Authentication: Part 3

Gaurav S. Kasbekar
Dept. of Electrical Engineering
IIT Bombay

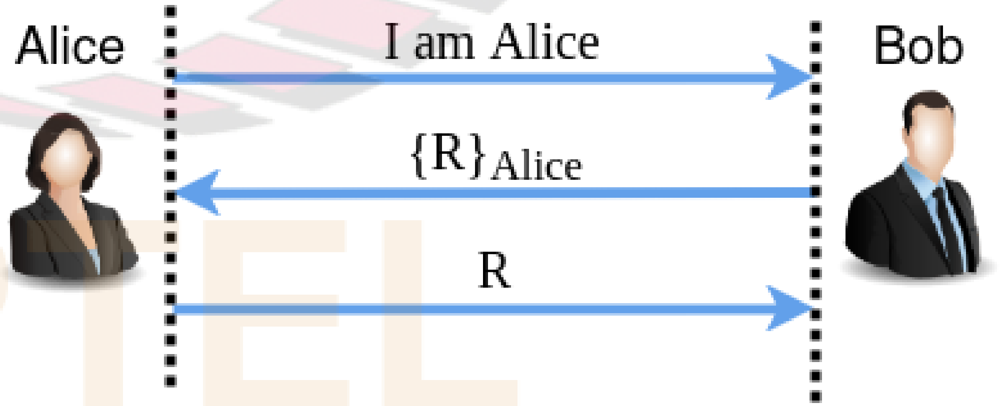
References

- 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

Other Protocols for One-Way Authentication

Recall: Authentication Using Public Keys

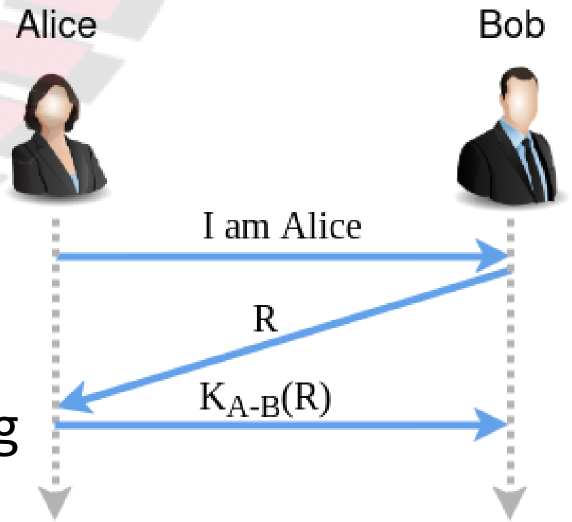
- Alice has a public key-private key pair, with the public key being known to Bob
- In figs., $[R]_{\text{Alice}}$ is nonce, R, signed by Alice; $\{R\}_{\text{Alice}}$ is nonce encrypted using Alice's public key
- Both of the protocols in fig. defend against:
  - ❑ eavesdropping by intruder; as well as
  - ❑ server database reading attack
- That is, intruder, Trudy, will not be able to impersonate Alice if:
  - ❑ she eavesdrops on conversation or reads database at Bob or both



Defence Against Eavesdropping and Server Database Reading Attack

- Want to authenticate Alice to Bob, while defending against *both* eavesdropping and server database reading attack
- Can this be done *without using public-key cryptography*?
- First, we show that it is easy to defend against any one of the two attacks, if we do not defend against the other

- Defence against eavesdropping:
  - ❑ Suppose Alice and Bob have a shared symmetric key $K_{Alice-Bob}$
  - ❑ Protocol ap4.0 (shown in fig.) and its variation in which Alice sends $H(K_{Alice-Bob}, R)$ to Bob instead of $K_{Alice-Bob}(R)$ defend against eavesdropping
  - ❑ Do not defend against server database reading attack



Defence Against Eavesdropping and Server Database Reading Attack (contd.)

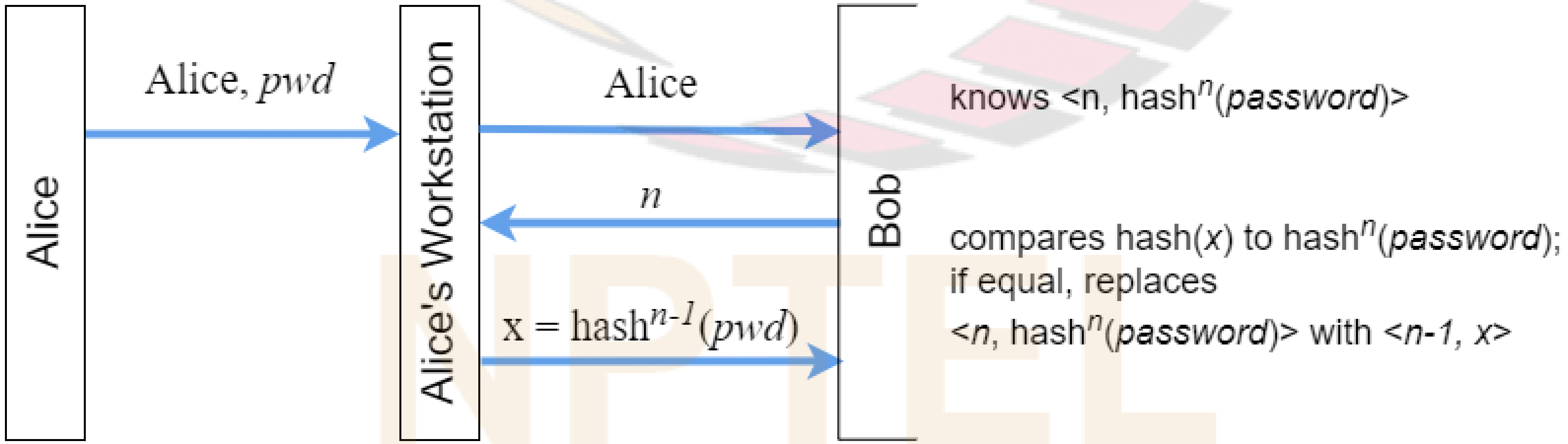
- Defence against server database reading attack:
  - Alice selects a password, say p
  - Bob stores hash of p, say $H(p)$
  - To authenticate, Alice sends the message "I am Alice, p" to Bob
  - Bob finds hash value of p and checks whether it equals $H(p)$
  - Defends against server database reading attack, but not against eavesdropping
- Next, we will study a protocol, which defends against *both* eavesdropping and the server database reading attack *without using public-key cryptography*
  - called "*Lamport's Hash*" (since it was invented by Leslie Lamport)

Lamport's Hash

- Alice (a human) remembers a password
- Bob (a server) has a database, where it stores for each user:
  - username
  - n, an integer which decrements each time Bob authenticates the user
  - $\mathrm{hash}^n(password)$, i.e., $\mathrm{hash}(\mathrm{hash}(\ldots(\mathrm{hash}(password))\ldots))$
- Before Alice and Bob can use the authentication protocol, the password database entry at Bob for Alice is configured as follows:
  - Alice chooses a password, her workstation computes $\mathrm{hash}^n(password)$, where n is some large value like 1000
  - the values <Alice, n, $\mathrm{hash}^n(password)$> are sent to Bob
  - some way is needed to securely communicate <Alice, n, $\mathrm{hash}^n(password)$> from Alice to Bob; but this is required only once every n authentication attempts by Alice
    - this is a drawback of Lamport's Hash scheme

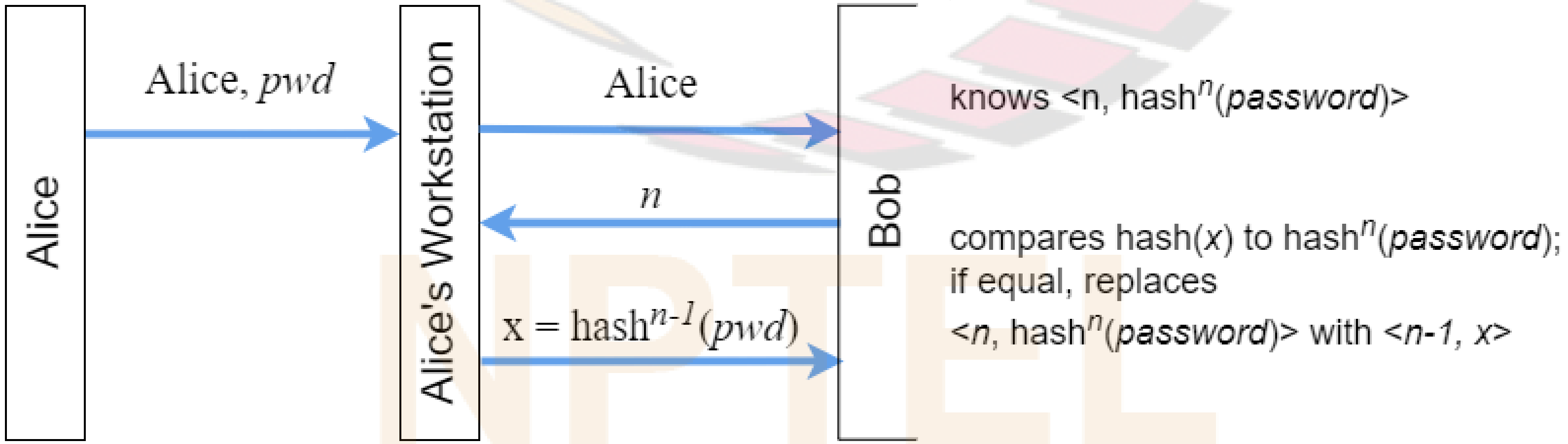
Lamport's Hash (contd.)

- Suppose password database entry at Bob for Alice has been configured
- When Alice wants to authenticate to Bob:
  - ❑ Alice types her username and password at her workstation
  - ❑ Alice's workstation sends username "Alice" to Bob
  - ❑ Bob sends "n" to Alice's workstation
  - ❑ Alice's workstation computes $x = \mathrm{hash}^{n-1}(password)$ and sends it to Bob
  - ❑ Bob takes x, computes its hash, $\mathrm{hash}(x)$, and compares result to $\mathrm{hash}^{n}(password)$
  - ❑ If the two match, Bob considers response valid and Alice's authentication is successful
  - ❑ Then Bob replaces <$n, \mathrm{hash}^{n}(password)$> with <$n-1, x$>
- When value of n gets to 1:
  - ❑ Password database entry at Bob for Alice has to be reconfigured



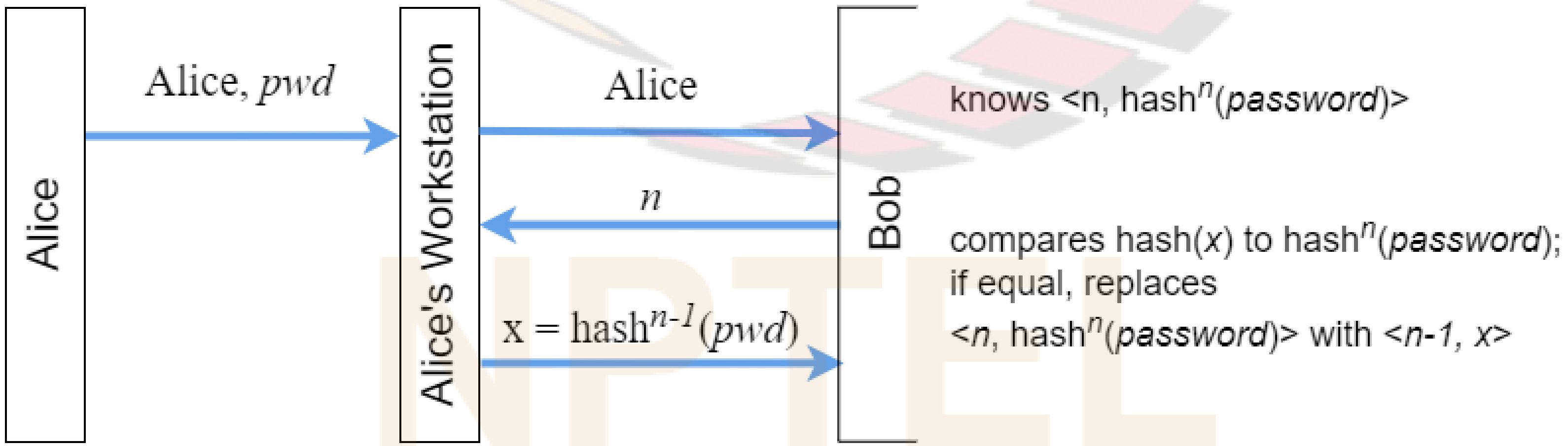
Lamport's Hash (contd.)

- Does this scheme defend against eavesdropping by an intruder?
  - ❑ Yes; an eavesdropping intruder, Trudy, will obtain the values of n and $\text{hash}^{n-1}(password)$
  - ❑ But if she attempts to authenticate as Alice, Bob will send $n-1$ to her and Trudy does not know $\text{hash}^{n-2}(password)$
- Does this scheme defend against the server database reading attack?
  - ❑ Yes; if Trudy reads database at Bob, she will obtain $<n, \text{hash}^{n}(password)>$
  - ❑ But to authenticate as Alice, $\text{hash}^{n-1}(password)$ needs to be known, which is not known to Trudy



Lamport's Hash (contd.)

- Suppose Alice uses the same password to login to multiple servers
- Then an eavesdropping intruder who obtains the values of n and $\text{hash}^{n-1}(password)$ when Alice logs in to one server can:
  - use these values to authenticate as Alice at another server
- Defence against this attack:
  - During configuration, <Alice, n, $\text{hash}^{n}(password|servername)$> is communicated from Alice to the server
  - Authentication exchange: server sends n to Alice; Alice responds with $\text{hash}^{n-1}(password|servername)$



Lamport's Hash (contd.)

- Suppose an intruder, Trudy, impersonates Bob's network address and waits for Alice to log in
- When Alice attempts to log in to Bob:
  - ❑ Trudy sends a small value of n to Alice, say $n = 50$
  - ❑ suppose actual $n = 1000$
- When Alice responds with $\mathrm{hash}^{49}(password)$, Trudy has obtained enough information to impersonate herself as Alice to Bob ≈ 950 times
  - ❑ called "*small n attack*"
- Defence against this attack:
  - ❑ maintain a counter at Alice, which keeps track of correct value of n