COMPUTER & NETWORK SECURITY

Lecture 7:
Key Management

CRYPTOBULLETIN: IN THE LAST WEEK

- **OpenSSL Patch to Plug Severe Security Holes**
  http://krebsonsecurity.com/2015/03/openssl-patch-to-plug-severe-security-holes/
  also: http://www.itnews.com.au/News/401891,openssl-patches-denial-of-service-vulnerabilities.aspx

- **Drupal SQL injection vulnerability attacks persist, despite patch release**
  http://www.scmagazine.com/trustwave-details-drupal-sql-injection-attack/article/404719/

- **New BIOS Implant, Vulnerability Discovery Tool to Debut at CanSecWest**
  https://threatpost.com/new-bios-implant-vulnerability-discovery-tool-to-debut-at-cansecwest/111710

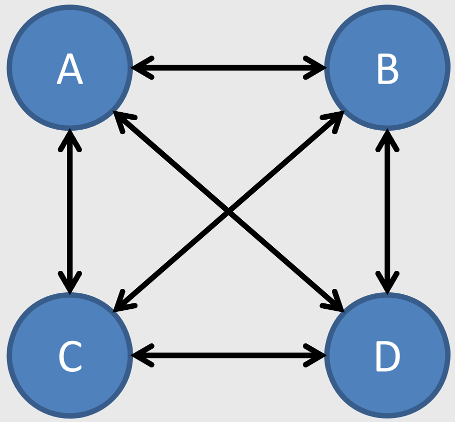
- **Meet "badBIOS", the mysterious Mac & PC malware that jumps airgaps (historical)**
  http://arstechnica.com/security/2013/10/meet-badbios-the-mysterious-mac-and-pc-malware-that-jumps-airgaps/

- **Shopping for Spy Gear: Catalog Advertises NSA Toolbox (historical)**
  http://www.spiegel.de/international/world/catalog-reveals-nsa-has-back-doors-for-numerous-devices-a-940994.html

- **RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis (historical)**
  http://www.cs.tau.ac.il/~tromer/acoustic/

KEY MANAGEMENT

Suppose we have a symmetric key network:



k_{ab}, k_{ac}, k_{ad}, k_{bc}, k_{bd}, k_{cd}

Alice, Bob, Carol and Dave want to talk to each other

For secure communication with n parties, we require

$$\binom{n}{2} = n(n-1) / 2 \text{ keys}$$

Key distribution and management becomes a major issue!

DEFINITIONS

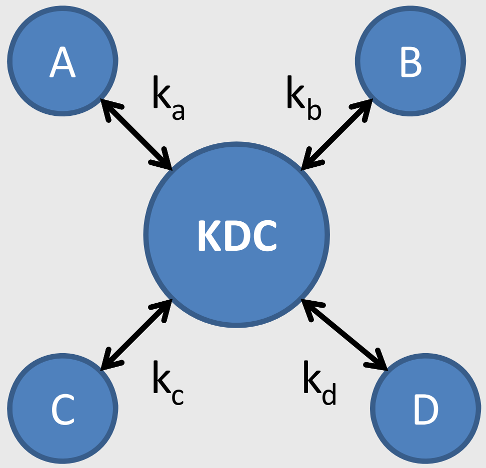
Key establishment is any process whereby a shared key becomes available to two or more parties for subsequent cryptographic use

Key management is the set of processes and mechanisms which support key establishment and the maintenance of on-going keying relationships between parties, including replacing older keys with newer ones:

-- key agreement

-- key transport

KEY DISTRIBUTION CENTRE: NAIVE



Protocol:

(1) Alice → KDC : "want to talk with Bob"

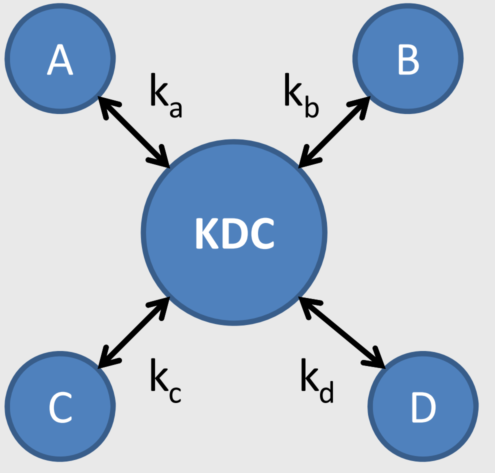
(2) KDC → Alice : KDC picks random key k_{ab}, sends $E_{ka}[k_{ab}]$, $E_{kb}[k_{ab}$, "ticket a-b"]

(3) Alice → Bob : Alice decrypts $E_{ka}[k_{ab}]$, sends ticket to Bob

(4) Bob : Bob decrypts ticket

Alice and Bob now share secret key k_{ab}

KEY DISTRIBUTION CENTRE: NAIVE



Problems:

- The Key Distribution Centre is a single point of failure (likely to be attacked)
- No authentication
- Poor scalability
- Slow

MERKLE’S PUZZLES

- **Ralph Merkle (Stanford, 1974)**
- **Merkle’s puzzles are a way of doing key exchange between Alice and Bob without the need for a KDC**

(1) Alice creates lots of puzzles $P_i = E_{pi}$[“This is puzzle #X_i”, k_i]
where i = 1 .. 220, $|p_i|$ = 20 bits (weak), $|k_i|$ = 128 bits (strong)
X_i, p_i and k_i are chosen randomly and different for each i
(2) Alice sends all puzzles P_i to Bob
(3) Bob picks a random puzzle j ϵ {1 … 220} and solves P_j by brute force
(i.e. search on key p_j) -- this recovers X_j and k_j from the puzzle
(4) Bob sends X_j to Alice in the clear
(5) Alice looks up the index j of X_j (from a table) to get k_j
=> Alice and Bob now both share a secret key k

MERKLE'S PUZZLES

Alice makes 2^{20} puzzles p_1 p_2 p_3 p_4 … p_n

$p_i = E_{pi}$ i k_i

Alice sends the puzzles to Bob

Bob selects a random puzzle p_j retrieving (i, k_i)

Bob sends Alice a message saying he's retrieved the i^{th} puzzle

Alice looks up k_i from her selection of puzzles

Alice and Bob now use the shared key k_i for all future interaction

ATTACK ON MERKLE'S PUZZLES

Eve must break on average half the puzzles to find X_j (hence k_j)

– Time required to do so for 2^{20} puzzles = 2^{19} x 2^{19} = 2^{38}

If Alice and Bob can try 10,000 keys/second:

– It will take a minute for each of them to perform their steps (2^{19} for Bob)

– Plus another minute to communicate the puzzles on a 1.544MB (T1) link

With comparable resources, it will take Eve about a year to break the system

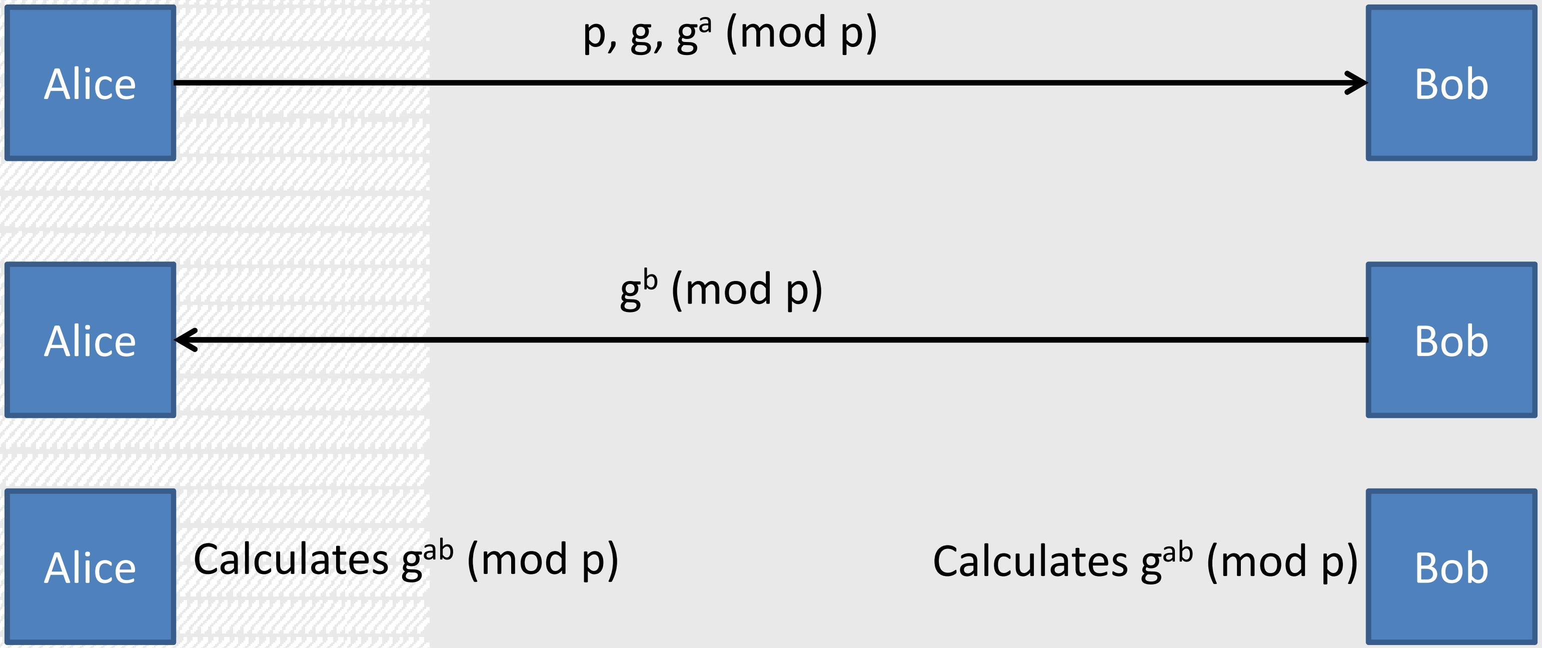
Note: Merkle's Puzzles uses a lot of bandwidth (impractical!)

DIFFIE-HELLMAN KEY EXCHANGE

Diffie-Hellman (Stanford, 1976)
Worldwide standard used in smart cards, SSL, etc.

Consider the finite field Z_p = <0, … p-1> where p is prime (p is 300 digits or longer)
Let $g \in Z_p$ (the generator)
(1) Alice : Alice chooses a random large integer $a \in Z_p$
(2) Bob : Bob choses a random large integer $b \in Z_p$
(3) Alice → Bob : Alice sends Bob $g^a \pmod p$
(4) Bob → Alice : Bob sends Alice $g^b \pmod p$
(5) Alice and Bob : compute g^{ab}
: Alice computes $(g^b)^a = g^{ab} \pmod p$
: Bob computes $(g^a)^b = g^{ab} \pmod p$
=> Alice and Bob now share secret g^{ab}

DIFFIE-HELLMAN KEY EXCHANGE



Eve calculates **???**
(only knows p, g, g^a, g^b

STRENGTH OF DIFFIE-HELLMAN

- **The strength of Diffie-Hellman is based upon two issues:**

– given p, g, g^a, it is difficult to calculate a (the discrete logarithm problem)

– given p, g, g^a, g^b it is difficult for Eve to calculate g^{ab} (the Diffie-Hellman problem)

– we know that DL ⇒ DH but it is not known if DH ⇒ DL.

- **Essentially, the strength of the system is based on the difficulty of factoring numbers the same size as p**
- **The generator, g, can be small**
- **Do not use the secret g^{ab} directly as a session key**

– it is better to either hash it or use it as a seed for a PRNG – not all bits of the secret have a flat distribution

REFERENCES

Handbook of Applied Cryptography

– read §1, §2-2.4.4, §2.5 - 2.5.3

Stallings (3rd Ed)

– 6.3 – 6.4