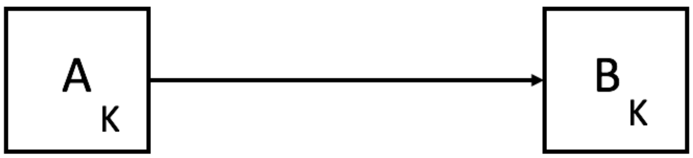
Scibe Notes: Arman Sanentz

Public Crypto

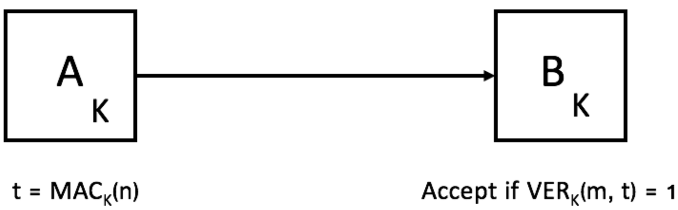
Message Authentication Core

- Here, the keys are symmetric:

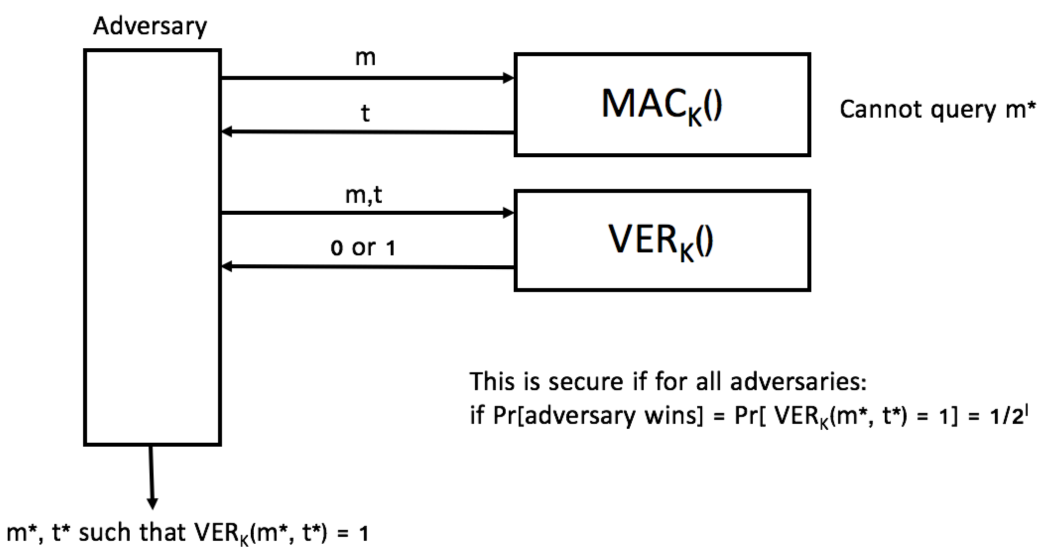


- Each person has a public key, and uses these keys to securely communicate.
- Public Key: one computation and everyone will verify

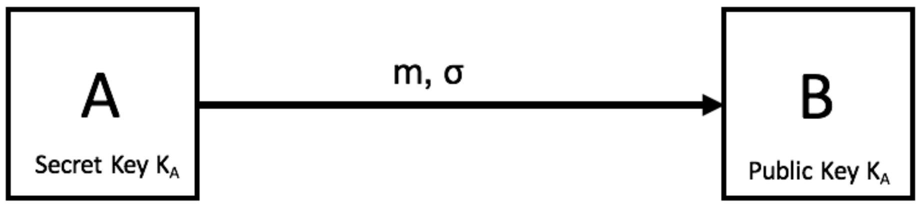
Recap of MAC



Recap of MAC security



Digital Signature



$\sigma = \text{SIGN}_{\text{SECRET KEY}_A}(m)$

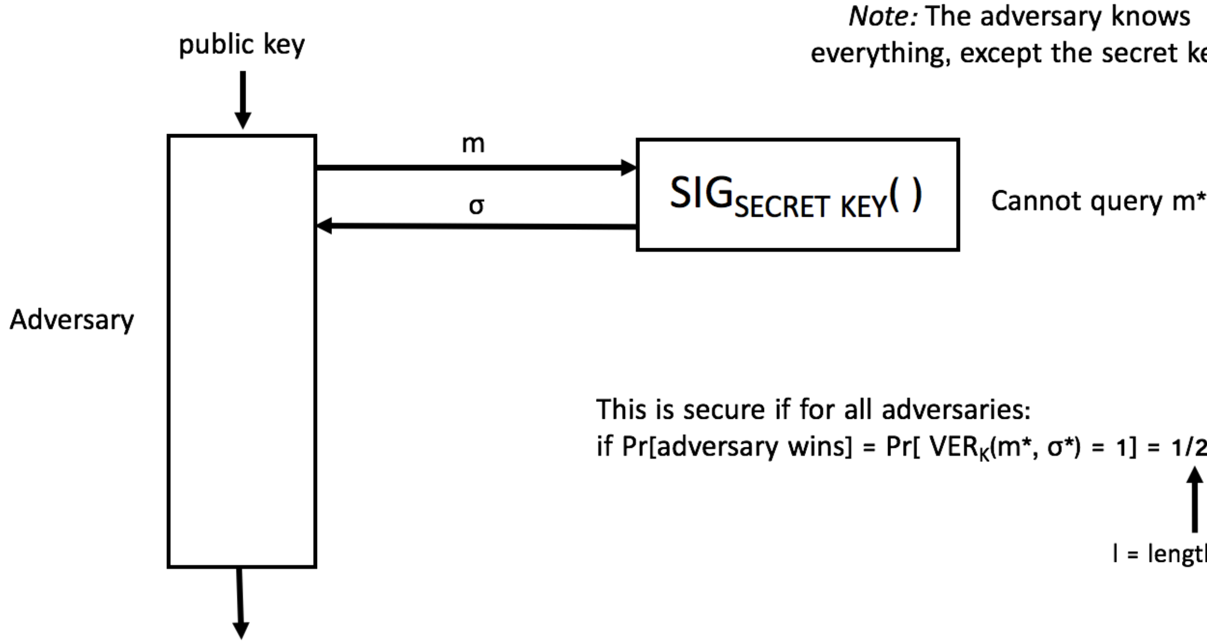
Accept if $\text{VER}_{\text{PUBLIC KEY}_A}(m, \sigma) = 1$

Note: The verification uses the **public key**

To show correctness: $\text{VER}_{\text{PUBLIC KEY}_A}(m, \text{SIGN}_{\text{SECRET KEY}_A}(m)) = 1$

Public Key Signatures

Note: The adversary knows everything, except the secret key

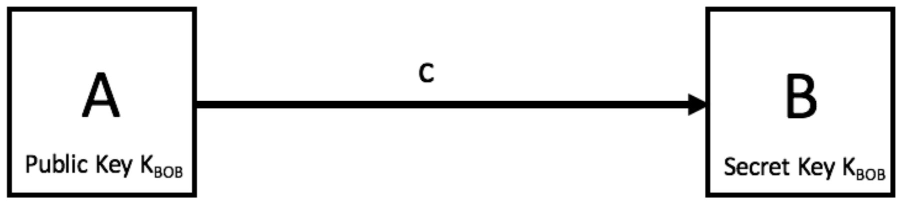


This is secure if for all adversaries:
if Pr[adversary wins] = Pr[$\text{VER}_K(m^*, \sigma^*) = 1$] = $1/2^l$

l = length of σ

m^*, σ^* such that $\text{VER}_{\text{PUBLIC KEY}}(m^*, \sigma^*) = 1$

<u>Public Key Encryption</u>



$ENC_{PUBLIC\ KEY_{BOB}}(m)$ $DEC_{SECRET\ KEY_{BOB}}(c) = m$

Note: The Secret Key/Public Keys are switched from the Digital Signature model

<u>How to Set Public Key Crypto?</u>

- One way is using the RSA Function:

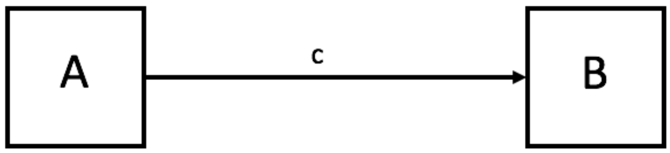
Prime numbers (big, 2048 bits) p & q	← These are secret to everybody
n = p * q	← RSA modules
e = encryption exponent	← usually standardized (ex: e = 3)

EASY:
(e, N, M) → [(to write an algorithm here)] → m^e mod N

NOT EASY:
(e, N, y) → [(to write an algorithm here)] → m such that y = m^e mod N (essentially "$\sqrt[e]{y} = m$")

ALSO EASY:
If you know the decryption exponent d:
$d = e^{-1} \bmod \phi(N) = (p-1)(q-1)$
where $\phi(N) = (p-1)(q-1)$

Then you can solve: y^d mod N = m where m such that y = m^e mod N



$Y = ENC_{PUBLIC\ KEY_{BOB}}(m)$ $DEC_{SECRET\ KEY_{BOB}}(y) = m$

PUBLIC KEY = (e, N)

$Y = ENC_{PUBLIC\ KEY_{BOB}}(m)$ $= m^e \bmod N = y$ ← textbook RSA encryption
$= [PAD(m)]^e \bmod N$ ← actually what is needed to serve RSA encryption (see lab)

RSA Gen() = p, q, e ← p and q are random 2048 bit integers
$N = p * q$
$d = e^{-1} \bmod (p-1)(q-1)$
SECRET KEY = (d, N, p, q) ← keep secret
PUBLIC KEY = (e, N) ← make public

To generate RSA keys, choose random p, q, fixed e

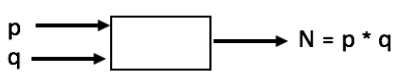
Output:
PUBLIC KEY = (N, e) where N = P * e
SECRET KEY = (N, d) or (p, q)

Here, we are "implying that factoring is hard."

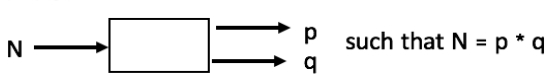
Rabin Encryption Scheme

We are using prime numbers (big, 2048 bits) p & q

EASY:



NOT EASY:



Alice:
$Dec_{SECRET\ KEY}(y) = y^d \bmod N$ = message
PUBLIC KEY = (e, N)