Some fundamental security concerns...

Confidentiality - could someone else read my data?

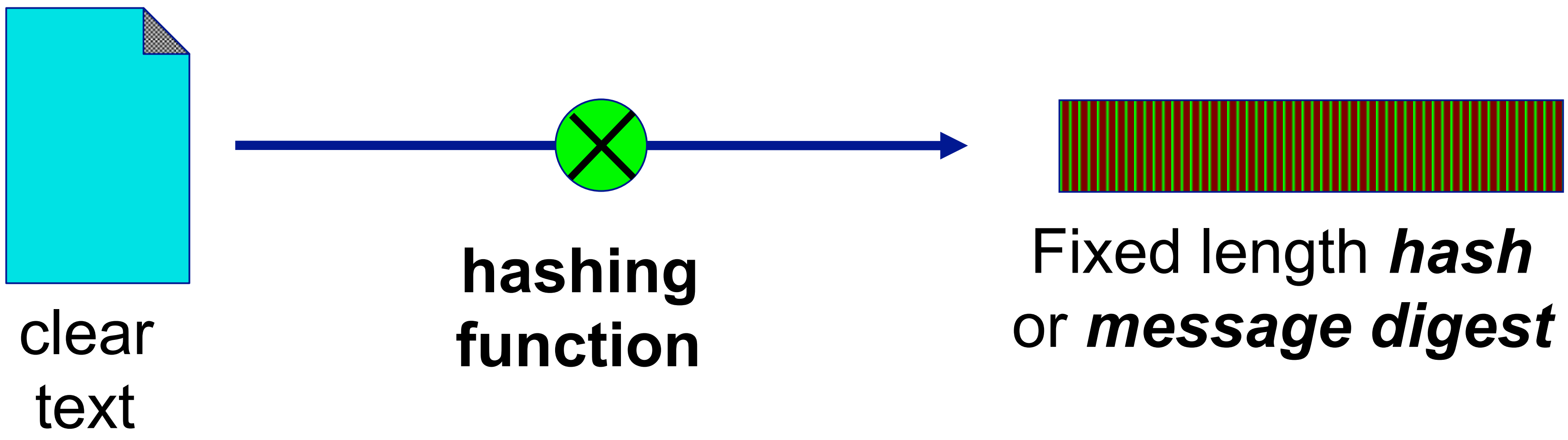
Integrity - has my data been changed?

Authentication - is this who they claim to be?

*Cryptography offers **genuinely secure** solutions to these problems*
We'll look briefly at four main components

Hashing

One-Way Encryption



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

Examples of Hash algorithms

MD5 - 128 bits of output

SHA1 - 160 bits

RIPEMD-160 - 160 bits

SHA256 - 256 bits

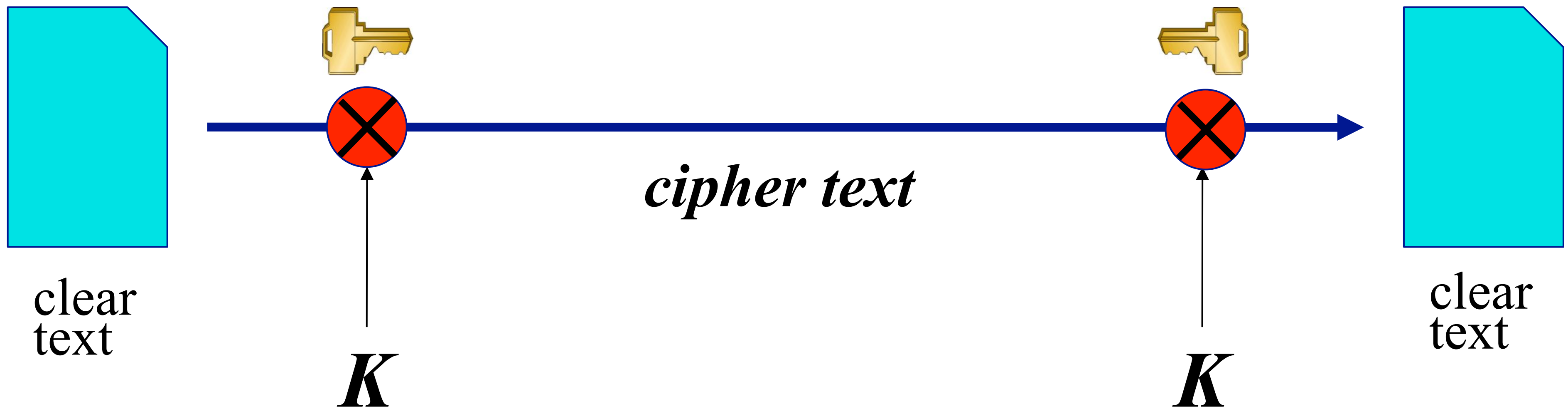
Properties of Cryptographic Hashes

- Running the same hash algorithm on the same document always gives the same result
- It is infeasible to modify the document whilst keeping the hash the same - or even to find *any* other document with the same hash
- Hence a powerful check of **integrity**
- Important: MD5 is now BROKEN!
  - all it takes is 3 days and 200 playstation3's *
  - SHA1 not yet, but has known weaknesses

* Google for "MD5 considered harmful today"

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

Examples of Symmetric Ciphers

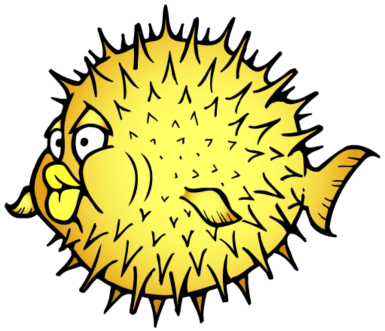
DES - 56 bit key length

3DES - effective key length 112 bits

RC4 (Arcfour) - 128 bits

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

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



Properties of Symmetric Ciphers

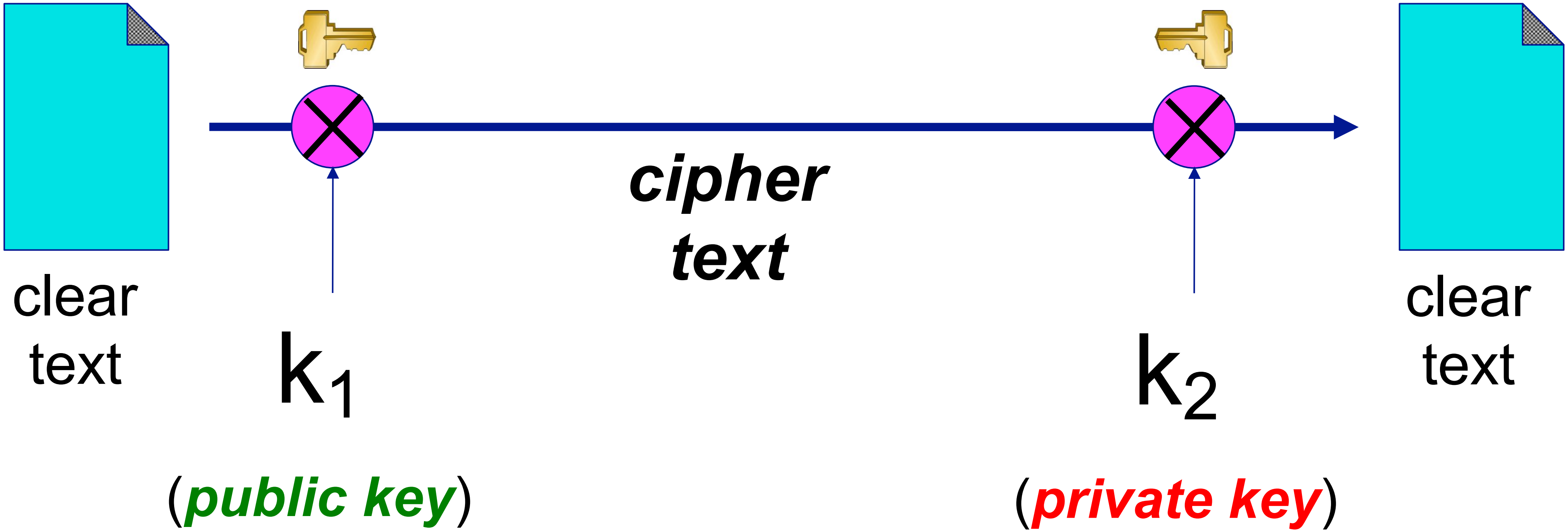
- Provides confidentiality: infeasible to decrypt data without knowing the secret key K
- Provides integrity: a small change to the ciphertext will cause it to decrypt to garbage
- Provides authenticity: if I can decrypt the data with my key K, I know it must have been encrypted by someone who knows K
- Fast to encrypt and decrypt, suitable for large volumes of data

Attacks on Symmetric Ciphers

- Good ciphers resist attacks on the algorithm; brute-force attack is directly related to the key length.
- Current recommendation is a key length of 90+ bits, for data protection of 20 years.*
- Relies entirely on secrecy of the key. How can you distribute it securely to your peer without it being intercepted by an attacker?
- Use a hash to convert a passphrase into a value suitable for a key (passphrase easier to remember)

*See http://www.keylength.com/ for a collection of recommendations

4. Public key cipher



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

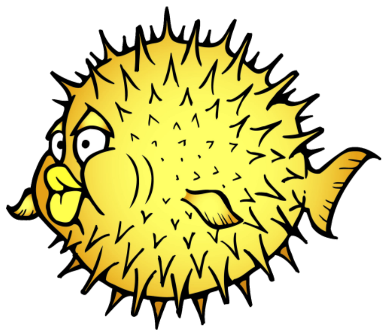
This is a big deal!

Examples of Public Key Ciphers

RSA - named after the three inventors

ElGamal - was popular while RSA was patent-protected, forms basis of DSA

Elliptic Curve - newer, stronger, not widely used yet



Properties of Public Key Ciphers

- The keys are mathematically related
  - Easy to convert private key into public key
  - Infeasible to convert public key into private key
  - You can safely post your public key anywhere!! (That's why it's called "public")
- Can provide confidentiality: encrypt with public key, decrypt with private key
- Can provide authenticity: encrypt with private key, decrypt with public key

Example application: gpg

gpg lets you:

- generate a public/private key pair
- encrypt messages with any public key, and/or
- sign messages with your private key

Used for sending encrypted E-mail, verifying integrity of software packages, etc

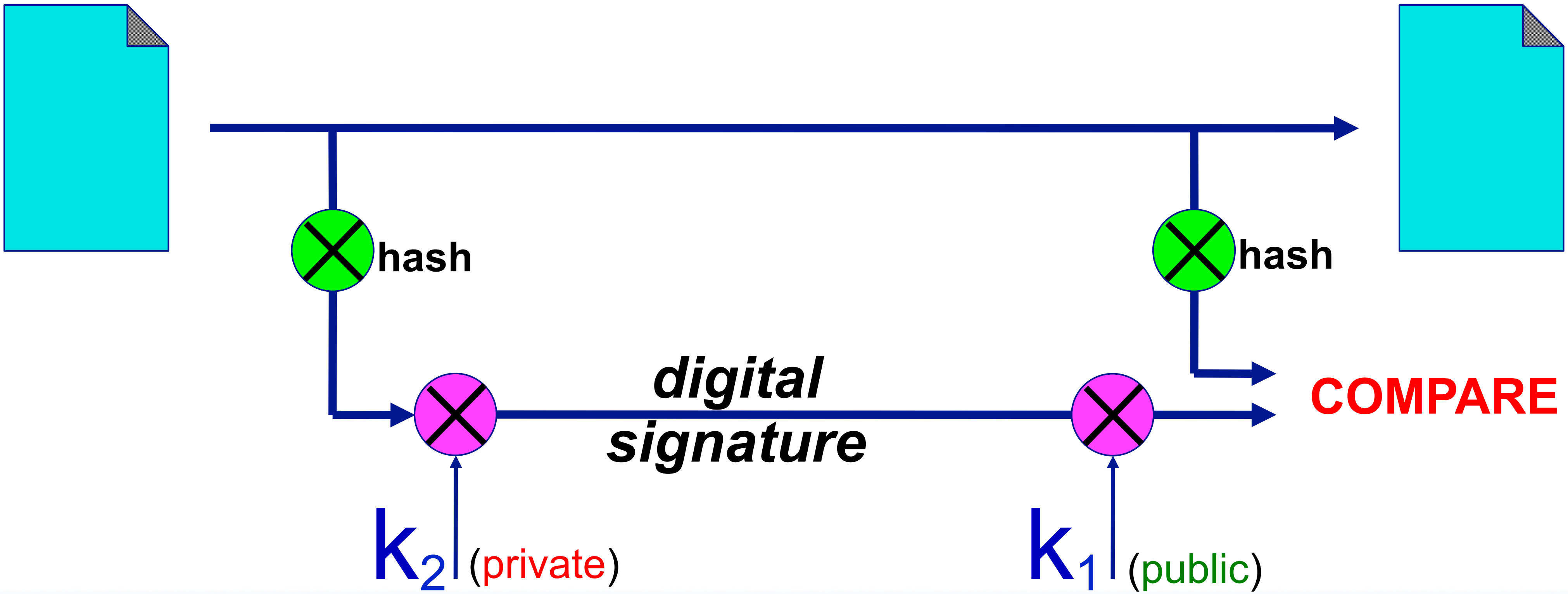
Digital Signatures

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

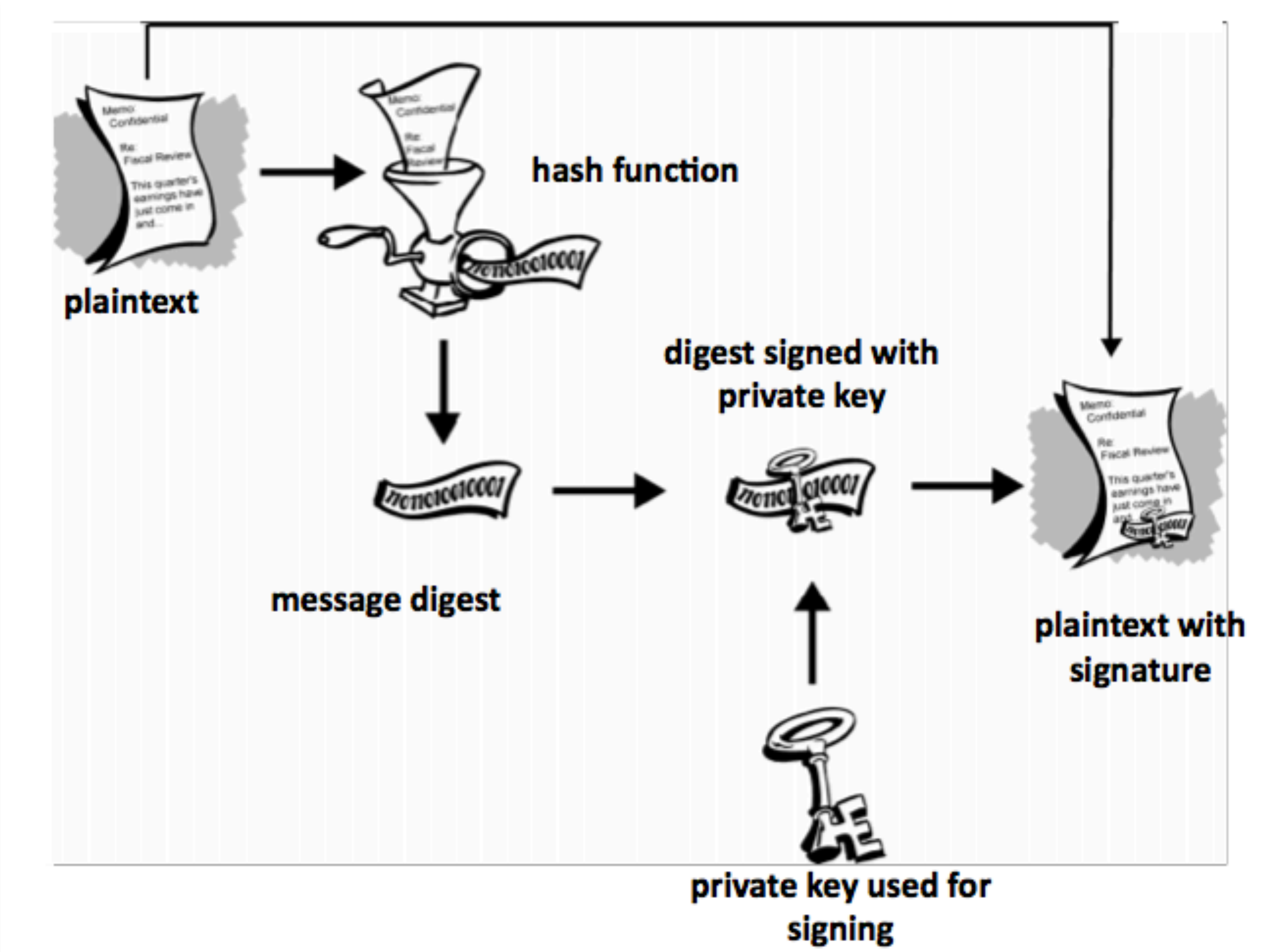
- *Munge* a document.
- Encrypt the *message digest* with your private key.
- Send the document plus the encrypted message digest.
- On the other end munge the document *and* decrypt the encrypted message digest with the person's public key.
- If they match, the document is authenticated.

Digital Signatures cont.

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



Another View



Use for authentication

- If you have my public key, I can prove to you that I own the corresponding private key (without sending it to you)
- My public key is therefore a form of identity
- Similarly, you can prove your identity to me
- Solves the man-in-the-middle problem, as long as we both know each other's public keys
  - If not, we can use a third party - a Certificate Authority - to confirm identity of key owner

And don't forget the human element