DNSSEC Cryptography Review

DNSSEC Tutorial

February 21, 2011
Hong Kong

Will.i.am Hervey Allen

DNSSEC and Cryptography

Key Concepts

- Public / Private keys
- Hashes
- Digital signatures
- Chain of trust
- HSMs (Hardware Security Module)

Are at the core of DNSSEC. If these do not make sense, then DNSSEC will not make sense.

Ciphers ==> ciphertext

We start with *plaintext*. Something you can read.

We apply a mathematical algorithm to the plaintext.

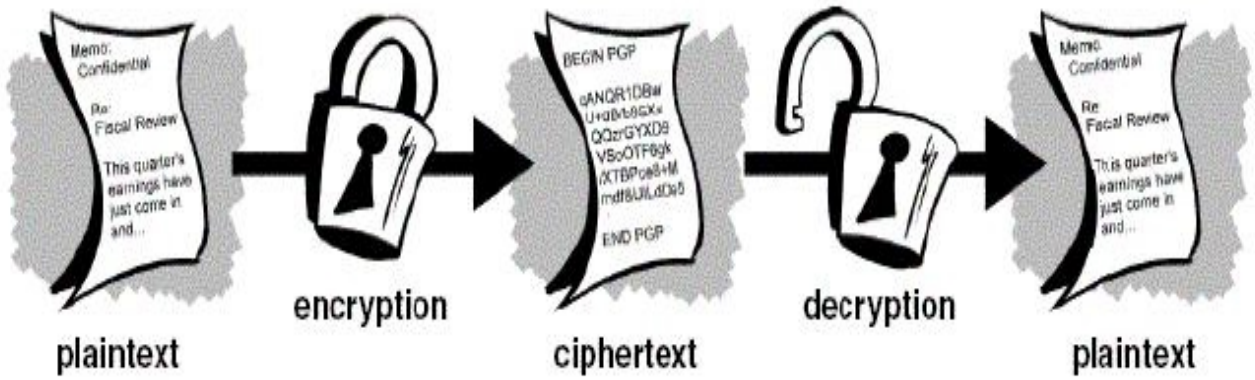
The algorithm is the *cipher*.

The plaintext is turned in to *ciphertext*.

Almost all ciphers were secret until recently.

Creating a secure cipher is *HARD*.

What it Looks Like



Keys

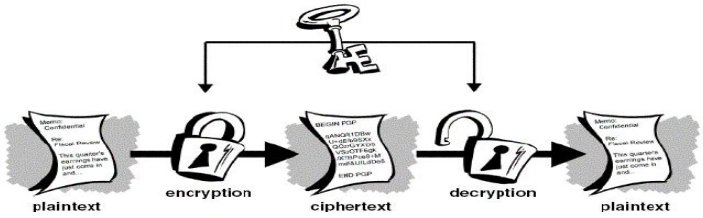
To create ciphertext and turn it back to plaintext we apply a key to the cipher.

The security of the ciphertext rests with the key. This is a *critical* point. If someone gets your key, your data is compromised.

This type of key is called a *private key*.

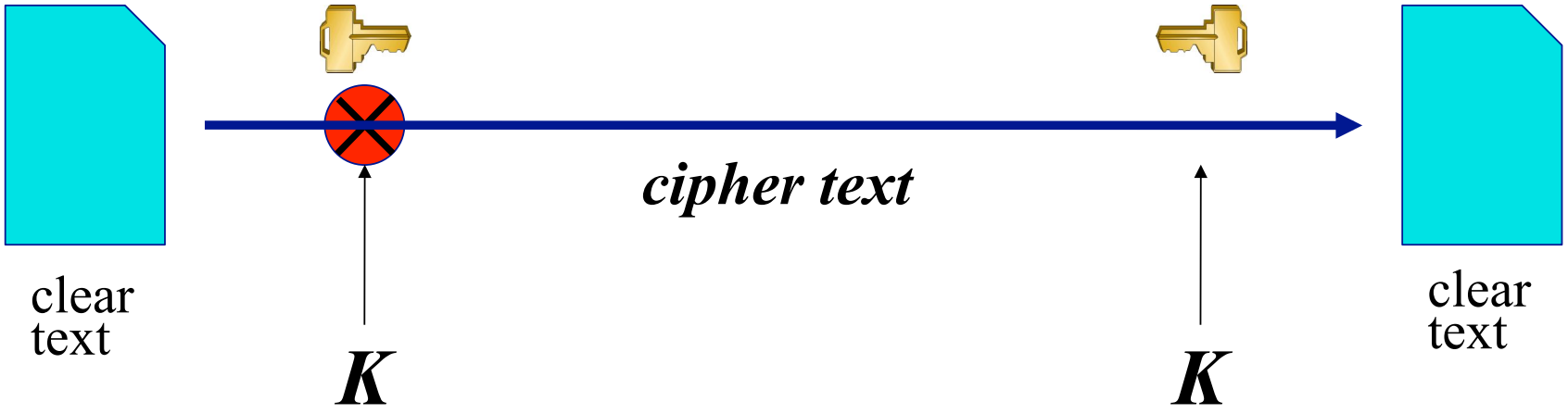
This type of cipher system is efficient for large amounts of data.

This is a *symmetric cipher*.



Symmetric Cipher

Private Key/Symmetric Ciphers



The same key is used to encrypt the document before sending and to decrypt it once it is received

Features of Symmetric Ciphers

- Fast to encrypt and decrypt, suitable for large volumes of data
- A well-designed cipher is “only” subject to brute-force attack; the strength is therefore directly related to the key length.
- Current recommendation is a key length of around 128 bits, for data protection around 20 years.*
- Problem - how do you distribute the keys?

*See http://www.keylength.com/ for a good and fun discussion.

Examples of Symmetric Ciphers

DES - 56 bit key length, designed by US security service

3DES - effective key length 112 bits

AES *(Advanced Encryption Standard) - 128 to 256 bit key length*

Blowfish - 128 bits, optimized for fast operation on 32-bit microprocessors

IDEA - 128 bits, patented (requires a license for commercial use)

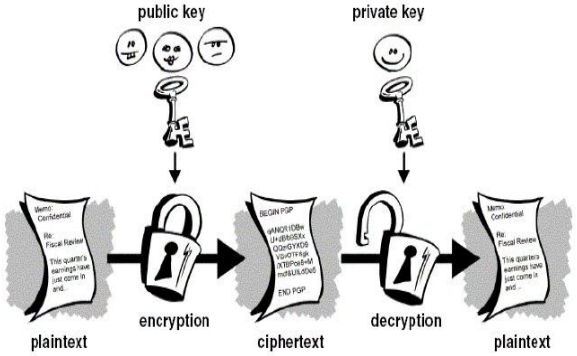
Public/Private Keys

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

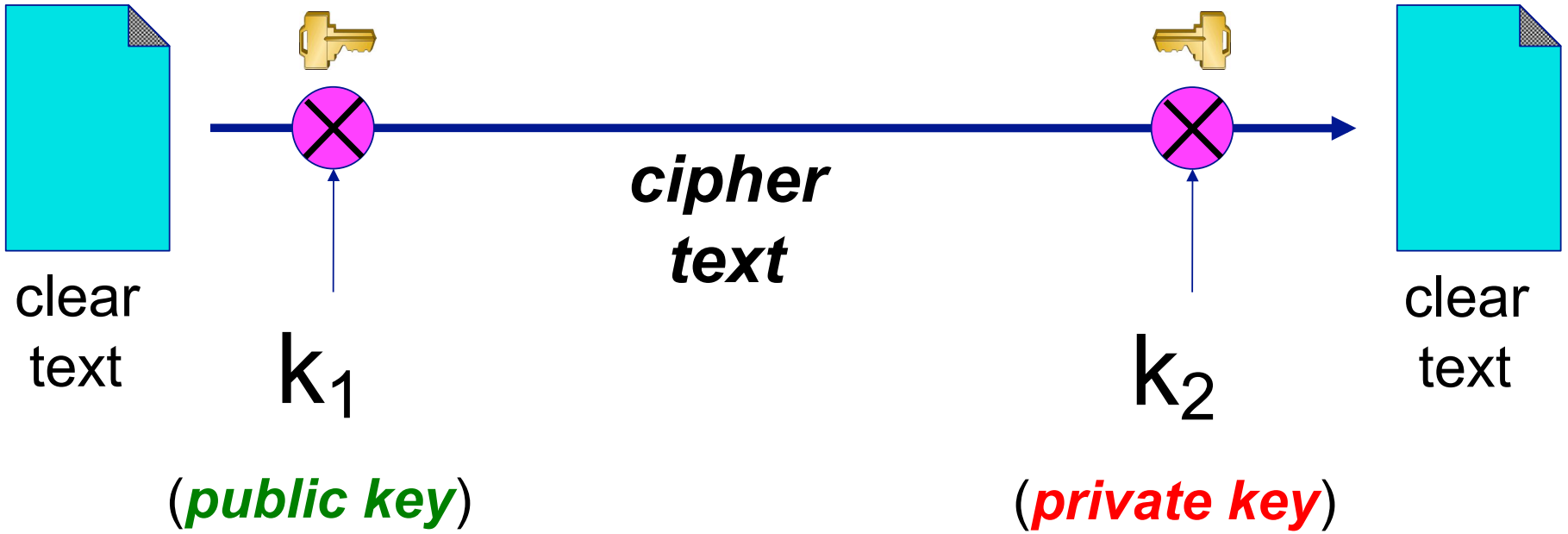
The *private key* remains secret and should be protected.

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 document,
a different key is used to decrypt it.

This is a big deal!

Less Efficient & Attackable

- Symmetric ciphers (one private key) are *much* more efficient. About 1000x more efficient than public key algorithms for data transmission!
- Attack on the public key is possible via chosen-plaintext attack. Thus, the public/ private key pair need to be large (2048 bits).

Remember, symmetric cipher attack is to steal the private key...

One-Way Hashing Functions

A mathematical function that generates a fixed length result regardless of the amount of data you pass through it. Generally very fast.

You cannot generate the original data from the fixed-length result.

Hopefully you cannot find two sets of data that produce the same fixed-length result. If you do this is called a *collision*.

Hashing Function Examples

- Unix ***crypt***() function, based on DES, 56 bits (***not secure)***
- ***MD5*** (Message Digest 5) - 128 bit hash (**deprecated**)
- ***SHA1*** (Secure Hash Algorithm) - 160 bits
- Until August 2004, no two documents had been discovered which had the same MD5 digest.
  - *No collisions have yet been found in SHA-1, but it is now known to be compromised and will likely be phased out in the next few years. See http://en.wikipedia.org/wiki/SHA for details.*
- Still no feasible method to create any document which has a given MD5 digest
- Currently SHA2* in use to be replaced by “SHA3” soon.

Hashing
One-Way Encryption

clear text

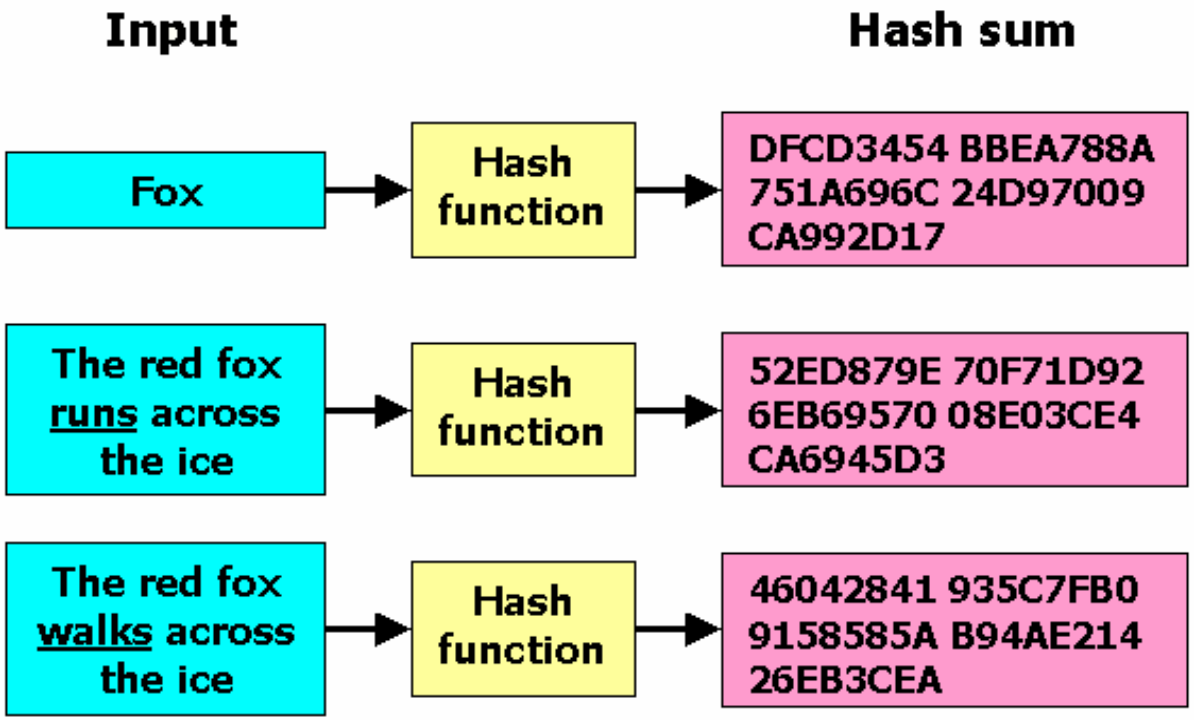
hashing function

Fixed length ***hash*** or ***message digest***

Munging the document gives a short ***message digest*** (hash). Not possible to go back from the digest to the original document.

Hashing

one-way encryption: another example



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.

One-Way Hashing Functions cont.

Applying a hashing function to plaintext is called *munging the document*.

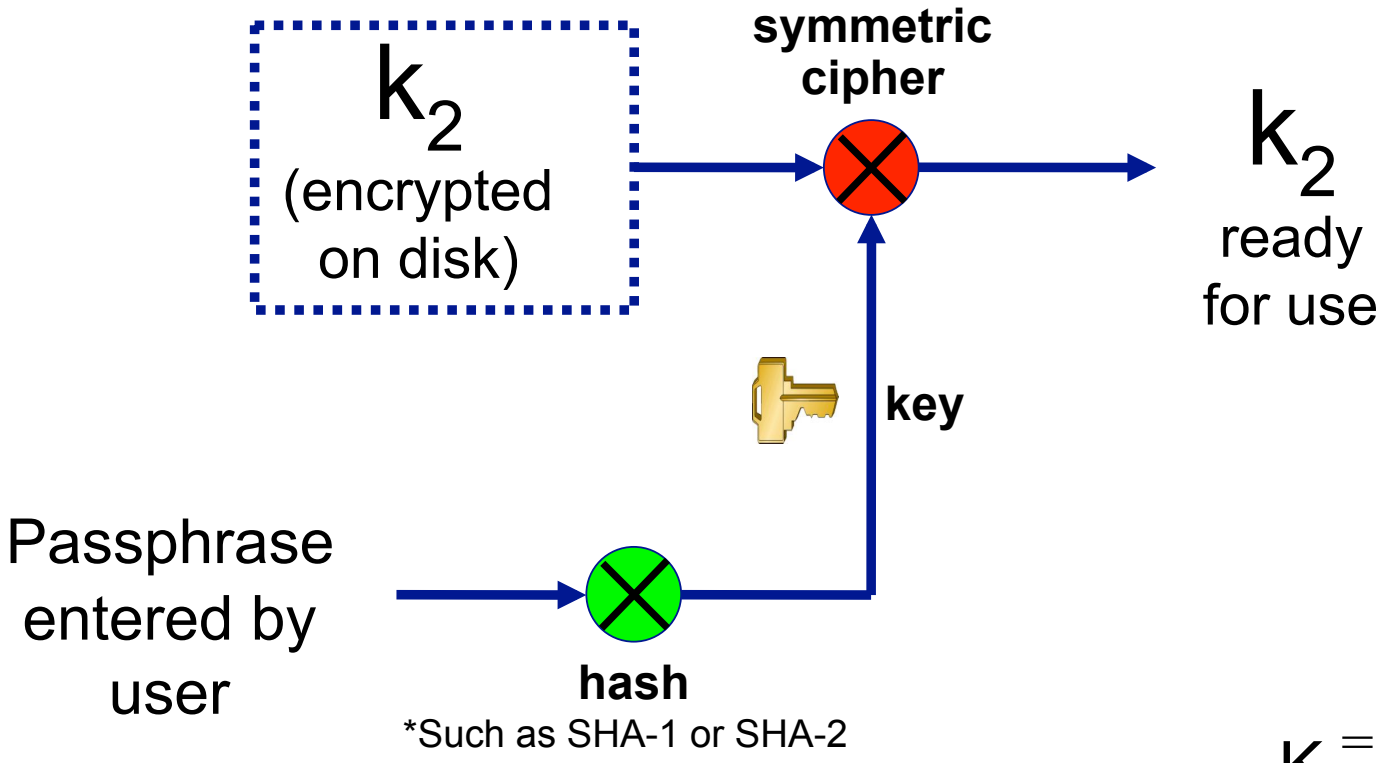
The fixed-length result is referred to as a *checksum*, *fingerprint*, *message digest*, *signature*, *digest*, *hash*, *hash sum*…

What use is this?

- You can run many megabytes of data through a hashing function, but only have to check 160* bits of information. A compact and *unique document signature*.*
- You can generate a *passphrase* for your data – such as your private key. If someone gets your private key, they still must know your passphrase to decrypt anything using your private key.
- This is how Unix, Linux and Windows protect user passwords (but not effectively).

* May increase after 2012 if a new SHA-3 algorithm is approved for use.

Protecting the Private Key



K_2 = ***private key***

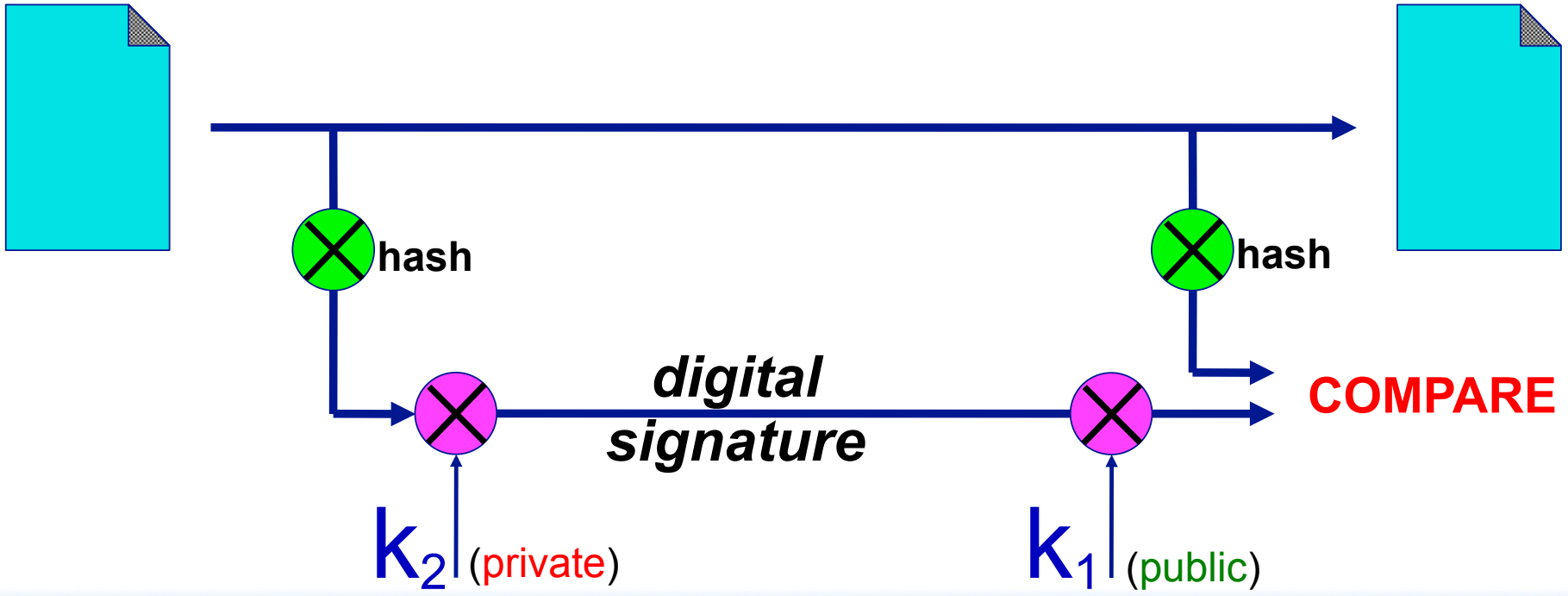
Digital Signatures

Let's reverse the role of public and private keys.
To create a digital signature on a document do:

- *Munge* a document.
- Encrypt the *hash* with your private key.
- Send the document plus the encrypted hash.
- On the other end munge the document *and* decrypt the encrypted message digest with the person's public key.
- If they match, the document is authenticated.

Digital Signatures cont.

Take a hash of the document and encrypt only that. An encrypted hash is called a "digital signature"



Uses for Digital Signatures

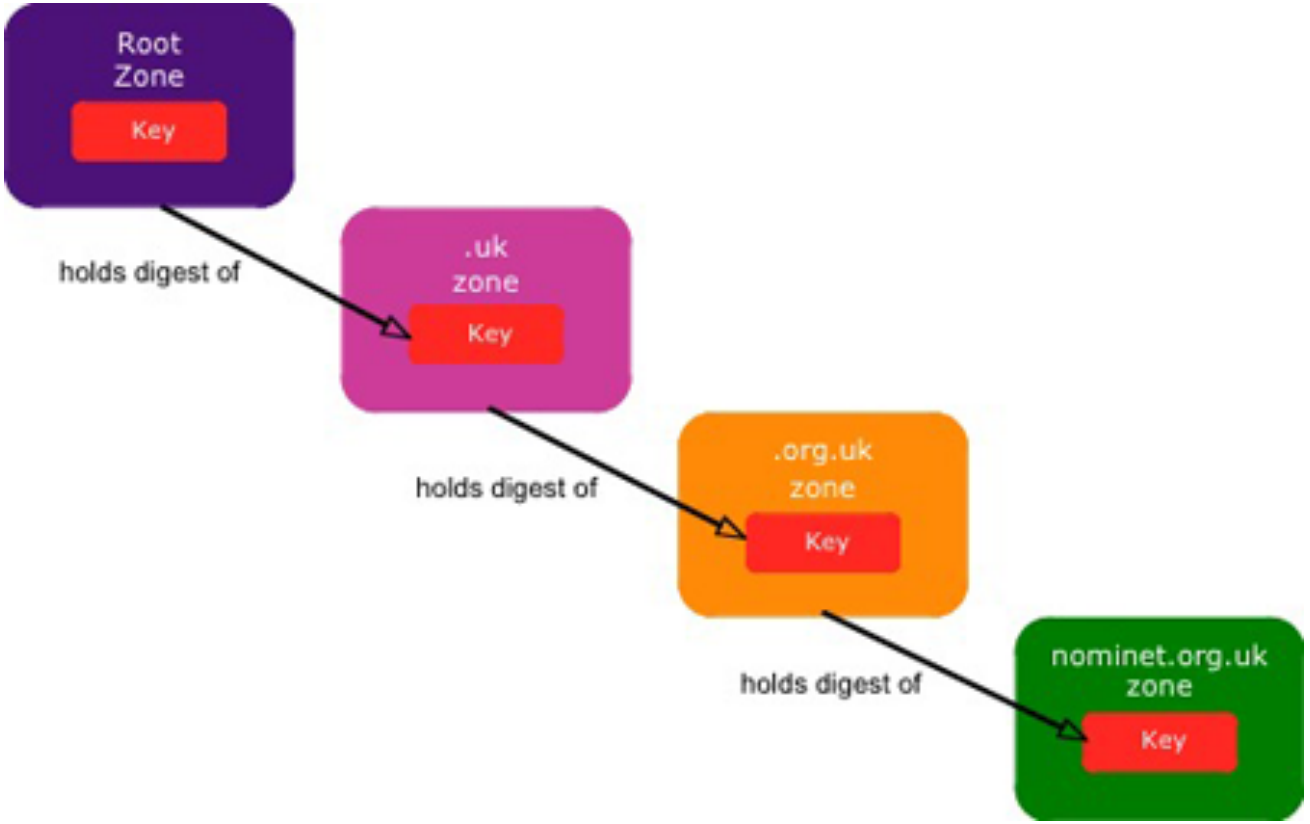
- E-commerce. An instruction to your bank to transfer money can be authenticated with a digital signature.
- A trusted third party can issue declarations such as "the holder of this key is a person who is legally known as Alice Hacker"
  - Like a passport binds your identity to your face
- Such a declaration is called a "certificate"
- Trusted third party is referred to as a CA, or *Certificate Authority*.
- And… we sign records in the DNS to prove they are authentic, unchanged and come from a trusted source.

Chain of Trust – Securing the DNS

1. Data authenticity and integrity by signing DNS data with a private key.
2. Publish the Public Key
3. Parent signs child's DNS data (hash of data)
4. Child signs DNS data with their private key and publish the Public key.
5. Repeat 1 to 4 down the DNS hierarch.

- Root zone's Public key and signed DNS data act as CA for the DNS.

Chain of Trust – Securing the DNS



HSMs:Hardware Security Modules

A hardware device that provides:

- A protected keystore:
  - Tamper-proof memory
  - Detection of tampering or key destruction
- Cryptographic processing in hardware
  - Accelerate asymmetric and symmetric processing
  - Support for AES, RSA, 3DES, AES ciphers
- An API for key extraction
  - Hashing may be done on the host

Why use an HSM?

- Protect the private keys (move to HSM)
- Separate Keystore from signing software
- Implements standards compliant security
  - Verifiable security such as FIPS 140-2

Requires features that show evidence of tampering, including tamper-evident coatings or seals that must be broken to attain physical access to the plaintext cryptographic keys and critical security parameters (CSPs) within the module, or pick-resistant locks on covers or doors to protect against unauthorized physical access.

What's an HSM look like?

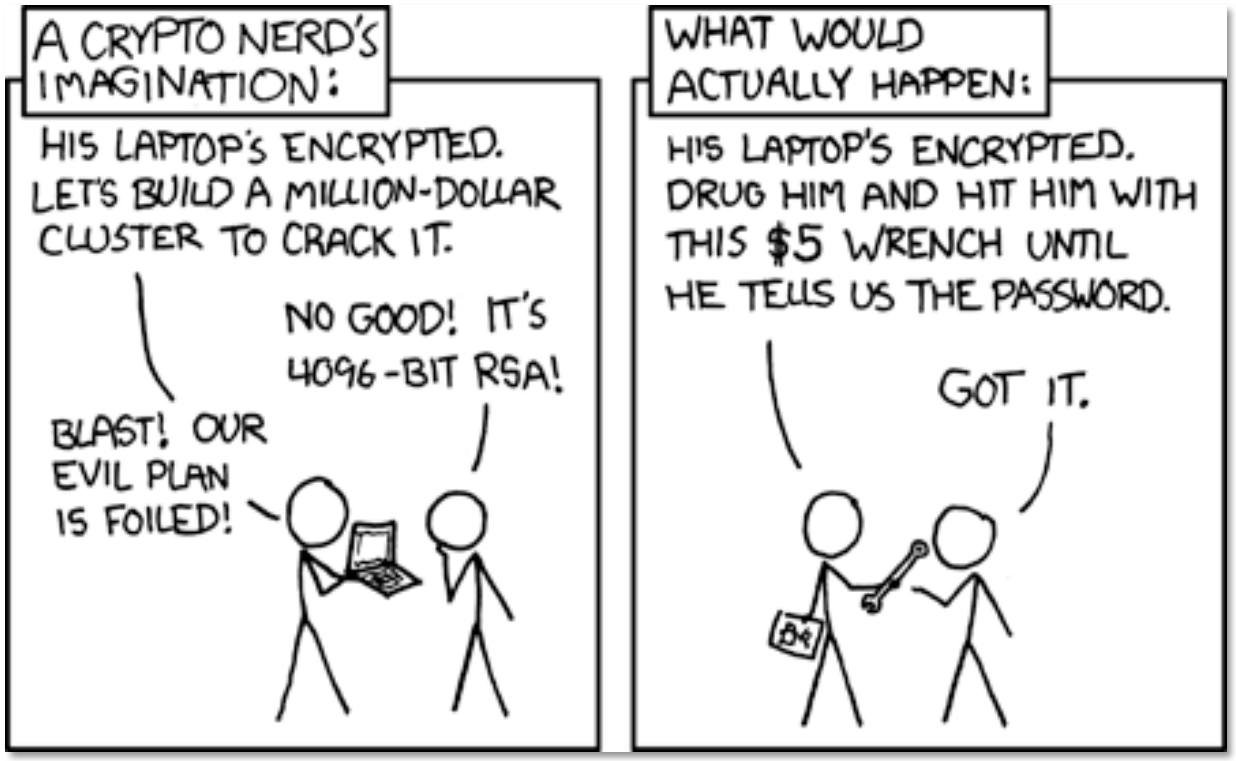
- **Sun Crypto Accelerator 6000**
- Supports: Solaris, RHEL, SUSE
- PCI interface
- Inexpensive (USD $1,400)

- **AEP Keyper 9720**
- Supports: Solaris, Linux, Windows
- Ethernet interface to PC
- Expensive (USD $20,000+)

- **Software HSM**
- Supports: UNIX
- Software interface to PC
- Free

SoftHSM

Attacking the Cipher… Reality…?



Summary

Public / Private keys

- Give out public key. Encrypt with this. Decrypt with private key.

Hashes

- Create a unique, fixed-length signature (hash) of a data set.

Digital signatures

- Munge document, encrypt hash with private key. Decrypt with public key.

Chain of trust

- Verify parent zone signature (RRSIG) with public key of parent.

HSMs (Hardware Security Module)

- Store private key in hardware implementing security standards.