Confidentiality using Symmetric Key Encryption

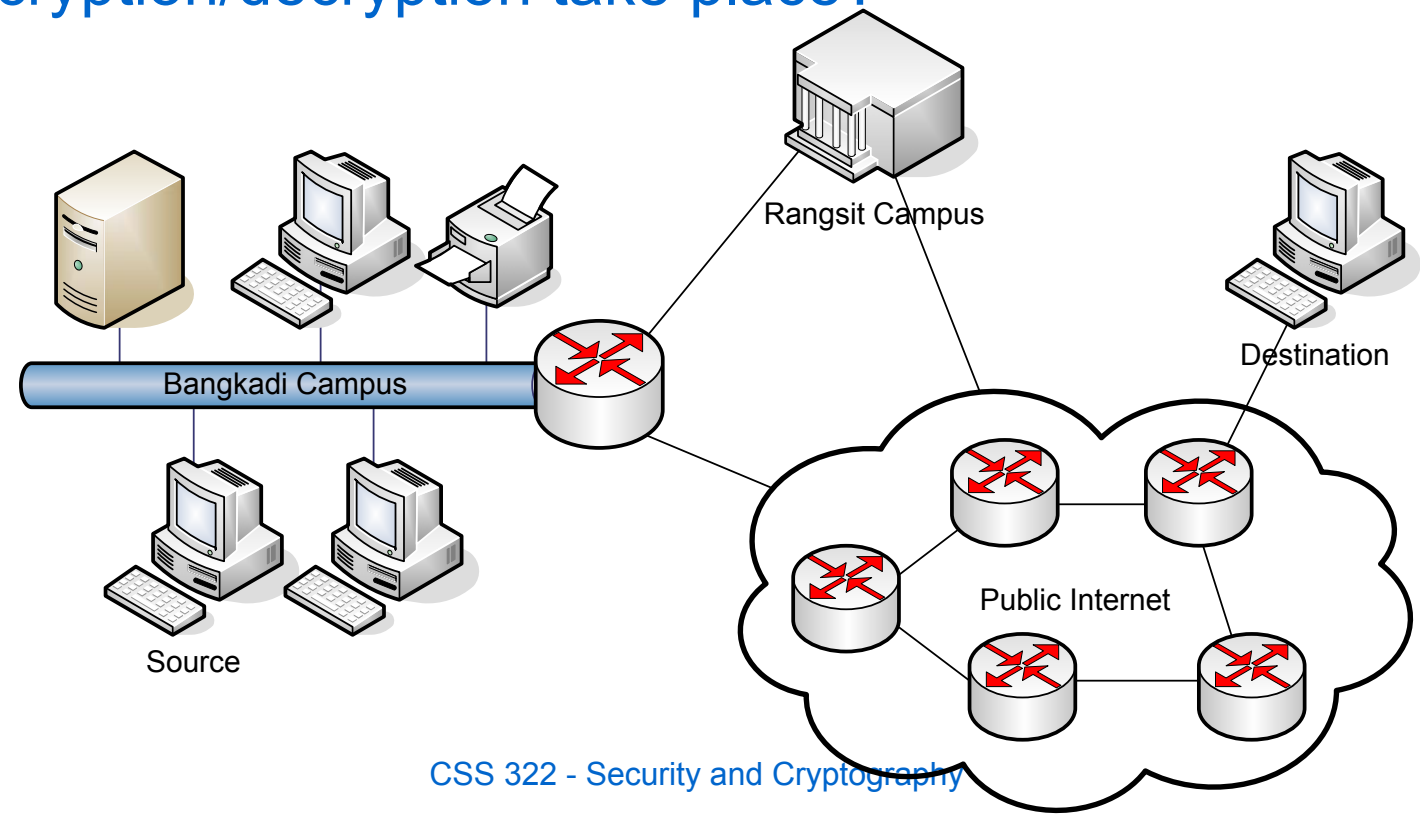
CSS 322 – Security and Cryptography

Contents

- Implementing on a network
- Distributing keys
- Random number generators

Where to Encrypt?

- Traditionally, encryption is used to provide confidentiality (privacy) of information
- For network communications, where should encryption/decryption take place?



Confidentiality Attacks

- An attacker can listen to communications from many places:
  - On the LAN computers both locally and remotely (e.g. log in to LAN from outside via modem)
  - Wiring closet (or similar) used in buildings
  - Network hardware such as routers and switches on the LAN
  - Network hardware in ISP and Telco networks
  - Communication links
    - Especially wireless/satellite
    - Fibre optics are harder to attack without detection

Link versus End-to-End Encryption

- Link Encryption
- Encrypt/decrypt at endpoints of each link
- Requires many encrypt/decrypt devices
- Requires *all* links to use encryption
- Must decrypt/encrypt at each device in path
  - Message is vulnerable at switches (Layer 2 devices)
- E.g. ATM or MPLS switch has a unique key with each of its neighbour switches

- End-to-end Encryption
- Encrypt/decrypt at source/destination hosts
- Hosts do not have to rely on network operators Only data can be encrypted – header information is needed for routers/switches to determine where to send message
  - Vulnerable to traffic analysis
- Message vulnerable at gateways between systems e.g. Internet to phone network

Best to use a combination of link and end-to-end encryption!

Encryption at Different Layers

(a) Application-Level Encryption (on links and at routers and gateways)

On links and at routers

In gateways

(b) TCP-Level Encryption

On links

In routers and gateways

(c) Link-Level Encryption

Traffic Analysis Attacks

- Even with encrypted messages, users may want to hide traffic patterns
- Traffic patterns can reveal:
  - Who is communicating
  - How often they are communicating
  - Types of communications, e.g. message lengths, number of messages, time of communications
  - Correlation between events (in real world) and communications
- Hiding patterns using link encryption
  - Network (IP) and application headers already encrypted
  - Traffic padding: send ciphertext continuously, even when no plaintext. Attacker does not know when real plaintext is sent
- Hiding patterns using end-to-end encryption
  - Much harder. Can send null messages to hide patterns

Key Distribution

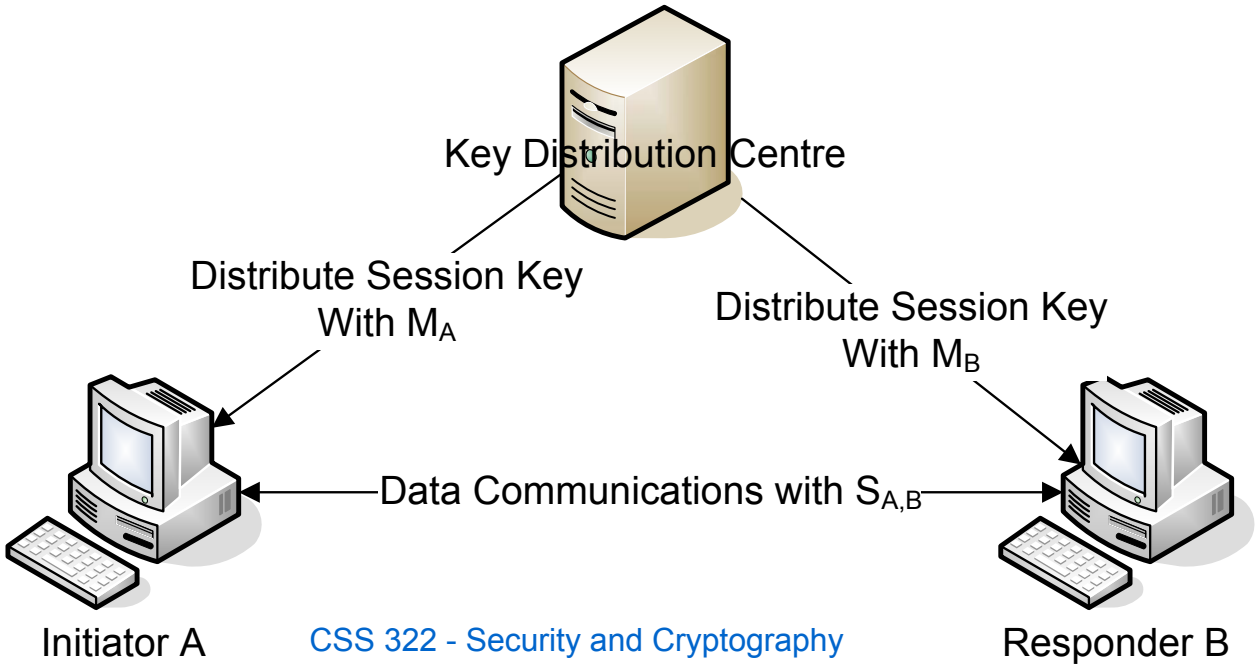
- Until now, we assumed encrypt symmetric key was available at encrypting host and decrypting host
- Security of symmetric key systems relies on key secrecy
- How do we securely distribute this key?
  1. A can physically deliver to B
  2. Third party, C, can physically deliver to A and B
  3. If A and B already have a key, can securely transmit new key to each other, encrypted with old key
  4. If A and B have secure connection with third party C, C can securely send keys to A and B

How Many Keys?

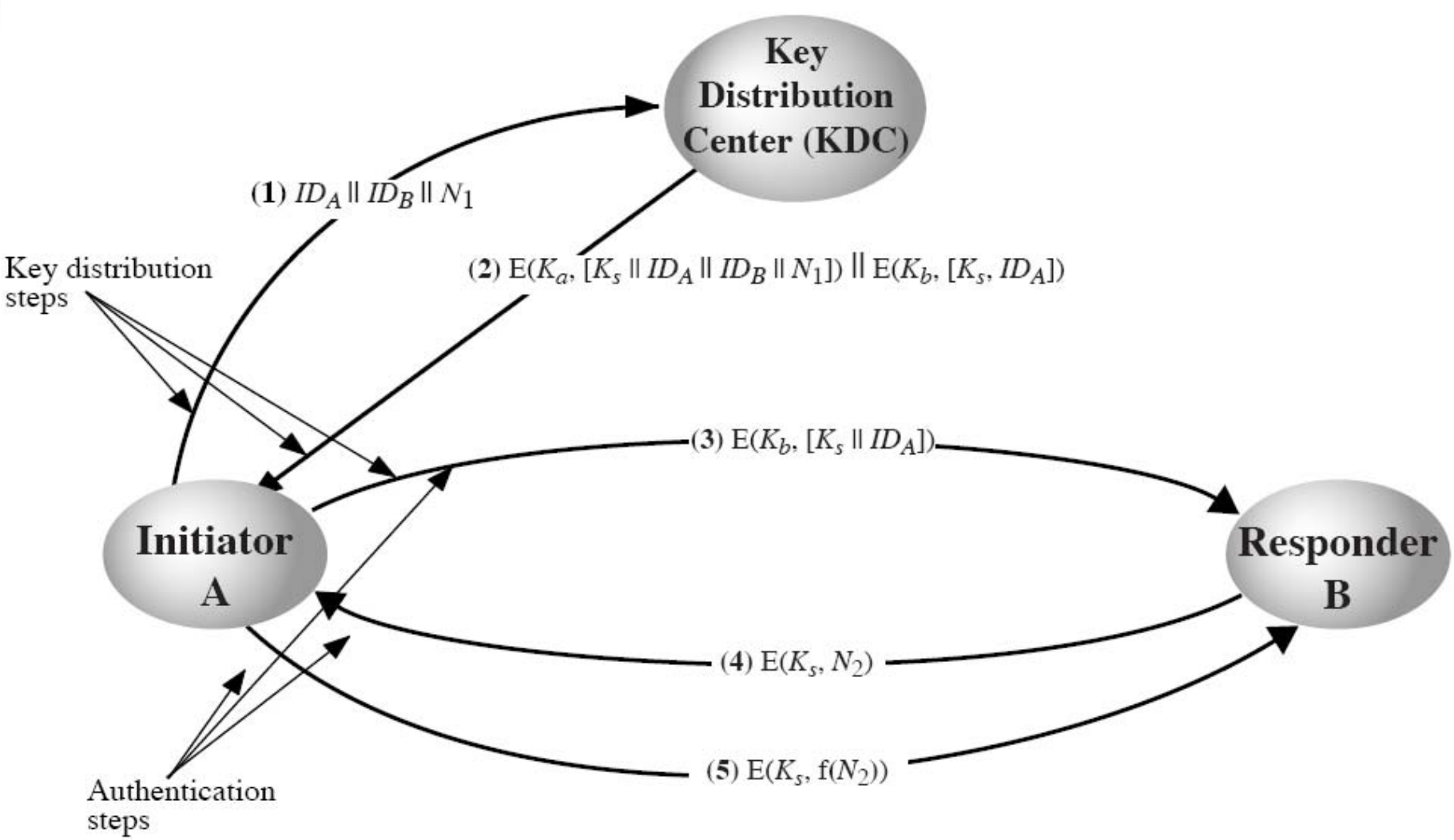
- If N entities, each pair must have unique key:
  - Number of keys = [N(N-1)]/2
- Link encryption
  - Option 1 and 2 are possible, e.g. a switch only exchanges keys with its neighbours (10 to 100 keys)
- End-to-end encryption at IP
  - Number of hosts = 1000; approx 500,000 keys
- End-to-end encryption at application level
  - Number of users/processes = 10,000; approx 50,000,000 keys
- For end-to-end encryption, option 4 is used

Key Distribution Centres

- Each host shares a master key with the KDC (e.g. physical distribution of M_A between KDC and A, and M_B between KDC and B)
  - Only N (e.g. 1000) master keys required – 1 for each host
- For session (e.g. download, voice call) between A and B, KDC generates a session key, $S_{A,B}$
  - $S_{A,B}$ sent to A and B encrypted with corresponding master keys
- Session encrypted with session key $S_{A,B}$



KDC Example

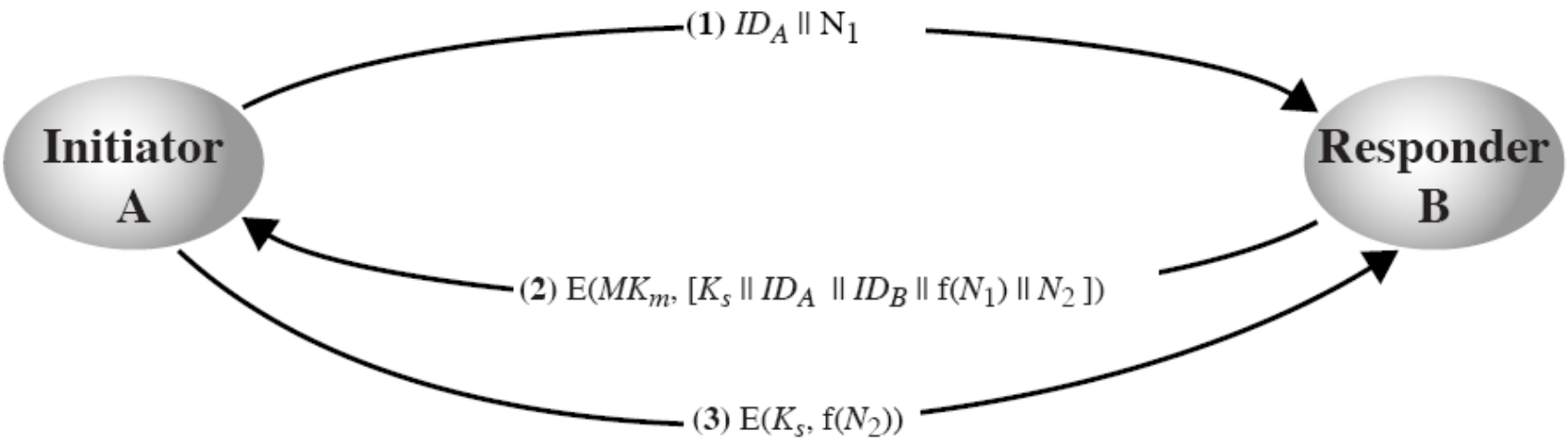

Key Distribution Center (KDC)
(1) $ID_A \| ID_B \| N_1$
Key distribution steps
(2) $E(K_a, [K_s \| ID_A \| ID_B \| N_1]) \| E(K_b, [K_s, ID_A])$
(3) $E(K_b, [K_s \| ID_A])$
Initiator A
Responder B
(4) $E(K_s, N_2)$
(5) $E(K_s, f(N_2))$
Authentication steps

KDC Issues

- Can use more than one KDC
  - In large networks use hierarchy of KDCs; but most trust each other
- Session key lifetime
  - Short lifetime is good; means less time using 1 key, less chance of attacker to analyse ciphertext and plaintext
  - But session key distribution does involve some overhead; need a reasonable trade-off
- KDC must be trusted and protected
  - There are ways for making KDC distributed for smaller networks
- Good practice to use different sessions keys can be used for applications/purposes

Decentralised Key Distribution

- No need for third party C
  - Don't have to trust someone else
- But need many master keys, one for each pair:
  - [N(N-1)]/2 master keys



Random Numbers in Security

- Random numbers are used by many security algorithms:
  - Session key generation in KDC
  - Generation of keys for RSA
  - Handshaking to prevent replay attacks in key distribution
- What is a random number? Two important criteria are:
  - Randomness
    - Uniform distribution: frequency of occurrence of numbers should be about the same (easy to test)
    - Independence: no value can be inferred from any other value (harder to prove)
  - Unpredictability
    - If truly independent, then unpredictable
    - In practice, must be careful that cannot predict patterns and future numbers in a random sequence

Testing for Randomness

- Several different tests can be applied:
  - Number of 1's and 0's in entire sequence should be equal
  - Number of 1's and 0's in M-bit blocks should be equal
  - Analyse the runs of 1's (or 0's) – how many consecutive 1's
  - Matrix, fast Fourier Transform and other mathematical tests abut patterns
  - Compression: it should not be able to be compressed
  - …

Random Number Generators

- How do we (and especially computers) generate random numbers?
- Use deterministic algorithms to calculate pseudo-random numbers
  - Hence, Pseudo-Random Number Generators (PRNG)
- Most widely used PRNG is the linear congruential method

Linear Congruential Generator

- Sequence of random numbers $\{X_n\}$ generated from:
  - m, the modulus: $m > 0$
  - a, the multiplier: $0 < a < m$
  - c, the increment: $0 \leq c < m$
  - X_0, the seed: $0 \leq X_0 < m$

$$X_{n+1} = (aX_n + c) \bmod m$$

- Choice of a, c and m is critical in getting random sequence
  - Sequence should be as long as possible, hence m very large
    - m should be close to the largest nonnegative integer
  - Tests have shown that if m is prime, c is 0, then only several values of a are appropriate, e.g. $7^5 = 16807$
  - For 32-bit computer: $X_{n+1} = (aX_n) \bmod (2^{31} - 1)$
- Problem: once parameters are known, an attacker can determine the sequence of random numbers

Other PRNGs

- Cryptographical PRNGs
  - Use encryption algorithms (aim of encryption is produce 'random looking' ciphertext)
  - Cyclic encryption
    - Use incremental counter as input into encryption algorithm (e.g. DES)
  - ANSI X9.17
    - Uses the current date/time and 64-bit seed as input to 3DES
    - Used in financial applications and PGP
  - Blum Blum Shub generator and others …
- True Random Number Generators
  - Use a nondeterministic source, such as radiation measurements and leaky capacitors
  - Require additional hardware for measurements
- Use published collections of random numbers (in books and on web sites)

ANSI X9.17

- Used in PGP and financial applications
- Uses Triple DES and Date/Time (DT) value as input