CIS 6930/4930 Computer and Network Security

Topic 4. Cryptographic Hash Functions

The SHA-1 Hash Function

Secure Hash Algorithm (SHA)

- Developed by NIST, specified in the Secure Hash Standard, 1993
- SHA is specified as the hash algorithm in the Digital Signature Standard (DSS)
- SHA-1: revised (1995) version of SHA

SHA-1 Parameters

- Input message must be < 2^{64} bits
- Input message is processed in 512-bit blocks, with the same padding as MD5
- Message digest output is 160 bits long
  - Referred to as five 32-bit words **A, B, C, D, E**
  - IV: **A** = 0x67452301, **B** = 0xEFCDAB89, **C** = 0x98BADCFE, **D** = 0x10325476, **E** = 0xC3D2E1F0
- Footnote: bytes of words are stored in big-endian order

Preprocessing of a Block

- Let 512-bit block be denoted as sixteen 32-bit words $\mathbf{W_0..W_{15}}$
- Preprocess $\mathbf{W_0..W_{15}}$ to derive an additional sixty-four 32-bit words $\mathbf{W_{16}..W_{79}}$, as follows:

for $16 \leq t \leq 79$

$$\mathbf{W}_t = (\mathbf{W}_{t\text{-}16} \oplus \mathbf{W}_{t\text{-}14} \oplus \mathbf{W}_{t\text{-}8} \oplus \mathbf{W}_{t\text{-}3}) << 1$$

Block Processing

- Consists of 80 steps! (vs. 64 for MD5)
- Inputs for each step $0 \leq t \leq 79$:
  - $\mathbf{W}_t$
  - K_t – a constant
  - **A**,**B**,**C**,**D**,**E**: current values to this point
- Outputs for each step:
  - **A**,**B**,**C**,**D**,**E** : new values
- Output of last step is added to input of first step to produce 160-bit Message Digest

Constants K_t

- Only 4 values (represented in 32 bits), derived from $2^{30} * i^{1/2}$, for $i = 2, 3, 5, 10$
  - for $0 \le t \le 19$: K_t = 0x5A827999
  - for $20 \le t \le 39$: K_t = 0x6ED9EBA1
  - for $40 \le t \le 59$: K_t = 0x8F1BBCDC
  - for $60 \le t \le 79$: K_t = 0xCA62C1D6

Function f(t,B,C,D)

- 3 different functions are used in SHA-1 processing

Round	Function f(t,B,C,D)	Compare with MD-5
$0 \le t \le 19$	$(B \wedge C) \vee (\sim B \wedge D)$	$\mathcal{F} = (x \wedge y) \vee (\sim x \wedge z)$
$20 \le t \le 39$	$B \oplus C \oplus D$	$\mathcal{H} = x \oplus y \oplus z$
$40 \le t \le 59$	$(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$	
$60 \le t \le 79$	$B \oplus C \oplus D$	$\mathcal{H} = x \oplus y \oplus z$

- No use of MD5's $\mathcal{G}$ $((x \wedge z) \vee (y \wedge \sim z))$ or $\mathcal{I}$ $(y \oplus (x \vee \sim z))$

Processing Per Step

- Everything to right of “=” is input value to this step

```
for t = 0 upto 79
    A = E + (A << 5) + Wt + Kt + f(t,B,C,D)
    B = A
    C = B << 30
    D = C
    E = D
endfor
```

Comparison: SHA-1 vs. MD5

- SHA-1 is a stronger algorithm
  - brute-force attacks require on the order of 2^{80} operations vs. 2^{64} for MD5
- SHA-1 is about twice as expensive to compute
- Both MD-5 and SHA-1 are much faster to compute than DES

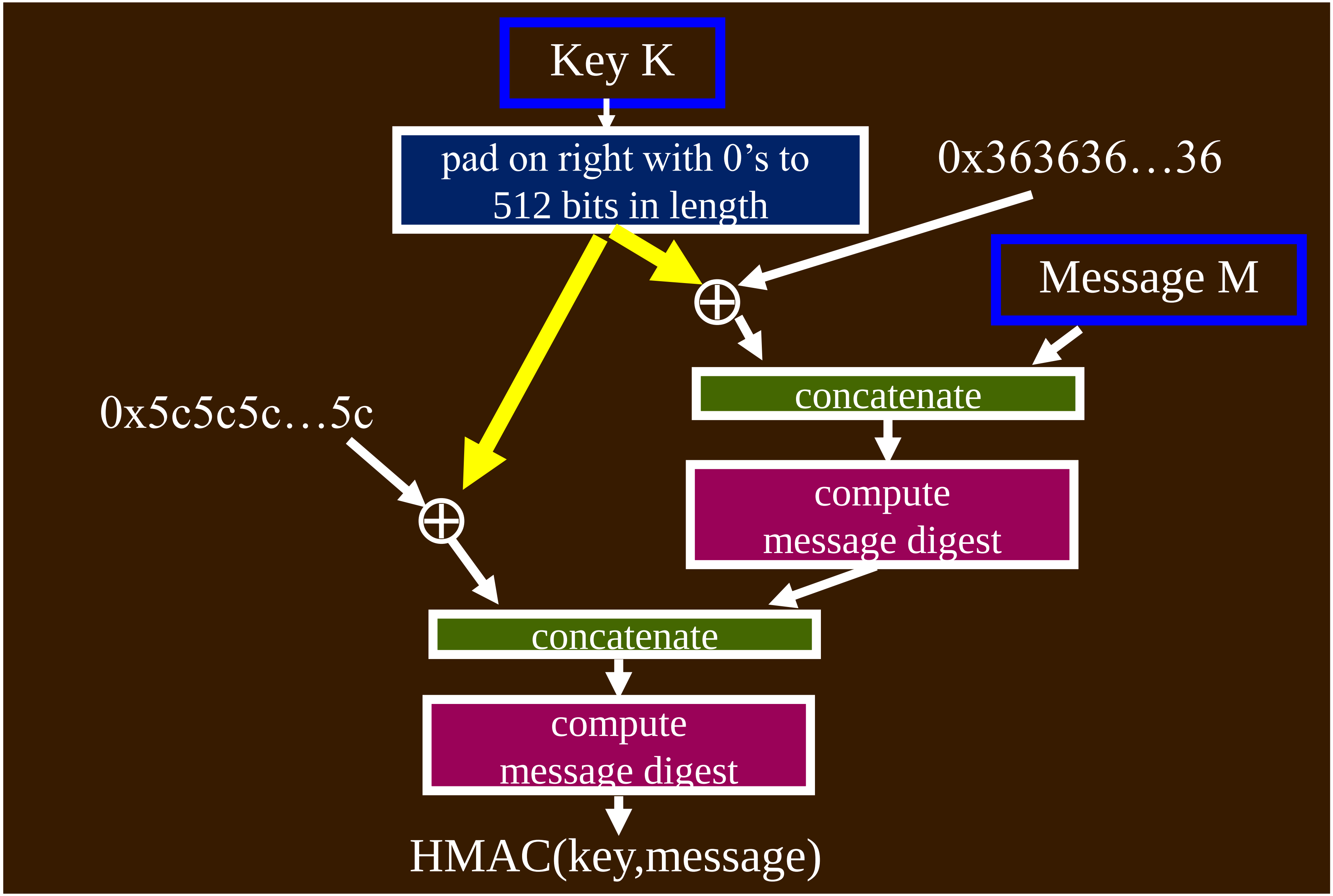
Security of SHA-1

- SHA-1
  - "Broken", but not yet cracked
  - Collisions in 2^{69} hash operations, much less than the brute-force attack of 2^{80} operations
  - Results were circulated in February 2005, and published in CRYPTO '05 in August 2005

The Hashed Message Authentication Code (HMAC)

- HMAC generates the message digest of both a message and a key
- Essence: digest-inside-a-digest, with the secret used at both levels
- The particular hash function used determines the length of HMAC output

HMAC Processing



Summary

- Hashing is fast to compute
- Has many applications (some making use of a secret key)
- Hash images must be at least 128 bits long
  - but longer is better
- Hash function details are tedious ☹
- HMAC generates the message digest of both a message and a key

CIS 6930/4930 Computer and Network Security

Topic 5.1 Basic Number Theory -- Foundation of Public Key Cryptography

GCD and Euclid's Algorithm

Some Review: Divisors

- Set of all integers is $\mathcal{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$
- *b divides a* (or b is a *divisor* of a) if $a = mb$ for some m
  - denoted $b|a$
  - any $b \neq 0$ divides 0
- For any a, 1 and a are *trivial divisors* of a
  - all other divisors of a are called *factors* of a

Primes and Factors

- a is *prime* if it has no non-trivial factors
  - examples: 2, 3, 5, 7, 11, 13, 17, 19, 31,…
- Theorem: there are infinitely many primes
- Any integer $a > 1$ can be factored in a unique way as $p_1^{a_1} \bullet p_2^{a_2} \bullet \ldots p_t^{a_t}$
  - where all $p_1 > p_2 > \ldots > p_t$ are prime numbers and where each $a_i > 0$

Examples:

$91 = 13^1 \times 7^1$

$11{,}011 = 13^1 \times 11^2 \times 7^1$

Common Divisors

- A number d that is a divisor of both a and b is a *common divisor* of a and b

Example: common divisors of 30 and 24 are 1, 2, 3, 6

- If $d|a$ and $d|b$, then $d|(a+b)$ and $d|(a-b)$

Example: Since 3 | 30 and 3 | 24 , 3 | (30+24) and 3 | (30-24)

- If $d|a$ and $d|b$, then $d|(ax+by)$ for any integers x and y

Example: 3 | 30 and 3 | 24 → 3 | (2*30 + 6*24)

Greatest Common Divisor (GCD)

- $gcd(a,b) = \max\{k \mid k|a \text{ and } k|b\}$

Example: gcd(60,24) = 12,   gcd(a,0) = a

- Observations
  - gcd(a,b) = gcd(|a|, |b|)
  - gcd(a,b) $\leq$ min(|a|, |b|)
  - if $0 \leq n$, then gcd(an, bn) = n*gcd(a,b)
- For all positive integers d, a, and b…
  …if $d \mid ab$
  …and gcd(a,d) = 1
  …then $d|b$

GCD (Cont'd)

- Computing GCD by hand:
  if $a = p_1^{a1}\ p_2^{a2} \ldots p_r^{ar}$ and
  $b = p_1^{b1}\ p_2^{b2} \ldots p_r^{br}$,
  …where $p1 < p2 < \ldots < pr$ are prime,
  …and ai and bi are nonnegative,
  …then $\gcd(a, b) =$

$$p_1^{\min(a1,\, b1)}\ p_2^{\min(a2,\, b2)} \ldots p_r^{\min(ar,\, br)}$$

$\Rightarrow$Slow way to find the GCD

- – requires factoring a and b first (which, as we will see, can be slow)

Euclid's Algorithm for GCD

- Insight:
  gcd(x, y) = gcd(y, x mod y)
- Procedure **`euclid(x, y)`**:

```
r[0] = x, r[1] = y, n = 1;
while (r[n] != 0) {
   n = n+1;
   r[n] = r[n-2] % r[n-1];
}
return r[n-1];
```

Example

n	r_n
0	595
1	408
2	595 mod 408 = 187
3	408 mod 187 = 34
4	187 mod 34 = 17
5	34 mod 17 = 0

gcd(595,408) = 17

Running Time

- Running time is logarithmic in size of *x* and *y*

```
Enter x and y: 102334155 63245986
Step   1: r[i] = 39088169
Step   2: r[i] = 24157817
Step   3: r[i] = 14930352
Step   4: r[i] =  9227465
…
Step  35: r[i] =        3
Step  36: r[i] =        2
Step  37: r[i] =        1
Step  38: r[i] =        0
gcd of 102334155 and 63245986 is        1
```

Extended Euclid's Algorithm

- Let $\mathcal{LC}(x,y) = \{ux+vy : x,y \in \mathcal{Z}\}$ be the set of linear combinations of x and y
- Theorem: if x and y are any integers > 0, then gcd(x,y) is the smallest positive element of $\mathcal{LC}(x,y)$
- Euclid's algorithm can be extended to compute u and v, as well as gcd(x,y)
- Procedure exteuclid(x, y):

  (next page…)

Extended Euclid's Algorithm

floor function

```
r[0] = x, r[1] = y, n = 1;
u[0] = 1, u[1] = 0;
v[0] = 0, v[1] = 1;
while (r[n] != 0) {
  n = n+1;
  r[n] = r[n-2] % r[n-1];
  q[n] = (int) (r[n-2] / r[n-1]);
  u[n] = u[n-2] - q[n]*u[n-1];
  v[n] = v[n-2] - q[n]*v[n-1];
}
return r[n-1], u[n-1], v[n-1];
```

Extended Euclid's Example

n	q_n	r_n	u_n	v_n
0	-	595	1	0
1	-	408	0	1
2	1	187	1	-1
3	2	34	-2	3
4	5	17	11	-16
5	2	0	-24	35

gcd(595,408) = 17 = 11*595 + -16*408

Relatively Prime

- Integers a and b are *relatively prime* iff $\gcd(a,b) = 1$
  - example: 8 and 15 are relatively prime
- Integers $n_1, n_2, \ldots n_k$ are pairwise relatively prime if $\gcd(n_i, n_j) = 1$ for all $i \neq j$