Message Integrity, Cryptographic Hash Functions and Digital Signatures: Part 1

Gaurav S. Kasbekar
Dept. of Electrical Engineering
IIT Bombay

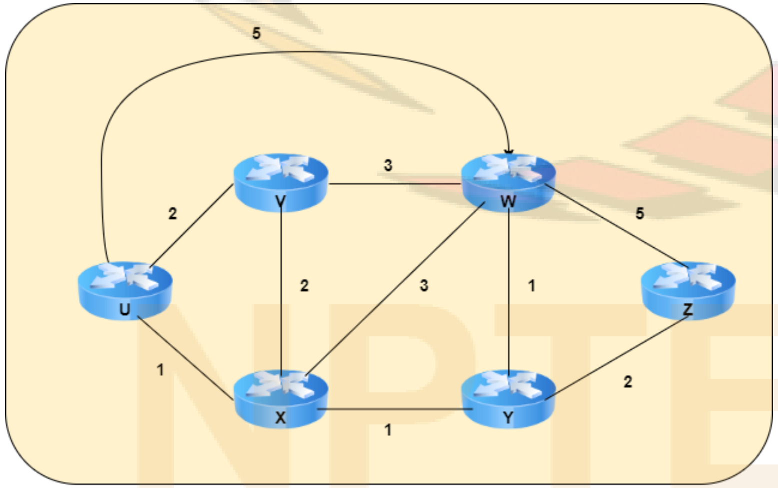
References

- J. Kurose, K. Ross, "*Computer Networking: A Top Down Approach*", Sixth Edition, Pearson Education, 2013
- C. Kaufman, R. Perlman, M. Speciner, "*Network Security: Private Communication in a Public World*", Pearson Education, 2nd edition, 2002
- A. Tanenbaum, D. Wetherall, "Computer Networks", Fifth Edition, Pearson Education, 2012

Message Integrity

Example

- Consider a network using Dijkstra's algorithm to find routes between every pair of routers
  - ❑ each router broadcasts a list of its neighbors and costs of corresponding links to all other routers
    - message containing this list called "link-state message"
  - ❑ each router then knows network topology; computes shortest paths using Dijkstra's algorithm
- Attack on such a routing algorithm:
  - ❑ intruder can send a bogus link-state message to a router B, with source address that of router A
  - ❑ or can modify link-state message from A before forwarding it to B
- So when a router B receives a link-state message with source address that of router A, needs to verify that:
  - ❑ router A actually created the message
  - ❑ message was not modified while being forwarded from A to B



Message Integrity Problem

- Bob receives a message with source address that of Alice
  - ❑may be encrypted or in plaintext
- Bob needs to verify that:
  - ❑Alice actually created the message
  - ❑Message was not modified while being forwarded from Alice to Bob

Attempt 1

- Let m denote the message to be sent from Alice to Bob
- Alice performs the following actions:
  - ❑computes checksum of m, say $c(m)$
    - o *e.g.*, odd parity checksum
  - ❑sends $(m, c(m))$ to Bob
- Bob checks whether checksum of m is $c(m)$
- Does this achieve message integrity?
- No, since an intruder, say Trudy, can:
  - ❑create a bogus message m'
  - ❑send $(m', c(m'))$ to Bob with source address that of Alice

Attempt 2

- Alice performs the following actions:
  - ❑computes checksum of m, say $c(m)$
  - ❑encrypts it to get $K_A(c(m))$
  - ❑sends $(m, K_A(c(m)))$ to Bob
- Bob finds $K_B(K_A(c(m))) = c(m)$ and checks whether it equals checksum of m
- Does this achieve message integrity?
- No since an intruder, say Trudy, can:
  - ❑create a bogus message m' such that $c(m') = c(m)$
  - ❑send $(m', K_A(c(m)))$ to Bob with source address that of Alice

Attempt 3

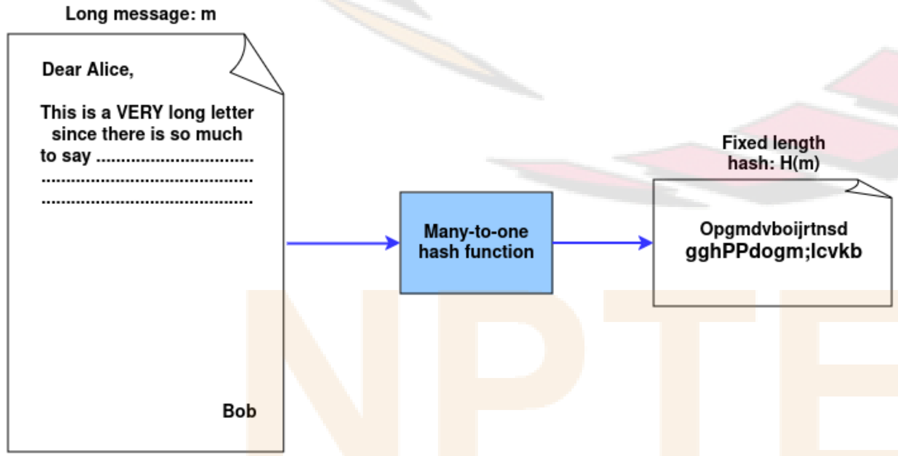
- Alice performs the following actions:
  - computes checksum of m, say $c(m)$
  - concatenates m and $c(m)$ to get $(m, c(m))$
  - sends its encrypted version, $K_A(m, c(m))$, to Bob
- Bob finds $K_B(K_A(m, c(m))) = (m, c(m))$ and checks whether checksum of m equals $c(m)$
- Does this achieve message integrity?
  - Yes
- Shortcoming of this approach:
  - requires sender to encrypt entire message m, which is time-consuming
  - m may not be confidential (*e.g.*, link-state message)
  - also, in some cases, encryption of message may not be allowed (*e.g.*, to allow security agencies to monitor all communications)
- Want to send m in plaintext form, and still achieve message integrity

Basic Idea

- Recall: Attempt 2 failed because intruder could create a bogus message m' such that $c(m') = c(m)$
- Suppose we use a function $H(.)$ in place of $c(.)$ with the property that given m, it is computationally infeasible to find a message m' such that $H(m') = H(m)$
- Then Attempt 2 would work

Cryptographic Hash Function

- A cryptographic hash function is a function $H(.)$ such that:
  - ❑ for an input message m of arbitrary length, the output $H(m)$ is a fixed length string ($e.g.$, 128 bits)
  - ❑ given m, $H(m)$ can be computed fast (much faster than encrypting m)
  - ❑ it is computationally infeasible to find a message m that has a pre-specified hash h, i.e., m such that $H(m) = h$
  - ❑ it is computationally infeasible to find two messages m and m' such that $H(m') = H(m)$
  - ❑ given m, it is computationally infeasible to find a different message m' such that $H(m') = H(m)$
- Popular cryptographic hash functions:
  - ❑ SHA-1, SHA-2, SHA-3 (Secure Hash Algorithm 1, 2, 3)
    - o A SHA-1 collision was found in 2017: https://www.theregister.co.uk/2017/02/23/google_first_sha1_collision/
  - ❑ MD5 (Message Digest 5)
    - o collisions were found between 2004 and 2007 (see Tanenbaum, Section 8.4.3)



Example

- Bob owes Alice $\$100.99$ and sends the following message to her:
  - “IOU100.99BOB”
- ASCII representation found and groups of four bytes each added to get checksum
- Fraudulent message with same checksum:
  - “IOU900.19BOB”
- So above checksum would make a poor cryptographic hash function
- Much more complicated functions used in practice

Message				ASCII Representation				
I	O	U	1	49	4F	55	31	
0	0	.	9	30	30	2E	39	
9	B	O	B	39	42	4F	42	
				B2	C1	D2	AC	Checksum

Message				ASCII Representation				
I	O	U	9	49	4F	55	39	
0	0	.	1	30	30	2E	31	
9	B	O	B	39	42	4F	42	
				B2	C1	D2	AC	Checksum

Modified Version of Attempt 2

- Alice performs the following actions:
  - computes hash of m, say $H(m)$
  - encrypts it to get $K_A(H(m))$
  - sends $(m, K_A(H(m)))$ to Bob
- Bob finds $K_B(K_A(H(m))) = H(m)$ and checks whether it equals hash of m
- *Achieves message integrity*

Achieving Message Integrity without using Encryption

- Modified version of Attempt 2 achieves message integrity, but requires encryption; time-consuming
- Assume that Alice and Bob have a shared secret bit string s
  - ❑ possibly shared using public-key encryption
- Want to achieve message integrity without using encryption
- Alice performs the following actions:
  - ❑ concatenates m and s to get (m, s); computes $H(m, s)$
  - ❑ sends $(m, H(m, s))$ to Bob
- Bob computes $H(m, s)$ using m and s; checks whether it equals $H(m, s)$ sent by Alice
- Above approach achieves message integrity without using encryption
- **Terminology**: "Message Integrity" problem also called "Message authentication" problem; s called "*authentication key*" and $H(m, s)$ called "*Message Authentication Code*" (MAC)