Threshold Schemes for Cryptographic Primitives

Computer Security Division

Information Technology Laboratory — National Institute of Standards and Technology

NIST National Institute of Standards and Technology U.S. Department of Commerce

INFORMATION TECHNOLOGY LABORATORY

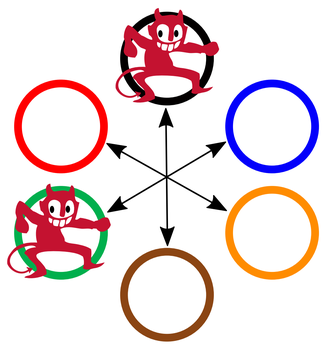
The Threshold Cryptography project

How can we address single-points of failure in implementations of cryptographic primitives ?

- Attacks exploit vulnerabilities in implementations
- Rogue operators make bad usage of secret keys

Solution: Standardize Threshold Schemes

- Each component operates only on a share of the key
- Tolerance to compromise of f-out-of-n components
- Enhanced resistance to side-channel attacks



Example primitives to thresholdize

1. Key-generation (e.g., AES, ECC, RSA)

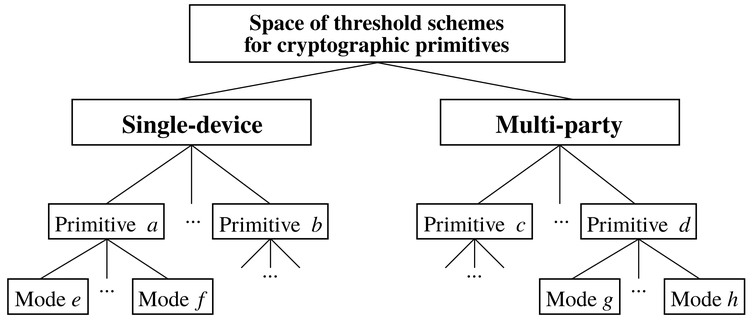
2. Signing (e.g., RSA, ECDSA, EdDSA)

3. Enciphering (e.g., AES)

4. Decryption (e.g., RSA)

5. Random number generation

The space of possibilities is large



Example modes:

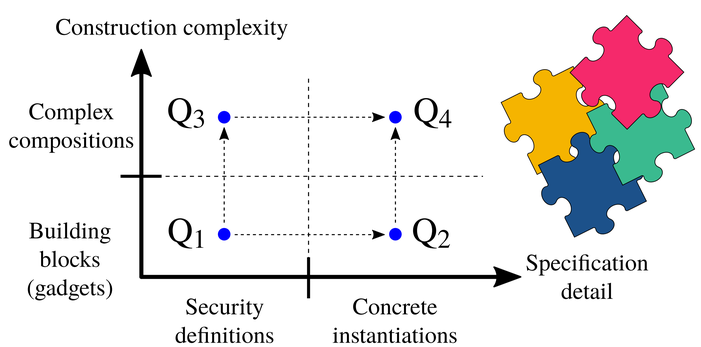
- **Interchangeable:** the input/output interface does not change
- **Shared-I/O:** the **i**nput and/or the **o**utput are secret-shared
- **Auditable:** can prove the output was produced by a threshold entity

Important considerations

1. Focus on NIST-approved primitives
2. Advanced security features (composability, etc.)
3. Testing and validation; formal verification
4. Granularity (gadgets, ideal functionalities, ...)
5. Collaboration with stakeholders

The modularity challenge

Threshold schemes can be composed of several building blocks, with specifications ranging from security definitions to concrete mathematical objects.



The figure above is an abstract representation of the states and paths of the evolution process from conceptual building blocks to concrete complex threshold schemes.

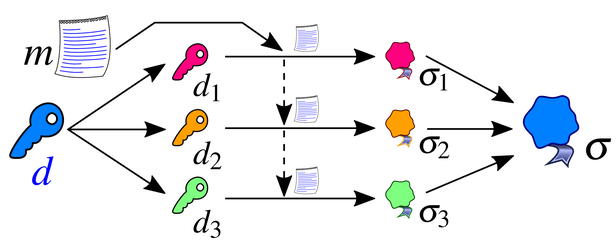
Motivating applications

The standardization of threshold schemes for cryptographic primitives can be useful for myriad potential applications.

1. **Secrets protected at rest** (e.g., for high-value signature keys)
2. **Confidential communication** (e.g., via shared-O decryption)
3. **Distributed key generation** (e.g., to avoid *dealers*)
4. **Leakage-resistant hardware** (e.g., via threshold circuit design)
5. **Accountable transactions** (e.g., via multi-signatures)
6. **Password authentication** (e.g., via threshold hashing)
7. **Distributed computation** (e.g., across HSMs or VMs)

Example: Threshold n-out-of-n RSA signature

- **Traditional parameters:**
  - **Public:** modulus $N = p \times q$; verification key e.
  - **Private:** signing key d; group order $\phi = p-1 \times q-1$
- **Secret-share the key d:** $d \to d_1, d_2, d_3$: $d_1 + d_2 + d_3 = d \bmod \phi$
- **Produce partial signatures:** $\sigma_i = m^{d_i}$, for $i = 1, 2, 3$
- **Obtain final signature:** $\sigma = \sigma_1 \sigma_2 \sigma_3 = m^{d_1+d_2+d_3} = m^d \bmod \phi$



Legend: AES (Advanced Encryption Standard); CA (Certification Authority); ECC (Elliptic-Curve Cryptography); ECDSA (Elliptic-Curve Digital Signature Algorithm); EdDSA (Edwards Curve Digital Signature Algorithm); HSM (Hardware Security Module); I/O (Input/Output); NISTIR (NIST Internal Report); NTCW (NIST Threshold Cryptography Workshop); RSA (Rivest-Shamir-Adleman); VM (Virtual Machine).

Threshold Cryptography contact: threshold-crypto@nist.gov

Webpage: `https://csrc.nist.gov/Projects/Threshold-Cryptography`

Poster produced for: NIST-ITL Science Day 2019 — November 06, 2019 (Gaithersburg, USA)