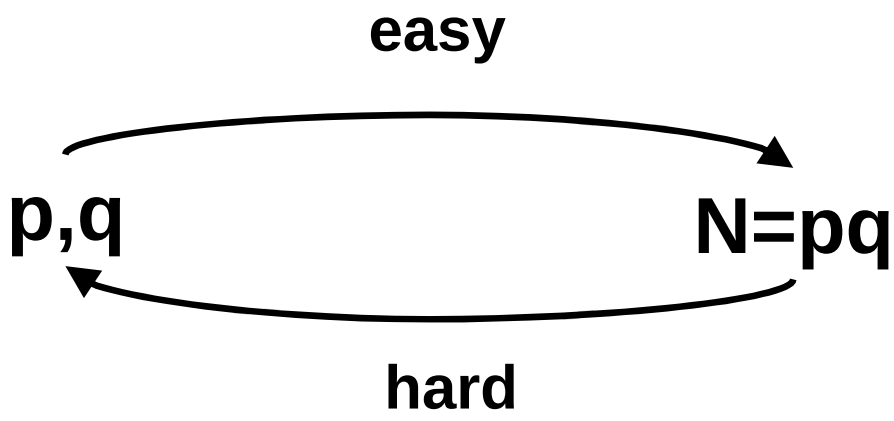
Introduction to Cryptography

Lecture 10

Benny Pinkas

Integer Multiplication & Factoring as a One Way Function.



Can a public key system be based on this observation ?????

Excerpts from RSA paper (CACM, 1978)

The era of "electronic mail" may soon be upon us; we must ensure that two important properties of the current "paper mail" system are preserved: (a) messages are *private*, and (b) messages can be *signed*. We demonstrate in this paper how to build these capabilities into an electronic mail system.

At the heart of our proposal is a new encryption method. This method provides an implementation of a "public-key cryptosystem," an elegant concept invented by Diffie and Hellman. Their article motivated our research, since they presented the concept but not any practical implementation of such system.

The Multiplicative Group Z_{pq}^*

- *p* and *q* denote two large primes (e.g. 512 bits long).
- Denote their product as $N = pq$.
- The multiplicative group $Z_N^* = Z_{pq}^*$ contains all integers in the range *[1,pq-1]* that are relatively prime to both *p* and *q*.

- The size of the group is
  - $\phi(n) = \phi(pq) = (p-1)(q-1) = N - (p+q) + 1$

- For every $x \in Z_N^*$, $x^{\phi(N)} = x^{(p-1)(q-1)} = 1 \bmod N$.

Exponentiation in Z_N^*

- Motivation: use exponentiation for encryption.
- Let e be an integer, $1 < e < \phi(N) = (p-1)(q-1)$.
  - Question: When is exponentiation to the e^{th} power, ($x \rightarrow x^e$), a one-to-one operation in Z_N^* ?
- Claim: If e is relatively prime to $(p-1)(q-1)$ (namely $gcd(e, (p-1)(q-1))=1$) then $x \rightarrow x^e$ is a one-to-one operation in Z_N^*.
- Constructive proof:
  - Since gcd$(e, (p-1)(q-1))=1$, e has a multiplicative inverse modulo $(p-1)(q-1)$.
  - Denote it by d, then $ed=1+c(p-1)(q-1)=1+c\phi(N)$.
  - Let $y=x^e$, then $y^d = (x^e)^d = x^{1+c\phi(N)} = x$.
  - I.e., $y \rightarrow y^d$ is the inverse of $x \rightarrow x^e$.

The RSA Public Key Cryptosystem

- Public key*:*
  - $N=pq$ the product of two primes (we assume that factoring N is hard)
  - e such that $\gcd(e,\phi(N))=1$ *(are these hard to find?)*
- Private key:
  - d such that $de \equiv 1 \bmod \phi(N)$
- Encryption of $M \in Z_N^*$
  - $C=E(M)=M^e \bmod N$
- Decryption of $C \in Z_N^*$
  - $M=D(C)=C^d \bmod N$ *(why does it work?)*

Constructing an instance of the RSA PKC

- Alice
  - picks at random two large primes, p and q.
  - picks (uniformly at random) a (large) d that is relatively prime to (p-1)(q-1) (namely, gcd$(d,\phi(N))=1$).
  - Alice computes e such that $de \equiv 1 \bmod \phi(N)$

- Let $N=pq$ be the product of p and q.
- Alice publishes the public key (N,e).
- Alice keeps the private key d, as well as the primes p, q and the number $\phi(N)$, in a safe place.

Efficiency

- The public exponent *e* may be small.
  - It is common to choose its value to be either 3 or $2^{16}+1$. The private key *d* must be long.
  - Each encryption involves only a few modular multiplications. Decryption requires a full exponentiation.
- Usage of a small *e* $\Rightarrow$ Encryption is more efficient than a full blown exponentiation.
- Decryption requires a full exponentiation ($M=C^d \bmod N$)
- Can this be improved?

The Chinese Remainder Theorem (CRT)

- Thm:
  - Let $N=pq$ with gcd$(p,q)=1$.
  - Then for every pair $(y,z) \in Z_p \times Z_q$ there exists a *unique* $x \in Z_n$, s.t.
    - $x=y \bmod p$
    - $x=z \bmod q$
- Proof:
  - The extended Euclidian algorithm finds a,b *s.t.* $ap+bq=1$.
  - Define $c=bq$. Therefore $c=1 \bmod p$. $c=0 \bmod q$.
  - Define $d=ap$. Therefore $d=0 \bmod p$. $d=1 \bmod q$.
  - *Let* $x=cy+dz \bmod N$.
    - $cy+dz = 1y + 0 = y \bmod p$.
    - $cy+dz = 0 + 1z = z \bmod q$.
  - (How efficient is this?)
  - (The inverse operation, finding (y,z) from x, is easy.)

More efficient RSA decryption

- CRT:
  - Given *p,q* compute *a,b* s.t. *ap+bq=1.*
  - *c=bq; d=ap*

} Once for all messages

- Decryption, given *C*:
  - Compute $y'=C^d \bmod p$. (instead of *d* can use $d'=d \bmod p-1$)
  - Compute $z'=C^d \bmod q$. (instead of *d* can use $d''=d \bmod q-1$)
  - Compute $M=cy'+dz' \bmod N$.

- Overhead:
  - Two exponentiations modulo *p,q,* instead of one exponentiation modulo *N*.
  - Overhead of exponentiation is *cubic* in length of modulus.
  - I.e., save a factor of $2^3/2$.

Security reductions

- Security by reduction
  - Define what it means for the system to be “secure” (chosen plaintext/ciphertext attacks, etc.)
  - State a “hardness assumption” (e.g., that it is hard to extract discrete logarithms in a certain group).
  - Show that if the hardness assumption holds then the cryptosystem is secure.

- Benefits:
  - To examine the security of the system it is sufficient to check whether the assumption holds
  - Similarly, for setting parameters (e.g. group size).

RSA Security

- (For ElGamal encryption, we showed that if the DDH assumption holds then El Gamal encryption has semantic security.)
- We know that if factoring N is easy then RSA is insecure
  - can factor $N \Rightarrow$ find $p,q \Rightarrow$ find $(p-1)(q-1) \Rightarrow$ find d from $e \Rightarrow$ decrypt RSA
  - Is the converse true? (we would have liked to show that decrypting RSA $\Rightarrow$ factoring N)
- Factoring assumption:
  - For a randomly chosen p,q of good length, it is infeasible to factor $N=pq$.
  - This assumption might be too weak (might not ensure secure RSA encryption)
  - Maybe it is possible to break RSA without factoring N ?
  - We don't know how to reduce RSA security to the hardness of factoring.

  - Fact: finding d is equivalent to factoring.
  - I.e., if it is possible to find d given (N,e) , then it is easy to factor N.
  - can find d from $e \Rightarrow$ can factor N
  - But perhaps it is possible to break RSA without finding d?

The RSA assumption: Trap-Door One-Way Function (OWF)

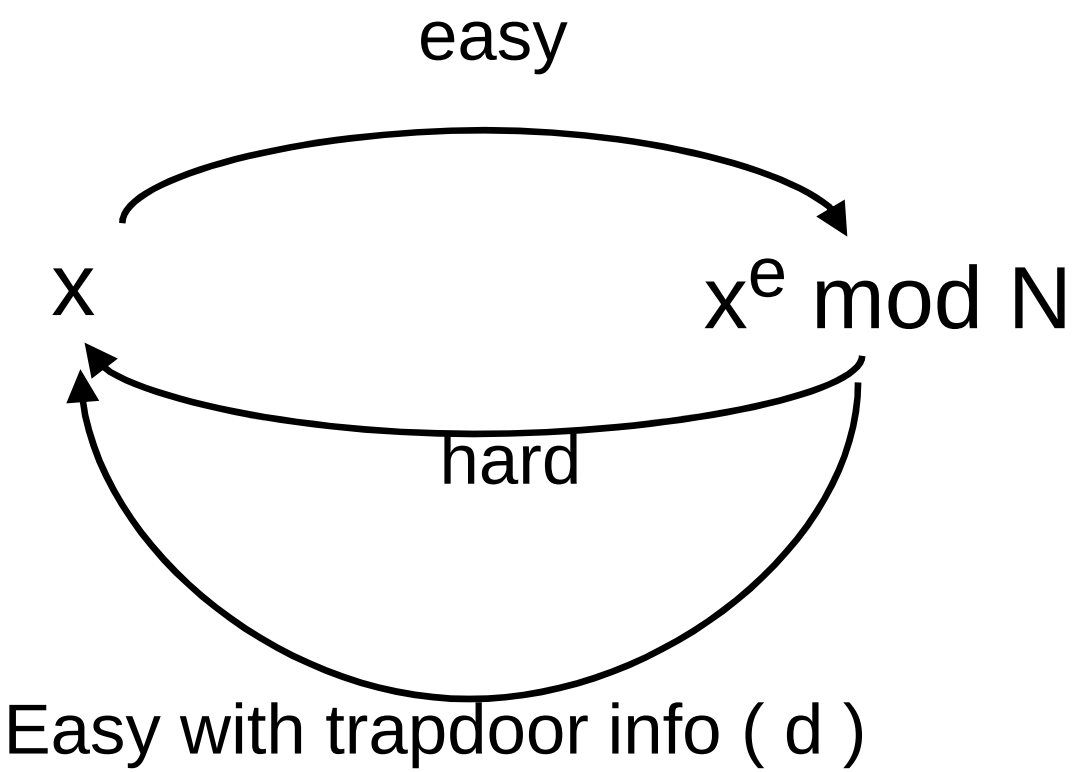
- (what is the minimal assumption required to show that RSA encryption is secure?)
- (Informal) definition: $f : D \rightarrow R$ is a *trapdoor one way function* if there is a trap-door s such that:
  - Without knowledge of s, the function f is a one way. I.e., for a randomly chosen x, it is hard to invert $f(x)$.
  - Given s, inverting f is easy

The RSA assumption: Trap-Door One-Way Function (OWF)

- Example: $f_{g,p}(x) = g^x \bmod p$ is *not* a trapdoor one way function. (Therefore El Gamal encryption is not based on assuming the existence of a trapdoor one way function.)

- The RSA assumption: the RSA function is a trapdoor OWF
  - The setting: Generate random RSA keys *(N,e,d).* Choose random $y \in Z^*_N$. Provide the adversary with *N,e,y.*
  - The assumption that is the there is no efficient algorithm which can output x such that $x^e = y \bmod N$.
  - The trap-door one-way function is $f_{N,e}(x) = x^e \bmod N$. (with *N,e,x,* chosen at random)
  - The trapdoor is d s.t. $ed = 1 \bmod \phi(N)$

RSA as a One Way Trapdoor Permutation



RSA assumption: cautions

- The RSA assumption is quite well established:
  - RSA is a Trapdoor One-Way Permutation
  - Hard to invert on random input – without secret key
- But is it a secure cryptosystem?
  - Given the assumption it is hard to reconstruct the input (if the input was chosen randomly), but is it hard to learn *anything* about the input?
- Theorem [G]: RSA hides the log(log(*n*)) least *and* most significant bits of a uniformly-distributed random input
  - But some (other) information about pre-image may leal

Security of RSA

- Deterministic encryption. In textbook RSA:
  - M is always encrypted as M^e
  - The ciphertext is as long as the domain of M
- Corollary: textbook RSA does not have semantic security.
  - If we suspect that a ciphertext is an encryption of a specific message m, we can encrypt m and compare it to the ciphertext. If the result is equal, then m is indeed the message encrypted in the ciphertext.
- It can be proved that if the message M is chosen uniformly at random from Z^*_N, then the RSA assumption means that no efficient algorithm can recover M from N,e,M^e.

Security of RSA

- Chosen ciphertext attack: (homomorphic property)
  - Textbook RSA is also susceptible to chosen ciphertext attacks:
    - We are given a ciphertext $C=M^e$
    - We can choose a random R and generate $C'=CR^e$ (an encryption of $M \cdot R$).
    - Suppose we can receive the decryption of C'. It is equal to $M \cdot R$.
    - We divide it by R and reveal M.

Padded RSA

- In order to make textbook RSA semantically secure we must change it to be a probabilistic encryption
- For example, we could pad the message with random bits.
  - Suppose that messages are of length $|N|-L$ bits
  - To encrypt a message M, choose a random string r of length L, and compute $(r \mid M)^e \bmod N$.
  - When decrypting, output only the last $|N|-L$ bits of $C^d \bmod N$
- Any message has 2^L possible encryptions. L must be long enough so that a search of all 2^L pads is inefficient.
- There is no known proof that this secure.
- Similar schemes are known to be secure under certain assumptions

Is it safe to use a common modulus ?

- Consider the following environment:
  - There is a global modulus N. No one knows its factoring.
  - Each party has a pair (e_i,d_i), such that $e_i,d_i = 1 \bmod \phi(N)$.
    - Used as a public/private key pair.
- The system is insecure.
- Party 1, knowing (e_1,d_1)
  - can find a multiple of $\phi(N)$, since $e_1 \cdot d_1 = c \cdot \phi(N)+1$.
  - Using it, can find d_i for any other party i. (I'm hiding some details here.)

RSA with a small exponent

- Setting $e=3$ enables efficient encryption
- Might be insecure if not used properly
  - Assume that the message is short, for example $|M|<|N|/3$
  - In this case, $M^3 < N$, and therefore $M^3 \bmod N = M^3$ (over the integers).
  - For example, $M=10$. In this case $M^3 \bmod N = 1000$. (If $N>1000$.)
  - Extracting roots over the integers is easy, and therefore it is easy to find M.

RSA with a small exponent

- Another security problem with using short exponents (for example, $e=3$)
- Assume three users with public keys N_1, N_2, N_3.
  - Alice encrypts the same (long) message to all of them
    - $C_1 = m^3 \mod N_1$
    - $C_2 = m^3 \mod N_2$
    - $C_3 = m^3 \mod N_3$
- Can an adversary which sees C_1, C_2, C_3 find m?
  - $m^3 < N_1 N_2 N_3$
  - N_1, N_2 and N_3 are most likely relatively prime (otherwise can factor).
  - Chinese remainder theorem -> can find $m^3 \mod N$ (and therefore m^3 over the integers)
  - Easy to extract 3rd root over the integers.

Digital Signatures

Handwritten signatures

- Associate a document with an signer (individual)
- Signature can be verified against a different signature of the individual
- It is hard to forge the signature…
- It is hard to change the document after it was signed…
- Signatures are legally binding

Desiderata for digital signatures

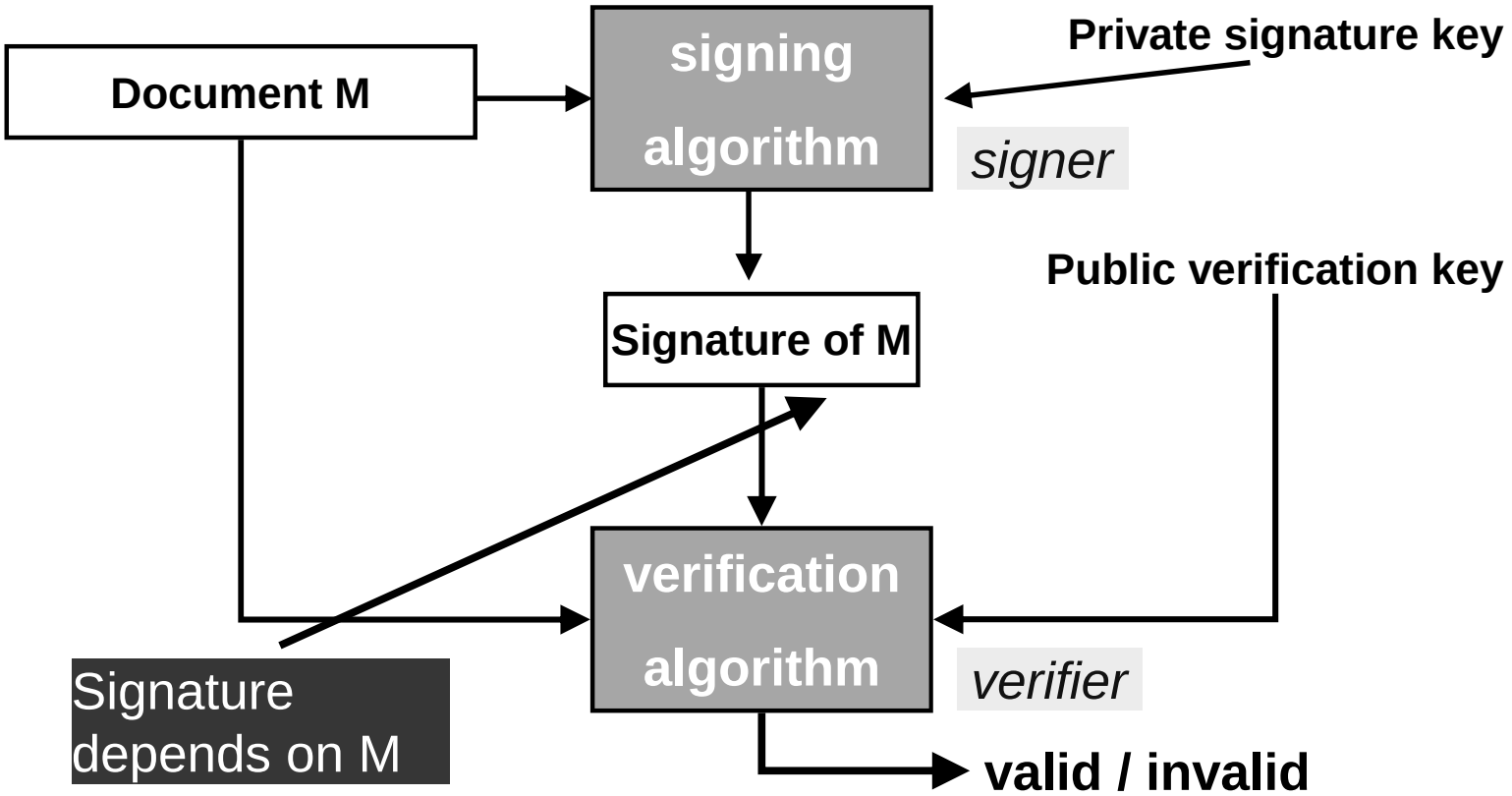
- Associate a document to an signer
- A digital signature is attached to a document *(rather then be part of it)*
- The signature is easy to verify but hard to forge
  - Signing is done using knowledge of a private key
  - Verification is done using a public key associated with the signer *(rather than comparing to an original signature)*
  - It is impossible to change even one bit in the signed document
- *A copy of a digitally signed document is as good as the original signed document.*
- Digital signatures could be legally binding…

Non Repudiation

- Prevent signer from denying that it signed the message
- I.e., the receiver can prove to third parties that the message was signed by the signer

- This is different than message authentication (MACs)
  - There the receiver is assured that the message was sent by the receiver and was not changed in transit
  - But the receiver cannot prove this to other parties
    - MACs: sender and receiver share a secret key *K*
    - If R sees a message MACed with *K*, it knows that it could have only been generated by S
    - But if R shows the MAC to a third party, it cannot prove that the MAC was generated by S and not by R

Signing/verification process



Diffie-Hellman
"New directions in cryptography" (1976)

- In public key encryption
  - The encryption function is a trapdoor permutation *f*
    - Everyone can encrypt = compute *f().* (using the public key)
    - Only Alice can decrypt = compute $f^{-1}()$. (using her private key)
- Alice can use *f* for signing
  - Alice signs m by computing $s=f^{-1}(m)$.
  - Verification is done by computing $m=f(s)$.

- Intuition: since only Alice can compute $f^{-1}()$, forgery is infeasible.
- Caveat: none of the established practical signature schemes following this paradigm is provably secure

Example: simple RSA based signatures

- Key generation: (as in RSA)
  - Alice picks random p,q. Finds $e \cdot d = 1 \bmod (p-1)(q-1)$.
  - Public verification key: (N,e)
  - Private signature key: d
- Signing: Given m, Alice computes $s = m^d \bmod N$.
- Verification: given m,s and public key (N,e).
  - Compute $m' = s^e \bmod N$.
  - Output "valid" iff m'=m.