CS 578 – Cryptography

Prof. Michael Backes

Digital Signatures

June 20, 2006

Administrative Announcements

- Handouts:
  - New exercise sheet
  - New lecture notes

Recall: RSA Trapdoor Permutation

- Setup:
  - Pick random n-bit primes p, q (e.g., n = 512 bits)
  - Set $N = p \cdot q$.
  - Choose arbitrary value e such that $\gcd(e, \varphi(N)) = 1$
- Function: $F: Z_N \rightarrow Z_N$
  - $F(x) = x^e \bmod N$
- Trapdoor for given pk = (e,N):
  $d := e^{-1} \bmod \varphi(N)$

Recall: Improving RSA's performance

- Speeding up RSA by small values of e (fast function evaluation → fast encryption)
  - Minimal possible value: e = 3
    → be careful: sending the same message encrypted with 3 different public keys then breaks naïve RSA!
  - Recommended value: $e=65537=2^{16}+1$
    Runtime for encryption: 17 modular multiplications.
- Chinese Remainder Theorem to compute separately modulo p and q
  - Exponents and modulo shorter by half the size
  - Additionally: Two multiplications, one addition
  - Gain in speed: roughly factor 4

Recall: Wiener's Attack (Sketch)

- (Bad) Idea: Use small values of d to speed up decryption
- Fact: Cannot make d small and still stay secure !!
  - Wiener '89: Suppose $d < 1/3\ N^{0.25}$. Then given (N,e), it's easy to recover d.
  - Boneh, Durfee, Frankel '98: Suppose $d < N^{0.292}$. Then given (N,e), it's easy to recover d.
  - Open problem for $d < N^{0.5}$

Recall: Wiener's Attack (Sketch)

- Recall: $e \cdot d = 1 \pmod{\varphi(N)} \rightarrow \exists\, k \in Z: e \cdot d = k \cdot \varphi(N) + 1$
- Thus $\left| \frac{e}{\varphi(N)} - \frac{k}{d} \right| \leq \frac{1}{d\varphi(N)}$
- $\varphi(N) = N-p-q+1$, hence $|N- \varphi(N)| \leq p+q \leq 3N^{0.5}$
- If $d < 1/3\ N^{0.25}$, the following holds:
$$\left| \frac{e}{N} - \frac{k}{d} \right| \leq \frac{1}{3d^2}$$
- Theorem (Continued Fractions): If gcd(a,b) = gcd(c,d) = 1, and
$$\left| \frac{a}{b} - \frac{c}{d} \right| \leq \frac{1}{2d^2}$$
then c/d is one of the convergents of the continued fraction expansion of a/b.

Summary of Public-key Encryption

- Need cryptographic assumptions: Usually number-theoretic: DLog, CDH, DDH, Factoring, RSA, Quadratic residuosity, etc.
- ElGamal: Public-key encryption scheme strongly related to Diffie-Hellman key exchange, CPA-secure in G_q, vulnerable to active attacks
- Cramer-Shoup encryption: CCA-2 secure encryption scheme, extension of ElGamal
- RSA: Trapdoor one-way permutation, extensively used for encryption and signatures, naïve use as encryption insecure
- OAEP, OAEP+, SAEP+: CCA2-secure encryption schemes in the random oracle model, rely on trapdoor permutation/RSA
- Next: Digital signatures

Digital Signatures

- Main Idea: signature is function of message being signed and a secret key
- Main properties of signature schemes:
  - Only knowing the secret key allows for creating signatures
  - Everybody can verify the validity of signatures using the respective public key
  - Signatures serve as undisputable evidence that the respective person signed the message

Definition of Digital Signatures

- Definition (Digital Signature Scheme): A digital signature scheme is a triple of efficient algorithms (Gen,S,V):
  - Gen(n): Generates a secret/public key pair (pk,sk) for security parameter n
  - S(sk,m): Creates a signature sig for message m and secret key sk
  - V(pk,m,sig): Outputs true or false.

  such that for all (pk,sk) ← Gen(n), and for all m: If sig ← S(sk,m), then V(pk,m,sig) = true.

Definition of Digital Signatures

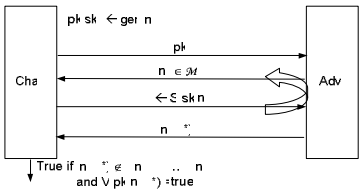
- Technical difference to public-key encryption: Signature schemes often maintain state
- Differences to MACs and consequences thereof:
  - Key transmission has to be authentic but not necessarily secret
  - Non-repudiation!
    (Can use signatures as evidence at a third party)

On Attacker Goals

- Definition (Secure signature schemes, intuitively):
  - Attacker's power: chosen-message attack
    Attacker outputs $m_1 \ldots m_q$, and gets $t_i \leftarrow S(sk,m_i)$
  - Attacker's goal: existential forgery
    Produce valid pair (m^*,t^*), i.e., $V(pk,m^*,t^*) = true$, where $(m^*,t^*) \notin \{(m_1,t_1), \ldots, (m_q,t_q)\}$
- Similar to MACs: Attacker cannot even forge signatures on non-sensitive messages

Security of Digital Signatures

- For a signature scheme Sig = (Gen,S,V), define the following CMA game:



Security of Digital Signatures

- The advantage of adversary A attacking Sig is $Adv^{CMA}[A,Sig] = Pr[\text{Challenger outputs true}]$
- Definition (Security of Digital Signatures): A digital signature scheme Sig=(Gen,S,V) is a secure against existential forgery under chosen-message attack (CMA) if for all efficient algorithms A: $Adv^{CMA}[A,Sig]$ is negligible.

Constructing Signature Schemes

1. Can we construct signatures from generic one-way functions (block-ciphers, PRPs)?
- Yes! Work by Lamport, Merkle, …
- Example (one-time signature):
- Generation:
  - Choose 2n random values:
    $sk = (r_1^0, r_1^1), (r_2^0, r_2^1), \ldots, (r_n^0, r_n^1)$
    $pk = F(r_1^0), F(r_1^1), F(r_2^0), F(r_2^1),\ldots, F(r_n^0), F(r_n^1)$

Constructing Signature Schemes

- S(sk,m)
  - Write m in binary representation $m_1 \ldots m_n \in \{0,1\}^n$
  - $sig = r_1^{m1}, r_2^{m2},\ldots, r_n^{mn}$
- V(pk,m,sig)
  - Test if $F(r_i^{mj})$ is correct for i = 1,…,n.
- Secure for one-time signing if F is one-way [proof on the board]

Constructing Signature Schemes

- Problem: Cannot sign more then one message using (pk,sk)
- Can be fixed: can be made to work for multiple messages. However, there are drawbacks:
  - Signatures are very long
  - Signer must maintain state
- Benefit: Fast!

Constructing Signature Schemes

1. Can we construct signatures from generic one-way functions (block-ciphers, PRPs)? Yes!
2. Can we use the F^{DLog} function?

- Yes! → ElGamal signatures

ElGamal Signatures

- From 1985, motivated by hardness of DLog
- Key generation as for ElGamal encryption in Z_p^*, pk = (p,g,h = g^x), sk = (p,g,x)
- S(sk,m):
  - Choose r $\leftarrow_R$ {1,…,p-1}
  - Compute

    $s := g^r \bmod p \qquad t := (m - xs)r^{-1} \bmod p-1$

    and let $sig := (s,t)$
- V(pk,m,sig = (s,t))
  - Output true iff

    $h^s s^t = g^m \bmod p$

Attacks on ElGamal Signatures

- Existential forgery under passive attacks:
  - Adversary has to solve
    $$h^s s^t = g^m \bmod p$$
  - Idea: Choose s of the form $s = g^i \cdot h^j \bmod p$ and solve equations in the exponent
    [proof on the board]
- Countermeasure: Sign hash(m) instead of m (later more)

Variation of ElGamal Signatures

- Key generation as for ElGamal signatures
- V(pk,m,sig = (s,t))
  - Output true iff
    $$h^t s^s = g^m \bmod p$$
- S(sk,m):
  - Choose $r \leftarrow_R \{1,\ldots,p\text{-}1\}$
  - Compute
    $s := g^r \bmod p \qquad t := (m - rs)x^{-1} \bmod p - 1$
    and let $sig := (s,t)$
- Advantage: Precomputation of x^{-1} possible

Schnorr Signatures

- Extension of ElGamal to subgroups G_q of Z^*_p
- Additionally uses hash function H
  → message space {0,1}*
- Key generation as for ElGamal encryption in G_q,
  pk = (q,p,g,h = g^x), sk = (q,p,g,x)
- S(sk,m):
  - Choose $r \leftarrow_R \{1,\ldots,q\}$
  - Compute
    $s := H(m \| g^r) \qquad t := (r + xs) \bmod q$
    and let $sig := (s,t)$
- V(pk,m,sig = (s,t))
  - Output true iff
    $$H(m \| g^t h^{-s}) = s$$

Digital Signature Algorithm (DSA)

- Invented by RSA 1991, federal standard
- Key generation as for ElGamal encryption in G_q, pk = (q,p,g,h = g^x), sk = (q,p,g,x),
- q = 160 bits, 512 ≤ log p ≤ 1024, log p multiple of 64
- S(sk,m):
  - Choose r $\leftarrow_R$ {1,…,q}
  - Compute

    $s := (g^r \bmod p) \bmod q \quad t := (SHA\text{-}1(m) + xs)r^{-1} \bmod q$

    If s = 0 or t = 0, start with a new r. Let $sig := (s,t)$
- V(pk,m,sig = (s,t))
  - Output true iff

    $(g^{SHA\text{-}1(m)t^{-1} \bmod q} h^{st^{-1} \bmod q} \bmod p) \bmod q = s$

Constructing Signature Schemes

1. Can we construct signatures from generic one-way functions (block-ciphers, PRPs) → Yes!
2. Can we use the F^{DLog} function? → Yes!
3. Trapdoor permutation → efficient and simple signatures

Naïve RSA-based Signatures

- Naive use:
- Setup:
  - Generate public parameters pk = (N,e)
  - Keep trapdoor sk = d secret
- S(sk,m) := m^d mod N
- V(pk,m,sig) : Test if sig^e = m mod N

Attacks on Naïve RSA-based Signatures

- Existential forgery under passive attacks:
  - Given (N,e)
  - Goal: Find pair (m,sig) with $sig^e = m \bmod N$
  - Pick arbitrary sig, output (sig^e,sig)
  - Forgery on the message $sig^e \bmod N$

Attacks on Naïve RSA-based Signatures

- Selective forgery under active attacks:
- Blinding attack:
  - Adversary want signature on m
  - Pick random $r \in Z_N$ and compute $m^* = m \cdot r^e \bmod N$
  - Asks signer to sign m^*
  - Result: (m^*,s^*) where $(s^*)^e = m^* \bmod N$
  - Compute $s \leftarrow s^* / r \bmod N$
  - Indeed we have
    $s^e = (s^*)^e / r^e = m^* / r^e = (m \cdot r^e) / r^e = m$
- Originally attack against RSA signature schemes
- Now special primitive (blind signature), used in anonymous digital cash, election systems, etc.

Attacks on Naïve RSA-based Signatures

- Countermeasures:

1. Add redundancy to the message
2. Hash message before signing
   - Hash-then-sign general concept, often even introduced as “the only way to sign” in books
   - Advantage: Allows for signing arbitrarily long messages
   - Required properties for hash to make the system secure?

RSA using Redundancy

- Instead of signing m, sign (m,red(m)) for a redundancy function red
- Test becomes $sig^e = (m,red(m)) \bmod N$
- Resistance against common attacks? [investigate on the board]
- Examples of redundancy:
  - 100 zeroes (bad against homomorphic attacks)
  - Chaotic redundancy, e.g., use DES with fixed (and publicly known) key. OK, and even in ISO standard

RSA using Hash-then-Sign

- Properties of hash-then-sign for RSA:
- Collision-resistance necessary?
  - Suppose that H is not collision-resistant:
  - Existential forgery: Find $m \neq m'$ such that $H(m) \neq H(m')$
  - Ask for signature on m
  - Obtain signature on m'
- One-wayness necessary?
  - Suppose that H is not one-way:
  - Compute $m^* = sig^e \bmod N$ and m such that $hash(m) = m^*$.
  - Output (m, sig).

Full-Domain Hash

- Works for every trapdoor permutation. Here specifically for RSA
- Generation as for RSA
- S(sk,m):
  - D := Hash(m) for Hash: $\{0,1\}^* \rightarrow Z^*_N$
  - $sig := D^d \bmod N$

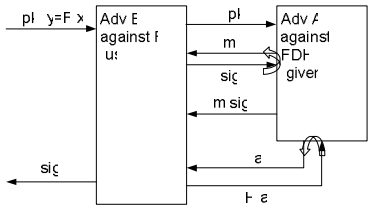
  (Hash: collision-resistance + maps into the full set Z^*_N (thus the name "full domain"))
- V(pk,m,sig):
  - D := Hash(m)
  - Test if $D = sig^e \bmod N$

Full-Domain Hash

- Theorem (Bellare, Rogaway'94): If hash is a "random oracle" then RSA-FDH is existentially unforgeably under chosen-message attack assuming that RSA is one-way.

  [proof on the board]

Full-Domain Hash – Proof Sketch



RSA-based Signatures in Practice

- RSA-based signatures in practice: PKCS1 V1.5 mode 1
- Let N = pq be 1024 bit value
- H = {0,1}* → $\{0,1\}^{160}$ (e.g., SHA-1)
- S(sk,m):
  - d := Hash(m), D ∈ $\{0,1\}^{160}$
  - EB = [$\underbrace{01}_{16\text{ bits}}$ 11 11 11 11 … 00 || d]
  - sig := $(EB)^d$ mod N

On Provably secure Signatures

- Efficient CMA-secure signatures exist nowadays
- Usually based on stronger assumptions, e.g., “Strong RSA assumption”: Given (c,N), hard to find (m,e) such that $c = m^e \bmod N$
- Typically not used in practice. I would though…
- Lots of variants: Undeniable/fail-stop/group/blind/private contract signatures, …
- Generic constructions from zero-knowledge proofs

Summary of Digital Signatures

- CMA-secure signature schemes exist if OWF exist
- First one-time signatures: Lamport, Merkle, etc.
- ElGamal signatures: Security based on hardness of discrete logarithms, security not proven (!), relies on ability to solve equations in the exponents, common variants: Schnorr, DSA
- RSA-based signatures: Naïve use insecure, Full-domain Hash (FDH) based on any trapdoor permutation. CMA-secure in the random oracle model
- Hash-then-Sign: Often used in practice, however not mandatory (!), lots of books do as if this had to be done
- Next: Key management, certificates, trust