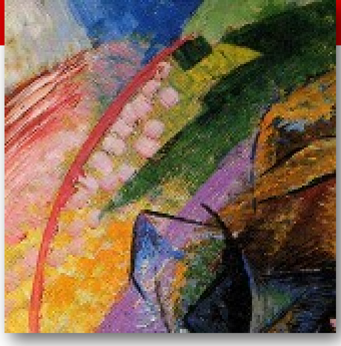
the information security provider

NIST Crypto Agility Workshop
Ideas from the Field and a FIPS Lab

Stephan Müller <Stephan.Mueller@atsec.com>



Background

- A FIPS 140 Tester
  - Involved in technical discussions with vendors
- A developer of cryptographic modules
  - Jitter RNG entropy source (https://chronox.de/jent)
  - Leancrypto library (https://leancrypto.org)
  - HSM-Style Key Manager

Agenda

- Consideration of Crypto Agility
- Reports from the Field
- Reports from the Lab

Consideration of Crypto Agility Transition Challenge

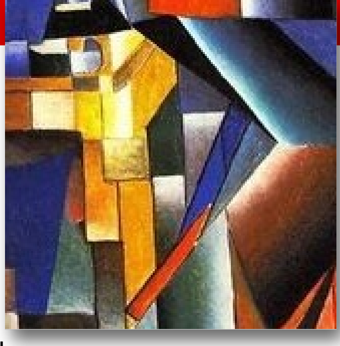
- Existing cryptographic algorithm use cases embedded into larger environment
  - Environments have their own constraints and requirements
  - Assumptions: constraints are / were valid at time of inception
- New cryptographic algorithms intended to replace older
  - New algorithms have their own constraints and requirements

→ Key Issue: Vendor has to synchronize both constraints and requirements

Consideration of Crypto Agility
Synchronize Constraints

- The **faster** constraints from crypto algorithm and usage environments can be **matched**, the **faster** the **adoption** of the new algorithm
- The **smaller** the constraint **deviation** between crypto algorithms and environments, the **faster** the **adoption** of the new algorithm
- Synchronization effort:
  - Environmental constraints are adjusted → change in the code using the crypto module
  - Crypto algorithms allow flexibility in their use

Consideration of Crypto Agility NIST: Support Synchronization



- Define new algorithms with greatest degree of freedom
  - Allow only freedom that is cryptographically sound
- Goal: Number of constraints of crypto algorithm is reduced → vendors may match them more easily with environmental constraints
- NIST provided good examples:
  - ML-DSA: pure, pre-hash, external-Mu, hedged, deterministic, seed preservation
  - SLH-DSA: pure, pre-hash, hedged, deterministic, f and s variants
  - ML-KEM: seed preservation

Report from the Field
Example I

- Module with key store holding 256 bit keys
  - May hold also RSA or ECC keys
- Module can immediately hold ML-DSA keys by storing their seed values
  - Recomputes the key pair if needed
- Module cannot (yet) immediately hold ML-KEM or SLH-DSA keys as their seeds are larger (512 bits for ML-KEM, up to 768 bits for SLH-DSA)

  → NIST can apply proposed change to SP800-133 allowing to store 256 bits seed

  - to KDF ML-KEM seeds
  - to KDF seeds for ML-DSA/ML-KEM plus classic algorithm key pairs

  → NIST may consider allowing seed usage for SLH-DSA

→ Module does not require data base format change to hold (hybrid) PQC key pairs

→ Adoption by vendor is simplified and perhaps faster

Report from the Field
Example II

- Attestation keys stored in hardware
  - Storing of PQC key pairs not economic due to size
  - Storing of seeds of PQC key pairs will fit into existing storage schemes similar to previous slide

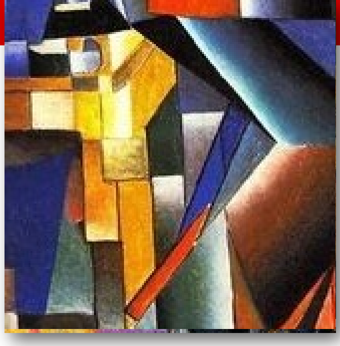
Report from the Lab

- From FIPS 140 lab perspective, no concern or challenge around crypto agility is currently identified
- However, available freedom around crypto algorithms may be misused
  - Cryptographically illiterate developers
  - Missing awareness of where allowed freedom is used in cryptographically unsound manner

→ NIST is recommended to foster guidance around freedoms (using SP800-227 as example)

→ NIST is recommended to collaborate with standardization organizations like IETF on sensible use cases

- Cryptographic agility and resistance to misuse is challenging



Thank you!
Questions?