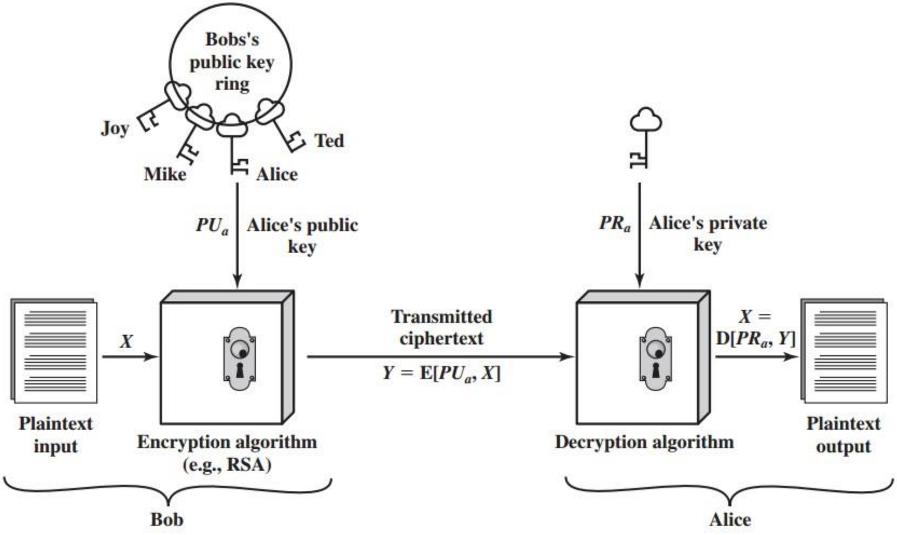
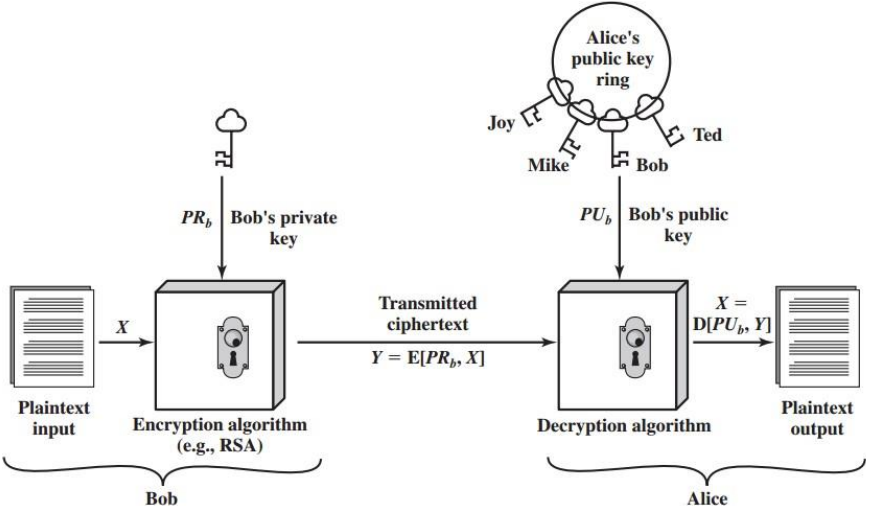
BlockChain Technologies

Digital Signature

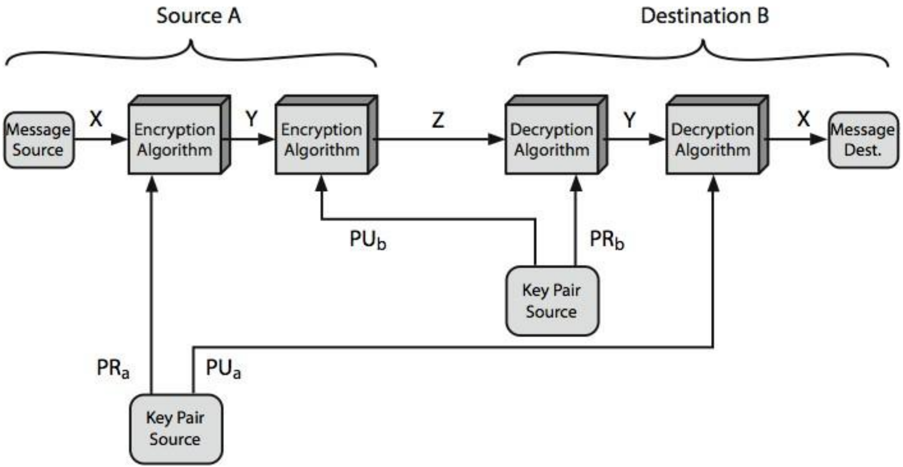
PUBLIC KEY SCHEME FOR CONFIDENTIALITY



PUBLIC KEY SCHEME FOR AUTHENTICATION



CONFIDENTIALITY AND AUTHENTICATION



RSA

- One of the first proposals on implementing the concept of public-key cryptography was that of Rivest, Shamir, Adleman – 1977: RSA
- The RSA scheme works like a block cipher in which the plaintext and the ciphertext are integers between 0 and $n-1$ for some fixed n
  - Typical size for n is 1024 bits (or 309 decimal digits)
  - To be secure with today's technology size should between 1024 and 2048 bits
- Idea of RSA: it is a difficult math problem to factorize (large) integers
  - **Choose p and q odd primes, and compute $n = pq$**
  - **Choose integers d, e such that $M^{ed} = M$ mod n, for all $M < n$**
  - **Plaintext**: can be considered a number M with $M < n$
  - **Encryption**: $C = M^e \ mod \ n$
  - **Decryption**: $C^d \ mod \ n = M^{ed} \ mod \ n = M$
  - **Public key**: $PU = \{e, n\}$ and **Private key**: $PR = \{d, n\}$

ATTACKING RSA

- **Brute force attacks**: try all possible private keys
  - As in the other cases defend using large keys: nowadays integers between 1024 and 2048 bits

- **Mathematical attacks**
  - Factor n into its two primes p, q: this is a hard problem for large n
    - Challenges by RSA Labs to factorize large integers
    - Last solved RSA challenge: 829 bits (Feb 2020)
  - Determine $\phi(n)$ directly without first determining p, q: this math problem is equivalent to factoring
  - Determine d directly, without first determining $\phi(n)$: this is believed to be at least as difficult as factoring

DISCRETE LOGARITHM PROBLEM

- Let p be a prime number. We represent the set of all powers of number a modulo p with $< a>_p$:

$$<2>_7 = \{1,2,4\} \qquad <3>_7 = \{1,3,2,6,4,5\}$$

- We call g a generator of $\mathbb{Z}^*_p$ if $< g>_p = \mathbb{Z}^*_p$
- $\mathbb{Z}^*_p$ definitely has a generator which is not necessarily unique.
  - Having the factorization of $p-1$, it is easy to find a generator for $\mathbb{Z}^*_p$.
- Discrete Logarithm problem (DLP): Given prime number p, an arbitrary generator g of $\mathbb{Z}^*_p$ and $g^\alpha \bmod p$ (where α is a random integer in $\mathbb{Z}_{p-1}$), find α.
- For large values of p, solving DLP is computationally infeasible.

DIFFIE-HELLMAN PROBLEM

- Diffie-Hellman Problem (DHP): Given prime number p, an arbitrary generator g of Z_p^* , g^{α} mod p and g^{β} mod p (where α and β are random integers in $\mathbb{Z}_{p-1}$), find $g^{\alpha\beta}$ **mod** p.
  - Solving DHP is easier than solving DLP.
    - It is obvious that if we solve DLP efficiently, we have solved DHP efficiently!
  - The opposite is not proved yet, i.e. solving DHP efficiently will not result in an efficient solution for DLP.
  - There is no known method for solving DHP without solving DLP first.

DIFFIE-HELLMAN KEY AGREEMENT

Alice

Bob

$g^{\alpha} \; mod \; p$

Alice selects random α

$g^{\beta} \; mod \; p$

Bob selects random β

Alice computes $\left(g^{\beta}\right)^{\alpha} = g^{\alpha\beta} \; mod \; p$ as the shared key **(session key)**

Bob computes $\left(g^{\alpha}\right)^{\beta} = g^{\alpha\beta} \; mod \; p$ as the shared key **(session key)**

DIGITAL SIGNATURE

WHAT WE WANT FROM SIGNATURES

Only you can sign, but anyone can verify

Signature is tied to a particular document

can’t be cut-and-pasted to another doc

API FOR DIGITAL SIGNATURES

(sk, pk) := generateKeys(keysize)

sk: secret signing key

pk: public verification key

sig := sign(sk, message)

isValid := verify(pk, message, sig)

can be randomized algorithms

REQUIREMENTS FOR SIGNATURES

"valid signatures verify"

verify(pk, message, sign(sk, message)) == true

"can't forge signatures"

adversary who:

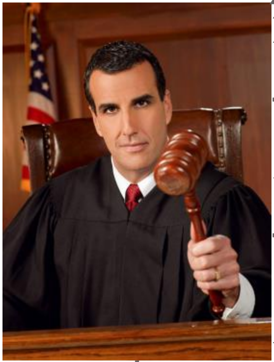
knows pk

gets to see signatures on messages of his choice

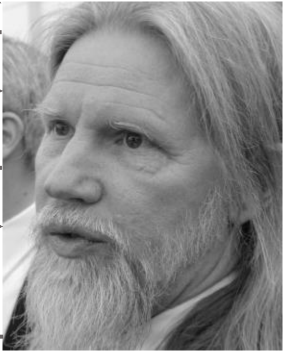
can't produce a verifiable signature on another message

(sk, pk)

challenger



attacker



m_0

sign(sk, m_0)

m_1

sign(sk, m_1)

. . .

M, sig

M not in { m_0, m_1, … }

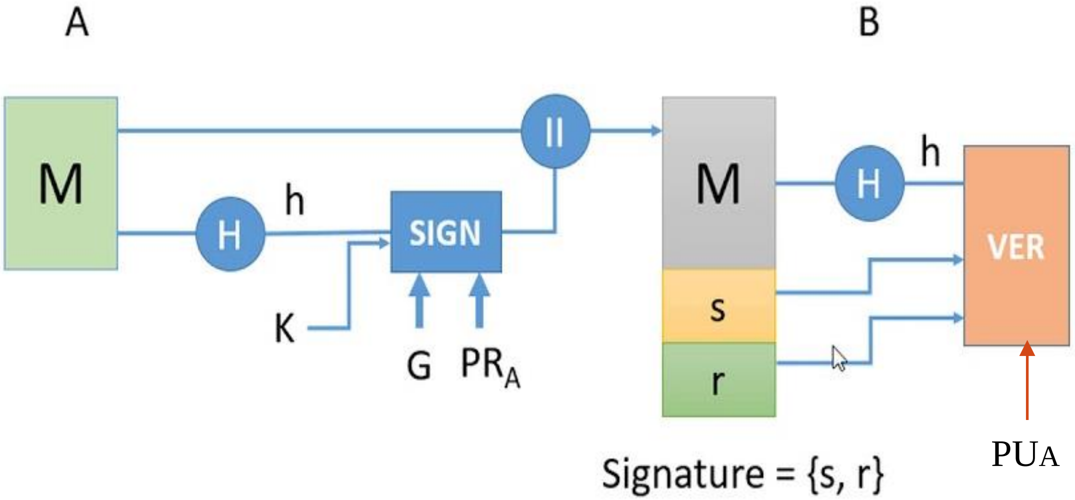
verify(pk, M, sig)

if true, attacker wins

DIGITAL SIGNATURE ALGORITHM

- **What Is DSA (Digital Signature Algorithm)?**
- DSA is a United States Federal Government standard for digital signatures.
- It was proposed by the National Institute of Standards and Technology (NIST) in August 1991 for use in their Digital Signature Standard (DSS)
- It was specified in FIPS 186 in 1993.
  - Latest update as a standard (FIPS 186-4) in 2013
- DSA is based on ElGamal public-key cryptosystem

DSA SCHEMA



KEY GENERATION IN DSA

- The first part of the DSA is the public key and private key generation:
- Choose a prime number q, which is called the **prime divisor**.
- Choose another prime number p, such that $(p-1) \bmod q = 0$.
  - p is called the **prime modulus** and its length is more than 512 bits.
- Choose an integer g, such that $1 < g < p$, $g^q \bmod p = 1$
  - This may be done by setting $g = h^{(p-1)/q}\, mod\ p$.
  - q is also called g's multiplicative order modulo p.
- Choose an integer, such that $0 < x < q$.
- Compute y as $g^x \bmod p$
- Package the public key as $\{p, q, g, y\}$
- Package the private key as $\{p, q, g, x\}$

SIGNING IN DSA

- Let H be the hashing function and m the message.
- Generate a random per-message value k where $1 < k < q$
- Calculate $r = (g^k \bmod p) \bmod q$
  - In the unlikely case that $r = 0$, start again with a different random k
- Calculate $s = k^{-1}(H(m) + xr) \bmod q$
  - In the unlikely case that $s = 0$, start again with a different random k
- Package the digital signature as (r, s).

VERIFYING THE SIGNATURE

- Reject the signature if $0 < r, s < q$ is not satisfied.
- Calculate $w = s^{-1} \bmod q$
  - $s = k^{-1}(H(m) + xr) \bmod q$ $\longrightarrow$ $w = k(H(m) + xr)^{-1} \bmod q$
- Calculate $u_1 = H(m).w \bmod q$
  - $u_1 = H(m)k(H(m) + xr)^{-1} \bmod q$
- Calculate $u_2 = r.w \bmod q$
  - $u_2 = kr(H(m) + xr)^{-1} \bmod q$
- Calculate $v = (g^{u_1} y^{u_2} \bmod p) \bmod q$
  - $v = (g^{u_1 + xu_2} \bmod p) \bmod q = (g^{k(H(m)+rx)(H(m)+rx)^{-1}}) \bmod p) \bmod q$
    $= (g^k \bmod p) \bmod q$
- The signature is invalid unless $v = r$.