Hashes & MAC, Digital Signatures

Lecture 16

One-time MAC

With 2-Universal Hash Functions

RECALL

- Trivial (very inefficient) solution (to sign a single n bit message):

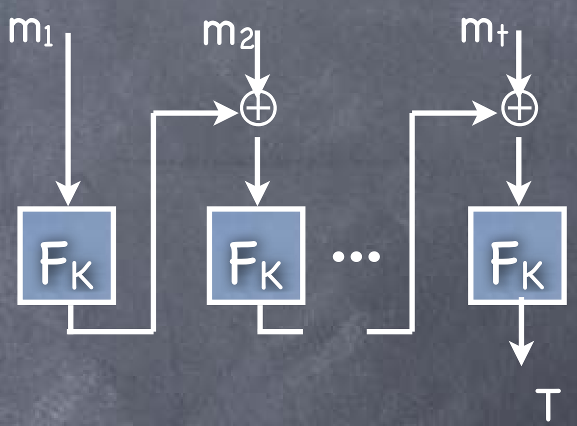
r^1_0	r^2_0	r^3_0
r^1_1	r^2_1	r^3_1

  - 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

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

  - Weak-CRHFs can be based on OWF. Or, can be more efficiently constructed from fixed input-length MACs

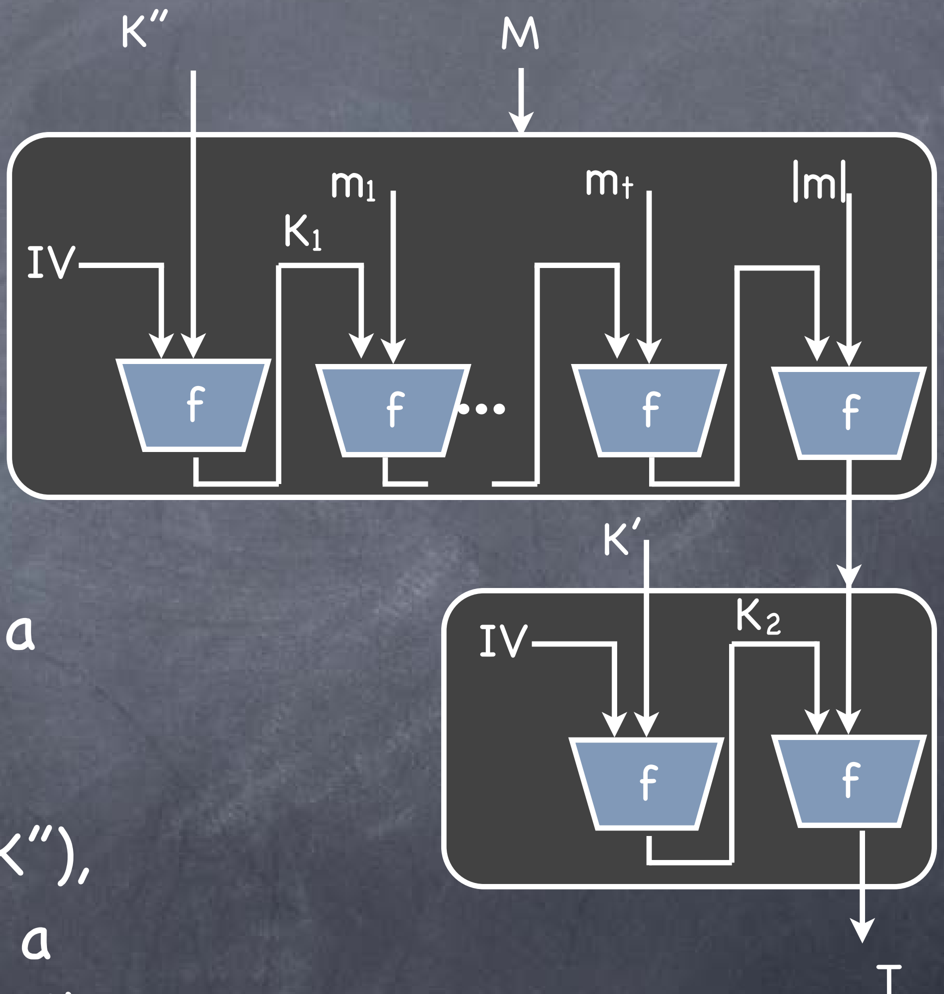
MAC

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



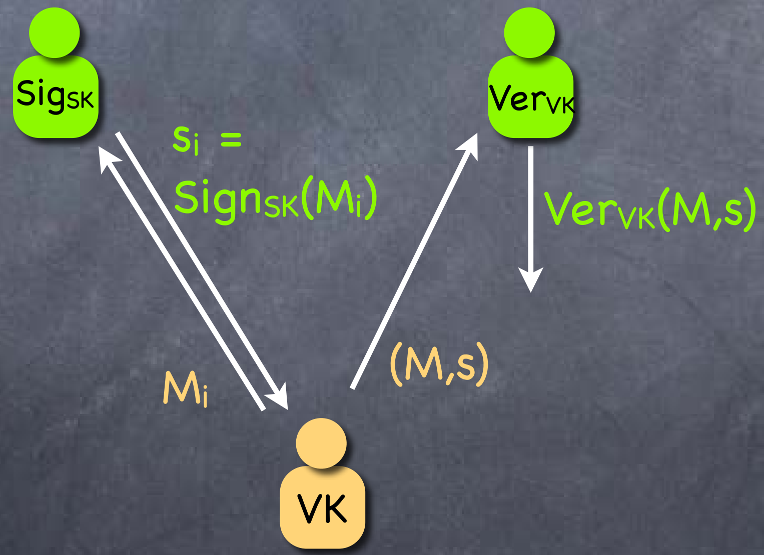
Hash Not a Random Oracle!

- 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 = $\Pr[\ \text{Ver}_{VK}(M,s)=1 \text{ and } (M,s) \notin \{(M_i,s_i)\}\ ]$

Weaker variant: Advantage = $\Pr[\ \text{Ver}_{VK}(M,s)=1 \text{ and } M \notin \{M_i\}\ ]$

Digital Signatures

- Online verification of real life identity is difficult
- But the verification key for a digital signature can serve as your digital identity
  - OK to own multiple digital identities
  - Compromised if you lose your signing key
- Central to identity on the internet (with the help of certificate authorities), crypto currencies, etc.

"On the Internet, nobody knows you're a dog."

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]

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

$r^1{}_0$	$r^2{}_0$	$r^3{}_0$
$r^1{}_1$	$r^2{}_1$	$r^3{}_1$

Signatures from OWF

- Lamport's scheme based on OWF
  - One-time and has a fixed-length message
- One-time, fixed-length message signatures (Lamport)
  $\xrightarrow{\text{Domain-Extension}}$ arbitrary length messages (using UOWHF)
  $\xrightarrow{\text{“Certificate Tree”}}$ many-time signatures (using PRF)
- So, in principle, full-fledged digital signatures can be entirely based on OWF
- Coming up:
  - Hash-and-Sign domain extension for signatures
    - Domain extension can be done using CRHF (more efficient) or UOWHF (more secure)
  - “Certificate tree”

Domain Extension of Signatures using Hash

- 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*
  - Security: Forgery gives either a hash collision or a forgery for the original (finite domain) signature
- Formal reduction: Given adversary A for Sign*, define
  - Event_1: A outputs (M,σ) s.t. $h(M)=h(M_i)$, $M_i \neq M$, where A had asked for signature on M_i.
    Event_2: A's forgery not on such an M.
  - Advantage $\leq$ Pr[Event_1 or Event_2] $\leq$ Pr[Event_1] + Pr[Event_2]
  - CRHF adversary: given h, sample (SK,VK), let VK*=(VK,h), and run A; answer signing queries of A using (SK,h). If A outputs (M,σ) s.t. $\exists i\ h(M)=h(M_i)$, $M_i \neq M$, then output (M,M_i). Advantage = Pr[Event_1]
  - Signature adversary: given VK, pick h, let VK*=(VK,h), and run A; answer signing queries of A using signature oracle. If A outputs forgery (M,σ), output $(h(M),\sigma)$. Advantage = Pr[Event_2]

Domain Extension of Signatures using Hash

- 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
  - Security: To use a signature s_i in a forgery, need M such that $h_i(M)=h_i(M_i)$. But h_i is picked by signing algorithm after M_i is submitted. Breaks UOWHF security by finding such a collision.
    - In reduction, UOWHF adversary guesses an i where collision occurs and sends h it received as h_i (others picked unif'ly)
    - $\text{Event}_{1,i}$: A outputs $(M,(h,\sigma))$ where $(h,h(M)) = (h_i,h_i(M_i))$
    - Event_2 : A's forgery s.t. $(h,h(M)) \neq (h_i,h_i(M_i))$ for all i
    - Let q be an upper bound on number of queries by A
    - Advantage of A $\leq (\sum_{i=1}^{q} \Pr[\text{Event}_{1,i}]) + \Pr[\text{Event}_2]$
    - UOWHF adversary has advantage $= 1/q\ (\sum_{i=1}^{q} \Pr[\text{Event}_{1,i}])$
    - Signature adversary has advantage $= \Pr[\text{Event}_2]$

q=1 suffices if Sign* is to be a one-time scheme

One-Time → Many-Times

- Certificate chain: $VK_1 \to (VK_2, \sigma_2) \to \ldots \to (VK_t, \sigma_t) \to (m,\sigma)$ where σ_i is a signature on VK_i that verifies w.r.t. VK_{i-1}, and σ is a signature on m w.r.t. VK_t
  - Suppose a "trustworthy" signer only signs the verification key of another "trustworthy" signer. Then, if VK_1 is known to be issued by a trustworthy signer, and all links verified, then the message is signed by a trustworthy signer.
- Certificate tree for one-time → many-times signatures
  - Idea: Each message is signed using a unique VK for that message
    - Verifier can't hold all VKs: A binary tree of VKs, with each leaf designated for a message. Parent VK signs its pair of children VKs (one-time, fixed-length sign). Verifier remembers only root VK. Signer provides a certificate chain to the leaf VK used.
    - Signer can't remember all SKs: Uses a PRF to define the tree (i.e., SK for each node), and remembers only the PRF seed

Signatures from OWF

Summary

- One-time, fixed-length message signatures (Lamport)

  $\xrightarrow{\text{Domain-Extension}}$ arbitrary length messages (using UOWHF)

  $\xrightarrow{\text{“Certificate Tree”}}$ many-time signatures (using PRF)

- So, in principle, full-fledged digital signatures can be entirely based on OWF
- Not very efficient: Say hashes are O(k) bits long. Then, a signature contains O(k) VKs of Lamport signature, each of which, to allow signing O(k) bit messages, is $O(k^2)$ bits long
- Next time: More efficient schemes