Hash Functions in Action

Lecture 11

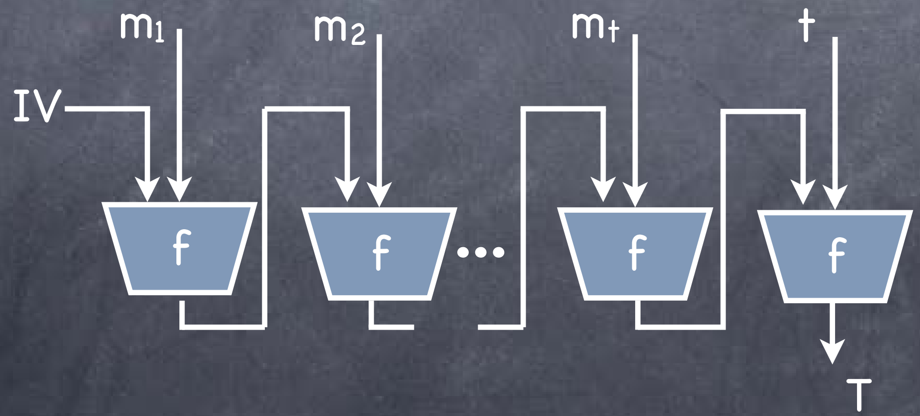
Hashes and MAC

Hash Functions

- Main syntactic feature: Variable input length to fixed length output
- Primary requirement: collision-resistance
    - If for all PPT A, Pr[x≠y and h(x)=h(y)] is negligible in the following experiment:
        - $A \rightarrow (x,y)$; $h \leftarrow \mathcal{H}$: Combinatorial Hash Functions
        - $A \rightarrow x$; $h \leftarrow \mathcal{H}$; $A(h) \rightarrow y$: Universal One-Way Hash Functions
        - $h \leftarrow \mathcal{H}$; $A(h) \rightarrow (x,y)$: Collision-Resistant Hash Functions (Typically used)
            - $h \leftarrow \mathcal{H}$; $A^h \rightarrow (x,y)$: Weak Collision-Resistant Hash Functions
- Also often required: "unpredictability"

Constructions

- 2-Universal Hash Function: e.g., $h_{a,b}(x) = chop(ax+b)$ over field $GF(2^n)$
- CRHF: e.g., $h_{\mathbb{G},g1,g2}(x_1,x_2) = g_1^{x1}g_2^{x2}$ (in $\mathbb{G}$, a prime order DL group)
- CRHF in practice: e.g., SHA 256, SHA3
- SHA 256 (and many others) using a Merkle-Damgård iterated hash function, iterating a fixed input-length compression function



Today

- CRHF: Domain Extension
  - Merkle trees and Merkle-Damgård iterated hash function
- Combinatorial Hash: A weaker notion
  - Almost XOR Universal (AXU) hash function family

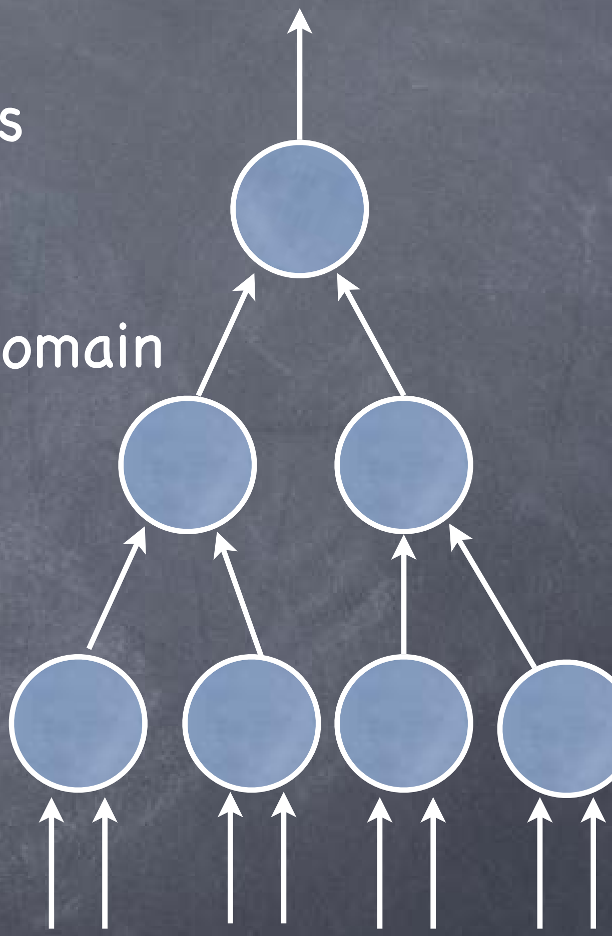
- Using hash functions for MAC
  - One-time MAC
  - Proper MACs (any number of times, variable length message)
    - With a PRF
      - GMAC (Also, recall CMAC, EMAC.)
    - Without a PRF
      - HMAC

Domain Extension

- Full-domain hash: hash arbitrarily long strings to a single hash value
    - Note that CRHF which have a fixed domain
- First, simpler goal: extend to a larger, fixed domain
    - Assume we are given a hash function from two blocks to one block (a block being, say, k bits)
        - E.g., $h_{\mathbb{G},g1,g2}(x_1,x_2) = g_1^{x1} g_2^{x2}$

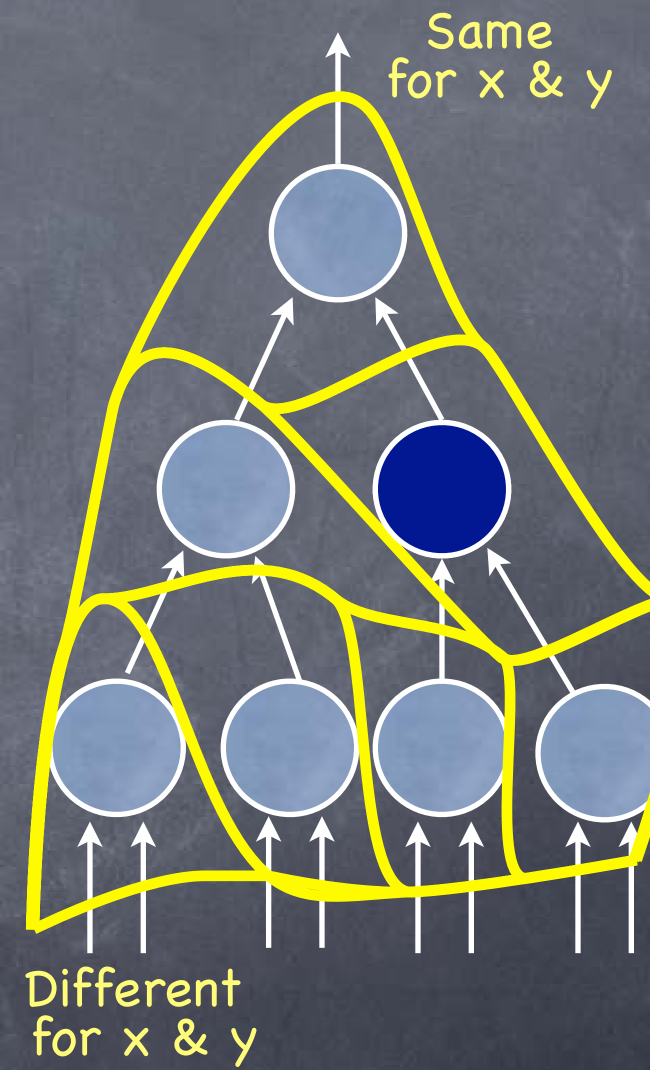
CRHF Domain Extension

- Full-domain hash: hash arbitrarily long strings to a single hash value
- First, simpler goal: extend to a larger, fixed domain
- Merkle tree
  - Uses a basic hash from $\{0,1\}^{2k}$ to $\{0,1\}^k$
  - Example: A hash function from $\{0,1\}^{8k}$ to $\{0,1\}^k$ using a tree of depth 3
  - Any tree can be used, with consistent I/O sizes
  - Same basic hash used at every node in the Merkle tree. Hash description same as for a single basic hash



Domain Extension for CRHF

- If a collision ($(x_1 \ldots x_n)$, $(y_1 \ldots y_n)$) over all, then some collision (x', y') for basic hash
  - Consider moving a "frontline" from bottom to top. Look for equality on this front.
    - Collision at some step (different values on i^{th} front, same on $i+1^{st}$); gives a collision for basic hash
- $A^*(h)$: run $A(h)$ to get $(x_1 \ldots x_n)$, $(y_1 \ldots y_n)$. Move frontline to find (x', y')

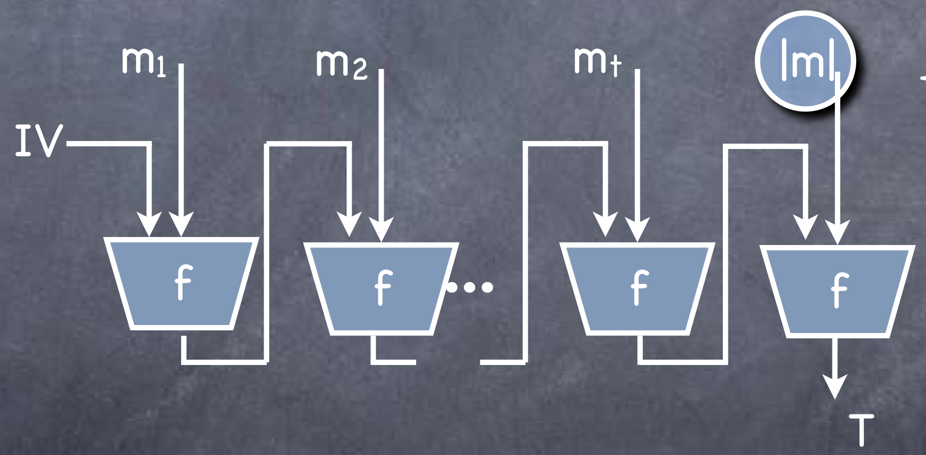


Domain Extension for CRHF

- Full-domain hash: hash arbitrarily long strings to a single hash value
  - Merkle-Tree construction extends the domain to any fixed input length
- Hash the message length (number of blocks) along with the original hash
  - Collision in the new hash function gives either collision at the top level, or if not, collision in the original Merkle tree and <u>for the same message length</u>

CRHF in Practice

- A single function, not a family (e.g. SHA-3, SHA-256, MD4, MD5)
- Often based on a fixed input-length compression function
- Merkle-Damgård iterated hash function, MD^f:



Collision resistance even with variable input-length.

Note: Unlike CBC-MAC, here "length-extension" is OK, as long as it results in a different hash value

If f is not keyed, but "concretely" collision resistant, so is MD^f

- If f is "concretely" collision resistant then so is MD^f (for any IV)

XOR-Universal Hash

- Recall Combinatorial HF: $A \rightarrow (x,y)$; $h \leftarrow \mathcal{H}$. $h(x)=h(y)$ w.n.p
- 2-Universal hash function family
  - $\forall x \neq y, w, z \ \Pr_{h \leftarrow \mathcal{H}} [\ h(x)=w,\ h(y)=z\ ] = 1/|\text{range}|^2$
- **XOR-Universal hash function** family (range = $\{0,1\}^k$, say)
  - $\forall x \neq y, z \ \Pr_{h \leftarrow \mathcal{H}} [\ h(x) \oplus h(y) = z\ ] = 1/|\text{range}|$
- **ε-Almost XOR-Universal hash function** family
  - $\forall x \neq y, z \ \Pr_{h \leftarrow \mathcal{H}} [\ h(x) \oplus h(y) = z\ ] \leq \varepsilon$

A 2UHF is an XUHF

Converse not true [Exercise]

- AXUHF example: Variable length input, $m = (m_1, \ldots, m_t)$, t k-bit blocks
  - $h_\alpha(m) = m_1 \alpha + m_2 \alpha^2 + \ldots + m_t \alpha^t + |m| \alpha^{t+1}$
    - m defines a polynomial P_m and $h_\alpha(m) = P_m(\alpha)$
  - $\Pr_{h \leftarrow \mathcal{H}} [\ h(m) \oplus h(m') = z\ ] = \Pr_{\alpha \leftarrow GF(2^k)}[\Delta(\alpha) = z] \leq \text{degree}(\Delta)/2^k$ where Δ is a non-zero polynomial of degree $\leq \max\{|m|,|m'|\}+1$

Over $GF(2^k)$. Addition is XOR

Hashes for MAC

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

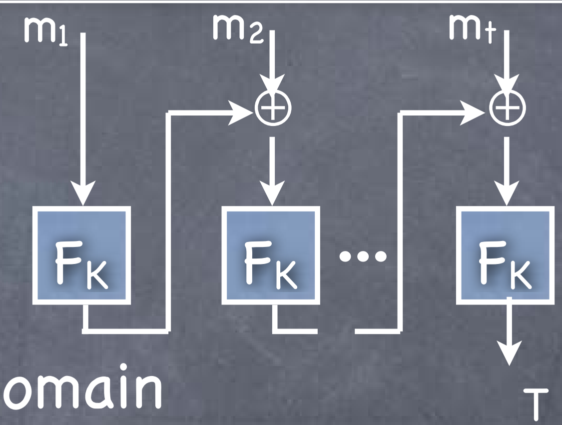
  - Key: 2n random strings (each k-bit long) $(r^i_0, r^i_1)_{i=1..n}$
  - Signature for $m_1 \ldots m_n$ be $(r^i_{m_i})_{i=1..n}$
  - Negligible probability that Eve can produce a signature on $m' \neq m$
- A much more efficient solution, using 2-UHF (and still no computational assumptions):
  - Onetime-MAC$_h$(M) = 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: Beyond One-Time

With Combinatorial Hash Functions and PRF

RECALL

- MACs can be based entirely on PRFs
  - PRF is a MAC (on one-block messages)
  - CBC-MAC: Extends PRF to any fixed length domain
  - Can also make it work 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′ is an independent key (EMAC)
    - Or, XOR last message block with another key K′ (CMAC)



- Using hash & PRF (for fixed length domains):
  - $MAC_{K,h}^*(M) = PRF_K(h(M))$ where $h \leftarrow \mathcal{H}$, and $\mathcal{H}$ is a 2-UHF

h(M) not revealed

MAC: Beyond One-Time

With Combinatorial Hash Functions and PRF

- Using an ε-AXUHF & PRF (for variable length domains)
  - $MAC_{K,h}^*(M) = (r, PRF_K(r) \oplus h(M))$ where $h \leftarrow \mathcal{H}$, $\mathcal{H}$ ε-AXUHF, r random
    - Forgery with a fresh r prevented by PRF.
    - Forgery reusing an r requires knowing $h(M) \oplus h(M')$, given no information about h (due to encryption with PRF)
- GMAC, a NIST standard: With polynomial evaluation over $GF(2^k)$, i.e., $h_{K'}(m) = P_m(K')$, being the ε-AXUHF
- Note that GMAC is randomised as it needs a nonce r
  - But not a problem when used as part of Authenticated Encryption, which already needs a nonce
- Galois Counter Mode (GCM): Authenticated encryption using encrypt (AES in CTR mode) then MAC (GMAC).
  - Nonce r (with counter 0) used for GMAC, and $PRF_K(r+i)$ with $i > 0$, for encryption. (Nonce itself is not MAC'ed.)

MAC: Beyond One-Time

With Cryptographic Hash Functions

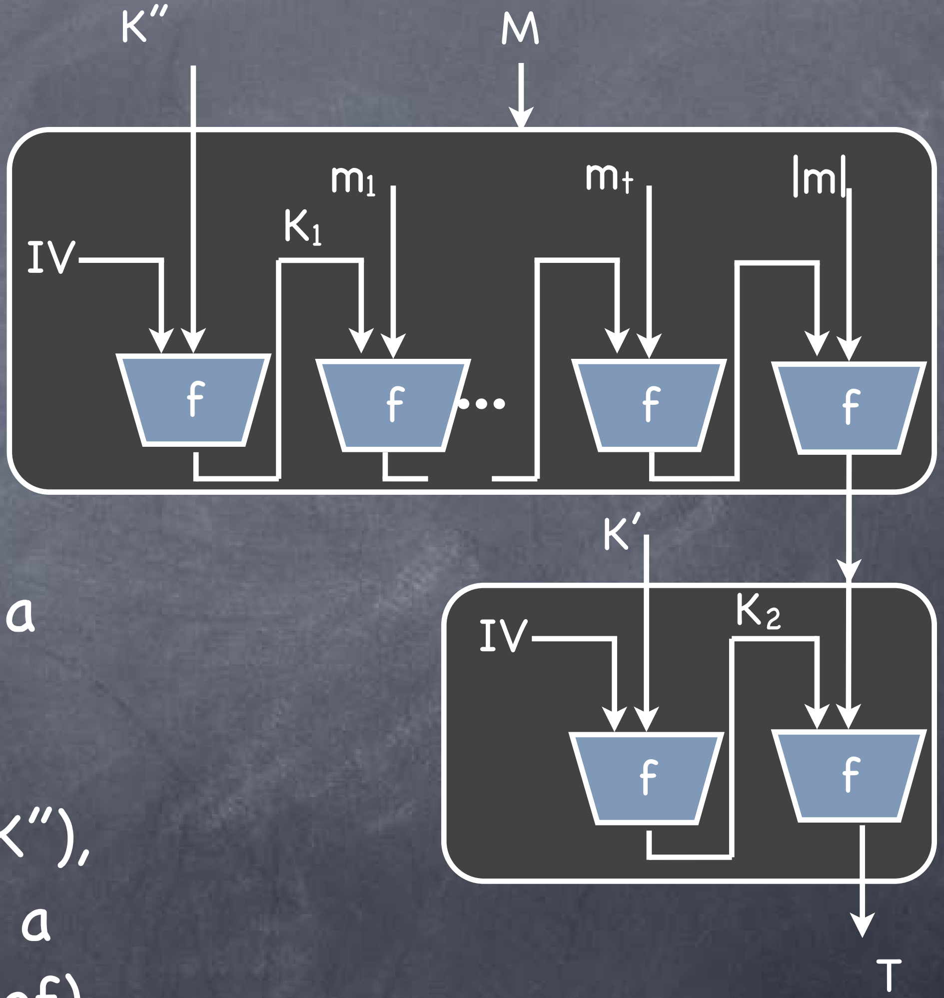
- Previous solutions required pseudorandomness
- What if we should base it only on fixed input-length MAC (not PRF)?
  - Why? "To avoid export restrictions!" (Was a consideration in the 1990's). Also security/efficiency
  - Candidate fixed input-length MACs in practice that do not use a block-cipher: compression functions (with key as IV)
- $MAC^*_{K,h}(M) = MAC_K(h(M))$ where $h \leftarrow \mathcal{H}$, and $\mathcal{H}$ a weak-CRHF
  - Weak-CRHFs can be based on OWF (unlike CRHF). Efficient heuristic construction from compression functions (again)

h(M) may be revealed. Only oracle access to h

MAC: Beyond One-Time

With Cryptographic Hash Functions

- **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!

- 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
    - Take M' = M || pad_M || X, where pad_M is a block encoding |M| (used by the Merkle-Damgård iterated-hash) and X is arbitrary. Then, can compute H(K||M') from H(K||M).
        - (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

Today

- CRHF domain extension using Merkle trees
- Merkle-Damgård iterated hash function for full-domain hash
- ε-AXUHF as a full-domain combinatorial hash function
- Hash functions for MACs
  - Using a PRF: encipher 2UHF, or encrypt AXUHF
    - Using AXUHF GHASH: GMAC and GCM
  - Hash-then-MAC
    - Using weak CRHF and fixed input-length MAC
    - Underlying HMAC/NMAC: compression function assumed to (1) be a fixed input-length MAC, and (2) when used in a keyed iterated-hash function, yield a weak CRHF.
- Next: Digital Signatures