Message Authentication and Hash Functions

- At cats' green on the Sunday he took the message from the inside of the pillar and added Peter Moran's name to the two names already printed there in the "Brontosaur" code. The message now read: “Leviathan to Dragon: Martin Hillman, Trevor Allan, Peter Moran: observe and tail.” What was the good of it John hardly knew. He felt better, he felt that at last he had made an attack on Peter Moran instead of waiting passively and effecting no retaliation. Besides, what was the use of being in possession of the key to the codes if he never took advantage of it?

- —Talking to Strange Men, Ruth Rendell
Message Authentication

- message authentication is concerned with:
  - protecting the integrity of a message
  - validating identity of originator
  - non-repudiation of origin (dispute resolution)

- will consider the security requirements

- then three alternative functions used:
  - message encryption
  - message authentication code (MAC)
  - hash function
Security Requirements

- disclosure
- traffic analysis
- masquerade
- content modification
- sequence modification
- timing modification
- source repudiation
- destination repudiation
Message Encryption

- 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 now key used.
  - Know content cannot of been altered.
  - If message has suitable structure, redundancy or a checksum to detect any changes.
Message Encryption

- if public-key encryption is used:
  - encryption provides no confidence of sender
  - since anyone potentially knows public-key
  - however if
    - sender **signs** message using their private-key
    - then encrypts with recipients public key
    - have both secrecy and authentication
  - again need to recognize corrupted messages
  - but at cost of two public-key uses on message
Message Authentication Code (MAC)

- generated by an algorithm that creates a small fixed-sized block
  - 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 it matches the MAC
- provides assurance that message is unaltered and comes from sender
Message Authentication Code
Message Authentication Codes

- as shown the MAC provides 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 (e.g. archival use)
- note that a MAC is not a digital signature
MAC Properties

- a MAC is a cryptographic checksum
  \[ \text{MAC} = C_K(M) \]
  - 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 have same MAC
  - but finding these needs to be very difficult
Requirements for MACs

- taking into account the types of attacks
- need the MAC to satisfy the following:
  1. knowing a message and MAC, is infeasible to find another message with same MAC
  2. MACs should be uniformly distributed
  3. MAC should depend equally on all bits of the message
Using Symmetric Ciphers for MACs

- can use any block cipher chaining mode and use 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 ≤ M ≤ 64) of final block
- but final MAC is now too small for security
Hash Functions

- condenses arbitrary message to fixed size
  \[ h = H(M) \]

- usually assume that the hash function is public and not keyed
  - cf. MAC which is keyed

- hash used to detect changes to message

- can use in various ways with message

- most often to create a digital signature
Hash Functions & Digital Signatures
Requirements for Hash Functions

1. can be applied to any sized message $M$
2. produces fixed-length output $h$
3. is easy to compute $h = H(M)$ for any message $M$
4. given $h$ is infeasible to find $x$ s.t. $H(x) = h$
   - one-way property
5. given $x$ is infeasible to find $y$ s.t. $H(y) = H(x)$
   - weak collision resistance
6. is infeasible to find any $x, y$ s.t. $H(y) = H(x)$
   - strong collision resistance
Simple Hash Functions

- are several proposals for simple functions
- based on XOR of message blocks
- not secure since can manipulate any message and either not change hash or change hash also
- need a stronger cryptographic function (next chapter)
Birthday Attacks

- might think a 64-bit hash is secure
- but by **Birthday Paradox** is not
- **birthday attack** works thus:
  - opponent generates $2^{m/2}$ variations of a valid message all with essentially the same meaning
  - opponent also generates $2^{m/2}$ variations of a desired fraudulent message
  - two sets of messages are compared to find pair with same hash (probability > 0.5 by birthday paradox)
  - have user sign the valid message, then substitute the forgery which will have a valid signature
- conclusion is that need to use larger MAC/hash
Block Ciphers as Hash Functions

- can use block ciphers as hash functions
  - using $H_0=0$ and zero-pad of final block
  - compute: $H_i = E_{M_i}[H_{i-1}]$
  - and use final block as the hash value
  - similar to CBC but without a key

- resulting hash is too small (64-bit)
  - both due to direct birthday attack
  - and to “meet-in-the-middle” attack

- other variants also susceptible to attack
Hash Functions & MAC Security

- like block ciphers have:
- brute-force attacks exploiting
  - strong collision resistance hash have cost $2^{m/2}$
    - have proposal for h/w MD5 cracker
    - 128-bit hash looks vulnerable, 160-bits better
  - MACs with known message-MAC pairs
    - can either attack keyspace (cf key search) or MAC
    - at least 128-bit MAC is needed for security
Hash Functions & MAC
Security

- **cryptanalytic attacks** exploit structure
  - like block ciphers want brute-force attacks to be the best alternative
- have a number of analytic attacks on iterated hash functions
  - CVᵢ = f(CVᵢ₋₁, Mᵢ); H(M) = CVₙ
  - typically focus on collisions in function f
  - like block ciphers is often composed of rounds
  - attacks exploit properties of round functions
Summary

have considered:

- message authentication using
- message encryption
- MACs
- hash functions
- general approach & security
At cats' green on the Sunday he took the message from the inside of the pillar and added Peter Moran's name to the two names already printed there in the "Brontosaur" code. The message now read: “Leviathan to Dragon: Martin Hillman, Trevor Allan, Peter Moran: observe and tail.” What was the good of it John hardly knew. He felt better, he felt that at last he had made an attack on Peter Moran instead of waiting passively and effecting no retaliation. Besides, what was the use of being in possession of the key to the codes if he never took advantage of it?

—Talking to Strange Men, Ruth Rendell
Message Authentication

- Message authentication is concerned with:
  - protecting the integrity of a message
  - validating identity of originator
  - non-repudiation of origin (dispute resolution)

- Will consider the security requirements

- Then three alternative functions used:
  - message encryption
  - message authentication code (MAC)
  - hash function
Security Requirements

- disclosure
- traffic analysis
- masquerade
- content modification
- sequence modification
- timing modification
- source repudiation
- destination repudiation
Message Encryption

- Message encryption by itself also provides a measure of authentication.
- If symmetric encryption is used then:
  - Receiver know sender must have created it.
  - Since only sender and receiver now key used.
  - Know content cannot of been altered.
  - If message has suitable structure, redundancy or a checksum to detect any changes.
Message Encryption

- if public-key encryption is used:
  - encryption provides no confidence of sender
  - since anyone potentially knows public-key
  - however if
    - sender \textbf{signs} message using their private-key
    - then encrypts with recipients public key
    - have both secrecy and authentication
  - again need to recognize corrupted messages
  - but at cost of two public-key uses on message
Message Authentication Code (MAC)

- generated by an algorithm that creates a small fixed-sized block
  - 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 it matches the MAC
- provides assurance that message is unaltered and comes from sender
Message Authentication Code

(a) Message authentication

Source A

M

K

C

Destination B

M

C

C(K, M)

Compare

K
Message Authentication Codes

- as shown the MAC provides 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 (e.g., archival use)
- note that a MAC is not a digital signature
MAC Properties

- a MAC is a cryptographic checksum
  
  \[ \text{MAC} = C_K(M) \]

  - 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 have same MAC
  - but finding these needs to be very difficult
Requirements for MACs

- taking into account the types of attacks
- need the MAC to satisfy the following:
  1. knowing a message and MAC, is infeasible to find another message with same MAC
  2. MACs should be uniformly distributed
  3. MAC should depend equally on all bits of the message
Using Symmetric Ciphers for MACs

- can use any block cipher chaining mode and use 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 ($16 \leq M \leq 64$) of final block
- but final MAC is now too small for security
Data Authentication Algorithm
Hash Functions

- condenses arbitrary message to fixed size
  \[ h = H(M) \]
- usually assume that the hash function is public and not keyed
  - cf. MAC which is keyed
- hash used to detect changes to message
- can use in various ways with message
- most often to create a digital signature
Hash Functions & Digital Signatures
Requirements for Hash Functions

1. can be applied to any sized message $M$
2. produces fixed-length output $h$
3. is easy to compute $h=H(M)$ for any message $M$
4. given $h$ is infeasible to find $x$ s.t. $H(x)=h$
   - one-way property
5. given $x$ is infeasible to find $y$ s.t. $H(y)=H(x)$
   - weak collision resistance
6. is infeasible to find any $x,y$ s.t. $H(y)=H(x)$
   - strong collision resistance
Simple Hash Functions

- are several proposals for simple functions
- based on XOR of message blocks
- not secure since can manipulate any message and either not change hash or change hash also
- need a stronger cryptographic function (next chapter)
Birthday Attacks

- might think a 64-bit hash is secure
- but by **Birthday Paradox** is not
- **birthday attack** works thus:
  - opponent generates $2^{m/2}$ variations of a valid message all with essentially the same meaning
  - opponent also generates $2^{m/2}$ variations of a desired fraudulent message
  - two sets of messages are compared to find pair with same hash (probability > 0.5 by birthday paradox)
  - have user sign the valid message, then substitute the forgery which will have a valid signature
- conclusion is that need to use larger MAC/hash
Block Ciphers as Hash Functions

- can use block ciphers as hash functions
  - using $H_0=0$ and zero-pad of final block
  - compute: $H_i = E_{M_i} [H_{i-1}]$
  - and use final block as the hash value
  - similar to CBC but without a key

- resulting hash is too small (64-bit)
  - both due to direct birthday attack
  - and to “meet-in-the-middle” attack

- other variants also susceptible to attack
Hash Functions & MAC Security

- like block ciphers have:
- **brute-force** attacks exploiting
  - strong collision resistance hash have cost $2^{m/2}$
    - have proposal for h/w MD5 cracker
    - 128-bit hash looks vulnerable, 160-bits better
  - MACs with known message-MAC pairs
    - can either attack keyspace (cf key search) or MAC
    - at least 128-bit MAC is needed for security
Hash Functions & MAC

Security

- **cryptanalytic attacks** exploit structure
  - like block ciphers want brute-force attacks to be the best alternative

- have a number of analytic attacks on iterated hash functions
  - $CV_i = f[CV_{i-1}, M_i]; H(M)=CV_N$
  - typically focus on collisions in function f
  - like block ciphers is often composed of rounds
  - attacks exploit properties of round functions
Summary

- have considered:
  - message authentication using
  - message encryption
  - MACs
  - hash functions
  - general approach & security
To guard against the baneful influence exerted by strangers is therefore an elementary dictate of savage prudence. Hence before strangers are allowed to enter a district, or at least before they are permitted to mingle freely with the inhabitants, certain ceremonies are often performed by the natives of the country for the purpose of disarming the strangers of their magical powers, or of disinfecting, so to speak, the tainted atmosphere by which they are supposed to be surrounded.

—The Golden Bough, Sir James George Frazer
Digital Signatures

- have looked at message authentication
  - but does not address issues of lack of trust
- digital signatures provide the ability to:
  - verify author, date & time of signature
  - authenticate message contents
  - be verified by third parties to resolve disputes
- hence include authentication function with additional capabilities
Digital Signature Properties

- must depend on the message signed
- must use information unique to 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 save digital signature in storage
Direct Digital Signatures

- involve only sender & receiver
- assumed receiver has sender’s public-key
- digital signature made by sender signing entire message or hash with private-key
- can encrypt using receivers public-key
- important that sign first then encrypt message & signature
- security depends on sender’s private-key
Arbitrated Digital Signatures

- involves use of arbiter A
  - validates any signed message
  - then dated and sent to recipient
- requires suitable level of trust in arbiter
- can be implemented with either private or public-key algorithms
- arbiter may or may not see message
Authentication Protocols

- used to convince parties of each other's identity and to exchange session keys
- may be one-way or mutual
- key issues are
  - confidentiality – to protect session keys
  - timeliness – to prevent replay attacks
- published protocols are often found to have flaws and need to be modified
Replay Attacks

where a valid signed message is copied and later resent
- simple replay
- repetition that can be logged
- repetition that cannot be detected
- backward replay without modification

countermeasures include
- use of sequence numbers (generally impractical)
- timestamps (needs synchronized clocks)
- challenge/response (using unique nonce)
Using Symmetric Encryption

- as discussed previously can use a two-level hierarchy of keys
- usually with a trusted Key Distribution Center (KDC)
  - each party shares own master key with KDC
  - KDC generates session keys used for connections between parties
  - master keys used to distribute these to them
Needham-Schroeder Protocol

- original third-party key distribution protocol
- for session between A B mediated by KDC
- protocol overview is:
  1. A -> KDC: \( ID_A \| ID_B \| N_1 \)
  2. KDC -> A: \( E_{Ka}[Ks \| ID_B \| N_1 \| E_{Kb}[Ks\|ID_A] \] \)
  3. A -> B: \( E_{Kb}[Ks\|ID_A] \)
  4. B -> A: \( E_{Ks}[N_2] \)
  5. A -> B: \( E_{Ks}[f(N_2)] \)
Needham-Schroeder Protocol

- used to securely distribute a new session key for communications between A & B
- but is vulnerable to a replay attack if an old session key has been compromised
  - then message 3 can be resent convincing B that is communicating with A
- modifications to address this require:
  - timestamps (Denning 81)
  - using an extra nonce (Neuman 93)
Using Public-Key Encryption

- have a range of approaches based on the use of public-key encryption
- need to ensure have correct public keys for other parties
- using a central Authentication Server (AS)
- various protocols exist using timestamps or nonces
Denning AS Protocol

- Denning 81 presented the following:
  1. A -> AS: $ID_A \parallel ID_B$
  2. AS -> A: $E_{PRas}[ID_A\parallel PU_a\parallel T] \parallel E_{PRas}[ID_B\parallel PU_b\parallel T]$
  3. A -> B: $E_{PRas}[ID_A\parallel PU_a\parallel T] \parallel E_{PRas}[ID_B\parallel PU_b\parallel T] \parallel E_{PUb}[E_{PRas}[K_s\parallel T]]$

- note session key is chosen by A, hence AS need not be trusted to protect it
- timestamps prevent replay but require synchronized clocks
One-Way Authentication

- required when sender & receiver are not in communications at same time (eg. email)
- have header in clear so can be delivered by email system
- may want contents of body protected & sender authenticated
Using Symmetric Encryption

- can refine use of KDC but can’t have final exchange of nonces, vis:
  1. A->KDC: $ID_A \| ID_B \| N_1$
  2. KDC -> A: $E_{K_a}[Ks \| ID_B \| N_1 \| E_{K_b}[Ks\|ID_A] ]$
  3. A -> B: $E_{K_b}[Ks\|ID_A] \| E_{K_s}[M]$

- does not protect against replays
  - could rely on timestamp in message, though email delays make this problematic
Public-Key Approaches

- have seen some public-key approaches
- if confidentiality is major concern, can use:
  
  A\rightarrow B: E_{\text{Pub}}[K_s] \parallel E_{K_s}[M]

  - has encrypted session key, encrypted message

- if authentication needed use a digital signature with a digital certificate:
  
  A\rightarrow B: M \parallel E_{\text{PrA}}[H(M)] \parallel E_{\text{PrAs}}[T||\text{ID}_A||\text{PU}_a]

  - with message, signature, certificate
Digital Signature Standard (DSS)

- US Govt approved signature scheme
- designed by NIST & NSA in early 90's
- published as FIPS-186 in 1991
- revised in 1993, 1996 & then 2000
- uses the SHA hash algorithm
- DSS is the standard, DSA is the algorithm
- FIPS 186-2 (2000) includes alternative RSA & elliptic curve signature variants
Digital Signature Algorithm (DSA)

- creates a 320 bit signature
- with 512-1024 bit security
- smaller and faster than RSA
- a digital signature scheme only
- security depends on difficulty of computing discrete logarithms
- variant of ElGamal & Schnorr schemes
Digital Signature Algorithm (DSA)
DSA Key Generation

- have shared global public key values \((p,q,g)\):
  - choose \(q\), a 160 bit
  - choose a large prime \(p = 2^L\)
    - where \(L = 512\) to 1024 bits and is a multiple of 64
    - and \(q\) is a prime factor of \((p-1)\)
  - choose \(g = h^{(p-1)/q}\)
    - where \(h < p-1\), \(h^{(p-1)/q} \mod p > 1\)

- users choose private & compute public key:
  - choose \(x < q\)
  - compute \(y = g^x \mod p\)
DSA Signature Creation

- **to sign a message** $M$ **the sender:**
  - generates a random signature key $k$, $k < q$
  - nb. $k$ must be random, be destroyed after use, and never be reused

- **then computes signature pair:**
  
  $r = (g^k \mod p)(\mod q)$
  $s = (k^{-1}.H(M) + x.r)(\mod q)$

- **sends signature** $(r, s)$ **with message** $M$
having received M & signature \((r, s)\)

to verify a signature, recipient computes:

\[
\begin{align*}
  w &= s^{-1} \pmod{q} \\
  u_1 &= (H(M).w)(\mod q) \\
  u_2 &= (r.w)(\mod q) \\
  v &= (g^{u_1}.y^{u_2}(\mod p)) \pmod{q}
\end{align*}
\]

if \(v=r\) then signature is verified

see book web site for details of proof why
Summary

- have discussed:
  - digital signatures
  - authentication protocols (mutual & one-way)
  - digital signature algorithm and standard