Key Management

Luke Anderson

luke@lukeanderson.com.au

13th April 2018

University Of Sydney

Overview

1. **Crypto-Bulletin**

2. **Key Exchange**

2.1 Preliminaries

3. **Key Distribution Centre**

4. **Merkle's Puzzles**

5. **Diffie Hellman**

5.1 Strength of Diffie-Hellman

Crypto-Bulletin

Crypto-Bulletin

Tearing New Holes into Intel/iPhone Cellular Modems

($\leqslant$ iOS 11.2.6)

https://comsecuris.com/blog/posts/theres_life_in_the_old_dog_yet_tearing_new_holes_into_inteliphone_cellular_modems/

Georgia Passes Anti-Infosec Legislation

https://www.eff.org/deeplinks/2018/03/georgia-passes-anti-infosec-legislation

Don't Give Away Historic Details About Yourself

https://krebsonsecurity.com/2018/04/dont-give-away-historic-details-about-yourself/

Key Exchange

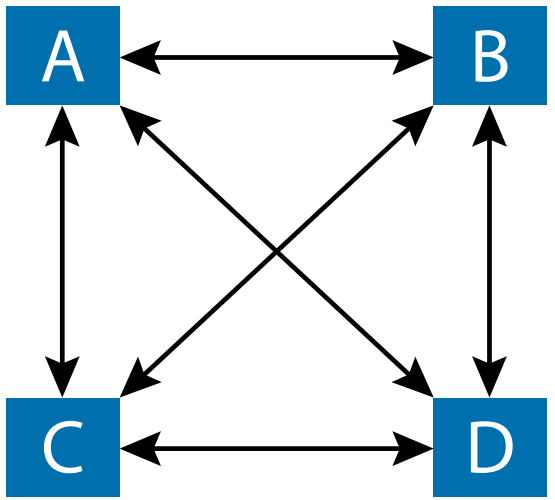
Key Management

Suppose we have a symmetric key network where Alice, Bob, Carol and Dave want to talk to each other.

For secure communication with n parties, we require:

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

Key distribution and management becomes a major issue!



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

Definitions

Key Establishment is the 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 key relationships between parties, including replacing older keys with newer ones. Includes:

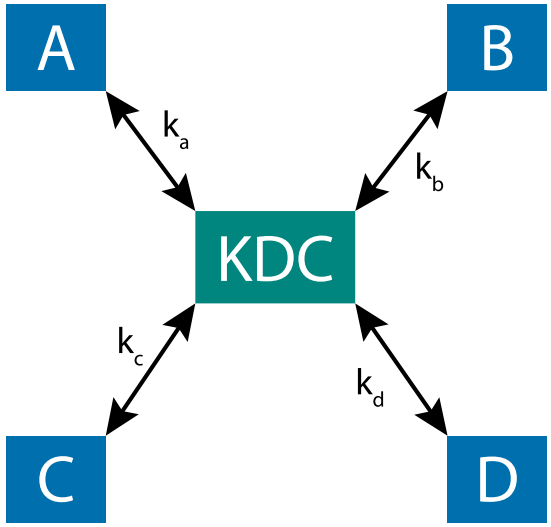
- Key agreement
- Key transport

Key Distribution Centre

Key Distribution Centre: Naïve

Protocol:

1. Alice $\rightarrow$KDC
   `I want to talk to Bob`
2. KDC $\rightarrow$Alice
   - KDC chooses random k_{ab}
   - Returns:
     $E_{k_a}(k_{ab}), E_{k_b}(k_{ab}, \text{“for talking to Alice”})$
3. Alice decrypts $E_{k_a}(k_{ab})$ to get k_{ab}
4. Alice $\rightarrow$Bob
   $E_{k_b}(k_{ab}, \text{“for talking to Alice”})$
5. Bob decrypts using k_b to get k_{ab}
6. Alice & Bob now share k_{ab}

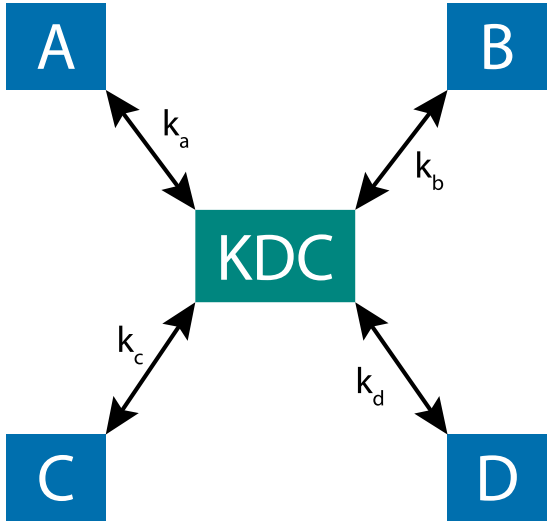


A Key Distribution Centre

Key Distribution Centre: Naïve

Problems:

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



A Key Distribution Centre

Merkle's Puzzles

Merkle's Puzzles

Merkle's Puzzles are a way of doing key exchange between Alice and Bob without the need for a third party.

1. Alice creates N puzzles $P_1, P_2, \ldots, P_N$, of the form
$$P_i = E_{p_i}(\text{“This is puzzle \#}X_i\text{”}, k_i)$$
   - $N \approx 200$
   - $|p_i| \approx 20$ bits (weak)
   - $|k_i| \approx 128$ bits (strong)
   - X_i, p_i, and k_i are chosen randomly and *different* for each i.
2. Alice sends all puzzles to Bob: $P_1, P_2, \ldots, P_N$.
3. Bob chooses a random puzzle P_j for some $j \in \{1, 2, \ldots, N\}$.
   - Finds p_j by brute force (key space search)
   - Recovers k_j and X_j
   - Bob sends X_j to Alice unencrypted
4. Alice looks up the index of X_j to find they key k_j chosen by Bob.
5. Alice & Bob both share key k_j

Attacking Merkle's Puzzles

On average, Eve must break half of the puzzles to find which puzzle contains X_j (and hence obtain k_j).

So for 2^{20} puzzles, Eve must try 2^{19} puzzles on average.
Each puzzle is encrypted with the 20 bit key p_i. Eve must search, on average, half of the key space: 2^{19}. $2^{19} \times 2^{19} = 2^{38}$

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

- It will take about 1 minute for each to perform their steps
  Alice to generate, and Bob to break $p_j = 2^{19}$ keys
- Plus another minute to communicate all the puzzles over ADSL

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

Diffie-Hellman Key Exchange

Diffie-Hellman key exchange (Stanford, 1976) is a protocol for establishing a cryptographic key using mathematical tricks. It is a worldwide standard for use in SSL, smartcards, etc.

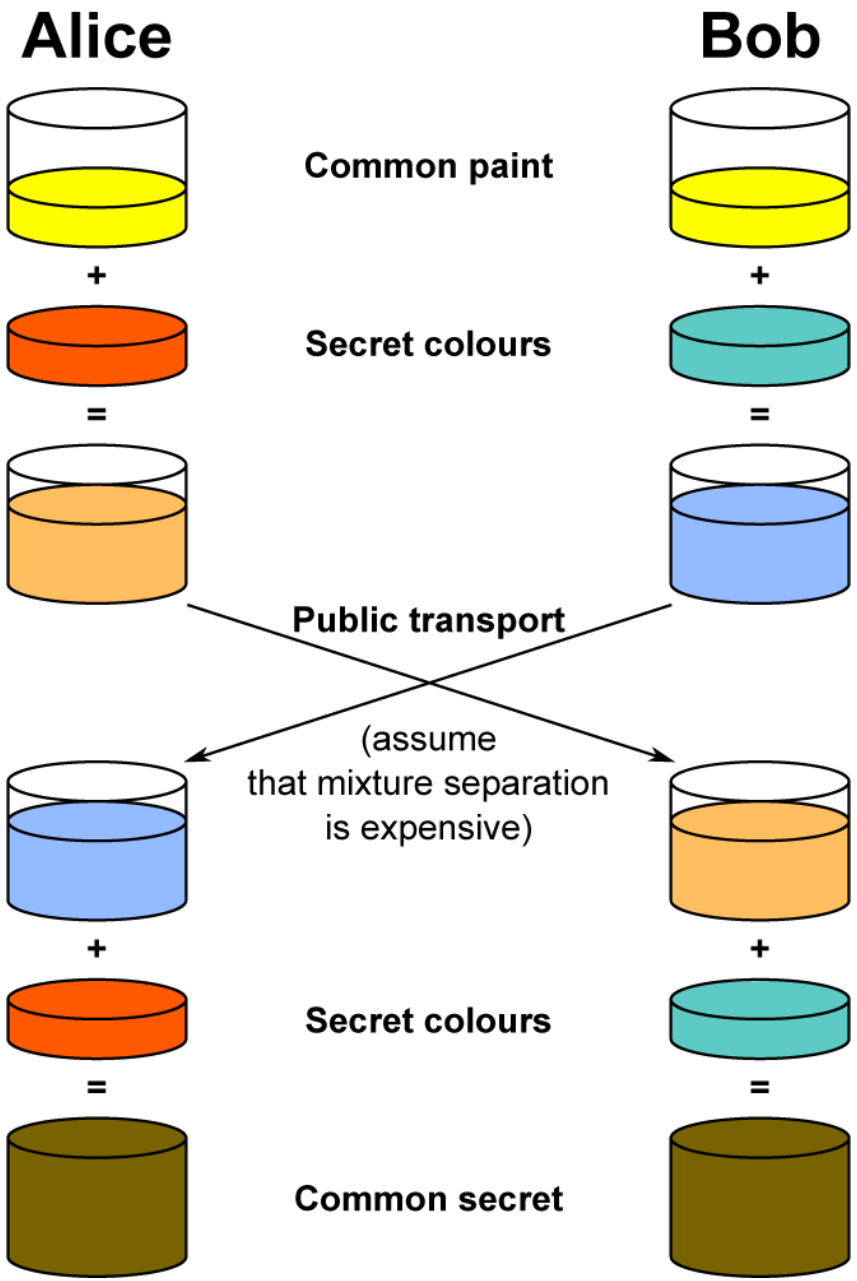
The *rough idea* is this: (details later)

1. Alice and Bob agree on some number g.
2. Alice generates a random number a, and sends g^a to Bob.
3. Bob generates a random number b, and sends g^b to Alice.
4. Alice and Bob can each compute g^{ab}, their shared secret.

An eavesdropper only has g^a, g^b, and g. *Assuming that taking logarithms is hard*, they cannot recover a or b.

Next lecture: the maths behind making logarithms hard.

Diffie-Hellman Key Exchange



The 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 to calculate g^{ab}
  (*the **Diffie-Hellman problem***)
- We know that $\mathrm{DL} \Rightarrow \mathrm{DH}$ but it is not known that $\mathrm{DH} \Rightarrow \mathrm{DL}$.

The Strength of Diffie-Hellman

Essentially the security is based on number tricks:

- The strength is proportional to the difficulty of factoring numbers the same size as p.
- The generator (g) can be **small**.
- **Do not use the secret g^{ab} as a key!**

Protip: 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!