Introduction to Cryptography Lecture

RSA encryption, Rabin encryption, digital signatures

Benny Pinkas

Integer Multiplication & Factoring as a One Way Function.

$$\begin{tikzcd}[column sep=huge] \mathbf{p,q} \arrow[r, bend left, "\textbf{easy}"] & \mathbf{N=pq} \arrow[l, bend left, "\textbf{hard}"] \end{tikzcd}$$

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

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 $x \in Z_N^*$, $x^{\phi(N)} = x^{(p-1)(q-1)} = 1 \bmod N$, and therefore $x^{1+c\cdot\phi(N)} = x \bmod N$

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$(e,\phi(N))=1$
- Private key:
  - d such that $de\equiv 1$ mod $\phi(N)$

- 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?)*

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:
  - gcd$(p,q)=1$ $\Rightarrow$ The extended Euclidian alg finds a,b *s.t.* $ap+bq=1$.
  - Define $c=bq$. It holds that $c=1 \bmod p$, $c=0 \bmod q$.
  - Define $d=ap$. It holds that $d=0 \bmod p$, $d=1 \bmod q$.
  - Given y,z, define $x = cy+dz \bmod N$.
    - $cy+dz = 1y + 0 = y \bmod p$.
    - $cy+dz = 0 + 1z = z \bmod q$.
  - (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

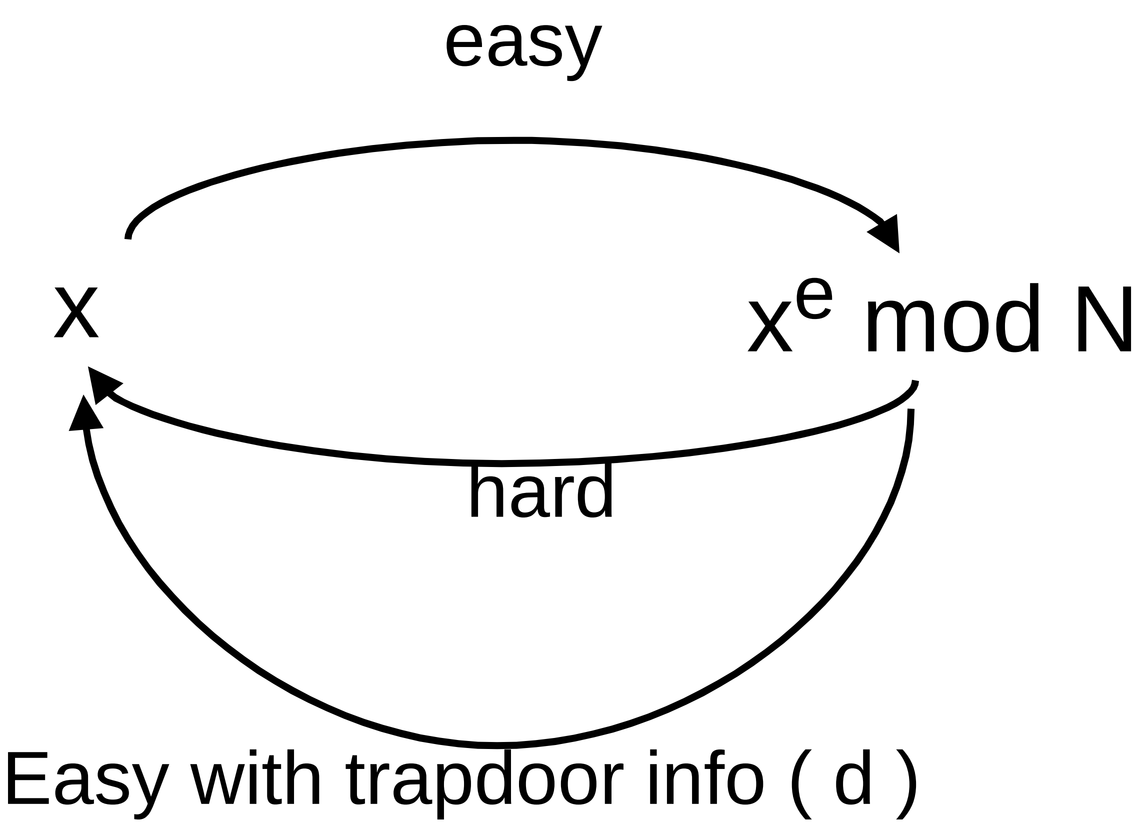
- (For ElGamal encryption, we showed that if the DDH assumption holds then El Gamal encryption has semantic security.)
- If factoring N is easy then RSA is insecure
  - (factor $N \Rightarrow$ find $p,q \Rightarrow$ find $(p-1)(q-1) \Rightarrow$ find d from e)
- Factoring assumption:
  - For a randomly chosen p,q of appropriate 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.
- Therefore, "hardness of finding d assumption" no stronger than hardness of factoring.

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
- Example: $f_{g,p}(x) = g^x \bmod p$ is *not* a trapdoor one way function.
- Example: the assumption that RSA is a trapdoor OWF
  - $f_{N,e}(x) = x^e \bmod N$. (assumption: for a random N,e,x, inverting is hard.)
  - The trapdoor is d s.t. $ed = 1 \bmod \phi(N)$
  - $[f_{N,e}(x)]^d = x \bmod N$

RSA as a One Way Trapdoor Permutation



RSA assumption: cautions

- The RSA assumption is quite well established:
  - RSA is actually a Trapdoor One-Way *Permutation*
  - Hard to invert on random input (if you don't know the secret key)
- But is it a secure cryptosystem?
  - Given the assumption it is hard to reconstruct the input, 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 leak
  - And… adversary can detect a repeating message
- And, of course, as a deterministic cipher RSA does not provide semantic security.

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 $e_i,d_i = 1$ mod $\phi(N)$.
    - Used as a public/private key pair.

- The system is insecure.

- Party 1, knowing (e_1,d_1)
  - can factor N
  - Find d_i for any other party i.

RSA with a small exponent

- Setting $e=3$ enables efficient encryption
- Might be insecure if not used properly
  - Assume three users with public keys N_1, N_2, N_3.
  - Alice encrypts the same message to all of them
    - $C_1 = m^3 \bmod N_1$
    - $C_2 = m^3 \bmod N_2$
    - $C_3 = m^3 \bmod N_3$
- 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 we can factor them).
  - Chinese remainder theorem -> can find $m^3 \bmod N$ (and therefore m^3 over the integers)
  - Easy to extract 3rd root over the integers.

Reminder: RSA Public Key Cryptosystem

- The multiplicative group $Z_N^* = Z_{pq}^*$. The size of the group is $\varphi(n) = \varphi(pq) = (p-1)\,(q-1)$
- Public key:
  - $N=pq$ the product of two primes
  - e such that $\gcd(e, \varphi(N))=1$ *(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?)*

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.
    - $x=y \bmod p$
    - $x=z \bmod q$

- Quadratic Residues:
  - The square root of $x \in Z_p^*$ is $y \in Z_p^*$ s.t. $y^2=x \bmod p$.
  - $x \in Z_p^*$ 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 $x^{(p-1)/2} = 1 \bmod p$.

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: $c = m^2 \bmod N$. *(very efficient)*
- Decryption: Compute $c^{1/2} \bmod N$.

Square roots modulo N

- $\Rightarrow$ Let x be a quadratic residue (QR) modulo $N=pq$, 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 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. $A^2=x$ 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.

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

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

Square roots modulo *N*

- If x has a square root modulo *N* then it has 4 different 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.)
  - Therefore -1 is an NQR mod *p* and mod *q* (Euler's thm).
  - We know that the square roots of *x* modulo *N* are *A, -A, cA, -cA,* where $A^2=x$ mod *N*, and *c=1* mod *p*, *c=-1* mod *q*.
  - Therefore exactly one of the roots is a QR mod *p* and a QR 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 $[m_p , p\text{-}m_p] \times [m_q , q\text{-}m_q]$

- 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

- 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 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 $c=x^2 \bmod N$, to adversary.
  - Adversary answers with y s.t. $c=y^2 \bmod N$.
  - If $y=x$ or $y=N-x$, go back to step *1.*
  - Otherwise
    - $x^2 - y^2 = 0 \bmod N$.
    - $0 \neq (x-y)(x+y) = cN = cpq.$
    - 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 $c=x^2$ mod 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 $C=M^2$ mod 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

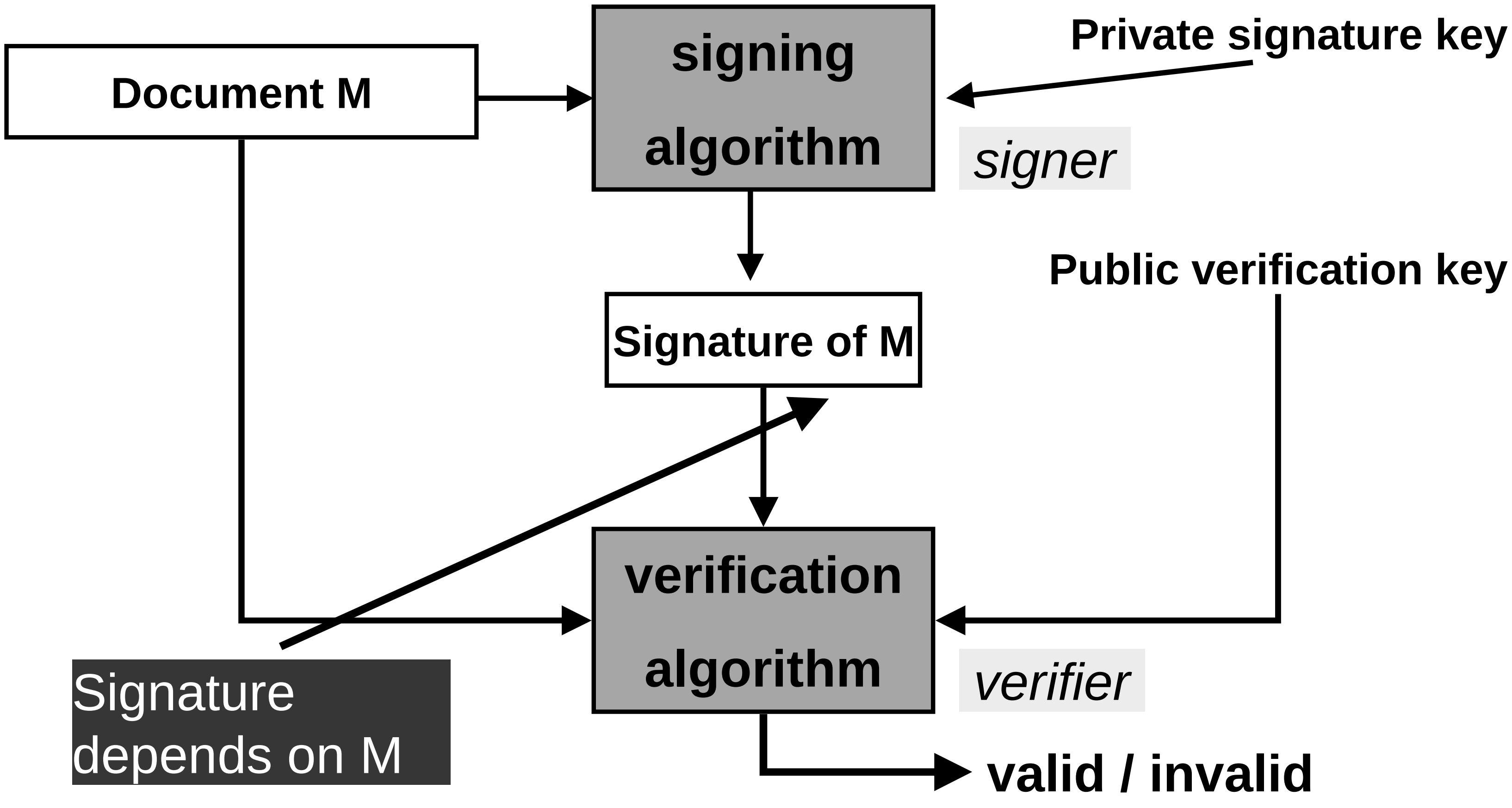
- Associate a document to a signer
- A digital signature is attached to a document *(rather then be part of it)*
- The signature is easy to verify but hard to forge
  - Signing is done using knowledge of a private key
  - Verification is done using a public key associated with the signer *(rather than 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 $e \cdot d=1$ 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.

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.