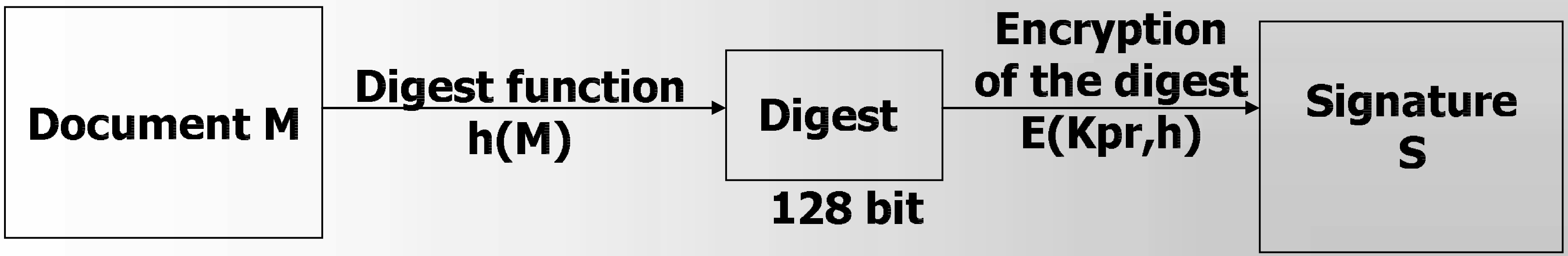
Digital Signatures

Introduction

- Why do we need digital signatures?
  - To confirm the identity of the sender of a message
  - To confirm a message has not been altered during transfer
  - Signing a message is much faster than crypting the whole message but is cryptographically as strong

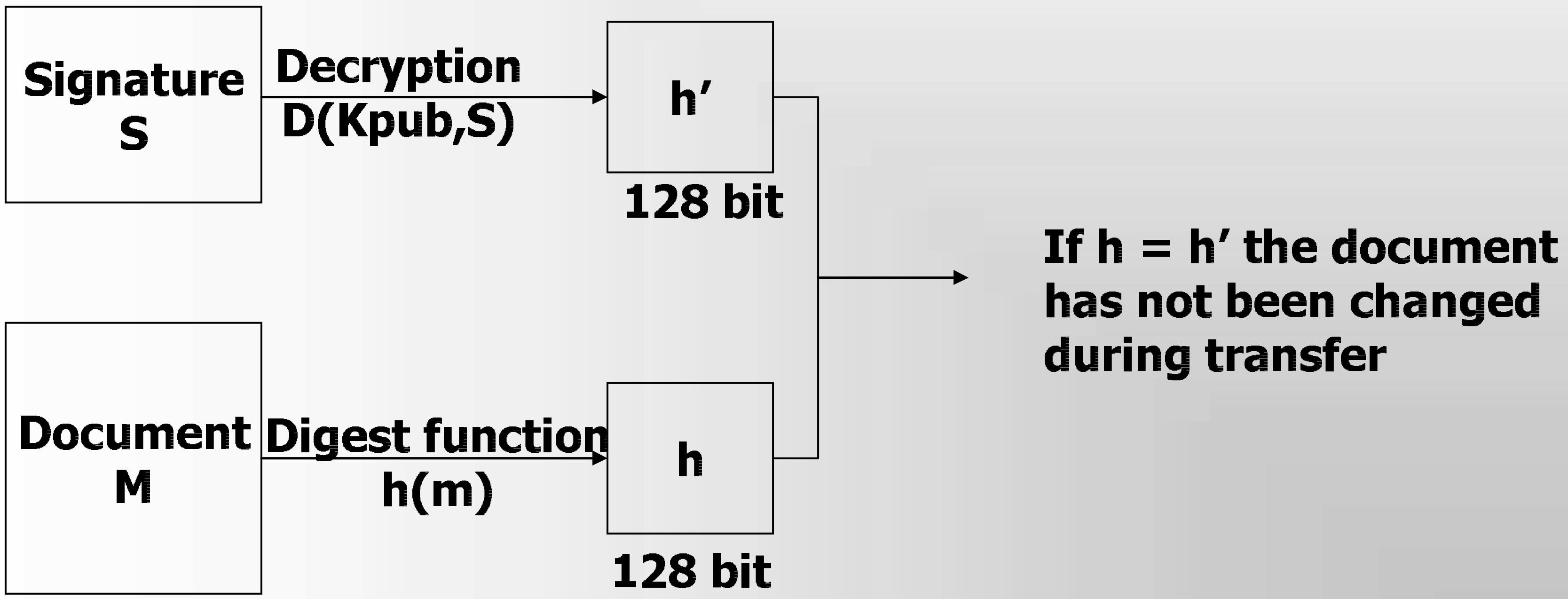
How does it work?

- A document can be signed using assymetrical cryptography and digest functions



How does it work?

- Verification of the signature



The digest function

- The digest function must have the following properties
  - The digest must be easy to calculate
  - The function must be a one way function
  - It should be almost impossible to get the same digest from a different message
  - For example MD5, SHA (Secure Hash Algorithm)

The keys

- Same problems as with the asymmetrical cryptography
  - How to make sure the person sending you the key is who he claims to be
  - How to make sure the keys are large enough?

Elliptic curve digital signatures

- Smaller key sizes
- More complex key generation
- But still faster to compute because of the smaller key sizes
- Can be applied easily to existing digital signaturing algorithms (for example DSA)

Elliptic curve digital signatures

- The RSA relies in the difficulty of the factoring of modulus n
- to aviod the factoring the modulus size should be something like 2^{1024} or more

Elliptic curve digital signatures

- The DSA relies on the difficulty of the discrete log problem in group of intergers
- Find a private key a for a public key α and private key a and $α^a$ mod p
- The size of the modulus is typically something like 2^{1024}
- The speed depends on the modulus size

Elliptic curve digital signatures

- The security of ECDSA relies on the discrete log problem in the group of points on an elliptic curve
- The traditional attacks against DSA doesn’t work on these -> smaller key sizes
- The key sizes of RSA and ECDSA

RSA modulus	1024	2048	4096
ECDSA field size	160	211	296

DSA & Elliptic curves

- With DSA we have a pair of integers (r,s)
  - where $x = a^k \bmod p$, $r = x \bmod q$,
    $s = k^{-1}(h(m)+ar) \bmod q$
  - a generates a subgroup of order q in Z
- With ECDSA we also have (r,s)
  - where $(x,y) = kP$, $r = x \bmod n$,
    $s = k^{-1}(h(m)+ar) \bmod n$
  - P generates a subgroup of order n in the curve E(F(q))

Elliptic Curve

- Elliptic curve over Finite Field
  - Let $p > 3$ be a prime and $GF(p^n)$ be a finite field
  - Let x^3+ax+b where a and b ∈ F be a cubic polynomial with no multiple roots
  - An elliptic curve E over F is the set of points (x,y) with x,y ∈ F, which satisfy the equation $y^2=x^3+ax+b$ with an element *O* "the point of infinity"

Elliptic Curve

- Addition law:
  - For $P(x_1,y_1) \in E$, $Q = (x_2,y_2) \in E$, then
    - $-P = (x_1,-y_1)$
    - $P+Q = (x_3,y_3)$ where $x_3 = \lambda^2-x_1-x_2$,

      $y_3 = \lambda(x_1-x_3)-y_1$

      where $\lambda = (y_2-y_1)/(x_2-x_1)$, if $P \neq Q$

      $\lambda = (3x_1^2+a)/2y_1$ if $P = Q$

Elliptic Curve - DSA

- Choose p, a prime, and n, an integer,

  f(x), an irreducible polynomial over GF(p) of degree n, generating finite field $GF(p^n)$ with the defining polynomial f(x), and assume that α is a root of f(x) in $GF(p^n)$
- Generate a curve E over $GF(p^n)$
- Choose a point P = (x,y) on E of order q, which is prime
- Converting function: c(x): $GF(p^n)$ -> Z_pn, which is given by $c(x) = \Sigma c_k p^i \in Z_pn$, i = {0,1,…,n-1} for

  $x = \Sigma c_k \alpha^i \in GF(p^n)$, i = {0,1,…,n-1} and $0 =< c_i < p$

Elliptic Curve - DSA

- Key:
  - Private key d, an integer, is selected randomly as $0 < d < q$
  - Public key Q, a point on curve E is computed by $Q = dP = (x_d, y_d)$.

Elliptic Curve - DSA

- Signing
  - Generate a key pair (k,R): choose a random number k: $0<k<q$ and compute point $R = kP = (x_k,y_k)$.
  - Compute r: using the converting function $c(x) = r$
  - Compute s: $h(m) = dr + ks \bmod q$,
    where h is the hashing algorithm

  The pair (r,s) is the signature of m

Elliptic Curve - DSA

- Verifying
  - Compute numbers:

    $t = s^{-1} \bmod q$

    $t_1 = tdr \bmod q$

    $t_2 = h(m)t \bmod q$
  - Compute the point on E by using private key P and public key Q:

    $t_1P - t_2Q = (x_e, y_e)$
  - Use the converting function for $c(x_e)$ and check if $r = c(x_e) \bmod q$. If so, then (r,s) is accepted

Elliptic curve digital signatures

- Summary of the elliptic curves
  - Elliptic curve algorithms can be faster if a smaller key size is used
  - Provides the same level of security with smaller key sizes
  - Choosing the parameters is much more difficult but the overhead is more than made up on the smaller key sizes