Hashes & MAC. Digital Signatures

Lecture 16

One-time MAC

With 2-Universal Hash Functions

- Trivial (very inefficient) solution (to sign a single n bit message):
  - Key: 2n random strings (each k-bit long) $(r^i_0, r^i_1)_{i=1..n}$
  - Signature for $m_1 \ldots m_n$ be $(r^i_{m_i})_{i=1..n}$
  - Negligible probability that Eve can produce a signature on $m' \neq m$

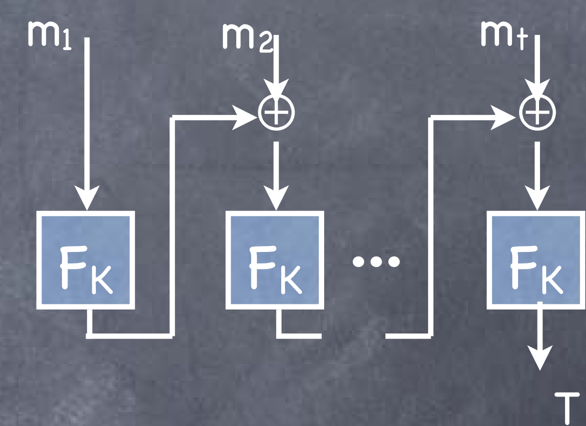
r^1_0	r^2_0	r^3_0
r^1_1	r^2_1	r^3_1

- A much more efficient solution, using 2-UHF (and still no computational assumptions):
  - $\text{Onetime-MAC}_h(M) = h(M)$, where $h \leftarrow \mathcal{H}$, and $\mathcal{H}$ is a 2-UHF
    - Seeing hash of one input gives no information on hash of another value

MAC

With Combinatorial Hash Functions and PRF

- Recall: PRF is a MAC (on one-block messages)
- CBC-MAC: Extends to any fixed length domain



- **Alternate approach** (for fixed length domains):
    - $MAC_{K,h}^*(M) = PRF_K(h(M))$ where $h \leftarrow \mathcal{H}$, and $\mathcal{H}$ a combinatorial hash function (e.g. 2-UHF)

If truly random function, adversary only learns if hash collision occurred or not (h nor h(M) revealed).

Combinatorial hash ⇒ Unlikely collision ever occurs

Finite domain

MAC

With Cryptographic Hash Functions

- A proper MAC must work on inputs of variable length
- Recall: making CBC-MAC work securely with variable input-length.
  - Derive K as $F_{K'}(t)$, where t is the number of blocks
  - Or, Use first block to specify number of blocks
  - Or, output not the last tag T, but $F_{K'}(T)$, where K' an independent key (EMAC)
  - Or, XOR last message block with another key K' (CMAC)
- Alternate idea: Leave variable input-lengths to the hash
  - But combinatorial hash functions worked with a fixed domain
  - Will use a cryptographic hash function
- $MAC^*_{K,h}(M) = MAC_K(h(M))$ where $h \leftarrow \mathcal{H}$, and $\mathcal{H}$ a weak-CRHF
  - Weak-CRHFs can be based on OWF. Or, can be more efficiently constructed from fixed input-length MACs

h(M) may be revealed, but only oracle access to h

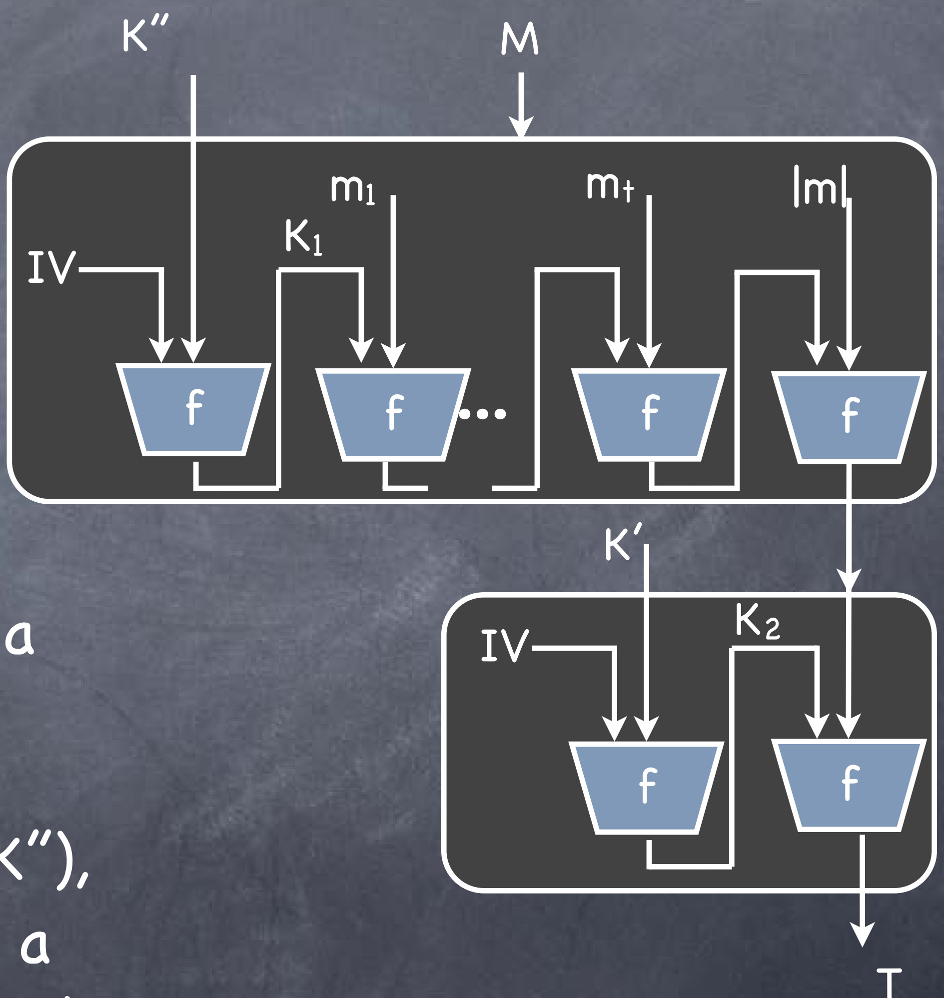
MAC

With Cryptographic Hash Functions

- $MAC^*_{K,h}(M) = MAC_K(h(M))$ where $h \leftarrow \mathcal{H}$, and $\mathcal{H}$ a weak-CRHF
  - Weak-CRHFs can be based on OWF. Or, can be more efficiently constructed from fixed input-length MACs.
- Unlike the domain extension (to fixed length domain) using 2-UHF, or CBC-MAC, this doesn't rely on pseudorandomness of MAC
  - Works with any one-block MAC (not just a PRF based MAC)
  - Could avoid "export restrictions" by not being a PRF
  - Candidate fixed input-length MACs: compression functions (with key as IV)
    - Recall: Compression functions used in Merkle-Damgård iterated hash functions

HMAC

- HMAC: Hash-based MAC
- Essentially built from a compression function f
  - If keys K_1, K_2 independent (called NMAC), then secure MAC if: f is a fixed input-length MAC & the Merkle-Damgård iterated-hash is a weak-CRHF
  - In HMAC (K_1,K_2) derived from (K′,K″), in turn heuristically derived from a single key K. If f is a (weak kind of) PRF K_1, K_2 can be considered independent



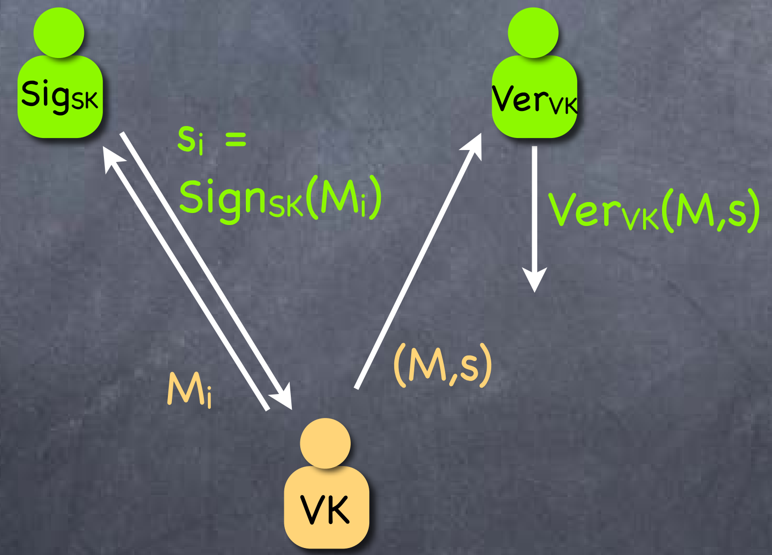
Hash Not a Random Oracle!

- Hash functions are no substitute for RO, especially if built using iterated-hashing (even if the compression function was to be modeled as an RO)
- If H is a Random Oracle, then just H(K||M) will be a MAC
  - But if H is a Merkle-Damgård iterated-hash function, then there is a simple length-extension attack for forgery
    - (That attack can be fixed by preventing extension: prefix-free encoding)
  - Other suggestions like SHA1(M||K), SHA1(K||M||K) all turned out to be flawed too (even before breaking SHA1)

Digital Signatures

Digital Signatures

- Syntax: KeyGen, $Sign_{SK}$ and $Verify_{VK}$.
  Security: Same experiment as MAC's, but adversary given VK



Advantage = Pr[$Ver_{VK}(M,s)=1$ and $(M,s) \notin \{(M_i,s_i)\}$]

Digital Signatures

- Syntax: KeyGen, Sign_{SK} and Verify_{VK}.
  Security: Same experiment as MAC's, but adversary given VK
- Secure digital signatures using OWF, UOWHF and PRF
  - Hence, from OWF alone (more efficiently from OWP)
- More efficient using CRHF instead of UOWHF
- Even more efficient based on (strong) number-theoretic assumptions
  - e.g. Cramer-Shoup Signature based on "Strong RSA assumption"
- Efficient schemes secure in the Random Oracle Model
  - e.g. RSA-PSS in RSA Standard PKCS#1