Fundamentals of Information & Network Security
ECE 471/571

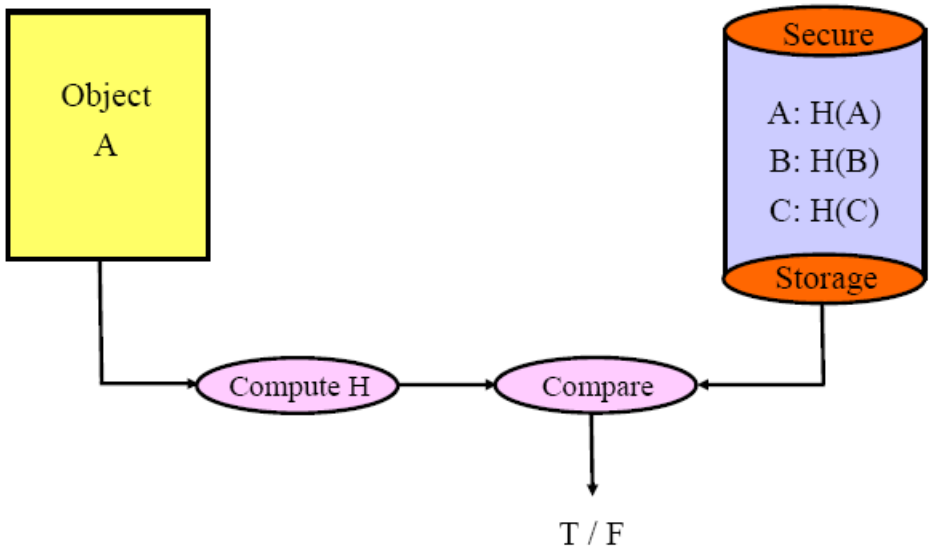
Lecture #19: More Applications of Hash

Instructor: Ming Li

Dept of Electrical and Computer Engineering

University of Arizona

File Fingerprint with Hash



Source Authentication

- Assume that a company wants to store a file with all the passwords of its clients.
- Store plaintext passwords: problems?
- Store hash of passwords – better solution
  - Problems again?

One-Way Hash Chains

- Construction
  - Pick random r_0 and public one-way function h
  - $r_{i+1} = h(r_i)$
  - Secret value: r_0 , **public** value r_N

- Properties
  - Use in reverse order of construction: r_N , r_{N-1} … r_1
  - Infeasible to derive r_i from r_j (j>i)
  - Efficiently authenticate r_i knowing r_j (j>i): verify $r_j = h^{j-i}(r_i)$
  - Robust to missing values

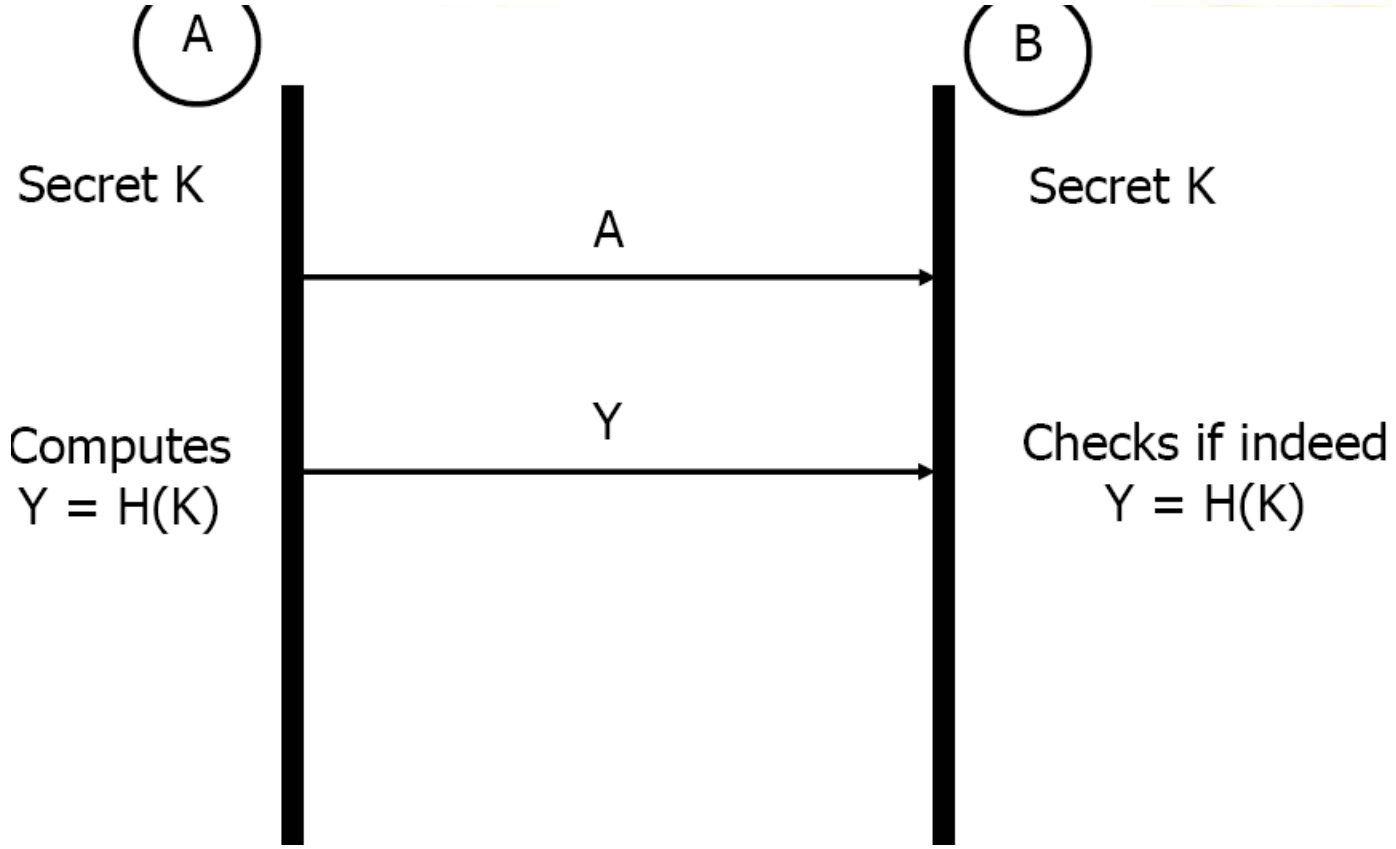
One-Way Chain Application

- Secret key one-time password system
  - Use a different password at every login
  - Server cannot derive password for next login
- Solution:
  - User pick random password P_0
  - Prepare sequence of passwords $P_{i+1} = h(P_i)$ and register P_N
  - Use passwords P_N , P_{N-1} , …, P_1 , P_0
  - Server can easily authenticate user

Authentication Protocol with Hash

- Goal: A wishes to identify and authenticate himself to B
- Infrastructure: A and B share a long-lived secret key K
- Naive Authentication Protocol
  - A identifies himself to B
  - A sends to B hash of secret key K
  - B verifies hash of secret key K

Authentication with Hash

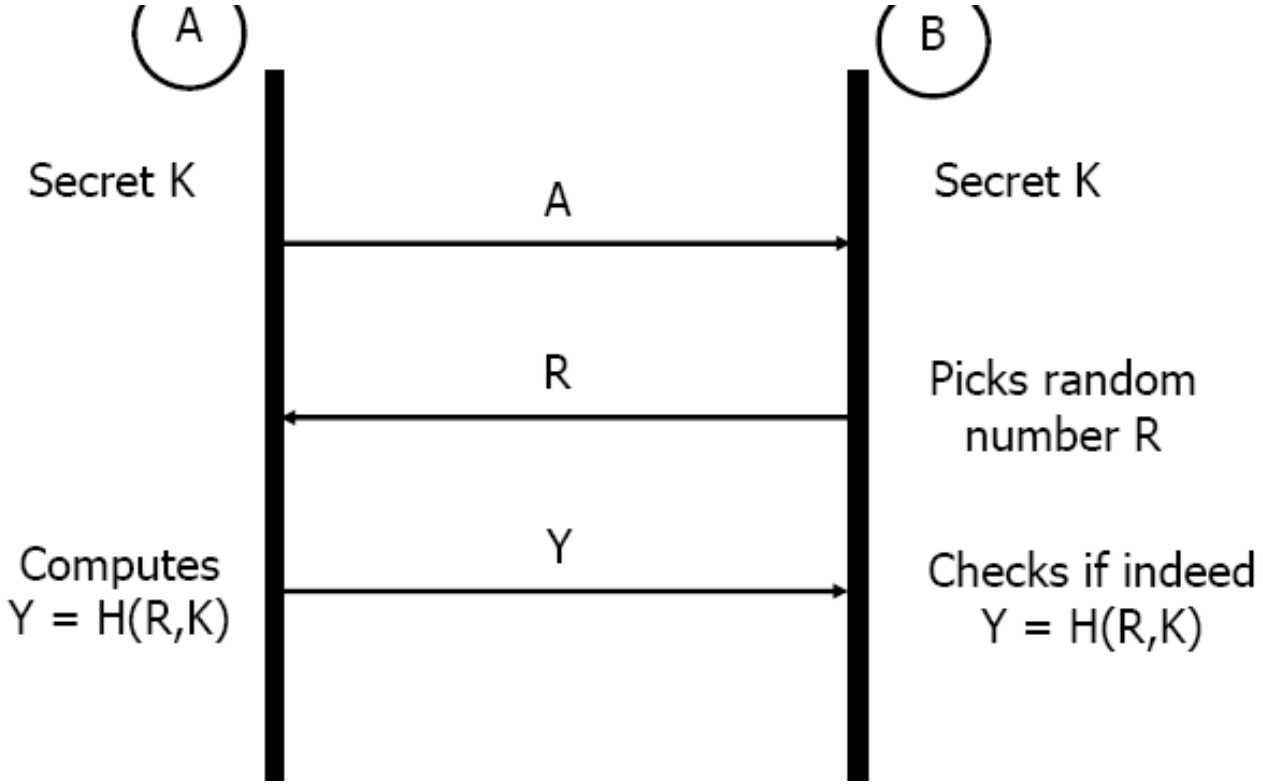


Problem?
Solution?

Authentication with Hash

- Modified Authentication Protocol
  - A identifies himself to B
  - B sends to A a random number R
  - A sends to B hash of random number R and secret key K
  - B verifies hash of random number R and secret key K

Authentication with Hash

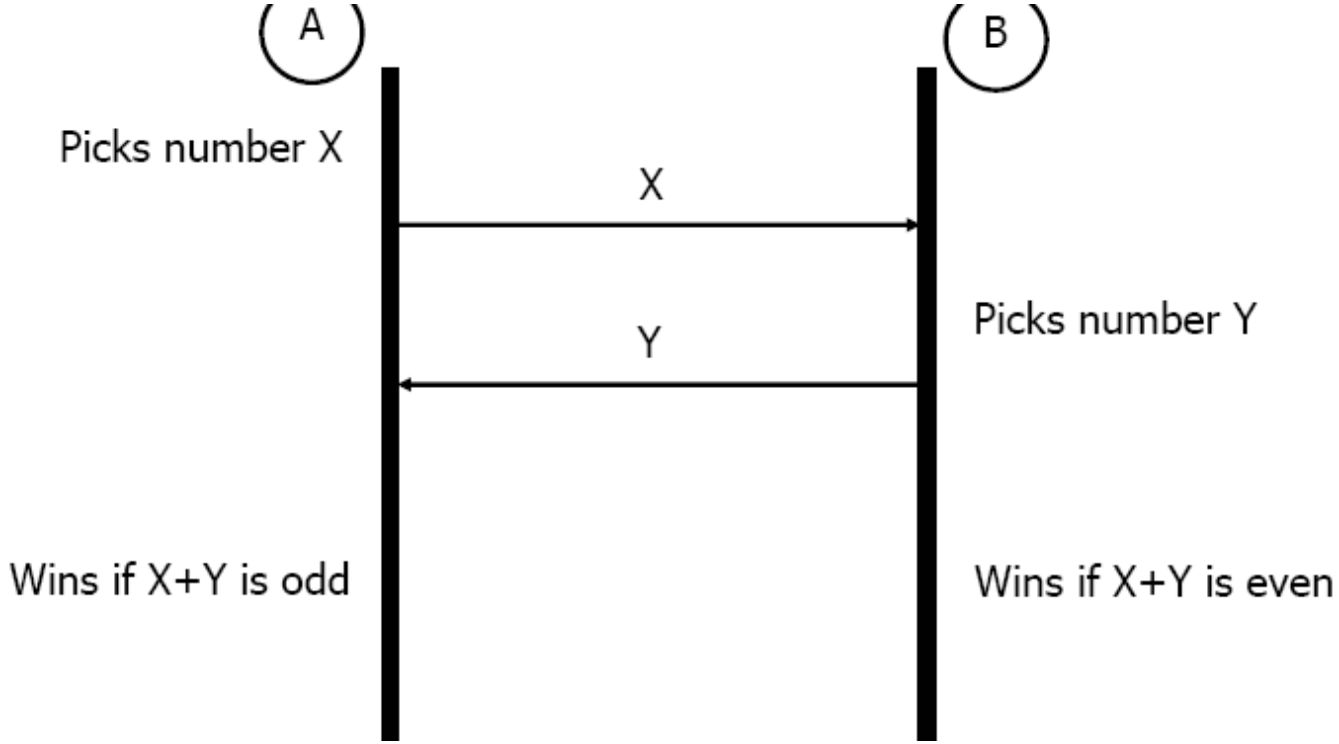


Solution: This protocol is sound against sniffing and replay

Commitment Protocol with Hash

- Goal: A and B wish to play “odd or even” over the network
- Naive Commitment Protocol
  - A picks a number X and sends it to B
  - B picks a number Y and sends it to A
  - A wins if X+Y is odd
  - B wins if X+Y is even
- Problem: How can we guarantee that B doesn’t cheat?
- Solution?

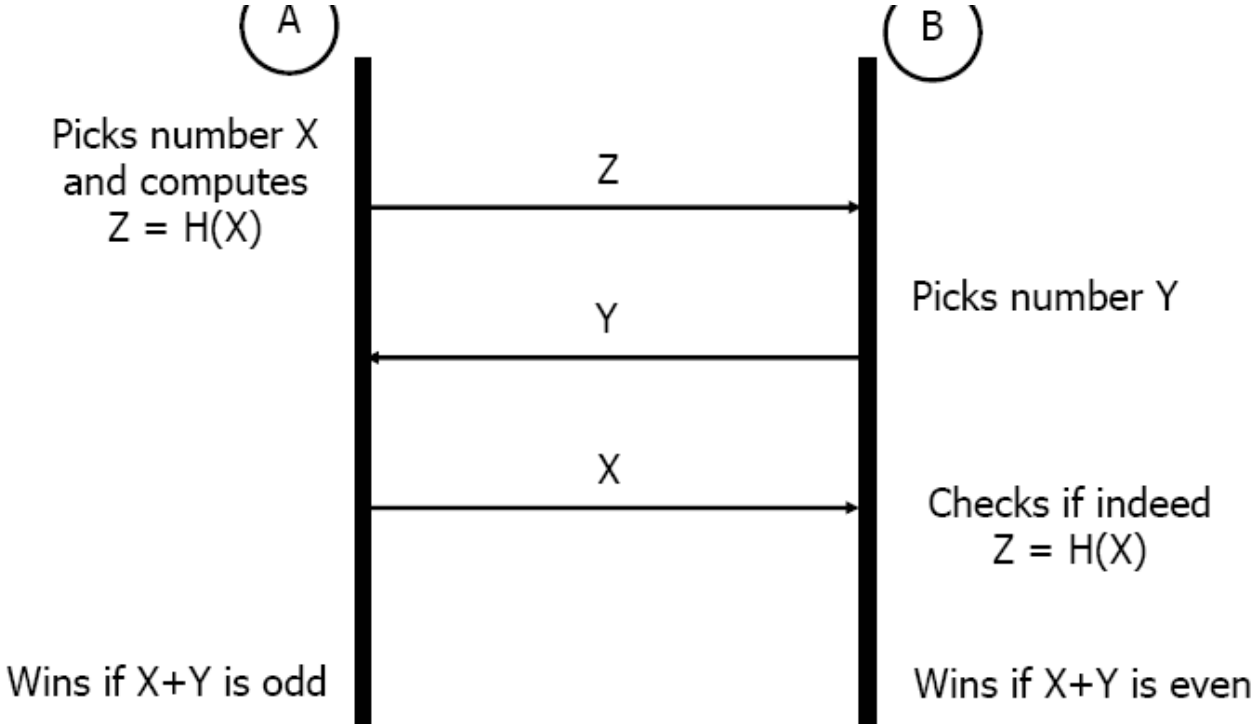
Commitment Protocol with Hash



Commitment Protocol with Hash

- Modified Commitment Protocol
  - A picks a number X and sends value of Z = H(X) to B
  - B picks a number Y and sends value of Y to A
  - A now sends value of X to B
  - B checks if X complies with Z that was sent before
  - A wins if X+Y is odd
  - B wins if X+Y is even
- Solution: In this protocol B cannot cheat

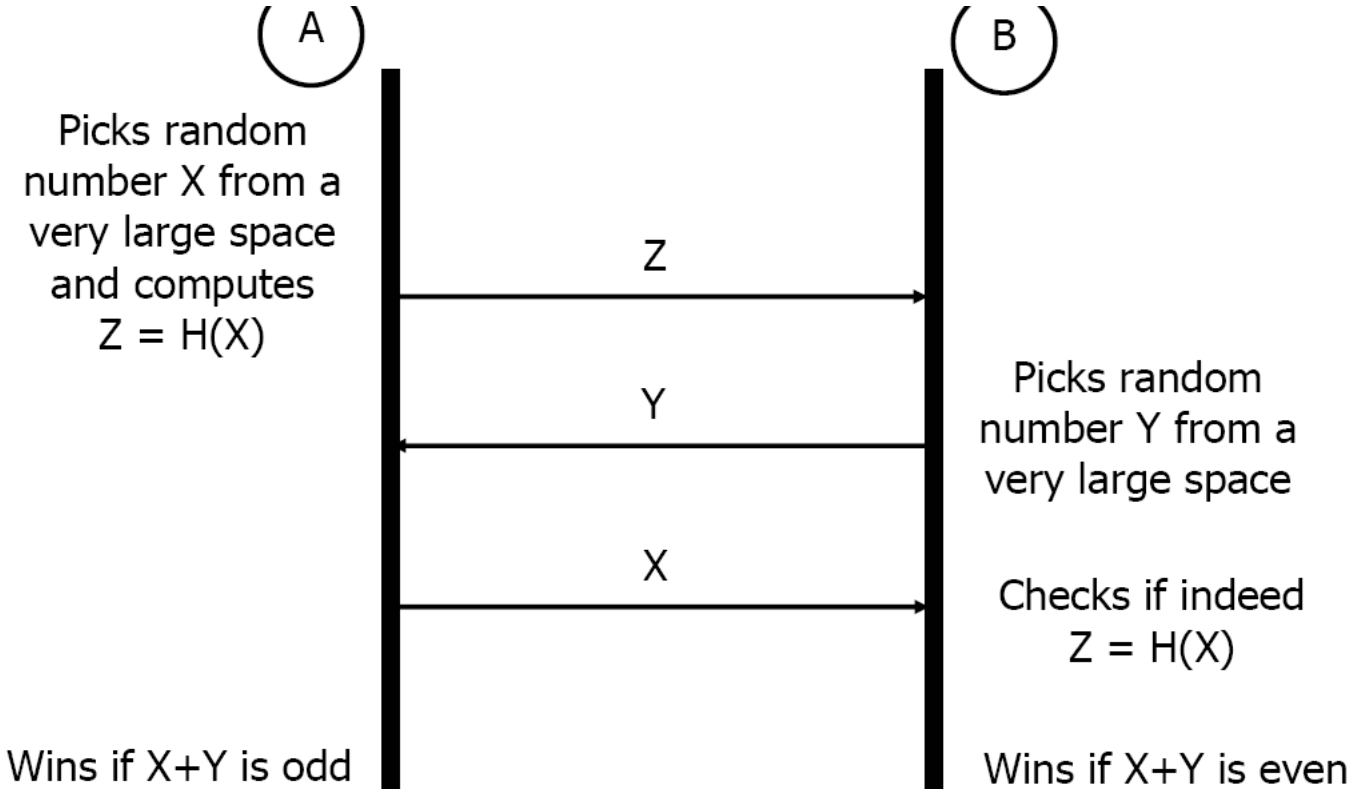
Commitment Protocol with Hash



Commitment Protocol with Hash

- Hash function does two things in the protocol:
  - Hides the number X from B at the beginning of the game
  - Makes A commit to the number X until the end of the game
- Question: What if A always picks small numbers so that B can make a list of all the hash values?
- Answer: A should select random values for the protocol:
  - Select the number X from a very large space of numbers
  - Mask the number X with a random noise from a very large space

Commitment Protocol with Hash



Encryption with Hash

- Counter-intuitive?

- Generate one-time pad (similar to OFB)
  $b_1 = MD(K_{AB}|IV)$
  $b_2 = MD(K_{AB}|b_1)$
  $b_3 = MD(K_{AB}|b_2)$
  …
- Then, XOR the message with the one-time pad bit sequence. (Problem with one-time pad? Recall OFB)

- Mixing in the plaintext
  $b_1 = MD(K_{AB}|IV)$, $c_1 = m_1$ XOR b_1
  $b_2 = MD(K_{AB}|c_1)$, $c_2 = m_2$ XOR b_2
  $b_3 = MD(K_{AB}|c_2)$, $c_3 = m_3$ XOR b_3
  …