Distributed Key Management and Cryptographic Agility

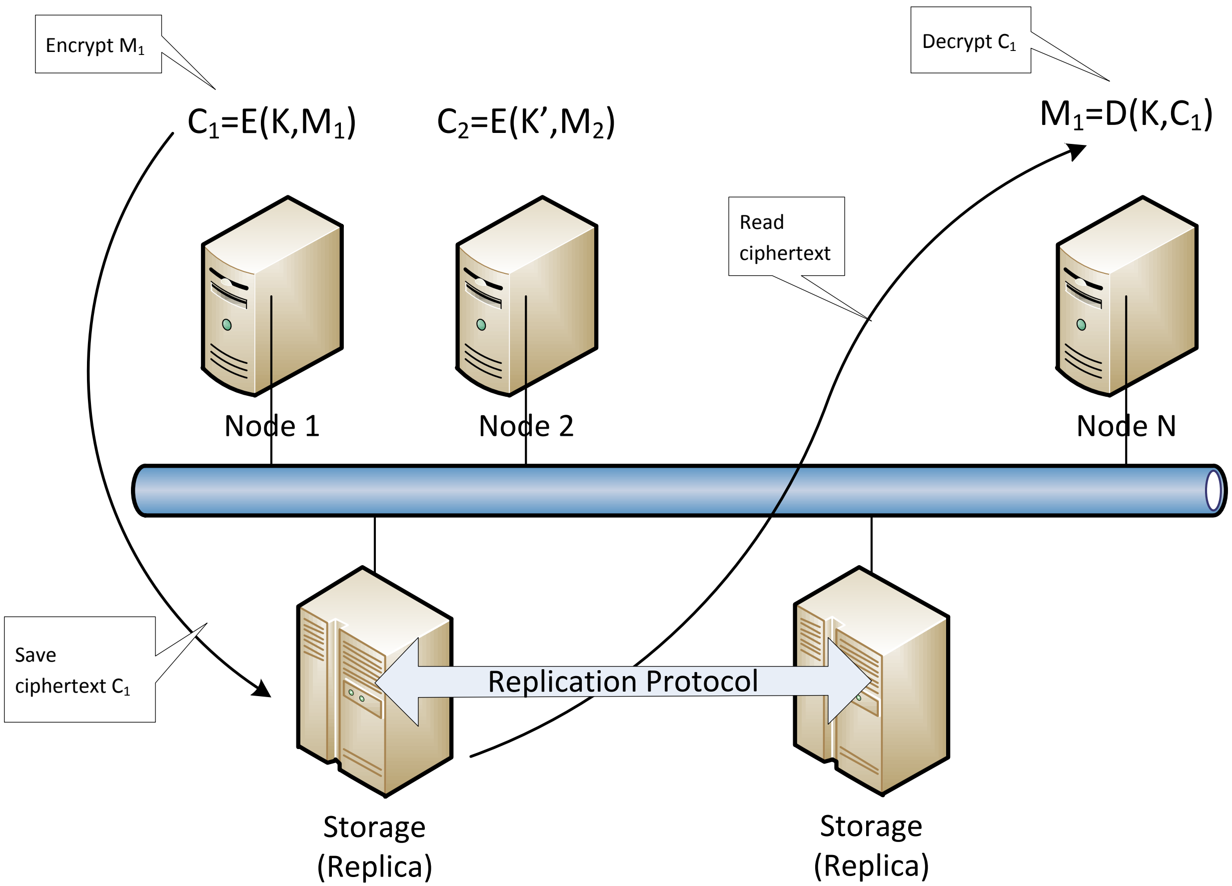
Tolga Acar

24 Feb. 2011

Overview

- Distributed Key Lifecycle
  - Problem statement and status quo
  - Distributed Key Manager
  - Typical application scenario and architecture
- Hardware Rooted Key Management
  - How to use TPMs for key management
  - TPM Key hierarchy
- Diving into Cryptographic Theory
  - Security Definitions
  - Cryptographic Agility

Distributed Key Management

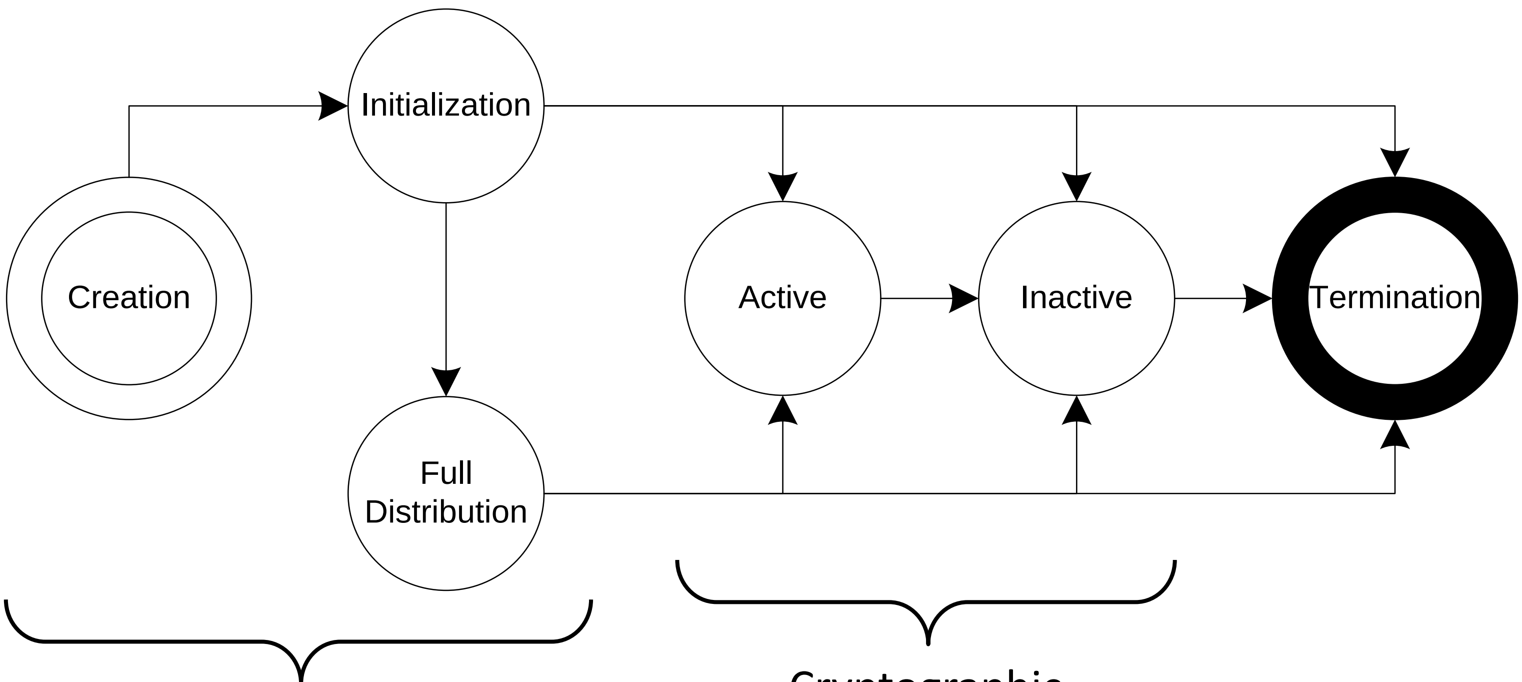


Where is the correct key?
How is it protected?

Key Lifecycle Model

- **Creation**. A key object is created on at least one replica, but its attributes (e.g., key value) are not set.
- **Initialization**. The key object has all its core key attributes set on at least one replica.
- **Full Distribution**. An initialized key is available on all replicas.
- **Active**. An initialized key is available for cryptographic operations on at least one replica.
- **Inactive**. An initialized key is available for some cryptographic operations on all replicas (e.g., decrypt, only).
- **Termination**. An initialized key is permanently deleted from all replicas.

Key State Transitions



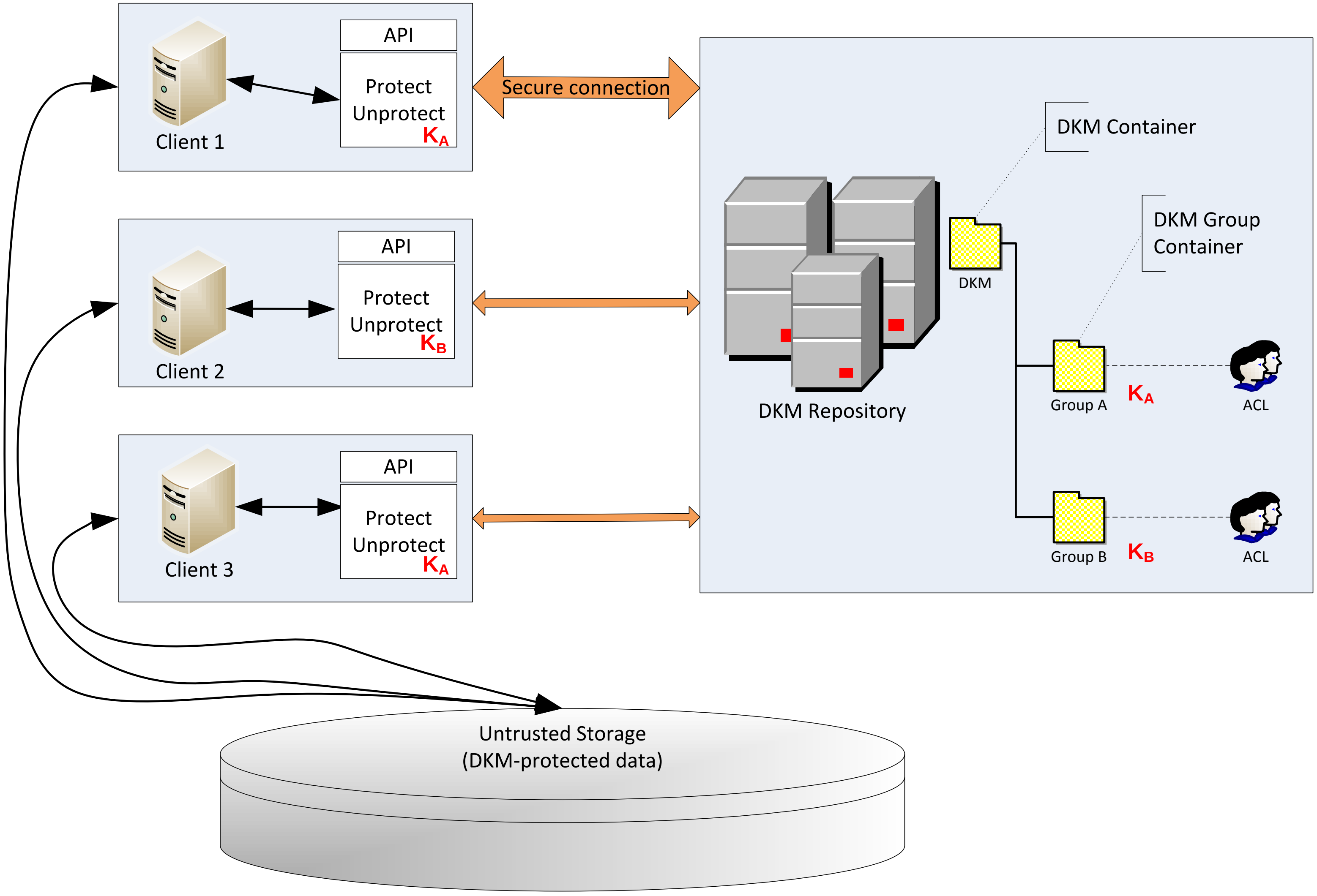
Create and
initialize a key

Cryptographic
operations

DKM Problem Statement

- No cross-user and cross-machine data protection
  - Windows Data Protection API (DPAPI) is single-user, single-machine.
  - KeyCzar and PKCS#11 uses local keys; no distribution mechanism.
- Engineering problem
  - Ad-hoc key management groups (protection siloes)
  - Scalability & Availability (10Ks of machines)
  - Geo-redundancy (multiple data centers)
  - Key lifecycle management (automation)
- Cryptography problem
  - Protect arbitrary data (broad applicability)
  - Use existing algorithms (e.g. AES, HMAC-SHA2)
  - Automatically update group keys (key rollover)
  - Crypto agile (algorithm and key length changes)

DKM Architecture



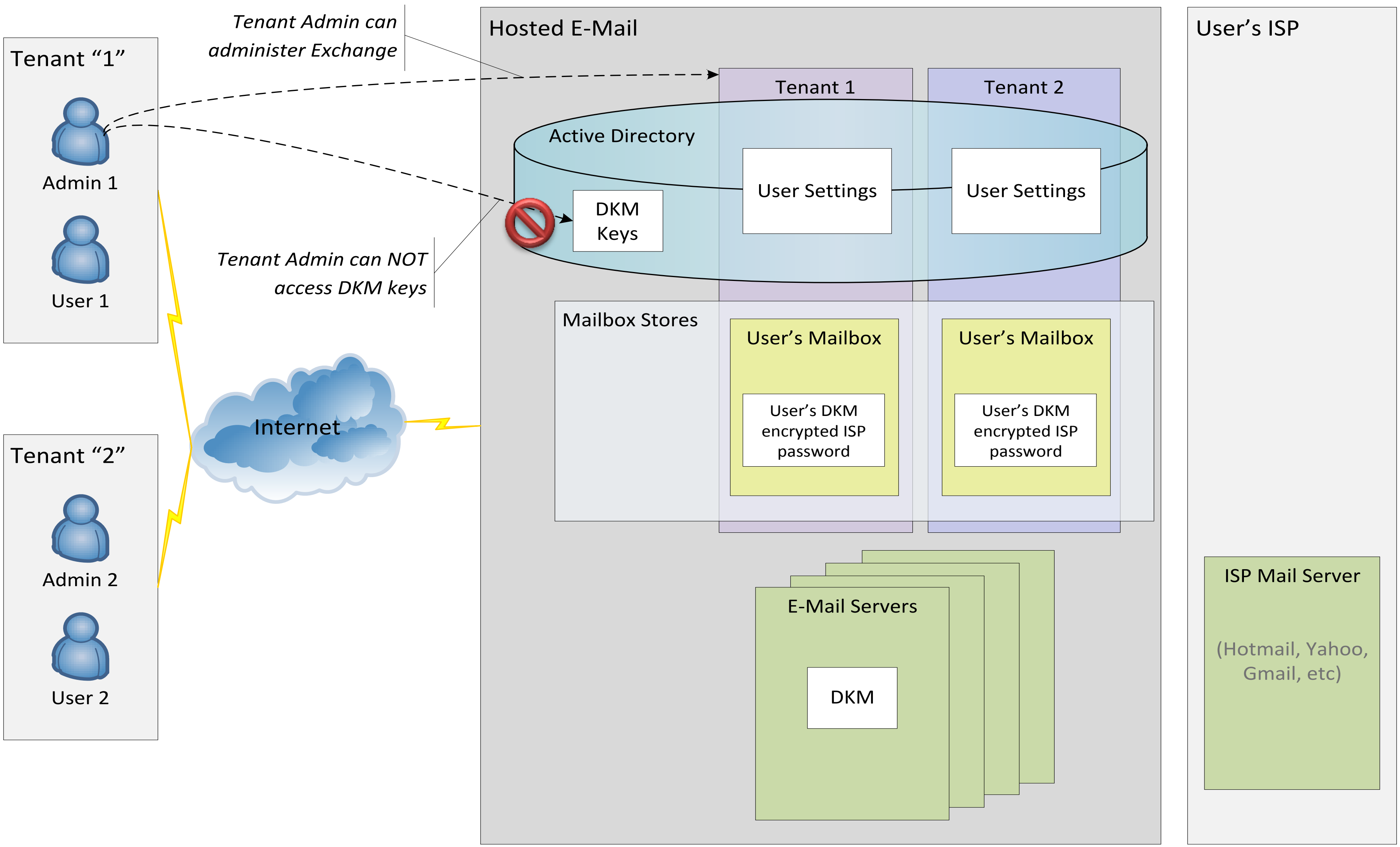
DKM Approach

- Active Directory Approach
  - Key storage is straightforward
    - Store group keys in AD objects
    - Protect keys with AD object ACLs
    - AD security groups correspond to principals / groups
  - Rely on Active Directory replication for high availability
  - Network transport is secure (LDAP with Kerberos)
- DKM provides
  - Auto key update mechanism
  - Multiple groups and multiple keys per group
  - Cryptographic policy per domain and per group
  - Crypto agility

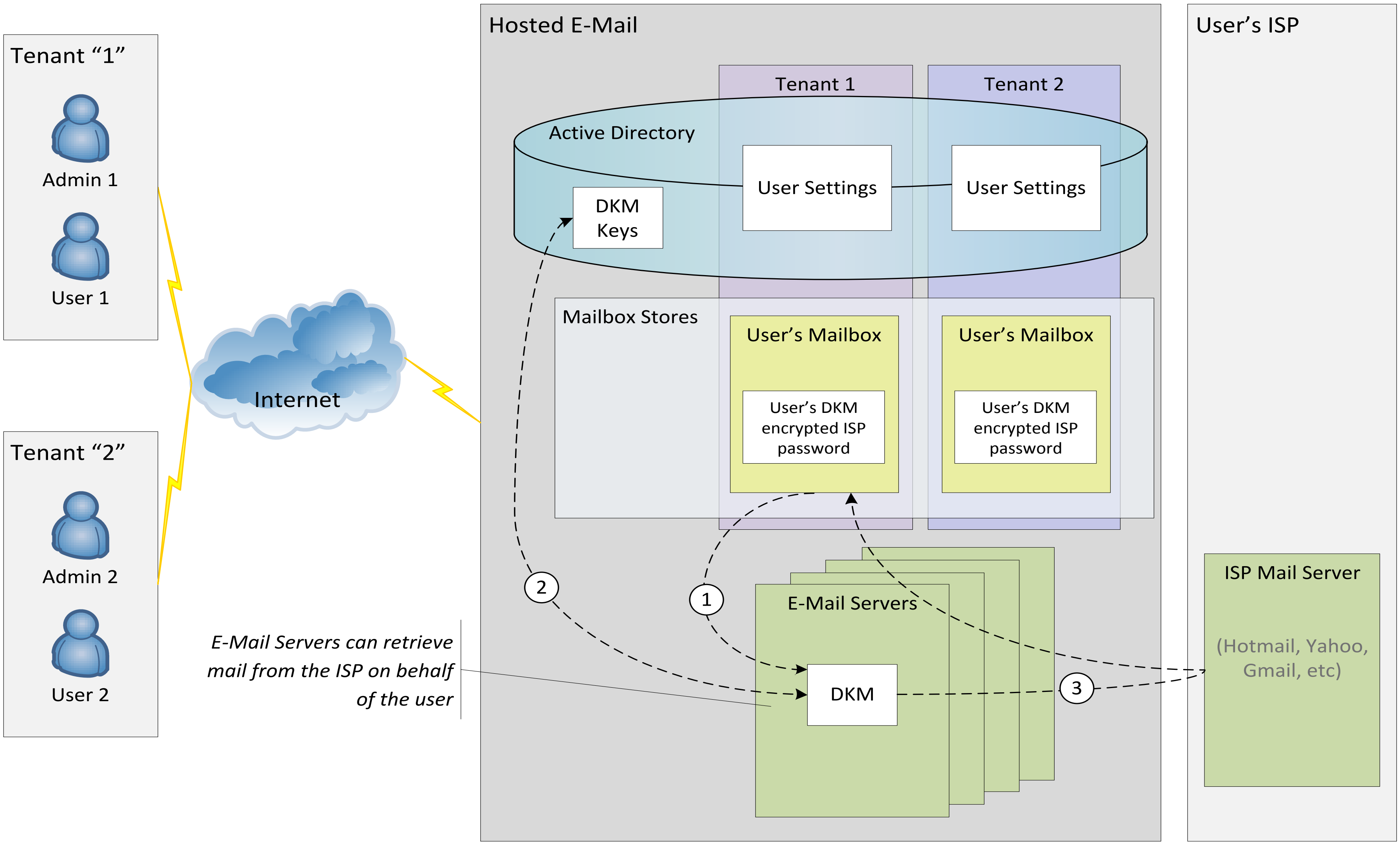
Walkthrough: DKM in Hosted E-Mail

- Scenario:
  - Hosting mail for multiple tenants in a datacenter
  - Product supports message aggregation from other providers for users with multiple email accounts
    - User signs in once
    - E-Mail Server fetches and aggregates mail
  - Tenant Admins must be able to perform Administrative tasks
    - But should NOT be able to read user credentials

Walkthrough: DKM in Hosted E-Mail



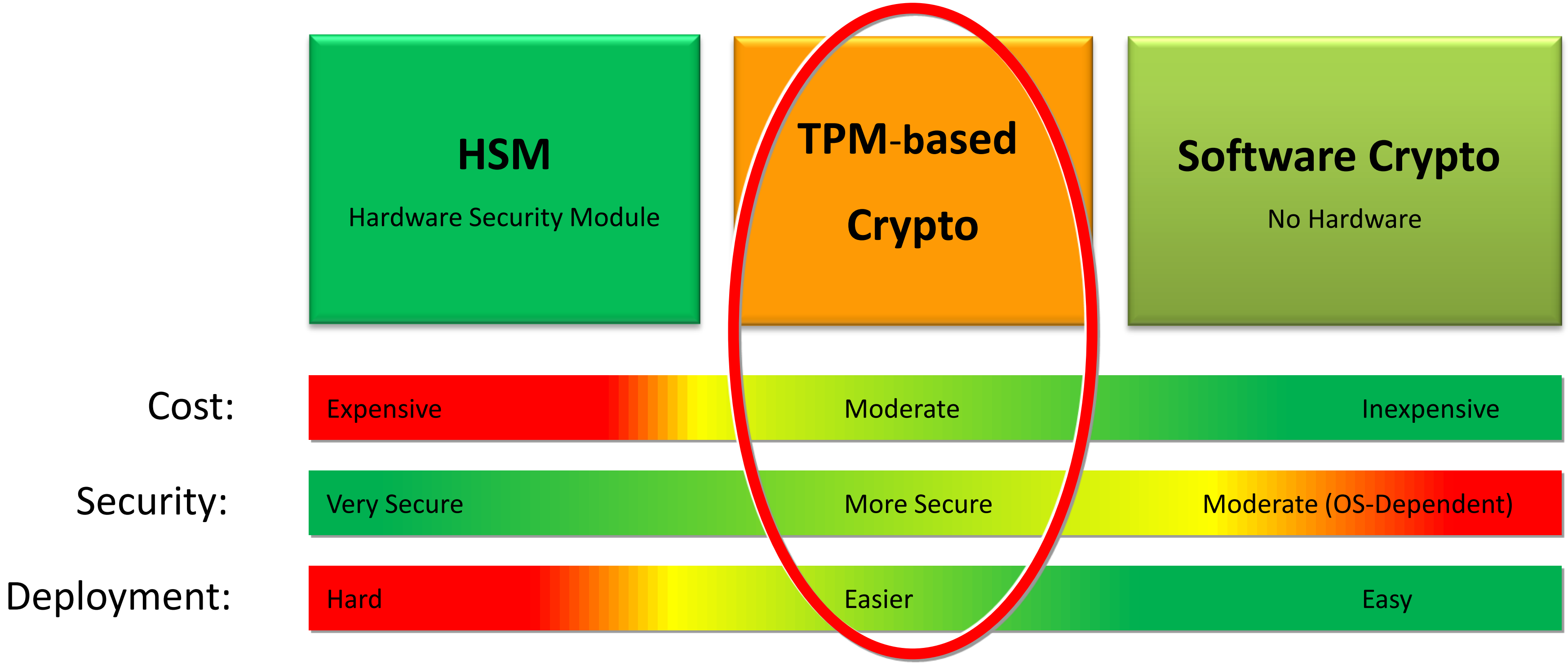
DKM in Hosted E-Mail



E-Mail Servers can retrieve mail from the ISP on behalf of the user

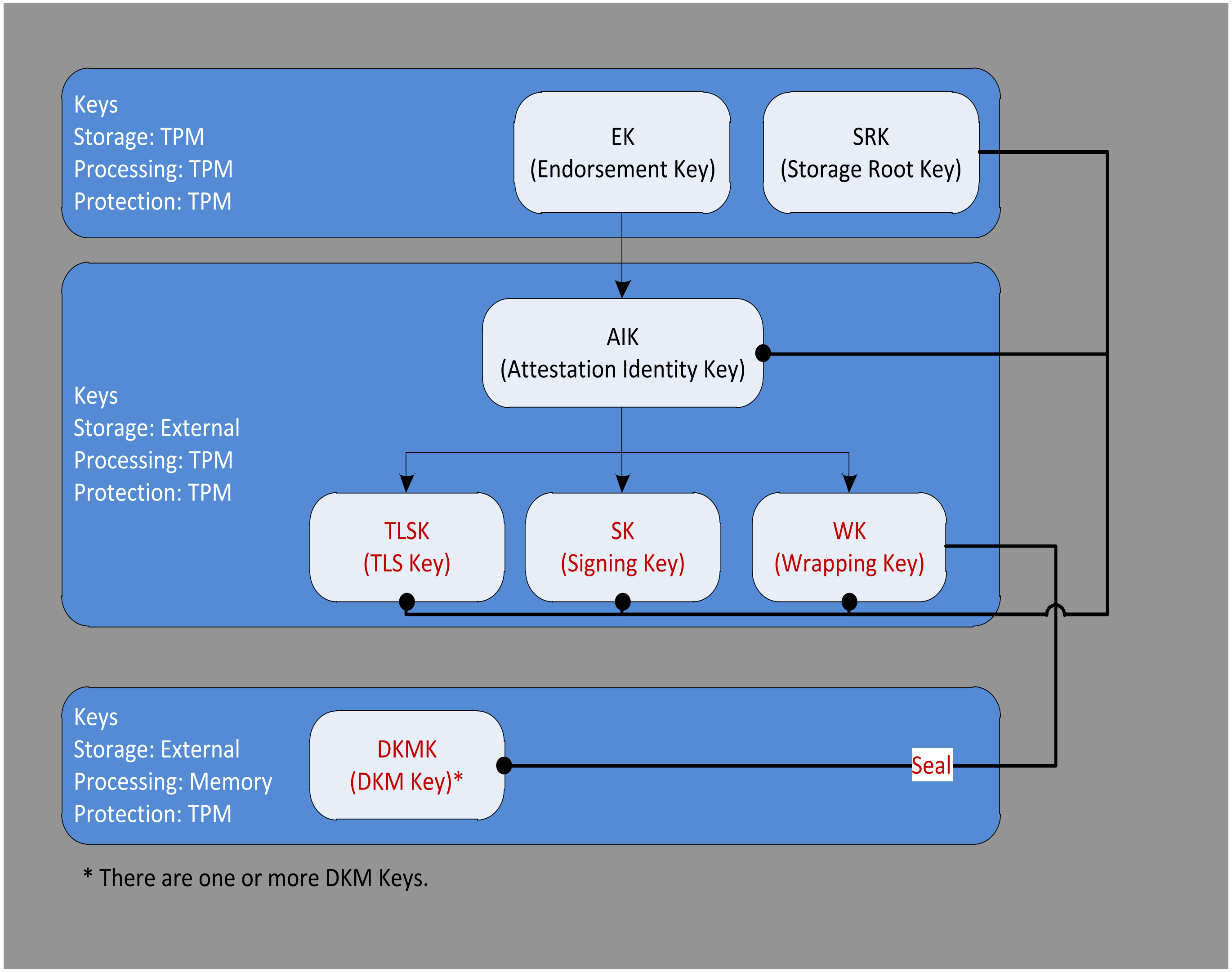
DKM-TPM Motivation

Secret Protection Technology:



- Approach sits between a pure HSM solution and a full software solution.

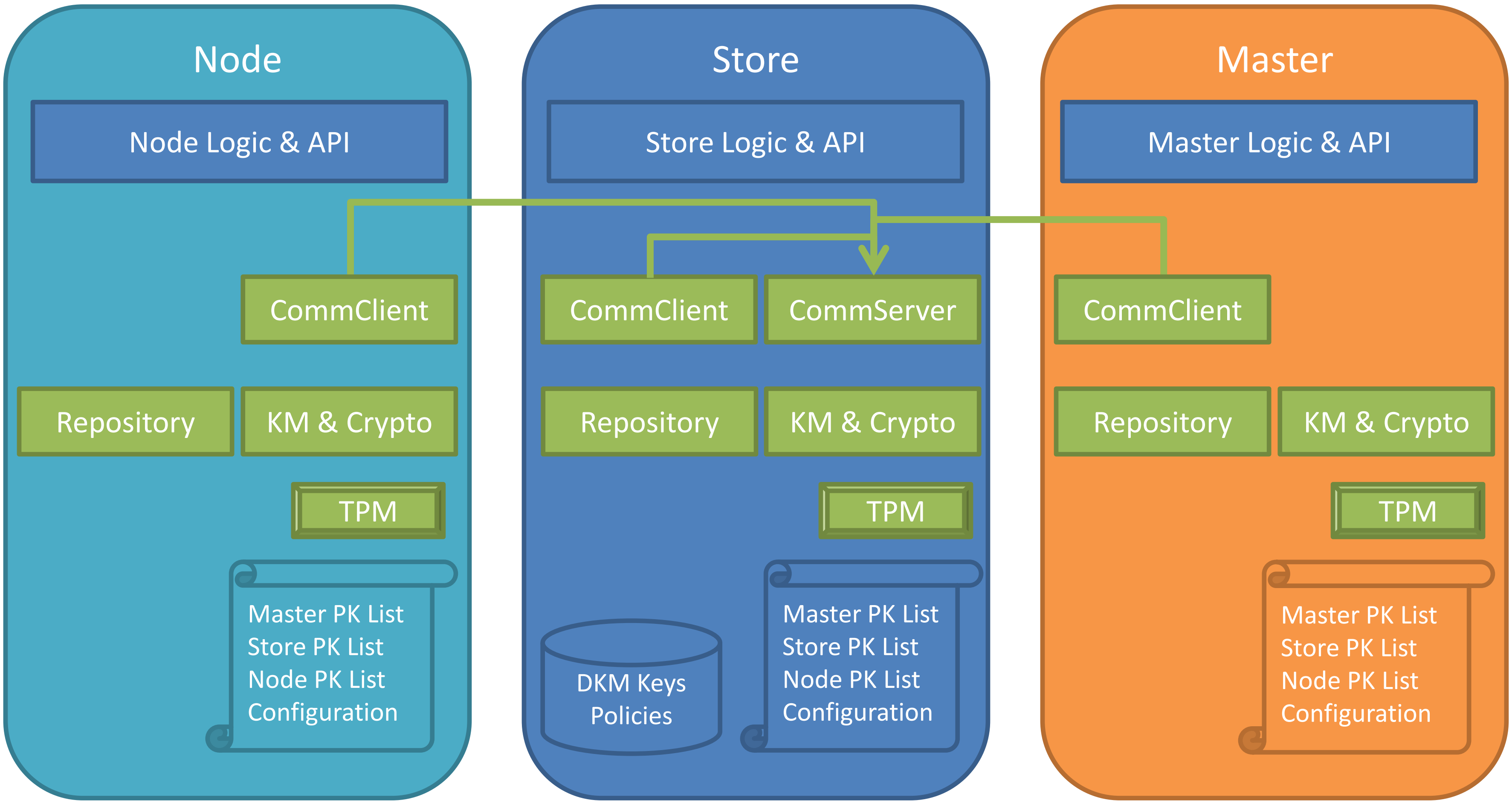
DKM-TPM Key Hierarchy



DKM-TPM Roles

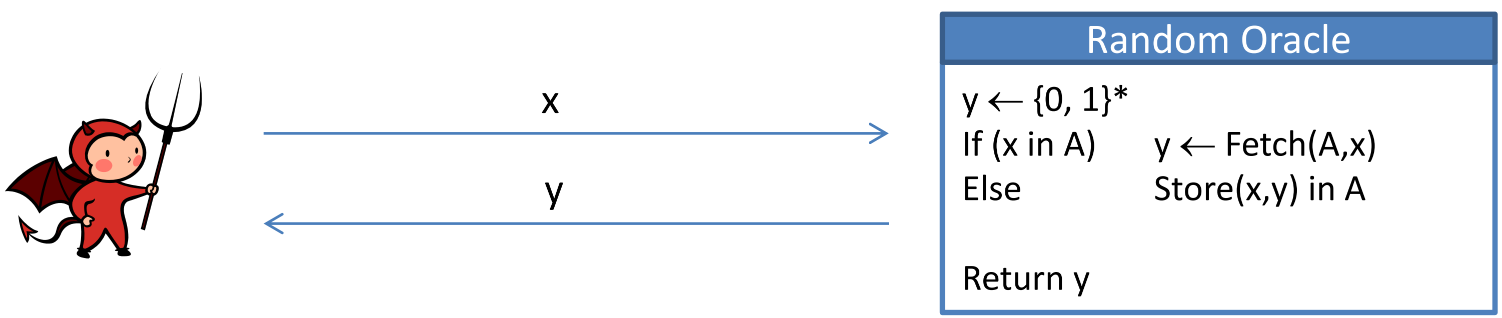
1. Master (Root of Trust)
    - Root of Trust for TPM public keys
    - Role assignment to TPM public keys
    - Push to Stores
2. Store (Repositories)
    - DKM repository (keys, policies, and metadata)
    - DKM Responder
    - Responds to requests from Masters, Stores, and Nodes
3. Node (Application servers)
    - Cryptographic operations with DKM keys
    - Client API
    - Sends requests to Stores

DKM-TPM Roles



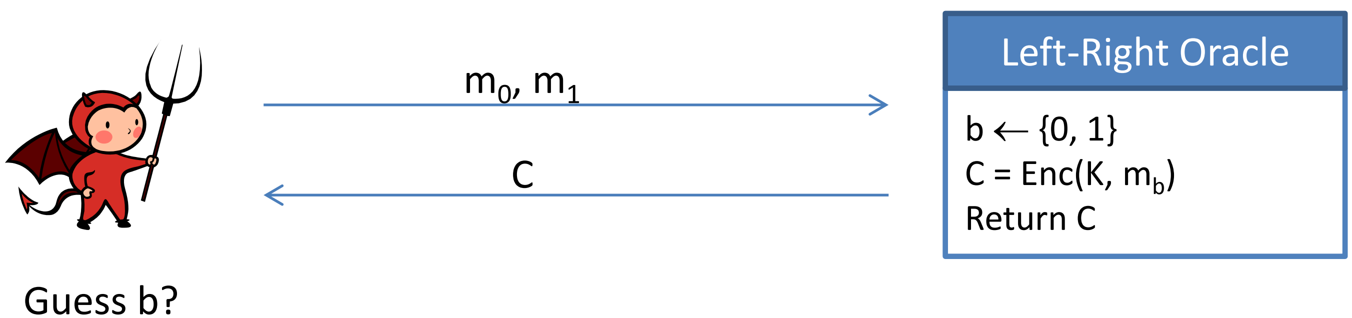
Cryptosystem Security Definitions

- Probabilistic Polynomial-Time (PPT) adversaries
  - Probabilistic randomized algorithm that gives the correct answer with > ½ probability.
- Random Oracle Model (RO or ROM)
  - Black box with a stateful uniform random response



Attack Game

- Encryption scheme security definitions
  - IND-R: Indistinguishability from Random
  - IND-CPA: Indistinguishability under Chosen Plaintext Attack (a.k.a. semantic security)
  - IND-CCA: Indistinguishability under Chosen Ciphertext Attack
- IND-CPA $\subset$ IND-CCA



IND-CPA Game

Ciphertext Attacks

- IND-CCA2: Indistinguishability under adaptive chosen ciphertext attack
  - Decryption Oracle access (non-trivial)
- Non-adaptive
  - Query the decryption oracle till the challenge ciphertext is received
- Adaptive
  - Continuous queries to the oracle (max q queries)
- IND-CPA $\subset$ IND-CCA $\subset$ IND-CCA2

IND-CCA/CCA2 Game

Free Oracle Access

Queries {m,C} →

← Responses {C,m}

Encrypt

C = Enc(K, m)

Decrypt

m = Dec(K, C)

Challenge

m_0, m_1 →

← C

Left-Right Oracle

$b \leftarrow \{0, 1\}$

$C = Enc(K, m_b)$

Adaptive (CCA2) Adversary

C →

← m

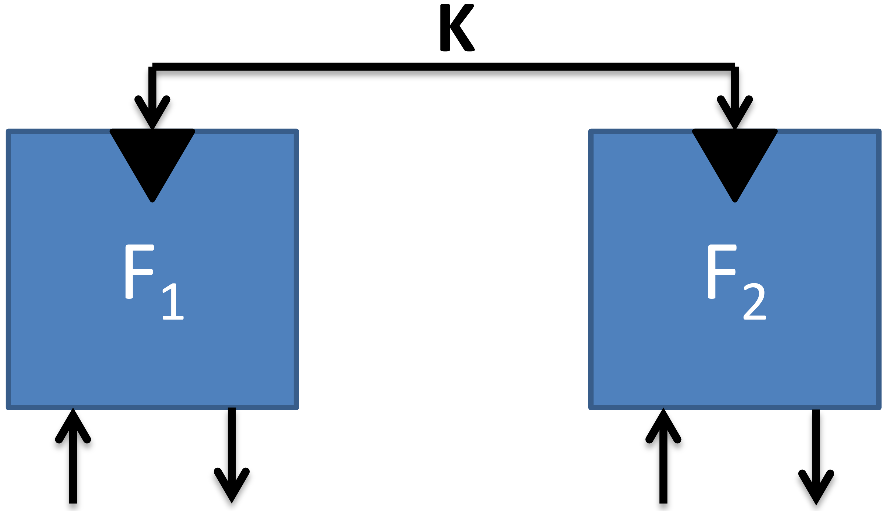
Decrypt

m = Dec(K, C)

Guess b?

Cryptographic Agility

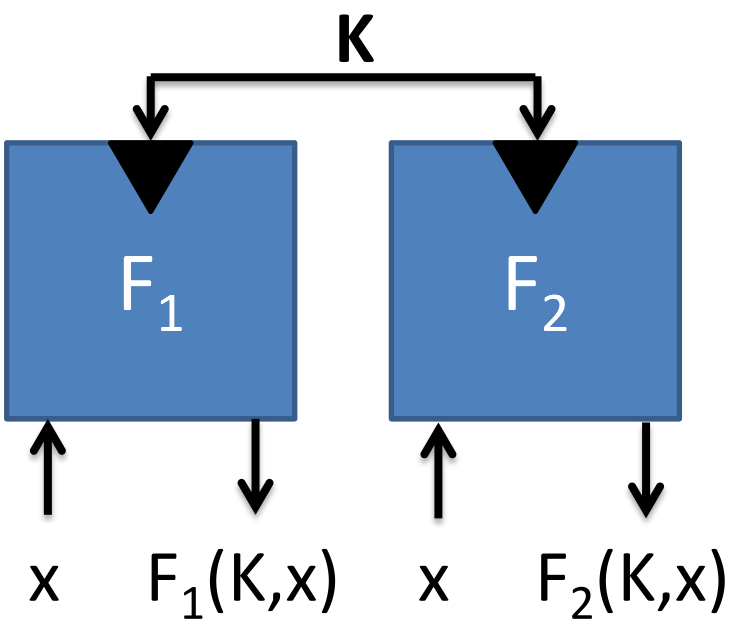
- Cryptographic primitives as sets:
  - PRF = {F : F is a secure pseudorandom function}
  - AE = {F : F is a secure authenticated encryption scheme}
- Assume F_1 and F_2 have the same key space and length
- **Informal Definition**: A primitive Π is **agile** if any $F_1, F_2 \in \Pi$ can securely use the **same** key.



Pseudo Random Function Agility

Facts

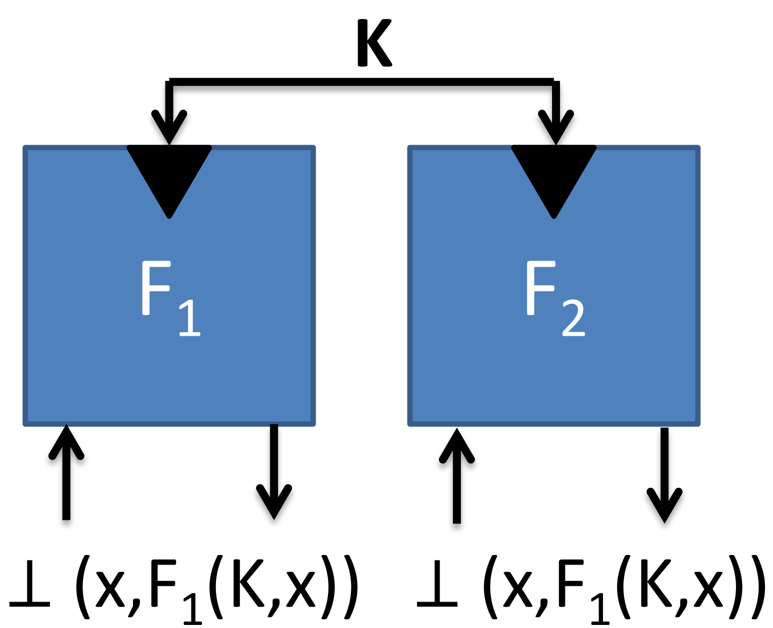
- PRF: F is a PRF if no efficient adversary can distinguish F(K,.) from a random function.
- $F_1(K_1,x)$ and $F_2(K_2,x)$ are not distinguishable from a pair of random functions.



- **Definition:** A set $\{F_1,F_2\}$ is **agile** if $F_1(K,x)$ and $F_2(K,x)$ are not distinguishable from a pair of random functions.
- Question: Are PRFs agile?
  - Yes, if every $\{F_1,F_2\}$ is agile.
- Answer: No.
  - Example: $F_2(K,x)$ = NOT $(F_1(K,x))$
- Now, what?

Agility in Practice

- Certain primitives are agile: collision-resistant hash functions
- Strong agility is achievable in practice: Authenticated Encryption
  - Don’t use the key directly in the encryption algorithm <ae>
  - Use a derived subkey in <ae>



- PRF-based security for Authenticated Encryption: CCM, GCM, etc.
  - Pick a PRF from a small agile set
- Encryption of M with K, with PRF
  - K_{ae} = PRF(K,<ae>)
  - C = E(K_{ae}, M)
- Decryption
  - K_{ae} = PRF(K,<ae>)
  - M = D(K_{ae}, C)