RSAConference2021
May 17 – 20 | Virtual Experience

RESILIENCE

SESSION ID: **CRYP-W13A**

SoK: How (not) to Design and Implement Post-Quantum Cryptography

Thomas Prest

Senior Cryptography Researcher
PQShield
https://tprest.github.io/

James Howe

Cryptography Engineer
PQShield
https://jameshowe.eu/

This SoK: a transversal survey of post-quantum cryptography

Most works focus on (one aspect of) one family:

- **Lattices:** LWE, SIS, etc. → Falcon, Kyber, etc. → Implementation → Deployment
- **Codes:** SD, RSD, etc. → McEliece, Wave, etc. → Implementation → Deployment
- **Isogenies:** SSDDH, CSIDH, etc. → SIKE, CSIDH, etc. → Implementation → Deployment
- etc.

We tried to abstract away the family, and focus on the process:

- **Family X:** Assumption → Scheme → Implementation → Deployment

Plan

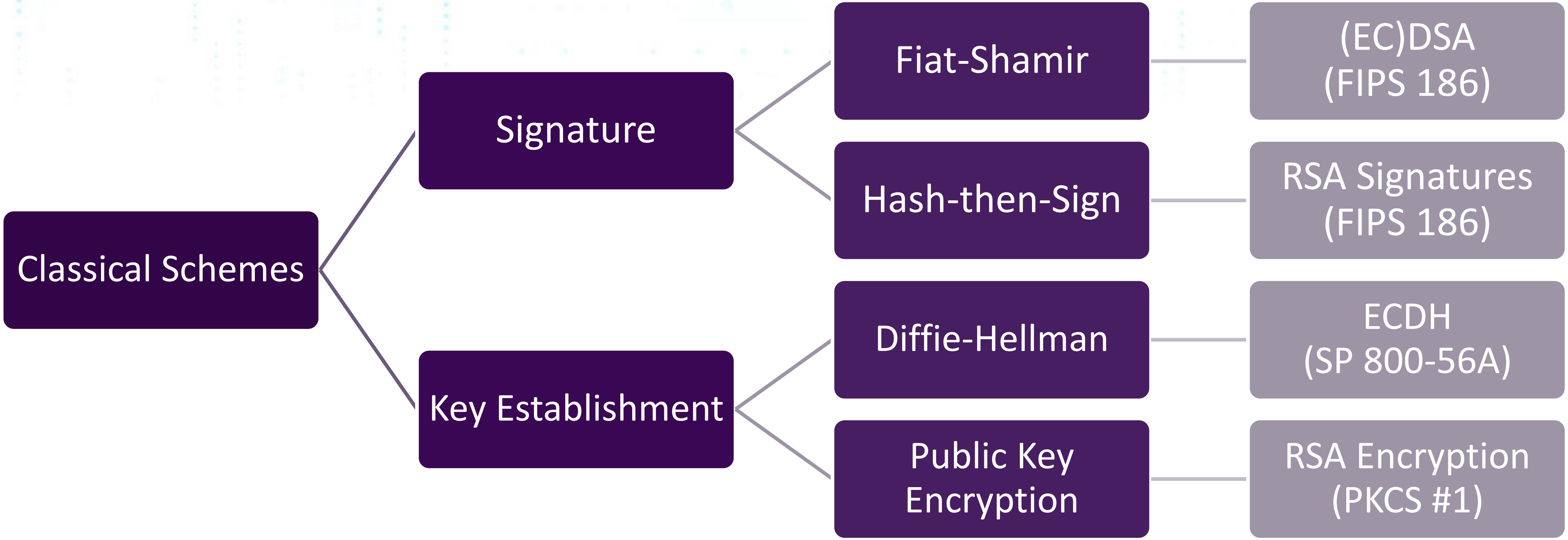
- **This talk:** Scheme → Implementation
- **The paper:** Assumption → Scheme → Implementation
- **The ePrint:** Assumption → Scheme → Implementation → Deployment

Our goals

- Survey essential works
- Establish trends and patterns
- Provide `` lessons learned ''

RSAConference2021

Theory and Design of Schemes



CLASSICAL SCHEMES

- Post-Quantum Schemes
  - Signature
    - Fiat-Shamir
      - Dilithium (L)
      - MQDSS, PKPDSS (M)
      - Picnic, Banquet (O)
      - Stern signatures, Durandal (C)
    - *Hash-then-Sign*
      - Falcon (L)
      - Rainbow, GeMSS (M)
  - Key Establishment
    - Diffie-Hellman
      - SIDH (I)
    - Public Key Encryption
      - FrodoKEM, Kyber, Saber (L)
      - BIKE, Classic McEliece, HQC (C)

MOST POST-QUANTUM SCHEMES - (L)ATTICES, (C)ODES, (I)SOGENIES, (M)ULTIVARIATE, (O)NE-WAY FUNCTIONS

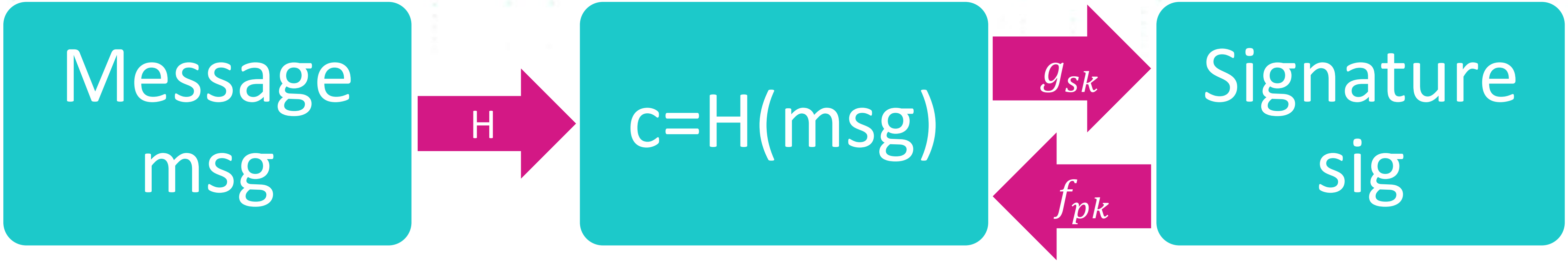
``Just apply one of these four paradigms to your favourite problem'' ? Not so simple.

- These paradigms are useful guidelines, but they are **no panacea**.
  - Rigidly applying a paradigm may result in an **inefficient scheme** or a **broken assumption**.
  - It is more important to preserve the **security proof** than the paradigm.

- Most efficient schemes tweak paradigms to fit the assumption.
  - Full-domain hash **without permutations** (see next slide)
  - Fiat-Shamir **with aborts** and further refinements [Lyu09,BG14,Dilithium]
  - Various soundness-amplification tricks [DG19,KKW18]

  See the paper for a complete discussion.

Example: Full-Domain Hash Signatures



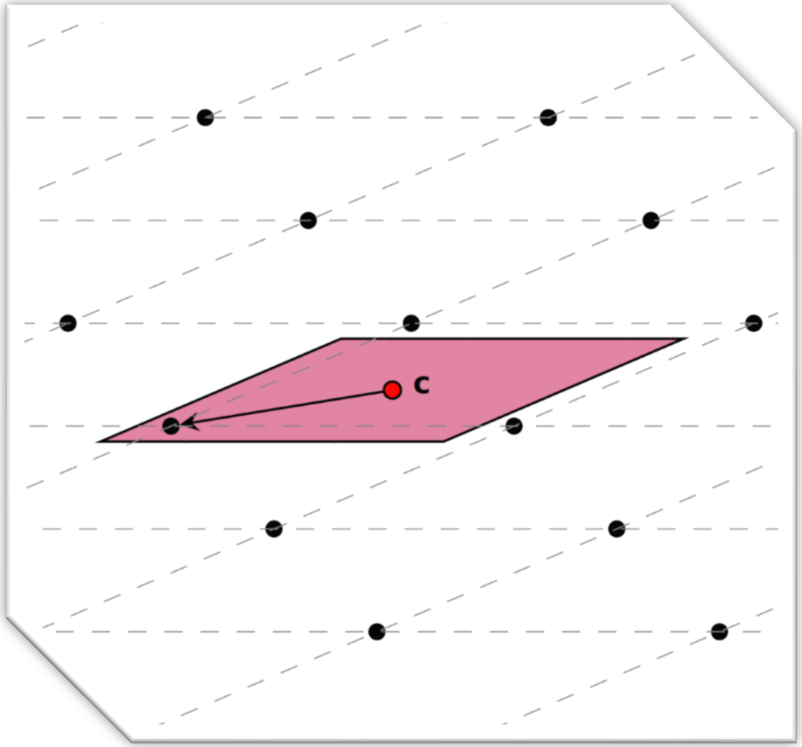
We say that the pair ($f_{pk}: X \to Y$, $g_{sk}: Y \to X$) is a trapdoor permutation if:

1. Given only pk, it is computationally hard to invert f_{pk} on (almost) all inputs.
2. $f_{pk} \circ g_{sk}$ is the identity over Y, and $X = Y$ (hence f_{pk} and g_{sk} are permutations).

Canonical example: RSA signatures [RSA78] and its many variants.
Provable security is well-studied [BR96,Coron00].

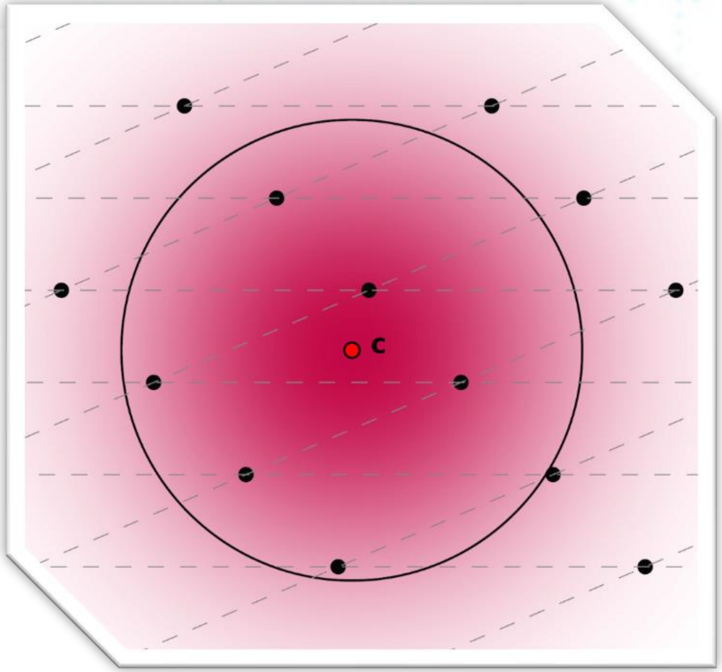
Can we transpose this to the post-quantum setting?

- Initial attempts: GGHSign [GGH97](lattices), [CFS01](codes), NTRUSign [HHPSW03] (lattices)
  - **CFS'01:** poor scalability of parameters
  - **GGHSign and NTRUSign:** (f_{pk}, g_{sk}) was not a trapdoor permutation.
    - The usual security proof did no longer apply
    - Worse, each signature leaked information about the signing key sk, leading to practical attacks [NR06,DN12].



Solution: relaxing the notion of trapdoor permutation

- Gentry-Peikert-Vaikuntanathan [GPV08]:
  - *Trapdoor preimage sampleable functions* (TPSF)
  - Weaker than trapdoor permutation
    - Can be instantiated from lattices (but not codes)
    - Still strong enough for a security proof
- A further relaxation [DST19,CGM19,CD20]:
  - *Average TPSF*
  - Can be instantiated from codes and lattices



- Trapdoor permutation ⇒ TPSF ⇒ Average TPSF (⇒ ?)
- Examples: Falcon [FHK+17] (TPSF, lattices), Wave [DST19] (Average TPSF, codes)

RSAConference2021

Implementation

An Overview of PQC Implementations

- Transitioning to PQC will be tough.
- PQC will bring many new and unique challenges.
  - Larger public/secret keys, signs, ciphertexts.
  - Needs more resources; time, hardware, energy.
  - New operations and paradigms.
- Implementations will be more complex.
  - Rejection sampling,
  - Sampling non-uniform distributions,
  - Decryption failures, etc.

Implementation Attacks on PQC

- Many initial implementations not isochronous.
  - Timing leakage [Str10,AHP+12,Str13,ELP+18,EFG+17].
  - FLUSH+RELOAD cache attacks [BHL+16,PBY17].
  - Data-dependent branching / branch tracing [EFG+17].
- Recent attacks exploit implementation mistakes.
  - Non-isochronous memcmp() in FO transform [GJN20].
  - Errors in domain separation in FO [BDG20].
- Fixable by following secure coding practices.

Side-Channel Attacks on PQC

- A talk on this topic has been done [AH21].
- Power analysis targets secrets via:
  - Matrix multiplication [ATT+18,PSK+18,BFM+19].
  - Polynomial multiplication [HCY20].
  - Syndrome decoding [RHH+17,SKC+19].
- Also, fault, cold-boot, and key reuse attacks.
- Side-channel "hints" for security evaluations.
  - For lattice reduction [DDG+20] and ISD [HPR+21].

Masking and Hiding in PQC

- We've only just begun protecting these schemes.
- So far, only masked Saber, Kyber and Dilithium.
  - First-order [MGT+19], higher-order [BDK+19,BGR+21].
- Countermeasures also not a guarantee.
  - QcBits masking [RHH+17] was attacked [SKC+19].
  - Masked comparison [BPO+20] was attacked [BDH+21].
- Hedging mitigates faults for Fiat-Shamir signs [AOT+20].

Benchmarking PQC

- Evaluate performances in SW, HW, and SCA.
  - On ARM Cortex M4 and Xilinx Artex 7 FPGA.
- Plenty of repos exist for SW benchmarking.
  - PQClean, pqm4, SUPERCOP, etc.
- Seed expanding in pqm4 can take >50% runtime [KRS+19].
- Lots of HW designs exist, too.
  - With a large variety in resources/performance.

RSAConference2021

Thank you!

Full paper: https://eprint.iacr.org/2021/462