Post-Quantum

Cryptography Conference

Migrating and benchmarking a banking application

There is a hesitance to start the migration to quantum-safe solutions, which leads to slower adoption of PQC. This stems from multiple PQC algorithms to choose from, each having specific strong and weaker points, especially with respect to performance, storage and bandwidth. How to deploy these? Hybrid or not? What is the impact on my application? By helping (financial) organisation in migrating their application and testing them in an operational setting, we capture system effects and benchmarking results. In this presentation, I will discuss the results and experiences gained during the migration and share common pitfalls.

Alessandro Amadori

Cryptographic Researcher at Netherlands Organisation for Applied Scientific Research (TNO)

SSL.com PQ SHIELD HID KEYFACTOR ENTRUST

January 15 and 16, 2025 - Austin, TX (US) | Online

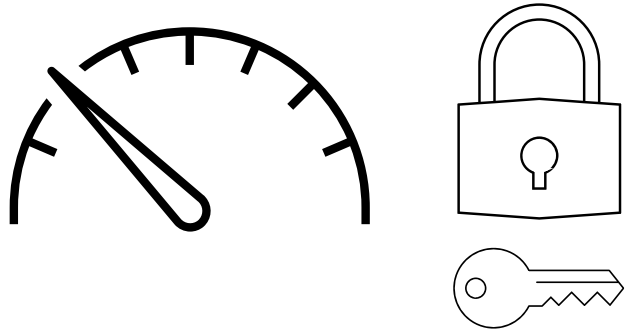
PCSI – PQC Benchmarking

Alessandro Amadori

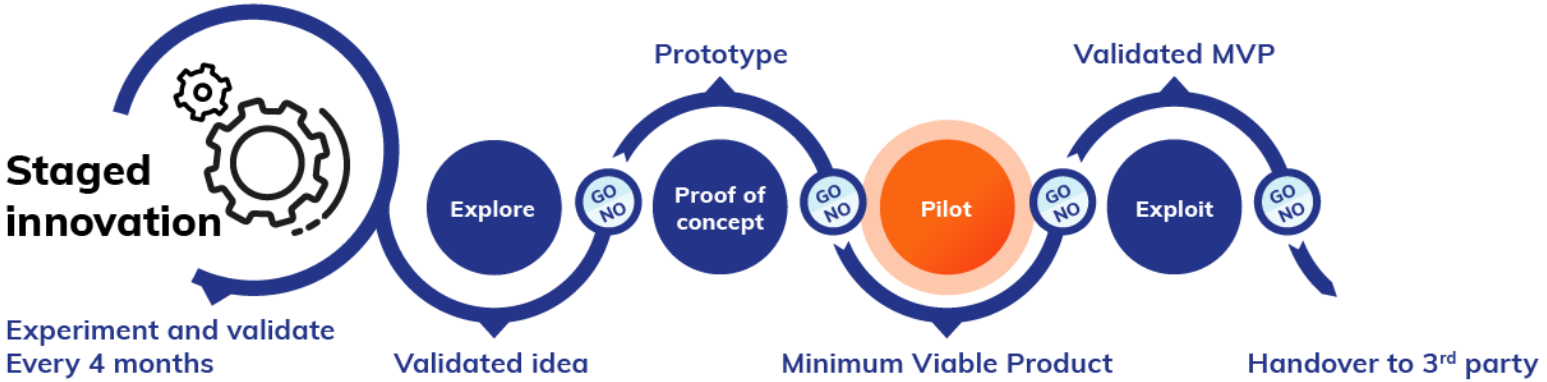
PQC Conference Austin ‘25

PROJECT SET-UP

Goal and Process

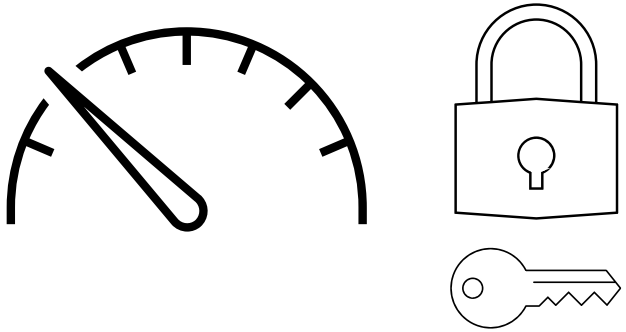


- Project part of the Partnership for CyberSecurity Innovation (PCSI) initiative
  - Partners gather to share research ideas and directions
- Goal: Investigate methods and impact of the PQC migration



PQC benchmarking | Projects

Problem introduction



- Starting point:
  - There is a lot of **uncertainty** about the migration to a quantum safe solution
  - The effect of this uncertainty is **hesitance to act**, leading to increased risks

- The uncertainty stems from:
  - There is not one replacement, but multiple,
  - With specific (performance and functional) drawbacks and
  - Multiple ways of replacing it (e.g. hybrid)
  - The new algorithms differ from our current cryptography in terms of performance, size, bandwidth and capabilities

- Available benchmarks are academic or very limited in scope
  - Not capturing *system effects*: bottlenecks, unexpected behaviour, architectural changes needed
  - Not sharing migration experiences (and pitfalls)

Project impact

Impact “for the world”:

- **Knowledge** on **impact** of PQC migration on common IT systems
- **Guidance** on which algorithm/migration **option** might suit them
- **Pitfall/bottleneck** information

Impact for each partner (doing a benchmark):

- PQC migration with relatively little pain (due to support TNO and other partners)
- Precise information on migration choices for tested system.
- Tested system is practically migrated (or knowledge on why it cannot be migrated)
- An experienced internal team, that can share experiences and help other migrations go smoothly
- Increased internal awareness

PoC details

- PQC Algorithms (NIST level 1 or equivalent):
  - **Key exchange:**
    - **Crystals-Kyber (ML-KEM),**
    - **FrodoKEM,**
    - **Classic McEliece**
  - Digital signature:
    - Crystals-Dilithium (ML-DSA)
    - Falcon (FN-DSA)
    - Sphincs+ (SLH-DSA)

- Metrics:
  - CPU usage
  - Storage
  - Memory
  - Network information (bandwidth, package size, package drops, latency)
  - Time (connections/signings/key exchanges per second)

- Test cases:

1. **Current algorithms (ground comparison)**
2. Everything ECC
3. **Only key exchange using PQC**
4. Digital signatures replaced by PQC
5. Key exchange and digital signatures to PQC
6. Key exchange hybrid (with ECC curve from 2)
7. Digital signature hybrid (with ECC curve from 2)
8. Key exchange and digital signatures to hybrid (with ECC curves from 2)

Information gathered:

- Performance indicators
- System details (hardware, platform, version of bouncycastle, etc)
- Algorithm details currently being tested (key size, algorithm variant)
- Encountered problems, bottlenecks, learned lessons about implementation, adjustments required outside of replacing the algorithm

Description of ABN AMRO use case PCC

Use Case ABN AMRO

Personal Communication Channel with PQC algorithms

- Protocol meant for undeniable communication and digital signing
- Transactions and Documents can be signed with PCC
- PCC relies heavily on key exchange

Application inside ABN AMRO

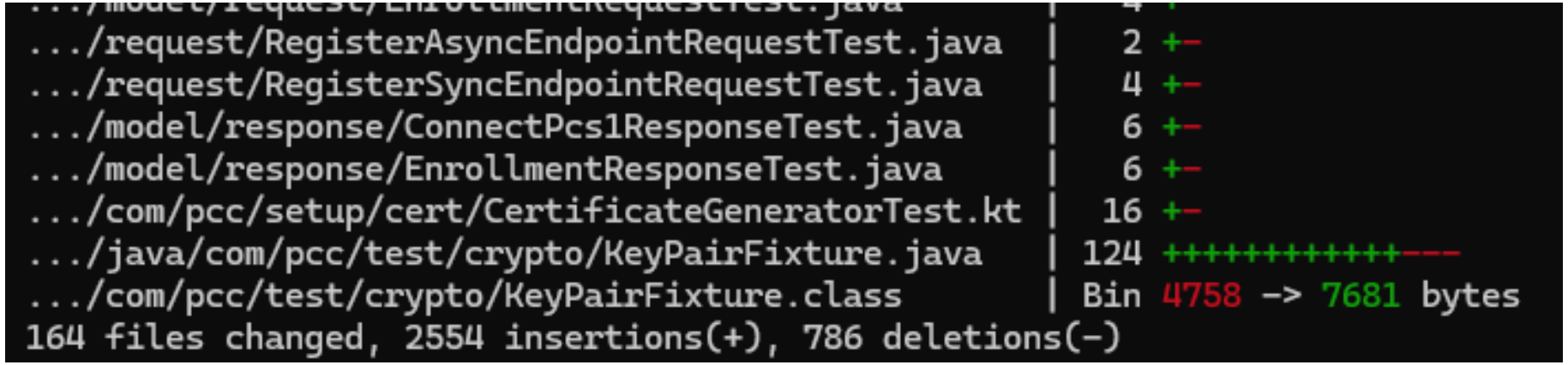
- Java application with Bouncy Castle crypto library

Reprogram PCC with

- ML-DSA, SLH-DSA, FN-DSA, ML-KEM, FrodoKEM & Classic McEliece
- Benchmark the current crypto algorithms versus the PQC algorithms

ABN AMRO Use Case – git diff

The protocol is left unchanged, but many files had to be adapted in order to make it crypto-agile and easily use the PQC algorithms.



*It’s slightly less because a .class is included in this diff

BEST CODING PRACTICES: LESSONS LEARNED and CRYPTO-AGILITY

ABN AMRO Use Case – lessons learned

- A more **crypto-agile** system should be easier to migrate
  - Tests also need to be migrated and made crypto-agile!
- Introducing extra packages for KEX is undesirable and probably a lot of work in code.
  - Something to consider if you want to provide forward secrecy.
  - You can’t simply replace ECC 1-on-1 with ML-KEM
- If you want to outsource your migration, expect quite some time for other party to comprehend your (multi-year) project

ABN AMRO Use Case – BouncyCastle cipher strings

Use the correct strings for the crypto algorithms to use in BouncyCastle (BC & BCPQC)

In BC

"LMS", "SPHINCSPlus", "Dilithium", "Falcon", "NTRU“

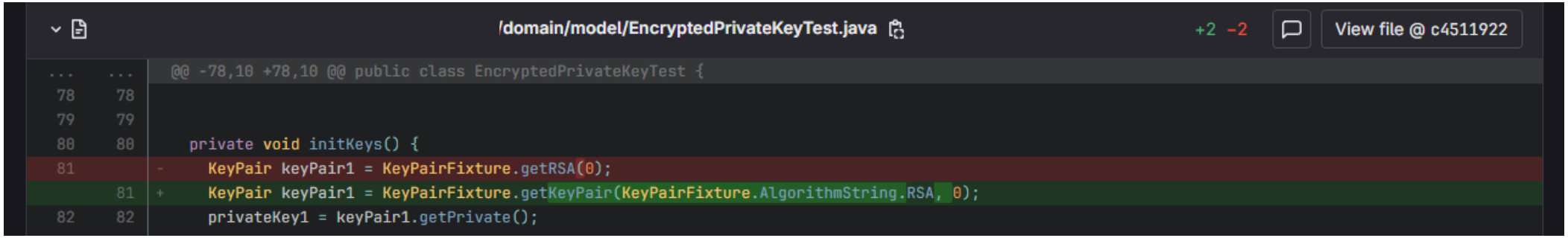
Note: `Security.getAlgorithms( serviceName: "BC");` returns *mceliece348864_r3* a.o. but this doesn’t work.

In BCPQC

"SPHINCS", "LMS", "NH", "XMSS", "SPHINCSPlus", "CMCE", "Frodo", "SABER", "Picnic", "NTRU", "Falcon", "Kyber", "Dilithium", "NTRUPrime", "BIKE", "HQC", "Rainbow"

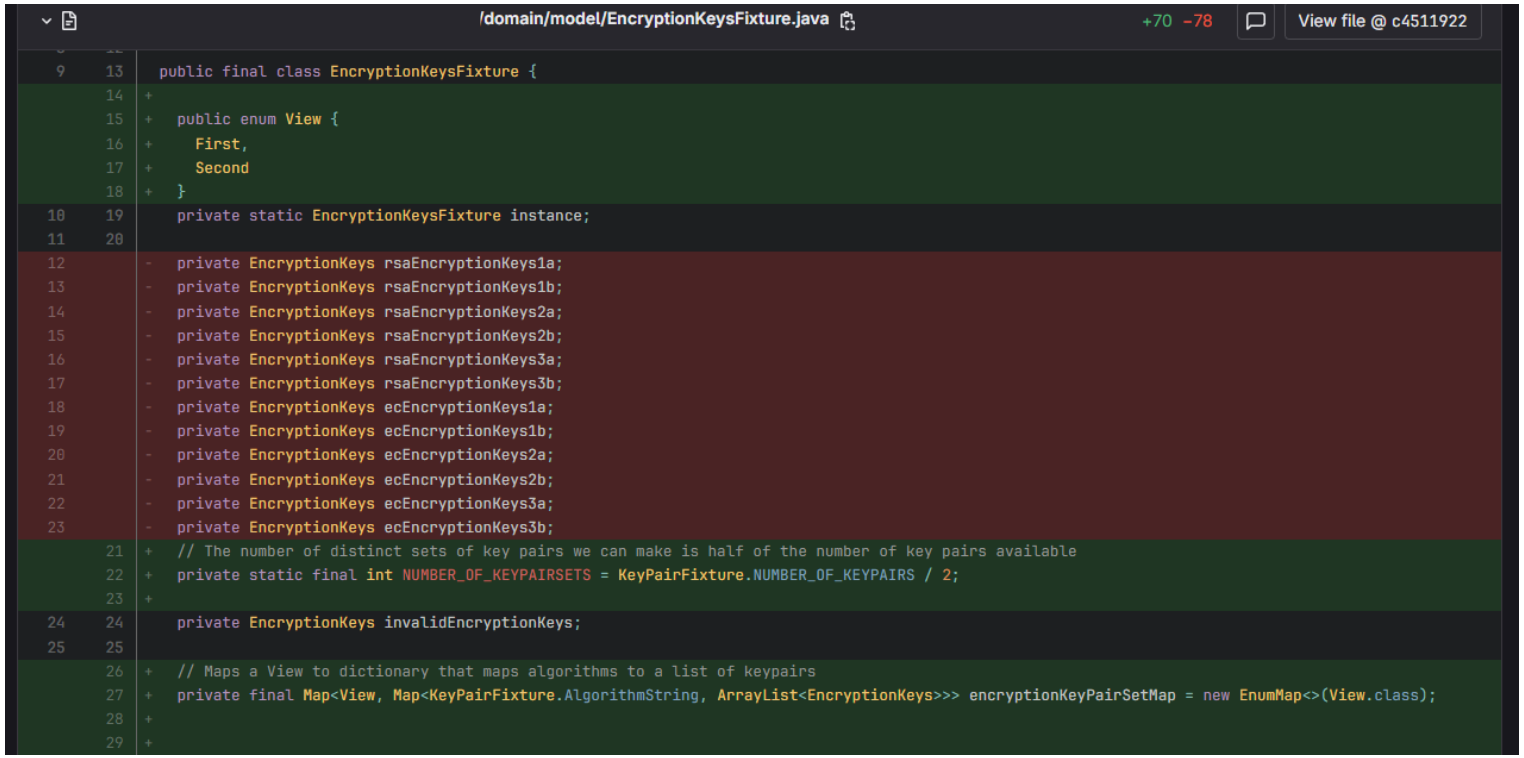
```
KeyPairGenerator kpg = KeyPairGenerator.getInstance( algorithm: "Frodo", provider: "BCPQC");
```

ABN AMRO Use Case – modular referencing



- Use algorithms as parameters instead of having special methods for them

ABN AMRO Use Case – dynamic storage objects

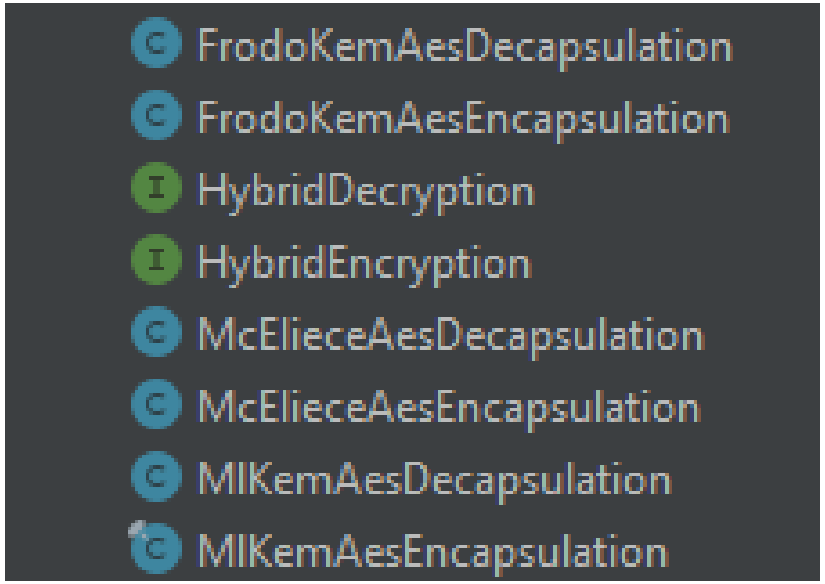


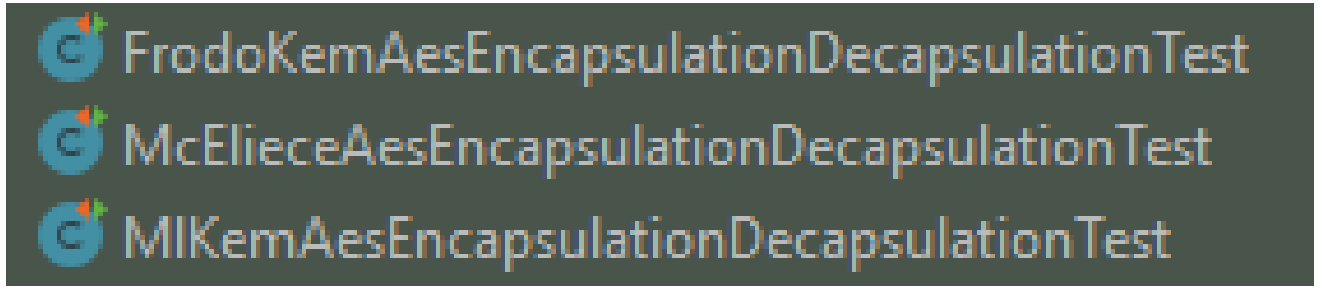
/domain/model/EncryptionKeysFixture.java +70 −78 View file @ c4511922

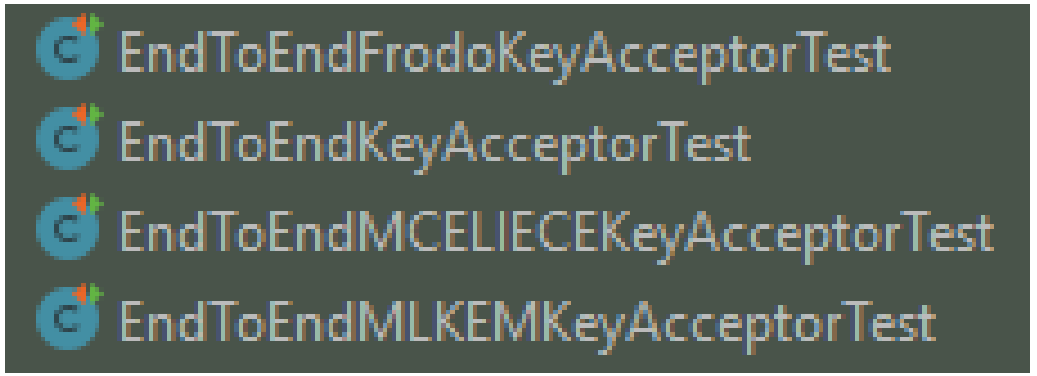
```
 9  13  public final class EncryptionKeysFixture {
     14  +
     15  +   public enum View {
     16  +     First,
     17  +     Second
     18  +   }
10  19     private static EncryptionKeysFixture instance;
11  20
12      -  private EncryptionKeys rsaEncryptionKeys1a;
13      -  private EncryptionKeys rsaEncryptionKeys1b;
14      -  private EncryptionKeys rsaEncryptionKeys2a;
15      -  private EncryptionKeys rsaEncryptionKeys2b;
16      -  private EncryptionKeys rsaEncryptionKeys3a;
17      -  private EncryptionKeys rsaEncryptionKeys3b;
18      -  private EncryptionKeys ecEncryptionKeys1a;
19      -  private EncryptionKeys ecEncryptionKeys1b;
20      -  private EncryptionKeys ecEncryptionKeys2a;
21      -  private EncryptionKeys ecEncryptionKeys2b;
22      -  private EncryptionKeys ecEncryptionKeys3a;
23      -  private EncryptionKeys ecEncryptionKeys3b;
     21  +  // The number of distinct sets of key pairs we can make is half of the number of key pairs available
     22  +  private static final int NUMBER_OF_KEYPAIRSETS = KeyPairFixture.NUMBER_OF_KEYPAIRS / 2;
     23  +
24  24     private EncryptionKeys invalidEncryptionKeys;
25  25
     26  +  // Maps a View to dictionary that maps algorithms to a list of keypairs
     27  +  private final Map<View, Map<KeyPairFixture.AlgorithmString, ArrayList<EncryptionKeys>>> encryptionKeyPairSetMap = new EnumMap<>(View.class);
     28  +
     29  +
```

- Defining a modular mapping from algorithms to keypairs, instead of having multiple hard-coded private variables

ABN AMRO Use Case – modular class structure







- This should be improved for crypto-agility.
- Suggestion: QSEncapsulation and QSDecapsulation, pass the algorithm name to the abstract class instead of defining separate classes for each and every crypto algorithm

ABN AMRO Use Case – Database lay-outs

/domain/model/e2e/PreparedTrustedKeys.java +40 −34 View file @ 23470f23

```
@@ -20,35 +20,37 @@ public class PreparedTrustedKeys {
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

-   @NotNull
-   @Valid
-   @Embedded
-   @AssociationOverride(name = "encryptedPublicKey", joinColumns = @JoinColumn(name = "EC_PU_KEY_ID"))
-   @AssociationOverride(name = "encryptedPrivateKey", joinColumns = @JoinColumn(name = "EC_PR_KEY_ID"))
-   private EncryptionKeys ecKeys;
-
-   @NotNull
-   @Valid
-   @Embedded
-   @AssociationOverride(name = "encryptedPublicKey", joinColumns = @JoinColumn(name = "RSA_PU_KEY_ID"))
-   @AssociationOverride(name = "encryptedPrivateKey", joinColumns = @JoinColumn(name = "RSA_PR_KEY_ID"))
-   private EncryptionKeys rsaKeys;
-
-   @NotNull
-   @Valid
-   @Embedded
-   @AssociationOverride(name = "encryptedPublicKey", joinColumns = @JoinColumn(name = "MLKEM_PU_KEY_ID"))
-   @AssociationOverride(name = "encryptedPrivateKey", joinColumns = @JoinColumn(name = "MLKEM_PR_KEY_ID"))
-   private EncryptionKeys mlKemKeys;
```

Hardcoded key types were used as columns in a database. In a crypto-agile system, you do not know the key types in advance, so a different database lay-out is necessary. As the code uses Jakarta persistence annotations to define the database lay-out, new decorators need to be used to handle this new lay-out. A new table stores the algorithm and values as rows and is referenced using identifiers from the original table.

```
+   @ElementCollection
+   @CollectionTable(name = "prepared_trusted_encryption_keys_map_table", joinColumns = @JoinColumn(name = "entity_id"))
+   @MapKeyColumn(name = "key_type")
    private Map<AsymmetricKeyType, EncryptionKeys> encryptionKeysMap;
```

ABN AMRO Use Case – Modular Tests

```java
public static Stream<Arguments> testGenerateKeyPair() {
  return Stream.of(
      // AsymmetricKeyType
      // -----------------
      arguments(AsymmetricKeyType.EC), //
      arguments(AsymmetricKeyType.RSA),
      arguments(AsymmetricKeyType.MLKEM));
}
```

- Write tests that check for specific algorithm-agnostic behaviour
- Load in parameters dynamically from a specified set of algorithms or set-up parameter-based tests that require minimal manual work to update
- Here, the code could be made even more crypto-agile by using all enumeration values in the AsymmetricKeyType enumerator as arguments, removing the need for manual updates

LESSONS LEARNED: ORGANISATIONAL

Project – lessons learned

- We do not recommend delegation of the PQC migration:
  - Overhead of code access
  - Code will break
  - High number of dependencies (correct integration is very time-consuming)
  - You will not learn from the process

- We recommend:
  - Involve PQC experts **for support**
  - Ensure good code practices (crypto-agility, modularity, documentation, Javadoc) across your organisation.
  - Involve both software developers and architects

Project – lessons learned

- The availability of quantum-safe products (VPNs, HSMs, …) is limited. A detailed overview is needed
  - PQC Capabilities Matrix (PQCCM) | PKI Consortium is an excellent good start, but we also need information on versions, dependencies, integrations, upgradeability etc.

- Having close relations and concrete discussions with your vendors is crucial to make agreements on your migration.

- Trend: very aware and enthusiastic architects and engineers, but management is hesitant.
  - No PoC is possible without management support and prioritisation.
  - Informal tip: convincing your manager for a PoC is a lot easier than for a big general PQC migration, and snowballs awareness for the PQC migration, and can be good marketing

LOOKING FORWARD

Next Phase

During this first PoC we got experience to move on to the next phase

We expect the results of the benchmark soon. Stay tuned for the results!

We are working towards helping migrating two essential applications for two different partners

- OCSP responses in combination with HSM usage
- Connection setups in a high throughput messaging system

PQC benchmarking | Projects

Partnership for Cyber Security Innovation is a collaboration of

ING

TNO

Belastingdienst

ABN·AMRO

achmea

ASML

www.pcsi.nl

follow us