MONASH University
Information Technology

FIT2093 INTRODUCTION TO CYBER SECURITY

MONASH University
Information Technology

FIT2093 INTRODUCTION TO CYBER SECURITY

Digital Signature & Integrity Management

Unit Structure

- Introduction to security of
- Authentication
- Access Control Fundamental
- Fundamental concepts of cryptography
- Symmetric encryption techniques
- Introduction to number theory
- Public key cryptography
- Integrity management
- Practical aspects of cyber security
- Hacking and countermeasures
- Database security
- IT risk management & Ethics and privacy

Previous Lecture

- **Why public key cryptography ?**
- **General principles of public key cryptography**
- **Diffie-Hellman public key exchange**
  - g, p known to Alice & Bob and select private keys a, b
  - Exchange public keys $A = g^a \bmod p$; $B = g^b \bmod p$;
  - Calculate shared secret key, $K = B^a \bmod p = A^b \bmod p$
- **RSA public key cryptosystem**
  - Primes p,q => n=pq; e=public key; d = private key;
  - $ed \bmod \Phi(n)=1$; $C=m^e \bmod n$; $m = C^d \bmod n$;
- **RSA Security**

Outline

- **Why do we need digital signatures?**
- **Another look at the Public Key Cryptography**
- **How are the required properties satisfied by digital signatures?**
- **How to generate a digital signature?**
- **Integrity Management**
  - Verification of modification

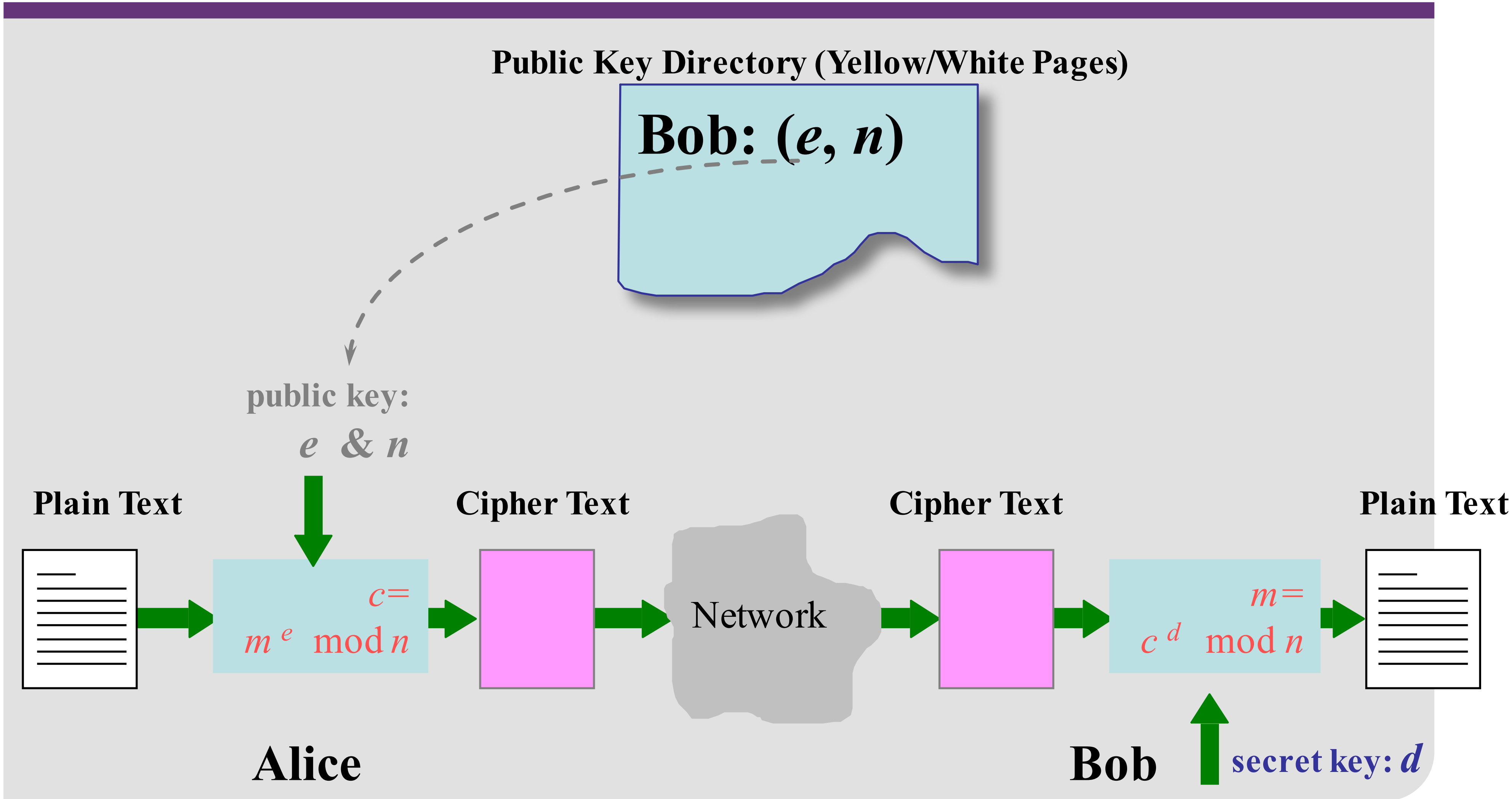
The Need for a Digital Signature

- **Social & business activities and their associated documents are becoming digital**
  - digital conferences
  - digital contract signing
  - digital cash payments, ...
- **Hand-written signatures are not applicable to digital data**

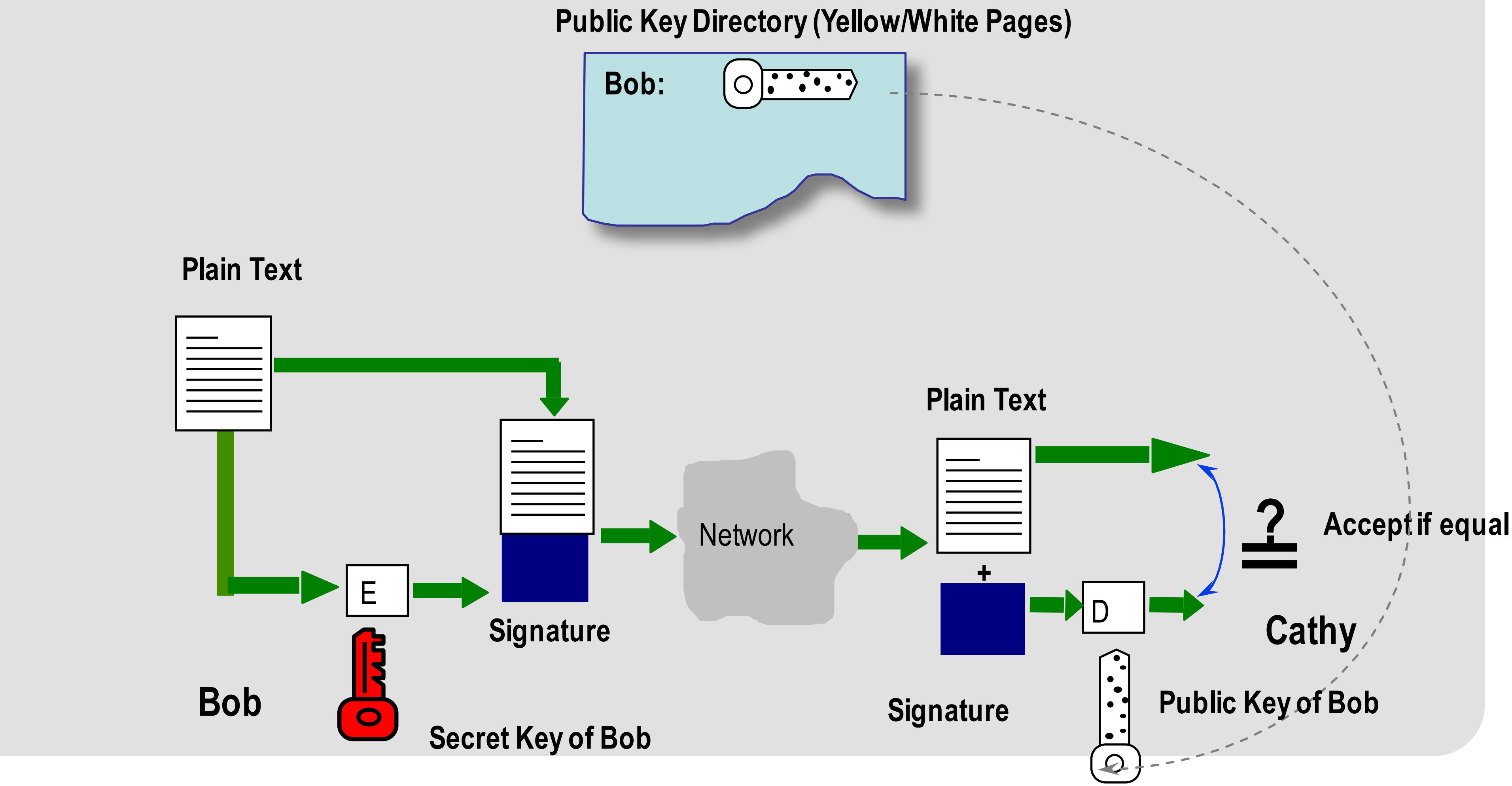
Digital Signatures

- **Signatures are used for non-repudiation**
  - Authenticate/verify the source/issuer
- **Non-repudiation is the property where an entity cannot deceive another by falsely denying responsibility for an act**
- **Can also be used for integrity of data, if the signature contains non-forgeable information about the data.**
  - Detect forgery or tampering

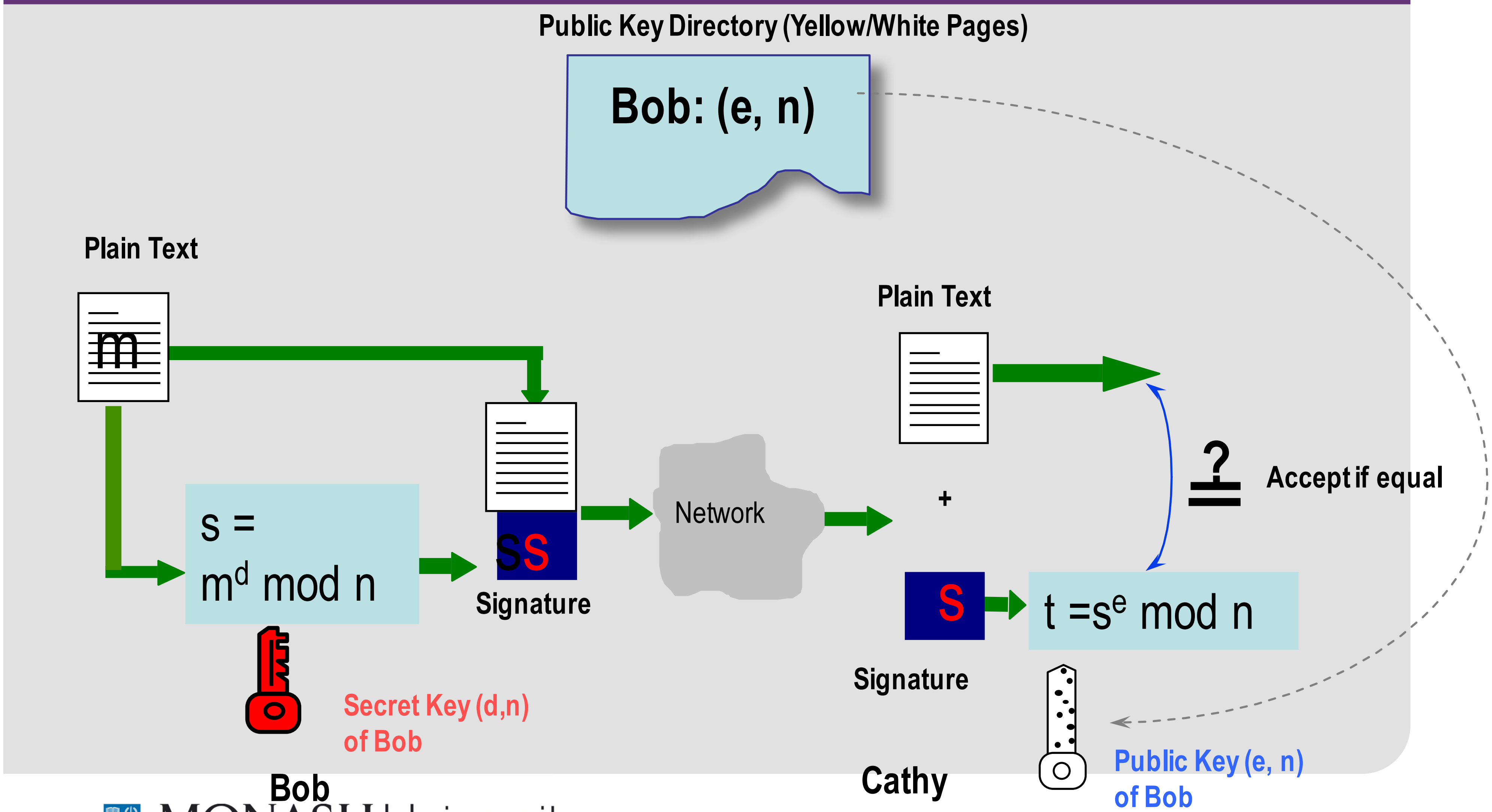
RSA Public Key Cryptosystem



Digital Signature (based on Public Key Cryptography)



Digital Signature (based on RSA)



Digital Signature --- an example(1)

- **Bob:**
  - chooses 2 primes: $p=5, q=11$
    multiplies p and q: $n = p*q = 55;$

    $$\phi(n)=(p-1)*(q-1)=4*10=40$$
  - finds out two numbers $e=3$ & $d=27$ which satisfy

    e*d mod $\phi(n)=1;$

    $(3 * 27) \ mod \ 40 = 1 ;$
  - Bob's public key
    - > 2 numbers: $(3, 55)$
    - > encryption alg: modular exponentiation
  - secret key: $(27,55)$

Digital Signature --- an example(2)

- **Bob has a document *m=19* to sign:**
  - uses his secret key *d=27* to calculate the digital signature of *m=19*:

$$s = m^d \ (mod\ n)$$
$$= 19^{27} \ mod\ 55 = (19^3)^9 mod55 = (31X19)^9 mod55$$
$$= (39^3)^3 mod55 = (36X39)^3 mod55 = 29^3 mod55$$
$$= (16X29) mod55$$
$$= 24$$

  - appends 24 to 19. Now *(m, s) = (19, 24)* indicates that the doc is 19, and Bob's signature on the doc is 24.

MONASH University
Information Technology

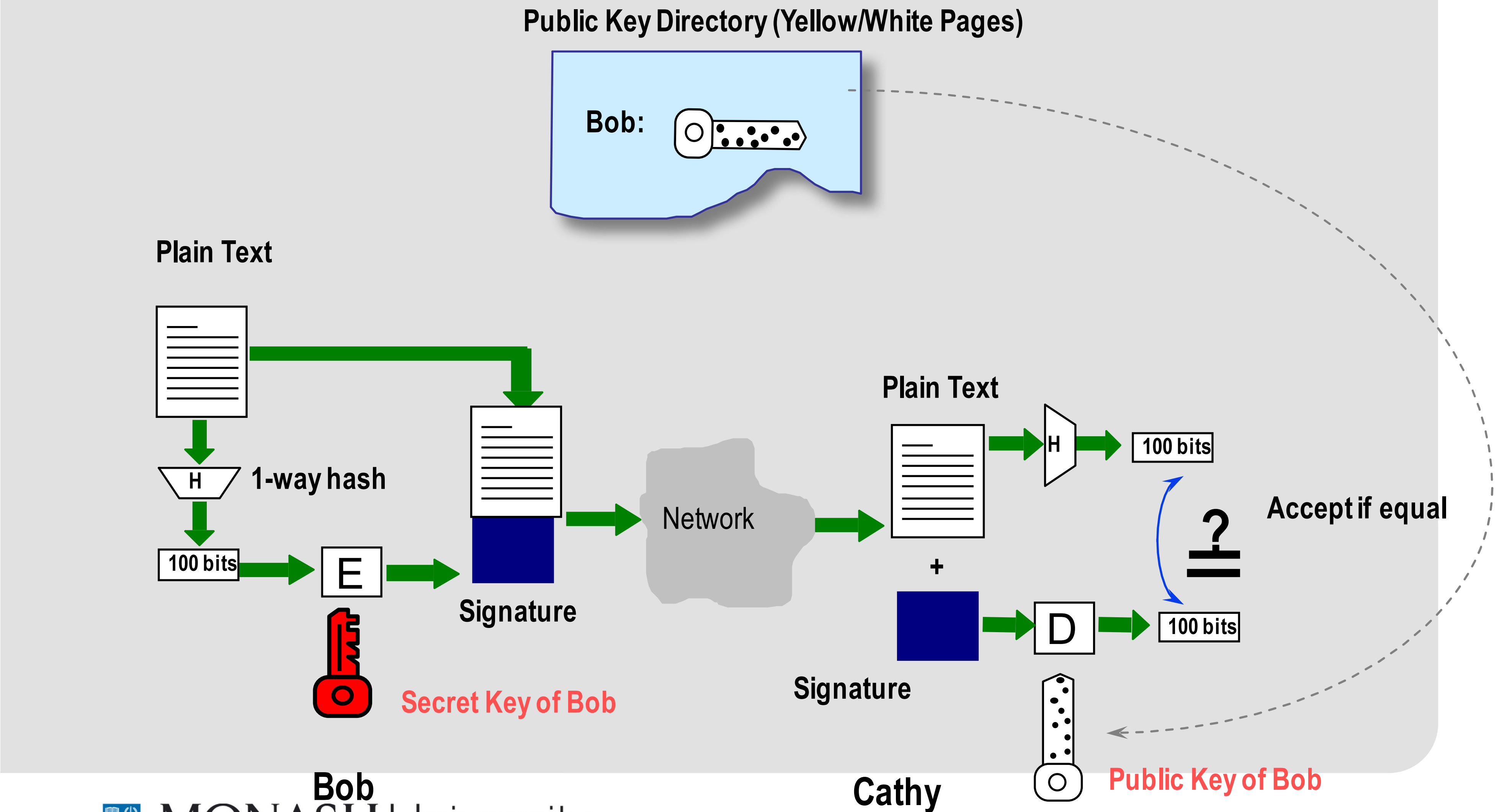
Digital Signature --- an example(3)

- **Cathy, a verifier:**
  - receives a pair $(m,s)=(19, 24)$
  - looks up the phone book and finds out Bob's public key $(e, n)=(3, 55)$
  - calculates $t = s^e \ (mod\ n)$
    $= 24^3 \ (mod\ 55)\ ) = 26X24\ mod55$
    $= 19$
  - checks whether $t=m$
  - confirms that ($19,24$) is a genuinely signed document of Bob if $t=m$.

Signing Long Documents!

- In the previous example, a document has to be an integer in [0,...,(n-1)]
- To sign a very long document, we need a so called one-way hash algorithm
- Instead of signing directly on a doc, we hash the doc first, and sign the hashed data which is normally short.

Digital Signature (for long doc)

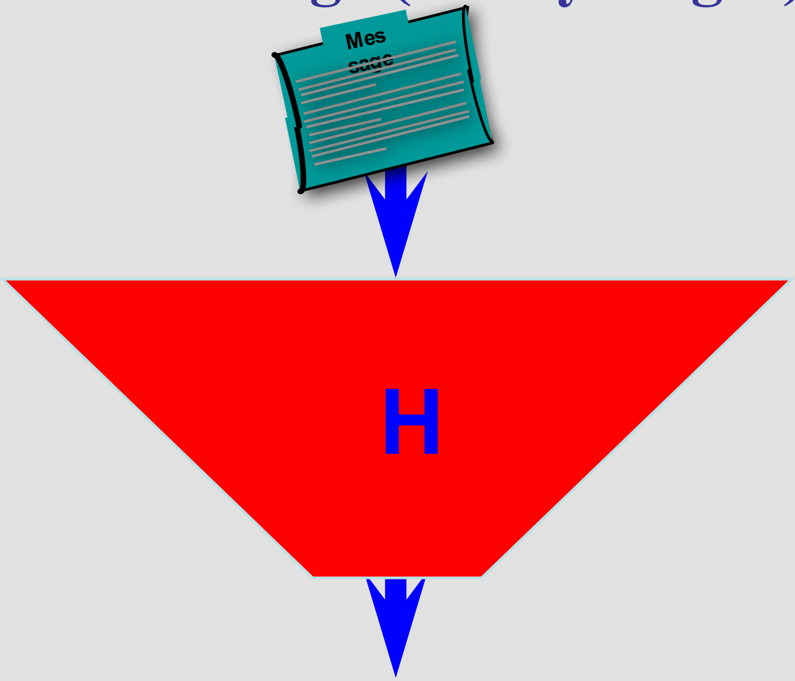


One-Way Hash Algorithm

- **A one-way hash algorithm hashes an input document into a condensed short output (say of 100 bits)**
  - Denoting a one-way hash algorithm by H(.), we have:
    - > Input: m - a binary string of any length
    - > Output: H(m) - a binary string of L bits, called the “hash of m under H”.
    - > The output length parameter L is fixed for a given one-way hash function H,
    - > eg
      - The one-way hash function “MD5” has L = 128 bits
      - The one-way hash function “SHA-1” has L = 160 bits

One-Way Hash Algorithm

Message (of any length)



Hash of the message

A condensed short output, say of 100 bits

Properties of One-Way Hash Algorithm

- **A good one-way hash algorithm H needs to have these properties:**

1. For any size of data

Can be applied to a block of data of any size

2. Output size:

produces a fixed-length output

Properties of One-Way Hash Algorithm

3. Easy to Evaluate

The hashing algorithm should be fast

i.e. given any document m, the hashed value h = H(m) can be computed quickly.

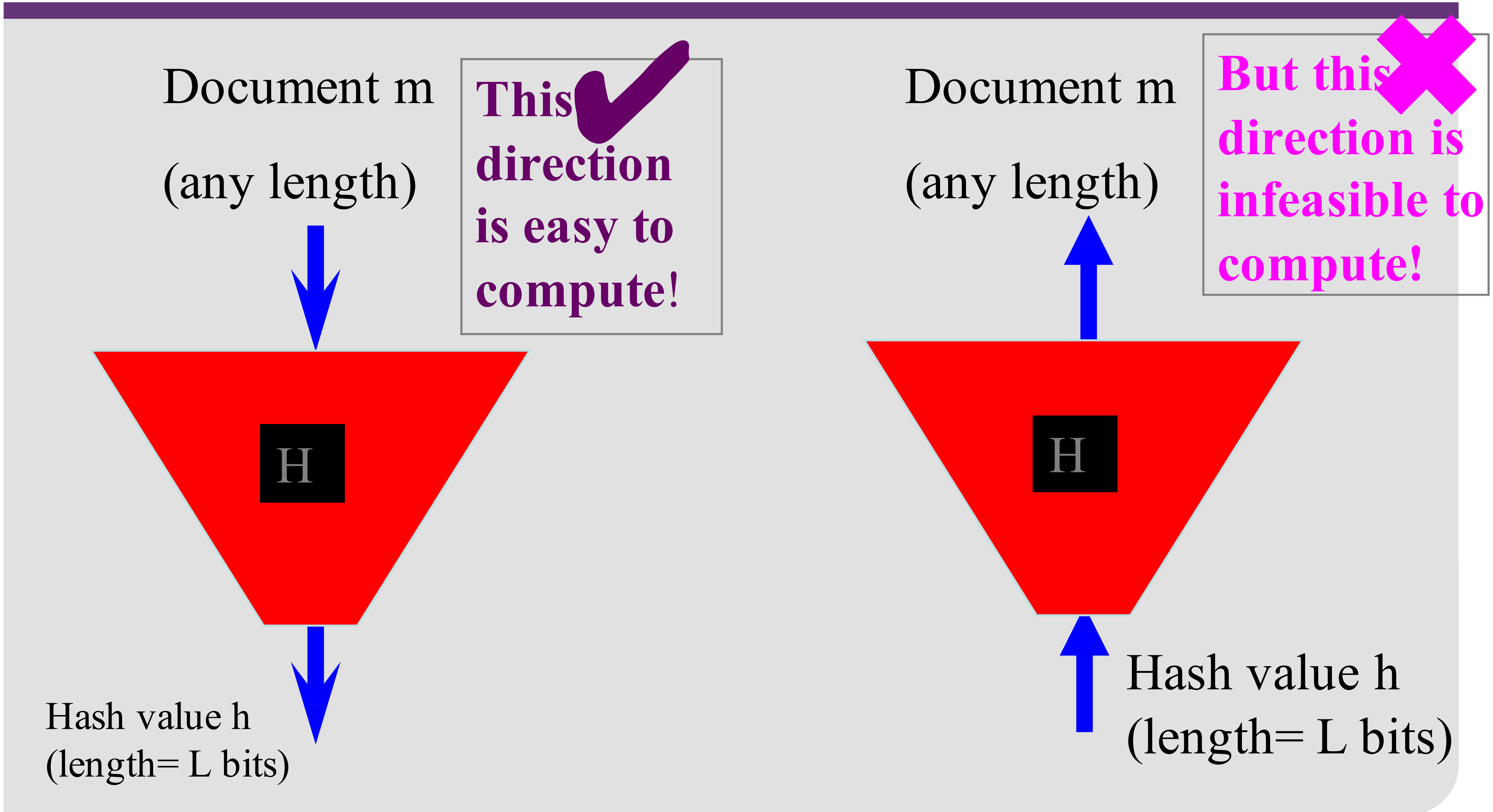
4. Hard to Reverse

There is no feasible algorithm to "reverse" a hashed value,

i.e. given any hashed value h, it is computationally infeasible to find any document m such that H(m) = h.

- **NOTE: An algorithm is called 'One-Way' if it has BOTH properties 3 and 4.**

The One-way Property



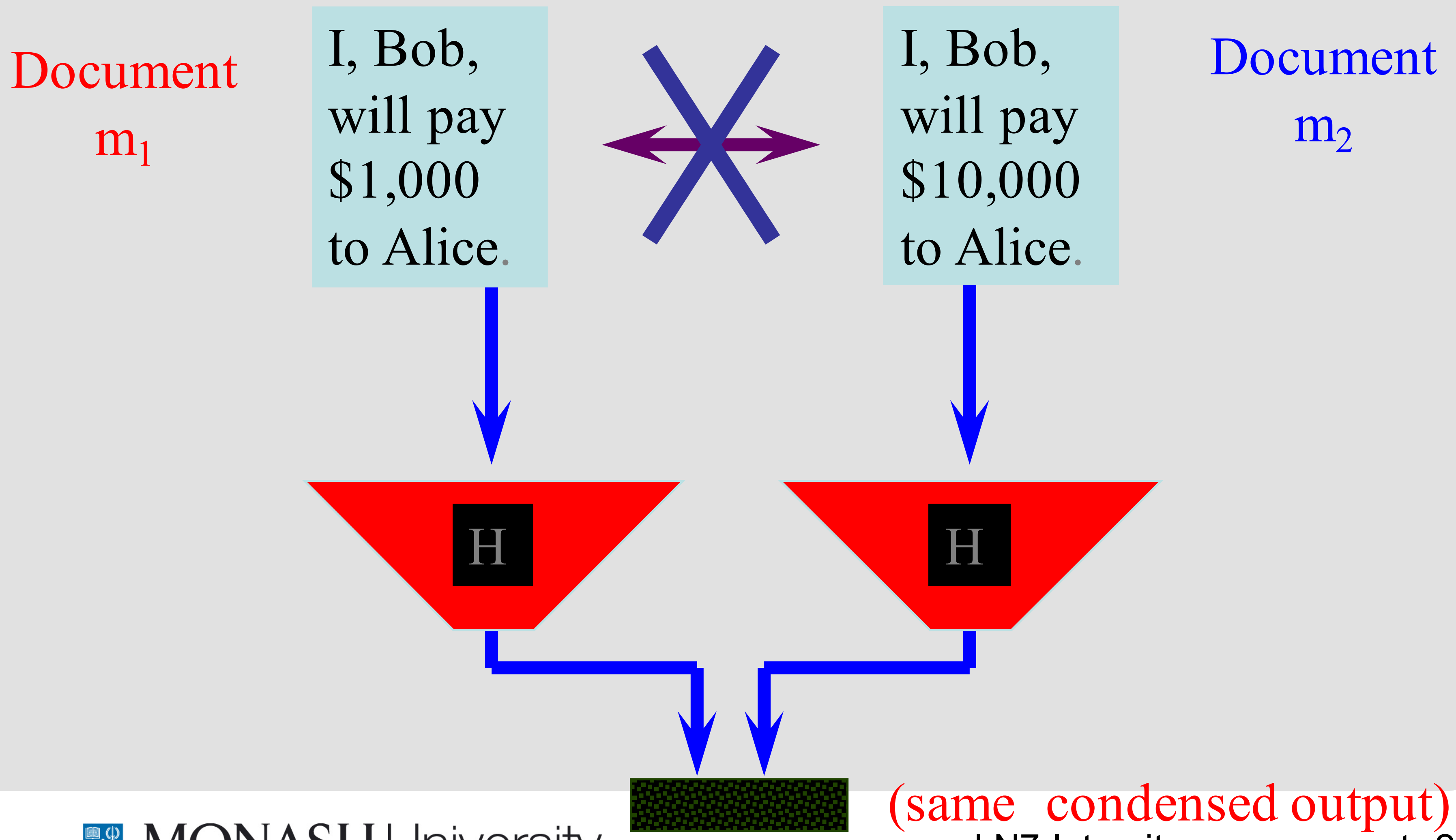
Properties of One-Way Hash Algorithm

5. Hard to find Collisions

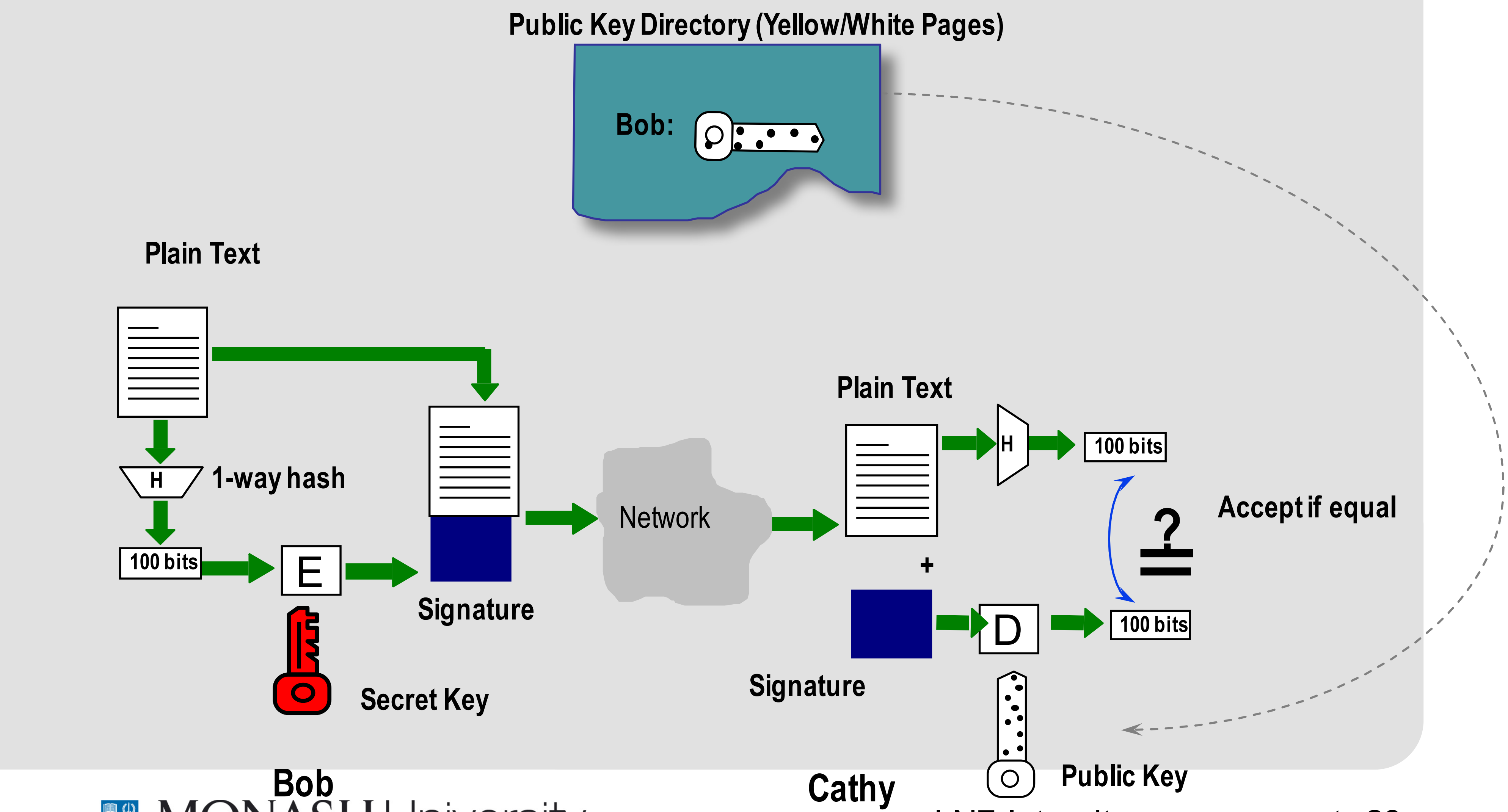
> There is no feasible algorithm to find two or more input documents which are hashed into the same condensed output,

> i.e., it is computationally infeasible to find any two documents m_1, m_2 such that $H(m_1) = H(m_2)$.

Finding Collision is Infeasible



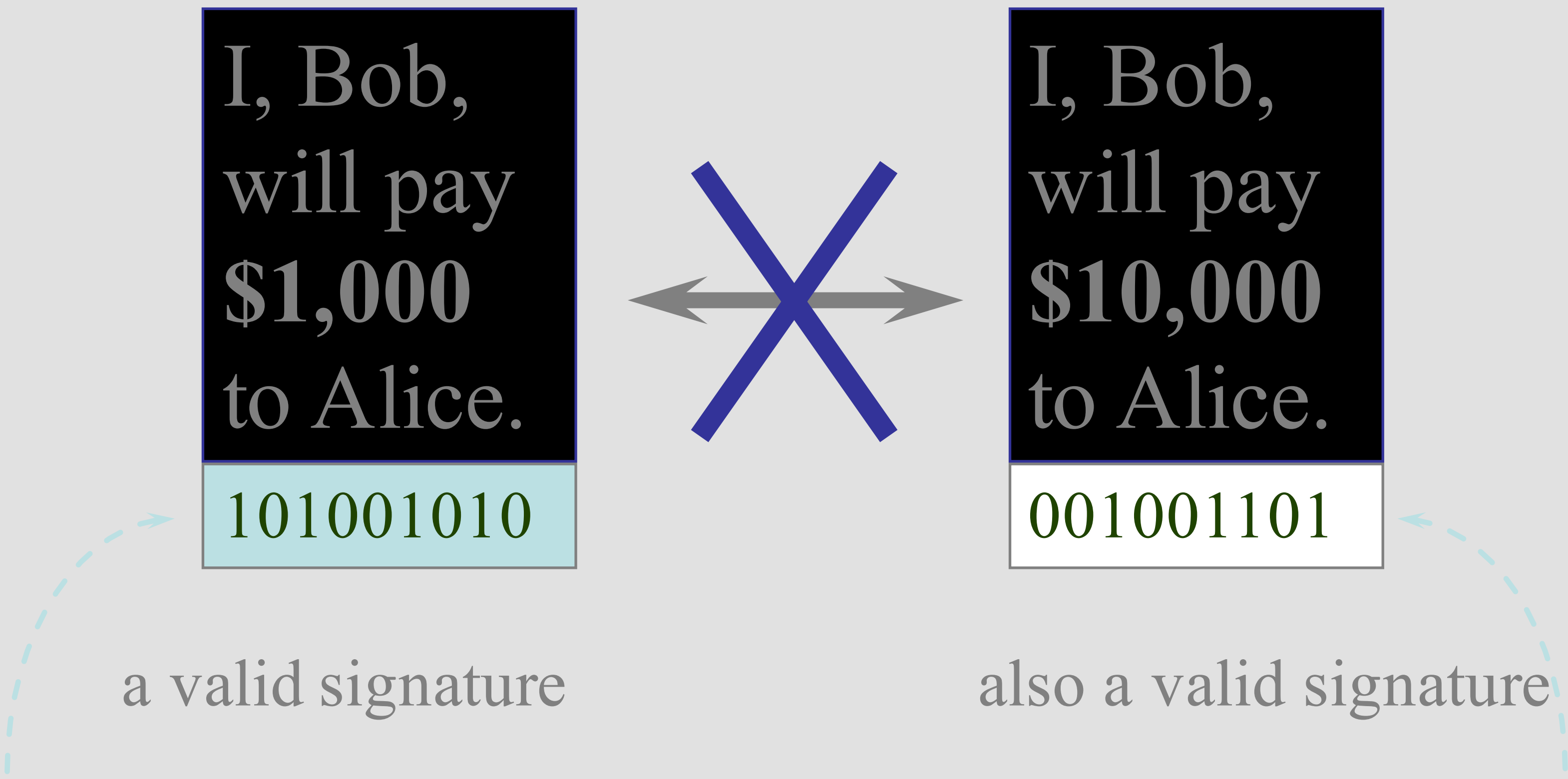
Digital Signature (for long doc)



Why Digital Signature ?

- **Unforgeable**
  - takes long, long time to forge !
- **Un-deniable by the signatory**
  - Because secret key of the signatory was used
- **Universally verifiable**
  - Signature is verified using the public key of the signatory which should be available to everyone!
- **Differs from doc to doc**
- **Easily implementable by**
  - Software, hardware or software + hardware

Unforgeable Digital Signature



Digital Signature Properties

- **A digital signature is analogous to the handwritten signature**
  - Provides a set of security capabilities that would be difficult to implement in any other way.
- **Properties**
  - It must verify the author and the date and time of the signature
  - It must authenticate the contents at the time of the signature
  - It must be verifiable by third parties, to resolve disputes
- **the digital signature function includes the authentication function.**

Digital Signature Requirements

- **Must depend on the message signed**
- **Must use information unique to the originator (sender)**
  - to prevent both forgery and denial
- **Must be relatively easy to produce**
- **Must be relatively easy to recognize & verify**
- **Be computationally infeasible to forge**
  - with new message for existing digital signature
  - with fraudulent digital signature for given message
- **Be practical to save digital signature in storage**

Digital Signature -- Summary

- **Three (3) steps are involved in digital signature**
  - Setting up public and secret keys
  - Signing a document
  - Verifying a signature

Setting up Public & Secret Keys

- **Bob does the following**
  - prepares a pair of public and secret keys
  - publishes his public key in the public key file (such as an on-line phone book)
  - keeps the secret key to himself
- **Note:**
  - Setting up needs only to be done once !

Signing a Document

- **Once setting up is completed, Bob can sign a document (such as a contract, a cheque, a certificate, ...) using the secret key**
- **The pair of document & signature is a proof that Bob has signed the document.**

Verifying a Signature

- **Any party, say Cathy, can verify the pair of document and signature, by using Bob's public key from the public key file.**
- **Important !**
  – Cathy does NOT need to have her own public or secret key !

MONASH University
Information Technology

Integrity → Verification of modification

Message (file contents) Authentication

- **Message authentication is generally concerned with:**
  - protecting the integrity of a message
  - validating identity of originator
    - > non-repudiation of the origin (dispute resolution)
- **Possible methods that can be used to produce an authenticator**
  - message encryption
  - message authentication code (MAC)
  - hash function

Security Requirements

- **In the context of communications across a network, the attacks can be:**
  - Disclosure
  - traffic analysis
  - masquerade
  - content modification
  - sequence modification
  - timing modification
  - source repudiation
  - destination repudiation

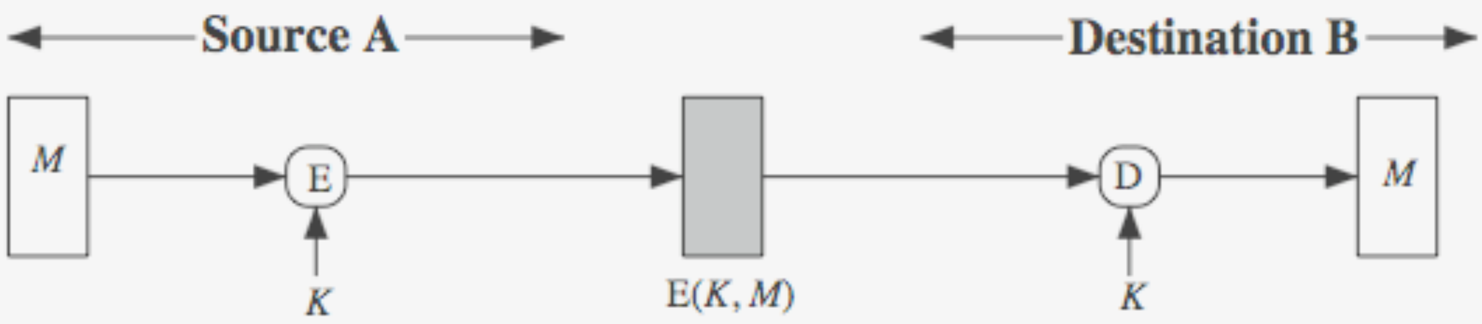
Message confidentiality

Message authentication

Message Authentication using Encryption (1)

- **message encryption by itself also provides a measure of authentication**
- **if symmetric encryption is used then:**
  - receiver knows sender must have created it
  - since only sender and receiver know key used
  - know content cannot be altered
  - if message has suitable structure, redundancy or a checksum to detect any changes

Symmetric Message Encryption

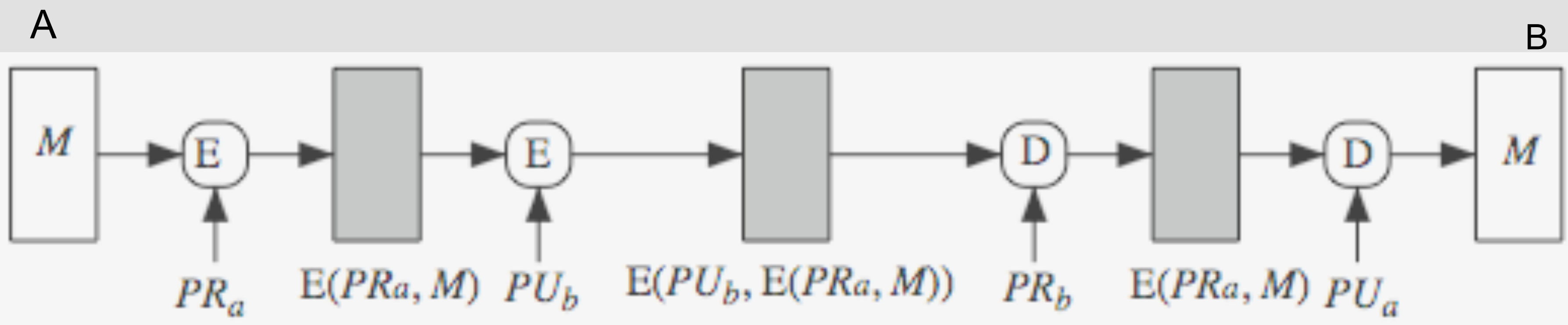


(a) Symmetric encryption: confidentiality and authentication

Message Authentication using Encryption (2)

- **if public-key encryption is used:**
  - encryption provides no confidence of sender
  - since anyone potentially knows public-key
  - however if
    - senders **sign** the message using their private-key
    - then encrypt with recipients public key
    - have both secrecy and authentication
  - again need to recognize corrupted messages
  - but at the cost of two public-key uses on the message

Public-Key Message Encryption

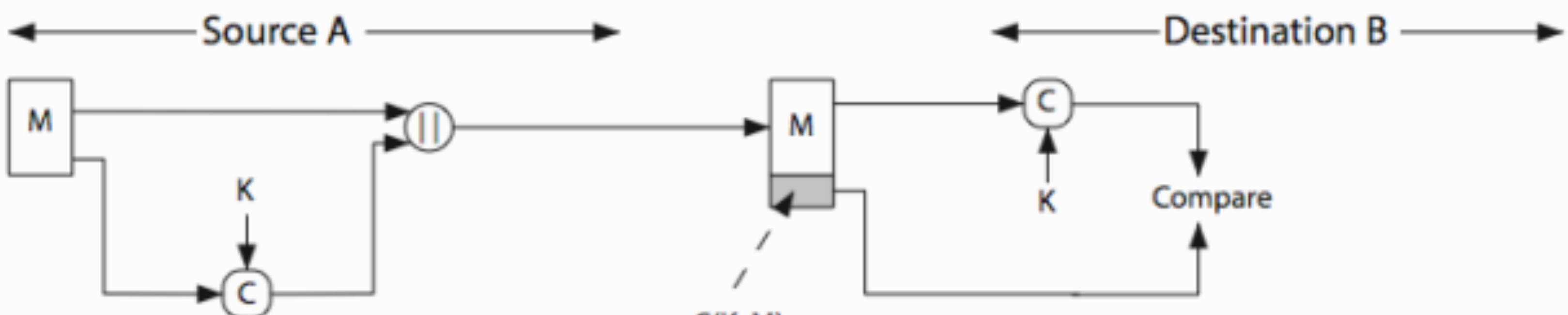


(d) Public-key encryption: confidentiality, authentication, and signature

Message Authentication Code (MAC)

- **Generated by an algorithm that creates a small fixed-sized block → like check sum in data transmission**
  - depending on both message and some key
  - like encryption though need not be reversible
- **Appended to message as a signature**
- **Receiver performs same computation on message and checks if it matches with the MAC**
- **Provides assurance that message is unaltered and comes from sender**

Message Authentication using a Message Authentication Code (MAC)



(a) Message authentication

Message Authentication Codes

- **MAC is an approach to message authentication**
- **Can also use encryption for secrecy**
  - generally use separate keys for each
  - can compute MAC either before or after encryption
  - is generally regarded as better done before
- **Why use a MAC?**
  - sometimes only authentication is needed
  - sometimes need authentication to persist longer than the encryption (eg. archival use)
- **note that a MAC does not satisfy non-repudiation**

MAC Properties

- **MAC can be viewed as a cryptographic checksum**
  - $\texttt{MAC} = \texttt{C}_{\texttt{K}}\texttt{(M)}$
  - condenses a variable-length message M
  - using a secret key K
  - to a fixed-sized authenticator
- **is a many-to-one function**
  - potentially many messages can generate the same MAC
  - but finding those messages should be difficult

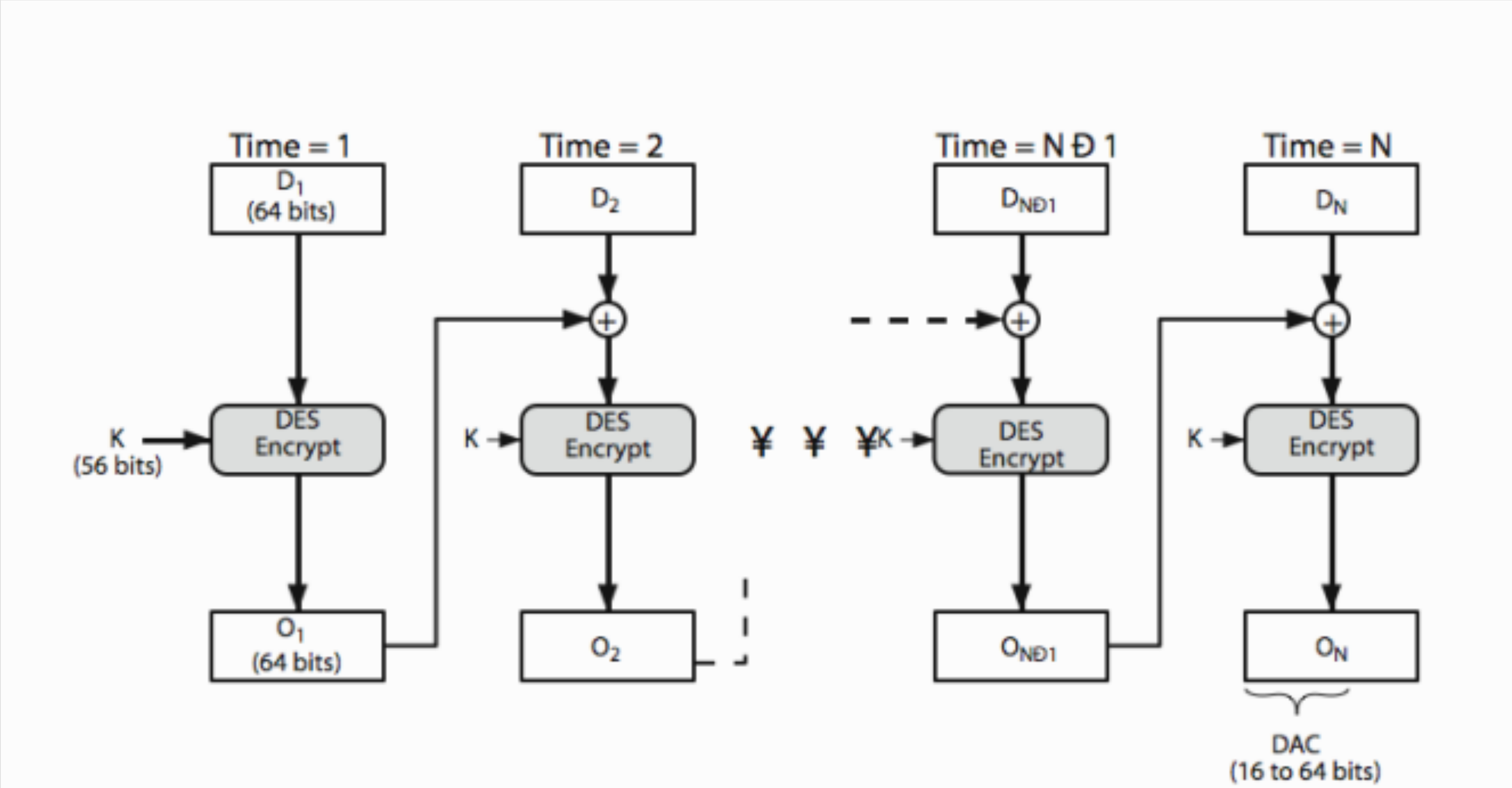
Requirements for MACs

- **Need the MAC to satisfy the following:**
  1. knowing a message and MAC, is infeasible to find another message with same MAC
     - deals with message replacement attacks
  2. MACs should be uniformly distributed across the messages
     - deals with the need to thwart a brute-force attack based on chosen plaintext
  3. MAC should depend equally on all bits of the message
     - dictates that the authentication algorithm should not be weaker with respect to certain parts or bits of the message than others.

Message Authentication Using Symmetric Ciphers for MACs

- **Can use any block cipher chaining mode and use the final block as a MAC**
- **Data Authentication Algorithm (DAA) is a widely used MAC based on DES-CBC**
  - using IV=0 and zero-pad of final block
  - encrypt message using DES in CBC mode
  - and send just the final block as the MAC
    - or the leftmost M bits ($16 \le M \le 64$) of final block
- **But final MAC now may be too small for security**

Data Authentication Algorithm



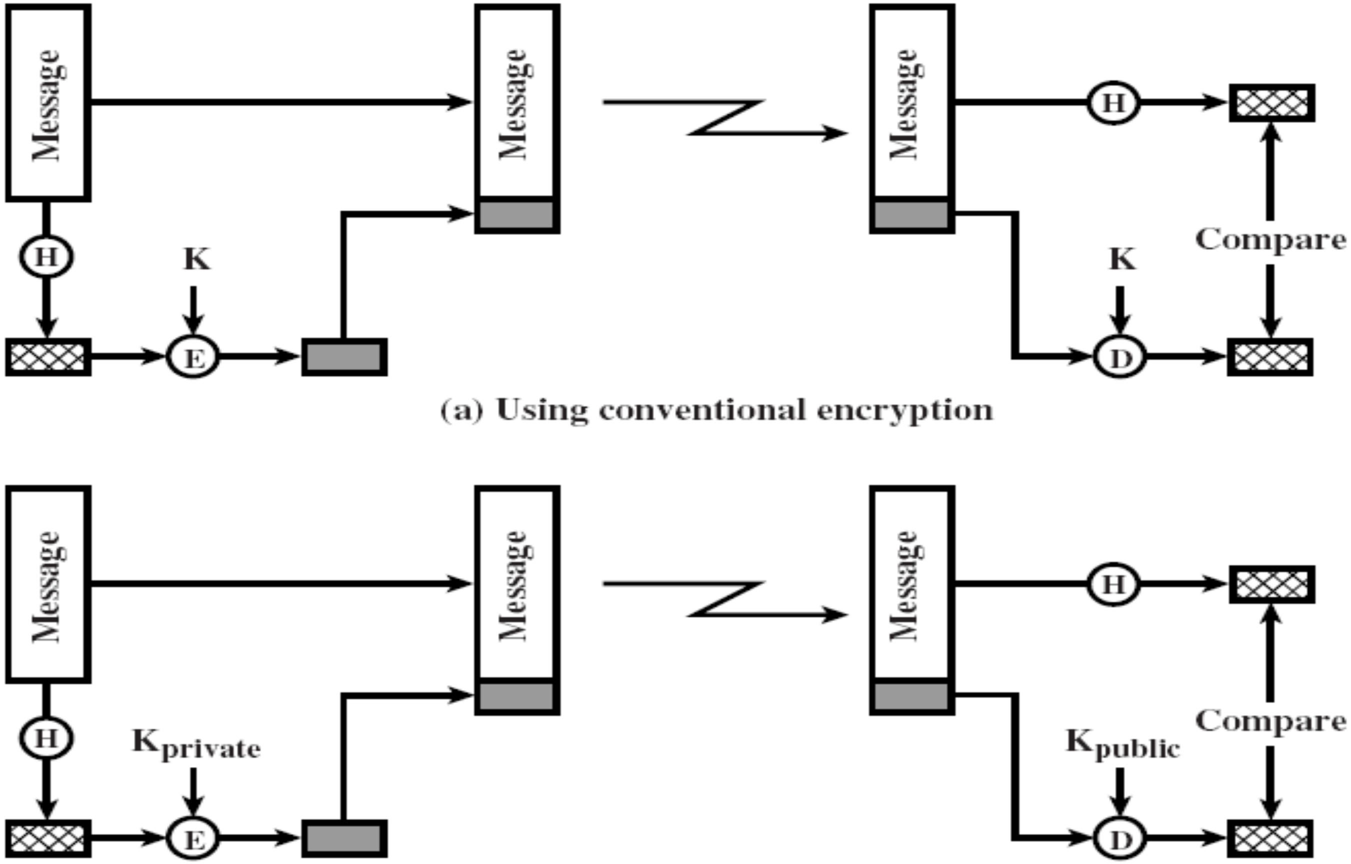
Review :

- **Digital Signatures**
  - Unforgeable, non-repudiation, universally verifiable, message dependent, easily implementable
  - Short documents
  - Long documents
    - > Properties of Hash functions
      - Any size input, fixed size o/p, one-way, strong collision resistance
- **Message Authentication**
  - Integrity, validate identity of originator
  - Encryption, MAC, Hash function
  - Requirements and properties of MAC
    - > Data Authentication algorithm with DES-CBC, IV=0

Hash Function

- **Like MAC, hash function is a one way function that accepts a message M and produces a fixed-size message digest H(M)**
- **Unlike MAC, hash function takes only the message (not the key) as input to generate the message digest**
- **Hash is used to detect changes to message**
- **Can be used with both symmetric and asymmetric encryption**
  - used in various ways with message
  - Most often to create a digital signature

Message Authentication using Hash Function (1)

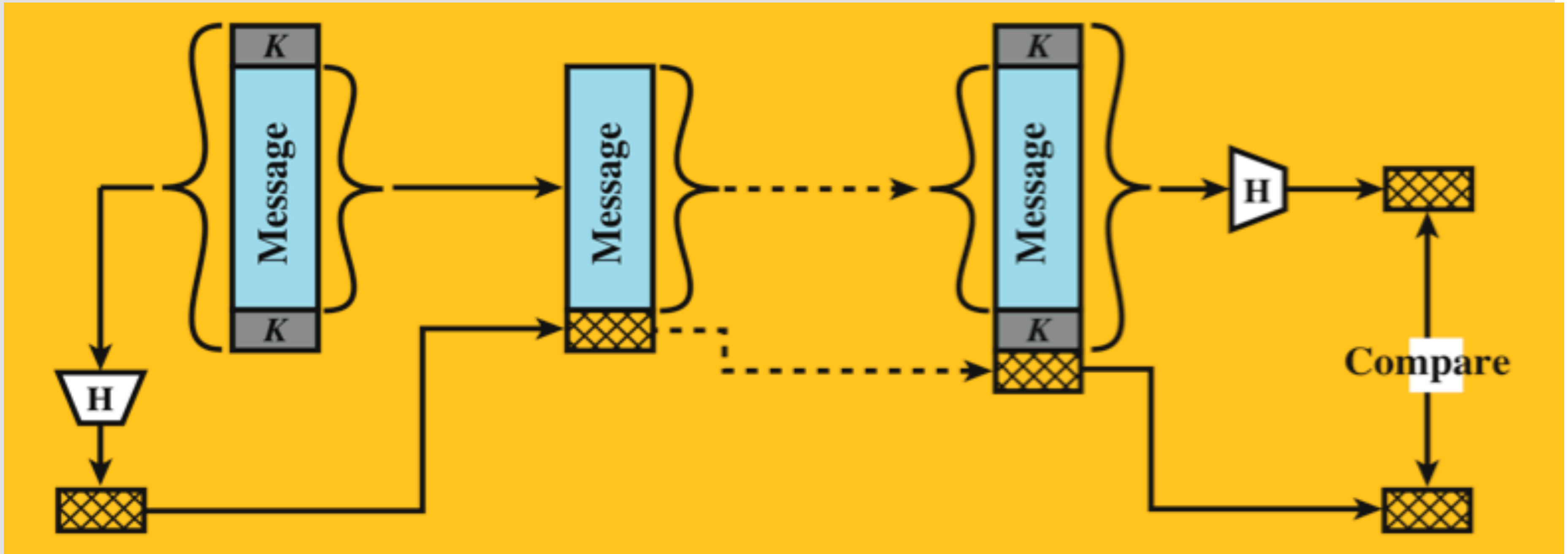


(a) Using conventional encryption

(b) Using public-key encryption

Message Authentication using Hash Function (2)

- Secret value K is known to A & B
- A calculates hash function of secret value + Message, i.e., MD=H(K‖M‖K)
- M‖MD is sent
- B recalculates H(K‖M‖K) and verifies



(c) Using secret value

Simple Hash Functions

- **a one-way or secure hash function used in message authentication, digital signatures**
- **all hash functions process an input block at a time in an iterative fashion**
- **one of simplest n-bit hash functions is the bit-by-bit exclusive-OR (XOR) of each block**

$$C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$$

  - C_i is i th bit of the hash code
  - m is the number of n-bit blocks in the input
  - b_{ij} is ith bit in the jth block
  - effective data integrity check on random data
  - less effective on more predictable data
  - virtually useless for data security

Hash Functions: Attacks

- **two attack approaches**
  - cryptanalysis
    - > exploit logical weakness in alg
  - brute-force attack
    - > trial many inputs
    - > strength proportional to size of hash code ($2^{n/2}$)
- **Secure Hash Algorithm (SHA) most widely used hash algorithm**
  - SHA-1 gives 160-bit hash
  - more recent SHA-256, SHA-384, SHA-512 provide improved size and security

Keyed Hash Functions as MACs

- **want a MAC based on a hash function**
  – because hash functions are generally faster
  – crypto hash function code is widely available
- **hash includes a key along with message**
- **original proposal:**
  – KeyedHash = Hash(Key||Message)
  – some weaknesses were found with this
- **eventually led to development of HMAC**

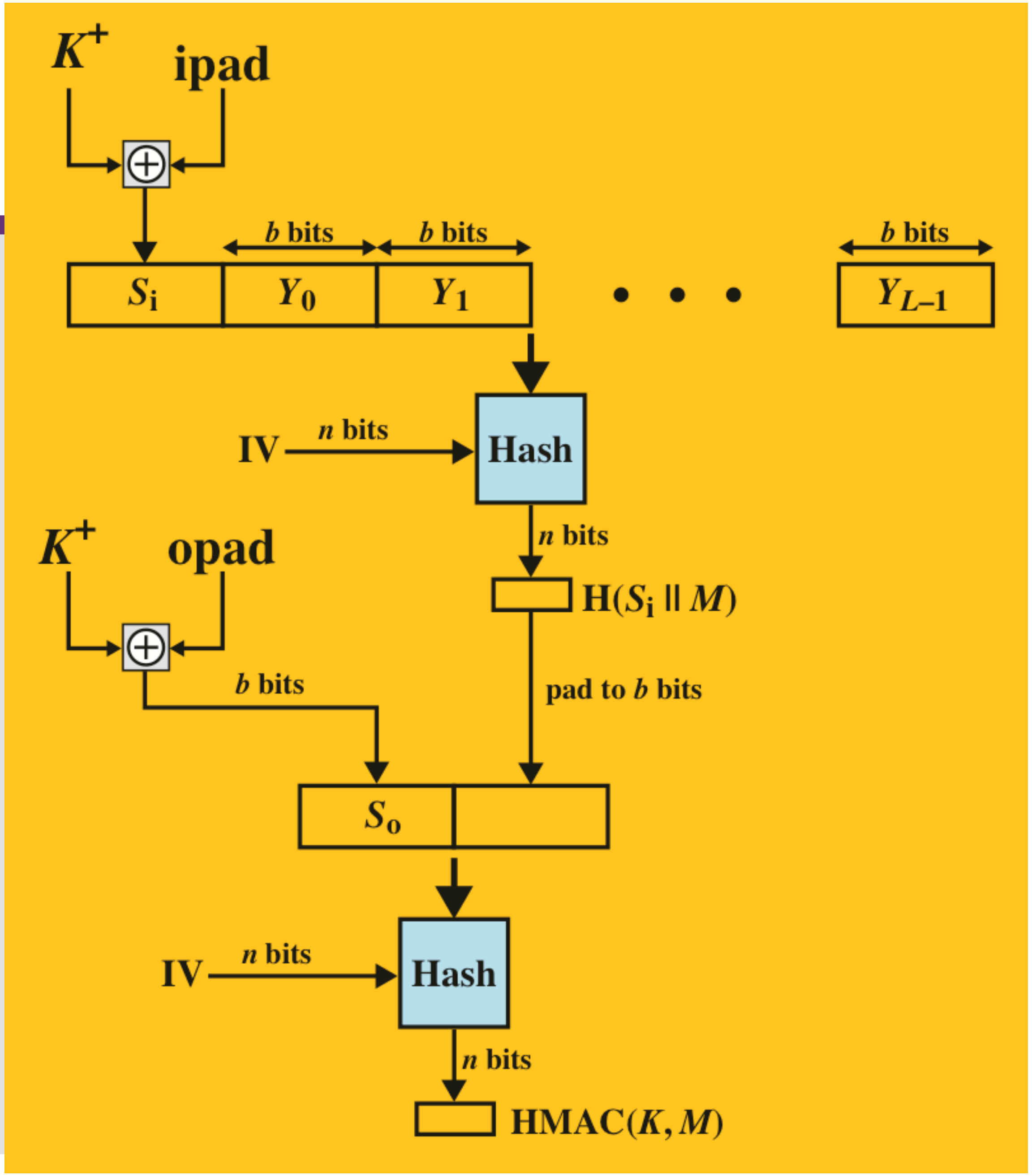
HMAC Design Objectives

- **use, without modifications, hash functions**
- **allow for easy replaceability of embedded hash function**
- **preserve original performance of hash function without significant degradation**
- **use and handle keys in a simple way.**
- **have well understood cryptographic analysis of authentication mechanism strength**

HMAC

- **specified as Internet standard RFC2104**
- **uses hash function on the message:**
  - HMAC(K,M)= Hash[(K+ XOR opad) ||
  - Hash[(K+ XOR ipad) || M)]]
  - where K+ is the key padded out to size
  - opad, ipad are specified padding constants
- **overhead is just hash calculations on 3 more blocks than hashing the message alone**
- **any hash function can be used**
  - eg. MD5, SHA-1, RIPEMD-160, Whirlpool

• HMAC Overview



HMAC Security

- **proved security of HMAC relates to that of the underlying hash algorithm**
- **attacking HMAC requires either:**
  - brute force attack on key used (2^n)
  - birthday attack (but since keyed would need to observe a very large number of messages)
    - Collision resistant attacks, an adversary wishes to find 2 messages that yield the same hash ($2^{n/2}$)
- **choose hash function used based on speed verses security constraints**

Summary

- **Non-repudiation**
- **Digital Signatures**
  - > Properties
  - > Short document
  - > Long document
- **Message Authentication**
  - – Message encryption
  - – Message Authentication code
    - > Properties & DAA
  - – Hash Algorithms
    - > Properties
    - > Simple Hash Function & HMAC

Further Reading

- **Chapters 2 & 21 of the textbook: *Computer Security: Principles and Practice*" *by William Stallings & Lawrie Brown,* Prentice Hall, 2015**

- **Acknowledgement: part of the materials presented in the slides was developed with the help of Instructor's Manual and other resources made available by the author of the textbook.**