Security Protocols

Why do we need security protocols?
What are typical attacks on protocols?
How to build a security protocol?
Building infrastructure for security protocols
What is the best security protocol?

Mark Vella
Sources: (Anderson, 2008) (Bishop, 2003)
(Menezes et al., 1997) (Stamp, 2011)
Contents

- Introduction
- Security Protocol Notation
- Protocol Attacks
- Authentication Protocols
- Timestamps
- Key Management
- Zero Knowledge proofs
- Designing the best authentication protocol
- Summary
Introduction
Protocols

- Protocols are the rules that must be followed in a particular interaction
  - E.g. Asking a question in class
  - E.g. Diplomatic protocols

- Network Protocols
  - Rules followed in networked communications
  - E.g. TCP, HTTP
Security Protocols (1)

- Security Protocols are the communication rules followed in security-critical applications
  - These are typically designed so that systems survive malicious attacks and can guarantee security requirements

- Evaluating security protocols
  - Is the threat model realistic?
  - Does the protocol deal with it?
Security Protocols (2)

- Ideal properties
  - Meet specified security requirements
  - Be efficient
    - Computational cost, bandwidth usage and delays
  - Not too fragile
    - Continues to work even during an attack attempt
    - Continues to work when environment changes
Security Protocols (3)

- Simple security protocols
  - Swiping a badge through a reader when entering a building

- Complex security protocols
  - Protocols governing Bank ATMs
    - Machine – customer interaction
    - Machine – bank interaction
    - Machine – network interaction
    - Alarm transmission
    - Encryption key management
Security Protocol Flaws

- Security flaws can be very subtle
  - E.g. ATMs that verified PIN off-line
- IPsec and GSM have serious security flaws
  - Even if a security protocol is not flawed, a particular implementation may be
Real world security protocols

- SSL
  - ECommerce
- IPSec
  - VPNs, IPv6
  - Application layer transparent (found in the Network layer of the OSI network model)
- Kerberos
  - Windows Authentication
Security Protocol Notation
Security Protocol Notation (1)

A -> B : \{X,Y\}_K

Principles : Message

- A -> B – A send to B
- \{X,Y\}_K – Message consisting of X and Y, encrypted using key K
Security Protocol Notation (2)

\[ M = [C]_k \]

- Plaintext \( M \) is recovered by decrypting ciphertext \( X \) using key \( K \)

\[ C = h(M) \]

- Ciphertext \( C \) is the hash value of the hash function \( h \) applied to plaintext \( M \)
Security Protocol Notation (3)

\[ A \rightarrow B : \{X,Y\}_{K_B} \]

- \( K_B \) - B’s public key

\[ A \rightarrow B : \{X,Y\}_{K_{AB}} \]

- \( K_{AB} \) - Symmetric key shared by A and B
Security Protocol Notation (4)

\[ S = [M]_k \]

- \( S \) constitutes of a signed message \( M \) using private key \( K \)
Protocol Attacks
Replay Attacks

- Password eavesdropping in authentication protocols
  - Garage doors remote openers using pre-configured passwords
- Grabber devices (early 90s)
  - Record encrypted password signal
  - Trudy can open door by replaying grabbed signal
- Actual countermeasures
  - Increased password to 32 bits
  - Separate code for lock and unlock
  - Both ineffective – attack was not on the encryption key!
Challenge Response Protocols

- Countermeasure for replay attacks
- Guarantees ‘freshness’ of message i.e. not replayed
- Example: Car Ignition
  
  \[ E \rightarrow T : N \]
  
  \[ T \rightarrow E : \{T,N\}_K \]

- E – Engine Ignition, T – Key Transponder
- N – Nonce (Number used once)
- K – shared key
Random Number generation

- If the next nonce is predictable, an attacker can challenge the key in the owner’s pocket and replay the response
- **This constitutes the primary attack on challenge response based protocols**
- A random factor must be included in the generation mechanism
  - Network traffic, hard disk rotational velocity
- Nonce must provide the required ‘**freshness**’ to protect against replay attacks
Challenge Response Case Study (1)

Internet Banking

- User request a transfer of funds – a highly sensitive operation
- Server sends a challenge to the user (e.g. an 8 digit number)
- User enters PIN and challenge in a password generator with an in-built shared secret key
- Generator returns the encrypted response in the form of a digit sequence
- Digit sequence sent to server via website form field
Challenge Response Case Study (2)

- Protocol formal specification

\begin{align*}
S & \rightarrow U : N \\
U & \rightarrow P : PIN, N \\
P & \rightarrow U : \{N\}_{K_{SP}} \\
U & \rightarrow S : \{N\}_{K_{SP}}
\end{align*}
Challenge Response Case Study (3)

- One time password schemes
  - C -> S: C, OTP₁
  - C <-> S: session
  - C -> S: C, OTP₂ ...

- Lamport’s OTP scheme (S/Key)
  - h(k) = k₁, h(k₁)=k₂, ..., h(kₙ₋₁)=kₙ
  - p₁ = kₙ, p₂ = kₙ₋₁, ..., pₙ₋₁=k₂, pₙ=k₁
  - The challenge is the sequence number
  - The seed key is retained for further OTP list generation sessions
Man-in-the-middle

- Originated from an alleged attack on an identify-friend-or-foe (IFF) system in anti-aircraft gunners
MITM attacks in networks

- ARP/DNS cache poisoning
  - Malicious update of routing/address resolution tables in order to route message to the attacker host instead of the intended recipient
  - Attacker then forwards message to intended recipient

- Attacks on public keys
  - Trudy issuing a public key in name of Bank.com
  - Alice encrypts sensitive information with Trudy’s public key thinking its Bank.com’s key

- Countermeasures
  - Mutual Authentication combined with secure session key establishment
Reflection Attacks (1)

- Occurs in mutual authentication scenarios
  - 2 principles need to authenticate each other (fighter and a bomber)
  - Suppose the air force simply installed the IFF in each of the two

\[
\begin{align*}
F & \rightarrow B : N \\
B & \rightarrow F : N \\
F & \rightarrow B : \{N\}^{K_{FB}} \\
B & \rightarrow F : \{N\}^{K_{FB}}
\end{align*}
\]

- The bomber can simply reflect back the fighter’s messages by starting its own authentication request using the same nonce
- Authentication succeed without B’s knowledge of $K_{FB}$
Reflection Attacks (2)

- Including the name of the 2 principles in the authentication exchange can serve as a countermeasure

\[ F \rightarrow B : N \]

\[ B \rightarrow F : \{B, N\}_K_{FB} \]

- A reflection attack would only yield

\[ B \rightarrow F : \{F, N\}_K_{FB} \]
Chosen protocol attacks

- Exploits the `same authentication mechanism within multiple different applications’ scenario

Diagram:
- Customer
  - Picture 143!
  - Prove your age by signing ‘X’
  - $\text{sig}_K \{X\}$
- Mafia porn site
  - $\text{sig}_K \{X\}$
- Bank
  - Buy 10 gold coins
  - Sign ‘X’
Changing environment attacks (1)

☐ Utrecht scam - Holland 1993/4
☐ Epidemic of phantom withdrawals
☐ Victims had used their card at a certain filling station near Utrecht
☐ Employee had tapped the line from the card reader to the controlling PC
☐ Recorded magnetic strip details
☐ Details utilized for cash withdrawal
Changing environment attacks (2)

- Magnetic strip protocol designers made two assumptions in the early 80s
  1. Magnetic strip content was not secret (card no, version, expiry) - strip data not encrypted
  2. Cards would operate only in trustworthy environments such as ATMs – PIN encryption was a function of the ATM

- Environment Change
  1. Clear authentication code (PIN) placed on magnetic strip due to card forgery – Assumption: ATM will encrypt it anyways
  2. Cards used in other environments other than ATMs (shop terminals that did not perform any encryption – since PIN was not involved back then)
Changing environment attacks (3)

- **Assumptions** are made since if protocols had to take into each account all possible threats and operational scenarios, protocol implementation would be prohibitive
  - *Ideally* – protocols resist environment change
  - *Realistically* – environment assumptions should be continuously monitored
Authentication Protocols
Authentication Protocols

- Protocols for performing Authentication and Mutual Authentication over a network
- Application of ciphers
  - Provision of Confidentiality and Integrity
- Resist protocol attacks
  - Replay, MITM, Reflection, Chosen Protocol
Simple Authentication Protocol (1)

A -> B : Username
B -> A : ProofRequired
A -> B : Password
Simple Authentication Protocol (2)

- Vulnerable to Replay Attacks

T -> B : Username
B -> T : ProofRequired
T -> B : Password

- Password is sent as plaintext

  Trudy can try this password anywhere else Alice may have reused the password
Simple Authentication Protocol (3)

- Bob must know Alice’s password
  - Vulnerable against password theft
- Inefficient
  - Same result can be achieved in a single message
- Not able to provide mutual authentication
Simple Authentication Protocol with a Hash (1)

A -> B : Username
B -> A : ProofRequired
A -> B : h(Password)
Simple Authentication Protocol with a Hash (2)

- Protects against
  - Disclosure of password either by observation or password theft
  - Password is sent as a validator
  - Bob needs only to store the hash of the password

- Vulnerable
  - Replay attacks
  - Recording the hashed password is enough
A Challenge Response Protocol (1)

A -> B : Username
B -> A : Nonce
A -> B : h(Password, Nonce)
A Challenge Response Protocol (2)

- Vulnerability
  - Bob must know Alice’s password
  - Storing the hashed password is not enough since the hash will include a unique nonce on each authentication request
Symmetric Key Authentication Protocols

- Employing symmetric key cryptography in security protocols
  - Concerned with protocol attacks not cipher attacks
  - Assume that safe ciphers are used
    - Exhaustive key search, no known clever attacks
First Attempt

A -> B : Username
B -> A : Nonce
A -> B : \{Nonce\}_{K_{AB}}

- Prevents Replay Attacks
- A shared symmetric key is more secure than a password
- **Lacks mutual authentication**
  - Only A is authenticated
  - T can pose as B, stealing sensitive information from A
Mutual Authentication (1)

A -> B : Username, Nonce
B -> A : \{Nonce\}_{K_{AB}}
A -> B : \{Nonce\}_{K_{AB}}

- Last message is clearly a repetition of the preceding one
- Vulnerable to a reflection attack
Mutual Authentication (2)

A -> B : Username, Nonce_A
B -> A : Nonce_B, {Nonce_A}^_{K_{AB}}
A -> B : {Nonce_B}^_{K_{AB}}

- Repeating the protocol in the first attempt twice

- *Still vulnerable to a reflection attack?*
Mutual Authentication (3)

1. T -> B : Username_A, Nonce_A
2. B -> T : Nonce_B, $\{\text{Nonce}_A\}_{K_{AB}}$
5. T -> B : $\{\text{Nonce}_B\}_{K_{AB}}$

3. T -> B : Username_A, Nonce_B
4. B -> T : Nonce_B2, $\{\text{Nonce}_B\}_{K_{AB}}$

☐ yes – through an interleaving attack
Mutual Authentication (4)

- One way authentication protocols may not be secure for mutual authentication – *it’s a bad idea to have two sides in a protocol do exactly the same thing*
- “*Obvious*” changes to protocols can raise serious security issues
- Attacks on protocols are subtle
Mutual Authentication (5)

\[\begin{align*}
A & \rightarrow B : \text{User}_A, \text{Nonce}_A \\
B & \rightarrow A : \text{Nonce}_B, \{\text{User}_B, \text{Nonce}_A\}^{K_{AB}} \\
A & \rightarrow B : \{\text{User}_A, \text{Nonce}_B\}^{K_{AB}}
\end{align*}\]

- Sending the name along the nonce
- Protects against reflection attacks
- $T$ can never use a response from $B$, for the third message due to different name
Public Key Encryption Authentication (1)

A -> B : Username
B -> A : \{Nonce\}_{K_{A}}
A -> B : Nonce

- B authenticates A
- Only A can compute the private key operation in order to reply with the required Nonce
Public Key Encryption Authentication (2)

☐ The same pair of keys should not be used both for encryption and authentication

☐ Say T gets hold of $C = \{M\}_{K_A}$

  A -> T : Username
  T -> A : $\{M\}_{K_A}$
  A -> T : M

☐ T can successfully recover M by spoofing B!
(if A uses the same public key for encryption and authentication)
Public Key Encryption Authentication (3)

- Authentication using digital signatures (private key encryption)
  
  \[ A \rightarrow B : \text{Username} \]
  
  \[ B \rightarrow A : \text{Nonce} \]
  
  \[ A \rightarrow B : [\text{Nonce}]_{K_A} \]

- Only A can compute the required private key operation
Session Keys (1)

- Using a separate session key to encrypt data within each connection
- Limiting the data encrypted with one particular key – limiting the damage if one session key is compromised
- **Goal:** Design an authentication protocol that also allows us to securely establish a shared symmetric key when the connection is initiated besides just mutual authentication
  - *Basic secure connection handshaking*
Session Keys (2)

- Attempt 1 – *Public Key Encryption*

\[
\begin{align*}
A \rightarrow B & : \text{Username, Nonce} \\
B \rightarrow A & : \{\text{Nonce}, K_S\}_{K_A} \\
A \rightarrow B & : \{\text{Nonce+1}, K_S\}_{K_B}
\end{align*}
\]

- No mutual authentication
  - A cannot authenticate B
Session Keys (3)

- Attempt 2 – Digital Signatures

A -> B : Username, Nonce
B -> A : [Nonce, K_S]_{K_B}
A -> B : [Nonce+1, K_S]_{K_A}

- Provides mutual authentication
- Flaw – Anybody can find out K_S by using the corresponding public key!
Session Keys (4)

- Sign and Encrypt

\[
\begin{align*}
A \rightarrow B &: \text{Username, Nonce} \\
B \rightarrow A &: \{[\text{Nonce}, K_S]_{K_B}\}_{K_A} \\
A \rightarrow B &: \{[\text{Nonce+1}, K_S]_{K_A}\}_{K_B}
\end{align*}
\]

- Mutual authentication and secure session key establishment is permitted
- Session key no longer public
Session Keys (5)

- Encrypt and Sign

\[ A \rightarrow B : \text{Username, Nonce} \]
\[ B \rightarrow A : [\{\text{Nonce, } K_s\}_{K_B}]_{K_A} \]
\[ A \rightarrow B : [\{\text{Nonce+1, } K_s\}_{K_B}]_{K_A} \]

- Mutual authentication and secure session key establishment is permitted also in an ‘Encrypt and Sign’ fashion

- Any MITM not attacking the protocol per se is reduced to simple brokerage of encrypted traffic
Perfect Forward Secrecy (PFS) (1)

- **Desired property for session key establishment protocols**
- **Delayed Data Transfer Attacks**
  - Preventing compromised long term keys (used during session key establishment) from enabling the decryption of recorded ciphertext encrypted with session keys
  - Trudy is able to decipher all recorded ciphertexts in case long term keys used by Alice and Bob are compromised
Perfect Forward Secrecy (PFS) (2)

- Session Keys must be ‘forgotten’ once used
- PFS – Finding a way for Alice and Bob to agree on a session keys $K_S$ using a long-term (a)symmetric key $K_{AB}$ whose disclosure doesn’t compromise all past session keys
- But can’t all session keys be simply just deleted and thus ‘forgotten’?
A Delayed Data Transfer Attack

A <-> B : \{K_S\}_{K_{AB}}
B <-> A : \{M\}_{K_S}

- **Fails** to provide PFS
- T can access both \{K_S\}_{K_{AB}} and \{M\}_{K_S}
- If long-term key $K_{AB}$ is compromised, all recorded messages can be deciphered
Diffie-Hellman Protocol (DH) (1)

- DH is a key exchange protocol
  - Addressing symmetric key encryption main pitfall – establishing a shared secret key
  - Relies on the computational difficulty of the discrete logarithm problem
    - While finding $k$ in $x = g^k$ is easily computed
    - Finding $k$ in $x = g^k \mod p$ is not
Diffie-Hellman Protocol (2)

A -> B : \( g^a \mod p \)
B -> A : \( g^b \mod p \)

- \( g,p \) are public
- \( p \) is prime
- \( A \) generates exponent \( a \)
- \( B \) generates exponent \( b \)
- \( A \) computes \((g^b)^a \mod p\)
- \( B \) computes \((g^a)^b \mod p\)
- Shared secret is \( g^{ab} \mod p \)
Diffie-Hellman Protocol (3)

- MIM attack weakness of DH

\[
\begin{align*}
A & \rightarrow T : g^a \mod p \\
T & \rightarrow A : g^t \mod p \\
B & \rightarrow T : g^b \mod p \\
T & \rightarrow B : g^t \mod p
\end{align*}
\]

- A,T - \( g^{at} \mod p \)
- B,T - \( g^{bt} \mod p \)
- A,B unaware
Ephemeral DH Protocol (1)

- Ephemeral (transitory) DH Protocol
- Enables PFS
  - A, B must forget their respective secret exponents
- Also protects against MITM attacks
  - A, B use the shared key $K_{AB}$ to encrypt the DH exchange
Ephemeral DH Protocol (2)

\[ A \rightarrow B : \{ g^a \mod p \}^{K_{AB}} \]
\[ B \rightarrow A : \{ g^b \mod p \}^{K_{AB}} \]

- **PFS achieved**
  - Finding a way for Alice and Bob to agree on a session key \( K_s \) using a long-term symmetric key \( K_{AB} \)

- **MITM protection**
  - \( T \) cannot establish an MITM attack in absence of \( K_{AB} \)
A complete authentication protocol

A -> B : Username, Nonce_A
B -> A : Nonce_B, [{Nonce_A, g^b mod p}]_{K_A}^B
A -> B : [{Nonce_B, g^a mod p}]_{K_B}^A

- A mutual authentication protocol that establishes a session key with PFS
  1. Diffie-Hellman provide a secure session key with PFS
  2. Nonces protect against replay attacks
  3. Signatures provide mutual authentication
  4. Public key encryption provides secure key exchange, avoids reflection and provides scalability

- Cipher over-employment leads to an inefficient solution
Security Protocol Efficiency

- Based on
  1. Reducing number of messages exchanged
  2. Reducing number of public key cryptographic operations
Timestamps
Timestamps (1)

- A timestamp $T$ contains the current time
- Replace nonce
  - A current timestamp can guarantee freshness
- Increased efficiency and security
  - Do not waste messages exchange nonce since both $A, B$ should know the current time
  - Removes random number generation concerns
- Concerns
  - Time becomes a *security critical* parameter
  - Perfect clock synchronization is impossible
  - Must allow for Clock Skew – opens a window for replay attacks
- Used in Kerberos authentication protocol (implemented in Windows NT authentication)
Timestamps (2)

- Authentication - Nonce version
  A -> B : Username
  B -> A : Nonce
  A -> B : \{Password, Nonce\}_{KB}

- Authentication - Timestamp version
  A -> B : Username, \{Password, T\}_{KB}
‘Sign and Encrypt’ with Timestamps

A -> B : $\text{Username}_A$, $\{[T, K_{AB}]_{K_A}\}_{K_B}$

B -> A : $\{[T+1, K_{AB}]_{K_B}\}_{K_A}$

- Equivalent result but eliminates Nonce generation process
'Encrypt and Sign' with Timestamps (1)

A -> B: Username_{A}, [{T, K_{AB}}_{K_B}]_{K_A}
B -> A: [{T+1, K_{AB}}_{K_A}]_{K_B}

☐ Is security affected?
■ This time yes!
Encrypt and Sign’ with Timestamps (2)

A -> B : Username_A, \[\{{T, K_{AB}}\}_{KB}\]_{KA}
B -> A : \[\{{T+1, K_{AB}}\}_{KA}\]_{KB}

- Vulnerable to following attack
  - T can recover \(\{{T, K_{AB}}\}_{KB}\) by using A’s public key
  - T is authenticated, but B does not know that the session key is the one to be shared with A!
  - \(K_{AB}\) Is available in the reply to decrypt!
  - Attack must occur in real time taking advantage of clock-skew

T -> B : Username_T, \[\{{T, K_{AB}}\}_{KB}\]_{KT}
B -> T : \[\{{T+1, K_{AB}}\}_{KT}\]_{KB}
Key Management
Interchange and Session Keys (1)

- **Interchange Key**
  - A cryptographic key associated with a principal of communication
  - Allow for the exchange of session keys
  - Do not change over time

- **Session Key**
  - A cryptographic key associated with the communication itself
  - Enciphers data only (not used for authentication)
  - Generated for each communication and discarded when the session ends
Session and Interchange Keys (2)

- **Session Key**
  1. Limits the amount of data enciphered by a single key
  2. Hinders effectiveness of replay attacks
  3. Avoids expensive asymmetric cipher operations
  4. Prevent forward search attacks
     - Scenarios where plaintext message space is small
     - Enciphered messages are compared with anticipated pre-computed ciphertext
       - E.g. Yes/No
       - E.g. Buy/Sell

- 72
Key Generation (1)

- Generating keys that are difficult to guess is critical
  - Only keys are secret; ciphers are not
- Random Key Generation
  - Given a set $K$ of potential keys, the probability of a key being guessed is at a minimum when the key is selected at random from the elements of $K$
  - Problem is equivalent to generating a random number between 0 and $|K|-1$
  - Typically many such keys are required – a sequence of random numbers is needed
Key Generation (2)

- A sequence of cryptographically strong random numbers
  - Is a sequence of numbers \( n_1, \ldots, n_k \) such that for any positive integer \( k \) an observer cannot predict \( n_{k+1} \) even if the number sequence up to \( n_k \) is known

- An h/w-based true random number generator requires a physical source of randomness
  - Background radiation, Disk latency
  - External devices may be tampered with
  - Generators built on oscillators or capacitors can be built on VLSI devices enclosed within tamper-resistant h/w
  - Software-based solutions provide a cheap alternative
Key Generation (3)

- Pseudo-random numbers
  - No statistical correlation between each number in the list, but deterministic on an initial seed
  - c’s \( \textit{rand(s)} \) or similar
    - A brute-force search on the 32-bit seed breaks the generator
- A \textit{linear} congruential generator
  - \( n_k = (a n_{k-1} + b) \mod n \)
  - \( n \) is the period of the sequence
  - \( a, b, n \) are relatively prime
  - Given some known \( n \)’s, \( a \) and \( b \) can be determined – breaking the generator
- A \textit{polynomial} congruential generator
  - \( n_k = (a_j n_{k-1}^j + ... + a_1 n_{k-1} + a_0) \mod n \)
  - Has also been broken
Key Generation (4)

- S/w-based true random number generators
  - Software-based random sources rather than physical sources
  - System clock, elapsed time between keystrokes, system and network statistics
  - Mixing functions are the most popular candidates for s/w-based TRNG’s
  - Pros – cryptographically strong
  - Cons – Each session key must be securely transferred as synchronized independent generation is not possible anymore (as per PRNG’s) since determinism is lost
Key Generation (5)

- Strong mixing functions
  - Non-linear functions with **unpredictable** and **irreproducible** input
  - **E.g.** \((\text{date; ps -e }) \mid \text{md5}\)
    - **2 inputs** – date; ps
    - **Unpredictable** – In UNIX the status of processes is highly variable
    - **Irreproducible** - An attacker is unlikely to reproduce the state at a future time
  - Non-linear function utilized for **de-skewing**
    - Eliminating bias and correlation possibly present in a s/w-based source of randomness
    - Hash functions or block ciphers
Cryptographically strong pseudo-random generation is possible after all
- One-way function based: $f(s), f(s+1), f(s+2), \ldots$
- $s$ is a (sufficiently large) secure secret key

Cryptographic Hash functions
- E.g. FIPS generator
  - $h(s), h(s+1), h(s+2), \ldots$
- E.g. Lamport’s OTP scheme
  - $h(k) = k_1, h(k_1) = k_2, \ldots, h(k_{n-1}) = k_n$
  - $\ldots, k_3, k_2, k_1$

Block ciphers
- E.g. ANSI generator
  - $\{s\}_k, \{s+1\}_k, \{s+2\}_k, \ldots$

Number theory-based
- E.g. Blum-Blum-Shub generator
  - $\ldots, s^8, s^4, s^2, s \mod N$
Key Exchange

- Goal of key exchange is to enable secret communication using a shared cryptographic key
  - E.g. Diffie-Hellman Protocol, Sign and Encrypt

- Key Exchange solution criteria
  - Shared key cannot be transmitted in clear - Enciphering the key/data exchange from which the key can be derived
  - Trusting a third party is possible for scale-up
  - Ciphers and protocols are public – Cryptographic keys are the only secret

- 1-Symmetric key and 2-Public key crypto use different protocols for key exchange
Symmetric crypto key exchange (1)

- Relying on a trusted 3rd party
  - (Key Distribution Centre - KDC)

  $A \rightarrow S : \{\text{Request } K_{AB}\}_{K_{AS}}$

  $S \rightarrow A : \{K_{AB}\}_{K_{AS}}, \{K_{AB}\}_{K_{BS}}$

  $A \rightarrow B : \{K_{AB}\}_{K_{BS}}$

  $A \leftrightarrow B : \{\text{Message}\}_{K_{AB}}$

- B does not authenticate A
- Replay attacks possible
Symmetric crypto key exchange (2)

- Possible Attack
  
  \[ A \rightarrow B : \{ \text{Message} \}_{K_{AB}} \]
  
  \[ T \rightarrow B : \{ K_{AB} \}_{K_{BS}} , \{ \text{Message} \}_{K_{AB}} \]

- \( T \) records
  
  - \( \{ K_{AB} \}_{K_{AS}} , \{ K_{AB} \}_{K_{BS}} \)
  
  - \( \{ \text{Message} \}_{K_{AB}} \)

- \( B \) cannot know that its \( T \) who sent the message
Symmetric crypto key exchange (3)

- Needham-Schroeder protocol
  
  \[
  \begin{align*}
  A &\rightarrow S : A, B, N_A \\
  S &\rightarrow A : \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}} \\
  A &\rightarrow B : \{K_{AB}, A\}_{K_{BS}} \\
  B &\rightarrow A : \{N_B\}_{K_{AB}} \\
  A &\rightarrow B : \{N_B + 1\}_{K_{AB}}
  \end{align*}
  \]

  - Protocol rests on assumption that \(S\) is trusted – \(A\) can trust that its talking to \(S\) and receives a session key to \(B\).
  - The identifier in the 3rd message confirms to \(B\) that is talking to \(A\).
  - \(N_A\) protects against replay attacks.
  - \(N_B\) is utilized for testing \(K_{AB}\).
Symmetric crypto key exchange (4)

- Assumes all keys are secure but

  - What if $T$ gets holds of $K_{AS}$? [E.g. Via password theft]
    - $T \rightarrow B : \{K_{AB}, A\}_{KBS}$
    - $B \rightarrow T : \{N_B\}_{K_{AB}}$
    - $T \rightarrow B : \{N_B + 1\}_{K_{AB}}$

  - $T$ replays old $\{K_{AB}, A\}_{KBS}$ messages, nullifying the effect of reporting the compromised $K_{AS}$ to $S$
Symmetric crypto key exchange (5)

- Fixing Needham-Schroeder by utilising timestamps (*Denning and Sacco*)

\[A \rightarrow S : A, B, N_A\]
\[S \rightarrow A : \{N_A, B, K_{AB}, \{T, K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}\]
\[A \rightarrow B : \{T, K_{AB}, A\}_{K_{BS}}\]
\[B \rightarrow A : \{N_B\}_{K_{AB}}\]
\[A \rightarrow B : \{N_B+1\}_{K_{AB}}\]

- \(B\) rejects the 3\(^{rd}\) message if the timestamp is too old
- Problem – Requires synchronized clocks
Symmetric crypto key exchange (6)

- Fixing Needham-Schroeder while avoiding timestamps (Otway-Rees)

\[ \begin{align*}
A & \rightarrow B : \text{num}, A, B, \{N_A, \text{num}, A, B\}_{K_A} \\
B & \rightarrow S : A, B, \{N_A, \text{num}, A, B\}_{K_A}, \{N_B, \text{num}, A, B\}_{K_B} \\
S & \rightarrow B : \{N_A, \text{num}, K_{AB}\}_{K_A}, \{N_B, \text{num}, K_{AB}\}_{K_B} \\
B & \rightarrow A : \{N_A, \text{num}, K_{AB}\}_{K_A}
\end{align*} \]

- This time B is immediately informed of the intended communication
- **Num** is an integer to associate all messages with a particular exchange – [in the last message A is expecting num from the first message]
- Nonces protect against replay attacks
- Once again S is assumed to be trusted
Public crypto key exchange

- Conceptually very simple
  \[ A \rightarrow B : A, \{K_{AB}\}_{KB} \]

- Attack
  - \[ T \rightarrow B : A, \{K_{TB}\}_{KB} \]
  - B cannot be sure to be communicating with A

- Solution
  - \[ A \rightarrow B : \{[K_{AB}]_{KA}\}_{KB} \]
    - Sign and Encrypt
    - Encrypt and Sign (not in conjunction with the previously presented timestamp version)
Key Storage (1)

- Need to protect a cryptographic key in a way other than by remembering it
- Public keys
  - Need to protect the key’s integrity
  - Any certificate-based mechanism suffices
- Symmetric and private keys
  - Need to have their confidentiality protected also
Key Storage (2)

1. Protect by access control
   - Access control mechanisms can be defeated
     - Password theft, Elevation of Privilege

2. Enciphering the files containing the keys
   - When user enters the password to decipher the files, key resides in memory
     - Unauthorized memory access attack
     - Key logging attacks
Key Storage (3)

3. Place keys onto a physical device
   - Special terminal, ROM, smart cards
   - Key never enters into computer memory
     - E.g. Smart card inserted in special terminal that enciphers messages sent from computer memory
   - At no point is the key stored in the computer memory

4. Place keys onto multiple physical devices
   - Key split over multiple storage devices
   - Attacker need to steal all devices to get access to the key
Key Escrow (1)

- A system in which a third party can recover a cryptographic key
- Recovery of lost keys or of keys that won’t get revealed
  - **Business**
    - Recovery of backup keys
  - **Law enforcement**
    - Recovery of keys used to encipher communications to which an authority requires access, such as enciphered mail or telephone messages

- 3 alternatives
  1. Weak key
  2. Weak cipher
  3. **Store a copy of the key**
Key Escrow (2)

- 5 desirable properties

1. The escrow system should not depend on the cipher algorithm
   - The escrow techniques should work regardless of how the messages are enciphered

2. Privacy protection mechanisms must work from one end to the other and be part of the user interface
   - This protects the user's privacy unless the escrowed keys are used, and then only those who have the escrowed keys can access the messages
Key Escrow (3)

3. Requirements (legal or business) must map to the key exchange protocol
   - This prevents a user from enciphering a message and then entering it directly into the communications channel, bypassing the escrow system

4. If the message is to be observable for a limited time, the key escrow system must ensure that the keys are valid for that interval exactly (no more and no less)
5. A system supporting key escrow must require that all parties authenticate themselves

- In particular, if a principal uses the escrowed keys, the system must ensure that the principal is authenticated not only by name but also by the time and place of the principal and by any equipment used in the interception and the decipherment

- This protects against unauthorized parties using escrowed keys
Key Escrow (5)

- Key Escrow System Components
  1. User Security
     - Carries our d/encryption
     - Supports key escrow component
  2. Key Escrow
     - Manages the storage and use of the data recovery keys
  3. Data Recovery
     - Carries out data recovery
  - E.g. - U.S. government's **Clipper Chip**
    - Balancing the need for law enforcement access to enciphered traffic against citizens' right to privacy
    - Suffered from a serious vulnerability and later de-classified
Key Revocation (1)

- Revocation of compromised keys
  - A revoked key has been cancelled at the request of the owner or issuer for some reason other than expiration

- 2 problems with public key revocation
  1. Ensure that the revocation is correct
     - Ensure that the entity revoking the key is authorized to do so
  2. Ensure timeliness of revocation throughout the infrastructure
     - Ideally, notice of the revocation will be sent to all parties when received, but invariably there will be a time lag
Key Revocation (2)

- Certificate revocation list
  - A list of certificates that are no longer valid
  - Used by X.509 and Internet public key infrastructures (PKIs)
  - Contains the serial numbers of the revoked certificates, the dates on which they were revoked, name of the issuer, the date on which the list was issued, and when the next list is expected to be issued
  - Certificate issuer (CA) also signs the list
Zero Knowledge Proofs
Zero Knowledge Proofs (ZKPs)

- A secret-preserving authentication scheme
  - Alice wants to prove to Bob that she knows a secret without disclosing any information about it to Bob
  - Bob must be able to verify that Alice knows the secret without gaining information about it

- Allows for the construction of Privacy Enhanced Protocols – authentication with anonymity
  - i.e. - an eavesdropper cannot figure out who is being authenticated
  - Private key? - public key crypto discloses identity to an eavesdropper of authenticated party when public keys are exchanged
Ali Baba’s Cave Protocol (1)
Ali Baba’s Cave Protocol (2)

- Bob is at Q
- Alice is at R
- Alice claims she knows the secret work to open the door between R and S
- Bob wants to verify whether Alice knows the secret, **without Bob gaining any knowledge about the secret**
- Bob randomly selects a side for Alice to appear from
  - If Bob chooses R, Alice can appear whether she knows the secret or not
  - If Bob chooses S, Alice can appear only if she knows the secret
Ali Baba’s Cave Protocol (3)

- Alice can cheat Bob with a probability of $\frac{1}{2}$
- If Bob repeats test $n$ times, probability is only $(\frac{1}{2})^n$
- If $n$ is large enough Bob can verify that Alice knows the secret to a high probability
Ali Baba’s Cave Protocol (4)

- Can we achieve the same effect in a ZKP without the cave?
  - Any NP problem can be used to replace the cave
    - Integer factorization
    - E.g. Alice must prove to Bob that she knows \((p,q)\) without the secret being revealed to Bob
Fiat-Shamir Protocol (1)

- Relies on the fact that finding square root of x modulo N is difficult
  - \( N = pq \) - 2 large prime numbers
  - Alice knows secret \( S \)
  - \( N \) and \( v = S^2 \mod N \) are made public
  - Alice must convince Bob that she knows \( S \) without revealing any information about \( S \)
Fiat-Shamir Protocol (2)

\[ A \rightarrow B : x = r^2 \mod N \]
\[ B \rightarrow A : e \in \{0,1\} \]
\[ A \rightarrow B : y = r^S e \mod N \]

1. Commitment\witness phase
   - A commits to choice of \( r \)

2. Challenge phase
   - B challenges A to provide the correct response

3. Response phase
   - A must respond with the correct value, that B must verify
Fiat-Shamir Protocol (3)

A -> B : x = r^2 \ mod \ N
B -> A : e \ in \ \{0,1\}
A -> B : y = r^*S^e \ mod \ N

□ B verifies that \( y^2 = x^v^e \ mod \ N \)
□ \( y^2 = r^2S^{2e} = r^2(S^2)^e = x^v^e \ mod \ N \)

□ \( x^v^e \ mod \ N \)
□ \( v \) and \( N \) are public
□ \( x \) is sent by A by choosing a random \( r \)
□ \( e \) is the random selection B makes
Fiat-Shamir Protocol (4)

- $e = 1$
  - $y^2 = r^2(S^{2^1}) = r^2(S^2) = xv \mod N$
  - Alice needs to know the secret in order to open the door in Ali Baba’s cave

- $e = 0$
  - $y^2 = r^2(S^{2^0}) = r^2(1) = x \mod N$
  - Alice does not need to open the door
Fiat-Shamir Protocol (5)

- Verifying the protocol security
  - Can T convince B to know S?
  - Thus convincing B of being A

- If T expects e = 0
  - T does not need to know S and sends
    - x = r^2 \mod N
    - y = r \mod N
  - B verifies
    - y^2 = x \mod N
    - x generated by T
If $T$ expects $e = 1$

- $T$ need to know $S$, but still can fool $B$ by sending
  - $x = r^2 v^{-1} \mod N$
  - $y = r \mod N$

- $B$ verifies
  - $y^2 = xv \mod N$
  - $y^2 = (r^2 v^{-1})v \mod N = r^2 \mod N$
  - $S$ out of the equation once again!!
    - $xv \mod N$ same as $r^2 \mod N$ in this case
Fiat-Shamir Protocol (7)

- But
  - If $e$ is generated at random each time, then $T$ can only fool $B$ with probability $1/2$.
  - After $n$ iterations, $T$ can fool $B$ with only a probability of $(1/2)^n$ ($T$ does not know whether to send $x = r^2 \mod N$ or $x = r^2 v^{-1} \mod N$).
Fiat-Shamir Protocol (8)

- Is the Fiat-Shamir Protocol really zero knowledge?
  - \( v = S^2 \mod N \) and \( N \) are public
  - B gets \( r^2 \mod N \) in the first message
  - If \( e = 1 \)
    - B gets \( rS \mod N \) in the third message
    - If B finds \( r \) from \( r^2 \mod N \) he can find \( S \)!
    - Assumption – Modular square root is computationally difficult
    - \( r \) is randomly generated per iteration
    - This assumption must hold for the protocol to be regarded as ZKP
Designing The Best Authentication Protocol
Design considerations

☐ What is the sensitivity of the application?
☐ What delay is tolerable?
☐ What type of crypto is supported?
☐ Is mutual authentication required?
☐ Is a session key required?
☐ Is perfect forward secrecy desired?
☐ Is anonymity a concern?
Summary
In a distributed environment encrypting messages is not enough, we need security protocols as well.

Protocol attacks are attacks on the interaction sequence between principals, and not on the underlying ciphers.

Replay and MITM attacks are the two most common type of attacks on such protocols.

In authentication protocols, both encryption and freshness are required to guarantee security.

Nonce and timestamp mechanisms are used for guaranteeing freshness.
Summary (2)

- Mutual authentication in security protocols
- Reflection attacks are possible in the mutual authentication protocols utilizing shared symmetric keys
- Key management is a sensitive issue that can defeat the security of a protocol in terms of key generation, exchange, storage, escrow and revocation
- ZKP’s allow for authenticating a principal by proving the knowledge of a secret but without disclosing the secret
Question - 1

☐ Some people are trying to sell the idea of a “multifunction smartcard,” an authentication device that could be used in a wide range of transactions to save users having to carry around dozens of different cards and keys

■ Discuss
Question - 2

- Can the reflection attack be carried out on a mutual authentication protocol using public key encryption?
Question - 3

Which security protocol design is the most secure: ‘sign and encrypt’ or ‘encrypt and sign’?
Question - 4

☐ In the Fiat Shamir protocol \( A \) does not need to know \( S \) when \( B \) challenges with \( e=0 \)

- why do we not send \( e=1 \) every time?
Question - 5

☐ In the Fiat-Shamir protocol suppose that A gets lazy and decides to use the same “random” \( r \) for each iteration

■ Show that T can determine A’s secret \( S \)
■ Why is this a security concern? (besides the actual disclosure of the secret)
Question - 6

Consider the case of Alice and her stockbroker, Bob. Suppose they decide not to use a session key but only the interchange public/private key pair. Instead, Alice pads the message (BUY or SELL) with random data.

- Explain under what conditions this approach would be effective
- Discuss how the length of the block affects your answer
Question - 7

- Consider the following authentication protocol, which uses a classical cryptosystem
  - Alice generates a random message $r$, enciphers it with the key $k$ she shares with Bob, and sends the enciphered message $\{r\}_k$ to Bob
  - Bob deciphers it, adds 1 to $r$, and sends $\{r + 1\}_k$ back to Alice. Alice deciphers the message and compares it with $r$. If the difference is 1, she knows that her correspondent shares the same key $k$ and is therefore Bob
  - If not, she assumes that her correspondent does not share the key $k$ and so is not Bob
  - Does this protocol authenticate Bob to Alice? Why or why not?
Question - 8

A -> B : A
B -> A : \{A, N_B\}_{K_{BS}}
A -> S : A, B, N_A, \{A, N_B\}_{K_{BS}}
S -> A : \{A, B, N_A, K_{AB}, \{A, N_B, K_{AB}\}_{K_{BS}}\}_{K_{AS}}
A -> B : \{A, N_B, K_{AB}\}_{K_{BS}}
B -> A : \{N_{B2}\}_{K_{AB}}
A -> B : \{N_{B2}+1\}_{K_{AB}}

Show up till which extent this protocol solves the problem of replay attacks in step 5 (refers to the Needham-Shroeder Protocol)
Question - 9

Consider the following protocol, where \( K = h(S, N_A, N_B) \) and \( \text{CLNT} \) and \( \text{SRVR} \) are constants, \( K_{AB} \) is a shared symmetric key, \( S \) is a message signed by \( A \)

\[
\begin{align*}
A & \rightarrow B : A, N_A \\
B & \rightarrow A : N_B \\
A & \rightarrow B : \{S\}_{K_{AB}}, \{\text{CLNT}\}_K \\
B & \rightarrow A : \{\text{SRVR}\}_K
\end{align*}
\]

- Does \( A \) authenticate \( B \)?
- Does \( B \) authenticate \( A \)?
The following two-message protocol is designed for mutual authentication and to establish a shared symmetric key $K$. Here, $T$ is a timestamp.

$A \rightarrow B : A, [T]_{K_A}, \{K\}_{K_B}$

$B \rightarrow A : [T+1]_{K_B}, \{K\}_{K_A}$

This protocol is insecure. Illustrate a successful attack by Trudy.
Question - 11

- Suppose $R$ is a random challenge sent in the clear from $A$ to $B$ and $K$ is a symmetric interchange key known to both $A$ and $B$. Which of the following are secure session keys and which are not?
  - $\{R\}_K$
  - $\{K\}_R$
  - $R \oplus K$
The following mutual authentication protocol is based on a shared symmetric key $K_{AB}$:

- $A \rightarrow B : A, N_A$
- $B \rightarrow A : N_B, \{N_A\}_{K_{AB}}$
- $A \rightarrow B : \{N_B\}_{K_{AB}}$

Show that $T$ can attack the protocol to convince $B$ that it is $A$, where we assume that the cryptography is secure.

Modify the protocol to prevent such an attack by $T$. 

---

Question - 12
Consider the following mutual authentication and key establishment protocol. This protocol uses a timestamp $T$ and public key cryptography

$$A \rightarrow B : A, \left[\{T, K\}_{KB}\right]_{KA}$$
$$B \rightarrow A : \left[\{T+1, K\}_{KA}\right]_{KB}$$

- Show that $T$ can attack the protocol to discover the key $K$. We assume that the cryptography is secure.
- Modify the protocol to prevent such an attack by $T$
Bibliography


