Integrity, Digital Signatures & Key Management

CS 161: Computer Security

Prof. Vern Paxson

TAs: Jethro Beekman, Mobin Javed,
Antonio Lupher, Paul Pearce
& Matthias Vallentin

http://inst.eecs.berkeley.edu/~cs161/

March 21, 2013

Goals For Today

- Revisit MACs: symmetric-key message integrity
  - And implicit message authentication
- *Digital signatures*
  - Public-key integrity + authentication
  - Includes "non-repudiation" property similar to physical signatures
- Managing keys: certificates
  - Enables bootstrapping of trust
- *Leap-of-Faith* key management (time permitting)

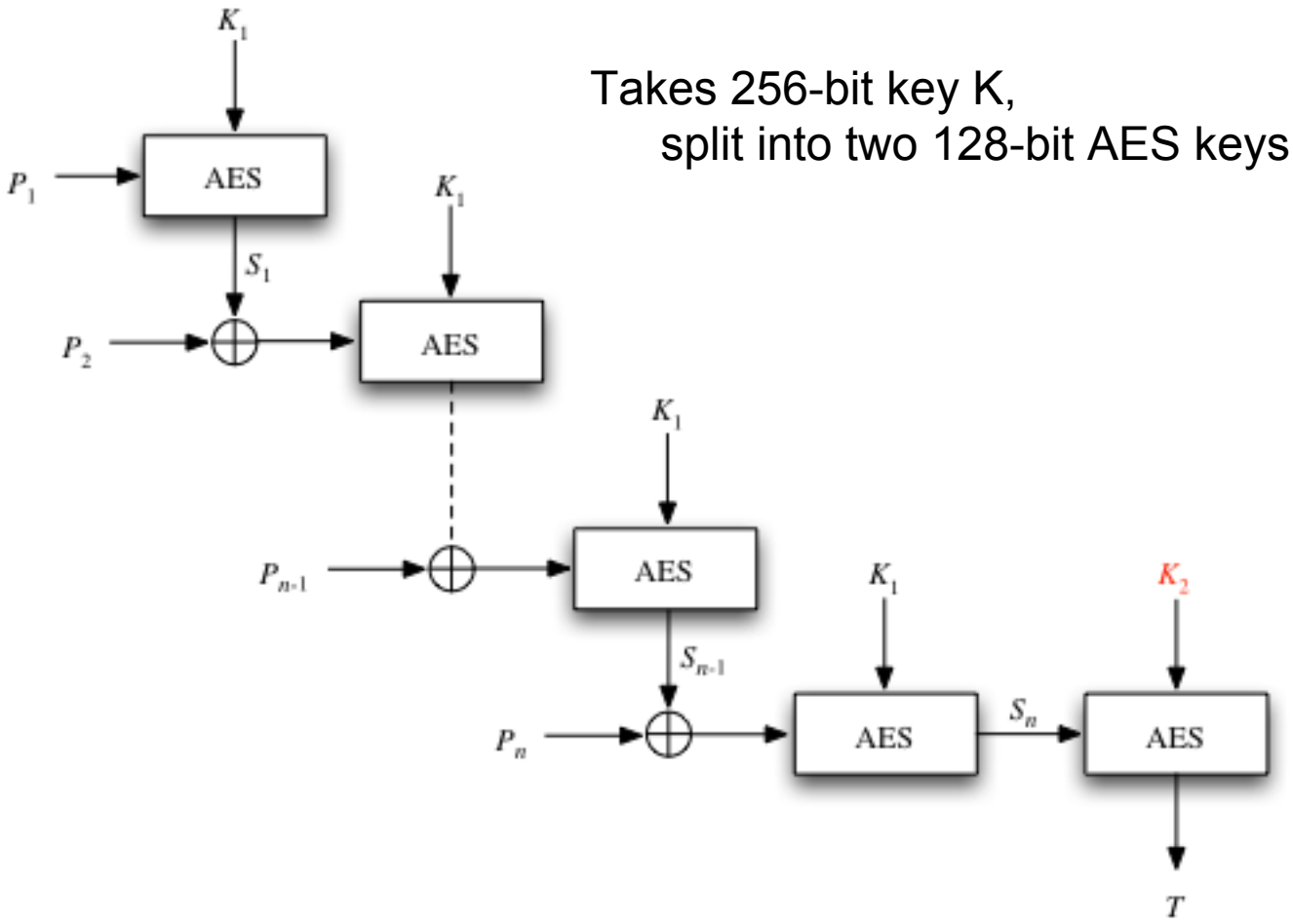
Message Authentication Codes (MACs)

- Symmetric-key approach for integrity
  - Uses a shared (secret) key K
- Goal: when Bob receives a message, can confidently determine it hasn't been altered
  - In addition, whomever sent it *must have possessed* K (⇒ message authentication)
- Conceptual approach:
  - Alice sends {M, T} to Bob, with tag T = F(K, M)
    - Note, M could instead be C = $E_{K'}$(M), but not required
  - When Bob receives {M', T'}, Bob checks whether T' = F(K, M')
    - If so, Bob concludes message untampered, came from Alice
    - If not, Bob discards message as tampered/corrupted

Requirements for Secure MAC Functions

- Suppose MITM attacker *Mallory* intercepts Alice's {M, T} transmission …
  - … and wants to replace M with altered M*
  - … but doesn't know secret key K
- We have secure integrity if MAC function T = F(M, K) has two properties:
  1. Mallory can't compute T* = F(M*, K)
     - Otherwise, could send Bob {M*, T*} and fool him
  2. Mallory can't find M** such that F(M**, K) = T
     - Otherwise, could send Bob {M**, T} and fool him

Example of a Secure MAC Function: AES-EMAC



Takes 256-bit key K,
split into two 128-bit AES keys, K_1 and K_2

(Note, a tad simpler than one popular algorithm, AES-CMAC)