Cryptography - A Review

Mahalingam Ramkumar
Mississippi State University, MS

March 3, 2014

1. Symmetric Cryptography
   - Symmetric Cryptography Overview
   - Block Cipher Modes
   - Multiple Encryption
   - Hash Functions
   - Message Authentication Codes
2. Asymmetric Cryptography
   - Number Theory
   - RSA (Rivest - Shamir - Adelman)
   - Diffie Helman Key Exchange
3. Key Management
   - PKI
   - Basic KDS
   - Kerberos
4. Network Security
   - Lack of Authentication
   - Authentication Protocols

Symmetric Cryptography

1. Data-mangling based on a key
2. Data-mangling should be **reversible**
3. Two basic types of reversible data-mangling - substitution and permutation
4. Modern ciphers use a combination of both - substitution permutation networks
5. Repeatedly...
6. Confusion and Diffusion
7. Two main classes - Block ciphers, Stream Ciphers

Symmetric Cryptography Overview

- $C = E_K(P)$
- $P = D_K(C)$
- No way to get P given C without knowledge of K
- Security lies only in the key. Algorithms are completely **open**.
- Against a good cipher, the only viable attack should be a brute force attack
- Brute force attacks are possible because of "redundancy" in plain text - K should be long enough to dissuade brute force attacks.

Compression and Encryption

- Is redundancy bad?
- What if all data is compressed before encryption? (to avoid brute force attacks)
- What happens when compressed and encrypted data is modified enroute by an attacker? (need redundancy for integrity verification)

Should Resist Attacks due to Known $P - C$ Pairs

- Many situations result in exposure of encrypted text - probably at a later time
- Most difficult part is *establishment* of shared key. Ideally should not need to be renewed frequently
- Knowledge of pairs $(P_1, C_1) \cdots (P_n, C_n)$ should not provide *any* information about the key K.
- Knowledge of pairs $(P_1, C_1) \cdots (P_n, C_n)$ should not provide *any* information about P_{n+1} given C_{n+1}
- Think of the encryption / decryption mechanism as a black box
  - and attacker has access to the black box (but not the key inside)
  - he can find any P for a given C or C for a given P.
  - but he should not be able to find the key K.

Block Ciphers

- Fiestel structure
- Encryption
  - $L_i = R_{i-1}$
  - $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$
- Decryption
  - $R_{i-1} = L_i$
  - $L_{i-1} = R_i \oplus F(R_{i-1}, K_i)$
- Repeated Fiestel rounds

CBC - Cipher Block Chaining

- $C_O = IV$.
- Encryption: $C_j = E_K(C_{j-1} \oplus P_j)$
- Decryption: $P_j = D_K(C_j) \oplus C_{j-1}$
- Problem: Say attacker changes IV to IV' enroute. But leaves $C_1, C_2, \ldots$ intact.
- Decryption: $P_1 = D_K(C_1) \oplus IV'$
- Attacker can modify select bits of P_1.

CFB - Cipher Feedback Mode

- $C_O = IV$.
- Encryption: $C_j = E_K(C_{j-1}) \oplus P_j$
- Decryption: $P_j = E_K(C_{j-1}) \oplus C_j$
- Attacker changes IV to IV' enroute. But leaves $C_1, C_2, \ldots$ intact.
- $P_1 = E_K(IV) \oplus C_1$. P_1 changes - but attacker does *not* know *how* exactly P_1 is modified.

OFB - Output Feedback Mode

- $O_O = IV$, $O_j = E_K(O_{j-1})$
- Encryption: $C_j = O_j \oplus P_j$
- Decryption: $P_j = O_j \oplus C_j$
- Attacker changes IV to IV' enroute. But leaves $C_1, C_2, \ldots$ intact - every P_j changes - unpredictably.
- Attacker can selectively change *any* bit in *any* C_j (and leave IV intact)
- Corresponding changes occur in P_js.

Stream Cipher

- This problem exists with *any* **stream** cipher.
- To detect deliberate errors a cryptographic hash should always accompany the data.
- Also, never reuse initial value. For the same key and initial value the same encryption stream is produced.
- If a stream $\mathbf{z}$ is reused

$$\begin{aligned} \mathbf{c}_1 &= \mathbf{p}_1 \oplus \mathbf{z} \\ \mathbf{c}_2 &= \mathbf{p}_2 \oplus \mathbf{z} \end{aligned}$$

- Now $\mathbf{c}_1 \oplus \mathbf{c}_2 = \mathbf{p}_1 \oplus \mathbf{p}_2$
- Easy to find both $\mathbf{p}_1$ and $\mathbf{p}_2$ given $\mathbf{p}_1 \oplus \mathbf{p}_2$

Counter Mode

- Starting counter value CTR
- Counter Mode $O_j = E_K(CTR + j)$
- Encryption: $C_j = O_j \oplus P_j$
- Decryption: $P_j = O_j \oplus C_j$
- Random access decryption possible (like ECB mode)
- But does not have the issue of "same C for same P" that ECB mode has.

Multiple Encryption

- $C_1 = E_{K_1}(P) \cdots C_i = E_{K_i}(C_{i-1}) \cdots C = E_{K_n}(C_{n-1})$
- Issue: Is there a single key K which yields $C = E_K(P)$
- Should hold for all P.
- In general, a single "substitution key" will not meet the requirement if the cipher does *not* exhibit "group" property
- Most well known symmetric ciphers don't!
- So increasing key sizes by using multiple encryption is possible
- Double, Triple DES.
- Triple DES. Typically $C = E_{K_3}(D_{K_2}(E_{K_1}(P)))$
- Why? Compatibility with single DES.

Compression Function

- $h = H(M)$
- M may be any number of bits (if less than 512 bits, usually padded to 512 bits)
- h is typically 128 / 160 bits
- Given M, easy to calculate h.
- Given h practically impossible to find M (even one M which satisfies $h = H(M)$.
- Pre-image resistant
- Collision resistant (stronger condition)

Random Oracle

- The inputs M_i are questions posed to the Oracle
- The hash $h_i = H(M_i)$ are the answers
- Given questions $M_1 \cdots M_n$ and their answers $h_1 \cdots h_n$ one should not be able to predict the answer to a question M_{n+1} - however large n is!

Pre-image Resistance

- Given h (say B bits) need to find corresponding M.
- With no prior knowledge, the best thing one can do is guess.
- The probability that a randomly chosen M will yield h is $\frac{1}{2^B}$
- Every 2^B hash is equally likely.
- Need to try order of 2^B candidate Ms to have a reasonable chance of being successful

Collision Resistance

- Given that $H()$ yields B bit hashes
- Find any two Ms (M_1 and M_2) such that $H(M_1) = H(M_2)$.
- Find a collision
- Birthday paradox - collision of birthday occurs with probability 0.5 for a pool size of around 20 people.
- Need to try order of $2^{B/2}$ candidate Ms to have a reasonable chance of finding a collision.
- That's the reason 64 bit ciphers are considered secure, but 64-bit hashes are not.
- Standard hash functions use 128 / 160 bits

- Key based hash function
- Sender and receiver share a secret K
- Example. Message M. $HMAC_M = H(K \parallel M)$. Sends $M \parallel HMAC_M$.
- Receiver extracts message M and verifies $HMAC_M$.
- Successful verification proves *two* things
- 1) Message has not been modified, and 2) Message was sent by the person who knows the secret K.
- CBC and CFB modes can also be used for HMAC.
- Compression vs Redundancy - introduce controlled redundancy (hash) - but still possible to brute force

Modular Arithmetic

- Arithmetic in a finite ring or field
- $\mathbb{Z}_m = \{0, 1, \cdots, m-1\}$
- If m is prime, the ring is a field
- Possible to perform additions, multiplication
- Multiplicative inverses
- In a field all numbers have a multiplicative inverse (except zero)
- In a ring only number relatively prime to the modulus have a multiplicative inverse

Fermat's and Euler-Fermat's Theorem

- Fermat's theorem $a^{p-1} \bmod p \equiv 1$
- Euler - Phi Function $\Phi(m)$ - number of numbers below m relatively prime to m
- Or the number of elements in $\mathbb{Z}_m$ that have a multiplicative inverse.
- Euler Fermat's theorem - $a^{\Phi(m)} \bmod m \equiv 1$ if $(a, m) = 1$.

If Modulus $m = pq$

- Fermat's theorem $a^{p-1} \bmod p \equiv 1$ or $a^p \bmod p \equiv a$
- Euler Fermat's theorem - $a^{\Phi(m)} \bmod m \equiv 1$ if $(a, m) = 1$, but
- if $m = pq$, $a^{\Phi(m)+1} \bmod m \equiv a$ for all a (even if $(a, m) \neq 1$)
- More generally, for any two integers k, a,

$$\begin{aligned} a^{k(p-1)+1} &\equiv a \bmod p \\ a^{k\Phi(m)+1} &\equiv a \bmod m \end{aligned} \tag{1}$$

Why Modular Arithmetic

- Deterministic operations yield seemingly random results
- Good for cryptography!
- Provides a good suite of one-way problems
- Factorization - Given two primes p and q finding $n = pq$ is trivial. But given n finding p and / or q is not.
- Discrete Logarithms. Let $\alpha = g^a \bmod p$. Given a, g and p easy to calculate α. But given α, g and p practically impossible to calculate a for large p.

What is easy, what is not

- Exponentiation is easy. Evaluating g^a mod m where g, a and m are integers of 200 digits each involves only order of $\log(m) \approx 665$ multiplications
- Square and Multiply algorithm
- Finding multiplicative inverse is easy enough - $\log(m)$ complexity
- Extended Euclidean algorithm
- Testing if a number m is prime is also doable
- Probabilistic primality testing

RSA - (Rivest - Shamir - Adelman)

- Choose two large primes p and q.
- $n = pq$ is the modulus ($\mathbb{Z}_n$ is a ring - not a field)
- $\Phi(n) = (p-1)(q-1)$.
- Choose e such that $(e, \Phi(n)) = 1$.
- Find d such that $de \equiv 1 \bmod \Phi(n)$. Or d is the multiplicative inverse of d mod $\Phi(n)$ (use extended Euclidean algorithm)
- Destroy p, q and $\Phi(n)$.
- n and e are public keys
- d is the private key
- Cannot determine p and q from n (factorization is hard)
- Cannot determine $\Phi(n)$ without factoring n. So finding d given e (and n) is hard.

RSA - As a Cipher

- Alice's public keys are n_a and e_a.
- Bob wished to send a message P to Alice
- $C = P^{e_a} \bmod n_a$. Bob sends C to Alice
- $P = C^{d_a} \equiv P^{e_a d_a} \equiv P^{k\Phi(n_a)+1} \equiv P \bmod n_a$.
- $e_a d_a \equiv 1 \bmod \Phi(n_a) \rightarrow e_a d_a = 1 + k\Phi(n_a)$.
- Only Alice (who has access to d_a) decrypt the message

RSA - As a Signature Scheme

- Alice's public keys are n_a and e_a.
- Alice wishes to send (broadcast) a signed message P
- $S = P^{d_a} \bmod n_a$. Alice broadcasts S
- $P = S^{e_a} \equiv P^{e_a d_a} \equiv P^{k\Phi(n_a)+1} \equiv P \bmod n_a$.
- $e_a d_a \equiv 1 \bmod \Phi(n_a) \rightarrow e_a d_a = 1 + k\Phi(n_a)$.
- Any recipient (who knows Alice's public key e_a) can verify that the message is from Alice.

RSA Example

- $p = 11, q = 13.\ n = pq = 143$
- $\phi(n) = (p-1)(q-1) = 120.$
- Choose $e = 7$
- $d \equiv e^{-1} \bmod 120 \equiv 103.$
- $P \equiv 41 \bmod 143.$
- $C \equiv 41^7 \bmod 143 \equiv 24.$
- $P \equiv C^d \equiv 24^{103} \bmod 143 \equiv 41.$

Diffie Helman Key Exchange

- RSA is based on difficulty of factorization
- DH is based on difficulty of calculating discrete logarithms
- A known p, and (preferably) a generator g in $\mathbb{Z}_p$.
- Alice chooses a secret a, calculates $\alpha \equiv g^a \bmod p$.
- Bob chooses a secret b, calculates $\beta \equiv g^b \bmod p$.
- Alice and Bob exchange α and β
- Alice calculates $K_{AB} \equiv \beta^a \bmod p$.
- Bob calculates $K_{AB} \equiv \alpha^b \bmod p$.
- Both of them arrive at $K_{AB} \equiv g^{ab} \bmod p$.
- K_{AB} is a secret that no one apart from Alice and Bob can calculate!

DH Example

- $p = 13, g = 2$.
- Alice chooses $a = 3$. $\alpha \equiv g^a \equiv 2^3 \equiv 8 \bmod 13$
- Bob chooses $b = 7$. $\beta \equiv g^b \equiv 2^7 \equiv 11 \bmod 13$
- Alice calculates $K_{ab} \equiv \beta^a \equiv 11^3 \equiv 5 \bmod 13$
- Alice calculates $K_{ab} \equiv \alpha^b \equiv 8^7 \equiv 5 \bmod 13$

DH Based Cipher

- Alice with p, g, a, α.
- Bob knows α, p, g, needs to send P to Alice
- Needs a mask. Chooses k. Sends g^k and $C = P\alpha^k$ to Alice
- Unmasking - $\alpha^k = (g^k)^a$. Only Alice can unmask α^k.
- Multiply C by inverse of α^k.
- El Gamal Cipher
- El Gamal Signature Scheme, Schnorr Scheme

El Gamal Cipher Example

- $p = 13, g = 2$.
- Alice chooses $a = 3$. $\alpha \equiv g^a \equiv 2^3 \equiv 8 \bmod 13$
- Bob chooses $k = 5$, $g^k \equiv 2^5 \equiv 6 \bmod 13$
- Say $P = 4$. $\alpha^k \equiv 8^5 \equiv 8 \bmod 13$. $C = P\alpha^k = 4 * 8 \equiv 6$
- Alice calculates $\alpha^k \equiv (g^k)^a \equiv 6^3 \equiv 8$
- Inverse of 8 is 5.
- $P = P\alpha^k(\alpha^k)^{-1} \equiv 6 * 5 \equiv 4 \bmod 13$.

Practical Signature Schemes

- Message M.
- $h = H(M)$.
- h is encrypted with the private key to get the signature S.
- Send (broadcast) $M \parallel S$
- Receiver:
- Extract M. Calculate $h = H(M)$.
- Verify decryption of S with public key yields h.

El Gamal Signature Scheme

- Public $p, g \in \mathbb{Z}_p$ - g preferably a generator.
- Alice's private key $a \in \mathbb{Z}_p$.
- Alice's public key $\alpha \equiv g^a \bmod p$.
- To sign a message M with hash $h = H(M)$ Alice chooses a random $1 \leq k \leq p-2$ and calculates:

$$\begin{aligned} \gamma &= g^k \bmod p \\ \delta &= (h - a\gamma)k^{-1} \bmod (p-1) \end{aligned}$$

- Signature is $\gamma \parallel \delta$
- Verification: $\alpha^\gamma \gamma^\delta \equiv g^h \bmod p$.

El Gamal Example

- $p = 79, g = 7$. Let $h = 12$ and $k = 5$.
- Alice's private key $a = 43$. Public key $\alpha \equiv 7^{43} \equiv 48 \mod 79$.
- $k^{-1} \mod (p-1) \equiv 47 \mod 78$.
- $\gamma = g^k \equiv 7^5 \equiv 59 \mod 79$.
- $\delta = (12 - 43 \times 59)41 \equiv 41 \mod 78$.
- $\alpha^\gamma \gamma^\delta \equiv 48^{59} 59^{41} \equiv 8 \mod 79$
- $g^h \equiv 7^{12} \equiv 8 \mod 79$.

Schnorr Signature Scheme

- Public p, q, g_q, hash function $H()$.
  - Large prime p (say 1024-bits)
  - Prime $q \mid (p-1)$ (say 160-bit prime)
  - $g_q \in \mathbb{Z}_p$, where g_q has order q.
- Alice's private key $a \in \mathbb{Z}_p$, public key $\alpha \equiv g_q^a \bmod p$.

Schnorr Signature Scheme

- To sign a message M Alice chooses a random $1 \leq k \leq q-1$ and calculates:

$$\begin{aligned} \gamma &= H(M \parallel g_q^k) \bmod q \\ \delta &= (k + a\gamma) \bmod q \end{aligned}$$

- Signature is $\gamma \parallel \delta$ (both γ and δ are 160-bit quantities)
- Verfier can calculate
  $x \equiv g_q^{\delta} \alpha^{-\gamma} \equiv g_q^{k+a\gamma} g_q^{-a\gamma} \equiv x \equiv g_q^k \bmod q$
- And verify $H(M \parallel x) \equiv \gamma \bmod q$.

Public Key Cryptography

- Computationally about 1000 times more expensive than symmetric cryptography
- Used sparingly
- Used for encrypting session keys
- Encrypting hash of messages
- RSA - bandwidth efficient, primality testing is hard
- DH - based schemes - no need to generate prime numbers. Everyone can use the same p, g. Needs twice the bandwidth.

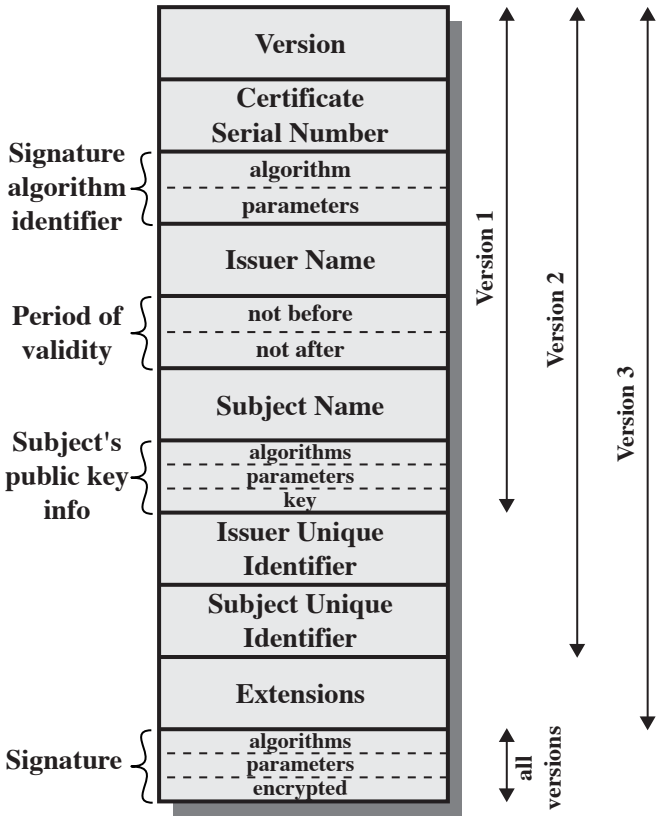
Key Distribution

- Involves distribution of secrets
- Which are used for *establishing* authenticated shared secrets (or session keys).
- PKI
- Basic KDS
- Kerberos (Needham - Schroeder protocol)

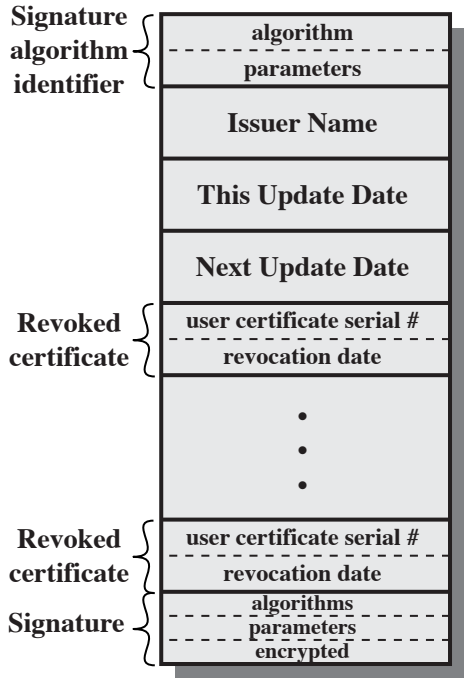
Public Key Infrastructure

- X.509 Authentication service
- Based on asymmetric cryptography
- Basic function - authentication of public keys
- Achieved by signing public keys
- Public key certificates issued by certifying authorities (CA)
- Permits different public key algorithms
- Revocation of certificates

X.509 Formats

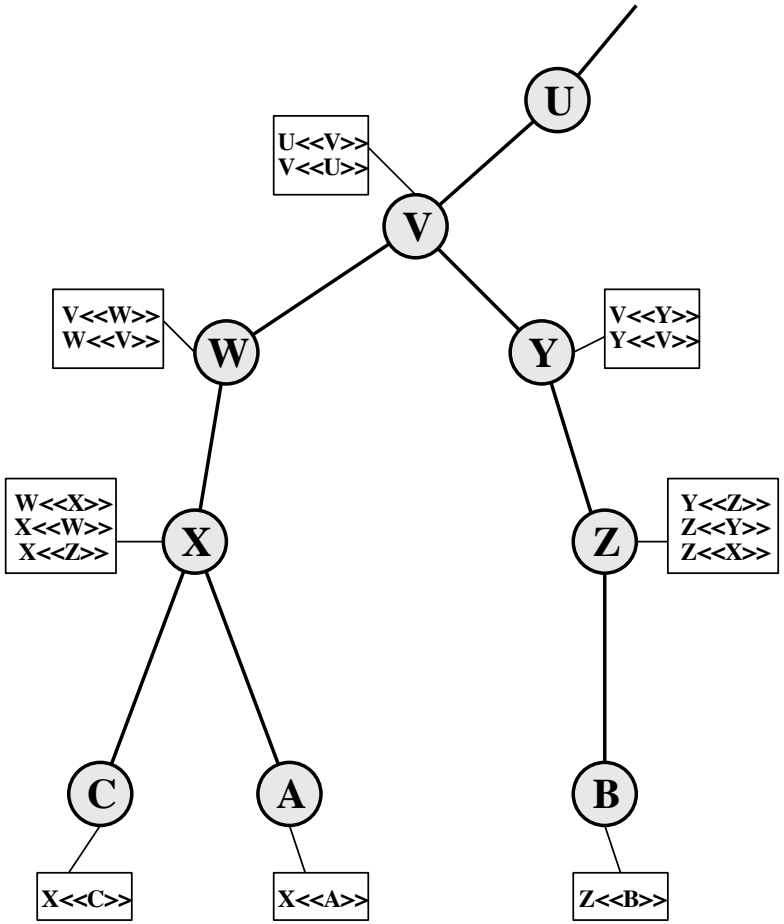


(a) X.509 Certificate



(b) Certificate Revocation List

X.509 Hierarchy



X.509 Certificates

- Forward and reverse signatures
- $CA << A >>$ - forward signature
- $A << CA >>$ - reverse signature
- Tree-hierarchical organization of CA's

Basic KDS

- Offline TA
- N nodes
- $\binom{N}{2}$ possible pairs. Each pair is assigned a key
- Each node gets $N - 1$ keys
- Does not scale well (for large network sizes)

Overview: Needham-Schroeder Key Establishment Protocol

- A trusted server S
- Every node shares a secret with the trusted server
- K_A is key shared between A and server
- To establish a session secret K with B

$$\begin{aligned} A \rightarrow S: &\quad A \parallel B \parallel E_{K_A}(A \parallel B) \\ S \rightarrow A: &\quad E_{K_A}(K \parallel T) \\ T = &\quad E_{K_B}(K \parallel A \parallel B) \\ A \rightarrow B: &\quad T \end{aligned}$$

- Inter-nodal secrets are established by using the trusted server for mediation
- Requires *on-line* server

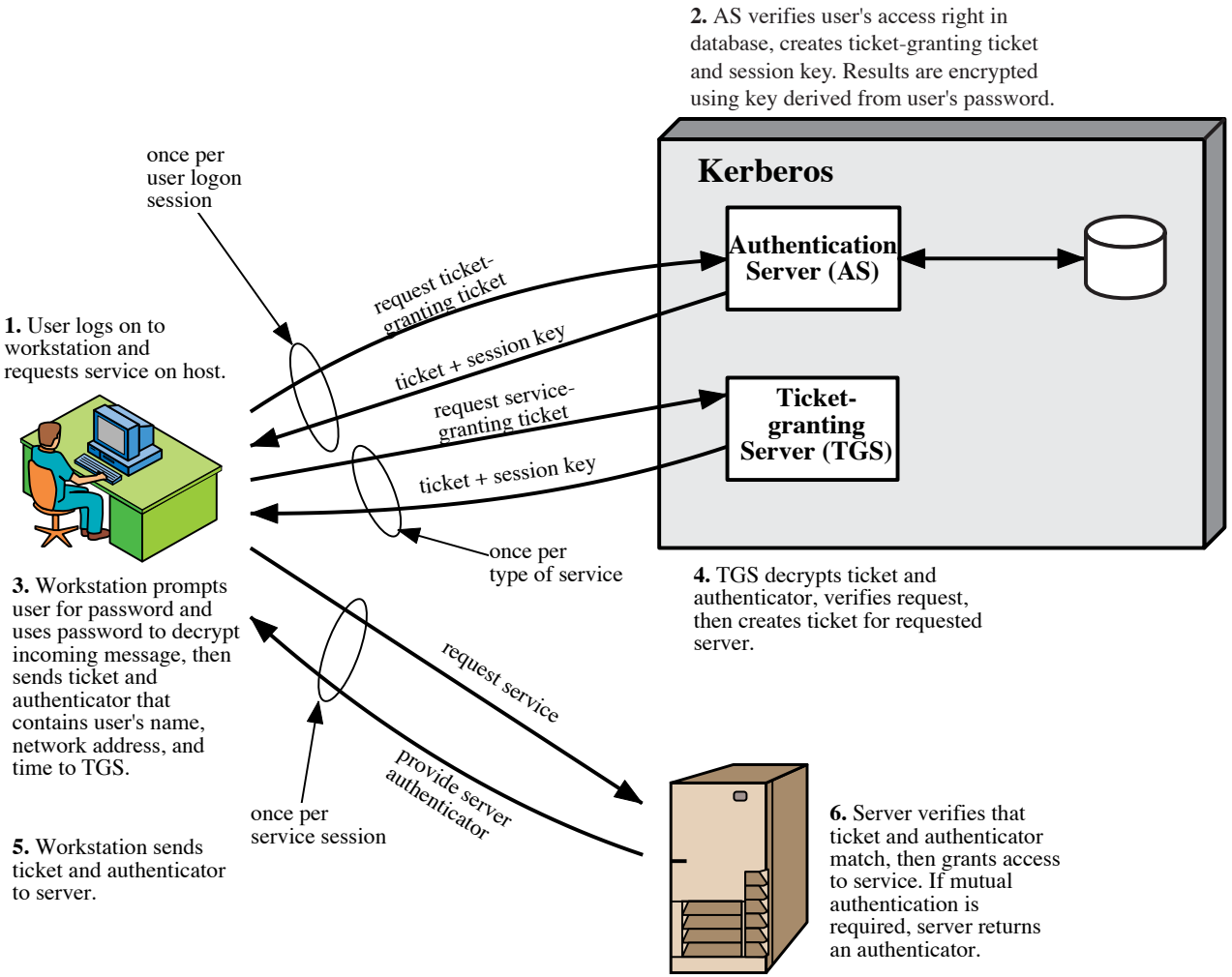
Kerberos Components

- Authentication server (AS)
- Ticket granting server (TGS)
- Servers (V)
- Clients (C)
- TGS and all clients share a secret (individually) with the AS
- All servers share a secret with the TGS.

Kerberos Operation

- Once a day: Clients login and obtain a ticket that will be honored by the TGS
- Once for every service (for example Email server, print server, file server etc.): Clients approch TGS to obtain a ticket (for each server)
- For every service session: Tickets provided by TGS (for that service) provided to the server (email / print / file)
- Tickets are authenticated with an "Authenticator" to prevent replay attacks.

Kerberos Overview



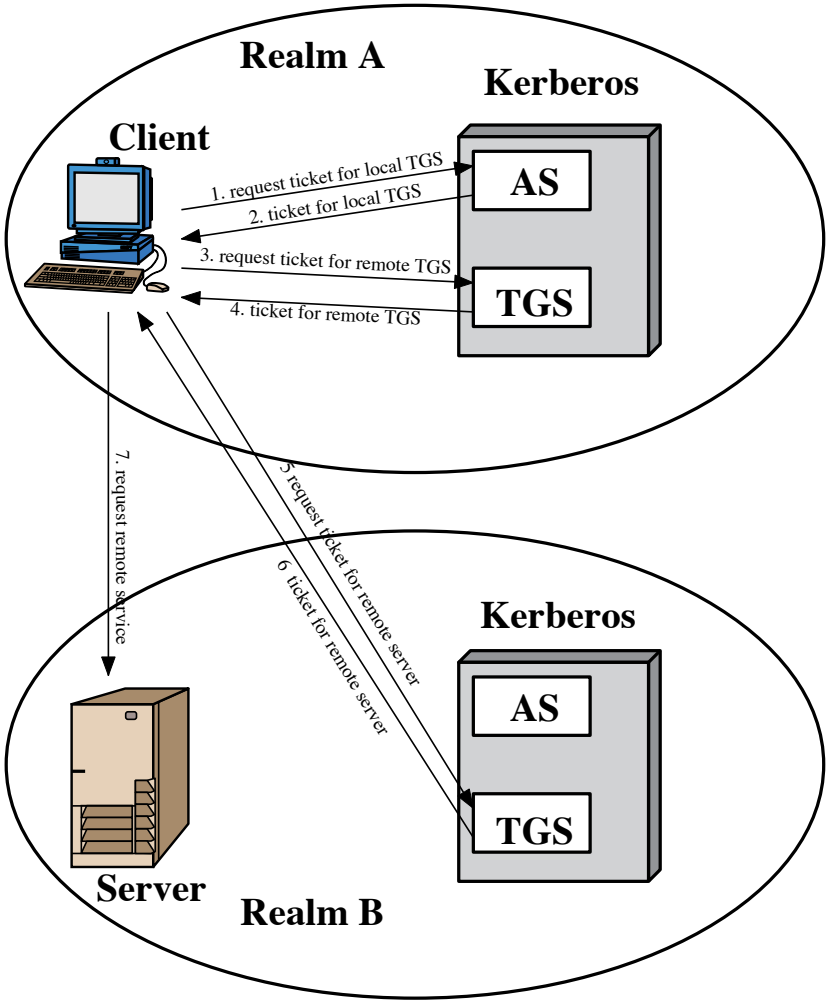
Kerberos (Version 4) Messages

$$
\begin{aligned}
C \rightarrow AS: &\quad ID_c \parallel ID_{tgs} \parallel TS_1 \\
AS \rightarrow C: &\quad E_C(K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel LT_2 \parallel T_{tgs}) \\
T_{tgs} = &\quad E_{TGS}(K_{c,tgs} \parallel ID_c \parallel AD_c \parallel ID_{tgs} \parallel TS_2 \parallel LT_2)
\end{aligned}
$$

$$
\begin{aligned}
C \rightarrow TGS: &\quad ID_v \parallel T_{tgs} \parallel \{A_{c,tgs} = E_{c,tgs}(ID_c \parallel AD_c \parallel TS_3)\} \\
TGS \rightarrow C: &\quad E_{c,tgs}(K_{c,v} \parallel ID_v \parallel TS_4 \parallel T_v) \\
T_v = &\quad E_{tgs,v}(K_{c,v} \parallel ID_c \parallel AD_c \parallel ID_v \parallel TS_4 \parallel LT_4)
\end{aligned}
$$

$$
\begin{aligned}
C \rightarrow V: &\quad T_v \parallel \{A_{c,v} = E_{c,v}(ID_c \parallel AD_c \parallel TS_5)\} \\
V \rightarrow C: &\quad E_{c,v}(TS_5 + 1)
\end{aligned}
$$

Multiple Kerberi



Multiple Kerberi

- The Kerberos servers in different realms share a key
- Typically the TGSs
- Usually AS and TGS run on the same server

Kerberos vs PKI

- Symmetric / asymmetric
- Online Server / Offline Server (ad hoc authentication)
- Scalability?
- Versatility? No equivalent for signatures with Kerberos.
- Revocation?
- Need for "next update" field in X.509 revocation messages?

Source Address Spoofing

- ARP spoofing
- Network Address Spoofing - most of them resulting in some form of DoS or DDoS
  - Smurf - sending ICMP ping request to broadcast addresses
  - Fraggle - sending UDP echoes to broadcast addresses
  - Pingpong - UDP - some servers respond even to malformed packets - day-time servers - so spoof a request from one DTS to another.
  - Land Attack - Same "from" and "to" IP address and port number - caused crashing of network stack in almost all OSes - Berkeley TCP/IP stack.
- DNS spoofing
- Browser address bar spoofing
- User spoofing

Prevention (Without using Cryptography)

- ARP spoofing - ??? (with cryptography - WEP???)
- IP address spoofing
  - TCP makes it difficult to establish connections with a spoofed IP address
  - Ingress and Egress filtering
- DNS spoofing - trusted DNS servers
- Browser
  - avoid full screen mode
  - periodically keep hitting F11(?) to switch between full screen and regular mode - even better disable automatic switching to full screen by scripts.
  - Don't use IE!
  - Take time to explore security options in browsers
- User spoofing - challenge response? (do not reuse!)

Similar Approaches - Different Layers

- PGP - authentication of user
- DNSsec - authentication of domain name - IP mapping
- SSL / TLS - authentication of server (and optionally client)
- SET - authentication of multiple parties in complex transactions
- IPSec - IP layer security

Similar How?

- Need to establish shared secrets - using public key cryptography
- Need to authenticate public keys - X.509 certificates, inherent hierarchy, building trust through "interactions"
- The actual (symmetric / asymmetric crypto) algorithm used not very important
- Most of them provide options for various algorithms
- Need to establish security associations - agree on algorithms, keys etc. - might need some initial interaction for setting up keys.

System Security

- Firewalls
- Intrusion Detection
- Password Management
- Main goal of an Intruder - escalation of privileges
- The path - gain common user privilege (password cracking) - then get root (admin) privilege using “exploits.”

Intrusion Detection

- Getting tougher everyday!
  - Too many applications
  - Heavy traffic (cf base-rate fallacy)
  - Too many entry points
  - Users do not want any kind of restrictions
- Basic approach - examine packets and determine if they are "good" or "bad"
- Probabilistic approaches, may use prior knowledge of exploits, user behaviour
- Probabilistic merit of success

Base Rate Fallacy

- $\mathcal{D}$ - disease present. $\mathcal{H}$ - healthy. $+$ - detection of disease
- $\Pr\{+ \mid \mathcal{D}\} = 0.85$
- $\mathcal{D}$ occurs only in 1% of the population
- Given that disease has been detected, what is the probability that the person actually has the disease
- Given that in a town (with a lot of people) 25 of them tested positive what is the percentage of *false alarm*?
- A - False alarm occurs when - $\Pr\{+ \mid \mathcal{H}\}$
- B - Total occurrences of $+$ - $\Pr\{+ \mid \mathcal{D}\} \times \Pr\{\mathcal{D}\} + \Pr\{+ \mid \mathcal{H}\} \times \Pr\{\mathcal{H}\}$
- Probablity of false alarm is $\frac{A}{B} = \frac{0.15 \times 0.99}{0.15 \times 0.99 + 0.85 \times 0.01} = 0.946$
- Even if $\Pr\{+ \mid \mathcal{D}\} = 0.99$, probability of false alarm is 0.5.

Gateways

- Circuit level gateways (CG)
- Application level gateways (AG)
- Difference?
  - Consider telnet between A and B going through the gateway G.
  - With AG, A and G establish a telnet connection, G and B establish a telnet connection
  - With CG, A and G establish a TCP connection, and G and B establish another TCP connection.
  - AG has to be application aware - CG does not have to be.
  - AG does not need any software change in the client. CG requires recompilation of application software in clients
  - CG - eg. SOCKS - applications have to be SOCKS aware (use SOCKS sockets instead of regular OS sockets).