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):

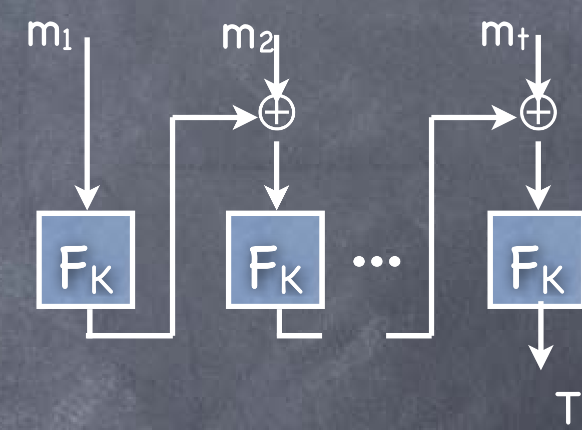
r^1_0	r^2_0	r^3_0
r^1_1	r^2_1	r^3_1

  - 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 $m' \neq m$
- 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 2-UHF

h(M) not revealed

MAC

With Cryptographic Hash Functions

- A proper MAC must work on inputs of variable length
- Can make 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)
- 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

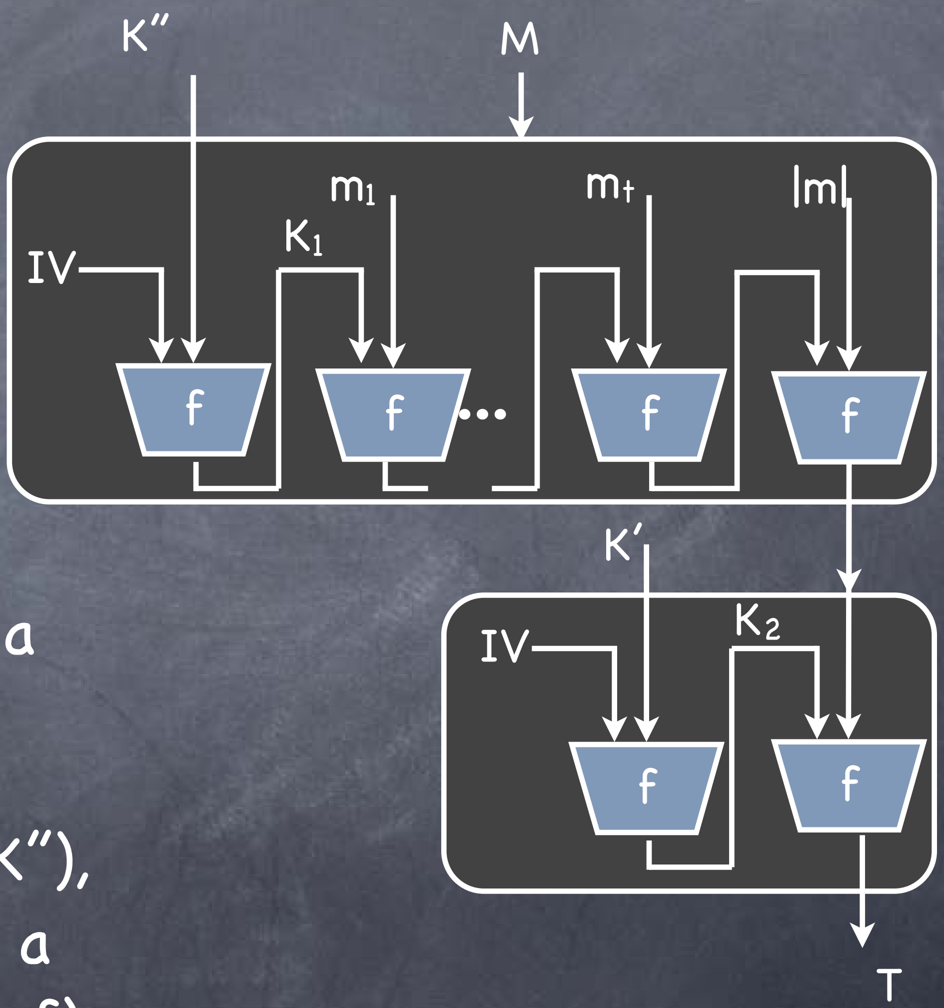
MAC

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
- 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

One-time Digital Signatures

Lamport's One-Time Signature

- Recall One-time MAC to sign a single n bit message
  - Shared secret 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 $(r^i_{m_i})_{i=1..n}$
- One-Time Digital Signature: Same signing key and signature, but VK= $(f(r^i_0), f(r^i_1))_{i=1..n}$ where f is a OWF
  - Verification applies f to signature elements and compares with VK
  - Security [Exercise]

$f(r^1_0)$	$f(r^2_0)$	$f(r^3_0)$
$f(r^1_1)$	$f(r^2_1)$	$f(r^3_1)$

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

Domain Extension of (One-time) Signatures

- Lamport's scheme has a fixed-length message (and SK/VK are much longer than the message)
- Hash-and-Sign domain extension for signatures
  - (If applied to one-time signature, still one-time, but with variable input-length)
  - Domain extension using a CRHF (not weak CRHF, unlike for MAC)
    - $\text{Sign*}_{SK,h}(M) = \text{Sign}_{SK}(h(M))$ where $h \leftarrow \mathcal{H}$ in both SK*,VK*
  - Can use UOWHF, with fresh h every time (included in signature)
    - $\text{Sign*}_{SK}(M) = (\, h, \text{Sign}_{SK}(h,h(M))\,)$ where $h \leftarrow \mathcal{H}$ picked by signer
- Using a "certificate chain/tree", can build a full-fledged signature scheme starting from one-time signatures (skipped)

More Efficient Signatures

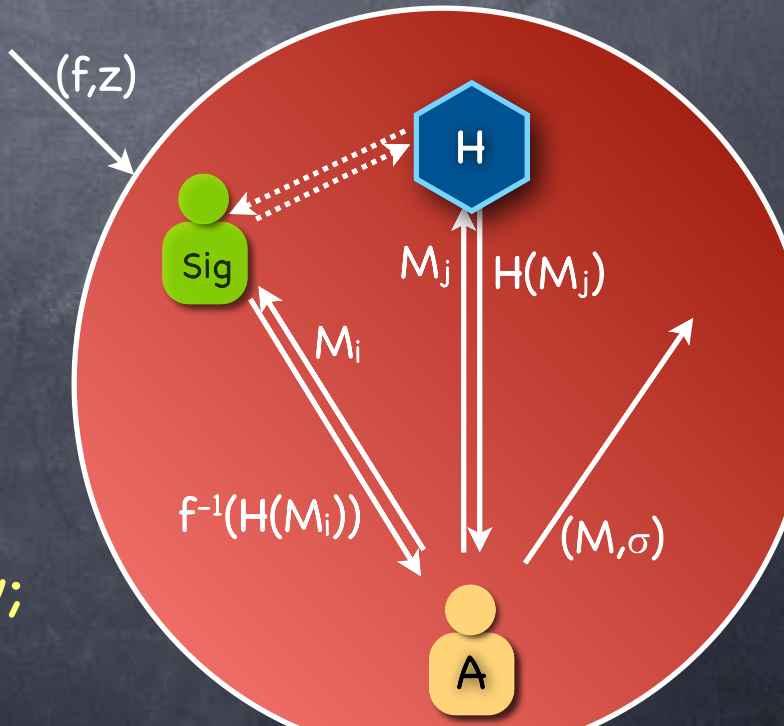
- Diffie-Hellman suggestion (heuristic): Sign(M) = $f^{-1}(M)$ where (SK,VK) = (f^{-1}, f), a Trapdoor OWP pair. Verify(M,σ) = 1 iff $f(\sigma)=M$.
  - Attack: pick σ, let $M=f(\sigma)$ (Existential forgery)
- Fix: **Sign(M) = f^{-1}(Hash(M))**
  - Secure? Adversary gets to choose M and hence Hash(M); so signing oracle gives adversary access to f^{-1} oracle. But Trapdoor OWP gives no guarantees when adversary is given f^{-1} oracle.
  - If Hash(.) modeled as a random oracle then adversary can't choose Hash(M), and effectively doesn't have access to f^{-1} oracle. Then indeed secure
  - "Standard schemes" like RSA-PSS are based on this

Proving Security in the RO Model

- To prove: If Trapdoor OWP secure, then Sign(M) = f^{-1}(Hash(M)) is a secure digital signature in the RO Model, with Hash modelled as a random oracle
  - Intuition: adversary only sees (x,f^{-1}(x)) where x is random, which it could have obtained anyway, by picking f^{-1}(x) first
- Modeling as an RO: RO randomly initialized to a random function H from $\{0,1\}^*$ to $\{0,1\}^k$
  - Signer and verifier (and forger) get oracle access to H(.)
  - All probabilities also over the initialization of the RO

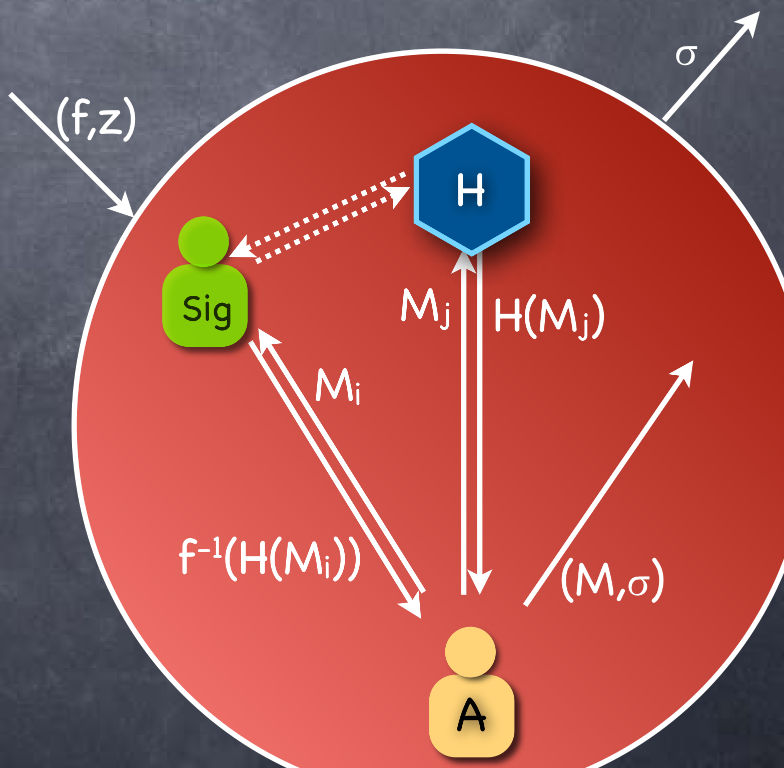
Proving Security in ROM

- Reduction: If A forges signature (where $Sign(M) = f^{-1}(H(M))$ with (f,f^{-1}) from Trapdoor OWP and H an RO), then A* that can break Trapdoor OWP (i.e., given just f, and a random challenge z, can find $f^{-1}(z)$ w.n.n.p). A*(f,z) runs A internally.
  - A expects f, access to the RO and a signing oracle $f^{-1}(Hash(.))$ and outputs (M,σ) as forgery
  - A* can implement RO: a random response to each new query!
  - A* gets f, but doesn't have f^{-1} to sign
    - But $x = H(M)$ is a random value that A* can pick!
    - A* picks H(M) as $x=f(y)$ for random y; then $Sign(M) = f^{-1}(x) = y$



Proving Security in ROM

- A^* s.t. if A forges signature, then A^* can break Trapdoor OWP
  - A^* implements H and Sign: For each new M queried to H (including by Sign), A^* sets H(M)=f(y) for random y; Sign(M) = y
  - But A^* should force A to invert z
    - For a random (new) query M (say t^{th}) A^* sets H(M)=z
      - Here queries include the "last query" to H, i.e., the one for verifying the forgery (may or may not be a new query)
  - Given a bound q on the number of queries that A makes to Sign/H, with probability 1/q, A^* would have set H(M)=z, where M is the message in the forgery
    - In that case forgery ⇒ $\sigma = f^{-1}(z)$



Schnorr Signature

- Public parameters: (G,g) where G is a prime-order group and g a generator, for which DLA holds, and a random oracle H
  - Or (G,g) can be picked as part of key generation
- Signing Key: $y \in Z_q$ where G is of order q. Verification Key: $Y = g^y$
- $Sign_y(M) = (e,s)$ where $e = H(M||g^r)$ and $s = r-ye$, for a random r
- $Verify_Y(M,(e,s))$: Compute $R = g^s \cdot Y^e$ and check $e = H(M||R)$
- Secure in the Random Oracle model under the Discrete Log Assumption for a group
  - Alternately, under a heuristic model for the group (called the Generic Group Model), but under standard-model assumptions on the hash function