

---

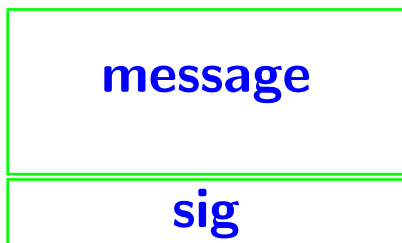
# **SIGNATURE SCHEMES & CRYPTOGRAPHIC HASH FUNCTIONS**

---

**CIS 400/628 — Spring 2005**  
**Introduction to Cryptography**

**This is based on Chapter 8 of Trappe and Washington**

# DIGITAL SIGNATURES



1. How do we **bind** a signature to a message?
2. How do we **verify** a signature is valid?
3. How do we **prevent reuse** of a signature?

## RSA SIGNATURES

### Setup (for each user)

$n = p \cdot q$ , where  $p$  and  $q$  are distinct, big primes

$e, d \in \mathbb{Z}_{\varphi(n)}^*$  such that  $e \cdot d \equiv 1 \pmod{\varphi(n)}$

$n$  and  $e$  public     $p, q,$  and  $d$  private

$\text{sig}_A(m) =_{\text{def}} m^{d_A} \pmod{n}$

$\text{ver}_A(m, y) =_{\text{def}} [m \stackrel{?}{=} y^{e_A} \pmod{n}]$

**Binding?** — the sig combines private key + message

**Verifying?** — calculate  $z = y^{e_A} \pmod{n}$ . Does  $z = m$ ?

**Making reuse hard?** — What is Eve's problem?

# BLIND SIGNATURES

Sealed bids, etc.

Bob wants Alice's signature on a message,  
but he does not want her to know the message

**Bob**

Chooses  $k \stackrel{\text{ran}}{\infty} \mathbb{Z}_n^*$ .

Computes  $t = (k^{e_A} \cdot m) \bmod n$

Sends  $t$  to Alice

**Alice**

Computes  $s = t^{d_A} \bmod n$

Sends  $s$  to Bob.

Note:  $s \equiv t^{d_A} \equiv (k^{e_A} \cdot m)^{d_A} \equiv k^{e_A \cdot d_A} \cdot m^{d_A}$   
 $\equiv k \cdot m^{d_A} \equiv k \cdot \text{sig}_A(m) \pmod{n}$

**Bob**

Computes  $s \cdot k^{-1} \equiv \text{sig}_A(m) \pmod{n}$

# ENCRYPTION AND SIGNATURES

## SCHEME 1

### Alice

Given  $x$

Computes  $y = \text{sig}_A(x)$

Computes  $z = \text{encrypt}_B(x||y)$

Sends  $z$  to Bob

### Bob

Computes  $(x||y) = \text{decrypt}_B(z)$

Checks  $\text{ver}_A(x, y)$

# ENCRYPTION AND SIGNATURES

## SCHEME 2

### Alice

Given  $x$

Computes  $z = \text{encrypt}_B(x)$  &  $y = \text{sig}_A(z)$

Sends  $(z, y)$  to Bob

### Bob

Checks  $\text{ver}_A(z, y)$  and computes  $x = \text{decrypt}_B(z)$

## BUT

### Eve

Obtains  $(z, y)$

Computes  $y' = \text{sig}_E(z)$

Sends Bob  $(z, y')$

### Bob

Checks  $\text{ver}_E(z, y')$  and computes  $x = \text{decrypt}_B(z)$

( $x = \text{"Send me \$1,000,000."}$ )

# THE ELGAMAL SIGNATURE SCHEME

## Setup

- ▶  $p$ , a prime with hard disc. log problem for  $\mathbb{Z}_p^*$
- ▶  $\alpha \in \mathbb{Z}_p^*$ , a prim. elm. of  $\mathbb{Z}_p^*$
- ▶ Plaintext:  $\mathbb{Z}_p^*$     Ciphertext:  $\mathbb{Z}_p^* \times \mathbb{Z}_{p-1}$
- ▶  $a \in \mathbb{Z}_{p-1}^*$     ▶  $\beta = \alpha^a \pmod{p}$
- ▶  $p, \alpha, \beta$  : public     $a$  : private

**Alice** (wants to sign  $m - (p, \alpha, \beta)$  Alice's public key)

Chooses  $k \stackrel{\text{ran}}{\in} \mathbb{Z}_{p-1}^*$

Computes  $r = \alpha^k \pmod{p}$

Computes  $s = k^{-1}(m - a \cdot r) \pmod{p-1}$

Sends  $(m, r, s)$  to Bob.

**Bob**

Computes  $v_1 = \beta^r r^s \pmod{p}$  &  $v_2 = \alpha^m \pmod{p}$

Checks that  $[v_1 \equiv v_2 \pmod{p}]$  (check on board)

# SECURITY REQS FOR SIGNATURE SCHEMES

## Key only attack

Eve has Alice's public key — only!

## Known message attack

Eve has  $(m_1, s_1), \dots, (m_k, s_k)$  messages signed by Alice

## Chosen message attack

Eve chooses  $m_1, \dots, m_k$  and has Alice sign all of them  $s_i = \text{sig}_A(m_i)$ ,  
 $i = 1, \dots, k$

## Eve's possible goals

### Total break

Find Alice's private key

### Selective forgery

Given  $m$ , Eve creates a valid signature of Alice for  $m$  — **with reasonable probability**

### Existential forgery

Eve is able to create a valid  $\text{sig}_A$  for at least one message (**probably junk**)

# EXAMPLES OF SECURITY PROBLEMS

Using the RSA signature scheme

∃ forgery/key only

Given  $s$ , compute  $x = e_A(s)$ .

∃ forgery/known messages

Alice sends  $(m_1, s_1), (m_2, s_2)$

Eve computes  $(m_1 \cdot m_2 \bmod n, s_1 \cdot s_2 \bmod n)$

So what?

Recall  $\text{sig}_A(m) = d_A(m) = m^{d_A} \bmod n$ .

**CLAIM**  $(e_U(x_1) \cdot e_U(x_2)) \bmod n = e_U(x_1 \cdot x_2) \bmod n$

**CLAIM**  $(d_U(x_1) \cdot d_U(x_2)) \bmod n = d_U(x_1 \cdot x_2) \bmod n$

proof on board

**COR**  $(s_1 \cdot s_2) \bmod n$  is a valid sig. for  $(m_1 \cdot m_2) \bmod n$ .

## MORE FORGERY EXAMPLES

Eve wants to forge Alice's signature on  $m$ .

- ▶ Eve finds  $x_1$  and  $x_2$  such that  $m = (x_1 \cdot x_2) \bmod n$ .
- ▶ Eve asks Alice to sign  $x_1$  and  $x_2$ .
- ▶ Alice produces  $y_1 = \text{sig}_A(x_1)$  and  $y_2 = \text{sig}_A(x_2)$ .
- ▶ Then  $(y_1 \cdot y_2) \bmod n$  is a valid signature for  $m$ !

How can we guard against stuff like this?

# CRYPTOGRAPHIC HASH FUNCTIONS

$h: \text{strings} \rightarrow \{0, 1\}^k$       where  $k = 160$  or so

**We want:**

- ▶  $h$  is fast to compute.
- ▶  $h$  is one-way, i.e.,  $h(y) \mapsto y$  is hard.
- ▶  $h$  is **strongly collision-free** (a silly name)  
i.e., it is hard to find  $m_1$  and  $m_2 \ni$   
 $m_1 \neq m_2$  and  $h(m_1) = h(m_2)$ .

Why are these things cool?

**Alice**

message:  $m$

message digest:  $z = h(m)$        $h$  is public

signature:  $y = \text{sig}_A(z)$

**Bob**

receives:  $(m, y)$

computes:  $z' = h(m)$

checks  $\text{ver}_A(z', y)$

Now we can sign arbitrary size messages

# HASHING, CONTINUED

**A bad hash function:**  $h(m) = m \pmod{n}$

- ▶ fast to compute
- ▶ not 1-way (Why?)
- ▶ not strongly collision-free (Why?)

**A better hash function:** Chaum, van Heijst, & Pfitzmann (CvHP)

## Setup

Find a prime  $p$  such that  $q = \frac{p-1}{2}$  is also prime.

Find two prim. elms of  $\mathbb{Z}_p^*$ ,  $\alpha$  and  $\beta \ni$

$$\beta = \alpha^a \pmod{p}, \text{ for some } a.$$

## Function

$m \in \mathbb{Z}_{q^2}$  or equiv.  $m = x_0 + x_1 \cdot q$  where  $x_0, x_1 \in \mathbb{Z}_{q-1}$

$$h(m) \stackrel{\text{def}}{=} \alpha^{x_0} \beta^{x_1} \pmod{p}$$

**THEOREM** Given  $m, m'$  with  $m \neq m'$  &  $h(m) = h(m')$ ,  
we can compute  $a = L_\alpha(\beta)$ .

## CVHP, CONTINUED

**Setup:** ▶  $p, q = \frac{p-1}{2}$  prime. ▶  $\alpha, \beta = \alpha^a \pmod{p}$  prim. elms of  $\mathbb{Z}_p^*$

**Function:** ▶  $m = x_0 + x_1 \cdot q, x_0, x_1 \in \mathbb{Z}_q$  ▶  $h(m) = \alpha^{x_0} \beta^{x_1} \pmod{p}$ .

**Proposition:** From  $m \neq m'$  with  $h(m) = h(m')$  we can compute  $a = L_\alpha(\beta)$

**Proof** Suppose

$$m = x_0 + x_1 \cdot q \ \& \ m' = x'_0 + x'_1 \cdot q \ \& \ m \neq m' \ \& \\ \alpha^{x_0} \beta^{x_1} = \alpha^{x'_0} \beta^{x'_1} \pmod{p}.$$

Since  $\beta = \alpha^a \pmod{p}$  we have  $\alpha^{a(x_1 - x'_1) - (x'_0 - x_0)} \equiv 1 \pmod{p}$ .

$$\therefore a(x_1 - x'_1) \equiv (x'_0 - x_0) \pmod{p-1}. \quad (\star) \quad (\text{Why?})$$

Let  $d \stackrel{\text{def}}{=} \gcd(x_1 - x'_1, p-1)$ .

Since  $p-1 = 2 \cdot q$ ,  $d \in \{1, 2, q, p-1\}$ .

Since  $0 \leq x_1, x'_1 < q$ ,  $|x_1 - x'_1| < q$ .

Since  $m \neq m'$ , we have  $x_1 \neq x'_1$  (Why?) and so  $d \in \{1, 2\}$ .

$\therefore$  There are at most two poss. for  $a$  in  $(\star)$  and they are easy to find.

Try them both. One yields  $\alpha^a = \beta$ .  $\therefore$  You have found  $a$  **QED**

## COLLISION-FREE $\implies$ 1-WAY

**Setup:**  $\blacktriangleright p, q = \frac{p-1}{2}$  prime.  $\blacktriangleright \alpha, \beta = \alpha^a \pmod{p}$  prim. elms of  $\mathbb{Z}_p^*$   
**Function:**  $\blacktriangleright m = x_0 + x_1 \cdot q, x_0, x_1 \in \mathbb{Z}_q$   $\blacktriangleright h(m) = \alpha^{x_0} \beta^{x_1} \pmod{p}$ .  
**Proposition:** From  $m \neq m'$  with  $h(m) = h(m')$  we can compute  $a = L_\alpha(\beta)$

$\therefore$  If discrete log is hard, then  $h$  is strongly coll.-free. **But what about 1-way?**

Suppose  $h$  is fast to invert.

I.e., there is a fast to compute  $g$  such that

given a message digest  $y, g(y) = m_y \ni h(m_y) = y$ .

Repeat forever

Choose  $m \stackrel{\text{ran}}{\in} \mathbb{Z}_{q^2}^*$

Compute  $y = h(m)$

Compute  $m_y = g(y)$

If  $m \neq m_y$ , then return  $(m, m_y)$

- $\blacktriangleright$  The search prob. halts after not many iterations.
- $\blacktriangleright$  So, by the Proposition, we can quickly find disc. logs.
- $\blacktriangleright$  So,  $g$  is unlikely to exist.
- $\blacktriangleright$  So,  $h$  is likely to be 1-way.

**QED**

# HASHING & SIGNING

$$m \xrightarrow{h} h(m) \xrightarrow{\text{sig}_A} \text{sig}_A(h(m))$$

$m$
$\text{sig}_A(h(m))$

## ALICE

Computer  $y = \text{sig}_A(h(m))$ .

Sends  $(m, y)$  to Bob.

## BOB

Receives  $(m, y)$

Checks  $\text{ver}_A(h(m), y)$ .

## EVE

Given  $(m, \text{sig}_A(h(m)))$ ,

Eve wants to forge Alice's signature on  $m'$ .

$\therefore$  Eve needs  $y' = \text{sig}_A(h(m')) = \text{sig}_A(h(m))$ .

**But** if  $h$  is strongly collision-free, Eve is out of luck.

# BIRTHDAY ATTACKS

## Probability Puzzle

Given 23 people,  $\text{Prob}[\text{two have the same b-day}] \geq 0.5$

Given 30 people,  $\text{Prob}[\text{two have the same b-day}] \geq 0.7$

Given 40 people,  $\text{Prob}[\text{two have the same b-day}] \geq 0.89$

## Prob the 23 people all have different birthdays

$$1 \cdot \left(1 - \frac{1}{365}\right) \cdot \left(1 - \frac{2}{365}\right) \cdots \left(1 - \frac{22}{365}\right) \approx .493.$$

$$\therefore \text{Prob. two have the same birthday} \approx 1 - .493 = .507$$

## What does this have to do with hashing?

$h$ : People  $\rightarrow$  Days of the year

$h(x)$  =  $x$ 's birthday

collision  $\equiv$  finding two people with the same birthday

## BIRTHDAY ATTACKS, CONTINUED

### A generalization

We have

$n$  objects

two groups of  $r$  people each

each person selects an object

Picture on board

**Q:** What is the prob. that some object is selected by a person from each group?

**A:** If  $\lambda = r^2/n$ , then

probability of one match  $\approx 1 - e^{-\lambda}$

probability of  $k$  matches  $\approx \lambda^k e^{-\lambda}/k!$

# BIRTHDAY ATTACKS, CONTINUED

**EVE:**

has a message  $m$  she knows Alice will sign &

has a message  $m'$  she knows Alice would **never** sign

For  $i = 1, \dots, 2^{30}$ , ( $2^{30} = 1,073,741,824$ )

$m_i$  is a slight mod of  $m$  &

$m'_i$  is a slight mod of  $m'$

Eve looks for a collision:  $h(m_i) = h(m'_j)$ .

**If** she finds one, **then:**

she has Alice sign  $m_i$  (with  $\text{sig}_A(h(m_i))$ ) &

she constructs  $(m'_j, \text{sig}_A(\text{sig}_A(h_{m_i}))) = (m'_j, \text{sig}_A(\text{sig}_A(h_{m'_j})))$  &

Alice is in trouble.

**Q: How likely is such a collision?**

$n = 2^{50} / r = 2^{30} / \lambda = 2^{10} / 1 - e^{-1024} \approx 1 = \text{certainty!}$

**Fix 1:** 50 is too small in  $h$ :messages  $\rightarrow \mathbb{Z}_2^{50}$ . Pick a bigger number.

**Fix 2:** Alice makes the last move by randomly changing  $m$  before signing

# THE DIGITAL SIGNATURE ALGORITHM

$$h:\text{messages} \rightarrow \mathbb{Z}_2^{160}$$

## SETUP

### Alice

Finds a prime  $q \ni$

$q$  160 bits & there is another prime  $p$  with  $q|(p-1)$ .

Finds  $g$  a prim. elm. of  $\mathbb{Z}_p^*$ .

Lets  $\alpha \stackrel{\text{def}}{=} g^{(p-1)/q} \bmod p$ . (So  $\alpha^q \equiv 1 \pmod{p}$ .)

Chooses  $a \in \mathbb{Z}_q^*$  and lets  $\beta \stackrel{\text{def}}{=} \alpha^a \bmod p$ .

**Public:**  $p, q, \alpha, \beta$     **Private:**  $a$

## DSA, CONTINUED

- ▶  $p$  and  $q$  primes with  $q | (p - 1)$ .
- ▶  $g$  prim. elm. of  $\mathbb{Z}_p^*$ .
- ▶  $\alpha = g^{(p-1)/q} \pmod{p}$ .
- ▶  $\beta = \alpha^a \pmod{p}$  with  $a \in \mathbb{Z}_q^*$

### Alice wants to sign $m$

Chooses  $k \stackrel{\text{ran}}{\in} \mathbb{Z}_q^*$ . **Private**

Computes  $r = (\alpha^k \pmod{p}) \pmod{q}$  &  $s = (k^{-1}(m + ar)) \pmod{q}$ .

Sends  $(m, r, s)$ .

### Bob wants to verify $(m, r, s)$

Computes

$$u_1 = s^{-1}m \pmod{q}$$

$$u_2 = s^{-1}r \pmod{q}$$

$$v = (\alpha^{u_1} \beta^{u_2} \pmod{p}) \pmod{q}$$

Checks  $v \stackrel{?}{=} r$

Check on board

This is a bit faster than ElGamal.

ElGamal: 3 mod-exps.

DSA: 2 mod-exps.

# MESSAGE AUTHENTICATION CODES (MAC)

A MAC is a key-dependent one-way hash.

Say what?

## one-way hash

We know what this is: think SHA or MD5.

- ▶ fixed size output
- ▶ collision-resistant
- ▶ preimage resistant

## key-dependent

- ▶ uses a secret key
- ▶ only someone who shares the key can verify

Very much like a digital signature for a secret-key system.

# A PRACTICAL PROBLEM (IN-CLASS EXERCISE)

## The Players

- ▶ Alice, Bob, and Charlie, three friends who wish to communicate by radio (a broadcast, limited bandwidth medium).
- ▶ Eve, who wishes to hear what Alice, Bob, and Charlie are saying.
- ▶ Spencer, who wishes to spoof identities.

## Scenario 1

Alice, Bob, and Charlie all want to talk to each other.

Discuss **options** for each of three possibilities:

- ▶ They don't mind if Eve hears them, but they don't want Spencer to spoof anyone.
- ▶ They don't want Eve to hear them, but they're not worried about Spencer impersonating anyone.
- ▶ They want to foil both Spencer and Eve.

## A PRACTICAL PROBLEM II

### Scenario 2

Alice and Charlie have a fight.

Bob still wants to talk to both of them.

Same concerns with Spencer and Eve.

**What changes from the last solution?**

## A PRACTICAL PROBLEM II

### Scenario 2

Alice and Charlie have a fight.

Bob still wants to talk to both of them.

Same concerns with Spencer and Eve.

**What changes from the last solution?**

### Scenario 3

Dennis, Fern, Goliath, Herb, ..., and Zeke want to talk too.

- ▶ Everyone knows and trusts Bob.
- ▶ Only Bob has a full-time net connection; the others can only connect to the net once a day.
- ▶ No one knows or trusts everyone else.

Discuss the impact of this on secret-key and public-key solutions.