A Few Basic Elements of Communication Security

Antonio Carzaniga

Faculty of Informatics
University of Lugano

June 1, 2011

Outline

- Communication security model
- Information-theoretic privacy
- Substitution ciphers
- Intro to modern cryptography
- One-time pad
- Block siphers
- Cryptographic hash functions
- Public-key cryptosystems

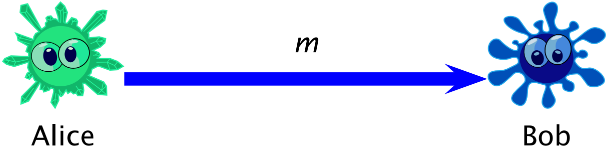
Communication Security

Communication Security

- *Communication model:* Alice sends a message *m* to Bob

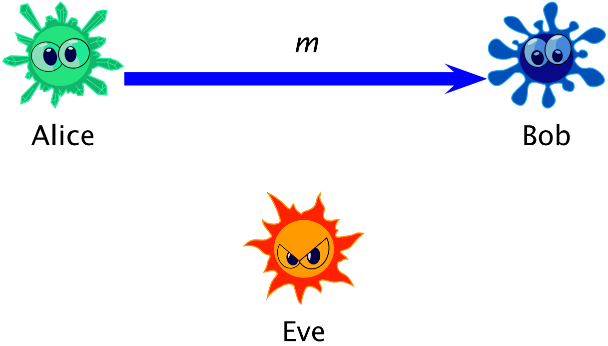
Communication Security

- *Communication model:* Alice sends a message m to Bob



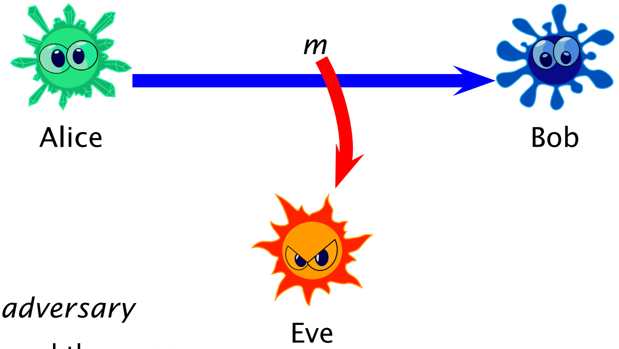
Communication Security

- *Communication model:* Alice sends a message m to Bob



Communication Security

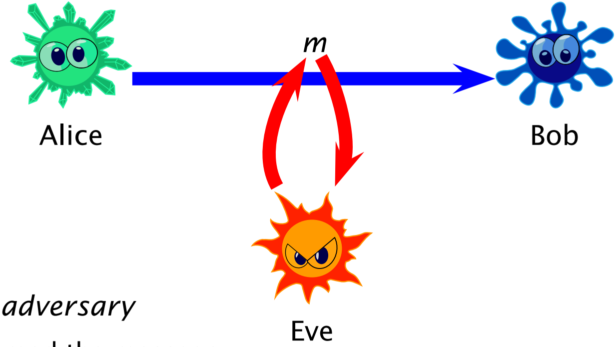
- *Communication model:* Alice sends a message m to Bob



- *Passive adversary*
  - can read the message

Communication Security

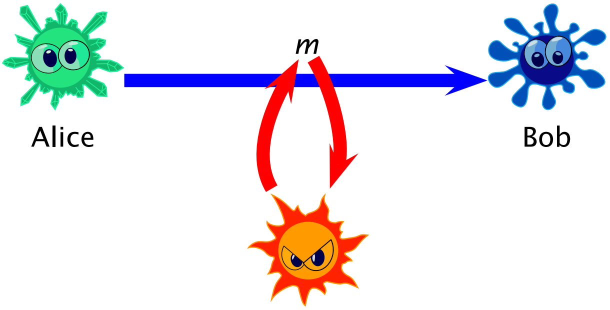
- *Communication model:* Alice sends a message *m* to Bob



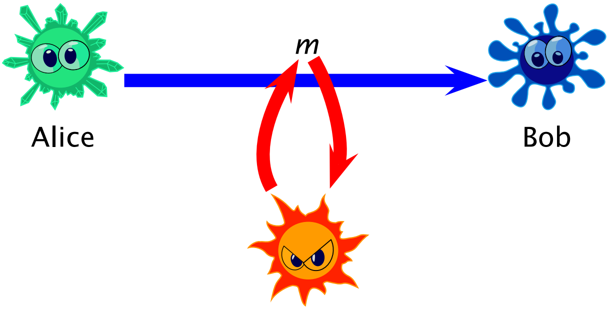
- *Passive adversary*
  - can read the message
- *Active adversary*
  - can modify the message

Goals

Goals

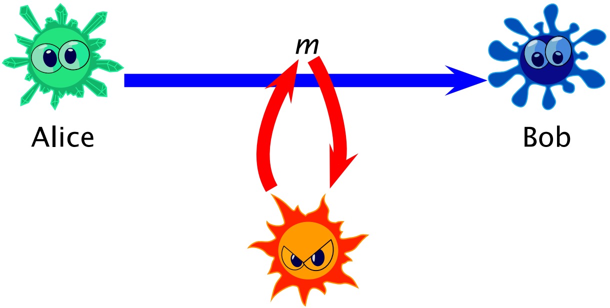


Goals



- *Confidentiality (a.k.a., privacy):* Alice wants to make sure that only Bob sees the message

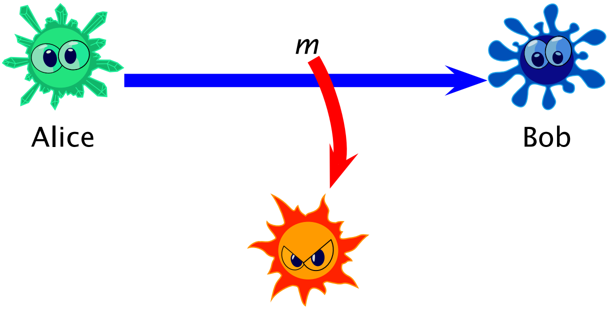
Goals



- *Confidentiality (a.k.a., privacy):* Alice wants to make sure that only Bob sees the message
- *Authentication:* Bob wants to make sure that the message he reads was exactly what Alice wrote

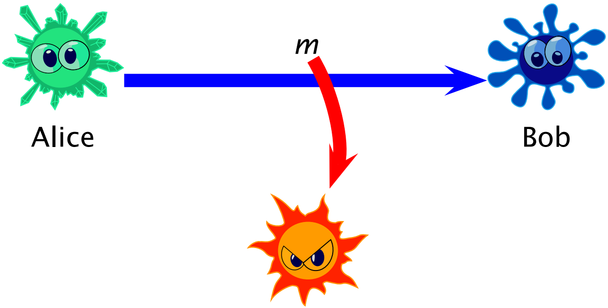
What is Privacy, Exactly?

What is Privacy, Exactly?



- Alice wants to make sure that only Bob "sees" the message

What is Privacy, Exactly?



- Alice wants to make sure that only Bob “sees” the message
- What if Eve can *guess* the message?

“Shift” Cipher

“Shift” Cipher

- The ciphertext is

```
BUUBDL BU EBXO
```

"Shift" Cipher

- The ciphertext is

  BUUBDL BU EBXO

- Plaintext is

  ATTACK AT DAWN

"Shift" Cipher

- The ciphertext is

```
BUUBDL BU EBXO
```

- Plaintext is

```
ATTACK AT DAWN
```

- How many possible ciphers?
  - How many key bits?

Problem

- Decrypt this ciphertext which is an Italian phrase encrypted with a shift-cipher:

  ```
  ulsgtlffvgklsgjhttpugkpguvz yhgbp hgtpgyp
  yvbhpgwlygauhgzlsbhgvzjayh
  ```

Substitution Cipher

Substitution Cipher

- *Substitution cipher*

Substitution Cipher

- *Substitution cipher*
  - alphabet $\Sigma = \{\text{`A'}, \text{`B'}, \ldots, \text{`Z'}, \text{`~'}\}$

Substitution Cipher

- *Substitution cipher*
  - alphabet $\Sigma = \{\text{'A'}, \text{'B'}, \ldots, \text{'Z'}, \text{' '}\}$
  - encryption function: a *permutation*

$$E : \Sigma \rightarrow \Sigma$$

Substitution Cipher

- *Substitution cipher*
  - alphabet $\Sigma = \{\text{'A'}, \text{'B'}, \ldots, \text{'Z'}, \text{' '}\}$
  - encryption function: a *permutation*

$$E : \Sigma \rightarrow \Sigma$$

Example:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_
V	Z	L	Q	X	T	_	R	D	U	C	O	J	N	F	M	G	E	H	W	P	I	S	Y	A	B	K

Substitution Cipher

- *Substitution cipher*
  - alphabet $\Sigma = \{\text{'A'}, \text{'B'}, \ldots, \text{'Z'}, \text{' '}\}$
  - encryption function: a *permutation*

$$E : \Sigma \rightarrow \Sigma$$

Example:

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z _
V Z L Q X T _ R D U C O J N F M G E H W P I S Y A B K
```

How many possible permutations?

Substitution Cipher

- *Substitution cipher*
  - alphabet $\Sigma = \{\text{'A'}, \text{'B'}, \ldots, \text{'Z'}, \text{' '}\}$
  - encryption function: a *permutation*

$$E : \Sigma \rightarrow \Sigma$$

Example:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_
V	Z	L	Q	X	T	_	R	D	U	C	O	J	N	F	M	G	E	H	W	P	I	S	Y	A	B	K

How many possible permutations?

27!

Substitution Cipher

- *Substitution cipher*
  - alphabet $\Sigma = \{\text{'A'}, \text{'B'}, \ldots, \text{'Z'}, \text{' '}\}$
  - encryption function: a *permutation*

$$E : \Sigma \rightarrow \Sigma$$

Example:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_
V	Z	L	Q	X	T	_	R	D	U	C	O	J	N	F	M	G	E	H	W	P	I	S	Y	A	B	K

How many possible permutations?

$$27! = 10888869450418352160768000000 \approx 2^{93}$$

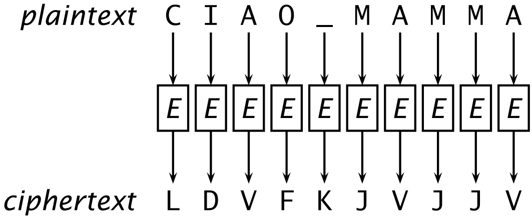
Substitution Cipher

- Encrypting some text using a substitution cipher

plaintext `C I A O _ M A M M A`

Substitution Cipher

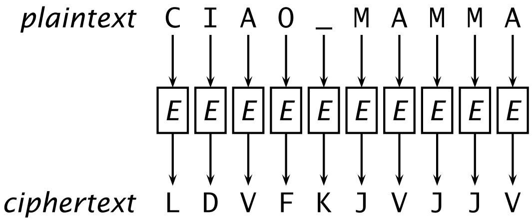
- Encrypting some text using a substitution cipher



- Problems?

Substitution Cipher

- Encrypting some text using a substitution cipher



- Problems?
  - easy to break just by *guessing!*
  - ...

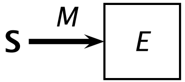
Symmetric Encryption

Symmetric Encryption

S

R

Symmetric Encryption

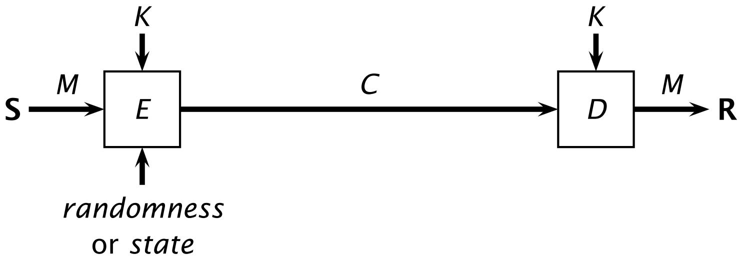


R

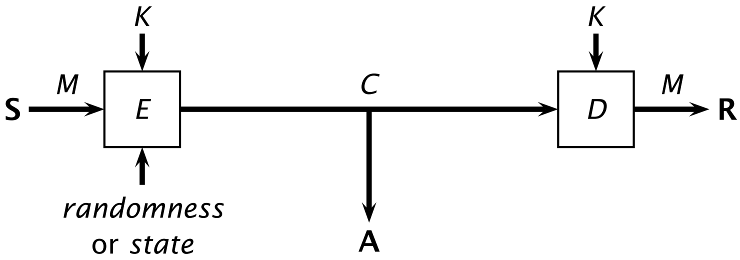
Symmetric Encryption

Symmetric Encryption

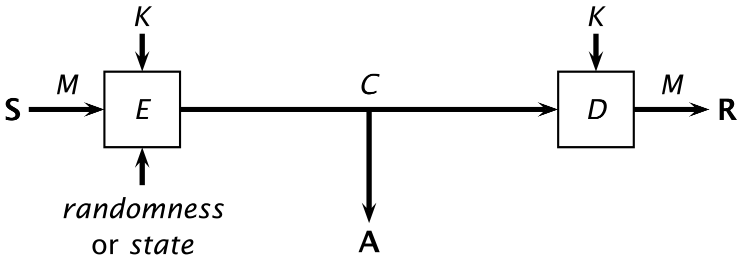
Symmetric Encryption



Symmetric Encryption



Symmetric Encryption



S	sender
R	receiver
A	adversary
E	encryption algorithm
D	dencryption algorithm
M	plaintext message
C	ciphertext message
K	key

One-Time Pad

One-Time Pad

- *Assumptions:* the message M and the key K are two n-bit strings

$$M \in \{0,1\}^n; \qquad K \xleftarrow{\$} \{0,1\}^n$$

One-Time Pad

- *Assumptions:* the message M and the key K are two n-bit strings

  $$M \in \{0,1\}^n; \qquad K \xleftarrow{\$} \{0,1\}^n$$

  the key K is chosen uniformly at random from $\{0,1\}^n$

One-Time Pad

- *Assumptions:* the message M and the key K are two n-bit strings

$$M \in \{0,1\}^n; \qquad K \xleftarrow{\$} \{0,1\}^n$$

  the key K is chosen uniformly at random from $\{0,1\}^n$

- *Scheme*
  - encryption:

$$E(K,M) := M \oplus K$$

    the key K is then thrown away an never reused
  - decryption:

$$D(K,C) := C \oplus K$$

One-Time Pad

- *Assumptions:* the message M and the key K are two n-bit strings

$$M \in \{0,1\}^n; \qquad K \xleftarrow{\$} \{0,1\}^n$$

  the key K is chosen uniformly at random from $\{0,1\}^n$

- *Scheme*
  - encryption:

$$E(K, M) := M \oplus K$$

    the key K is then thrown away an never reused
  - decryption:

$$D(K, C) := C \oplus K$$

- **Example:** M 0110010110111011

One-Time Pad

- *Assumptions:* the message M and the key K are two n-bit strings

$$M \in \{0,1\}^n; \qquad K \xleftarrow{\$} \{0,1\}^n$$

  the key K is chosen uniformly at random from $\{0,1\}^n$

- *Scheme*
  - encryption:

$$E(K,M) := M \oplus K$$

    the key K is then thrown away an never reused
  - decryption:

$$D(K,C) := C \oplus K$$

- **Example:**

M	0110010110111011
K	1011000101000101

One-Time Pad

- *Assumptions:* the message M and the key K are two n-bit strings

$$M \in \{0,1\}^n; \qquad K \stackrel{\$}{\leftarrow} \{0,1\}^n$$

  the key K is chosen uniformly at random from $\{0,1\}^n$

- *Scheme*
  - encryption:

$$E(K,M) := M \oplus K$$

    the key K is then thrown away an never reused
  - decryption:

$$D(K,C) := C \oplus K$$

- **Example:**

M	0110010110111011
K	1011000101000101
C	1101010011111110

So, What is Privacy Exactly?

So, What is Privacy Exactly?

- A scheme is secure if *we learn nothing from the ciphertext C*

So, What is Privacy Exactly?

- A scheme is secure if *we learn nothing from the ciphertext C*
- A more formal definition:
  let $K \stackrel{\$}{\leftarrow} \mathcal{K}$; for every $m_1 \neq m_2 \in \mathcal{M}$, and for any C

So, What is Privacy Exactly?

- A scheme is secure if *we learn nothing from the ciphertext C*
- A more formal definition:
  let $K \xleftarrow{\$} \mathcal{K}$; for every $m_1 \neq m_2 \in \mathcal{M}$, and for any C

$$\Pr_{K \in \mathcal{K}}[E_K(m_1) = C] = \Pr_{K \in \mathcal{K}}[E_K(m_2) = C]$$

So, What is Privacy Exactly?

- A scheme is secure if *we learn nothing from the ciphertext C*
- A more formal definition:
  let $K \stackrel{\$}{\leftarrow} \mathcal{K}$; for every $m_1 \neq m_2 \in \mathcal{M}$, and for any C

$$\Pr_{K \in \mathcal{K}}[E_K(m_1) = C] = \Pr_{K \in \mathcal{K}}[E_K(m_2) = C]$$

- *Given a ciphertext C, every plaintext m is equiprobable*
  - so, seeing any particular $C = E_K(M)$ tells us *nothing* about M

So, What is Privacy Exactly?

- A scheme is secure if *we learn nothing from the ciphertext C*
- A more formal definition:
  let $K \stackrel{\$}{\leftarrow} \mathcal{K}$; for every $m_1 \neq m_2 \in \mathcal{M}$, and for any C

$$\Pr_{K\in\mathcal{K}}[E_K(m_1) = C] = \Pr_{K\in\mathcal{K}}[E_K(m_2) = C]$$

- *Given a ciphertext C, every plaintext m is equiprobable*
  - so, seeing any particular $C = E_K(M)$ tells us *nothing* about M
- Is a shift cipher perfectly secure?

So, What is Privacy Exactly?

- A scheme is secure if *we learn nothing from the ciphertext C*
- A more formal definition:
  let $K \xleftarrow{\$} \mathcal{K}$; for every $m_1 \neq m_2 \in \mathcal{M}$, and for any C

$$\Pr_{K \in \mathcal{K}}[E_K(m_1) = C] = \Pr_{K \in \mathcal{K}}[E_K(m_2) = C]$$

- *Given a ciphertext C, every plaintext m is equiprobable*
  - so, seeing any particular $C = E_K(M)$ tells us *nothing* about M
- Is a shift cipher perfectly secure?
- Is a substitution cipher perfectly secure?

So, What is Privacy Exactly?

- A scheme is secure if *we learn nothing from the ciphertext C*
- A more formal definition:
  let $K \stackrel{\$}{\leftarrow} \mathcal{K}$; for every $m_1 \neq m_2 \in \mathcal{M}$, and for any C

$$\Pr_{K \in \mathcal{K}}[E_K(m_1) = C] = \Pr_{K \in \mathcal{K}}[E_K(m_2) = C]$$

- *Given a ciphertext C, every plaintext m is equiprobable*
  - so, seeing any particular $C = E_K(M)$ tells us *nothing* about M
- Is a shift cipher perfectly secure?
- Is a substitution cipher perfectly secure?
- Is one-time-pad perfectly secure?

The Cost of Perfect Privacy

The Cost of Perfect Privacy

- *Perfect privacy* implies that

$$|\mathcal{K}| \geq |\mathcal{M}|$$

The Cost of Perfect Privacy

- *Perfect privacy* implies that

$$|\mathcal{K}| \geq |\mathcal{M}|$$

- *Proof:* assume not.
  Fix a possible ciphertext C, i.e., there is a message m and a key k such that $E_K(m) = C$, and $\Pr_{K\in\mathcal{K}}[E_K(m) = C] > 0$

  Let $P_C = \{m \in \mathcal{M}$ such that $E_k(m) = C$ for some $k\}$

  Since every k maps exactly one message m to C, and since we have fewer keys than messages, then there is an $m' \notin P_C$ such that no key k maps m' to C; therefore $\Pr_{K\in\mathcal{K}}[E_K(m') = C] = 0$, which violates the perfect-secrecy condition that for all m and m', $\Pr_{K\in\mathcal{K}}[E_K(m) = C] = \Pr_{K\in\mathcal{K}}[E_K(m') = C]$

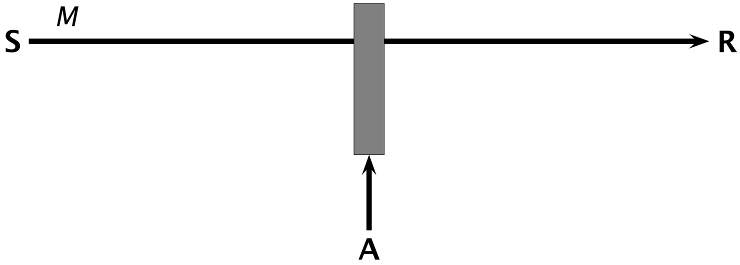
Message Authenticity

Message Authenticity

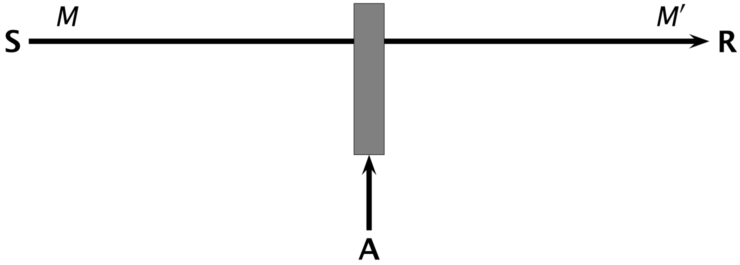
S **R**

Message Authenticity

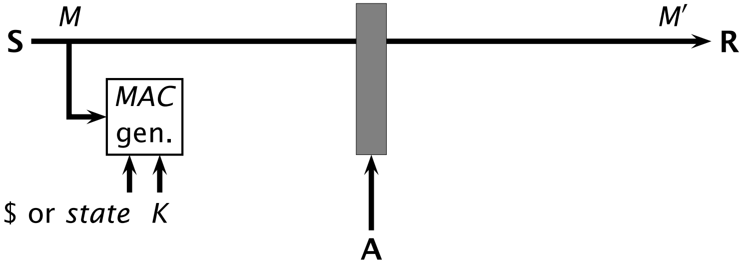
Message Authenticity



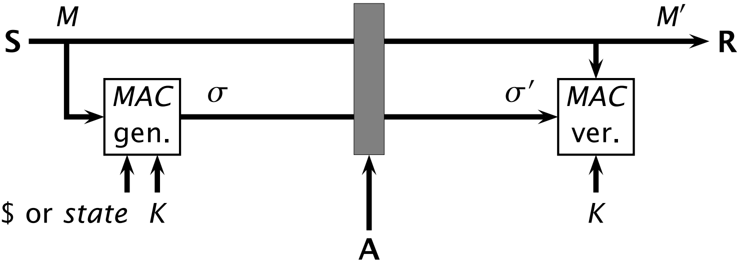
Message Authenticity



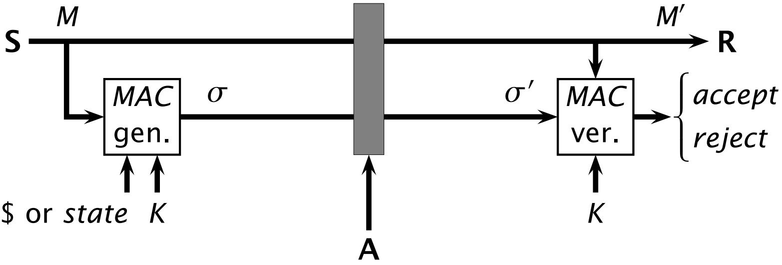
Message Authenticity



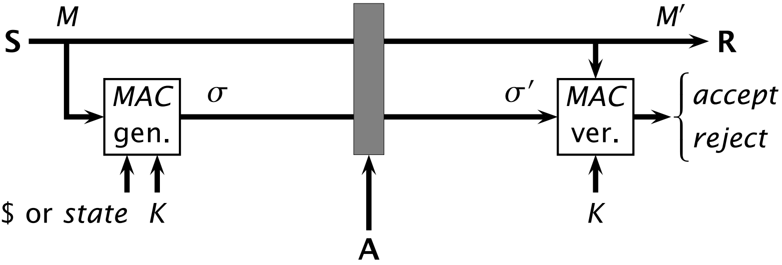
Message Authenticity



Message Authenticity



Message Authenticity



σ	message authentication code (MAC)
K	key
$	randomness
MAC gen.	MAC *generation* algorithm
MAC ver.	MAC *verification* algorithm

Asymmetric Encryption

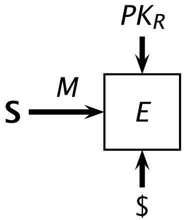
Asymmetric Encryption

S **R**

Asymmetric Encryption

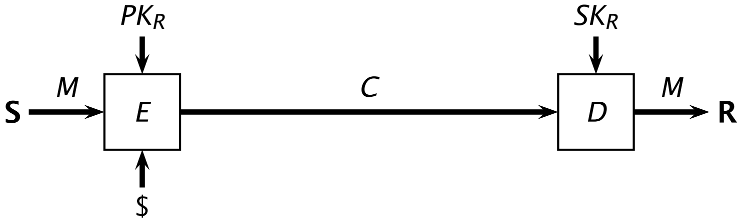
Asymmetric Encryption

Asymmetric Encryption

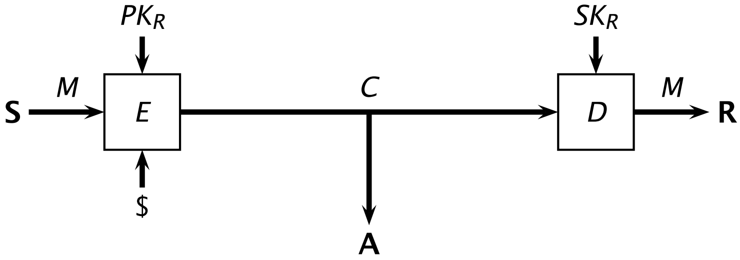


R

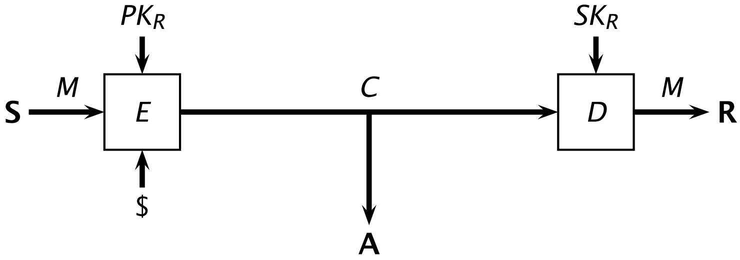
Asymmetric Encryption



Asymmetric Encryption



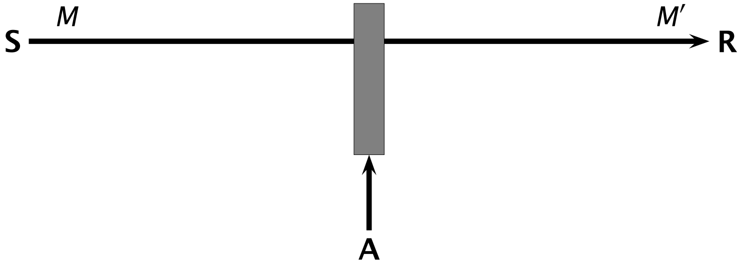
Asymmetric Encryption



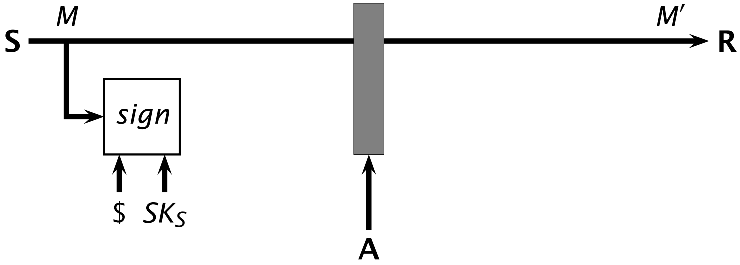
PK_R	receiver's *public key*
SK_R	receiver's *secret key*
M	plaintext message
C	ciphertext message

Digital Signatures

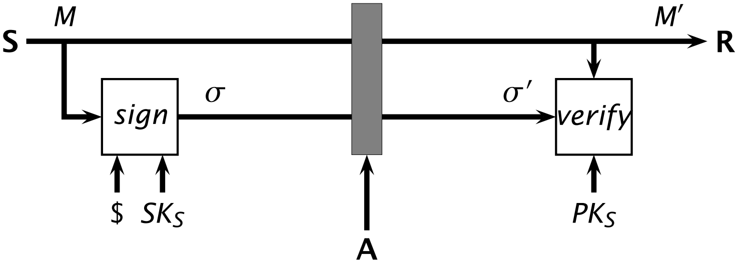
Digital Signatures



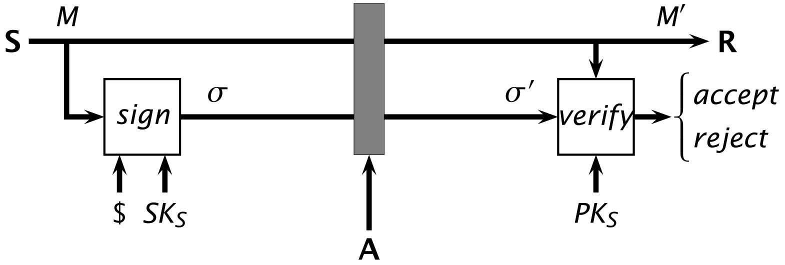
Digital Signatures



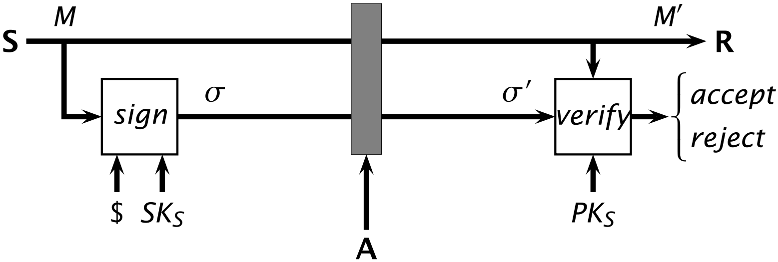
Digital Signatures



Digital Signatures



Digital Signatures



σ	digital signature
SK_S	sender's *secret key*
PK_S	sender's *public key*
\$	randomness
sign	*signing* algorithm
verify	*verification* algorithm

Primitives vs. Protocols

Primitives vs. Protocols

- *Protocol*
  - an *algorithm*
  - solves a specific security problem (e.g., signing a message)

Primitives vs. Protocols

- *Protocol*
  - an *algorithm*
  - solves a specific security problem (e.g., signing a message)
- *Primitive*

Primitives vs. Protocols

- *Protocol*
  - an *algorithm*
  - solves a specific security problem (e.g., signing a message)
- *Primitive*
  - also an *algorithm*
  - the elementary subroutines of protocols
  - implement (try to approximate) well-defined mathematical object
  - embody “hard problems”

Stream Ciphers

Stream Ciphers

- A *stream cipher* is a generator of a *pseudo-random streams*

Stream Ciphers

- A *stream cipher* is a generator of a *pseudo-random streams*
  - given an initialization key K
  - generates an infinite pseudo-random sequence of bits

Stream Ciphers

- A *stream cipher* is a generator of a *pseudo-random streams*
  - given an initialization key K
  - generates an infinite pseudo-random sequence of bits
- E.g., RC4

Padding with a Stream Cipher

Padding with a Stream Cipher

- *Assumptions:* S and R share a secret key K and agree to use a stream cipher S_K
  - S and R maintain some *state:* position s initialized to $s = 0$

Padding with a Stream Cipher

- *Assumptions:* S and R share a secret key K and agree to use a stream cipher S_K
  - S and R maintain some *state:* position s initialized to $s = 0$

- *Encryption protocol*

1. S computes $C \leftarrow M \oplus S_K[s \ldots s + |M| - 1]$
2. S updates its position $s \leftarrow s + |M|$

Padding with a Stream Cipher

- *Assumptions:* S and R share a secret key K and agree to use a stream cipher S_K
  - S and R maintain some *state:* position s initialized to $s = 0$

- *Encryption protocol*
  1. S computes $C \leftarrow M \oplus S_K[s \ldots s + |M| - 1]$
  2. S updates its position $s \leftarrow s + |M|$

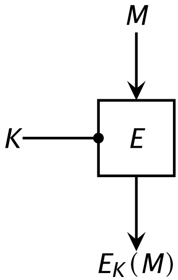
- *Dencryption protocol*
  1. R computes $M \leftarrow C \oplus S_K[s \ldots s + |C| - 1]$
  2. R updates its position $s \leftarrow s + |C|$

Block Ciphers

- *Block Cipher:* $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$

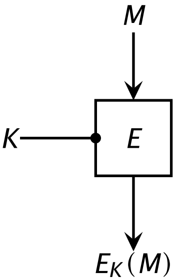
Block Ciphers

- *Block Cipher:* $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$



Block Ciphers

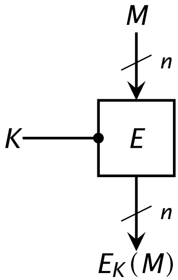
- *Block Cipher:* $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$



  - $E_K(\cdot)$ is a *permutation*, so $E_K^{-1}(\cdot)$ is always defined

Block Ciphers

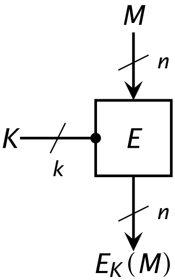
- *Block Cipher:* $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$



  - $E_K(\cdot)$ is a *permutation*, so $E_K^{-1}(\cdot)$ is always defined
  - fixed-length input and output (n)

Block Ciphers

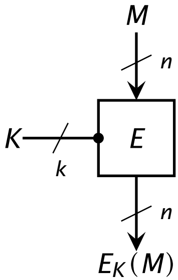
- *Block Cipher:* $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$



- $E_K(\cdot)$ is a *permutation*, so $E_K^{-1}(\cdot)$ is always defined
- fixed-length input and output (n)
- fixed-length key (k)

Block Ciphers

- *Block Cipher:* $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$



  - $E_K(\cdot)$ is a *permutation*, so $E_K^{-1}(\cdot)$ is always defined
  - fixed-length input and output (n)
  - fixed-length key (k)
  - e.g., DES, AES

An Encryption Protocol

An Encryption Protocol

- *Symmetric encryption*
  - *Input:* k-bit key K, N-bit message M
  - *Output:* N-bit ciphertext C

An Encryption Protocol

- *Symmetric encryption*
  - *Input:* k-bit key K, N-bit message M
  - *Output:* N-bit ciphertext C

- *Cipher Block Chaining (CBC)*
  - use a block cipher $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$
  - split M into n-bit blocks $M = M_0 || M_1 || \ldots || M_\ell \quad (\ell = \lfloor N/n \rfloor)$

An Encryption Protocol

- *Symmetric encryption*
  - *Input:* k-bit key K, N-bit message M
  - *Output:* N-bit ciphertext C
- *Cipher Block Chaining (CBC)*
  - use a block cipher $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$
  - split M into n-bit blocks $M = M_0 || M_1 || \ldots || M_\ell \quad (\ell = \lfloor N/n \rfloor)$

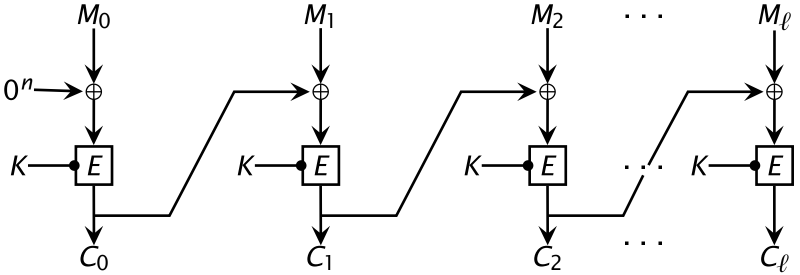
CBC(K, M)

$x \leftarrow 0^n$
for $i \leftarrow 0$ **to** $\lfloor |M|/n \rfloor$
    do $C[ni \ldots ni + n - 1] \leftarrow E_K(x \oplus M[ni \ldots ni + n - 1])$
        $x \leftarrow C[ni \ldots ni + n - 1]$
return C

An Encryption Protocol

- *Symmetric encryption*
  - *Input:* k-bit key K, N-bit message M
  - *Output:* N-bit ciphertext C

- *Cipher Block Chaining (CBC)*
  - use a block cipher $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$
  - split M into n-bit blocks $M = M_0||M_1||\ldots||M_\ell \quad (\ell = \lfloor N/n \rfloor)$



Exercise

- Write the decryption algorithm for CBC

Exercise

- Write the decryption algorithm for CBC

CBC-Decrypt(K, C)

$x \leftarrow 0^n$
for $i \leftarrow 0$ **to** $\lfloor |C|/n \rfloor$
    do $M[ni \dots ni+n-1] \leftarrow x \oplus E_K^{-1}(C[ni \dots ni+n-1])$
        $x \leftarrow C[ni \dots ni+n-1]$
return M

An Encryption Protocol (2)

An Encryption Protocol (2)

- Is this CBC protocol secure?

An Encryption Protocol (2)

- Is this CBC protocol secure?
  - *any deterministic stateless protocol is insecure*
  - we need *state* and/or *randomness*

An Encryption Protocol (2)

- Is this CBC protocol secure?
  - *any deterministic stateless protocol is insecure*
  - we need *state* and/or *randomness*
- What if $|M| \neq 0 \mod n$?

An Encryption Protocol (2)

- Is this CBC protocol secure?
  - *any deterministic stateless protocol is insecure*
  - we need *state* and/or *randomness*
- What if $|M| \neq 0 \mod n$?
- Is CBC parallelizable?

CBC With Random IV

- *CBC$:* cipher block chaining with random IV

CBC With Random IV

- *CBC\$:* cipher block chaining with random IV

CBC\$-Encrypt(K, M)

if $|M| = 0 \vee |M| \neq 0 \mod n$
  then return $\perp$
$M[1] \cdot M[2] \cdots M[\ell] \leftarrow M$
$IV \stackrel{\$}{\leftarrow} \{0,1\}^n$
$C[0] \leftarrow IV$
for $i \leftarrow 1$ **to** ℓ
    do $C[i] \leftarrow E_K(C[i-1] \oplus M[i])$
$C \leftarrow C[1] \cdot C[2] \cdots C[\ell]$
return $\langle IV, C \rangle$

CBC With Random IV (2)

- *CBC$:* cipher block chaining with random IV (decryption)

CBC With Random IV (2)

- *CBC\$:* cipher block chaining with random IV (decryption)

CBC\$-Decrypt(K, IV, C)

if $|C| = 0 \vee |C| \neq 0 \mod n$
  then return $\perp$
$C[1] \cdot C[2] \cdots C[\ell] \leftarrow C$
$C[0] \leftarrow IV$
for $i \leftarrow 1$ **to** ℓ
  do $M[i] \leftarrow C[i-1] \oplus E_K(C[i])$
$M \leftarrow M[1] \cdot M[2] \cdots M[\ell]$
return M

CBC With Stateful Counter

- *CBCC:* cipher block chaining with stateful counter

CBC With Stateful Counter

- *CBCC:* cipher block chaining with stateful counter

CBCC-Encrypt(K, M)

static $ctr \leftarrow 0$
if $ctr \geq 2^n \vee |M| = 0 \vee |M| \neq 0 \mod n$
  then return $\perp$
$M[1] \cdot M[2] \cdots M[\ell] \leftarrow M$
$IV \leftarrow [ctr]_n$
$C[0] \leftarrow [ctr]_n$
for $i \leftarrow 1$ **to** ℓ
    do $C[i] \leftarrow E_K(C[i-1] \oplus M[i])$
$C \leftarrow C[1] \cdot C[2] \cdots C[\ell]$
$ctr \leftarrow ctr + 1$
return $\langle IV, C \rangle$

CBC With Stateful Counter (2)

- *CBCC:* cipher block chaining with stateful counter

CBC With Stateful Counter (2)

- *CBCC:* cipher block chaining with stateful counter

CBCC-Decrypt(K, IV, C)

if $IV + |C| \geq 2^n \vee |C| = 0 \vee |C| \neq 0 \mod n$
    then return $\perp$
$C[1] \cdot C[2] \cdots C[\ell] \leftarrow C$
$IV \leftarrow [ctr]_n$
$C[0] \leftarrow IV$
for $i \leftarrow 1$ **to** ℓ
        do $M[i] \leftarrow C[i-1] \oplus E_K^{-1}(C[i])$
$M \leftarrow M[1] \cdot M[2] \cdots M[\ell]$
return M

Counter Mode

- *CTR$:* counter mode with random initial counter

Counter Mode

- *CTR\$:* counter mode with random initial counter
  - *family of functions:* $F : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$

Counter Mode

- *CTR\$:* counter mode with random initial counter
  - *family of functions:* $F : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$

CTR\$-Encrypt(K, M)

$R \xleftarrow{\$} \{0,1\}^n$
$Pad \leftarrow F_K([R]_n)$
for $i \leftarrow 1$ **to** $\lceil |M| / n \rceil - 1$
    do $Pad \leftarrow Pad \cdot F_K([R + i]_n)$
$Pad \leftarrow$ first $|M|$ bits of Pad
$C \leftarrow M \oplus Pad$
return $\langle R, C \rangle$

Counter Mode (2)

- *CTR$:* counter mode with random initial counter (decryption)
  - *family of functions:* $F : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$

Counter Mode (2)

- *CTR\$:* counter mode with random initial counter (decryption)
  - *family of functions:* $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$

CTR\$-Decrypt(K, R, C)

$Pad \leftarrow F_K([R]_n)$
for $i \leftarrow 1$ **to** $\lceil |C|/n \rceil - 1$
    do $Pad \leftarrow Pad \cdot F_K([R+i]_n)$
$Pad \leftarrow$ first $|C|$ bits of Pad
$M \leftarrow C \oplus Pad$
return M

Counter Mode (3)

- *CTRC:* counter mode with stateful counter
  - *family of functions:* $F : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$

Counter Mode (3)

- *CTRC:* counter mode with stateful counter
  - *family of functions:* $F : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$

```
CTRC(K, M)
    static R ← 0
    ℓ ← ⌈|M|/n⌉
    if R + ℓ − 1 ≥ 2^n
        then return ⊥
    Pad ← F_K([R]_n)
    for i ← 1 to ℓ − 1
            do Pad ← Pad · F_K([R + i]_n)
    Pad ← first |M| bits of Pad
    C ← M ⊕ Pad
    R ← R + ℓ
    return ⟨R − ℓ, C⟩
```

Counter Mode (4)

- *CTRC:* counter mode with stateful counter (decryption)
  - *family of functions:* $F : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$

Counter Mode (4)

- *CTRC:* counter mode with stateful counter (decryption)
  - *family of functions:* $F : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$

CTRC-Decrypt(K, R, C)

$Pad \leftarrow F_K([R]_n)$
for $i \leftarrow 1$ **to** $\lceil |C|/n \rceil - 1$
    do $Pad \leftarrow Pad \cdot F_K([R+i]_n)$
$Pad \leftarrow$ first $|C|$ bits of Pad
$M \leftarrow C \oplus Pad$
return M

Authentication Protocol

- *MAC generation*
  - *Input:* k-bit key K, N-bit message M
  - *Output:* n-bit message authentication code σ

Authentication Protocol

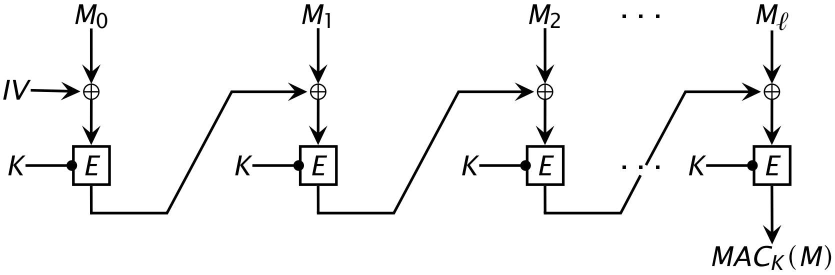
- *MAC generation*
  - *Input:* k-bit key K, N-bit message M
  - *Output:* n-bit message authentication code σ

- *CBC with random IV*
  - use a block cipher $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$
  - split M into n-bit blocks $M = M_0 || M_1 || \ldots || M_\ell \quad (\ell = \lfloor N/n \rfloor)$

```
MAC(K, M)
IV ←$ {0,1}^n
C ← IV
for i ← 0 to ⌊|M|/n⌋
        do C ← E_K(C ⊕ M[ni ... ni + n − 1])
return ⟨IV, C⟩
```

Authentication Protocol

- *MAC generation*
  - *Input:* k-bit key K, N-bit message M
  - *Output:* n-bit message authentication code σ
- *CBC with random IV*
  - use a block cipher $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$
  - split M into n-bit blocks $M = M_0 || M_1 || \ldots || M_\ell \quad (\ell = \lfloor N/n \rfloor)$



CBC MAC: Generation

- *CBC MAC:* cipher block chaining MAC with random IV

CBC MAC: Generation

- *CBC MAC:* cipher block chaining MAC with random IV

CBC-MAC\$$(K, M)$

if $|M| = 0 \vee |M| \neq 0 \mod n$
  then return $\perp$
$M[1] \cdot M[2] \cdots M[\ell] \leftarrow M$
$IV \xleftarrow{\$} \{0,1\}^n$
$C \leftarrow IV$
for $i \leftarrow 1$ **to** ℓ
  do $C \leftarrow E_K(C \oplus M[i])$
return $\langle IV, C \rangle$

CBC MAC: Verification

- *CBC MAC:* cipher block chaining MAC with random IV

CBC MAC: Verification

- *CBC MAC:* cipher block chaining MAC with random IV

CBC-MAC\$-Verify(K, IV, σ, M)

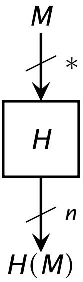
if $|M| = 0 \lor |M| \neq 0 \mod n$
    then return $\perp$
$M[1] \cdot M[2] \cdots M[\ell] \leftarrow M$
$C \leftarrow IV$
for $i \leftarrow 1$ **to** ℓ
        do $C \leftarrow E_K(C \oplus M[i])$
if $C = \sigma$
    then return Accept
    else return Reject

Cryptographic Hash Functions

- *Cryptographic Hash:* $H : \{0,1\}^* \to \{0,1\}^n$

Cryptographic Hash Functions

- *Cryptographic Hash:* $H : \{0,1\}^* \rightarrow \{0,1\}^n$



Cryptographic Hash Functions

- *Cryptographic Hash:* $H : \{0,1\}^* \rightarrow \{0,1\}^n$
  - $H(\cdot)$ is a good *hash* function when (*informally*)

$$\forall m \in \{0,1\}^*, h \in \{0,1\}^n, \Pr[H(m) = h] = \frac{1}{2^n}$$

Cryptographic Hash Functions

- *Cryptographic Hash:* $H : \{0,1\}^* \rightarrow \{0,1\}^n$
  - $H(\cdot)$ is a good *hash* function when (*informally*)

$$\forall m \in \{0,1\}^*, h \in \{0,1\}^n, \Pr[H(m) = h] = \frac{1}{2^n}$$

  - it is "difficult" to find *collisions*

$$\text{find } m_1, m_2 \in \{0,1\}^* : m_1 \neq m_2, H(m_1) = H(m_2)$$

Cryptographic Hash Functions

- *Cryptographic Hash:* $H : \{0,1\}^* \rightarrow \{0,1\}^n$
  - $H(\cdot)$ is a good *hash* function when (*informally*)

$$\forall m \in \{0,1\}^*, h \in \{0,1\}^n, \Pr[H(m) = h] = \frac{1}{2^n}$$

  - it is “difficult” to find *collisions*

$$\text{find } m_1, m_2 \in \{0,1\}^* : m_1 \neq m_2, H(m_1) = H(m_2)$$

  - it is “difficult” to find a *preimage*

$$\text{given } m \in \{0,1\}^*, \text{ find } m' : H(m') = m$$

Cryptographic Hash Functions

- *Cryptographic Hash:* $H : \{0,1\}^* \to \{0,1\}^n$
  - $H(\cdot)$ is a good *hash* function when (*informally*)

$$\forall m \in \{0,1\}^*, h \in \{0,1\}^n, \Pr[H(m) = h] = \frac{1}{2^n}$$

  - it is “difficult” to find *collisions*

$$\text{find } m_1, m_2 \in \{0,1\}^* : m_1 \neq m_2, H(m_1) = H(m_2)$$

  - it is “difficult” to find a *preimage*

$$\text{given } m \in \{0,1\}^*, \text{ find } m' : H(m') = m$$

  - e.g., SHA-1

Summary

- Basic ingredients: cryptographic primitives
  - secret-key (symmetric) cryptography (e.g., AES)
  - public-key (asymmetric) cryptography (e.g., RSA)
  - cryptographic hash functions (e.g., SHA-1)
  - stream ciphers (e.g., RC4)

Summary

- Basic ingredients: cryptographic primitives
  - secret-key (symmetric) cryptography (e.g., AES)
  - public-key (asymmetric) cryptography (e.g., RSA)
  - cryptographic hash functions (e.g., SHA-1)
  - stream ciphers (e.g., RC4)

- Recipes: cryptographic protocols
  - certificates (e.g., X.509)
  - secure transport (e.g., TLS, IPSec)
  - ...

Summary

- Basic ingredients: cryptographic primitives
  - secret-key (symmetric) cryptography (e.g., AES)
  - public-key (asymmetric) cryptography (e.g., RSA)
  - cryptographic hash functions (e.g., SHA-1)
  - stream ciphers (e.g., RC4)

- Recipes: cryptographic protocols
  - certificates (e.g., X.509)
  - secure transport (e.g., TLS, IPSec)
  - ...

- Applications
  - electronic commerce
  - secure shell
  - secure electronic mail
  - virtual private networks
  - ...