Train the Trainers

SSH and Public / Private Keys

Secure authentication with keys

- Verify you are who you say you are
- Avoid sending and using passwords
- Distribute “proof of identity” in a secure and simple manner.

Three important terms

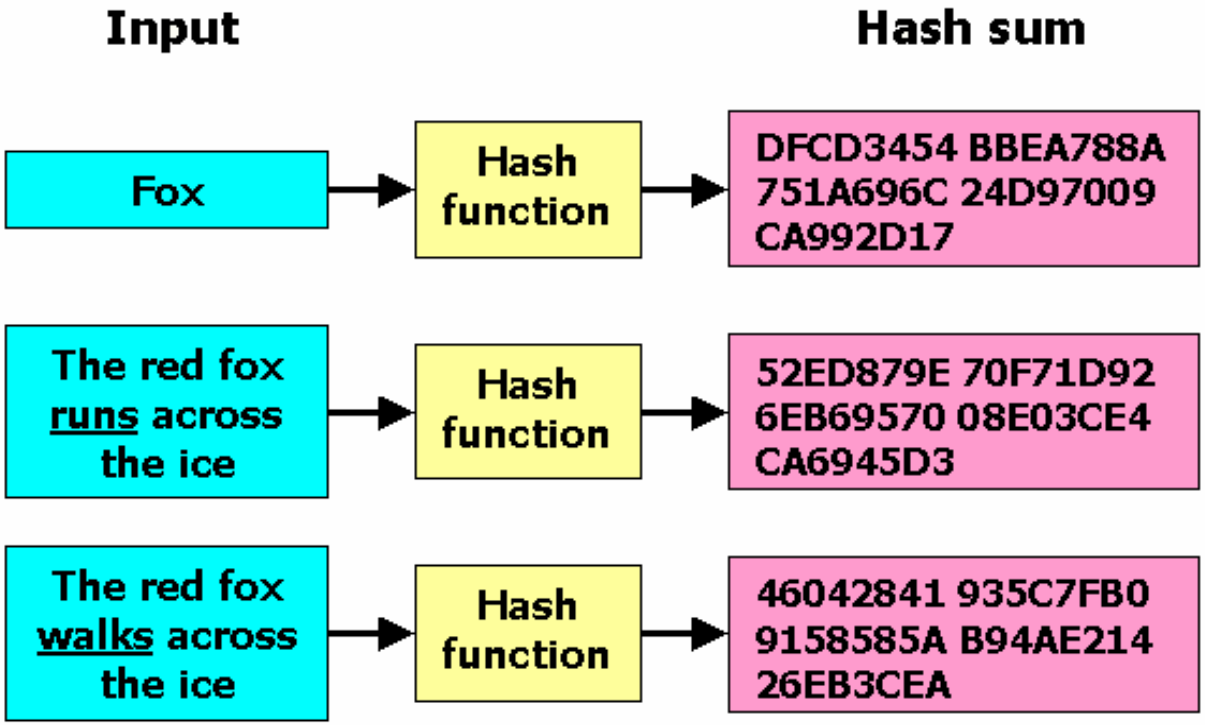
1. Hash
2. Cipher
3. public / private keys

1. Hash

One way hashing function

- Function, when applied to any amount of data produces the same, fixed-length result each time.
- The fixed-length result has several names, including:
  - ***hash***
  - *message digest*
  - *checksum*

“hash” continued



Note the significant change in the hash sum for minor changes in the input. Note that the hash sum is the same length for varying input sizes. This is extremely useful.

*Image courtesy Wikipedia.org.

2. Cipher

Cipher

In cryptography, a cipher (or cypher) is an algorithm for performing encryption or decryption — a series of well-defined steps that can be followed as a procedure.

i.e., some mathematical algorithm to scramble information in such a way that if you have the cipher you can unscramble it.

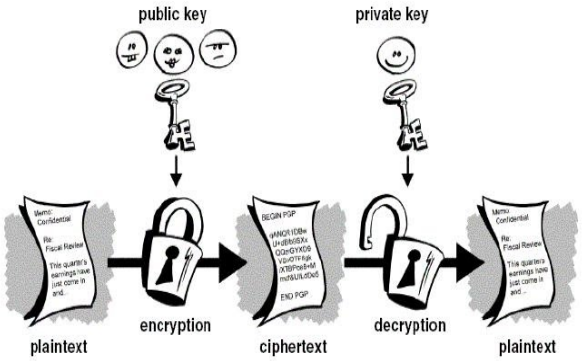
3. Public/Private Keys

We generate a key pair. One key is the *private key*, the other is the *public key*.

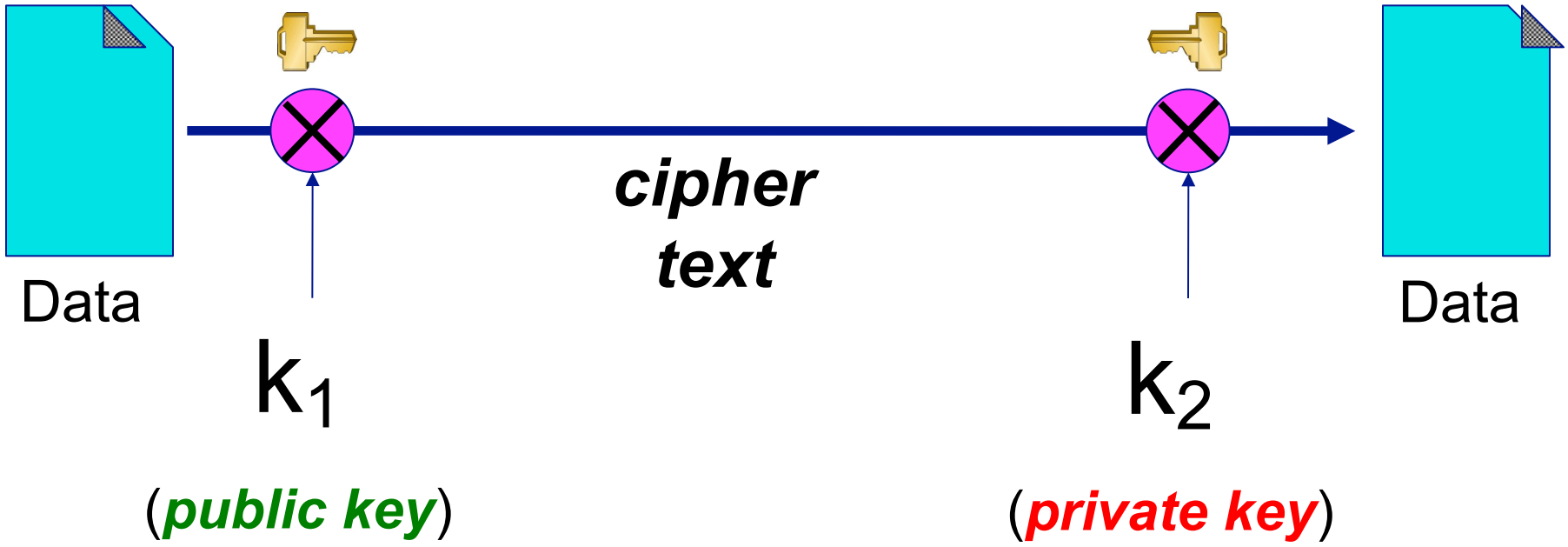
The *private key* remains secret and should be protected.

The *public key* is freely distributable. It is related mathematically to the private key, but you cannot (easily) reverse engineer the *private key* from the *public key*.

Use the *public key* to encrypt data. Only someone with the *private key* can decrypt.



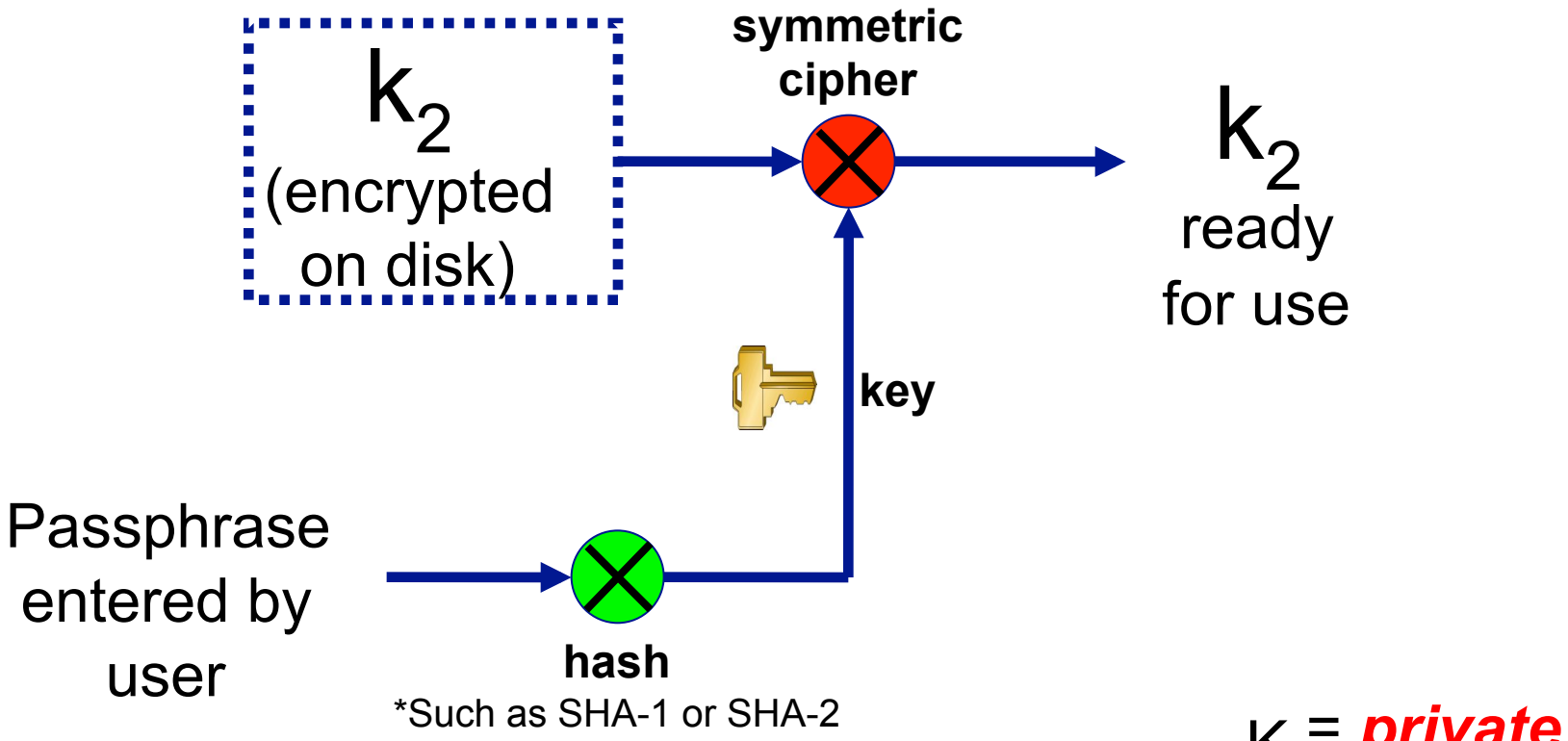
Example: Public/Private key pair



One key is used to encrypt the information,
a different key is used to decrypt it.

This is cool

Protecting the Private Key



K_2 = ***private key***

Checking the passphrase

Q.) How do you do this?

A.) It's very simple.

- Type in a passphrase for private key
- Hashing function and cipher are applied
- If result is correct (hash matches) passphrase is correct

Replacing password authentication

1. Your public key on distant server ➔ (Server)
2. Your private key on local machine ➔ (Client)
3. Server sends challenge data encrypted with your public key
4. User types in passphrase of private key on client to decrypt the challenge data.
5. Client returns challenge data as response, encrypted with *server's public key* (you get this the first time you connect*).
6. If challenge and response data match, user is allowed to connect to server.

What's next?

- Generate public/private key pairs on your laptops.
- Copy the public key to the *nsrc* and *root* accounts on your MacMinis.

SSH will see public key is available on the MacMini and use public/private key authentication instead of password authentication.

Questions

?