Introduction to Cryptography
Lecture 9

Rabin encryption, Digital signatures, Public Key Infrastructure (PKI)

Benny Pinkas

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: $c = m^2 \bmod N$. *(very efficient)*

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

Square roots modulo *N*

- $\Rightarrow$ Let *x* be a quadratic residue (QR) modulo *N=pq, t*hen
  - *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 also a QR mod *N.* (Follows from the Chinese remainder theorem.)

Square roots modulo *N*

- If x has a square root modulo *N* then it has 4 different square roots modulo *N*.
  - Let *A* be s.t. $A^2 = x \bmod N$.
  - Let *c* be s.t. $c = 1 \bmod p$, $c = -1 \bmod q$.
  - Then *A, -A, cA, -cA* are all square roots of *x* modulo *N.*

- Each combination of roots modulo *p* and *q* results in a root modulo *N*.
  - *x* therefore has four roots modulo *pq*:
  - $(y,z) \rightarrow A$, $(p - y, q - z) \rightarrow pq - A$
  - $(y, q - z) \rightarrow B$, $(p - y, z) \rightarrow pq - B$

$= (y,z) \cdot (1,-1)$

Square roots 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.)
  - Then it is easy to see that exactly one of the four roots is a QR mod *N.*
    - Note that -1 is an NQR mod *p* and mod *q* (Euler's thm).
    - Let one of the square roots of *x* modulo *N* be *A.*
    - Then the other square roots are *-A, cA, -cA,* where *c=1* mod *p*, *c=-1* mod *q*.
    - Assume that *A* is a QR mod *N*, and therefore it is a QR *p* and a QR mod *q.* Then none of the other roots is a QR mod *p* and a QR mod *q.*

Finding square roots modulo N

- Need to compute $y=x^{1/2} \bmod N$.
- 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 $x^{(p-1)/2}=1 \bmod p$.
  - Assume $p=3 \bmod 4$. (p is a Blum integer).
  - Compute the root as $y=x^{(p+1)/4} \bmod p$.
    - $(p+1)/4$ is an integer
    - $y^2 = (x^{(p+1)/4})^2 = x^{(p+1)/2} = x^{(p-1)/2}x = x$
  - If $p=1 \bmod 4$ 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 $m_p = c^{(p+1)/4} \bmod p$.
  - Compute $m_q = c^{(q+1)/4} \bmod q$.
  - Use CRT to compute the four roots mod *N*, i.e. four values mod *N* corresponding to (m_p, m_q), $(p-m_p, m_q)$, $(m_p, q-m_q)$, $(p-m_p, q-m_q)$.
- 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
    - Make sure that m is always a QR

Security of the Rabin cryptosystem

- Good news:
  - The Rabin cryptosystem is secure against passive attacks iff factoring is hard. ☺
- Bad news:
  - 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 completely breaks the system
  - Adversary's input: N, c
  - Adversary's output: m s.t. $m^2 = c \bmod N$.
- 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" in the sense defined here.

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.*
  - 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 is known that $x+y,x-y \neq 0,N$).

happens with prob 1/2

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 $c=x^2 \bmod N$ to the receiver
  - The receiver returns a square root y of c
  - With probability ½, $x \neq y$ and $x \neq -y$. 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).)

Comparing RSA and Rabin encryption

- RSA encryption is infinitely more popular than Rabin encryption (also more popular than El Gamal)
- Advantage of Rabin encryption: it seems more secure, security of Rabin is equivalent to factoring and we don't know to show that for RSA.
- Advantages of RSA
  - RSA is a permutation, whereas decryption in Rabin is more complex
  - Security of Rabin is only proven for encryption as $C=M^2 \bmod N$, and this mode
    - does not enable to identify the plaintext
    - is susceptible to chosen ciphertext attack.

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

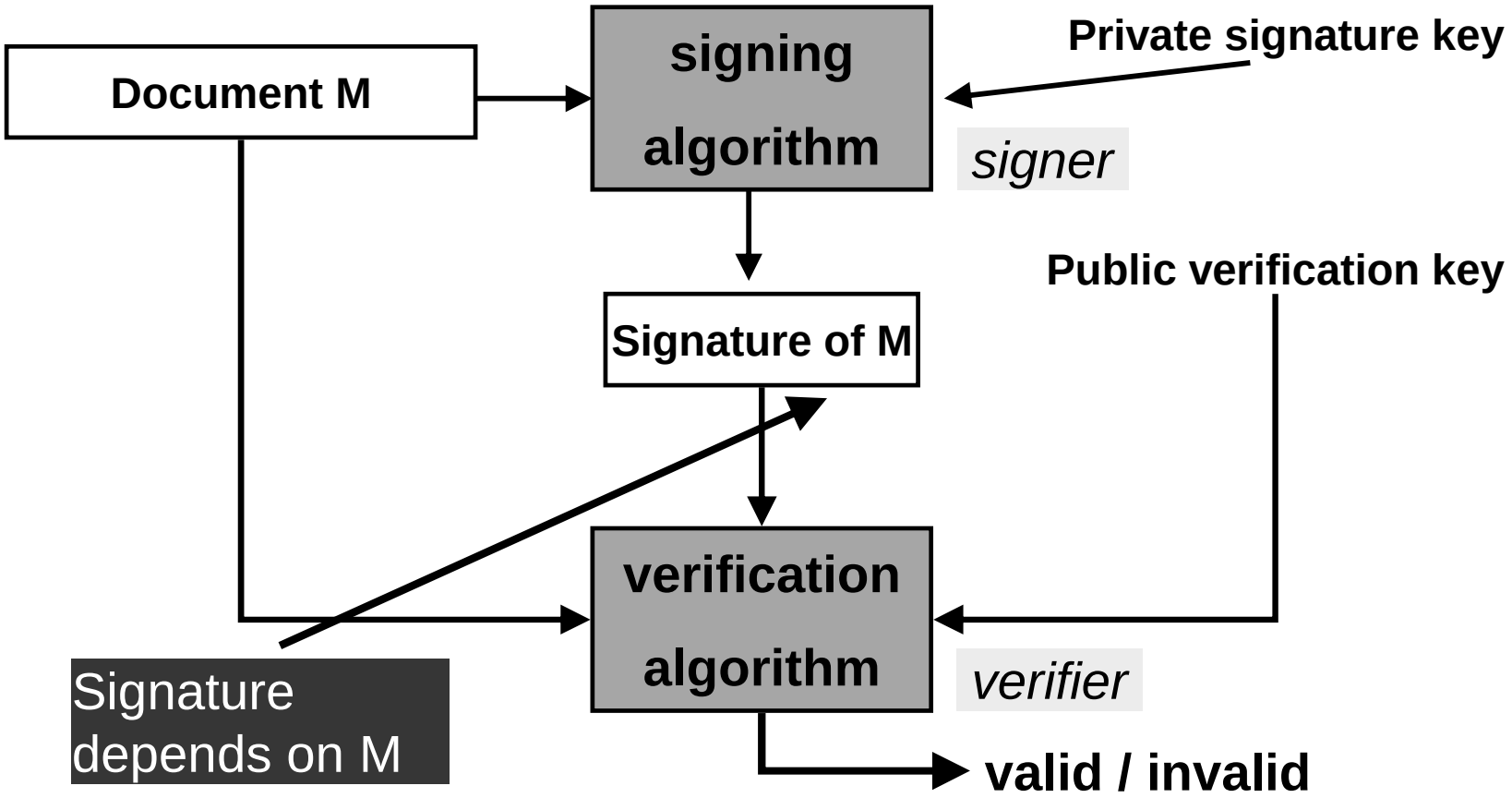
- Associate a document to an signer
- A digital signature is attached to a document *(rather then be part of it)*
- The signature is easy to verify but hard to forge
  - Signing is done using knowledge of a private key
  - Verification is done using a public key associated with the signer *(rather than 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.

Message lengths

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

Solution "hash-and-sign" paradigm:

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

- Use of *H() i*s also good for security reasons. See below.

Security of using a hash function

- Intuitively
  - Adversary can compute *H(), f(),* but not $H^{-1}()$, $f^{-1}()$.
  - 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 does not mean that the scheme is secure, only that these attacks fail.

Security definitions for digital signatures

- Attacks against digital signatures
  - *Key only attack:* the adversary knows only the verification key
  - *Known signature attack:* in addition, the adversary has some message/signature pairs.
  - *Chosen message attack:* the adversary can ask for signatures of messages of its choice (e.g. attacking a notary system).

    (Seems even more reasonable than chosen message attacks against encryption.)

Security definitions for digital signatures

- Several levels of success for the adversary
  - *Existential forgery:* the adversary succeeds in forging the signature of one message.
  - *Selective forgery:* the adversary succeeds in forging the signature of one message of its choice.
  - *Universal forgery:* the adversary can forge the signature of any message.
  - *Total break:* the adversary finds the private signature key.
- Different levels of security, against different attacks, are required for different scenarios.

Example: simple RSA based signatures

- Key generation: (as in RSA)
  - Alice picks random *p,q*. Defines *N=pq* and finds $e \cdot d = 1 \bmod (p-1)(q-1)$.
  - Public verification key: *(N,e)*
  - Private signature key: *d*

- Signing: Given *m*, Alice computes $s = m^d \bmod N$.
- *(suppose that there is no hash function H())*

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

Attacks against plain RSA signatures

- Signature of m is $s=m^d \bmod N$.

- Universally forgeable under a chosen message attack:
  - *Universal forgery:* the adversary can forge the signature of any message of its choice.
  - *Chosen message attack:* the adversary can ask for signatures of messages of its choice.

- Existentially forgeable under key only attack.
  - *Existential forgery:* succeeds in forging the signature of at least one message.
  - *Key only attack:* the adversary knows the public verification key but does not ask any queries.

RSA with a full domain hash function

- Signature is sig(m) = $f^{-1}(H(m)) = (H(m))^d \bmod N$.
  - *H()* is such that its range is *[1,N]*
- *The system is no longer homomorphic*
  - $sig(m) \cdot sig(m') \neq sig(m \cdot m')$
- *Seems hard to generate a random signature*
  - Computing s^e is insufficient, since it is also required to show *m* s.t. $H(m) = s^e$.
- Proof of security in the random oracle model – where *H()* is modeled as a random function

RSA with full domain hash –proof of security

- Claim: Assume that H() is a random function, then if there is a polynomial-time *A()* which performs existential forgery with non-negligible probability, then it is possible to invert the RSA function, on a random input, with non-negligible probability.
- Proof:
  - Our input: *y.* Should compute y^d mod *N.*
  - *A()* queries *H()* and a signature oracle *sig(),* and generates a signature *s* of a message for which it did not query *sig()*.
  - Suppose *A()* made at most *t* queries to *H()*, asking for $H(m_1),\ldots,H(m_t)$. Suppose also that it always queries *H(m)* before querying *sig(m)*. (In particular, it asked for *H(s)*.)
  - We will show how to use *A()* to compute y^d mod *N.*

RSA with full domain hash –proof of security

- Proof (contd.)
- Let us first assume that *A* always forges the signature of m_t (the last query it sends to *H()*),
  - We can decide how to answer *A's* queries to *H(),sig()*.
  - Answer queries to *H()* as follows:
    - The answer to the t^{th} query (m_t) is *y*.
    - The answer to the j^{th} query ($j<t$) is $(r_j)^e$, where r_j is random.
  - Answer to *sig(m)* queries:
    - These are only asked for m_j where $j<t$. Answer with r_j. *(Indeed* $sig(m_j)= (H(m_j))^d = r_j$ *)*
  - *A*'s output is (m_t,s).
    - If *s* is the correct signature, then we found y^d.
    - Otherwise we failed.
  - Success probability the same as the success probability of *A()*.

RSA with full domain hash –proof of security

- Proof (without assuming which m_i *A* will try to sign)
  - We can decide how to answer *A's* queries to *H(),sig()*.
  - Choose a random *i* in *[1,t]*, answer queries to *H()* as follows:
    - The answer to the *i*th query (m_i) is *y*.
    - The answer to the *j*th query ($j \neq i$) is $(r_j)^e$, where r_j is random.
  - Answer to *sig(m)* queries:
    - If $m=m_j$, $j \neq i$, then answer with r_j. *(Indeed* $sig(m_j) = (H(m_j))^d = r_j$ *)*
    - If $m=m_i$ then stop. (we failed)
  - *A*'s output is *(m,s).*
    - *If* $m=m_i$ and *s* is the correct signature, then we found y^d.
    - Otherwise we failed.
  - Success probability is *1/t* times success probability of *A()*.

Rabin signatures

- Same paradigm:
  - $f(m) = m^2 \bmod N$. $(N=pq)$.
  - $Sig(m) = s$, s.t. $s^2 = m \bmod N$. I.e., the square root of m.

- *Unlike RSA,*
  - Not all m are QR mod N.
  - Therefore, only ¼ of messages can be signed.
- *Solutions:*
  - Use random padding. Choose padding until you get a QR.
  - Deterministic padding (Williams system).
- A *total break* given a chosen message attack. (show)
- Must therefore use a hash function H as in RSA.

El Gamal signature scheme

- Invented by same person but different than the encryption scheme. (think why)
- A randomized signature: same message can have different signatures.
- Based on the hardness of extracting discrete logs
- The DSA (Digital Signature Algorithm/Standard) that was adopted by NIST in 1994 is a variation of El-Gamal signatures.

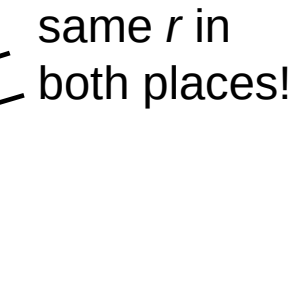
El Gamal signatures

- Key generation:
  - Work in a group Z_p^* where discrete log is hard.
  - Let g be a generator of Z_p^*.
  - Private key $1 < a < p-1$.
  - Public key $p, g, y=g^a$.

- Signature: (of M)
  - Pick random $1 < k < p-1$, s.t. gcd$(k,p-1)=1$.
  - Compute $m=H(M)$.
    - $r = g^k \bmod p$.
    - $s = (m - r \cdot a) \cdot k^{-1} \bmod (p-1)$
  - Signature is r, s.

El Gamal signatures

- Signature:
  - Pick random $1 < k < p-1$, s.t. gcd$(k,p-1)=1$.
  - Compute
    - $r = g^k \bmod p$.
    - $s = (m - r \cdot a) \cdot k^{-1} \bmod (p-1)$
- Verification:
  - Accept if
    - $0 < r < p$
    - $y^r \cdot r^s = g^m \bmod p$



- It works since $y^r \cdot r^s = (g^a)^r \cdot (g^k)^s = g^{ar} \cdot g^{m-ra} = g^m$
- Overhead:
  - Signature: one (offline) exp. Verification: three exps.

El Gamal signature: comments

- Can work in any finite Abelian group
  - The discrete log problem appears to be harder in elliptic curves over finite fields than in Z_p^* of the same size.
  - Therefore can use smaller groups $\Rightarrow$ shorter signatures.
- Forging: find $y^r \cdot r^s = g^m \bmod p$
  - E.g., choose random $r = g^k$ and either solve dlog of g^m/y^r to the base r, or find $s=k^{-1}(m - \log_g y \cdot r)$ *(????)*
- Notes:
  - A different k must be used for every signature
  - If no hash function is used (i.e. sign M rather than $m=H(M)$), existential forgery is possible
  - If receiver doesn't check that $0<r<p$, adversary can sign messages of his choice.