Messages Authenticity and Digital Signatures

Dr Basel Halak

Learning Outcomes

At the end of this unit you should be able to:

1. Explain the meaning of message integrity.
2. Describe the security requirement of message authentication codes(MACs).
3. Construct a MAC based on symmetric ciphers.
4. Explain the principles of Hash Functions.
5. Outline the principles of Digital Signatures

Learning Outcomes

At the end of this unit you should be able to:

1. Explain the meaning of message integrity.
2. Describe the security requirement of message authentication codes(MACs).
3. Construct a MAC based on symmetric ciphers.
4. Explain the principles of Hash Functions.
5. Outline the principles of Digital Signatures

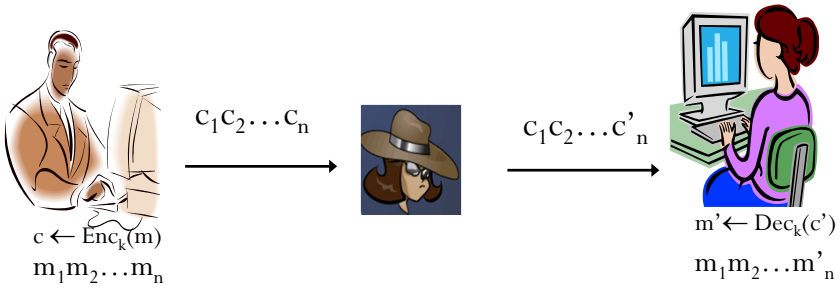
Message Integrity

- **Message integrity** is achieved by ensuring that a received message has actually originated from the intended party, and was not modified even if an attacker controls the channel
- Standard error-correction techniques not enough!, Why?
- It aims to achieve integrity not confidentiality.

Message Integrity

- Application Examples:
  - Protecting operating systems from viruses
  - Ensuring the integrity of banks transections

Privacy does not imply authenticity



- Secrecy and integrity are *orthogonal* concerns
  - Possible to have either one without the other
- Encryption does not (in general) provide integrity

Learning Outcomes

At the end of this unit you should be able to:

1. Explain the meaning of message integrity.
2. Describe the security requirement of message authentication codes(MACs).
3. Construct a message security code.
4. Explain the principles of Hash Functions.
5. Outline the principles of digital signatures

Message Authentication Code (MAC)

- A *message authentication code* is defined by two algorithms:

1. **Signing Algorithm** (Tag Generation):takes as input a message m and a key k and outputs a tag t
2. **Verification Algorithm** : takes key k, message m, and tag t as input; outputs 1 ("accept") or 0 ("reject")

Message Authentication Code (MAC)

{m, tag}

Signing Algorithm
tag ← S(k, m)

Verification Algorithm
V(k, m, tag) = {1, 0}

Definition: **MAC** I = (S,V) defined over (K,M,T) is a pair of algorithms:

- S(k, m) outputs t in T
- V(k, m, t) outputs `1' or `0'

Consistency Condition: For all m's and corresponding k's,
V(m, k S(k, m)) = '1'

Why Does Integrity requires a secret key

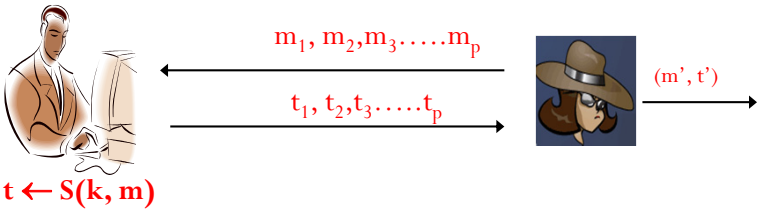
{m, tag}

Signing Algorithm
tag ← S(m)

Verification Algorithm
V(m, tag) = {1, 0}

MAC Security Definition

- **Adversary 's power:** Chosen Message Attack



- **A MAC= (S,V) is said to be secure if for all "efficient" A**:

$$Adv[A,MAC] = Pr[V(K, m', t')=1)] \quad \text{is "negligible."}$$

MAC Security Definition

- **What does *Forgery* means?**

If an attacker A is able to produce a pair (m', t') , such tat m' did not originate from the sender, and V(m', k, t')=1. Such a pair is called "*forgery*" and the attacker is said to have forged.

- **Attack Model:** Chosen-Message Attack

To construct a secure MAC, we assume worst case scenario, in this case: the attacker is assumed to be able to induce the sender to authenticate *messages of the attacker's choice*

MAC Security Definition

- **Security Goal:** Existential unforgeability
- **A message authentication code is said to be secure** if and only if MAC is able to detect any attempt by the adversary to modify the transmitted data.
- *Attacker should not be able to produce a new valid (message, tag) pair or even produce a new tag for an old message.*

Exercise

Consider MAC= (S,V), such that t= S(k,m) is always 6 bits long

Is this a secure MAC against Chosen Message Attack?

Replay attacks

- Replay attacks is when an attacker re-send old messages which have valid tags
- Note that *replay attacks* are not prevented by the use of MACs
  - No stateless mechanism can prevent them
- Need to protect against replay attacks at a higher level

Learning Outcomes

At the end of this unit you should be able to:

1. Explain the meaning of message integrity.
2. Describe the security requirement of message authentication codes(MACs).
3. Construct a MAC based on symmetric ciphers.
4. Explain the principles of Hash Functions.
5. Outline the principles of Digital Signatures

PRPs and PRFs

- Pseudo Random Function (**PRF**) defined over (K,X,Y):

  $F: K \times X \rightarrow Y$

  such that exists "efficient" algorithm to evaluate F(k,x)

- **Pseudo Random Permutation (PRP) defined over (K,X):**

  $E: K \times X \rightarrow X$

  such that:

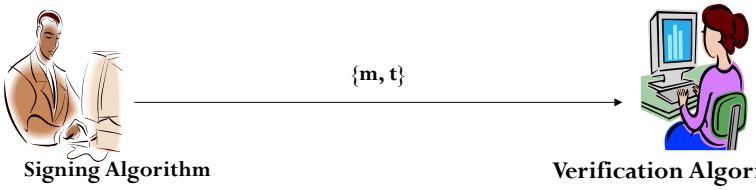
  1. Exists "efficient" deterministic algorithm to evaluate E(k, x)
  2. The function E(k, ·) is one-to-one
  3. Exists "efficient" inversion algorithm D(k, y)

Constructing a MAC from a secure PRF

- **A MAC can be constructed from a secure PRF as follows:**

For a PRF **F: K × X ⟶ Y** define a MAC = (S,V) as:

- S(k,m) := F(k,m)
- V(k,m,t): output '1' if t = F(k,m) and '0' otherwise.



Signing Algorithm
t ← F(k, m)

Verification Algorithm
V(k, m, t) = 1 only if t=F(k,m)

- **Example**: you can use AES-128 algorithm to construct a MAC for 16-byte messages.

Exercise

Suppose $F: K \times X \rightarrow Y$ is a secure PRF with $Y = \{0,1\}^8$

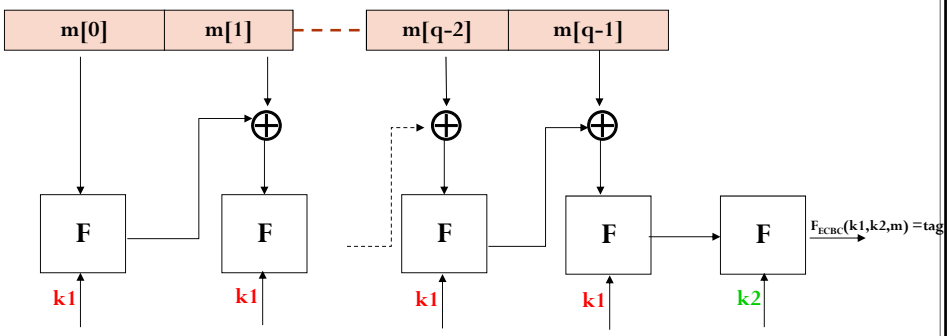
Is the derived MAC a secure MAC system?

How to generate a tag for long messages

- **Goal**: given a PRF for short messages (AES) construct a PRF for long messages
- Two main constructions used in practice:
  - **ECBC-MAC** (banking – ANSI X9.9, X9.19, FIPS 186-3)
  - **HMAC** (Internet protocols: SSL, IPsec, SSH, …)
- Both convert a small-MAC into a big-MAC.

ECBC-MAC

- **Definition**: Let $F: K \times X \longrightarrow X$ be a PRP, m is a long message which is divided into q segment m[0].....m[q-1], we define a new MAC as follows:
- $S(k1,k2,m) = F_{ECBC}(k1,k2,m)$
- V(k1,k2,m,t): output '1' if $t = F_{ECBC}(k1,k2,m)$ and '0' otherwise.

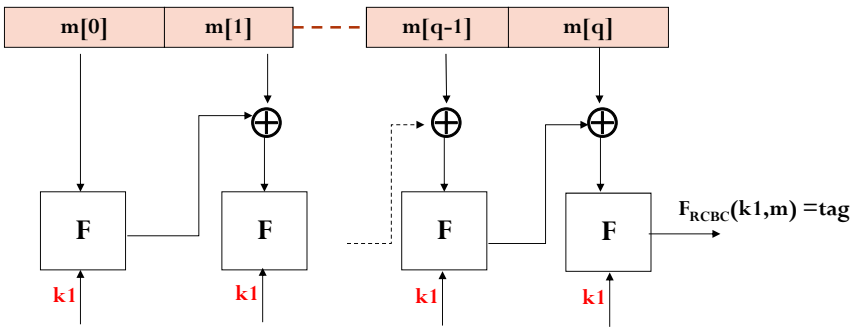


Why the last encryption step in ECBC-MAC?

Suppose we define the MAC with only one key as follows

$S(k1,m) = F_{RCBC}(k1,m)$

We call this scheme raw CBC



Simple Attack on Raw CBC

1. Adversary chooses an arbitrary one-block message $m \in X$
2. He requests tag for m. Get $t = F_{RCBC}(k,m)$ (Chosen Message Attack)
3. He simply outputs t as MAC forgery for the 2-block message $(m, t \oplus m)$

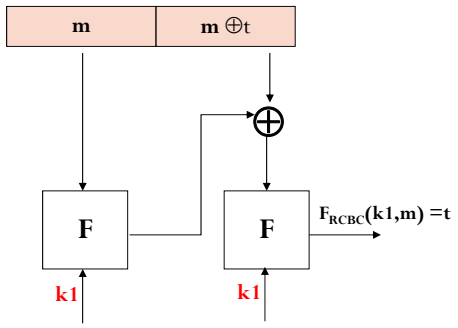
Proof:

$F_{RCBC}(k, (m, t \oplus m)) =$

$F(k, F(k,m) \oplus (t \oplus m)) =$

$F(k, t \oplus (t \oplus m)) =$

$F(K, m) = t$



Hashed Message Authentication Code (HMAC)

- This is another approach to constructing a secure MAC for variable length messages
- Most widely used MAC on the Internet.
- but, we first we need to discuss hash function.

Learning Outcomes

At the end of this unit you should be able to:

1. Explain the meaning of message integrity.
2. Describe the security requirement of message authentication codes(MACs).
3. Construct a MAC based on symmetric ciphers.
4. Explain the principles of Hash Functions.
5. Outline the principles of Digital Signatures

Hash Functions

- **A hash function**: maps arbitrary length inputs to a short, fixed-length *digest:*

  $$\mathbf{H: \{0,1\}^N \rightarrow \{0,1\}^n}$$

  where N is much larger then n

- Properties required of a hash function depend on its applications.

- **A classical application**: To create a one-way password file **such that** users passwords are stored in as (username, password), but as (username, h(password))

Security Requirements

- **Definitions**
- **Pre-image**: if y=H(x) , x is a pre-image of y. Each hash value typically has multiple pre-images
- **A *collision*** is a pair of distinct inputs x, x' such that H(x) = H(x')

Security Requirements

- **Hash functions used for secure applications must be:**

1. *Pre-image resistant*: if it is computationally infeasible to find a pre-image of a hash value (given a value y it is infeasible to find an x such that h(x) = y)
2. *Collision resistant*: if it is computationally infeasible to find a collision
   - Given x and h(x), infeasible to find y ≠ x such that h(y) = h(x)
   - Infeasible to find any x and y, with x ≠ y such that h(x) = h(y)

Hash Functions Applications Examples

1. **To build message authentication codes**
2. **To create a one-way password file** in order to store hash of password not actual password
3. **For intrusion detection and virus detection by creating** hashes of files on system and monitoring these for any changes

Hash Function Example

- **Example**

Let H: $\{0,1\}^N \rightarrow \{0,1\}^8$

Consider $X \in \{0,1\}^N$, such $X= (x_0,x_1,x_2,\ldots,x_{N-1})$, each x_i is a byte

We define : $H(X)= x_0+x_1+x_2+\ldots+x_{n-1}$

Is this secure?

Hash Function Example

Example

Let H: $\{0,1\}^N \rightarrow \{0,1\}^8$

Consider $X \in \{0,1\}^N$, such $X= (x_0,x_1,x_2,\ldots,x_{N-1})$, each x_i is a byte

We define : $H(X)= x_0+x_1+x_2+\ldots+x_{n-1}$

Is this secure?

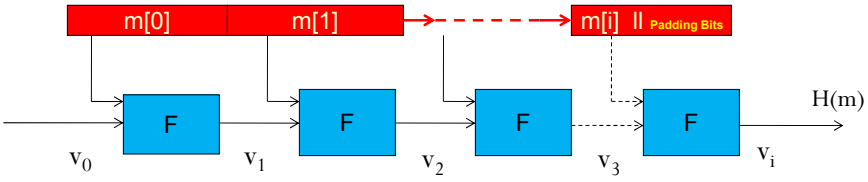
No because it is very easy to find collision

For example: assuming N=2

X1 = (00000000,11111111), X2 (11111111, 00000000)

H(X1)=H(X2)= 11111111

Collision Resistant Hash Function (Merkle-Damgard Scheme)

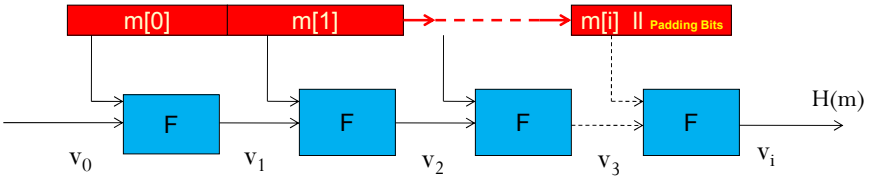


Merkle-Damgard Scheme

Operation Principles

- The message is broken into blocks of size K
- Padding bits are appended to the last block if its sise is smaller then K
- If the message happens to be a multiple of K , then an extra padding block will be added.
- Padding bits contain a series of 1000 which indicate the end of the message and also it include the length of the message
- The initial vector V_0 is fixed
- F is a collision resistant compression function

Collision Resistant Hash Function



Merkle-Damgard Scheme

Theorem: If F is collision-resistant, then is H collision-resistant

Collision Resistant Compression Function

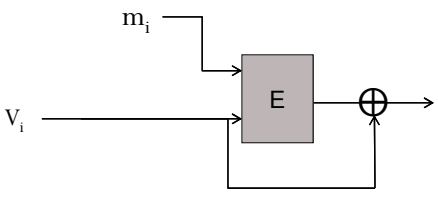
The Davies-Meyer compression function:

Consider a block cipher. $E (K \times \{0,1\}^n) \longrightarrow \{0,1\}^n$

The Davies-Meyer compression function (***F***)

is constructed as

$F(V, m) = E(m, H) \oplus H$

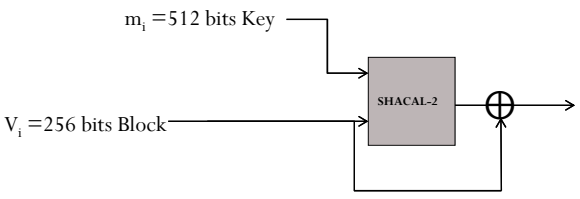


Secure Hash Algorithm (SHA)

- SHA originally designed by NIST & NSA in 1993
- It was revised in 1995 as SHA-1
- In 2005 results on security of SHA-1 have raised concerns on its use in future applications
- NIST issued revision in 2002 adds 3 additional versions of SHA : SHA-256, SHA-384, SHA-512
- Their structure & detail is similar to SHA-1, but security levels are rather higher

Example: SHA-256

- It is **based on Merkle-Damgard Scehme**
- It uses **Davies-Meyer compression function** with a block cipher called : SHACAL-2



SHA-3

Public competition by NIST, similar to AES:

- NIST's request for proposals (2007)
- 51 submissions (2008)
- 14 semi-finalists (2009)
- 5 finalists (2010)
- Winner: Keccak (2012)
  - Designed by Bertoni, Daemen, Peeters, Van Assche.
  - Based on "sponge construction", a completely different structure.

Keyed Hash Functions as MACs

- **Hash Function are used to construct a message authentication code** for variable length messages: because secure hash functions are generally faster and widely available.

- **To achieved authenticity** such MACs should include a key along with message so original proposal was as follows:

$$S(k, m) = H(k \,||\, m))$$

- Original construction suffered from weaknesses , which eventually led to the creation of HMAC

HMAC Design Objectives (as defined by RFC 2104)

1. To use, **without modifications, hash functions**
2. To allow for **easy replace-ability** of embedded hash function
3. To preserve **original performance of hash function** without significant degradation
4. To use and **handle keys in a simple way.**
5. To have **well understood cryptographic analysis** of authentication mechanism strength

Hash Message Authentication Codes (HMAC)

- It is specified as Internet standard RFC2104
- It uses hash function on the message:

HMAC: $S(k, m) = H(k \oplus opad \;||\; H(k \oplus ipad \;||\; m))$

where:
- opad, ipad are specified padding constants

- Any hash function can be used
  - eg. MD5, SHA-1, RIPEMD-160, Whirlpool

Security of MACs from Hash Functions

- **Why Collision resistance is necessary for security**:

Example: consider an MAC: **S(k,m) = H(m, k) =t**

Assuming H is not collision resistant, then: **S** is insecure under the following a 1-chosen message attack:

- an adversary finds: $m_0 \neq m_1$ s.t. $H(k,m_0) = H(k,m_1)$
- he asks for $t \leftarrow S(k, m_0)$
- he outputs (m_1, t) as forgery

Attacks on Hash Functions

- **Brute Force Attack**

Consider a hash function: H: $\{0,1\}^* \rightarrow \{0,1\}^n$?

Compute $H(x_1)$, …, $H(x_{2^n+1})$

- **This attack guarantees finding a collision in time $O(2^n)$ hashes**

The Birthday Problem

- **Quiz**:

What is the probability that at least two of k randomly selected people have the same birthday? (Same month and day, but not necessarily the same year.) . It is assumed nobody was born on February 29 and people's birthdays are equally distributed over the other 365 days of the year

The Birthday Problem

- **Solution**

In a room of k people; consider q: the probability that all people have different birthdays

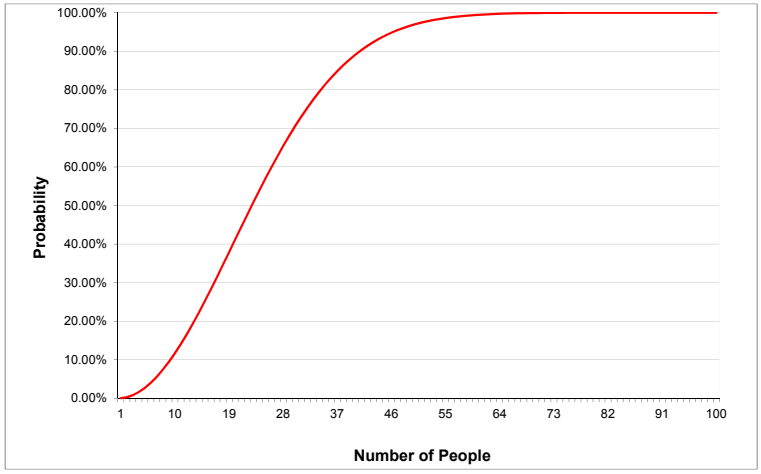
$$q = \frac{365}{365} \cdot \frac{364}{365} \cdot \frac{363}{365} \cdot \frac{362}{365} \cdots \frac{365-(k-1)}{365}$$

$$q = \frac{365!/(365-k)!}{365^k}$$

p: the prob. at least two of them have the same birthdays

$$p = 1 - q$$

The Birthday Problem



The birthday paradox

- **Assumptions**:

Consider a set of independent identically distributed integers of size B Let $r_1, \ldots, r_n \in \{1,\ldots,B\}$ a subset from B of size n, such that all of its elements are randomly selected

Theorem: when $\mathbf{n} = 1.2 \times \mathbf{B}^{1/2}$ then $\Pr[\exists i \neq j: r_i = r_j] \geq ½$

The birthday paradox

Consider q**: the probability that all integers have different values**

$$q = Pr[\forall i \neq j: r_i \neq r_j] = \left(\frac{B-1}{B}\right)\left(\frac{B-2}{B}\right) \dots \dots \left(\frac{B-n+1}{B}\right) = \prod_{i=1}^{n-1}\left(1-\frac{i}{B}\right)$$

***Let p*: the probability that at least two of them have the same values**

$$p = 1 - q == 1 - \prod_{i=1}^{n-1}\left(1-\frac{i}{B}\right)$$

Recall: $1 - x \leq e^{-x}$

Therefore:

$$p = 1 - \prod_{i=1}^{n-1}\left(1-\frac{i}{B}\right) \geq 1 - \prod_{i=1}^{n-1} e^{\frac{-i}{B}} = 1 - e^{\frac{-1}{B}\sum_{i=1}^{n-1} i} \geq 1 - e^{\frac{-n^2}{2B}}$$

When $\mathbf{n} = 1.2 \times \mathbf{B}^{1/2}$

$$p \geq 1 - e^{\frac{-n^2}{2B}} = 1 - e^{-0.72} = 0.53$$

Attacks on Hash Functions

Birthday Attack:

Let $H: M \rightarrow \{0,1\}^n$ be a hash function ($|M| >> 2^n$)

Choose $2^{n/2}$ random messages in M: $m_1, \dots, m_{2^{n/2}}$

For i = 1, …, $2^{n/2}$ compute $t_i = H(m_i) \quad \in \{0,1\}^n$

Look for a collision ($t_i = t_j$). If not found, got back to step 1.

This attacks can find a collision in time $O(2^{n/2})$ hashes

It should be noted here that a quantum computer can find a collision in time $O(2^{n/3})$ hashes

Example of Hash Functions

Function	Digest Size (Bits)	Implementation Speed	Brute Force Attack Time	Birthday Attack Time
SHA-265	256	Fast	$O(2^{256})$	$O(2^{128})$
SHA-512	512	Fast	$O(2^{512})$	$O(2^{256})$
Whirlpool	512	Slow	$O(2^{512})$	$O(2^{256})$

Learning Outcomes

At the end of this unit you should be able to:

1. Explain the meaning of message integrity.
2. Describe the security requirement of message authentication codes(MACs).
3. Construct a MAC based on symmetric ciphers.
4. Explain the principles of Hash Functions.
5. Outline the principles of Digital Signatures

Digital Signature

A digital signature on a message is additional data which provide:

1. **Data origin authentication of the signer**

   A digital signature validates the message in the sense that assurance is provided about the integrity of the message and of the identity of the entity that signed the message.

2. **Non-repudiation**

   A digital signature can be stored by anyone who receives the signed message as evidence that the message was sent and of who sent it. This evidence could later be presented to a third party who could use the evidence to resolve any dispute that relates to the contents and/or origin of the message.

Digital Ssignature Systems

Definition : a digital signature system (DSS) consists of a triple of efficient algorithm (G, S, V)

1. **A key generation algorithm** that generate a private key at random from a set of possible private keys and a corresponding public key. (pk, sk).
2. **A signing algorithm** that, given a message and a private key, produces a signature. S(sk, m) outputs t
3. **A signature verifying** algorithm that, given a message, public key and a signature, either accepts or rejects the message's claim to authenticity.

   V(pk, m, t) outputs '1' or '0'

The RSA Digital Signature Scheme

Basic Key Generation Algorithm:

1. Choose random primes p,q
2. Set **N=pq**.
3. Choose integers **e , d** s.t. **e·d = 1 (mod φ(N))**
4. Output **pk = (N, e) , sk = (p,q , d)**

The RSA Digital Signature Scheme

RSA Signing Algorithm

For each message x compute a signature y as follows:

$$F(sk, x): (Z_N)^* \rightarrow (Z_N)^*: y = RSA(x, d) = x^d$$

RSA Signature Verification Algorithm

For each received message y

1. Compute $F(pk, y): (\boldsymbol{Z_N})^* \rightarrow (\boldsymbol{Z_N})^*: RSA(y, e) = y^e$
2. If y^e = x then output 1 else output 0

Security of RSA Digital Signature Scheme

- **An example of an existential forgery attack**

Simple Attack

If Bob signed two messages $y_1 = {x_1}^d$, $y_2 = {x_2}^d$

Then the signature for $x_3 = x_1\ x_2$ can be easily forged as $y_3 = y_1 y_2$

Countermeasure:

Use collision resistant hash function before signing messages:

Non-non-repudiation

- Alice orders 100 shares of stock from Bob
- Alice computes **MAC** using symmetric key
- Stock drops, Alice claims she did not order
- Can Bob prove that Alice placed the order?
- **No!** Since Bob also knows symmetric key, he could have forged message
- **Problem:** Bob knows Alice placed the order, but he cannot prove it

Non-repudiation

- Alice orders 100 shares of stock from Bob
- Alice **signs** order with her private key
- Stock drops, Alice claims she did not order
- Can Bob prove that Alice placed the order?
- **Yes!** Only someone with Alice's private key could have signed the order
- This assumes Alice's private key is not stolen (revocation problem)