Computer Security

08r. Pre-exam 2 Last-minute Review – Cryptography

Paul Krzyzanowski

Rutgers University

Spring 2018

Cryptographic Systems

Types of ciphers

- Symmetric cipher: same key to encrypt and decrypt

  $c = \{ m \} k \qquad p = \{ c \} k$

- Asymmetric cipher: two *related* keys, public and private
  - Based on trapdoor functions:

    One-way functions – irreversible unless unless you have extra data (another key)
  - A message, m, encrypted with one can be decrypted *only* with the corresponding key:

    $c = \{ m \} k_{pub} \qquad p = \{ c \} k_{priv}$

    $c' = \{ m' \} k_{priv} \qquad p = \{ c \} k_{prub}$

Properties of good ciphers

- Kerckhoff's Principle
  - A cryptosystem should be secure even if everything about the system, except the key, is public knowledge
  - *Public algorithms, secret keys*

- Properties of good ciphers
  1. Ciphertext should look random
     - There should be nothing in the ciphertext that gives any indication of what the corresponding plaintext is
  2. There should be no way to generate plaintext from ciphertext without the correct key except via an exhaustive search
  3. Keys should be large enough that an exhaustive search is not practical

Classic ciphers

- Monoalphabetic substitution cipher
  - Replace one character (bit pattern) with another
    - always the same substitution
  - Caesar cipher: the substitution alphabet is the alphabet shifted by *n* positions
  - Vulnerable to frequency analysis: certain letters are more likely than others
- Polyalphabetic substitution cipher
  - Change the substitution alphabet based on the position of the character
    - Alberti cipher: shift the substitution alphabet every *N* characters
    - Vigenère cipher: select the substitution alphabet based on the next character of the key – use a *repeating key* (repeat the key to make it as long as the text)
    - Still vulnerable to frequency analysis but more difficult
- One-time pad
  - Key = long set of random characters
  - Each key character encrypts one plaintext character
    - Originally; add modulo alphabet size; on computers use XOR
  - Problem
    - Key must be (1) truly random, (2) as long as the message, (3) never reused

Classic ciphers

- What is perfect secrecy?
  - Ciphertext contains no information about the plaintext
  - It was proved that perfect secrecy can *only* be achieved if the key is truly random and as long as the message
  - The one-time pad is the only algorithm that provides perfect secrecy
- Stream cipher
  - Approximation of a one time pad
  - Instead of a random stream of bits for the long key, create the stream using a pseudorandom number generator
    - Generate a stream of numbers with random properties
    - Completely deterministic: based on a starting number (seed)
  - Only as secure as the seed
- Rotor machines
  - Electromechanical devices
  - Implement a polyalphabetic substitution cipher – but with a really long period before the same substitution alphabet is reused

Transposition ciphers

- Instead of substituting characters, we scramble them
  - Skytale (rhymes with Italy) – wrap a line of text around a rod; read the text along the rod
  - Equivalent to writing text horizontally in a matrix and reading it vertically
  - This is a block cipher: text has to be a multiple of N characters (width of the matrix)
    - You might need to add padding to the text
- Not vulnerable to frequency analysis
  - Same distribution of characters – they are just scrambled

Symmetric block ciphers

- Encrypt a fixed number of bits at a time
  - Key is applied to each block
- Multiple rounds per block. Each round =
  - Generate a subkey = use bits from the key and modify them
  - Send the rounds though a substitution-permutation network
    - Bit transpositions and XORs based on subkey
- DES
  - Based on Feistel cipher
    - Each round: apply substitution-permutation on half of the block & exchange halves
  - 56-bits: brute force attacks are feasible
  - 3DES: Use up to three 56-bit DES keys
    - Encrypt with key_1, decrypt with key_2, encrypt with key_3
- AES
  - 128 – 256 bit keys

Using block ciphers

- Electronic code book (ECB)
  - Each block of plaintext is encrypted individually
  - Problem: attacker can replace encrypted blocks
- Counter mode
  - An incrementing count is encrypted with the key for each block
  - Result is XORed with the block of plaintext to create ciphertext
  - Benefits
    - Result based on the position of the block in the message
    - Encryption can be distributed among computers
- Cipher block chaining (CBC) mode
  - Ciphertext from one block is XORed with the plaintext of the next block, which is then encrypted
  - Encryption of each block is a function of the previous blocks

Cryptanalysis

- Differential Cryptanalysis
  - Identify non-random behavior
  - Examine how changes in input affect changes in output
  - Find whether certain bit patterns are unlikely for certain keys
- Linear Cryptanalysis
  - Create equations to predict the relationships between ciphertext, plaintext, and the key
- In both of these, you won't break the code but you can find keys or data that are more likely are that are unlikely: reduces the keys to search

Secure communication

Secure communication

- Symmetric cryptography
  - Encrypt and decrypt with a secret key
  - Both parties must share the key
- Public key cryptography
  - If Alice & Bob want to talk
    - Alice encrypts a message with Bob's public key
      - Only Bob can decrypt the message with his private key
    - Bob encrypts messages with Alice's private key
- Hybrid cryptography
  - Public key cryptography is *really slow …* and generating keys takes time
    - Use it to send a *session key*
    - Alice generates a random session key & encrypts it with Bob's public key
    - Bob decrypts it with his private key
    - Now Alice & Bob communicate with symmetric cryptography

Key Exchange

Diffie-Hellman key exchange

- Not an encryption algorithm: only key exchange
- Each party generates public and private ”keys”
- For Alice & Bob to talk, they generate a common key
  - Alice: common key = f(Alice’s private key, Bob’s public key)
  - Bob: common key = f(Bob’s private key, Alice’s public key)
- Forward secrecy
  - Means that even if you steal someone’s private key, you cannot decipher their past communications
    - Requires single-use (ephemeral) keys
    - Diffie-Hellman is efficient for this: create new new key pairs on the fly
    - RSA key generation is far less efficient
      - Key generation takes far longer
      - RSA is good for long-term keys (e.g., your identity; for authentication)

Trusted third party

- With symmetric cryptography (without the Diffie-Hellman algorithm), we have to rely on a trusted third party to exchange keys
  - The trusted third party has everyone's keys
- If Alice wants to talk to Bob
  - Sends a request to the trusted party, Trent, encrypted with her secret key
  - Trent creates a session key for Alice & Bob to communicate
    - Sends the session key to Alice – encrypted with her secret key
    - Sends the session key to Bob – encrypted with his secret key
  - They both have the key and can communicate
- Implicit authentication
  - If Alice or Bob are imposters, they cannot get decrypt the key

Key exchange with a trusted third party

- Simple key exchange is vulnerable to replay attacks
  - Someone can just play back a set of messages to Bob and he'll think it's Alice
- Needham-Shroeder algorithm – use a trusted 3rd party to send a session key
  - Add random strings (nonces) to an encrypted request
  - Response must contain the same string: prove you can encrypt something
- Denning-Sacco modification: add timestamps
  - An attacker who cracked an old session key can replay the transmission of the session key to Bob and masquerade as Alice
  - Timestamps allow Alice & Bob to check that the messages are not old
- But clocks might not be synchronized
  - Use a random session ID for each message
  - Works sort of like a nonce but is present in every message

Kerberos

- Combined authentication, authorization, and key exchange service
- Kerberos is a trusted third party
- Similar protocol to Needham-Schroeder with Denning-Sacco timestamps
- Alice wants to talk to Bob
  - Send a request to Kerberos
  - Kerberos gives Alice:
    - An timestamp and session key – both encrypted with Alice's secret key
    - A "ticket" that contains a timestamp and the session key
      - Both encrypted with Bob's secret key, so only Bob (& Kerberos) can decrypt it
  - Alice sends the ticket to Bob
    - She proves that she was able to decrypt the session key by sending the timestamp encrypted with the session key
    - Bob proves he was able to decrypt the ticket by permuting the timestamp (adds one), encrypting the result, and sends it back to Alice

Kerberos Ticket Granting Server (TGS)

- Kerberos is split into two parts
  - Authentication Server
  - Ticket Granting Server
  - Both run the exact same protocol
- Authentication server
  - Used only to give users tickets for the Ticket Granting Server
    - Alice gets a session key to the TGS
    - She can cache this without worrying about leaking her password
- Ticket Granting Server
  - Used to access all other services
    - Alice asks the TGS for a ticket to talk to Bob
    - The response is encrypted with Alice's TGS session key

Public Key session key exchange

- Public key cryptography makes it super easy to send a session key
- For Alice to talk to Bob
  - Alice encrypts a random session key with Bob's public key
  - Only Bob can decrypt this
- This is vulnerable to forgery (anyone can do this)
  - Alice can sign the key too
    - Encrypt the random session key with her private key (only Alice can do this)
    - Encrypt the result with Bob's public key (only Bob can decrypt this)

Integrity

Cryptographic hash functions

- Cryptographic hash functions
  - One-way functions: variable input, fixed output
  - Pre-image resistant
    - Given H(M), you cannot compute M
  - Collision resistant
    - Given it is difficult to find M, M' where H(M) = H(M')
  - Output gives no information about input
    - Changing a bit of the input does not predictably change specific output bits
- Birthday paradox
  - Finding two messages M, M' where H(M) = H(M') is easier than finding a message M' where H(M) = H(M') for a specific message M
  - The birthday paradox tells us the complexity is approximately the square root of the number of messages

Message Integrity and Hash Pointers

- Cryptographic hash functions act as a checksum for the data they are generated from
  - If the data is modified, we trust that the corresponding hash will be different
- Hash Pointer = { pointer, *hash*(data pointed to) }
  - Allows us to not just get to the data but also validate that the data has not been modified
  - Used for tamper-proof data storage
- Blockchain = Linked list of hash pointers
  - A hash pointer in a data block points to the next data block, which contains a hash pointer to its next data block …
  - If an adversary modifies data in one block, she will have to modify all previous hashes back to the head of the list
- Merkle tree = Binary tree of hash pointers
  - Allows us to find the data block that has changed in $O(\log_2 n)$ time

Message Integrity & MACs

- If H(M) = H(M), we are confident that the messages are the same
- If H(M) ≠ H(M), we are confident that the messages are different

- We can transmit M and H(M) to validate that a message has not been modified
  - But what if an attacker can modify both M and H(M)?
- Message Authentication Codes (MAC)
  - Add a key so that only someone who has the key can create the MAC validate the message
  - HMAC – Hash-based MAC: hash key & message
  - CBC-MAC: use cipher block chaining on a symmetric algorithm (e.g., AES) and just take the last output block (each block is a function of all previous blocks)

Digital signatures

- If you encrypt a message with your private key
  - Anyone can decrypt it with your *public* key – but they know only you coud have created it.
- But
  - We may not want to hide the message
  - And encrypting with a public key algorithm can be much slower
- So
  - Digital signature = hash(message) encrypted with the owner's private key
- Digital signatures provide non-repudiation
  - Alice cannot say that she did not sign the message if it has her signature
  - Only Alice has her private key

Identity binding

- How do we know we really have Bob's public key … or Alice's?
- Digital certificates
  - Contain
    - User's public key
    - User's distinguished name (information such as name, email, organization)
    - The issuer – the Certification Authority (CA) who is responsible for stating that the distinguished name is associated with the public key
  - Signature
    - Hash of the data encrypted with the CA's private key
    - This makes the certificate data unforgable
- To validate a certificate
  - Hash the data
  - Decrypt the signature with the CA's public key
  - Compare the two values: they should be the same

Chains of trust

- Certificate chaining
  - How do you get the CA's public key?
    - From the CA's certificate
  - This may be signed by another CA
  - Eventually, you reach the root and just have to trust the CA and get the key (or self-signed certificate) in a trusted manner
- Revocation
  - Certificates have an expiration date in them
  - But may be revoked earlier
  - The CA can distribute a Certificate Revocation List that contains serial numbers of certificates that should no longer be accepted.

The end