Lecture 12

Digital Signatures from one-way functions

Signatures vs. MACs

Signatures

- n users require only n secret keys
- Same signature can be verified by all users
- Publicly verifiable and transferable
- Provide non-repudiation

MACs

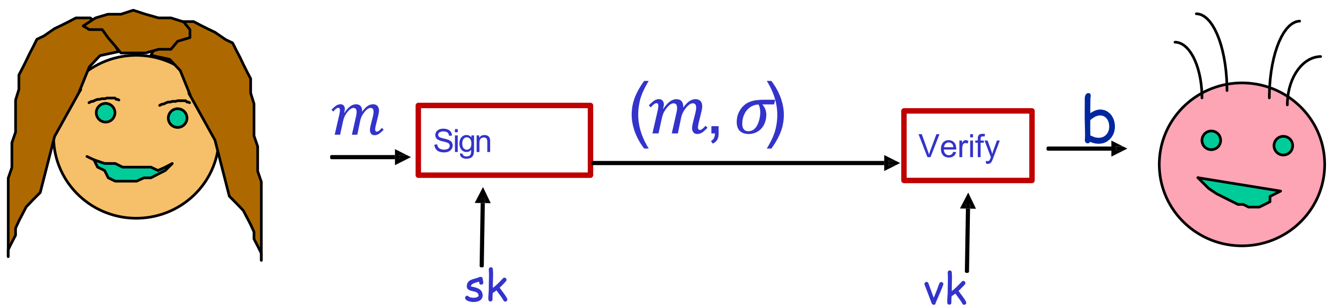
- n users require $\approx n^2$ secret keys
- Privately verifiable and non-transferable
- More efficient (2-3 orders of magnitude faster)

Digital Signatures

Key-generation: Gen (1^n) outputs pair
signing key sk and verification key vk

Signing: Sign(sk,m) outputs a signature s σ

Verification: Verify(vk,m,σ) outputs accept/reject (1/0)



Correctness: For every message m.
Verify(vk,m,σ)=accept if $\sigma \in$ Sign(sk,m)

Security of Signatures

- Adv knows vk and can adaptively ask for signatures of messages of its choice
- Adv tries to forge a signature on a new message m

Run Gen(1^n) to get (sk,vk)

$\mathcal{A}^{sign_{sk}}(vk)$

vk ←

(m, σ) →

Scheme Π = (Gen, Sign,Verify) is existentially unforgeable against an adaptive chosen message attack (EU-ACMA) if

∀ppt adversary $\mathcal{A}$ ∃neg function s.t. ∀n sufficiently large

Prob [Verify(vk,m,σ)=Accept &

$m \notin \{m_i$ asked to be signed by $\mathcal{A}\}]$ <neg(n)

Signatures vs MACS

There **do not** exist EU-ACAM signature schemes against unbounded adversaries. This holds regardless of the key length.

Why?

Secure mac schemes against unbounded adversaries exist with a key as long as the number of messages to be signed.

RSA Digital Signature Scheme 77

The first example of a digital signature scheme

- Key Generation(1^n): choose $N=pq$ for $|p| \approx |q|=n/2$ and e,d s.t. $ed=1 \bmod \phi(N)$

  vk=(N,e) the public verifying key

  sk=(N,d) the private signing key.

- Sign((N,d), m):

  sig := $m^d \bmod N$

- Verify ((N,e),m,sig) :

  Accept iff $sig^e \bmod N = m$.

RSA is **existentially forgeable** under Key Only attack.

RSA is **universally forgeable** under Chosen Message Attack

Can not securely sign specialized message sets, e.g. S={0,1}

Hash-then-Sign Paradigm for the Trapdoor Digital Signature Model(e.g.RSA)

Use a public "cryptographic" hash function H

Let Sig(sk,m)=$f^{-1}(H(m))$ ($=H(m)^d$ mod N for RSA)

Verify(vk,m,σ)= accept iff f(sig)=H(m)

Correctness certainly hold. What about unforgeability? Which properties need H have? Is collision resistance (CR) enough?

A: Counter to intuition, no proof of security, even if f is TDP and H is CRH. It depends on H & how H and f interact

Given TRP f, can be secure with one H & insecure with another.

Yet, popular paradigm where for H =MD5, SHA1 etc.

- Basis for standards (e.g., PKCS#1 of RSA inc. DSS of NIST)
- Basically assume that specific combination of F& H is secure

The Random Oracle Model

Theorem: if H is a **random oracle**, then Hashed RSA signatures is EU-ACMA under the assumption that f is trapdoor function (e.g. RSA assumption).

Unfortunately: H is not a random oracle but a deterministic function that everyone can evaluate

- No implication from "security in the random oracle model" to security of the actual scheme. In fact, it was shown that there CANNOT be a "generic” implication.

Todays Outline

- Construction of EU-ACMA from ANY one- way function (no trapdoors)
  1. One-time signatures from OWFs
     - Bounded-length messages
     - Unbounded length messages
  2. From one-time to multi-time: Stateful signatures
  3. Stateless signatures
- Many Flavors of Signatures
  - Incremental Signatures
  - Blind Signatures and Electronic Cash
  - Group Signatures

Signing 1-bit messages from **One-Way Functions** (no trapdoors!) Lamport

Let F be a one-way function collection

- Gen: choose $f \in F_n$, $x_{0,} x_1 \in \text{Domain}(f)$,
  signing key sk $= (x_{0,} x_1\ )$ &
  verifying key vk $=(f(x_0), f(x_1))$

sk $= \boxed{x_o}\boxed{x_{1.}}$

vk $= \boxed{f(x_0)}\boxed{f(x_1)}$

- Sign($(x_{0,} x_1)$, b): output x_b
- Verify($(\underbrace{f(x_0), f(\ x_1)}_{sk})$, b, sig) = accept if $f(\text{sig}) = f(x_b)$

Extension to t-bit Messages: bigger keys

Increase the size of the

signing key sk= $\{(x_0^i, x_1^i)\}_{i=1\ldots t}$

verifying key vk = $\{(f(x_0^i), f(x_1^i))\}_{i=1\ldots t}$

$x^i{}_o$	$x^i{}_1$.
$f(x^i{}_0)$	$f(x^i{}_1)$

- Sign(sk, $b_1 \ldots b_t$) = $x^i{}_{bi}$ for i=1…l

- Verify(vk, $b_1 \ldots b_l$, $\sigma^1 \cdots \sigma^i$) =accept

  if $f(\sigma^i) = f(x_{bi})^i$ for all i=1…t

Extension to t-bit Messages: bigger keys

Increase the size of the

signing key sk= $\{(x_0^i, x_1^i)\}_{i=1\ldots t}$

verifying key vk = $\{(f(x_0^i), f(x_1^i))\}_{i=1\ldots t}$

$x^i{}_o$	$x^i{}_1$.
$f(x^i{}_0)$	$f(x^i{}_1)$

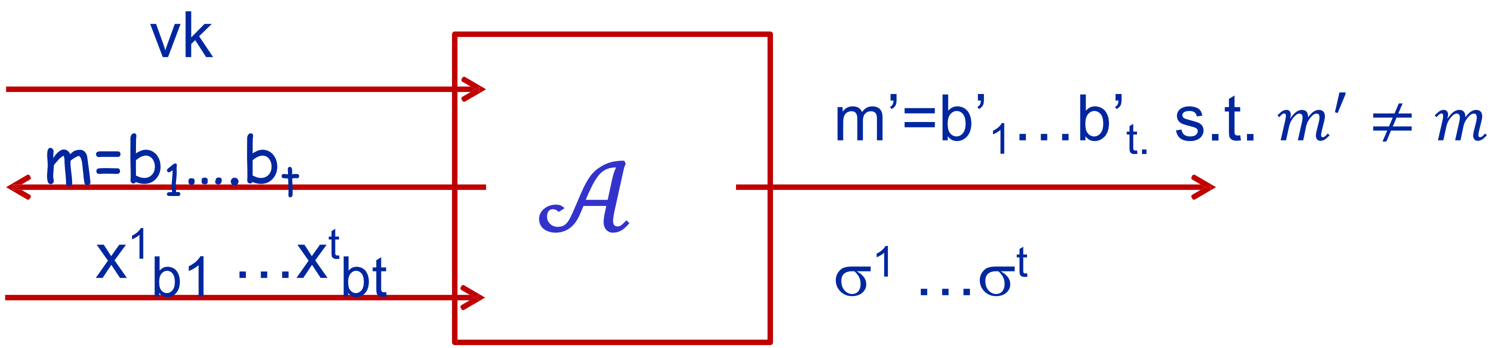
- Sign(sk, $b_1\ldots b_t$) = $x^i{}_{bi}$ for i=1…l

- Verify(vk, $b_1\ldots b_l$, $\sigma^{1\cdots}\sigma^i$) =accept

  if $f(\sigma^i) = f(x_{bi})^i$ for all i=1…t

Security of Lamport's One -Time Scheme

sk =	x^i_o	$x^i_{1.}$
vk =	$f(x^i_o)$	$f(x^i_1)$



Goal: for all ppt $\mathcal{A}$ Prob[$\mathcal{A}$ success] <ε

Intuition: $\exists j$: $b_j' \neq b_j$, this means that there exists A that produced σ^j an inverse of $f(x^j_{b'j})$, which it didn't see before,so A violates the assumption that f is a OWF.

Theorem: Lamport's method is existentially un-forgeable under ACMA for one length t signature

Proof Assume there exists forger A which forges with probability ε. We construct an adversary Inv to invert f with probability better than $\varepsilon/2t$.

Inv (y): choose at random $j \leftarrow \{1,...,t\}$; $b \leftarrow \{0,1\}$

1)choose signing key sk= $(x_0^i, x_1^i)_{i=1\ldots t}$ & verifying key vk = $\{(f(x_0^i), f(x_1^i))\}_{i=1\ldots t}$ at random except for position j where you put y instead of $f(x^j_b)$

2) run A(vk).When it requests a signature on $m = b_1 \cdots b_t$; answer by signing m, unless $b_j = b$; in which case, abort

3) if A forges signature $(\sigma_1, \ldots, \sigma_l)$ on $m' = b'_1 \cdots b'_l$. and $b'_j = b$, then output σ_j, else abort

Claim: Prob (A outputs an $\sigma_j = x$ s.t. $f(x)=y$) = $(1/2)(1/t)\varepsilon$

.

Only Signed 1 message of bounded length

How to Extend to 1 message of unbounded length?

Currently: Size of public key vk grows with number of bits to be signed

Collision Resistant Hash Function (CRHF)

Let k>m

$H:\{0,1\}^k \to \{0,1\}^t$ is collision resistant polynomial time hash function if for all PPT algorithms A, for all k sufficently large:

$\Pr[(x, y) \leftarrow A(1^k) \text{ s.t. } H(x) = H(y) \wedge x \neq y] \leq neg(k)$

- Asymptotically, speak of keyed hash functions

- Do they exist?

Use Collision-Resistant Hash Functions

- Apply a CRH to m to hash it to a smaller string before signingas before with the one-time signature for t size message.
  - The verification and signing keys will include also a description of CRH H
  - sign H(m) rather than signing m directly.
- Security: By reduction to the security of the underlying scheme and the CRH
- Straightforward Analysis
- first time we're proving security of a scheme based on the security of two different cryptographic primitives

Analysis

Let (Gen,Sig,Verify) be a EU-ACMA t-time signature scheme, and H be a CRH.

Claim: (Gen_H,Sig_H,Ver_H) - the new signature scheme for arbitrary length message is EU-ACMA

Proof: Let A be an adversary that forges with ε prob for size k.

Let COLL= the event that the forgery (m*,s*) generated by A is such that H(m*)=H(m) for some previous m that the signing oracle signed for A.

Lemma 1: Prob[COLL] < neg(n)

Assume not. Construct a collision-finder C for H. On input H, C chooses both signing sk and verification keys vk and runs A on vk Event COLL immediately corresponds to a collision in h.

Lemma 2: Prob[A' forges | not COLL] < neg(n).

Assume not . Reduce to the EU-ACMA security of underlying scheme (Gen,Sig,Ver).

Conditions Under which CRHF exist

Example (DLP). Let p be a prime, g generator

- Let $H(x)=g^{x'} h^b$ mod p, for $x=x' | b$ where $x < p-1$
- H compresses by 1 bit
- Collisions $x=x'|b_1$ $y=y'|b_2$ for H can be used to compute the discrete-log $DLOG_g$ (h) mod p

  1) if $b_1=b_2$ then $x'=y'$ (since $g^{x'} = g^{y'}$ & g generator) so must be that $b_1 \neq b_2$ and thus $g^{x'}h^{b1} = g^{y'}h^{b2}$ mod $p \Rightarrow$ (Say b=0) $g^{x'-y'} = h$ mod p and we solved DLP for h.

Better compression: Let $H(x)=g^{x'} h^{x''}$ mod p, for $x=x' | x''$ for large $q|(p-1)$ from 2log q to log (p-1)

Example (Factoring): derive from claw-free example

More generally:

(1) if claw-free permutations exist (no trapdoor), or

(2) if CPA-secure encryption exist with homomorphic addition [see web site]

Todays Outline

- Construction of EU-ACMA from ANY one- way function (no trapdoors)
  - ✓ One-time signatures from OWFs
    - Bounded-length messages
    - Unbounded length messages: |vk|< |m|

  2. From one-time signatures to multi-signatures: Stateful signatures

  3. Stateless signatures
- Many Flavors of Signatures
  - Incremental Signatures
  - Blind Signatures and Electronic Cash

From one-signatures to many-signatures

Idea: When signing a new message m_i

- generate also a new pair (sk_i, vk_i) of (one-time) public and private keys
- sign the pair (m_i, vk_i) instead of just signing m_i. (Note!: can sign |vk|+|m| bits)
- signature of m_i includes all previous signed vk_i's leading to the vk_0 in public-key

Size: The signature grows with number of previous signatures.

Complexity of verification algorithm: need to verify all the one-time signatures of previous vk_i's

Stateful: signer needs to maintain local (secret) state from one signature generation to the next.

Putting it all together: Signing many messages securely from **any** secure one message signature scheme

Let H be a collision resistant hash function (CRH) to t bits

Key Chain Method: start with (G,S,V) that can sign t-bits and let (sk_0,vk_0) be the signing, verifying key pair. Counter i=1

To sign message m_i ,

- choose $new_i=(sk_i, vk_i)$
- Hash $h_i = H(vk_i)$ and let $\sigma_i= S(sk_{i-1},h_i)$

  $\sigma' =S(sk_{i-1}, m_i)$

  $Chain_i = chain_{i-1} || vk_i||h_i||\sigma_i$
- Output $(i,chain_i, m,\sigma')$

- To verify $(i, chain_i, m, s)$

  Verify that $V(vk_{j-1},h_j, \sigma_j)$ =accept & $h_j= H(vk_j)$ (for all j=i..0)

  Verify that $V(vk_{i-1},m,\sigma)$ =accept

  Verify that vk_0 is in the public-key

Proof of Security

Forgery either means

1) find forgery for the original one-time scheme (G,S,V) since each instantiation of (vk,sk) of (G,S,V) is used to sign exactly one t-bit message, or

2) could find collisions, i.e a new (vk', sk') s.t. H (vk')=H(vk_i) for a previous signatures of $h_i = H(vk_i)$.

Final step: Replace CRHF by Universal One Way Hash Function

- A universal one-way hash functions (UOWHFs):
  - adversary cannot choose both x and y s.t. H(x)=H(y)
  - instead, the adversary is given a random x as challenge and must find y such that H(x) = H(y).
  - Adversary's job harder than for CRH, meaning that UOWHFs ⇒CRH but CRH may not ⇒ UOWHF (i.e UOWHF weaker requirement).
- UOWHFs can replace CRH in the signature scheme construction. Revisit the proof and verify this.

- OWF ⇒ UOWHF (Rompel: One-Way Functions are Necessary and Sufficient for Secure Signatures, STOC 1990

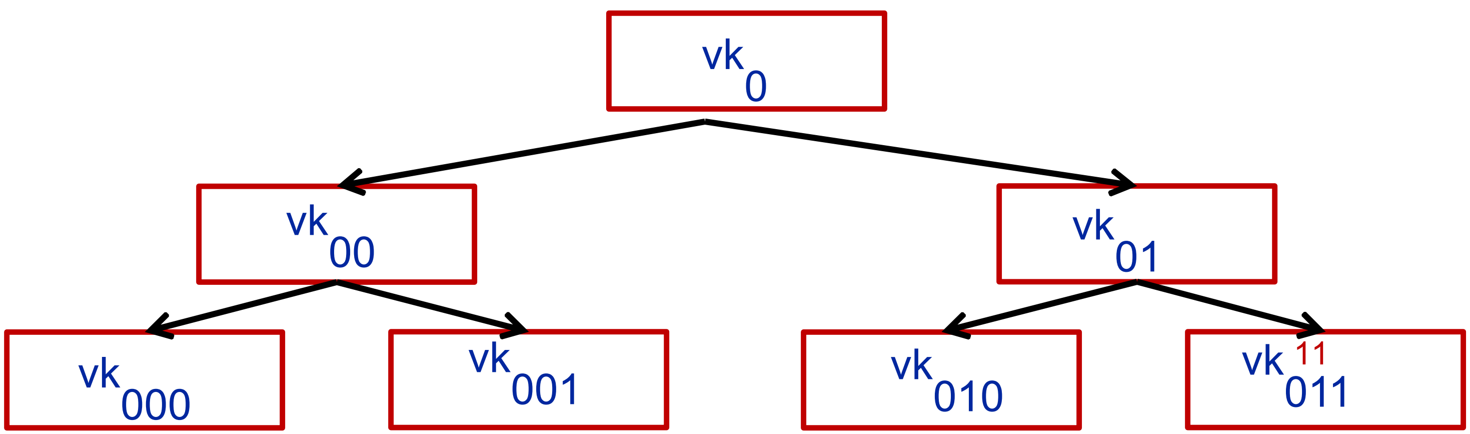
Problem 1:
Size of signatures grows linearly with the history

Signatures which do not grow Linearly with History: Tree solution

- Arrange (sk,vk) pairs in a virtual tree so that (sk_0,vk_0) is in the root, (sk_i,vk_i) are in an internal node specified by path i,
- Instead of a `chain' of previously authenticated $(sk_i,vk_i)$ include in a new signature a `path' from root to leaf of authenticated pairs
- Now for T messages ever to be signed, path-size is logT for each message

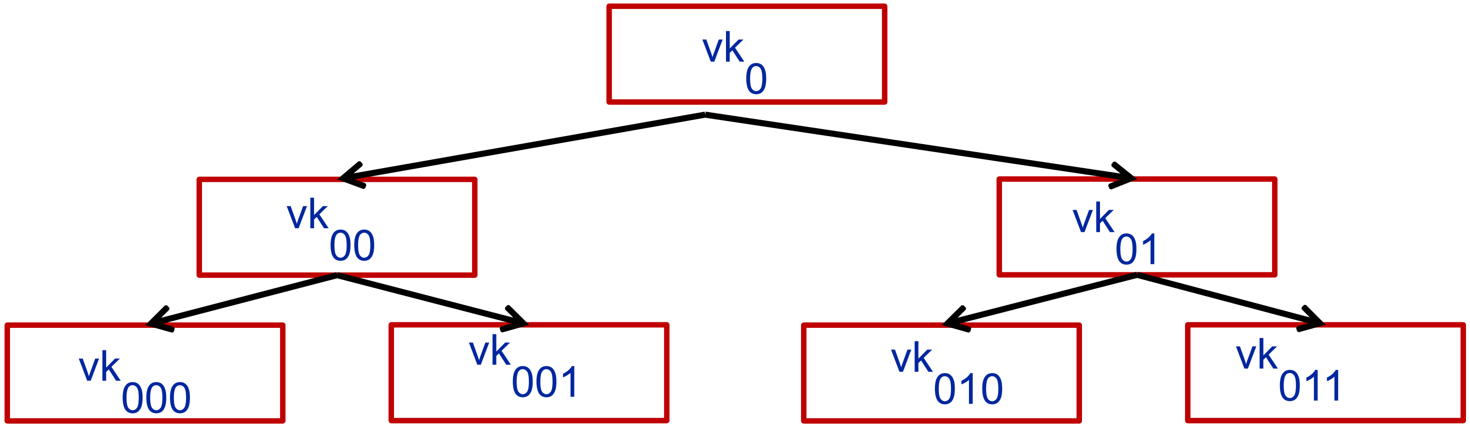
A Stateful Scheme

- Let Gen, Sign, Vrfy be a one-time signature scheme for signing "sufficiently long" messages , say size n
- The signer's state is binary tree with $2n$ leaves
- Each node w has a left child and a right child
- The tree is of exponential size but is never fully constructed



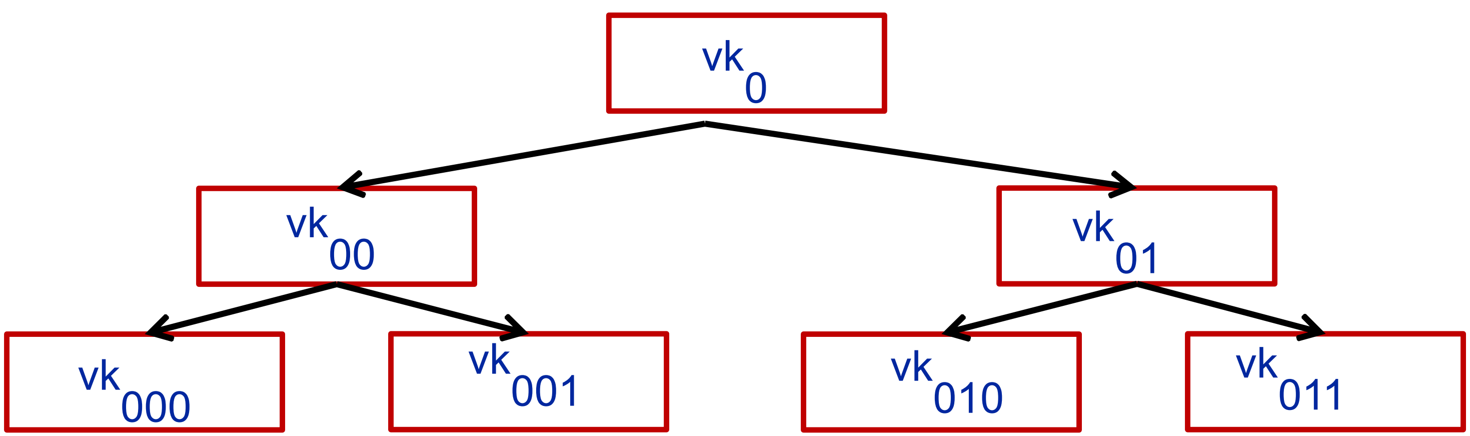
A Stateful Scheme

- Let Gen, Sign, Vrfy be a one-time signature scheme for signing "sufficiently long" messages ,
- The signer's state is binary tree with $2n$ leaves
- Each node w has a left child and a right child
- vk's are generated only if not previously generated
- Signature of ith message consists of path of vk's and their signatures + signature of ith message



A Stateful Scheme

- Let Gen, Sign, Vrfy be a one-time signature scheme for signing "sufficiently long" messages , say size n
- The signer's state is binary tree with $2n$ leaves
- Each node w has a left child and a right child
- vk's are generated only if not previously generated
- Signature of ith message consists of path of vk's and their signatures + signature of ith message
- Verify entire path upto vk_0 and check that its in the public key



Logarithmically Growing!

Now the state, the signature size, and the work for signing and verifying messages depend logarithmically on the number of signatures

Can we eliminate the state alltogether?

- This would make the scheme simpler to run, will allow distributed signing,
- Will make each signature independent of the activity in the rest of the system.

Problem 2: Randomized and Stateless?

- Idea: instead of remembering past choices we'll use a PRF to make the same choices again and again whenever presented with the same message prefix.
- Use pseudo-random functions for choosing new keys to sign m_i, i.e. $f(m_i)$ = randomness to choose (vk_i, sk_i)
- Signer uses m's value to find its place in the tree, rather than store i
- Signer re-computes path as necessary

Putting it together: details

- The signing key will have also a key k for a PRF F.
- To sign message m, use randomness $r=F\_k(m)$ and re-do the tree from scratch

- Correctness: clear.
- Unforgeability: Assume for contradiction

that the new scheme is forgeable, and construct a distinguisher between prf F and a random function.

Summary of Digital Signature Paradigms

- Diffie Hellman Trapdoor paradigms (insecure against CMA attack)
- Hash and Sign (oracle based)
- One Time Signature to Many via chain/tree based signatures (secure under OWF against CMA but inefficient)

- Remaining Goal: "Efficient" (signatures size don't grow with history) and EU-ACMA

Cramer-Shoup Digital Signature Scheme

Strong RSA problem:

Given n and y $\in Z_n^*$ find any x and e such that $y = x^e \bmod n$.

Strong RSA assumption:

$\forall$ PPT algorithms A,

$\mathrm{Prob}(A(n,y) = (x,e) \text{ s.t. } y=x^e \bmod n) < \mathrm{neg}(k)$

(taken over n=pq and $x \in Z_n^*$)

Note: Possibly easier than the classical RSA question, as e is not fixed in advance.

Cramer Shoup Digital Signatures

- **Key Generation:** Let vk=(N, x, h, e ,H) and sk={p,q}, where $N=pq$, $x,h \in Z_n^*$, $gcd(e,\phi(N))=1$, H collision resistant hash function

- **Sign ({p,q}, m):**
  - Choose random r in Zn*.
  - Let $(y')^e = x\, h^{H(r)} \bmod N$. Compute y'.
  - Let $y^{e'} = r\, h^{H(m)} \bmod N$. Compute y and e'.
  - Output signature $\sigma = (y,y',e')$
- **Verify((N, x,h, e' ,H), m, σ):**
  - Let $\sigma = (y,y',e')$
  - Check that $(y')^e = x\, h^{H(r)} \bmod N$.
  - Check that $y^{e'} = r\, h^{H(m)} \bmod N$
  - If all checks succeed accept, else reject

Security of Cramer-Shoup Signatures

Theorem: Under Strong-RSA Assumption, the Cramer-Shoup digital signature method is existentially unforgeable under chosen message attack.

Efficiency Improvements

- Incremental Signature Schemes: Signatures which can be quickly updated, with update work proportional to the amount of modifications document underwent since last time signed.
- On Line/Off Line: Major efficiency can be gained if one is careful to do whatever computation is possible before knowing which message exactly will need to be signed
- Batch Signing/Batch Verification:

  it is possible to verify whether many signatures are valid in a more efficient way that verifying the validity of each one individually .

Incremental Signatures

- Start with
  - (G,S,V) for fixed size B messages which produce signature of size k
  - a collision resistant hash $H:\{0,1\}^{2k} \rightarrow \{0,1\}^k$
- For longer messages M=B1…Bn
  - A signature is the contents of a balanced search tree:
    - Leafs contain $\sigma_i = S(sk, B_i)$ for message blocks
    - Internal nodes, parent to $\sigma 1, \sigma 2$, contains $S(sk, H(\sigma 1 | \sigma 2))$
  - To verify must verify signatures from root down to all leafs

Can Edit Incremental Signatures

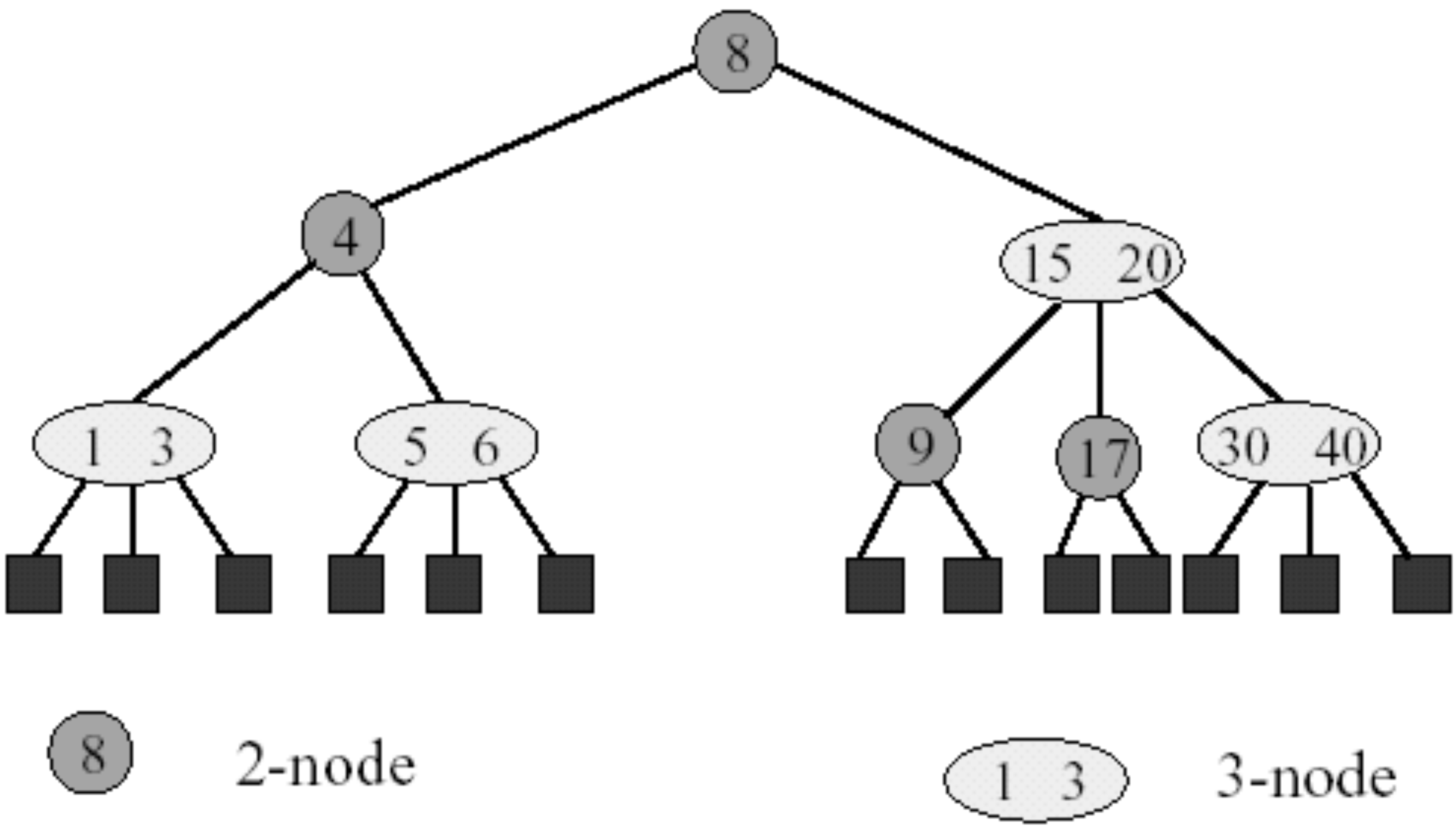
- Start with
  - (G,S,V) for fixed size B messages which produce signature of size k
  - a collision resistant hash $H:\{0,1\}^{2k} \to \{0,1\}^k$
- To modify the signature of M=B1…Bn

  by replacing block Bj by block Bj' :
  - go down the path to leaf where Bj is stored & store new block Bj',
  - updates signatures on internal nodes on path from modified leaf upward to root
  - cost of update: O(log n * (cost of single block signature +cost of evaluating H)

Incremental Signatures

- Can support cut and pastes, or whatever the balanced tree structure supports
- Structure of tree can reveal history of updates .. is this a problem?
- Yes, can fix and come up with a memoryless 2-3 tree (see web site).

Example 2-3 Tree



Variants on Digital Signatures

- Blind Signatures
- Group Signatures
- Undeniable Signatures

Blind Signatures

Introduced by Chaum, allow A to get a message m signed by Bob, without B knowing which m he signed

Why?

Ex1: Suppose Bob is notary public, Alice wants him to notarize a document. Bob does not need to know what document says,

only he notarized it at a certain time.

Ex2: Untraceable Checks (electronic cash)

Blind Signatures: How?

Blind Signatures Using RSA function

User B has RSA public Key (n,e) and secret key d

A chooses random r in Zn* r is a `blinder'

and asks B to sign $M=mr^e \bmod n$

B returns $y=M^d=m^d r \bmod n$

Now A sets the signature of m = y/r mod n

Using Blind Signatures: E-cash

Alice wants a virtual \$100 note.

- Alice goes to the bank and gets Banks signature on a \$100 note.
- Problem1: Bank can trace check back to Alice
- Solution: Bank signs check m via a blind signature.
- Problem2: Alice tricks the bank into signing a check for more than \$100
- Solution2:
  - Alice prepares 100 versions of check $m_1,\ldots,m_{100}$ and gives the Bank $y_i=r_i^e m_i$ mod n for randomly chosen r_i in Z_n^*
  - Bank challenges Alice to reveal all r_i's 1<i<100 except for one r.
  - If all checks revealed are ok, Bank signs the remaining un-opened one, and
  - Alice calculates $m^d=r^{-1}(r^e m)^d$ mod n.

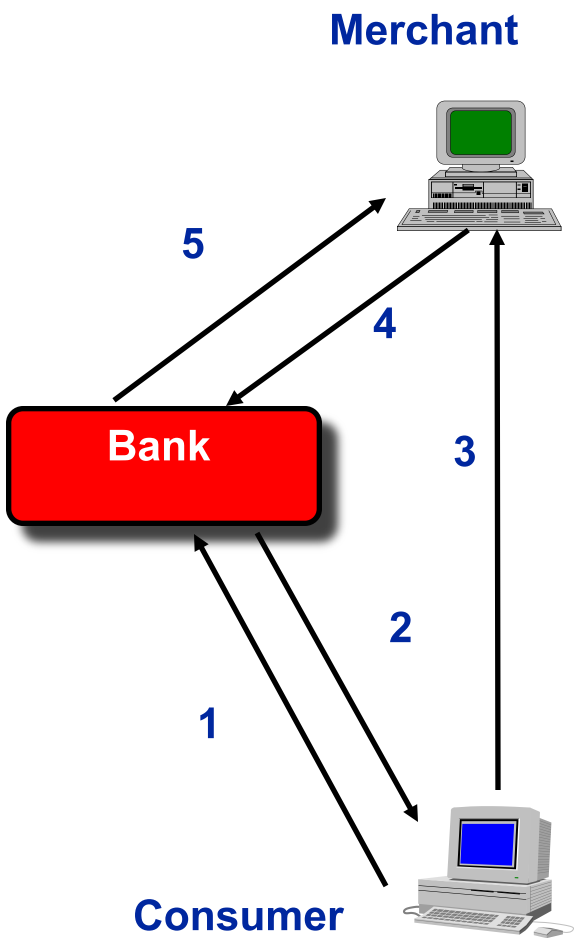
Security Concerns

- Can such a scheme be made secure against ACMA ?
- Not quite, but can induce a limit on the number of new signatures that can be created: schemes where cannot generate more valid (m,sig) pairs than given by Bank.

E-cash: Beyond Signatures

- How about Double Spending?
- E-cash scheme usually has 3 components: bank, merchant, and consumer
- There are protocols that are run between bank, merchant and consumer

E-cash Concept



1. Consumer buys e-cash from Bank
2. Bank sends e-cash to consumer
3. Consumer sends e-cash to merchant
4. Merchant checks with Bank that e-cash

   is not invalid
5. Bank verifies that e-cash has not been

   Used before
6. Parties complete transaction:

   e.g., merchant

   present e-cash to issuing back for deposit once goods or services are delivered

Consumer still has (invalid) e-cash

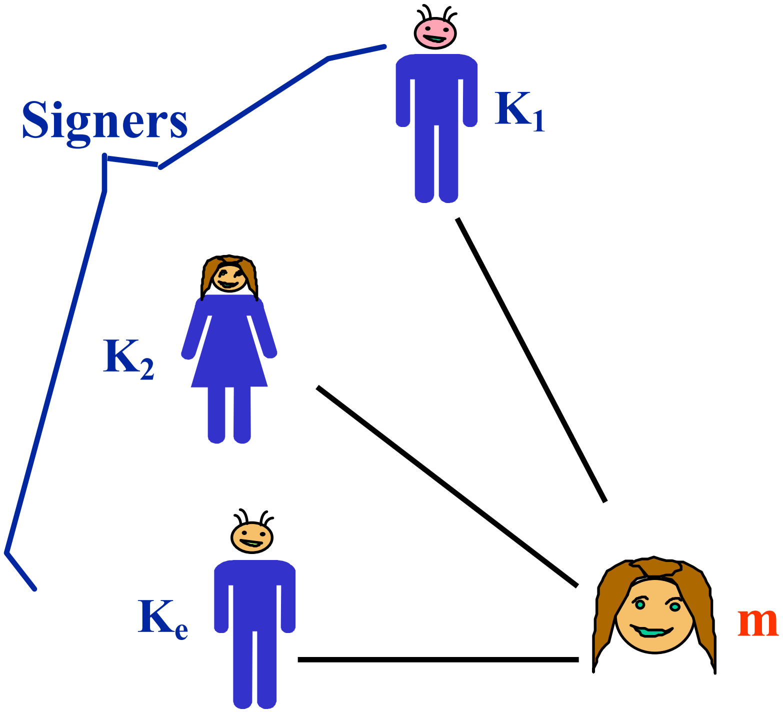
Group Signatures [D,DF]

An digital signature where:

- Secret key is shared among trustees,
- Trustees can produce valid signatures only if sufficient number cooperates
- Faulty trustees can't prevent signature

- Challenge: Size of public key and size of signatures should not be proportional to the number of group members

t-Threshold Signatures



Signature Scheme with n signers:

- where each signer has a share s_i of key s .
- < t signers cooperate can't sign
- >t honest signers can produce valid signatures

Signer$_i$ = Certification Authority

m = Alice's public-key

Will see how to do this once we learn about secret sharing

Undeniable Signatures

Undeniable signatures are a special form of signatures which require the cooperation of the signer in order to verify the validity of a signature. If the legal signer refuses to verify, he must be able prove that the signature is a fraud.

An undeniable signature consists of:

Key-Generation Algorithm,
Signing Algorithm,
interactive verification protocol,
disavowal protocol.

Usage for Undeniable Signatures

Ex1: Customer C wants to gain access to a secure area controlled by the bank B (e.g. deposit box).

- Solution: B requires a signature from C on a challenge document (with date and time) before access is granted.
- The use of undeniable signatures prevents B from using the signature as evidence that C was at the bank (since C must be present in verification).

Ex2: Software Pirating.

The vendor signs the software with an undeniable signature, which must be verified before the software can be installed on a new machine.

Signatures vs. Identification

- In many applications (e.g. password, access control etc) we only want to verify that the entity (e.g. person) claiming to be *A* is indeed *A*, rather than authenticating documents
- Given a signature scheme this identification problem is easily solved as follows

A' "I am *A*" →

← Challenge m

A's Signature of m →

If signature of m is valid, then *A*' is identified as *A*

- However, the identification problem may be easier than signing and may be solved with more efficient interactive solutions rather than requiring signatures.