Introduction to Cryptography
Lecture 7

Digital Signatures

Benny Pinkas

Handwritten signatures

- Associate a document to 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

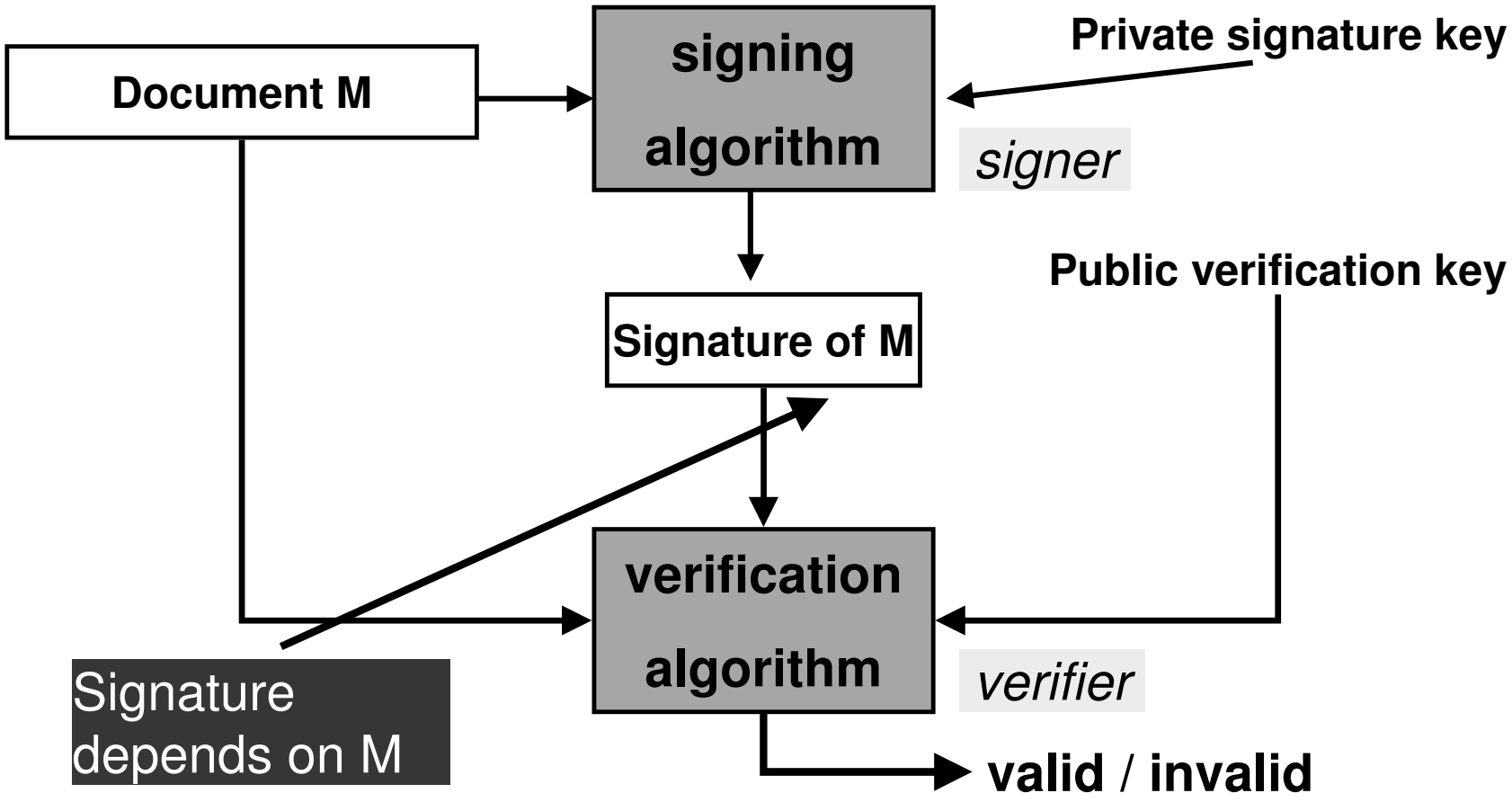
- Associate a document to an signer
- A digital signature is attached to a document *(rather then be part of it)*
- The signature is easy to verify but hard to forge
  - Signing is done using knowledge of a private key
  - Verification is done using a public key associated with the signer *(rather than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



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:

- 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 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 $H()$.
  - 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.

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 attack against encryption.

Security definitions for digital signatures

- Several levels of success for the adversary
  - *Existential forgery:* succeed 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.

Attacks against plain RSA signatures

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

- Universally forgeable under 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 one message.
  - *Key only attack:* the adversary knows the public verification key.

RSA + hash function

- Signature is sig(m) = $f^{-1}(H(m)) = (H(m))^d \bmod N$.
- *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. $H(m) = s^e$.
- Still, it is hard (but possible) to design a signature scheme and prove that it is secure based on the RSA assumption and collision intractability.

Rabin signatures

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

- *Unlike RSA,*
  - Not all m are QR mod N.
  - 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.
- Must 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 DSS (Digital Signature Standard) 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
    - $0 < r < p$
    - $y^r \cdot r^s = g^m \bmod p$

same r in both places!

- 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
  - Therefore can use smaller groups -> 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.

One time signatures

- Signature scheme for a single message

- Example: to sign a single bit
  - Private signature key: $x_0, x_1 \in \{0,1\}^k$
  - Public verification key: $h_0=h(x_0)$, $h_1=h(x_1)$, where h is one-way
  - Signature (of bit b): x_b
  - Verification: check that $h(x_b) = h_b$

- Very efficient
- Given signature of b, adversary cannot fake a signature of $1-b$

One time signatures

- Signing message of size *n:*
  - Private key: $\{ x_{i,0}, x_{i,1} \}_{i=1..n}$
  - Public key: $\{ h(x_{i,0}), h(x_{i,1}) \}_{i=1..n}$
  - Signature of $b_1,\ldots,b_n$: $x_{1,b1},\ldots,x_{n,bn}$
  - Very efficient
    - Can use a full signature scheme to sign public key of one-time scheme (offline).
    - When it is required to sign *m*, signing can be done very efficiently.
  - What happens if two different messages are signed with the same public key?