

# Authentication Protocols

Guevara Noubir  
College of Computer and Information Science  
Northeastern University  
[noubir@ccs.neu.edu](mailto:noubir@ccs.neu.edu)

---

---


---

---

---

---

---



# Outline

- Overview of Authentication Systems
  - [Chapter 9]
- Authentication of People
  - [Chapter 10]
- Security Handshake Pitfalls
  - [Chapter 11]
- Strong Password Protocols
  - [Chapter 12]

Network Security      Authentication Protocols

---

---


---

---

---

---

---



# Who Is Authenticated?

- Human:**
  - Limited in terms of computation power and memory
- Machine:**
  - More powerful: long secrets, complex computation
- Hybrid:**
  - User is only authorized to execute some actions from a restricted set of machines
  - Users equipped with computation devices

Network Security      Authentication Protocols

---

---

---

---

---

---

---

## Password-Based Authentication

- Node *A* has a secret (*password*): e.g., "lisa"
- To authenticate itself *A* states the password
- No cryptographic operation because:
  - Difficult to achieve by humans when connecting from dumb terminals (less true today with authentication tokens)
  - Crypto could be overly expensive in implementation time or processing resources
  - Export or legal issues
- Problems:
  - Eavesdropping, cloning, etc.
- Should not be used in networked applications**

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---

## Offline vs. Online Password Guessing

- Online attack:
  - How? try passwords until accepted
  - Protection:
    - Limit number of trials and lock account: e.g., ATM machine
      - DoS problem: lock all accounts
    - Increase minimum time between trials
    - Prevent automated trials: from a keyboard, yahoo technique
    - Long passwords: pass phrases, initials of sentences, reject easy passwords
    - What is the protection used by Yahoo? Hotmail? Gmail?
- Offline attack:
  - How?
    - Attacker captures  $X = f(\text{password})$
    - Dictionary attack: try to guess the password value offline
    - Obtaining  $X$  in a unix system: "ypcat passwd"
    - Unix system: using the *salt*
  - Protection:
    - If offline attacks are possible then the secret space should be large

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---

## L0pht Statistics (old)

- L0phtCrack against LM (LanMan – Microsoft)
  - On 400 MHz quad-Xeon machine
  - Alpha-numeric: 5.5 hours
  - Alpha-numeric some symbols: 45 hours
  - Alpha-numeric-all symbols: 480 hours
- LM is weak but was still used by MS for compatibility reasons up to Windows XP, ... NTLM, ...
- Side Note on choosing good passwords:
  - [http://www.atstake.com/products/lc/best\\_practices.html](http://www.atstake.com/products/lc/best_practices.html)
  - Best practice from: SANS, MS, Red-Hat, etc.

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---



## Password Length

- n Online attacks:
  - n 4/6 digits can be sufficient if a user is given only three trials?
- n Offline attacks:
  - n Need: 64 random bits = 20 digits
    - n Too long to remember by a human!
  - n Or 11 characters from a-z, A-Z, 0-9, and punctuation marks
    - n Too long to remember by a human
  - n Or 16 characters pronounceable password (a vowel every two characters)
  - n Conclusion:
    - n A secret a person is willing to remember and type will not be as good as a 64-bit random number

Network Security Authentication Protocols

---

---

---

---

---

---

---

---



## Storing User Passwords

- n Alternatives:
  - n Each user's secret information is stored in every server
  - n The users secrets are stored in an *authentication storage node*
    - n Need to trust/authenticate/secure session with the ASN
  - n Use an *authentication facilitator node*. Alice's information is forwarded to the authentication facilitator who does the actual authentication
    - n Need to trust/authenticate/secure session with the AFN
- n Authentication information database:
  - n Encryption
  - n Hashed as in UNIX (allows offline attacks)

Network Security Authentication Protocols

---

---

---

---

---

---

---

---



## Other Issues Related to Passwords

- n Using a password in multiple places:
  - n Cascade break-in vs. writing the list of passwords
- n Requiring frequent changes
  - n How do users go around this?
- n A login Trojan horse to capture passwords
  - n Prevent programs from being able to mimic the login: X11 (take the whole screen), read keyboard has "?", "Ctrl-Alt-Del"
  - n What happens after getting the password?
    - n Exit => alarm the user, freeze, login the user

Network Security Authentication Protocols

---

---

---

---

---

---

---

---



## Initial Password Distribution

- n Physical contact:
  - n How: go to the system admin, show proof of identity, and set password
  - n Drawback: inconvenient, security treats when giving the user access to the system admin session to set the password
- n Choose a random strong initial password (pre-expired password) that can only be used for the first connection

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---



## Authentication Tokens

- n Authentication through what you have:
  - n Primitive forms: credit cards, physical key
  - n Smartcards: embedded CPU (tamper proof)
    - n PIN protected memory card:
      - n Locks itself after few wrong trials
    - n Cryptographic challenge/response cards
      - n Crypto key inside the card and not revealed even if given the PIN
      - n PIN authenticates the user (to the card), the reader authenticates the card
    - n Cryptographic calculator
      - n Similar to the previous card but has a display (or speaker)

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---



## Address-Based Authentication

- n Trust network address information
- n Access right is based on *users@address*
- n Techniques:
  - n Equivalent machines: smith@machine1 = john@machine2
  - n Mappings: <address, remote username, local username>
- n Examples:
  - n Unix: */etc/host.equiv*, and *.rhost* files
  - n VMS: centrally managed proxy database for each <computer, account> => file permissions
- n Threats:
  - n Breaking into an account on one machine leads to breaking into other machines accounts
  - n Network address impersonation can be easy in some cases. How?

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---

## Cryptographic Authentication Protocols

- Advantages:
  - Much more secure than previously mentioned authentication techniques
- Techniques:
  - Secret key cryptography, public key crypto, encryption, hashing, etc.

Network Security Authentication Protocols

---

---

---

---

---

---

---

---

## Other Types of Human Authentication

- Physical Access
- Biometrics:
  - Retinal scanner
  - Fingerprint readers
  - Face recognition
  - Iris scanner
  - Handprint readers
  - Voiceprints
  - Keystroke timing
  - Signature

Network Security Authentication Protocols

---

---

---

---

---

---

---

---

## Passwords as Crypto Keys

- Symmetric key systems:
  - Hash the password to derive a 56/64/128 bits key
- Public key systems:
  - Difficult to generate an RSA private key from a password
  - Jeff Schiller proposal:
    - Password => seed for random number generator
    - Optimized by requesting the user to remember two numbers
      - E.g. (857, 533):  $p$  prime number was found after 857 trials, and  $q$  after 533 trials
    - Known public key makes it sensitive to offline attacks
  - Usual solution:
    - Encrypt the private key with the users password and store the encrypted result (e.g., using a directory service)

Network Security Authentication Protocols

---

---

---

---

---

---

---

---

## Eavesdropping & Server Database Reading

- Example of basic authentication using public keys:
  - Bob challenges Alice to decrypt a message encrypted with its public key
- If public key crypto is not available protection against **both** eavesdropping and server database reading is difficult:
  - Hash => subject to eavesdropping
  - Challenge requires Bob to store Alice's secret in a database
- One solution:
  - Lamport's scheme allows a finite number of authentications

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---

## Key Distribution Center

- Solve the scalability problem of a set of  $n$  nodes using secret key
  - $n*(n-1)/2$  keys
- New nodes are configured with a key to the KDC
  - e.g.,  $K_A$  for node  $A$
- If node  $A$  wants to communicate with node  $B$ 
  - $A$  sends a request to the KDC
  - The KDC securely sends to  $A$ :  $E_{K_A}(R_{AB})$  and  $E_{K_B}(R_{AB}, A)$
- Advantage:
  - Single location for updates, single key to be remembered
- Drawbacks:
  - If the KDC is compromised!
  - Single point of failure/performance bottleneck => multiple KDC?

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---

## Multiple Trusted Intermediaries

- Problem:
  - Difficult to find a single entity that everybody trusts
- Solution: Divide the world into domains
  - Multiple KDC domains interconnected through shared keys
  - Multiple CA domains: certificates hierarchy

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---



## Certification Authorities

- How do you know the public key of a node?
- Typical solution:
  - Use a trusted node as a certification authority (CA)
  - The CA generates certificates: Signed(A, public-key, validity information)
  - Everybody needs to know the CA public key
  - Certificates can be stored in a directory service or exchanged during the authentication process
- Advantages:
  - The CA doesn't have to be online => more physical protection
  - Not a performance bottleneck, not a single point of failure
  - Certificates are not security sensitive: only threat is DoS
  - A compromised CA cannot decrypt conversation but can lead to impersonation
  - A certification hierarchy can be used: e.g., X.509

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---



## Certificate Revocation

- What if:
  - Employer left/fired
  - Private key is compromised
- Solution: similar to credit cards
  - Validity time interval
  - Use a Certificate Revocation List (CRL): X.509
    - For example: lists all revoked and unexpired certificates

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---



## Session Key Establishment

- Authentication is not everything
  - What could happen after authentication?
    - E.g., connection hijacking, message modification, replay, etc.
  - Solution use crypto => need a share key between communicating entities because public encryption/decryption is expensive
  - Practically authentication leads to the establishment of a shared key for the session
    - A new key for each session:
      - the more data an attacker has on a key the easier to break
      - Replay between sessions
      - Give a relatively "untrusted" software the session key but not the long-term key
      - Good authentication protocol can establish session keys that provide forward secrecy

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---



## Delegation

- n Give a limited right to some third entity:
  - n Example: printserver to access your files, batch process
- n How?
  - n Give your password?
  - n ACL
  - n Delegation

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---



## Security Handshake Pitfalls

- n Developing a new encryption algorithm is believed to be an "art" and not a "science"
- n Security protocols build on top of these algorithms and have to be developed into various types of systems
- n Several Cryptographic Authentication Protocols exist however:
  - n Several protocols were proven to have flaws
  - n Minor modifications may lead to flaws
  - n Use in a different context may uncover flaws or transform a non-serious flaw into a serious one

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---



## Login Only: Shared Secrets

- n Sending the password on the clear is not safe: use shared secrets
  - n Challenge response:  $B$  sends  $R$  and  $A$  has to reply  $f(K_{AB}, R)$ . Weaknesses:
    - n Authentication is not mutual
    - n If the subsequent communication is not protected: hijacking treat
    - n Offline attack by an eavesdropper using  $R$  and  $f(K_{AB}, R)$
    - n An attacker who successfully reads  $B$ 's database can impersonate  $A$ 
      - n Cascade effect if the same password is used on multiple servers
  - n Variants:
    - n  $B$  sends:  $K_{AB}, R$ , and  $A$  replies  $R$ 
      - n Requires reversible cryptography which may be limited by export legislation
      - n Dictionary attacks if  $R$  is a recognizable value (padded 32 bits) don't need eavesdropping
    - n  $A$  sends  $K_{AB}, \{timestamp\}$  (a single message)
      - n Requires: clock synchronization
      - n Problems with impersonation:
        - n within the clock skew: remember timestamp
        - n at another server: include  $B$  in message

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---



## Login Only: One-Way Public Key

- n Shared secrets are vulnerable if  $B$ 's database is compromised
- n Public key protocols:
  - n  $A$  send the signature of  $R$  using its public key:  $[R]_A$
  - n Advantage:
    - n  $B$ 's database is no longer security sensitive to unauthorized disclosure
  - n Variant:  $B$  sends  $\{R\}_{public-A}$   $A$  has to recover  $R$  and send it back
  - n Problem:
    - n You can trick  $A$  into signing a message or decrypting a message
  - n General solution: never use the same key for two purposes

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---



## Mutual Authentication: Shared Secret

- n Basic protocol: 5 messages,
- n Optimized into 3 rounds but becomes subject to the Reflection attack:
  - n  $C$  impersonates  $A$  by initiating two sessions to  $B$  [both single/multiple servers]
- n Solutions:
  - n Use different keys for  $A \rightarrow B$  authentication and  $B \rightarrow A$  authentication
    - n For example:  $K_{a,b} = K_{b,a} + 1$
  - n Use different challenges:
    - n For example: challenge from the initiator be an odd number, while challenge from the responder be an even number, concatenate the name of the challenge creator to the challenge
- n Another problem: password guessing without eavesdropping
- n Solution: 4 messages protocol where the initiator proves its identity first
- n Alternative two messages protocol using *timestamp* and *timestamp+1* for  $R_1$  and  $R_2$

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---



## Mutual Authentication: Public Keys

- n Three messages protocol:
  - n  $A \rightarrow B: A, \{R_2\}_B$
  - n  $B \rightarrow A: R_2, \{R_1\}_A$
  - n  $A \rightarrow B: R_1$
- n Problems:
  - n Knowing the public keys
- n Solutions:
  - n Store Bob's public key encrypted with Alice's password in some directory
  - n Store a certificate of Bob's public key signed by Alice's private key

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---

## Integrity/Encryption for Data

- Key establishment during authentication
- Use  $f(K_{A-B}, R)$  as the session key where  $R$  is made out of  $R_1$  and  $R_2$ 
  - Example:  $f(K_{A-B}) = K_{A-B} + 1$
  - Why not use  $K_{A-B}(R+1)$  instead of  $f(K_{A-B})$ ?
- Rules for the session key:
  - Different for each session
  - Unguessable by an eavesdropper
  - Not  $K_{A-B}(X)$

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---

## Two-Way Public Key Based Authentication + Key Setup

- First attempt:
  - $A$  sends a random number encrypted with the public key of  $B$
  - Flaw:  $T$  can hijack the connection using her own  $R$
- Second attempt:
  - $A$  sends  $\{R\}_{B,A}$ : encrypt using public key of  $B$  and then private key of  $A$
  - If someone records the conversation and then gets access to  $B$  key it can recover  $R$
- Third attempt:
  - Both  $A$  and  $B$  participate through  $R_1$  and  $R_2$  shares: session key  $R_1 \oplus R_2$
- Fourth alternative:
  - Use Diffie-Hellman key establishment protocol and each entity signs its contribution

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---

## One-Way Public Key Based Authentication

- Context:
  - Only one of the parties has a public key (e.g., SSL server)
  - First the server is authenticated
  - If needed the user is authenticated (e.g., using a password)
- First solution:
  - $A$  sends a random number encrypted with  $B$ 's public key
  - The random number is used as a session key
  - Problem: if an attacker records the communication and later on breaks into  $A$  it can decode the whole communication
- Second solution:
  - Use Diffie-Hellman with  $B$  signing his contribution

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---



## Privacy and Integrity

- Privacy:
  - Use a secret key algorithm to encrypt the data
- Integrity:
  - Generate a Message Authentication Code (MAC)
- No clean solution for merged privacy and integrity:
  - Use two keys (may be one derived from the other)
  - Use a weak checksum then encrypt
- Replays:
  - Use sequence number to avoid replays, or
  - Include info about previous message
- Reflection: replay the message in a different direction
  - Different range for each direction
  - Use a direction bit
  - Use a direction dependent integrity algorithm
- Key rollover: change keys periodically during the communication

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---



## Needham-Schroeder Authentication 1978

- Basis for Kerberos and many other authentication protocols
- Uses *NONCE* (Number ONCE):
  1.  $A \rightarrow KDC: N_1, A, B$
  2.  $KDC \rightarrow A: K_A\{N_1, B, K_{AB}, \text{ticket-to-B}\}; \text{ticket-to-B} = K_B\{K_{AB}, A\}$
  3.  $A \rightarrow B: \text{ticket-to-B}, K_{AB}\{N_2\}$
  4.  $B \rightarrow A: K_{AB}\{N_2-1, N_3\}$
  5.  $A \rightarrow B: K_{AB}\{N_3-1\}$
- Why  $N_1$ ?  $T$  has stolen the old key of  $B$  and previous request from  $A$  to KDC requesting to communicate with  $B$
- Why  $B$  in second message?
- Reflection attack?

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---



## Expanded Needham-Schroeder

- Vulnerability of basic protocol:
  - $T$  steals  $A$ 's key and can impersonate  $A$  even after  $A$  changes it's key (ticket stays valid)
- Proposed solution [Need87]
  - Before talking to the  $KDC$   $B$  gives  $A$  a nonce that has to be included in the ticket => 7 messages protocol

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---



## Otway-Rees Authentication 1987

1.  $A \rightarrow B: N_C, A, B, K_A\{N_A, N_C, A, B\}$
2.  $B \rightarrow KDC: K_A\{N_A, N_C, A, B\}, K_B\{N_B, N_C, A, B\}$
3.  $KDC \rightarrow B: N_C, K_A\{N_A, K_{AB}\}, K_B\{N_B, K_{AB}\}$
4.  $B \rightarrow A: K_A\{N_A, K_{AB}\}$
5.  $A \rightarrow B: K_{AB}\{anything\ recognizable\}$

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---



## Nonces

- n Potential properties:
  - n Non-repeated, unpredictable, time dependent
  - n Context dependent
- n A nonce may have to be unpredictable for some challenge response protocols (with no session key establishment)
  - n Sequence number doesn't work for challenge response:  $K_{AB}\{R\}$
- n One solution is to use cryptographic random number generators

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---



## Random Numbers

- n If the random number generation process is weak the whole security system can be broken
- n Pure randomness is very difficult to define
- n Usually we differentiate:
  - n *Random*: specialized hardware (e.g., radioactive particle counter)
  - n *Pseudorandom*: a deterministic process determined by its initial state
    - n For testing purpose: hashing a seed using a good hashing function can work
    - n For security purpose: long seed, good hashing function

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---



## Performance Considerations

- n Metrics:
  - o Number of cryptographic operations using a private key
  - o Number of cryptographic operations using a public key
  - o Number of bytes encrypted/decrypted using a secret key
  - o Number of bytes to be cryptographically hashed
  - o Number of messages transmitted
- n Notes:
  - o Private key operations are usually much more expensive than public key operations
- n Some optimization techniques:
  - o Caching information such as tickets

Network Security      Authentication Protocols

---

---

---

---

---

---

---

---



## Authentication Protocols Checklist

- n Eavesdrop:
  - o Learn the content, learn info to impersonate  $A/B$  later or to another replica, offline password guessing
- n Initiating a conversation pretending to be  $A$ :
  - o Impersonate  $A$ , offline password guessing, delayed impersonation, trick  $B$  to sign/decrypt messages
- n Lie in wait at  $B$ 's network address and accept connections from  $A$ :
  - o Immediate/delayed impersonation of  $B$  or  $A$ , offline password guessing, trick  $A$  to sign/decrypt messages
- n Read  $A/B$ 's database:
- n Sit actively/passively on the net between  $A$  and  $B$  (router):
  - o Offline password guessing, learn the content of messages, hijack connections, modify/rearrange/replay/reverse direction of message
- n Combinations:
  - o Even after reading both  $A$  and  $B$  databases  $T$  shouldn't be able to decrypt recorded conversations
  - o Even after reading  $B$ 's database and eavesdropping on an authentication exchange it shouldn't be possible to impersonate  $A$  to  $B$

Network Security      Authentication Protocols

---

---

---

---

---

---

---

---



## STRONG PASSWORD PROTOCOLS

Network Security      Authentication Protocols

---

---

---

---

---

---

---

---



## Context & Solutions

- Context:
  - $A$  wants to use any workstation to log into a server  $B$
  - $A$  has only a password
  - The workstation doesn't have any user-specific information (e.g., users' trusted CAs, or private keys)
  - The software on the workstation is trustworthy
- Potential solutions:
  - Transmit the password in the clear
  - Use Diffie-Hellman key establishment (vulnerable to  $B$  impersonation)
  - Use SSL (relies on trust anchors: trusts configuration and certificates)
  - Challenge response authentication using a hash of the password as a key (vulnerable to dictionary attacks)
  - Use Lamport's hash or S/KEY
  - Use a strong password protocol (secure even if the shared secret could be broken by an offline dictionary attack)

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---

---

---

## Lamport's Hash: One Time Password



- Allows authentication
  - Resistant to eavesdropping and reading Bob's database
  - Doesn't use public key cryptography
- $B$ 's database:
  - Username (e.g.,  $A$ ),
  - $n$  (integer decremented at each authentication)
  - $hash^n(password)$
- Initialization:
  - Set  $n$  to a reasonably large number (e.g., 1000)
  - The user registration software computes:  $x_n = hash^n(password)$  and sends  $x_n$  and  $n$  to  $B$

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---

---

---

## Lamport's Hash (Cont'd)



- Authentication:
  - $A$  connects to a workstation and gives her username and password
  - The workstation sends  $A$ 's username to  $B$
  - $B$  sends back  $n$
  - The workstation computes  $hash^{n-1}(password)$  and sends it to  $B$
  - $B$  computes the hash of the received value and compares it with the stored value of  $hash^n(password)$
  - If equal: decrement  $n$  and store the last received value
  - When  $n$  gets to 1,  $A$  needs to reset its password (in a secure way)
- Enhancement: *Salt*
  - $x_n = hash^n(password | salt)$
  - Advantage:
    - Use the same password on multiple servers
    - Makes dictionary attacks harder (similar to Unix)
    - Do not have to change the password when  $n$  reaches 1 (just change the salt)

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---

---

---

## Pros and Cons

- Advantages:
  - Not sensitive to eavesdropping, or reading  $B$ 's database
- Disadvantages:
  - Limited number of logins
  - No mutual authentication, difficulty to establish a common key, or prevent man-in-the-middle
    - One can use this scheme followed by a Diffie-Hellman key establishment: but this is vulnerable to connection hijacking
  - Small  $n$  attack:
    - $T$  impersonates  $B$ 's address and sends back a small value of  $n$  (e.g., 50)
    - If the real value of  $n$  at  $B$  is 100 =>  $T$  can impersonate  $A$  50 times
- Use in the "human and paper" environment:
  - Print the list and give it to  $A$  (the user won't go back on the list)
  - Use 64 bits out of 128 MD5 hash function
  - What if you lose the list!
- Deployed in S/Key (Phil Karn) RFC 1938

Network Security Authentication Protocols

---

---

---

---

---

---

---

---

## Strong Password Protocols

- Goal:
  - Prevent off-line attacks
  - Even if eavesdropping or impersonating addresses
- Basic Form: Encrypted Key Exchange (EKE) [Bellovin & Merritt]
  - $A$  and  $B$  share a weak secret  $W$  (derived from  $A$ 's password)
  - $A$  and  $B$  encrypt their DH contributions using  $W$
  - Why is it secure? because  $W \{ g^r \text{ mod } p \}$  is just a random number and for any password  $W$  their could exist a  $r = g^r$  such that  $W \{ r \}$
- Variants:
  - Simple Password Exponential Key Exchange (SPEKE): use  $g = W$
  - Password Derived Moduli (PDM): Use  $p = f(W)$

Network Security Authentication Protocols

---

---

---

---

---

---

---

---

## Subtle Details

- A simple implementation may lead to flaws
- EKE:
  - If  $p$  is a little more than a power of 2
  - $g^r$  has to be less than  $p$
  - The attacker can try a password and if  $GUESS \{ W \{ g^r \text{ mod } p \} \}$  is higher than  $p$  then discard  $guess$
  - A password from a space of 50'000 can be guessed after about 20 exchanges
  - Solution?
- SPEKE:
  - Small problem if  $W$  is not a perfect square mod  $p$

Network Security Authentication Protocols

---

---

---

---

---

---

---

---

## Augmented Strong Password Protocol

- Goal:
  - If an attacker steals  $B$ 's database but doesn't succeed with an offline attack he cannot impersonate  $A$
- How:
  - avoid storing  $W$  in  $B$ 's database but only something derived from  $W$
- Augmented PDM:
  - $B$  stores " $A$ ",  $p$ ,  $2^W \bmod p$
  - $A$  sends  $2^a \bmod p$
  - $B$  sends:  $2^b \bmod p$ ,  $\text{hash}(2^{ab} \bmod p, 2^{bW} \bmod p)$
  - $A$  sends  $\text{hash}(2^{ab} \bmod p, 2^{bW} \bmod p)$

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---

## Augmented Strong Password Protocol

- RSA variant:
  - $B$  stores: " $A$ ",  $W$ ,  $A$ 's public key,  $Y = W \{A$ 's private key}
  - $A$  sends:  $A$ ,  $W \{g^a \bmod p\}$
  - $B$  sends:  $W \{g^b \bmod p\}$ ,  $(g^{ab} \bmod p) \{Y\}$ ,  $c$
  - $A$  replies:  $[\text{hash}(g^{ab} \bmod p, c)]_{\text{sign-}A}$

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---

## Secure Remote Protocol (SRP)

- Invented by Tom Wu 1998, RFC2945
  - $B$  stores  $g^W \bmod p$
  - $A$  choose  $a$  and sends: " $A$ ",  $g^a \bmod p$
  - $B$  choose  $b$ ,  $c_1$ , 32-bit number  $u$ , and sends  $g^b + g^u \bmod p$ ,  $u$ ,  $c_1$
  - => Share key is:  $K = g^{k(a+uW)} \bmod p$
  - $A$  sends:  $K\{c_1\}$ ,  $c_2$
  - $B$  sends:  $K\{c_2\}$
- How is the common key computed on both ends?

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---



## Credentials Download Protocols

- n Goal:
  - n  $A$  can only remember a short password
  - n When using a workstation  $A$  needs its environment (user specific information)
  - n The user specific information could be downloaded from a directory if  $A$  knew its private key
  - n Strong Password protocols can help
- n Protocol based on EKE:
  - n  $B$  stores: " $A$ ",  $W$ ,  $Y = W\{A\text{'s public key}\}$
  - n  $A$  sends: " $A$ ",  $W\{g^a \text{ mod } p\}$
  - n  $B$  sends:  $g^b \text{ mod } p$ ,  $(g^{ab} \text{ mod } p)\{Y\}$

Network Security

Authentication Protocols

---

---

---

---

---

---

---

---