European Conference on PQC Migration

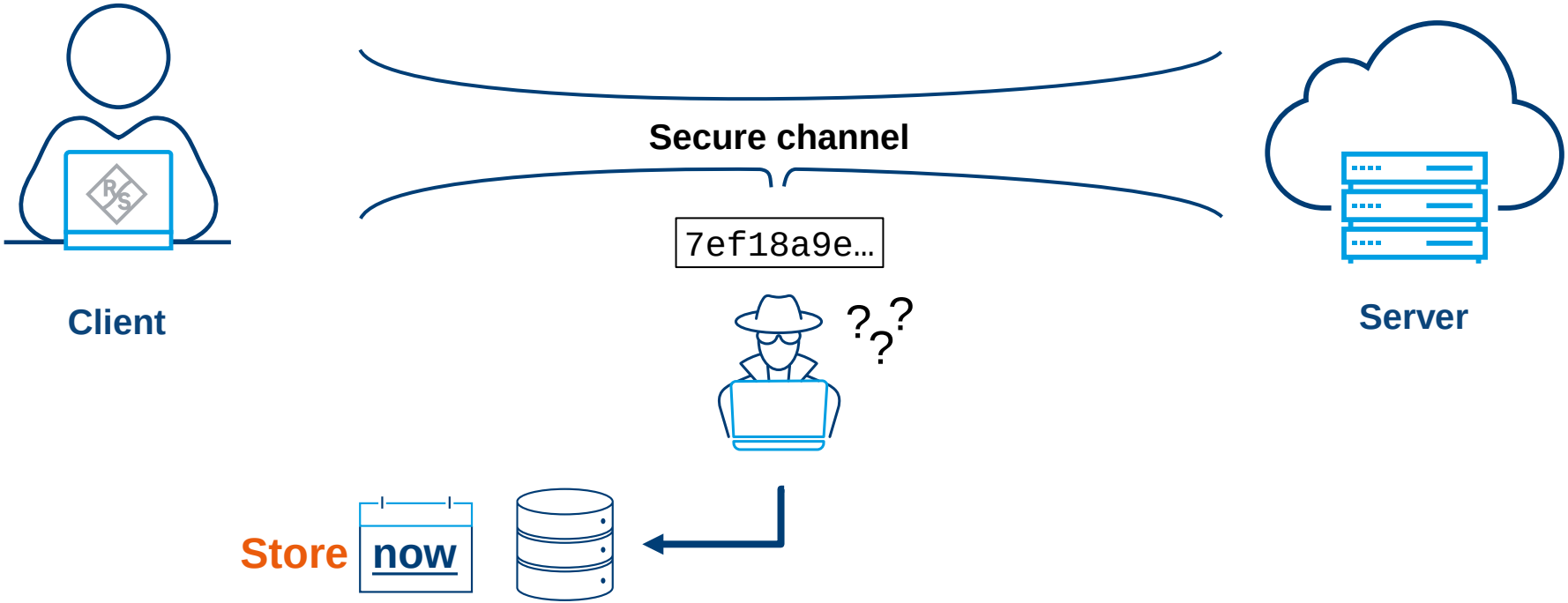
POST-QUANTUM, PRE-APPROVED: PRACTICAL PQC MIGRATION WITH THE BOTAN CRYPTOGRAPHIC LIBRARY

Dr.-Ing. Amos Treiber
Rohde & Schwarz Cybersecurity GmbH

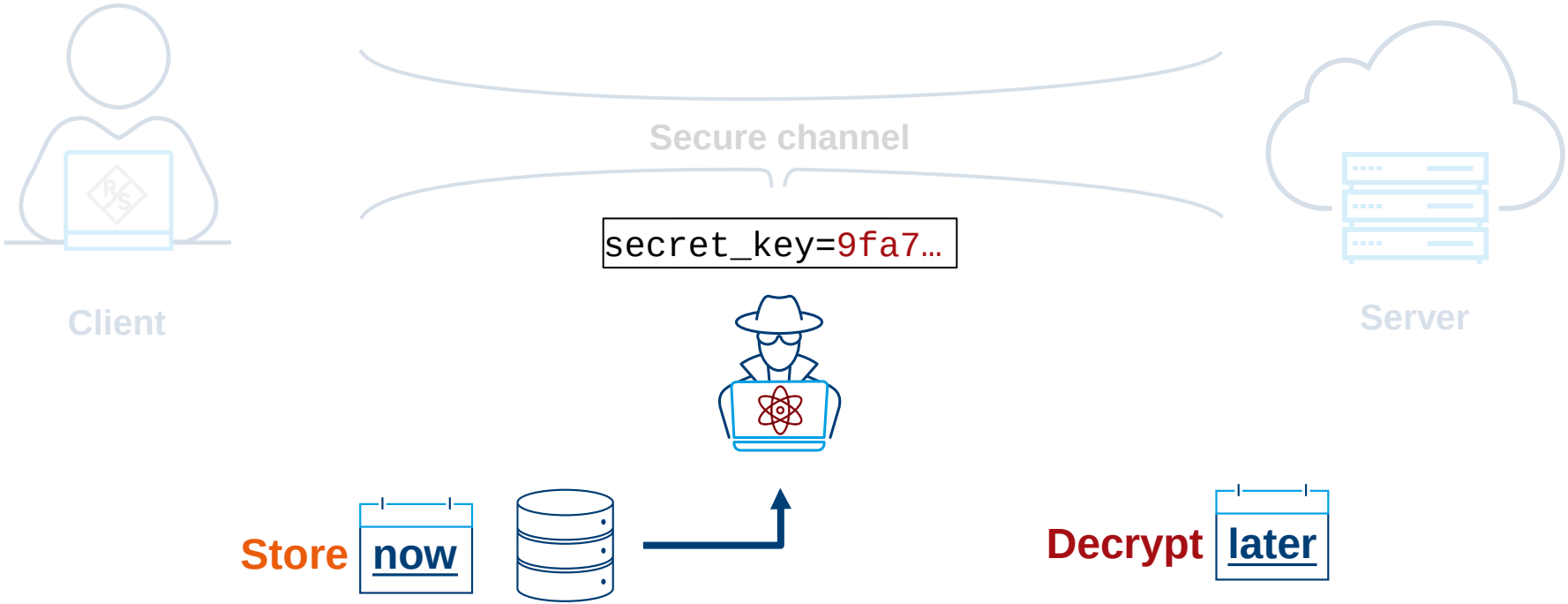
Includes slides by Fabian Albert and René Meusel

ROHDE&SCHWARZ
Make ideas real

TODAY'S EXAMPLE MIGRATION: TLS 1.3



TODAY'S EXAMPLE MIGRATION: TLS 1.3



> *High-risk use cases should be transitioned to PQC as soon as possible, no later than the end of 2030.*

Right now is the hot phase of the transition to Post-Quantum Cryptography (PQC)!

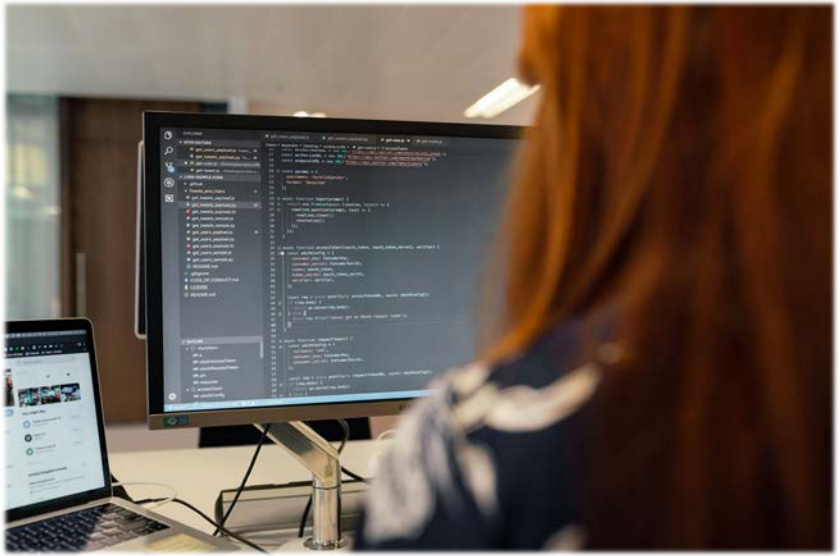
PRESS RELEASE | Publication 23 June 2025

EU reinforces its cybersecurity with post-quantum cryptography

The EU Member States, supported by the Commission, issued a roadmap and timeline to start using a more complex form of cybersecurity, the so-called post-quantum cryptography.

Designed with complex algorithms, this is a key milestone to deflect advanced cyber threats.

PQC MIGRATION IN JUST 2 STEPS?



1. Include into product/service

2. Fill out the paperwork

THE BOTAN CRYPTOGRAPHY C++ TOOLKIT

Botan is Japanese for “Peony Flower”

Recommended for security-sensitive applications

Federal Office for Information Security

github.com/sehlen-bsi/botan-docs

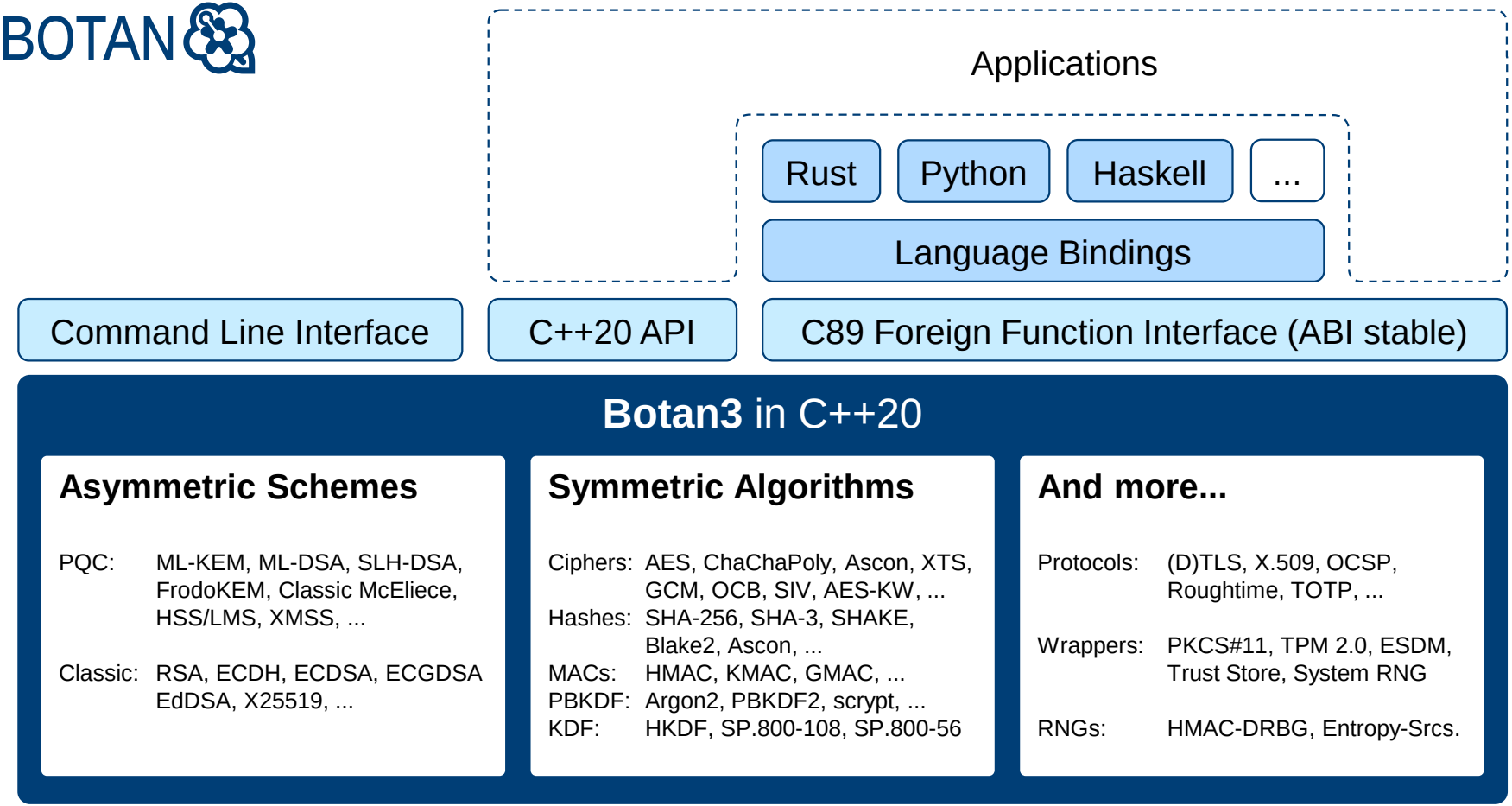
General Purpose Crypto Algorithms
Fundamental Cryptographic Protocols
Software Quality over Performance

Botan 2.x: C++11 (no longer supported)
Botan 3.x: C++20

botan.randombit.net
github.com/randombit/botan

Open Source: BSD 2-Clause “Simplified”

BOTAN


Applications
Rust
Python
Haskell
...
Language Bindings
Command Line Interface
C++20 API
C89 Foreign Function Interface (ABI stable)
Botan3 in C++20
Asymmetric Schemes
PQC: ML-KEM, ML-DSA, SLH-DSA, FrodoKEM, Classic McEliece, HSS/LMS, XMSS, ...
Classic: RSA, ECDH, ECDSA, ECGDSA EdDSA, X25519, ...
Symmetric Algorithms
Ciphers: AES, ChaChaPoly, Ascon, XTS, GCM, OCB, SIV, AES-KW, ...
Hashes: SHA-256, SHA-3, SHAKE, Blake2, Ascon, ...
MACs: HMAC, KMAC, GMAC, ...
PBKDF: Argon2, PBKDF2, scrypt, ...
KDF: HKDF, SP.800-108, SP.800-56
And more...
Protocols: (D)TLS, X.509, OCSP, Roughtime, TOTP, ...
Wrappers: PKCS#11, TPM 2.0, ESDM, Trust Store, System RNG
RNGs: HMAC-DRBG, Entropy-Srcs.

BOTAN RECEIVED PUBLIC FUNDING

InSiTo (BSI)

P197 (BSI)

P481 (BSI)

P663 ? (BSI)

open tender

KBLS (BMBF)

2007 2008 2015 2018 2020 2022 2025 2026

Project 481: Implement PQC Algorithms from NIST Competition

Final PQC standards *published by NIST (August 2024)*

... and available in Botan since 3.6.0 *(October 2024)*

boringSSL, ***Sep 2024***
wolfSSL, ***Oct 2024***
BouncyCastle 1.79, ***Nov 2024***
OpenSSL 3.5.0, ***Apr 2025***

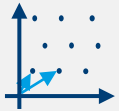
PQC ALGORITHMS INCLUDED IN BOTAN

ML-KEM *(Kyber)*

FIPS-203 / NIST-Standard since **2024**

Key Encapsulation

Recommended in TR-02102 by BSI



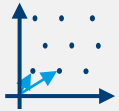
Lattice

FrodoKEM

Draft / ISO as well as Internet-Draft

Key Encapsulation

Recommended in TR-02102 by BSI



Lattice

Classic McEliece

Drafts / ISO as well as Internet-Draft

Key Encapsulation

Recommended in TR-02102 by BSI

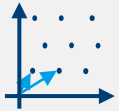
Code

ML-DSA *(Dilithium)*

FIPS-204 / NIST-Standard since **2024**

Signature

Recommended in TR-02102 by BSI



Lattice

SLH-DSA *(SPHINCS+)*

FIPS-205 / NIST-Standard since **2024**

Signature

Recommended in TR-02102 by BSI

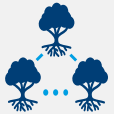
Hash

XMSS / LMS

RFC8391, RFC8554 / IRTF-Standards, with extensions from NIST: SP.800-208

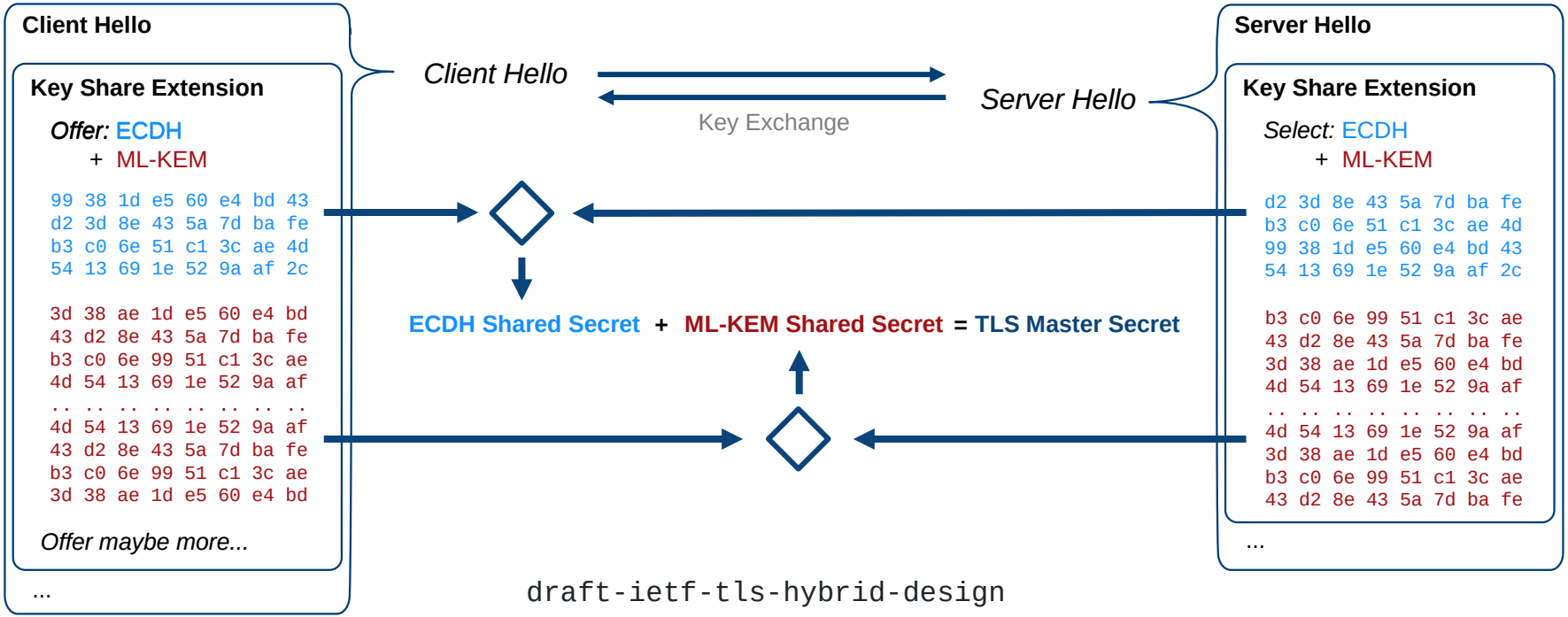
Signatures

Recommended in TR-02102 by BSI



Hash

PQC PROTOCOLS INCLUDED IN BOTAN



AUTOMATIC SIDE CHANNEL DETECTION [CT-GRIND]

```
auto kyber_encapsulate(RandomNumberGenerator& rng) {
    auto secret = rng.random_vec(32);
    CT::poison(secret);

    auto r = expand2(secret);
    auto ciphertext = kpke_encrypt(r);

    CT::unpoison(secret);
    CT::unpoison(ciphertext);
    return std::make_pair(secret, ciphertext);
}
```

Attention!

All clear.

Botan's nightly test results

Matrix: valgrind

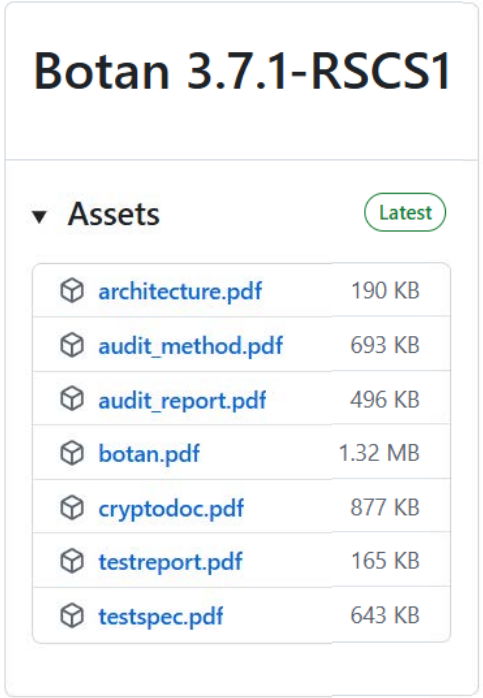
- valgrind (clang, -O1, valg... 1h 4m
- valgrind (clang, -O2, va... 57m 39s
- valgrind (clang, -O3, va... 49m 36s
- valgrind (clang, -Os, val... 59m 53s
- valgrind (gcc, -O1, valgr... 53m 37s
- valgrind (gcc, -O2, valg... 52m 14s
- valgrind (gcc, -O3, valg... 49m 25s
- valgrind (gcc, -Os, valgri... 1h 3m

- **Requires manual annotations**
  - Mark secret data as "sensitive"
  - Remove annotations from return values
- **Tracks dataflow automatically**
  - Values derived from "sensitive data" are also considered "sensitive data"
- **Detects major side-channels**
  - Conditional jumps depending on secrets
  - Secrets used as offsets into memory
- **Dynamic analysis of the binary**
  - No guarantees for other compilers or flags

SUPPORTING DOCUMENTATION

- BSI recommends using **Botan 3.6.1** and **3.7.1-RSCS1**
  https://www.bsi.bund.de/EN/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Kryptografie/Kryptobibliothek-Botan/kryptobibliothek-botan_node.html
  - Botan 3.7.1 required a backport for full TR-02102 compliance
    https://github.com/randombit/botan/pull/4794

- **Supporting Documents at** sehlen-bsi/botan-docs
  https://github.com/sehlen-bsi/botan-docs/releases
  - Cryptographic Documentation
  - Audit Reports for each Version
  - Test Specification and Report
  - API Reference for each Version



SUPPORTING DOCUMENTATION

8.8.1 Algorithm Internals

[FIPS-204] describes three primary operations: key generation, signing, and verification.

Internally, those operations are further split into two functional layers: the public-facing ML-DSA (Section 5), and ML-DSA-internal (Section 6). ML-DSA and ML-DSA-internal decouple the actual high-level logic from the generation of randomness. The functions in ML-DSA-internal receive pre-determined random bytes as needed and are, therefore, fully deterministic.

ML-DSA is a Schnorr-like signature where the typically interactive protocol is made non-interactive by pseudo-randomly deriving the verifier's challenge from the commitment and the message to be signed.

The *following table* provides an overview of the implementation's components and their locations within the Botan source code as well as a mapping to the relevant sections of [FIPS-204].

Table 11: ML-DSA Components and File Locations

Component	Purpose	[FIPS-204]
Types (src)	Strong Types	n/a
Constants (src)	Parameter Set Instantiation	4
Polynomials (src)	Polynomials, Structures on Polynomials and Operations	7.5, 7.6
Algorithms (src)	Encoding, Sampling, Key Expansion, Hint Generation, Rounding	7.1, 7.2, 7.3, 7.4
Symmetric Primitives (src)	ML-DSA Abstractions for PRFs, XOFs, Hashes, and KDFs	3.7
Internal Keys (src)	Internal Key Representation and Serialization	n/a
ML-DSA Implementation (src)	Functional Disambiguation to (also provided) Dilithium	3.7, 6.1, 6.2, 6.3, 7.2
ML-DSA (src)	Public ML-DSA API	5

Botan 3.7.1-RSCS1

▼ Assets — Latest

architecture.pdf	190 KB
audit_method.pdf	693 KB
audit_report.pdf	496 KB
botan.pdf	1.32 MB
cryptodoc.pdf	877 KB
testreport.pdf	165 KB
testspec.pdf	643 KB

SUPPORTING DOCUMENTATION

8.8.1 Algorithm Internals

[FIPS-204] describes thr e pri

Internally, those operati ns ar and ML-DSA-internal (S ectio generation of randomnes . The are, therefore, fully deter minis

ML-DSA is a Schnorr-l ke sig randomly deriving the v rifier'

The *following table* prov des ar source code as well as a mappi

Component		
Types (src)		
Constants (src)		
Polynomials (src)		
Algorithms (src)		
Symmetric Primitives (src)		
Internal Keys (src)		
ML-DSA Implementation (src)	Functional Disambiguation to (also provided) Dilithium	3.7, 6.1, 6.2, 6.3, 7.2
ML-DSA (src)	Public ML-DSA API	5

```
static constexpr void ntt(std::span<T, N> coeffs) {
    size_t j = 0;
    size_t k = 0;

    for(size_t len = N / 2; len > 0; len >>= 1) {
       for(size_t start = 0; start < N; start = j + len) {
          const T zeta = zetas[++k];
          for(j = start; j < start + len; ++j) {
             // Zetas contain the montgomery parameter 2^32 mod q
             T t = fqmul(zeta, coeffs[j + len]);
             coeffs[j + len] = coeffs[j] - t;
             coeffs[j] = coeffs[j] + t;
          }
       }
    }
}
```

Botan 3.7.1-RSCS1

▼ Assets — Latest

architecture.pdf	190 KB
audit_method.pdf	693 KB
audit_report.pdf	496 KB
botan.pdf	1.32 MB
cryptodoc.pdf	877 KB
testreport.pdf	165 KB
testspec.pdf	643 KB

SUPPORTING DOCUMENTATION

Since the previously audited version (3.6.1), Botan 3.7.1-RSCS1 brings some extensions and fixes. The most relevant changes are outlined below.

3.1 Classic McEliece

The code-based post-quantum key-encapsulation mechanism "Classic McEliece" was added in this release. It is implemented according to an ISO draft standard that is currently being finalized for publication.

3.2 (Hybrid) PQC in TLS 1.3

This release removes the experimental support for TLS 1.3 key exchanges using hybrid schemes based on the Kyber Round 3 specification. Instead, the default TLS 1.3 configuration now allows hybrid key exchanges using x25519/ML-KEM-768.

Support for pure ML-KEM key exchanges without a hybridization with an elliptic curve algorithm was added for TLS 1.3 but not enabled by default.

3.3 Elliptic Curve Cryptography

Various optimizations were applied to the new elliptic curve math implementation added back in Botan 3.5.0. The existing elliptic curve implementation is now marked as deprecated and can be disabled at build time. In Botan 3.7.1-RSCS1 this will remove support for user-defined curves, however. The next version of the library (Botan 3.8.0) will add support for such curves independent of the now-deprecated legacy elliptic curve implementation.

3.4 Wrapper for "Entropy Source and DRNG Manager" (ESDM)

This allows an easy integration with the [ESDM] to gain access to high-quality entropy via a user-space random number generator daemon on Linux.

Botan 3.7.1-RSCS1

▼ Assets — Latest

architecture.pdf	190 KB
audit_method.pdf	693 KB
audit_report.pdf	496 KB
botan.pdf	1.32 MB
cryptodoc.pdf	877 KB
testreport.pdf	165 KB
testspec.pdf	643 KB

BSI's Project 48

SUPPOR

- BSI recom
  - Botan 3.
    https://github.c
- **Supportin**
  https://github.com/s
  - Cryptog
  - Audit Re
  - Test Spe
  - API Refe

4.10 ECC Optimizations

Security Classification: *Category I - Critical for security*

4.10.1 Code authors

- Jack Lloyd (@randombit)

4.10.2 Description

Various optimizations for elliptic curves as well as other number theoretic implementations.

Notably, GH #4569, GH #4588, and GH #4618 facilitate optimizations by selectively using variable-time implementations of certain operations when the inputs are known to be public. It is worth noting that group parameters like the modulus and order of ECC groups or the modulus of discrete logarithm groups are considered to be public by these optimizations.

4.10.3 Relevant Patches

Ref.	Patch title / Audit comment	Impact	Author	Approver(s)
#4558 (5826859	**Modify Montgomery exponentiation to return results in Montgomery form** Optimizes the CRT-based RSA decryption algorithm to use in between values in Montgomery form using new classes for handling Montgomery transformed integers. The new implementation only works in combination with the CRT-based approach for balanced primes (bits of p = bits of q). Unbalanced primes are supported using a slower non-CRT approach (also constant time).	critical	@randombit	@FAlbertDev
#4588 (af60450)	**Reduce overhead from computing modular reduction parameters** When setting up barrett reduction parameters at runtime this now distinguishes whether the modulus is secret or public and only uses constant-time division in the former case.	critical	@randombit	@reneme

Botan 3.7.1-RSCS1

▼ Assets — Latest

architecture.pdf	190 KB
audit_method.pdf	693 KB
audit_report.pdf	496 KB
botan.pdf	1.32 MB
cryptodoc.pdf	877 KB
testreport.pdf	165 KB
testspec.pdf	643 KB

BSI's Project 48

SUPPOR

- Supportin
  - Cryptog
  - Audit Re
  - Test Spe
  - API Refe

4.10 ECC Optimizations

Modify Montgomery exponentiation to return results in Montgomery form #4558

Merged randombit merged 1 commit into master from jack/stay-in-monty on Jan 17

randombit commented on Jan 16 — Owner

DSA and ElGamal just immediately convert the result to standard form; the performance of these doesn't matter much at all anymore.

For RSA, keep the values in Montgomery form using the same flow as described as "Smooth-CRT RSA" in https://eprint.iacr.org/2007/039.pdf

...mentations.

...tively using variable-time imple-...orth noting that group parameters ...groups are considered to be public

4.10.3 Relevant Patches

Ref	Patch title / Audit comment	Impact	Author	Approver(s)
#4558 (5826859	**Modify Montgomery exponentiation to return results in Montgomery form** Optimizes the CRT-based RSA decryption algorithm to use in between values in Montgomery form using new classes for handling Montgomery transformed integers. The new implementation only works in combination with the CRT-based approach for balanced primes (bits of p = bits of q). Unbalanced primes are supported using a slower non-CRT approach (also constant time).	critical	@randombit	@FAlbertDev
#4588 (af60450)	**Reduce overhead from computing modular reduction parameters** When setting up barrett reduction parameters at runtime this now distinguishes whether the modulus is secret or public and only uses constant-time division in the former case.	critical	@randombit	@reneme

Botan 3.7.1-RSCS1

▼ Assets — Latest

architecture.pdf	190 KB
audit_method.pdf	693 KB
audit_report.pdf	496 KB
botan.pdf	1.32 MB
cryptodoc.pdf	877 KB
testreport.pdf	165 KB
testspec.pdf	643 KB

Acceptance rules

(automatically checked)

Each upstream patch must ...

- ... be accounted for,
- ... be authored or approved by a project member
- ... fulfill the "four-eye principle"

Supportin

https://github.com/s

- Cryptog
- Audit Re
- Test Spe
- API Refe

4.10 ECC Optimizations

tations.

ly using variable-time imple-
neting that group parameters
os are considered to be public

4.10.3 Relevant Patches

Ref.	Patch title / Audit comment	Impact	Author	Approver(s)
#4558 (5826859	**Modify Montgomery exponentiation to return results in Montgomery form** Optimizes the CRT-based RSA decryption algorithm to use in between values in Montgomery form using new classes for handling Montgomery transformed integers. The new implementation only works in combination with the CRT-based approach for balanced primes (bits of p = bits of q). Unbalanced primes are supported using a slower non-CRT approach (also constant time).	critical	@randombit	@FAlbertDev
#4588 (af60450)	**Reduce overhead from computing modular reduction parameters** When setting up barrett reduction parameters at runtime this now distinguishes whether the modulus is secret or public and only uses constant-time division in the former case.	critical	@randombit	@reneme

Botan 3.7.1-RSCS1

▼ Assets — Latest

architecture.pdf	190 KB
audit_method.pdf	693 KB
audit_report.pdf	496 KB
botan.pdf	1.32 MB
cryptodoc.pdf	877 KB
testreport.pdf	165 KB
testspec.pdf	643 KB

TODAY'S EXAMPLE MIGRATION: TLS 1.3

TODAY'S EXAMPLE MIGRATION: TLS 1.3

Botan PQ Context

```
std::shared_ptr<tls::Context> prepare_context_for(std::string_view hostname) {
   auto rng = std::make_shared<Botan::AutoSeeded_RNG>();
   const auto cfg =
      "allow_tls_12 = false\n"
      "allow_tls_13 = true\n"
      "key_exchange_groups = x25519/ML-KEM-768 brainpool256r1";

   return std::make_shared<tls::Context>(std::make_shared<System_Credentials_Manager>(),
                                         rng,
                                         std::make_shared<tls::Session_Manager_In_Memory>(rng),
                                         std::make_shared<tls::Text_Policy>(cfg),
                                         tls::Server_Information(hostname));
}
```

Botan 3.2.0

reneme released this Mar 15, 2024 · 427 commits to main since this release · 3.2.0 · 114a482

Notable additions:

- Support for TLS hybrid key exchange compatible with draft-ietf-tls-hybrid-design
- TLS 1.3 with pre-shared keys
- Addition of KMAC

▾ Assets 10

architecture-botan-3.2.0.pdf	156 KB	Mar 15, 2024
audit_method-botan-3.2.0.pdf	682 KB	Mar 15, 2024
audit_report-botan-3.2.0.pdf	410 KB	Mar 15, 2024

THANK YOU FOR YOUR ATTENTION!

ANY QUESTIONS?

?

Dr.-Ing. Amos Treiber

(Cryptography) Development Engineer

amos.treiber@rohde-schwarz.com

@atreiber94

amos-treiber