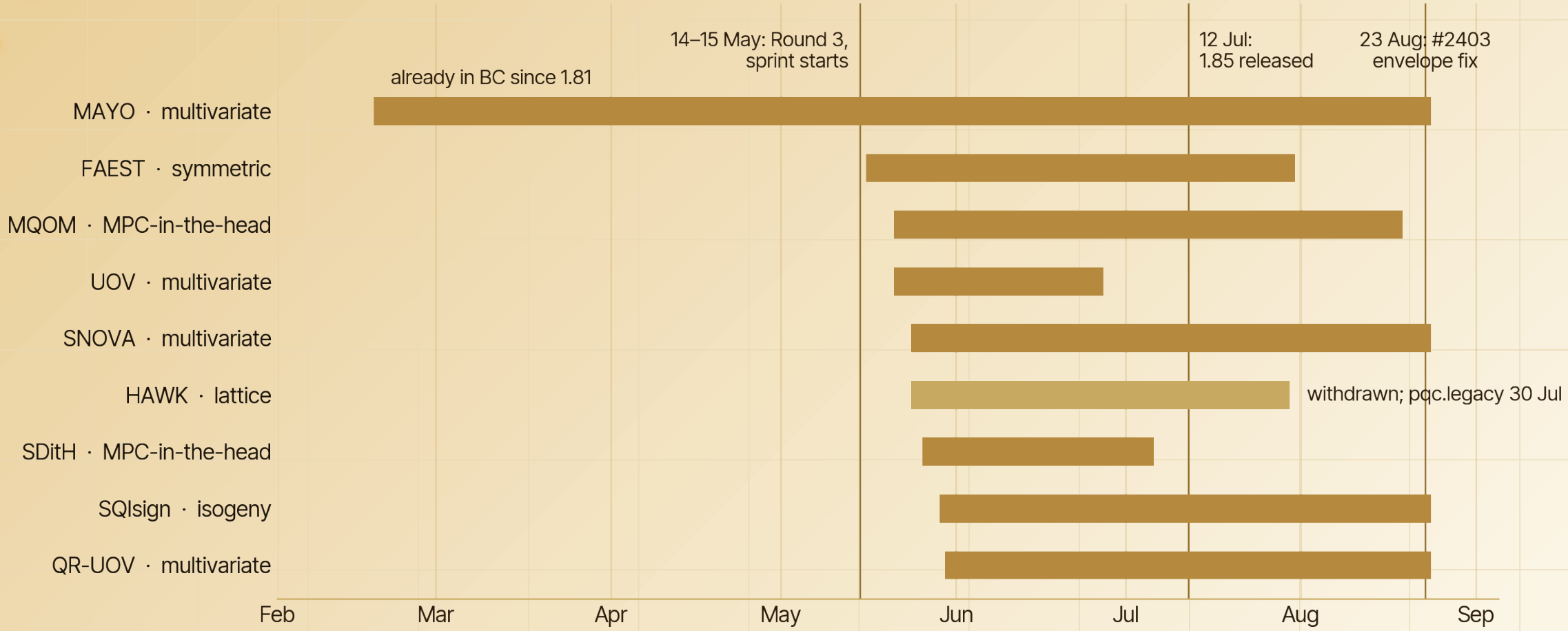
Our goal is to implement SDitH (Syndrome-Decoding-in-the-Head). An implementation is provided in https://github.com/sdith/sdith. The specification is provided in https://sdith.org/docs/sdith-v2.0.pdf. Follow the policies in PQC_SIGNING_POLICIES.

The kick-off prompt, one per scheme

- read ML-DSA, write PQC_SIGNING_POLICIES (the template problem starts here)
- branches: sqisign · mqom · sdith · qr-uov · uov · faest
- **Inputs** reference C beside the tree, the spec PDF, the KAT .rsp files
- **By 18 May** checkstyle hook that auto-formats every edit; the porting skill applied to SQIsign

Fourteen days to merge, 58 to release

OpenSSL CONFERENCE



Accelerates: the plumbing

OpenSSL CONFERENCE

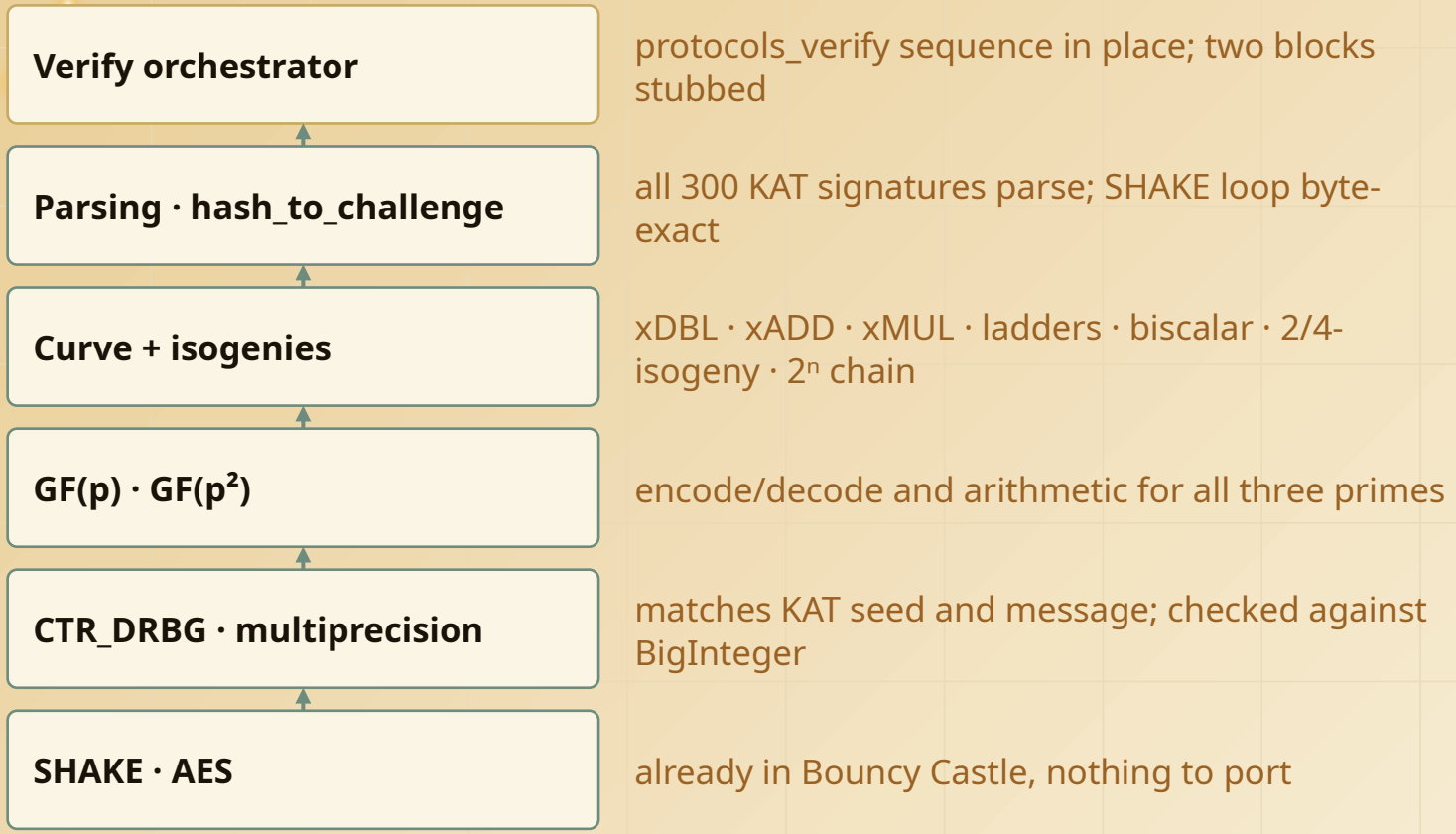
- **Mechanical, cross-cutting, easy to half-finish** exactly what the model does fast and consistently
- **Template chain** SNOVA → FAEST → HAWK
- **Codified** bcpqc-add-signature-algorithm skill: 16 steps, ~15 files, invoked by name
- **One person, six ports in flight** because the model holds the per-scheme reference quirks

1 Utils OID maps	9 KeyPairGeneratorSpi
2 PublicKeyFactory	10 SignatureSpi
3 PrivateKeyFactory	11 Mappings
4 SPKI / PKI factories	12 BCPQC ALGORITHMS
5 ParameterSpec	**13 loadPQCKeys bridge**
6 Key interface	14 module-info
7 Key wrappers	15 Tests
8 KeyFactorySpi	16 Release notes

The skill's checklist

Accelerates: bottom-up, with an oracle

OpenSSL CONFERENCE



Every tier: capture vectors from the C harness, pin the Java byte-exact, then move up.

Thousands of lines of field and curve arithmetic landed correctly.

The workflow got codified: repeatable, not heroic

CLAUDE.md

First version, then split into docs/claude/*.md: build and test, architecture, conventions. Reference quirks go here.

Hooks

After every edit: run the IntelliJ checkstyle. On failure run the formatter.

Skills

Bcpqc-add-signature-algorithm, vulnerability-analysis · bc-crypto-review

Memory

SDitH quirks, the BCPQC-vs-mainline rule, the bare-signature contract: written once, applied to every later port.

Stalls: 21,000 lines of constants

OpenSSL CONFERENCE

~29K
lines of algorithmic C · ported tier by tier

~21K
lines of Sage-generated constants per level · the stall

~1.5K
lines of theta-isogeny (2,2) chain · the other stall

- **The model ports algorithmic C.** It does not accelerate transcribing or regenerating tables that need the original Sage toolchain.
- **Left alone it computes the tables** rather than importing them.
- **Two verify blocks stayed stubbed:** basis-from-hint (needs precomp) and the theta chain.
- **Human time on SQIsign went here,** plus the quaternion-algebra and LLL routines in keygen.

Passes KAT, still wrong

OpenSSL CONFERENCE

Reference quirk in the SDitH port	cat1_gf256	cat3 / cat5 (p251)
Raw Keccak vs SHAKE padding	passes	fails
Opposite sign convention in the SD equation	passes	fails
Different secret-key byte layout	passes	fails
Different XOF consumption order	passes	fails
NB_PARTIES = field size, not 2^k	passes	fails
Different signed-message framing	passes	fails

A green KAT on the default parameter set is not evidence of correctness.

The model will declare victory. Force it onto the hard parameter set.

Why cat1 hid it: with GF(256) the field is a byte, so layout and XOF-order errors line up by accident, and XOR steps cancel sign errors. p251 has none of those coincidences.

The envelope that survived five ports

OpenSSL CONFERENCE

What the reference KAT harness writes (crypto_sign “sm”)

signature (fixed length)	message …

What the spec defines, and what a MessageSigner must return

signature (exact length)

verify(): exactly this length, else false

Round trips test that you match the harness, not the spec.

- **Five schemes shipped it:** MAYO · SNOVA · QR-UOV · SQIsign · AIMer
- **Why it survived:** sign-then-verify passes with the envelope on both ends
- **Consequence:** every cert, CRL, CMS SignedData and TLS CertificateVerify carried a copy of the signed data
- **Fixed 23–24 Aug 2026 for 1.86 (github #2403);** canonical-encoding bugs in QR-UOV and SNOVA found on the way
- **Now enforced:** exact length, and neither envelope order verifies

The invisible wire

OpenSSL CONFERENCE

BC provider
BouncyCastleProvider · what most applications register

loadPQCKeys()

one key-info converter per OID
without it: independent service tables

BCPQC provider
BouncyCastlePQCProvider · where the new scheme is wired

Path	With the bridge	Without it
KeyPairGenerator / Signature via BCPQC	works	works
Unit tests in the BCPQC suite	pass	pass
CertificateFactory.getInstance("X.509", "BC")	decodes the key	public key is null
PKCS#8 / CMS / S/MIME through BC	works	silently fails

90% done is 0% shipped. Test BouncyCastleProvider.getPublicKey() for every parameter set.

Wrong template, and the guardrails

Generalising from the wrong example

- **Policies written from ML-DSA:** FIPS 204, standardised, mainline
- **So SQIsign went to mainline too.** Wrong: Round 2/3 schemes belong in BCPQC
- **Corrected** by a human
- **Pattern:** the most prominent example wins, not the correct one

Its own safety tooling got in the way

"Algebraic-property tests (not byte-level, because the auto-mode classifier blocked the C harness build at this stage)"

sqisign-reference/JAVA_PORT_STATUS.md

- **The verification bar dropped** from byte-exact to algebraic
- **Crypto work trips classifiers constantly** notice when it lowers your standard

What KATs never see

OpenSSL CONFERENCE

Scheme	Secret-dependent behaviour found after landing	Fixed
FAEST	grind check and challenge decode returned early on secret-derived data	31 Jul
SNOVA	Gaussian elimination: pivot search, conditional row swap, zero-factor skip	31 Jul
UOV	zero-ness leak; guard added	16 Jun
SDitH	data-dependent zero-coefficient skips in secret loops	12 Jun
HAWK	data-dependent branch	28 May

“Performance is not as important as security!” · 21 May 2026, after the model proposed scratch-buffer promotions on secret data

All KAT-identical before and after. The countermeasure was manual audit passes, then the review arena.

OpenSSL CONFERENCE

The verification discipline

Every failure on the last six slides was caught by one of these five habits.

15.10.2026

1 The C reference is the oracle

Capture intermediate vectors; pin each primitive byte-exact before moving up.

2 Stash the fix

The test must fail on the original symptom without the fix, and pass with it.

3 Hard parameter set first

cat3/cat5, p251, the awkward field. Not the one where XOR hides errors.

4 Encoding assertions

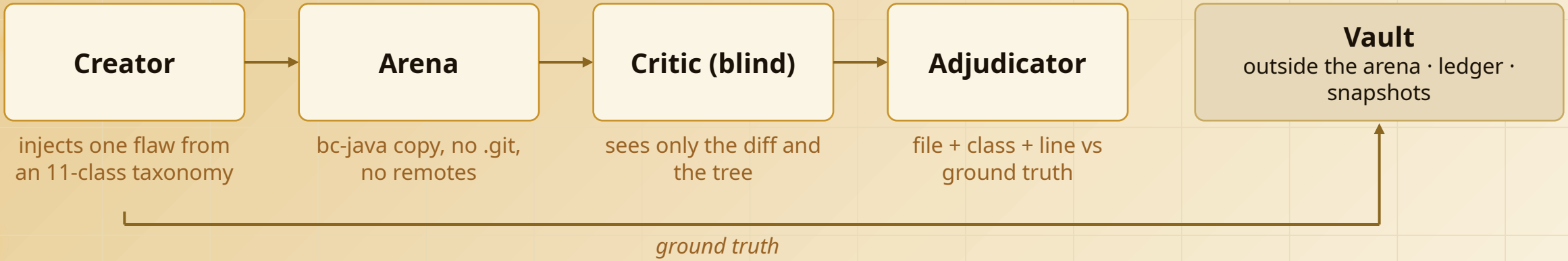
Exact length, no envelope, zero padding bits, elements in [0, q).

5 Memory across sessions

CLAUDE.md, docs/claude, the skill. Written once, used every port.

An adversarial reviewer, measured

OpenSSL CONFERENCE



59 / 59
planted flaws caught

0 / 11
clean controls flagged

- **Subtle catches:** widened HAWK norm bound · dropped MPC soundness round · FAEST opening relaxed to zero-fill · QR-UOV rank-deficient solve
- **70 rounds, 13 batches, v1.0 on 20 July 2026.** Shipped as the bc-crypto-review skill.

Shipped, honestly

1.85 · 12 July 2026

- **New:** FAEST · HAWK · QR-UOV · MQOM · UOV · SQIsign · SDitH
- **Also new:** AIMer · HAETAE (KpqC)
- **MAYO** on the OQS interop OIDs; usable in CMS and X.509
- All parameter sets, all KATs green

1.86 · in progress

- **#2403:** bare signatures for MAYO, SNOVA, QR-UOV, SQIsign, AIMer
- **Canonical encodings** inside QR-UOV and SNOVA signatures
- **MAYO** clears the secret buffers the reference clears
- **HAWK** withdrawn from NIST → pqc.legacy
- **Constant time:** FAEST, SNOVA, UOV, SDitH

Takeaways

Same story in both columns: the acceleration was real, and so were the things it let through.

15.10.2026

OpenSSL CONFERENCE

1 The mechanical 90% takes days, not weeks.
Fifteen files, sixteen steps, six ports in parallel.

2 The remaining 10% is still human.
Precomputed tables, the hardest maths, the one wire no file shows.

3 Green KATs lie. Round trips lie.
Speed is real only with an oracle, the hard parameter set, and encoding checks.

4 Maturity is a checked-in recipe.
CLAUDE.md, hooks, skills, a measured reviewer.

OpenSSL CONFERENCE

Thank you. Questions?

Roy Basmacier · roy@bouncycastle.org · github.com/bcgit/bc-java