CSS322

Introduction

Functions

Auth. with Encryption

Auth. with MAC

Security

Algorithms

Message Authentication Codes

CSS322: Security and Cryptography

Sirindhorn International Institute of Technology
Thammasat University

Prepared by Steven Gordon on 17 January 2011
CSS322Y10S2L10, Steve/Courses/CSS322/Lectures/mac.tex, r1630

Contents

Message Authentication Requirements and Functions

Authentication with Message Encryption

Authentication with Message Authentication Codes

Security of MACs

MAC Algorithms

Attacks on Communications across Network

1. Disclosure: encryption
2. Traffic analysis: encryption
3. Masquerade: message authentication
4. Content modification: message authentication
5. Sequence modification: message authentication
6. Timing modification: message authentication
7. Source repudiation: digital signatures
8. Destination repudiation: digital signatures

Message Authentication Functions

- Message authentication (and digital signature) mechanisms have two parts:
  1. Function that produces authenticator
  2. Protocol that enables receiver to verify authenticity
- Three types of authentication functions:
  1. Hash function
  2. Message encryption
  3. Message authentication code (MAC)

Contents

Message Authentication Requirements and Functions

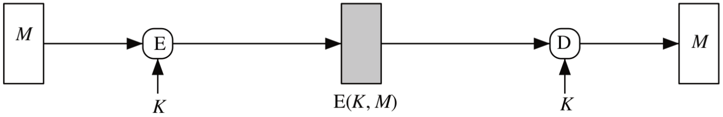
Authentication with Message Encryption

Authentication with Message Authentication Codes

Security of MACs

MAC Algorithms

Symmetric Encryption for Authentication



- Confidentiality: only B (and A) can recover plaintext
- Source Authentication: A is only other user with key; must have come from A
- Data Authentication: successfully decrypted; data has not been modified
- Assumption: decryptor can recognise correct plaintext

Recognising Correct Plaintext

Example 1

B receives ciphertext (supposedly from A, using shared secret key K):

```
DPNFCTEJLYONCJAEZRCLASJTDQFY
```

B decrypts with key K to obtain plaintext:

```
SECURITYANDCRYPTOGRAPHYISFUN
```

- Was the plaintext encrypted with key K (and hence sent by A)?
- Is the ciphertext received the same as the ciphertext sent by A?

Recognising Correct Plaintext

Example 2

B receives ciphertext (supposedly from A, using shared secret key K):

```
QEFPFPQEBTOLKDJBPPXDBPLOOVX
```

B decrypts with key K to obtain plaintext:

```
FTUEUEFTQIDAZSYQEEMSQEADDKM
```

- Was the plaintext encrypted with key K (and hence sent by A)?
- Is the ciphertext received the same as the ciphertext sent by A?

Recognising Correct Plaintext

Example 3

B receives ciphertext (supposedly from A, using shared secret key K):

```
01101001101011010101101110000010
```

B decrypts with key K to obtain plaintext:

```
01011101000011010010101001011 10
```

- Was the plaintext encrypted with key K (and hence sent by A)?
- Is the ciphertext received the same as the ciphertext sent by A?

Recognising Correct Plaintext

Example 1

- Assume the message is English
- Plaintext had expected structure; assume the plaintext is correct
  - Sent by A and has not been modified

Example 2

- Assume the message is English
- Plaintext had no structure in expected language; assume plaintext is incorrect
  - Either not sent by A or modified

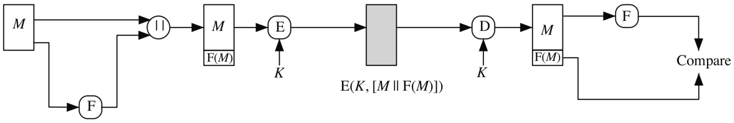
Example 3

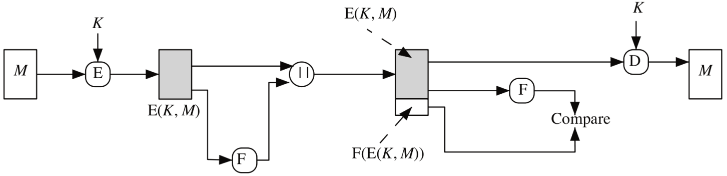
- Binary data, e.g. image, compressed file
- Cannot know whether correct or incorrect

Recognising Correct Plaintext

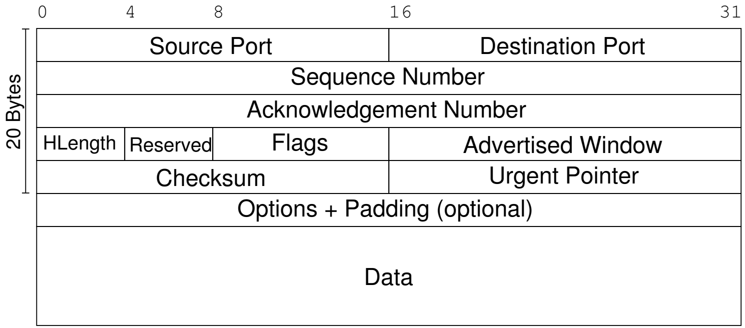
- Valid plaintexts should be small subset of all possible messages
  - E.g. 26^n possible messages of length n; only small subset are valid English phrases
- Plaintext messages have structure
- BUT automatically detecting structure can be difficult
- Add structure to make it easier, e.g.
  - Error detecting code or Frame Check Sequence
  - Packet header

Adding Error Detecting Code



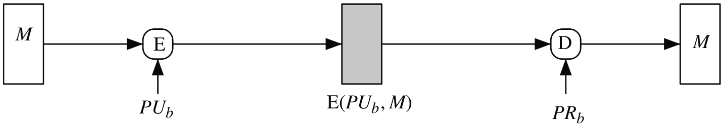


TCP Segment


0
4
8
16
31
20 Bytes
Source Port
Destination Port
Sequence Number
Acknowledgement Number
HLength
Reserved
Flags
Advertised Window
Checksum
Urgent Pointer
Options + Padding (optional)
Data

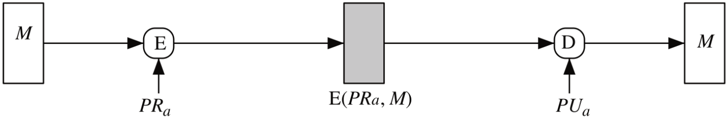
Public-Key Encryption for Authentication

- Only provides confidentiality
- Same assumption as before: plaintext must have structure so can be recognised as correct or incorrect



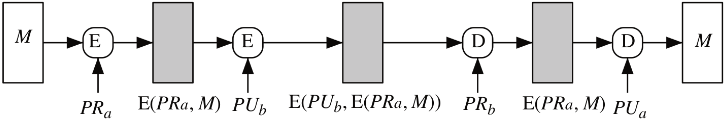
Public-Key Encryption for Authentication

- Data authentication (data has not been modified)
- Digital signature: proof that message came from A



Public-Key Encryption for Authentication

- Both confidentiality, authentication and digital signature



Contents

Message Authentication Requirements and Functions

Authentication with Message Encryption

Authentication with Message Authentication Codes

Security of MACs

MAC Algorithms

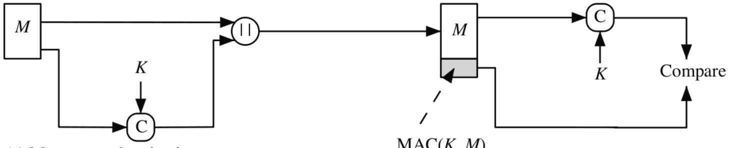
Authentication with Message Authentication Codes

- Append small, fixed-size block of data to message: cryptographic checksum or MAC

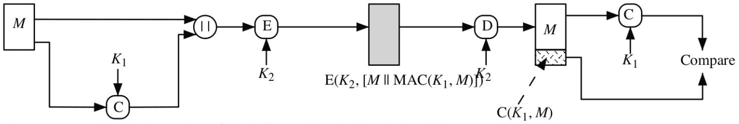
$$\mathrm{T} = \mathrm{MAC}(K, M)$$

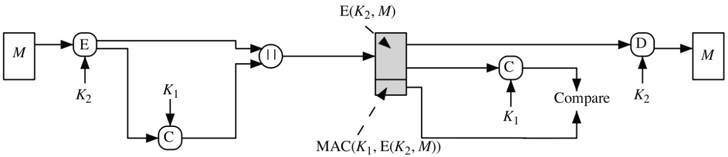
M = input message
MAC = MAC function
K = shared secret key of k bits
T = message authentication code (or tag) of n bits

- MAC function also called *keyed hash function*
- MAC function similar to encryption, but does not need to be reversible
    - Easier to design stronger MAC functions than encryption functions



(a) Message authentication





Contents

Message Authentication Requirements and Functions

Authentication with Message Encryption

Authentication with Message Authentication Codes

Security of MACs

MAC Algorithms

Requirement of MACs

Objective of Attacker

- Assume MAC function is known, key K is not
- For valid MAC code for given message x

Requirement of MAC Function

Computation Resistance : given one or more text-MAC pairs $[x_i, MAC(K, x_i)]$, computationally infeasible to computer any text-MAC pair $[x, MAC(K, x)]$ for new input $x \neq x_i$

Security of MACs

Brute Force Attack on Key

- Attacker knows $[x_1, T_1]$ where $T_1 = MAC(K, x_1)$
- Key size of k bits: brute force on key, 2^k
- But …many tags match T_1
- For keys that produce tag T_1, try again with $[x_2, T_2]$
- Effort to find K is approximately 2^k

Brute Force Attack on MAC value

- For x_m, find T_m without knowing K
- Similar effort required as one-way/weak collision resistant property for hash functions
- For n bit MAC value length, effort is 2^n

Effort to break MAC: $\min(2^k, 2^n)$

Security of MACs

Cryptanalysis

- Many different MAC algorithms; attacks specific to algorithms
- MAC algorithms generally considered secure

Contents

Message Authentication Requirements and Functions

Authentication with Message Encryption

Authentication with Message Authentication Codes

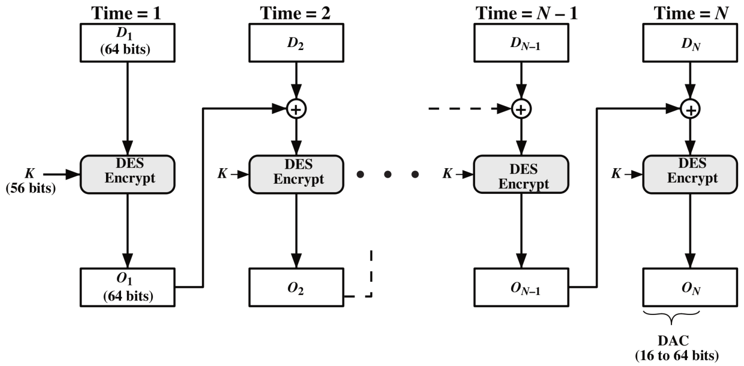
Security of MACs

MAC Algorithms

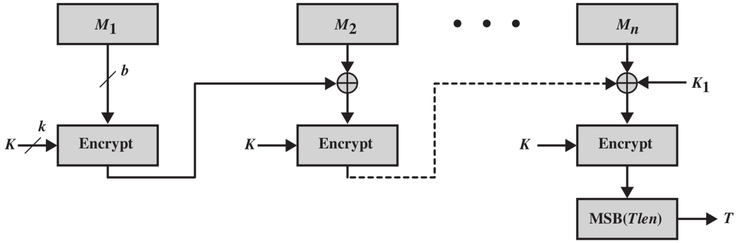
MACs Based on Block Ciphers

- Data Authentication Algorithm (DAA): based on DES; considered insecure
- Cipher-Based Message Authentication Code (CMAC): mode of operation used with Triple-DES and AES
- OMAC, PMAC, UMAC, VMAC, . . .

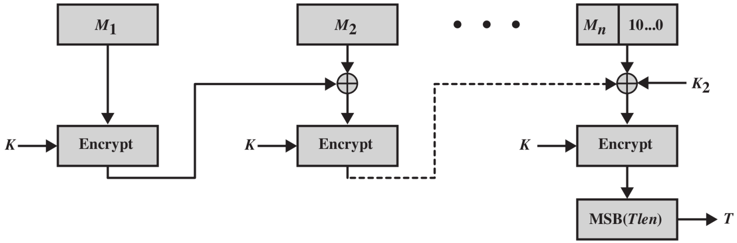
DAA


Time = 1
D_1 (64 bits)
K (56 bits)
DES Encrypt
O_1 (64 bits)
Time = 2
D_2
K
DES Encrypt
O_2
Time = $N-1$
D_{N-1}
K
DES Encrypt
O_{N-1}
Time = N
D_N
K
DES Encrypt
O_N
DAC (16 to 64 bits)

CMAC



(a) Message length is integer multiple of block size



(b) Message length is not integer multiple of block size

HMAC

- MAC function derived from cryptographic hash functions
  - MD5/SHA are fast in software (compared to block ciphers)
  - Libraries for hash functions widely available

$$\mathrm{HMAC}(K, M) = \mathrm{H}((K \oplus \mathrm{opad})||\mathrm{H}((K \oplus ipad)||M))$$

where ipad= 00110110 repeated, opad= 01011100 repeated

- Security of HMAC depends on security of hash function used