Introduction to Cryptography
Lecture 8

Rabin's encryption system,
Digital signatures

Benny Pinkas

Reminder: RSA Public Key Cryptosystem

- The multiplicative group $Z_N^* = Z_{pq}^*$. The size of the group is $\varphi(n) = \varphi(pq) = (p\text{-}1)\,(q\text{-}1)$
- Public key*:*
  - $N=pq$ the product of two primes
  - e such that gcd*(e, φ(N))=1*   *(are these hard to find?)*
- Private key:
  - d such that $de \equiv 1$ mod *φ(N)*

- Encryption of $M \in Z_N^*$
  - $C = E(M) = M^e$ mod N
- Decryption of $C \in Z_N^*$
  - $M = D(C) = C^d$ mod N *(why does it work?)*

Reminders

- The Chinese Remainder Theorem (CRT):
  - 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.
    - $x=y$ mod p
    - $x=z$ mod q

- Quadratic Residues:
  - The square root of $x \in Z_p^*$ is $y \in Z_p^*$ s.t. $y^2=x$ mod p.
  - $x \in Z_p^*$ has either *2* or *0* square roots, and is denoted as a Quadratic Residue (QR) or Non Quadratic Residue (NQR), respectively.
  - Euler's theorem: $x \in Z_p^*$ is a QR iff $x^{(p\text{-}1)/2} = 1$ mod p.

Rabin's encryption systems

- Key generation:
  - Private key: random primes p,q (e.g. 512 bits long).
  - Public key: $N=pq$.

- Encryption:
  - Plaintext $m \in Z_N^*$.
  - Ciphertext: $c = m^2$ mod N.   *(very efficient)*

- Decryption: Compute $c^{1/2}$ mod N.

Square roots modulo N

- $\Rightarrow$ Let x be a quadratic residue (QR) modulo $N=pq$, then
  - x mod p is a QR mod p. x mod q is a QR mod q
  - x mod p has *two* roots mod p: y and $p - y$
  - x mod q has *two* roots mod q: z and $q - z$
- $\Leftarrow$ If x is a QR mod p and mod q, it is a QR mod N. (Follows from the Chinese remainder theorem.)
  - Each combination of roots modulo p and q results in a root modulo N.
  - We get four roots modulo pq: A, B, $pq - A$, $pq - B$
  - $(y,z) \to A$, $(p - y, q - z) \to pq - A$
  - $(y, q - z) \to B$, $(p - y, z) \to pq - B$
    - $= (y,z) \cdot (1,-1)$

Square roots modulo N

- $N= pq$.
- If x has a square root modulo N then it has 4 different square roots modulo N.
  - Let A be s.t. $A^2=x$ mod N.
  - Let c be s.t. $c=1$ mod p, $c=-1$ mod q.
  - Then $A, -A, cA, -cA$ are all square roots of x modulo N.
- Exactly ¼ of the elements are QR mod N.
- $QR_N = QR_p \times QR_q$. $|QR_N| = (p-1)(q-1)/4$

- Assume that $p=q=3$ mod 4. (Blum integers.)
  - -1 is an NQR mod p and mod q (Euler's thm).
  - Exactly one of the roots is a QR mod p and a QR mod q.
  - Similarly, for every combination of QR/NQR mod p and mod q.

Finding square roots modulo N

- Need to compute $y=x^{1/2}$ mod N.
- Suppose we know (the private key) p, q.
  - Compute the roots of x modulo p, q. Use Chinese remainder theorem to find x.

- Computing square roots in Z_p^*,
  - Recall, $x \in QR_p$ iff $x^{(p-1)/2}=1$ mod p.
  - Assume $p=3$ mod 4. (p is a Blum integer).
  - Compute the root as $y=x^{(p+1)/4}$ mod p.
    - $(p+1)/4$ is an integer
    - $y^2 = (x^{(p+1)/4})^2 = x^{(p+1)/2} = x^{(p-1)/2}x = x$
  - If $p=1$ mod 4 the computation is more complicated (no deterministic algorithm is known)

Decryption of Rabin cryptosystem

- Input: c, p, q. ($p=q=3$ mod 4)
- Decryption:
  - Compute $m_p = c^{(p+1)/4}$ mod p.
  - Compute $m_q = c^{(q+1)/4}$ mod q.
  - Use CRT to compute the four roots mod N, i.e. four values mod N corresponding to $[m_p , p-m_p] \times [m_q , q-m_q]$

- There are four possible options for the plaintext!
  - The receiver must select the correct plaintext
  - This can be solved by requiring the sender to embed some redundancy in m
    - E.g., a string of bits of specific form

Security of the Rabin cryptosystem

- The Rabin cryptosystem is secure against passive attacks iff factoring is hard. ☺

- The Rabin cryptosystem is completely insecure against chosen-ciphertext attacks ☹

Security of the Rabin cryptosystem

- Security against chosen plaintext attacks
- Suppose there is an adversary that breaks the system
  - Adversary's input: *N, c*
  - Adversary's output: *m* s.t. $m^2 = c$ mod *N*.

- We show a reduction showing that given this adversary we can break the factoring assumption.
- I.e., we build an algorithm:
  - Input: *N*
  - Operation: can ask queries to the Rabin decryption oracle
  - Output: the factoring of *N*.
- Therefore, if one can break Rabin's cryptosystem it can also solve factoring.
- Therefore, if factoring is hard the Rabin cryptosystem is "secure".

The reduction

- Input: *N*
- Operation:
  - Choose random x.
  - Send *N* and $c=x^2$ mod *N*, to adversary.
  - Adversary answers with *y* s.t. $c=y^2$ mod *N*.
  - If *y=x* or *y=N-x,* go back to step *1.* ← happens with prob 1/2
  - Otherwise
    - $x^2 - y^2 = 0$ mod *N*.
    - $0 \neq (x-y)(x+y) = cN = cpq.$
    - Compute gcd*(x+y,N)* , gcd*(x-y,N)* and obtain *p* or *q*.
    - (The gcd is not *N* since *0<x,y<N,* and therefore $-N < x+y, x-y < 2N$, and it's known that $x+y, x-y \neq 0, N$).

Insecurity against chosen-ciphertext attacks

- A chosen-ciphertext attack reveals the factorization of *N*.
- The attacker's challenge is to decrypt a ciphertext *c*.
- It can ask the receiver to decrypt any ciphertext except *c*.
- The attacker can use the receiver as the "adversary" in the reduction, namely
  - Chooses a random *x* and send $c=x^2$ mod *N* to the receiver
  - The receiver returns a square root *y* of *c*
  - With probability ½, *x ≠ y* and *x ≠ -y*. In this case the attacker can factor N by computing gcd*(x-y,N).*

  - (The attack does not depend on homomorphic properties of the ciphertext. Namely, it is not required that E(x)E(y)=E(xy).)

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…
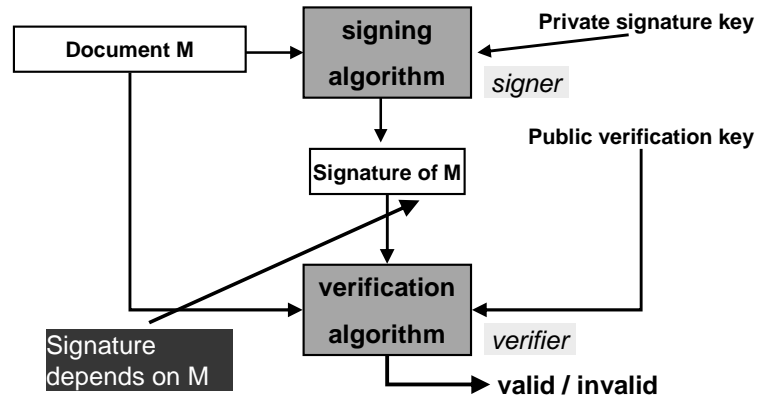- Signatures are legally binding

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

- 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

Document M → **signing algorithm** ← Private signature key

signer

→ Signature of M

Public verification key

verification algorithm *verifier*

Signature depends on M

→ **valid / invalid**

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 $e \cdot d=1$ mod $(p-1)(q-1)$.
  - Public verification key: (N,e)
  - Private signature key: d

- Signing: Given m, Alice computes $s=m^d$ mod N.

- Verification: given m,s and public key (N,e).
  - Compute $m' = s^e$ mod N.
  - Output "valid" iff m'=m.

Message lengths

- A technical problem:
  - |m| might be longer than |N|
  - m might not be in the domain of $f^{-1}()$

Solution:
- Signing: First compute $H(m)$, then compute the signature $f^{-1}(H(M))$. Where,
  - $H()$ is collision intractable. I.e. it is hard to find m, m' s.t. $H(m)=H(m')$.
  - The range of $H()$ is contained in the domain of $f^{-1}()$.
- Verification:
  - Compute $f(s)$. Compare to $H(m)$.

- Use of $H()$ is also good for security reasons. See below.

5

Security of using hash function

- Intuitively
  - Adversary can compute $H()$, $f()$, but not $f^{-1}()$.
  - Can only compute $(m, H(m))$ by choosing m and computing $H()$.
  - Adversary wants to compute $(m, f^{-1}(H(m)))$.
  - To break signature needs to show s s.t. $f(s) = H(m)$. (E.g. $s^e = H(m)$.)

  - Failed attack strategy 1:
    - Pick s, compute $f(s)$, and look for m s.t. $H(m) = f(s)$.
  - Failed attack strategy 2:
    - Pick m, m' s.t. $H(m) = H(m')$. Ask for a signature s of m' (which is also a signature of m).

    - (If $H()$ is not collision resistant, adversary could find m, m' s.t. $H(m) = H(m')$.)
  - This doesn't mean that the scheme is secure, only that these attacks fail.

6