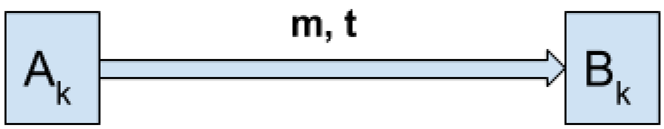
CS558 Scribe Notes

2/16/2017

Nicholas Mauro

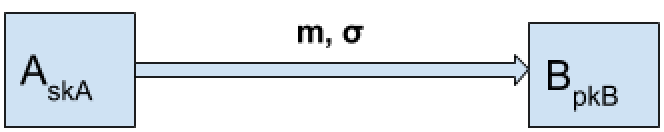
Public Key Crypto

So far, we've been thinking about crypto symmetrically…



With this method, we'd need keys for every possible link. In a class of 60 students, 60^2 keys would be required for all students to be able to communicate with one another.

Instead, we can use public keys to establish secret keys:



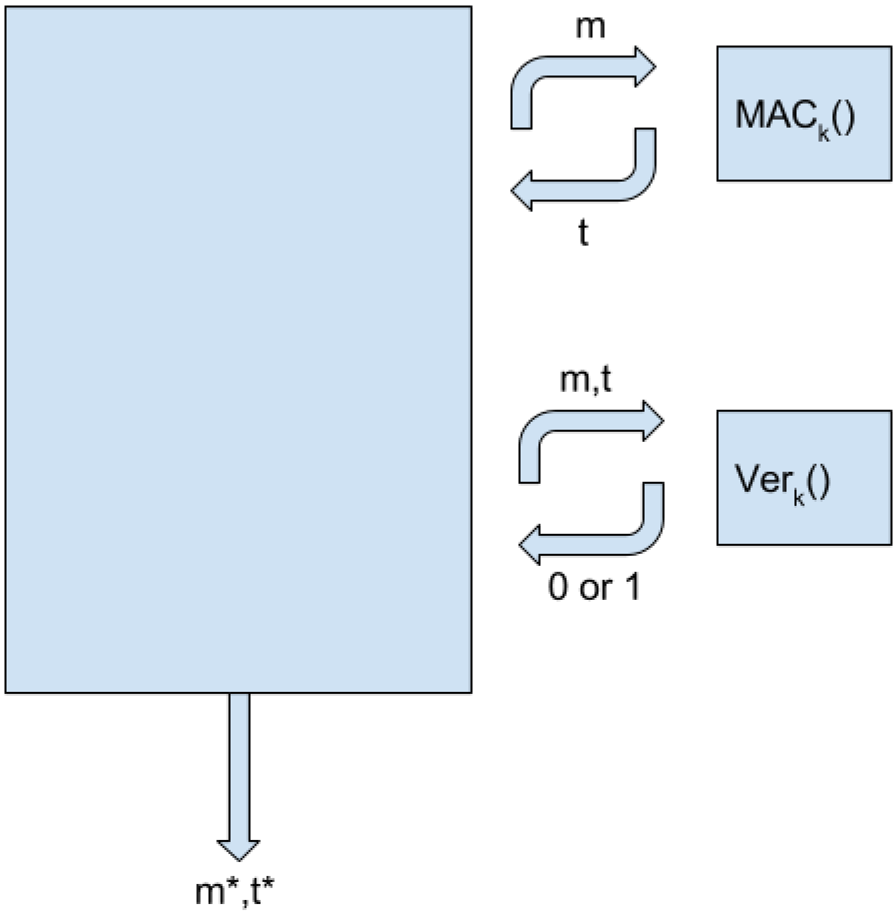
This is a "Digital Signature"

$\sigma = Sig_{skA}(m)$ Accept if: $Ver_{pkA}(m,\sigma) = 1$

- In this scenario, the secret key is held by the signer, and the public key is held by the verifier.
- Think of the public key as the "King's Stamp", which all members of the kingdom can recognize and verify comes from the king.
- Correctness: $Ver_{pk}(m, Sig_{sk}(m)) = 1$

Review of MAC (Message Authentication Code):

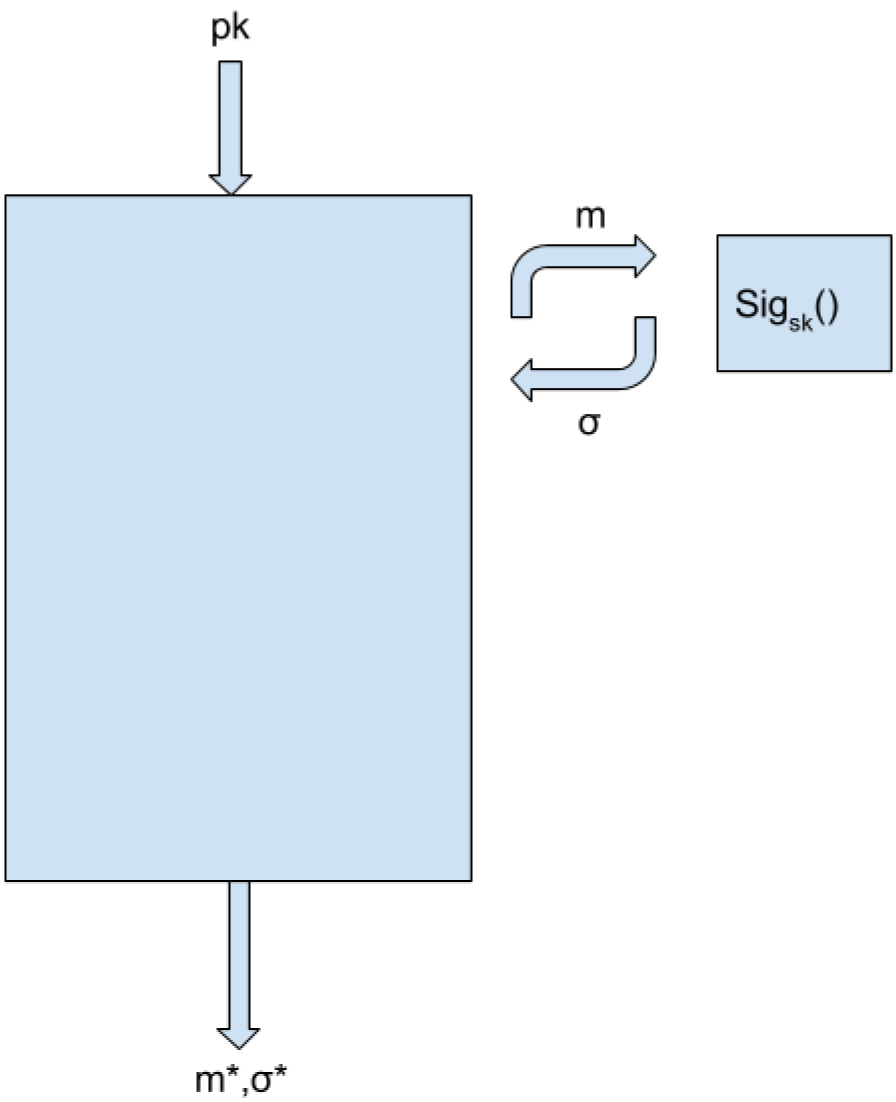
$t = MAC_k(m)$ Accept if: $Ver_k(m,t) = 1$



s.t. $Ver_k(m^*,t^*) = 1$

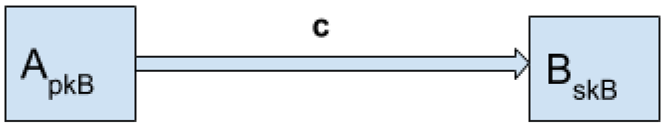
- MAC is secure if $\forall Adv(Pr[Adv\ Wins] = Pr[Ver_k(m^*,t^*) = 1]) = 1/2^l$
- Lowercase L is the length of t
- Cannot query m*

When using public and private keys, there is no verification oracle:



- $\forall$ Adv(Pr[Ver$_{pk}$(m*,σ*) = 1]) = 1/2^{l}
- Lowercase L is the length of σ
- Cannot query m*

Public Key Encryption:



$Enc_{pkB}(m) = c$ $Dec_{skB}(c) = m$

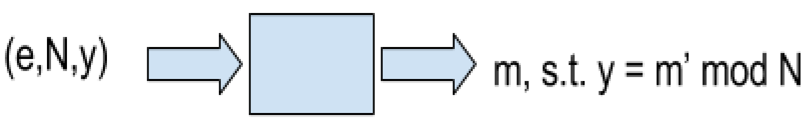
- This does NOT provide authenticity!

How do we get pk crypto?

One way is RSA function:

- Take 2 big primes, 2048 bits, p and q
- N = p*q = RSA modulus
- e = RSA encryption exponent
- Take m^e mod N
- For many years, e was always equal to 3

This is easy!



This is hard!

- m "sort of" equals the e^{th} root of y

If you know the decryption exponent "d", where $d = e^{-1} \bmod \Phi(N)$,

where $\Phi(N) = (p-1)*(q-1)$, you can take $y^d \bmod N = m$,

with m s.t. $y = m^e \bmod N$

RSA Encryption:

The following is “textbook” RSA encryption…

$pk_B = (e,N)$

$Enc_{pkB}(m) = m^e \bmod N = y$

$Dec_{skB}(y) = m$

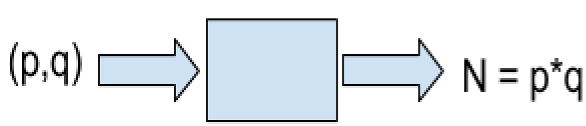
The “real life” secure example looks more like this…

$[PAD(m)]^e \bmod N$

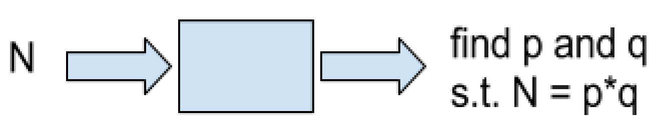
To generate RSA keys, choose random p and q, and find e

Output:

- pk = (N,e) where $N = p*q$
- sk = (N,d) or (p,q)
- “RSA function implies factoring is hard”



This is easy!



This is hard!

RSAGen() = p, q, e

- p and q are 2048 bit primes

$d = e^{-1} \bmod (p-1)*(q-1)$

- Keep secret: sk = (d, N, p, q)
- Make public: pk = (e, N)
- Decrypt: $y^d \bmod N$ $Dec_{sk}(y)$
- “Security of RSA rests on fresh moduluses”