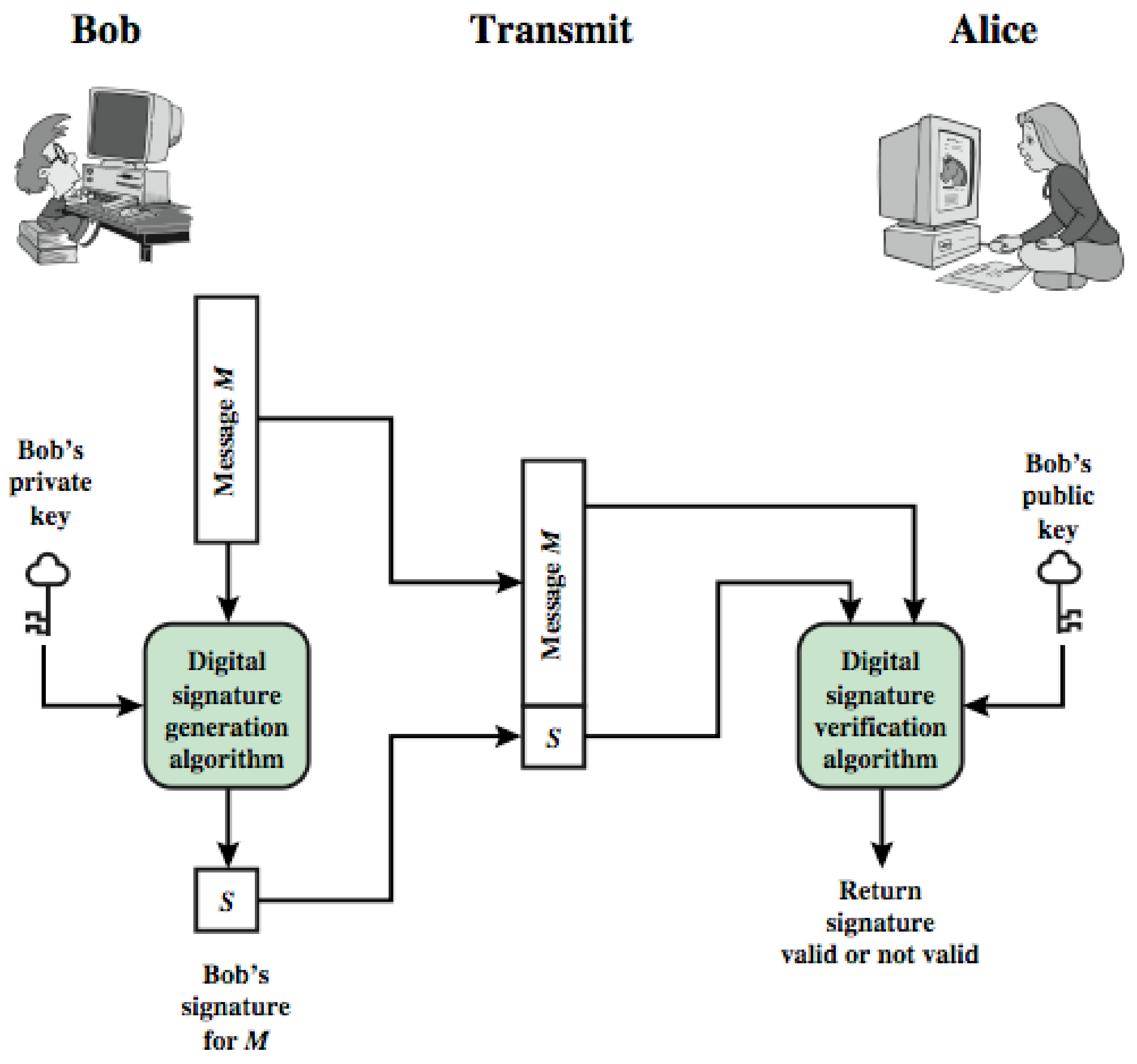
Plan of Talk : RSA Signatures

- **RSA Signature**
- **Issues with Textbook version of RSA Signature**
- **Next Week: Hash Functions and Practical Signature Schemes**

Digital Signature Model



Digital Signature Requirements

- The signature must be a bit pattern that depends on the message being signed.
- The signature must use some information unique to the sender to prevent both forgery and denial
- It must be relatively easy to produce the digital signature
- It must be relatively easy to recognize and verify the digital signature
- It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message
- It must be practical to retain a copy of the digital signature in storage

Digital Signatures

- The purpose of this discussion is to understand how RSA signature works and how it is different from RSA encryption we studied earlier.
- RSA signature is a public key signature scheme
- Signature is a means for a trusted third party (Network Security Manager) to bind the identity of a user to a public key.

Digital Signatures cont.

- RSA signature is complement of RSA public encryption.
- Owner for the private key signs messages; Anyone with the public key can verify the signatures.
- In RSA encryption, anyone with the public key can encrypt messages and the owner of the private key can decrypt the messages.

RSA : Alice's parameters

- $N = P*Q$; P, Q Large Primes
- Choose Public key e and private key d such that $e*d \equiv 1 \bmod \phi(N)$
- Public address – [N,e]
- Private address –[d]
- Signature Generation:
- Message $0 < M < N$;
- Compute: $s = M^d \bmod N$;
- Signature—[M,s]
- Verification – if $s^e \bmod N =?= M$

Some Consequences: Existential Forgery attack

Multiplicative property of RSA signature

$(M1 * M2)^d = M1^d * M2^d$

i.e. if s1 = Signature of M1;

s2 = Signature of M2

(s1 * s2) is the signature of (M1*M2).

This leads to a possibility of forgery of signature!

Further

- What if Message is very long?
- Also, a problem called blinding.

Blinding

- Alice's public key – [N,e]; Private key [d]
- You want to get Alice sign a message M which Alice normally may not do.
- Choose a random x – in the range [0..N-1]
- Form a blinded message -- $M_b = x^e M \mod N$
- Alice may sign $s_b = M_b^d \mod N$
- Now you can compute signature for M as
- $s = s_b/x \mod N$
- Note
- $s^e = s_b^e / x^e = (M_b)^{d\,e} / x^e =$
  $(M_b) / x^e = x^e M / x^e = M$
- Hence s is the signature of M.

RSA Signature Modified

- Public address – [N,e] ; Private address –[d]
- Signature Generation:
- Message $0 < M < N$;
- Find $M1 = R(M)$; where R is a redundancy function,
- $1 < R(M) < N$.
- Compute: $s = (M1)^d \bmod N$;
- Signature—[M,s]
- Verification –
- Compute $s^e \bmod N = M1$
- Verify R(M) ==?==M1

RSA Signature in Practice

A practical signature scheme should take care of the two problems discussed before.

- Messages are generally long
- RSA signature scheme needs a redundancy function to avoid existential forgery attacks.
- Also repeated messages carry same signature.
- In practice, generally a suitable cryptographic hash function is applied to the message (which could be arbitrarily large); and sign the hash.

Practical RSA Signature Scheme

- Public address – [N,e] ; Private address –[d]
- Signature Generation:
- Message M of arbitrary length,
- Find h(M); where h is an hash function,
- Format the message before signing
- M1 as [h(M), identity information, random number]
- such that $0 < M1 < n$
- Compute: $s = (M1)^d \bmod N$;
- Signature—[M,s]
- Verification –
- Extract M1 by computing $s^e \bmod N = M1$
- Any formatting violations– reject the signature
- Further Verify h(M) ==?==M1

Security of RSA

- Brute force Attack: (infeasible given size of numbers)
- Attack by making use of loop holes in Key distribution.
- Mathematical attacks (Factoring and RSA problem)
    - Elementary attacks
    - Advanced Factorization methods
- Brute force Attack: (infeasible given size of numbers)
- Network attacks
    - Timing attacks (on running of decryption)

Mathematical attacks

- The RSA function is one way
- The problem
- Given n,e, $c=M^e \pmod{n}$,
  - Can we determine M?
  - Do we have an algorithm to find the e^{th} root of c mod n?
  - Can we find d such that $de = 1 \bmod \Phi(n)$?
- Can we factor n?

Factorization Problem

- In general the factorization is hard.
- Brute force algorithm is exponential in b, where b is number of bits in the representation of the number n to be factored.
- Complexity of the best known algorithm for factorization:

$$\exp((c+O(1)b^{1/3}\,\mathrm{Log}^{2/3}(b)),$$

for some integer $c < 2$

- It is not worth thinking of factoring.
- May be quantum computers come to our rescue; but may not in our life time!

Elementary attacks

- Can we use common modulus for more than one user?
- Facts:
- Knowing e and d such that

$$ed = 1 \bmod \Phi(n)$$

Is equivalent to factoring.

- Knowing n, $\Phi(n)$ is also equivalent to factoring

Common modulus

- Every user chooses same modulus n=pq set up by a trusted central authority. But each user chooses their own private and public key pairs
- User i ------(e_i,d_i)
- So using the facts in previous slide, any user can factor common modulus n and
- can find the private information of other user by using only the public information.
- Hence it is extremely important that every entity chooses its own RSA modulus n.

Broadcast Problem

A group of entities may all have a same encryption exponent but should have different modulus. Further, to improve the public encryption, let the public key be small, say e=3.

- A wishes to send a common message m to three entities with modulus n1,n2 and n3.
- The cipher text for three entities are given by
- $c1 = m^3 \pmod{n1}$
- $c2 = m^3 \pmod{n2}$
- $c3 = m^3 \pmod{n3}$.

- Then, to recover the message m solve,
- $x = c1 \pmod{n1}$,
- $x = c2 \pmod{n2}$,
- $x = c3 \pmod{n3}$,

- You can use CRT and Then obtain an unique
- $x=m^3$ modulo n1 n2 n3
- m can then be obtained by taking the cube root of x.

RSA-PSS

- RSA Probabilistic Signature Scheme, included in the 2009 version of FIPS 186
- Latest of the RSA schemes and the one that RSA Laboratories recommends as the most secure of the RSA schemes
- For all schemes developed prior to PSS is has not been possible to develop a mathematical proof that the signature scheme is as secure as the underlying RSA encryption/decryption primitive
- The PSS approach first proposed by Bellare and Rogaway
- This approach, unlike the other RSA-based schemes, introduces a randomization process that enables the security of the method to be shown to be closely related to the security of the RSA algorithm itself
- More details on the algorithm are available in the textbook.

Timing Attacks

- Attack introduced by Paul Kocher in 1996.
- If implemented correctly, an attacker can recover private key by keeping track of how long a receiver computer takes to decipher messages.
- This is a serious attack as the previous models do not address this attack.
- The worry is that it is applicable many other crypto systems including symmetric key ciphers.

Some typical methods for Timing attacks

- Constant exponentiation time: Ensure that all exponentiations take the same amount of time before returning a result. This is a simple fix but does degrade performance.
- Random delay: Better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack.
- Blinding: Multiply the ciphertext by a random number before performing exponentiation. This process prevents the attacker from knowing what ciphertext bits are being processed inside the computer and therefore prevents the bit-by-bit analysis essential to the timing attack.

Fault-Based Attack

- An attack on a processor that is generating RSA digital signatures
  - Induces faults in the signature computation by reducing the power to the processor
  - The faults cause the software to produce invalid signatures which can then be analyzed by the attacker to recover the private key
- The attack algorithm involves inducing single-bit errors and observing the results
- While worthy of consideration, this attack does not appear to be a serious threat to RSA
  - It requires that the attacker have physical access to the target machine and is able to directly control the input power to the processor

Summary

- Security of Public Key Algorithms
- CCA Attack on Textbook RSA
- Textbook RSA Signature Algorithm
- Blinding
- RSA Attacks