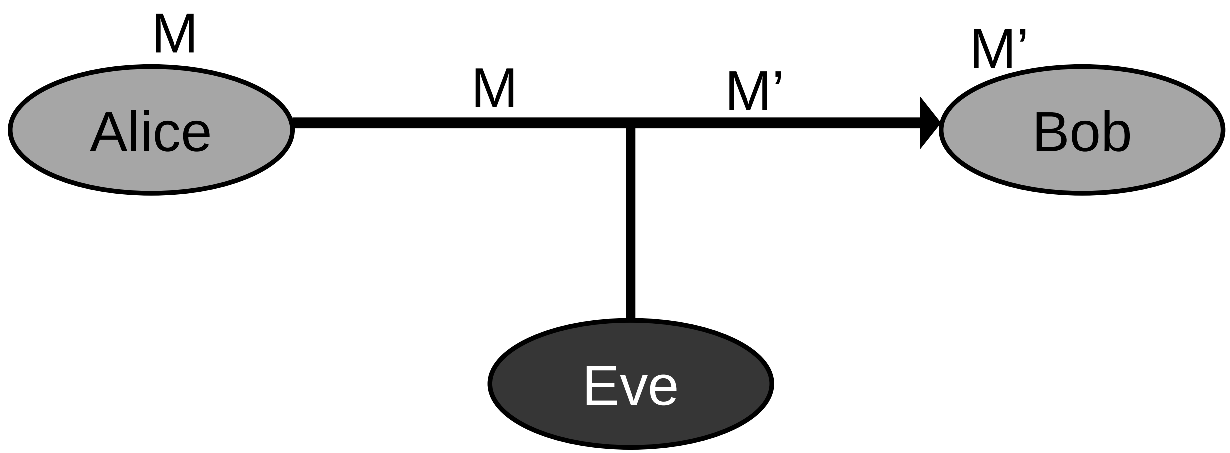
Introduction to Cryptography
Lecture 4

Message authentication
Hash functions

Benny Pinkas

Data Integrity, Message Authentication

- Risk: an *active* adversary might change messages exchanged between Alice and Bob



• Authentication is orthogonal to secrecy. A relevant challenge regardless of whether encryption is applied.

One Time Pad

- OTP is a perfect cipher, yet provides no authentication
  - Plaintext $x_1x_2\dots x_n$
  - Key $k_{1k2}\dots k_n$
  - Ciphertext $c_1=x_1\oplus k_{1,}$ $c_2=x_2\oplus k_2,\dots,c_n=x_n\oplus k_n$

- Adversary changes, e.g., c_2 to $1\oplus c_2$
- User decrypts $1\oplus x_2$

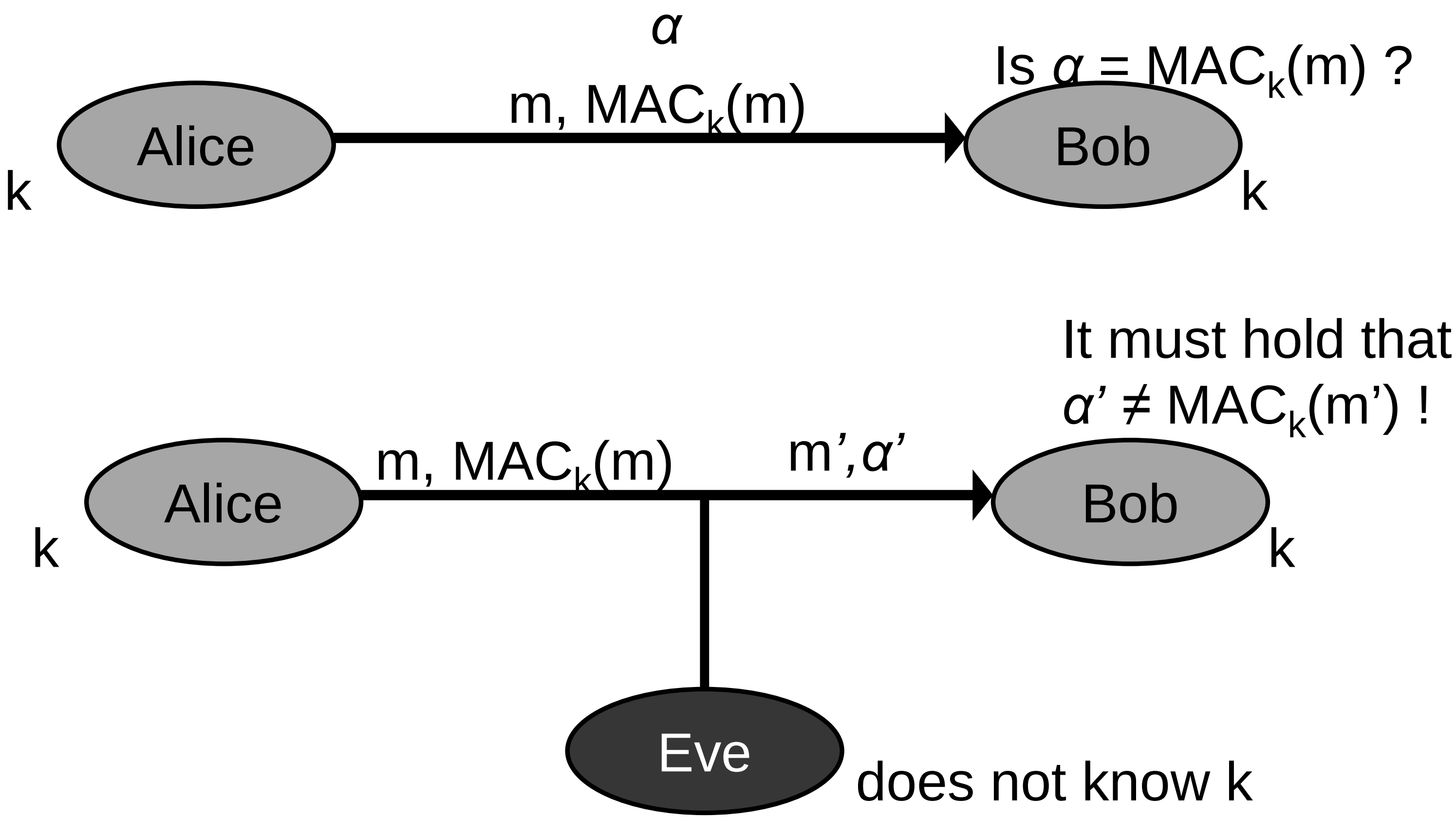
- Error-detection codes are insufficient. (For example, linear codes can be changed by the adversary, even if encrypted.)

Definitions

- Scenario: Alice and Bob share a secret key *K*.
- Authentication algorithm:
  - Compute a Message Authentication Code: $\alpha = MAC_K(m)$.
  - Send *m* and α
- Verification algorithm: $V_K(m, \alpha)$.
  - $V_K(m, MAC_K(m)) = accept.$
  - For $\alpha \neq MAC_K(m)$, $V_K(m, \alpha) = reject.$

- How does $V_k(m)$ work?
  - Receiver knows k. Receives *m* and α.
  - Receiver uses *k* to compute $MAC_K(m)$.
  - $V_K(m, \alpha) = 1$ iff $MAC_K(m) = \alpha$.

Common Usage of MACs for message authentication



Requirements

- Security: The adversary,
  - Knows the MAC algorithm (but not K).
  - Is given many pairs $(m_i, MAC_K(m_i))$, where the m_i values might also be chosen by the adversary (chosen plaintext).
  - Cannot compute $(m, MAC_K(m))$ for any new m $(\forall i\ m \neq m_i)$.
  - The adversary must not be able to compute $MAC_K(m)$ *even* for a message *m* which is "meaningless" (since we don't know the context of the attack).
- Efficiency: output must be of fixed length, and as short as possible.
  - $\Rightarrow$ The MAC function is not 1-to-1.
  - $\Rightarrow$ An n bit MAC can be broken with prob. of at least 2^{-n}.

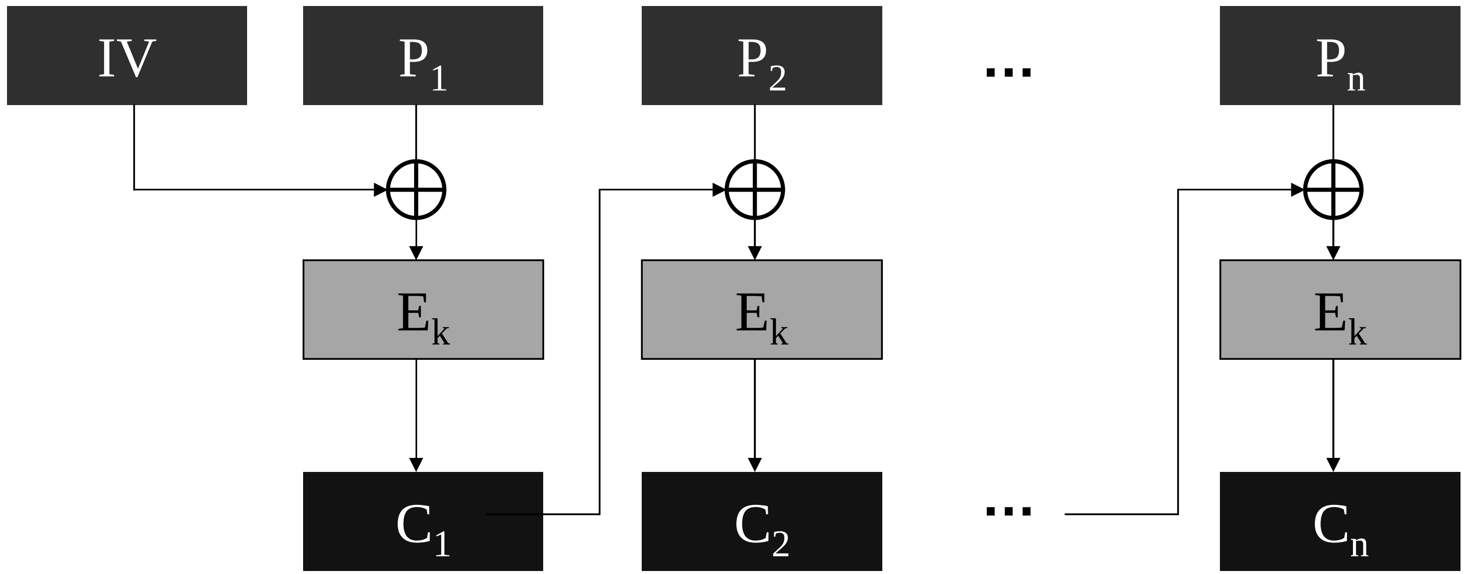
Constructing MACs

- Based on block ciphers (CBC-MAC)

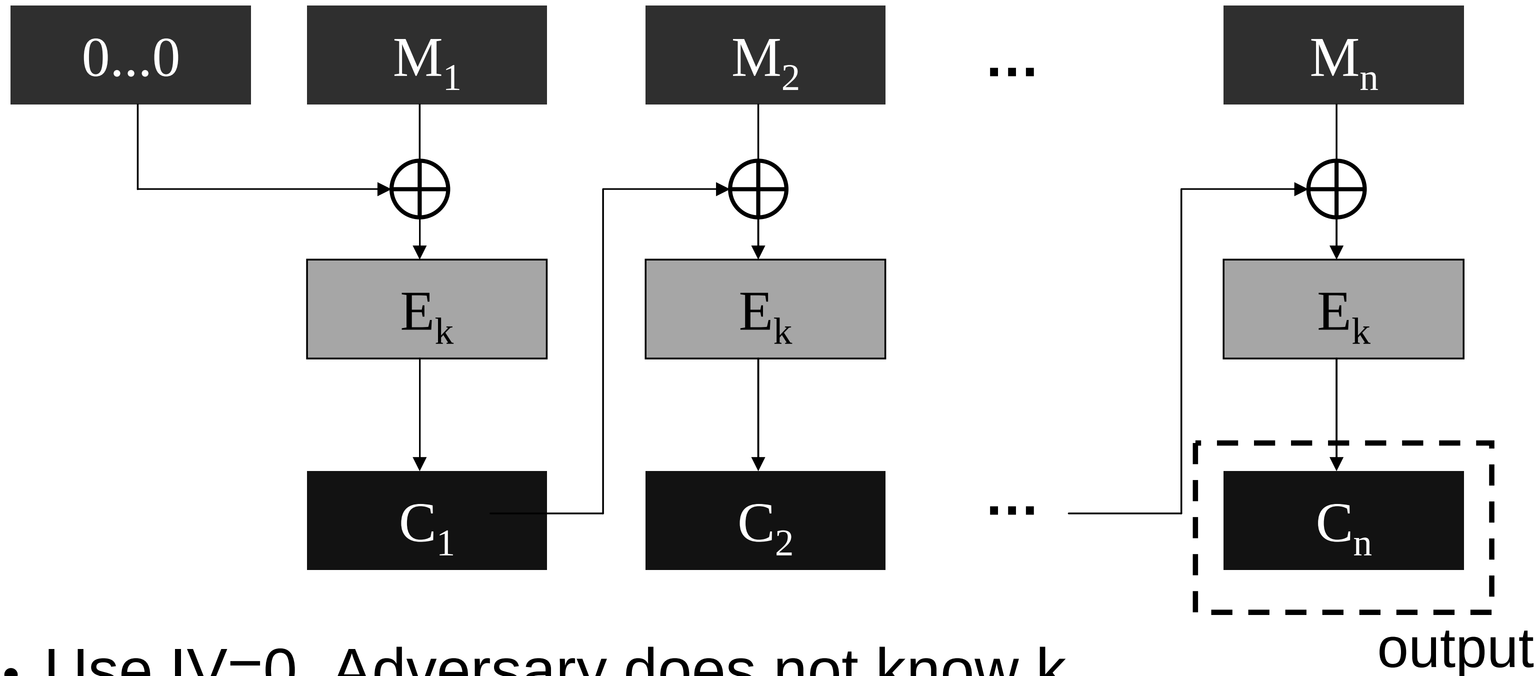
  or,
- Based on hash functions
  - More efficient
  - At the time, encryption technology was controlled (export restricted) and it was preferable to use other means when possible.

CBC

- Reminder: CBC encryption
- Plaintext block is xored with previous ciphertext block



CBC MAC



- Use IV=0. Adversary does not know k.
- Encrypt M in CBC mode, using the MAC key. Discard $C_1,\ldots,C_{n-1}$ and define $MAC_K(M_1,\ldots,M_n)=C_n$.

Security of CBC-MAC

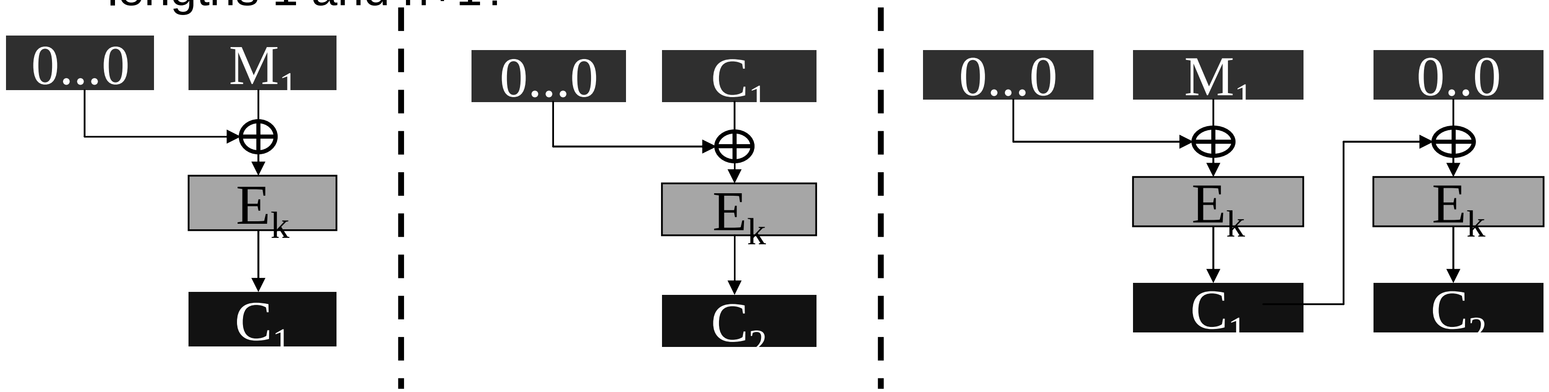
- Claim: if E_K is pseudo-random then CBC-MAC, applied to *fixed length messages*, is a pseudo-random function, and is therefore resilient to forgery.

- But, insecure if variable lengths messages are allowed

Security of CBC-MAC

- Insecurity of CBC-MAC when applied to messages of variable length:
  – Get C_1 = CBC-MAC$_K(M_1) = E_K(0 \oplus M_1)$
  – Ask for MAC of C_1, i.e., C_2 = CBC-MAC$_K(C_1) = E_K(0 \oplus C_1)$
  – But, $E_K(C_1 \oplus 0) = E_K(\ E_K(0 \oplus M_1) \oplus 0)$ = CBC-MAC$_K(M_1 \mid 0)$

  - It's known that CBC-MAC is secure if message space is prefix-free.
  - Can you show, for every n, a collision between two messages of lengths 1 and n+1?



CBC-MAC for variable length messages

- Solution 1: The first block of the message is set to be its length. I.e., to authenticate $M_1,\ldots,M_n$, apply CBC-MAC to $(n,M_1,\ldots,M_n)$.
  - Works since now message space is prefix-free.
  - Drawback: The message length (n) must be known in advance.
- "Solution 2": apply CBC-MAC to $(M_1,\ldots,M_n,n)$
  - Message length does not have to be known is advance
  - But, this scheme is broken (see, M. Bellare, J. Kilian, P. Rogaway, The Security of Cipher Block Chaining, 1984)
- Solution 3: (preferable)
  - Use a second key K'.
  - Compute $MAC_{K,K'}(M_1,\ldots,M_n) = E_{K'}(MAC_K(M_1,\ldots,M_n))$
  - Essentially the same overhead as CBC-MAC

Hash functions

- A hash function $h:X \rightarrow Y$ maps long inputs to fixed size outputs. $(|X|>|Y|)$

- No secret key. The hash function algorithm is public.

- If $|X|>|Y|$ there are collisions ($x \neq x'$ for which $h(x)=h(x')$).

Security definitions for hash functions

1. Preimage resistance: for any y, it is hard to find x such that h(x)=y.
2. Weak collision resistance: for any $x \in X$, it is hard to find $x' \neq x$ such that $h(x)=h(x')$. (Also known as "universal one-way hash", or "*second* preimage resistance").
3. Strong collision resistance: it is hard to find any x,x' for which $h(x)=h(x')$.

- It's easier to find collisions. (Under reasonable assumptions (3) $\rightarrow$ (1), and (3) $\rightarrow$ (2).) Therefore strong collision resistance is a stronger assumption.
- Real world hash functions: MD5, SHA-1, SHA-256.

Hmm..

The Birthday Phenomenon (Paradox)

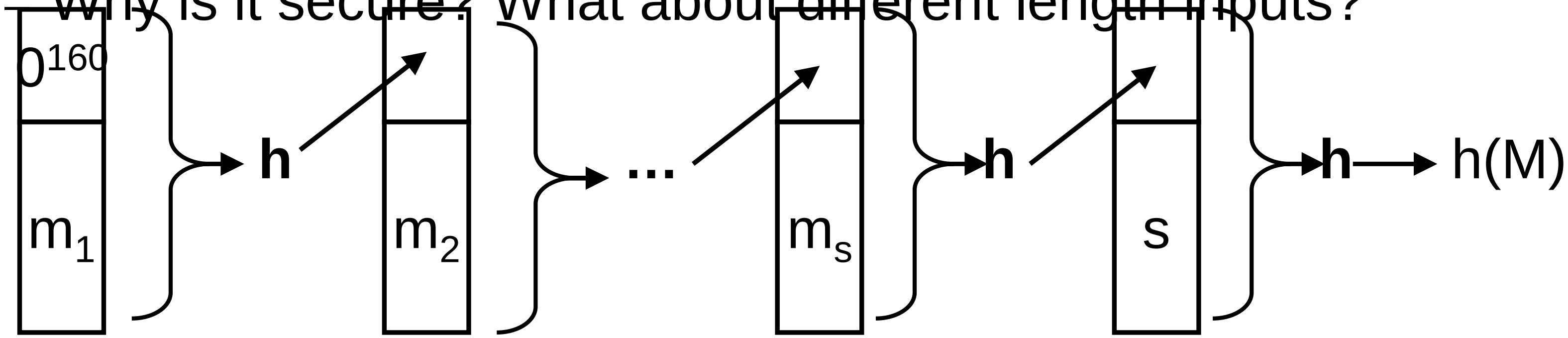
- For 23 people chosen at random, the probability that two of them have the same birthday is ½.
- Compare to: the prob. that one or more of them has the same birthday as Alan Turing is 23/365 (actually, $1-(1-1/365)^{23}$.)

- More generally, for a random $h:X \to Z$, if we choose about $|Z|^{½}$ elements of Z at random ($1.17\ |Z|^{½}$), the probability that two of them are mapped to the same image is > ½.

- Implication: it's harder to achieve strong collision resistance
  - A random function with a n bit output
    - Find x,x' with h(x)=h(x') after about $2^{n/2}$ tries.
    - Find x≠0 s.t. h(x)=h(0) after about 2^n attempts.

From collision-resistance for fixed length inputs, to collision-resistance for arbitrary input lengths

- Hash function:
  - Input block length is usually 512 bits (|X|=512)
  - Output length is at least 160 bits (birthday attacks)
- Extending the domain to arbitrary inputs
  - Suppose $h:\{0,1\}^{512} \to \{0,1\}^{160}$
  - Input: $M=m_1 \ldots m_s$, $|m_i|=512-160=352$. (what if $|M| \neq 352 \cdot i$ bits?)
  - Define: $y_0=0^{160}$. $y_i=h(y_{i-1},m_i)$. $y_{s+1}=h(y_s,s)$. $h(M)=y_{s+1}$.
  - Why is it secure? What about different length inputs?



Proof

- Show that if we can find $M \neq M'$ for which $H(M)=H(M')$, we can find blocks $m \neq m'$ for which $h(m)=h(m')$.

- Case 1: suppose $|M|=s$, $|M'|=s'$, and $s \neq s'$
  – Then, collision: $H(M)=h(y_s,s) = h(y_{s'},s')=H(M')$
- Case 2: $|M|=|M'|=s$
  – We know that $H(M)=h(y_s,s)=h(y'_s,s)=H(M')$
  – If $y_s \neq y'_s$ then we found a collision in h.
  – Otherwise, go from $i=s-1$ to $i=1$:
    - $y_{i+1} = y'_{i+1}$ implies $h(y_i,m_{i+1}) = h(y'_I,m'_{i+1})$.
    - If $y_i \neq y'_i$ or $m_{i+1} \neq m'_{i+1}$, then we found a collision.
    - $M \neq M'$ and therefore there is an i for which $m_{i+1} \neq m'_{i+1}$

Basing MACs on Hash Functions

- Hash functions are not keyed. MAC_K uses a key.
- Best attack should not succeed with prob > $\max(2^{-|k|}, 2^{-|MAC()|})$.
- Idea: MAC combines message and a secret key, and hashes them with a collision resistant hash function.
  - E.g. $MAC_K(m) = h(k,m)$. (insecure.., given $MAC_K(m)$ can compute $MAC_K(m,|m|,m')$, if using the MD construction)
  - $MAC_K(m) = h(m,k)$. (insecure.., regardless of key length, use a birthday attack to find m,m' such that h(m)=h(m').)

- How should security be proved?:
  - Show that if MAC is insecure than so is hash function h.
  - Insecurity of MAC: adversary can generate $MAC_K(m)$ without knowing k.
  - Insecurity of h: adversary finds collisions ($x \neq x'$, $h(x)=h(x')$.)

HMAC

- Input: message *m,* a key *K*, and a hash function *h*.
- $HMAC_K(m) = h(K \oplus opad, h(K \oplus ipad, m))$
  - where ipad, opad are 64 byte long fixed strings
  - K is 64 byte long (if shorter, append 0s to get 64 bytes).
- Overhead: the same as that of applying h to m, plus an additional invocation to a short string.

- It was proven [BCK] that if HMAC is broken then either
  - h is not collision resistant (even when the initial block is random and secret), or
  - The output of h is not “unpredcitable” (when the initial block is random and secret)
- HMAC is used everywhere (SSL, IPSec).

What we learned today

- Message authentication
  - CBC MAC
  - Hash functions
  - The birthday paradox
  - HMAC