Chapter 02: Protocol Building Blocks

Uday Manchanda

December 16, 2019

Abstract

Chapter 02 Notes in the book *Applied Cryptography* by Bruce Schneier

1 2.1: Introduction to Protocols

- Protocol: series of steps, involving two or more parties, designed to accomplish a task
- Everyone involved must know the protocol and all the steps involved
- Cryptographic protocol: protocol that uses cryptography
- The Purpose of Protocols
  - Purpose is to prevent/detect eavesdropping and cheating
  - **It should not be possible to do more or learn more than what is specified in the protocol**
  - Protocols help to formalize how things are document
  - Don't assume that everyone is honest; in-fact, it may be best to assume everyone is dishonest
- The Players
  - Alice + Bob are the two person protocols (Alice will initiate, Bob will respond)
  - Carol + David for later roles
- Arbitrated Protocols
  - Arbitrator: disinterested third party trusted to complete a protocol
  - He/she has no interested in the protocol and no particular allegiance to either party, all people involved trust this person

Alice	First participant in all the protocols
Bob	Second participant in all the protocols
Carol	Participant in the three- and four-party protocols
Dave	Participant in the four-party protocols
Eve	Eavesdropper
Mallory	Malicious active attacker
Trent	Trusted arbitrator
Walter	Warden; he'll be guarding Alice and Bob in some protocols
Peggy	Prover
— Victor	Verifier

  - In the real world, lawyers and bankers act as arbitrators
  - Arbitrator can be the vulnerable point in the protocol as everyone must trust him and he/she musn't be a crook

- Adjudicated Protocols
  - Two types of arbitrated protocols
  - Nonarbitrated protocol: the parties just want to complete the protocol
  - Arbitrated protocol: if a dispute arises, an adjudicator comes in to determine if the protocol was performed fairly
  - Not always necessary

- Self Enforcing Protocol
  - The best kind where the protocol itself guarantees fairness

- Attacks against Protocols
  - Passive attack: where someone not involved in the protocol can eavesdrop. He/she does not affect the protocol itself, but tries to gain info
  - Active attack: someone tries to alter the protocol in their favor. This is far more serious
  - The attacker could be one of the parties involved in the protocol. He/she is called a cheater. There are two types:
    * Passive cheater: follows protocol, tries to obtain more information
    * Active cheater: disrupt the protocol to their advantage

2 2.2: Communications Using Symmetric Cryptography

- To communicate securely, encrypt communications
- For Alice to send an encrypted message to Bob
  - Alice and Bob agree on a cryptosystem
  - Alice and Bob agree on a key
  - Alice takes her plaintext msg and encrypts it using the encryption algorithm and key. This creates a ciphertext message

  - Alice sends ciphertext to Bob
  - Bob decrypts the ciphertext message with the same algorithm and key reads it
- A good cryptosystem is one in which all the security is inherent in knowledge of the key and none ins inherent in knowledge of the algorithm
- Hence why key management is so important
- The problems with symmetric cryptosystems
  - Keys must be distributed in secret as key are as valuable as all of the messages they encrypt
  - If a key is compromised, then an attacker (Eve) can decrypt all message traffic encrypted with that key
  - If a separate key is used for each pair of users, the number of keys increases rapidly as the number of users goes uses
  - Specifically, $\frac{n(n-1)}{2}$ keys are needed for a group of n users

3 2.3: One-Way Functions

- A one-way function is a type of function where it is "easy" to compute but much more difficult to reverse
- Given x, one may calculate $f(x)$. But given $f(x)$, it is very difficult to compute x
- In a strictly mathematical sense, there is no evidence that one-way functions actually exist or even be constructed
- However, many cryptrographic functions are able to be computed easily but reversing them is nearly impossible
- A trapdoor one way function is a special type of one way function with a secret trapdoor
- It follows the same properties of a normal one way function; however, if you know the secret, you can easily compute the function in the other direction

4 2.4: One-Way Hash Functions

- Hash function: takes variable length input (pre-image) and converts to a fixed length output (hash value)
- A simple hash function would be pre-image $\rightarrow$ byte consisting of all the XOR of all the input bytes

- Hash functions are many-to-one and works in one direction
- Should also be collision free; it is hard to generate two pre-images with the same hash value
- The actual hash functions are public, but their one-wayness is the real security
- Can be useful for verifying files: have the person send you the hash of the file
- If they match, the files are the same
- MAC: Message Authentication Code. A one way hash function with the addition of a secret key
- Hash value is the function of the pre-image and key

5 2.5: Communications Using Public-Key Cryptography

- Symmetric algorithms are like safe where the key is the combination
- Public-key crytography involes two keys: one public and one private
- Computationally hard to deduce the private key from the public key
- Only the person with the private key can decrypt the message
- Public key crytography made it easier to send encrypted messages
- With symmetric encryption, both the sender and receiver needed to agree on a key
- Public key crypto is also easier to scale up
- Hybrid Cryptosystems
  - Public key algorithms are used to encrypt keys not messages
  - Therefore it is not a substitute for symmetric algorithms
  - Two reasons for encrypting keys not messages
    - 1. Public key algorithms are slow. Symmetric algorithms are 100x faster
    - 2. Public key algorithms are vulnerable to chosen-plaintext attacks.
    - If $C = E(P)$ where P is one plaintext out of a set of n plaintexts, then a cryptanalyst only has to encrypt all n possible plaintexts and compare the results with C

    - Encryption key is public
  - Chosen-plaintext attacks are effective if there are relatively few possible encrypted messages
  - In most practical implementations P-K cryptography is used to secure and distribute session keys
  - These session keys are then used with symmetric algorithms to secure message traffic
  - Known as a hybrid cryptosystem
  - Example of how it works
    - Bob sends Alice his public key
    - Alice generates a random session key, encrypts it using Bob's public key and sends to Bob
    - Bob decrypts Alice's message using his private key to recover the sesion key
    - Both of them encrypt their communications using the same session key
  - Session key is created when needed to encrypt communications and destroyed when no longer needed
  - This makes it harder for an intruder to listen to messages

- Merkle's Puzzles
  - Ralph Merkle published a paper regarding Secure Communications over Insecure Channels
  - His proposal was based on puzzles that were easier to solve for the sender and receiver than for an eavesdropper
  - Alice can send an encrypted message to Bob without having to exchange a key
    - Bob generates 2^{30} (million) messages of the form: *This is puzzle number x. This is the secret key number y*
    - x is a random number, y is a random secret key; different for each message
    - Using a symmetric algorithm he encrypts each message with a different 20bit key and sends them to Alice
    - Alice chooses one message at random and performs a brute force attack to recover the plaintext. A very large task
    - Alice encrypts her secret message with the key she recovered and some symmetric algorithm and sends to Bob along with x
    - Bob knows which secret key y he encrypts in message x so he can decrypt the message
  - Eve (the eavesdropper) can break this system but has to do far more work
  - To recover the message she has to perform a brute force attack against each of Bob's 2^{30} messages. Complexity of 2^{40}

6 2.6: Digital Signatures

- Need a way to prove authorship
- Signing documents with physical signatures is not secure because they can be forged or copied easily
- Signing documents with Symmetric Cryptosystems and an Arbitrator
  - Alice wants to sign digital message and send it to Bob
  - she can do it with the help of Trent, a trusted arbitator, and a symmetric cryptosystem
  - What happens:
    - Alice encrypts message to Bob K_A and sends to Trent
    - Trent decrypts message with K_A
    - Trent takes decrypted message and a statement that he has received this message from Alice and encrypts the whole bundle with K_B
    - Trent sends the encrypted bundle to Bob
    - Bob decrypts the bundle with K_B. He can read the message and Trent's certification that Alice sent it
  - Trent knows the message is from Alice because he infers from the message's encryption and because only they share the secret key
  - However the arbitrator (Trent) acts as a bottleneck because every time Alice wants to send a message to Bob, she must send it via Trent first
  - Also Trent must be perfect, even a slight error will render all the signatures he verified to be useless
- Digital Signature Trees
  - Uses a tree structure to produce an infinite number of one time signatures
  - Place the root of the tree in a public file, thereby authenticating it
  - Root signs one message and authenticates its sub nodes in the tree
  - Each node then signs one message and authenticates its sub nodes, and so on