DNSSEC Tutorial: Public / Private Key Refresher

Hervey Allen
Phil Regnauld

15 June 2009
Papeete, French Polynesia

http://nsrc.org/workshops/2009/pacnog5/meeting/dnssec/

DNSSec and Cryptography

Three Key Concepts

- Public / Private keys
- Message digests, checksums, hashes
- Digital signatures

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

Ciphertext

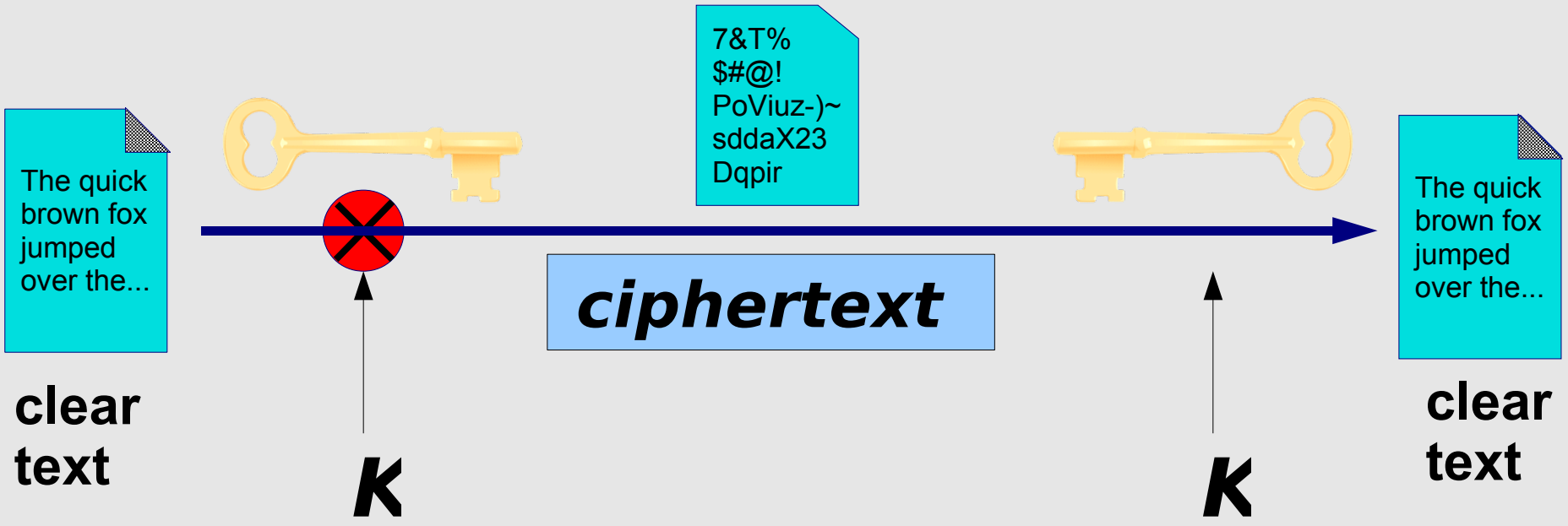
- We start with *plaintext*. Something you can read.
- We apply a mathematical algorithm to the *plaintext*.
- The algorithm is the *cipher*.
- The *plaintext* is turned in to *ciphertext*.
- Almost all *ciphers* were secret until recently.
- Creating a secure *cipher* is *HARD*.

Keys

- To create *ciphertext* and turn it back to *plaintext* we apply a *key* to the *cipher* on both ends.
- The security of the *ciphertext* rests with the *key*. This is a *critical* point. If someone obtains your *key*, your data is compromised.
- This type of single key use is part of a *symmetric cipher*.

Symmetric Cipher

Single Key/Symmetric Ciphers



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

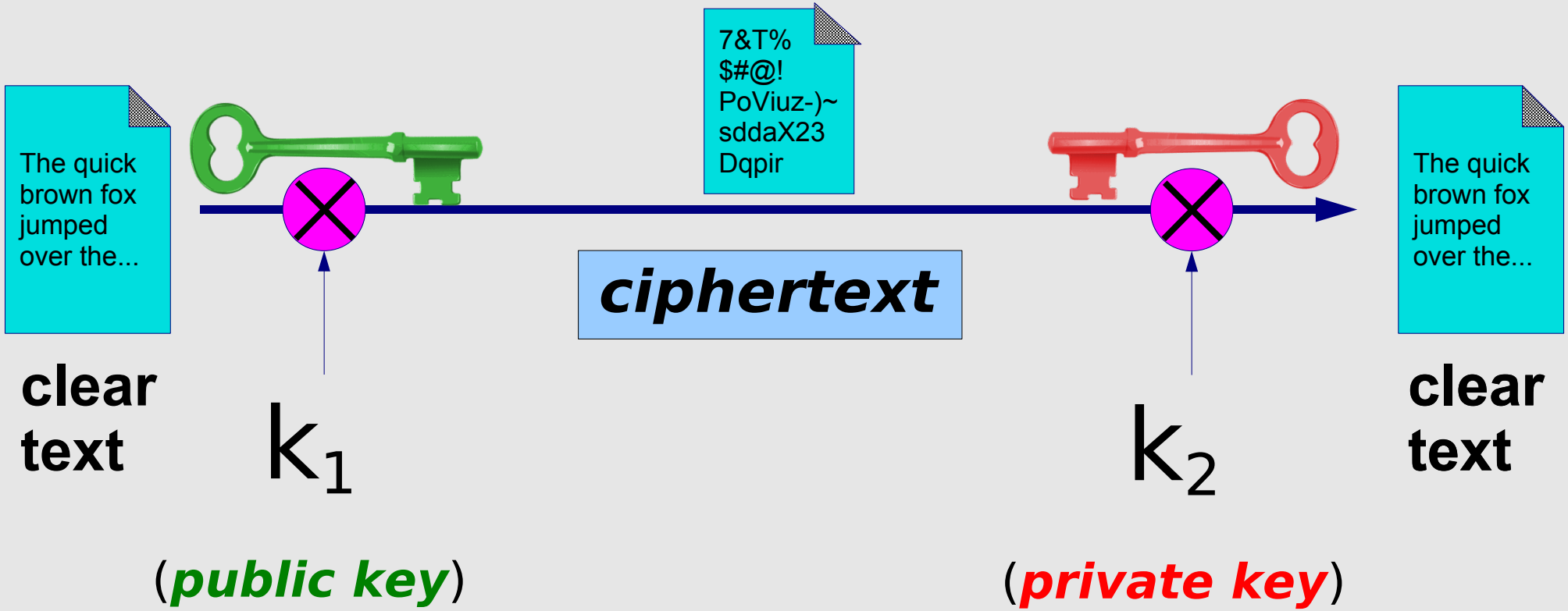
The Big Question...

How
do
you
distribute
the
keys
securely?

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 the encrypted data.

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!

Issues

- For larger data transmissions than used in DNSSEC we use *hybrid systems*.

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

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, thus the term “one-way”.
- Hopefully you cannot find two sets of data that produce the same fixed-length result. If you do, this is called a *collision*. (Example, md5).

- The fixed length result is known as a ***Message Digest* or a *checksum* or a *hash*.**

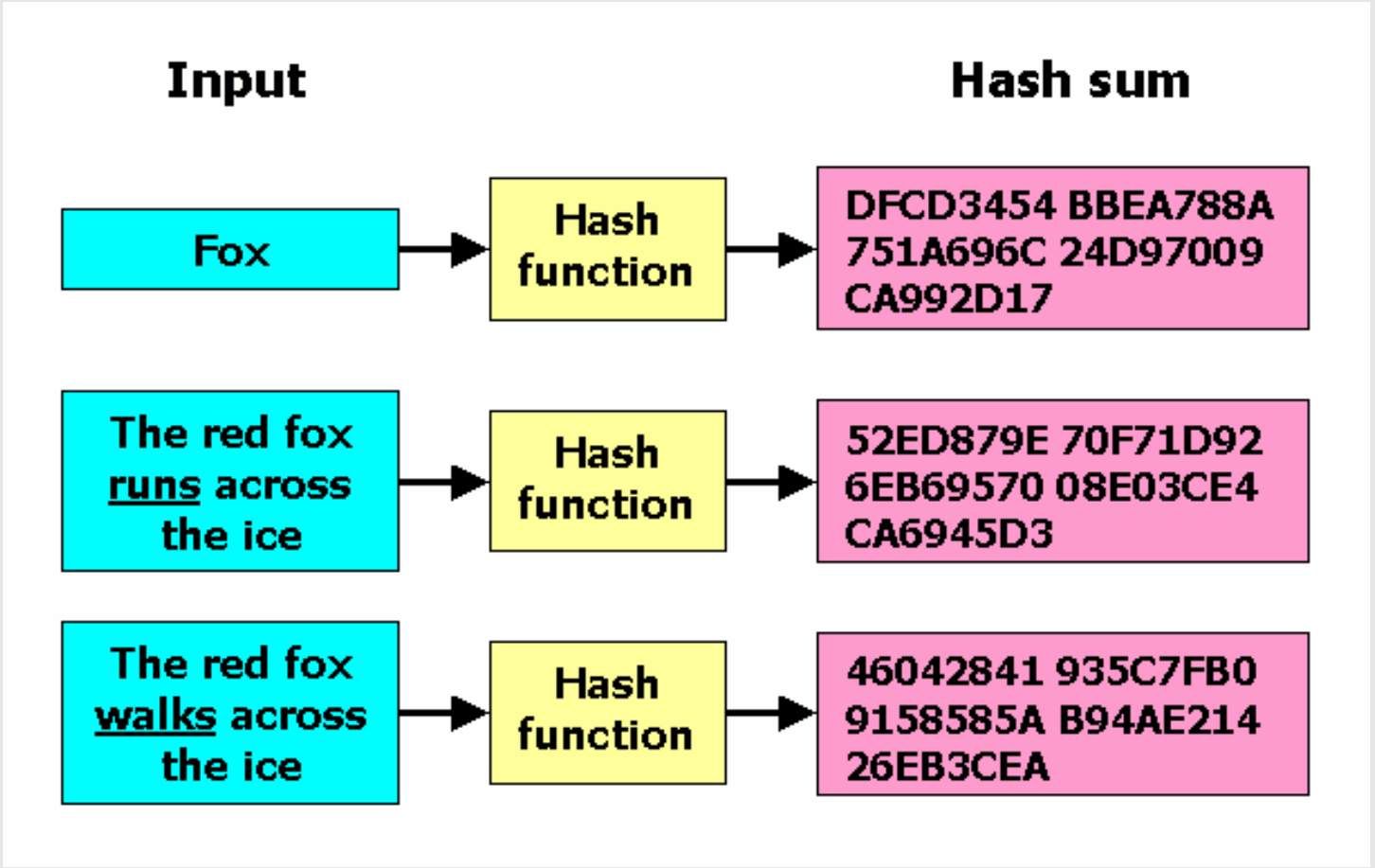
One-Way Hashing Functions cont.

- Applying a hashing function to plaintext is called *munging a document*.
- The fixed-length result is referred to as a *checksum, message digest* or *hash.*

Some popular hashing functions include:

- **md5**: Outputs 128 bit result. Fast. Collisions found. http://www.mscs.dal.ca/~selinger/md5collision/
- **sha-1**: Outputs 160 bits. Slower. Collisions in 2^{63}.
- **sha-2**: Outputs 224-512 bits. Slower. Collisions expected (2^{80} attack).
- **sha-3**: TBA: Currently in development via a new *NIST Hash Function Competition.*

Hashing
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.

What use is this?

There are several:

- Create passphrases for private keys.
- Passwords (in Linux, Unix and Windows).
- You can run many megabytes of data through a hashing function, but only have to check a fixed number of bits of information (160-512 bits). This can be used to create a *digital signature*.

Digital Signatures

Reverse the role of public and private keys.

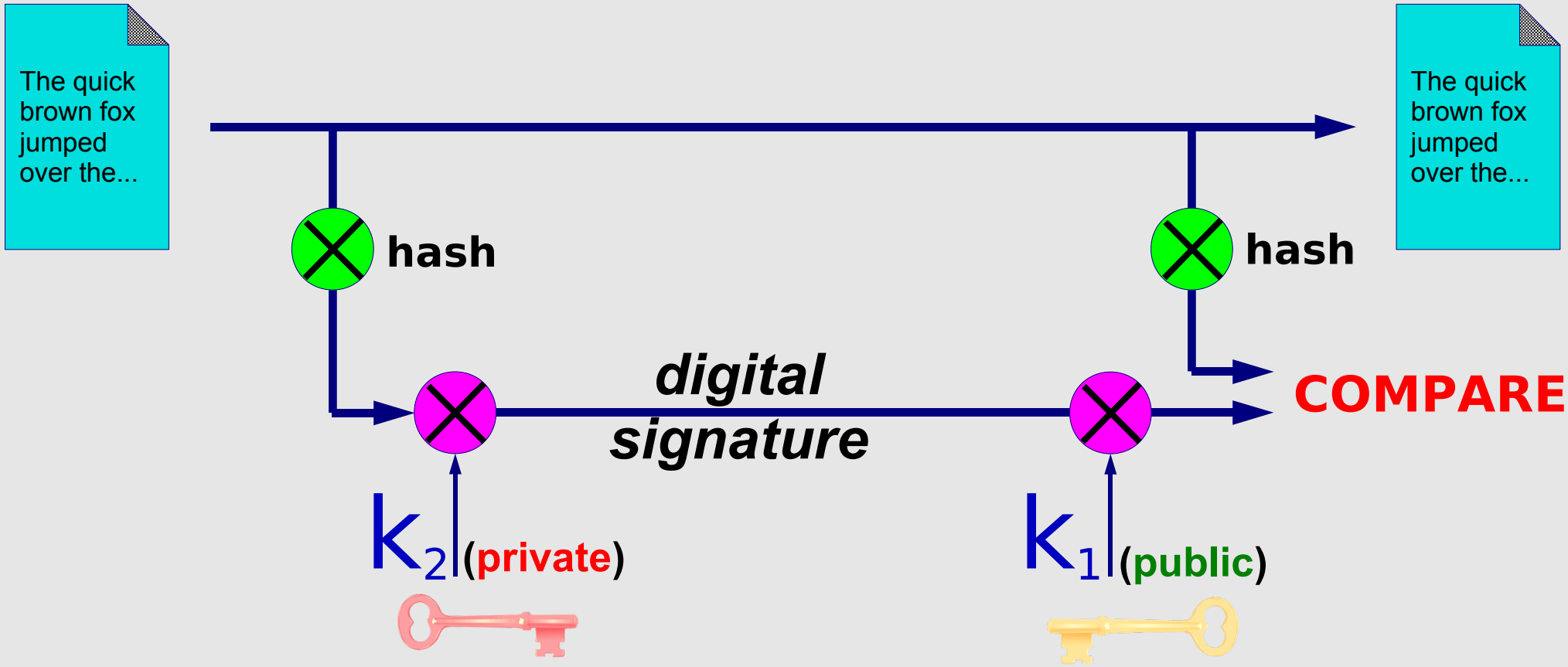
To create a digital signature on a document do:

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

This process creates a *digital signature*. (ta da!)

When Authenticating:

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



Conclusion

- Public / Private keys
- Message digests, checksums, hashes
- Digital signatures

Are at the core of DNSSEC. If these do not make sense, then DNSSEC will not make sense. Well, maybe not... ;-)