Practical Aspects of Modern Cryptography

Josh Benaloh & Brian LaMacchia

Public-Key History

- 1976 *New Directions in Cryptograhy*
  Whit Diffie and Marty Hellman
  - One-Way functions
  - Diffie-Hellman Key Exchange
- 1978 RSA paper
  Ron Rivest, Adi Shamir, and Len Adleman
  - RSA Encryption System
  - RSA Digital Signature Mechanism

The Fundamental Equation

$$Z=Y^X \bmod N$$

Diffie-Hellman

$$Z=Y^X \bmod N$$

When X is unknown, the problem is known as the *discrete logarithm* and is generally believed to be hard to solve.

Diffie-Hellman Key Exchange

Alice	Bob
Randomly select a large integer a and send $A = Y^a \bmod N$.	Randomly select a large integer b and send $B = Y^b \bmod N$.
Compute the key $K = B^a \bmod N$.	Compute the key $K = A^b \bmod N$.

$$B^a = Y^{ba} = Y^{ab} = A^b$$

Diffie-Hellman Key Exchange

What does Eve see?

Y, Y^a, Y^b

… but the exchanged key is Y^{ab}.

Belief: Given Y, Y^a, Y^b it is difficult to compute Y^{ab}.

Contrast with discrete logarithm assumption: Given Y, Y^x it is difficult to compute x.

One-Way Trap-Door Functions

$$Z=Y^X \bmod N$$

Recall that this equation is solvable for Y if the factorization of N is known, but is *believed* to be hard otherwise.

RSA Public-Key Cryptosystem

Alice

- Select two large random primes P & Q.
- Publish the product N=PQ.
- Use knowledge of P & Q to compute Y.

Anyone

- To send message Y to Alice, compute $Z=Y^X \bmod N$.
- Send Z and X to Alice.

Some RSA Details

When N=PQ is the product of distinct primes,

$$Y^X \bmod N = Y$$

whenever

$X \bmod (P-1)(Q-1) = 1$ and $0 \leq Y<N$.

Some RSA Details

When N=PQ is the product of distinct primes,

$$Y^X \bmod N = Y$$

whenever

$X \bmod (P-1)(Q-1) = 1$ and $0 \leq Y<N$.

Alice can easily select integers E and D such that $E \bullet D \bmod (P-1)(Q-1) = 1$.

Some RSA Details

Encryption: $E(Y) = Y^E \bmod N$.

Decryption: $D(Y) = Y^D \bmod N$.

$$\begin{aligned} &D(E(Y)) \\ &\quad = (Y^E \bmod N)^D \bmod N \\ &\quad = Y^{ED} \bmod N \\ &\quad = Y \end{aligned}$$

RSA Signatures

An additional property

$D(E(Y)) = Y^{ED} \bmod N = Y$

$E(D(Y)) = Y^{DE} \bmod N = Y$

Only Alice (knowing the factorization of N) knows D. Hence only Alice can compute $D(Y) = Y^D \bmod N$.

This D(Y) serves as Alice's signature on Y.

Remaining RSA Basics

- Why is Y^X mod PQ = Y whenever
  X mod (P-1)(Q-1) = 1, 0 ≤Y<PQ,
  and P and Q are distinct primes?

- How can Alice can select integers E and D such that E•D mod (P-1)(Q-1) = 1?

Modular Arithmetic

- To compute (A+B) mod N,
  compute (A+B) and take the result mod N.
- To compute (A-B) mod N,
  compute (A-B) and take the result mod N.
- To compute (A×B) mod N,
  compute (A×B) and take the result mod N.
- To compute (A÷B) mod N, …

Modular Division

What is the value of (1÷2) mod 7?
We need a solution to 2*x* mod 7 = 1.
Try *x* = 4.

What is the value of (7÷5) mod 11?
We need a solution to 5*x* mod 11 = 7.
Try *x* = 8.

Modular Division

Is modular division always well-defined?
(1÷3) mod 6 = ?
3*x* mod 6 = 1 has no solution!

<u>Fact</u>
(A÷B) mod N always has a solution when gcd(B,N) = 1.

Greatest Common Divisors

gcd(A , B) = gcd(B , A - B)
gcd(21,12) = gcd(12,9) = gcd(9,3)
= gcd(6,3) = gcd(3,3) = gcd(0,3) = 3
gcd(A , B) = gcd(B , A mod B)
gcd(21,12) = gcd(12,9) = gcd(9,3)
= gcd(0,3) = 3

Extended Euclidean Algorithm

Given integers A and B, find integers X and Y such that AX + BY = gcd(A,B).

When gcd(A,B) = 1, solve AX mod B = 1, by finding X and Y such that
AX + BY = gcd(A,B) = 1.

Compute (C÷A) mod B as C×(1÷A) mod B.

Extended Euclidean Algorithm

Given A,B > 0, set x_1=1, x_2=0, y_1=0, y_2=1, a_1=A, b_1=B, i=1.

Repeat while b_i>0: {$i = i + 1$;
$q = a_{i-1}$ div b_{i-1}; $b_i = a_{i-1} - qb_{i-1}$; $a_i = b_{i-1}$;
$x_{i+1} = x_{i-1} - qx_i$; $y_{i+1} = y_{i-1} - qy_i$}.

$Ax_i + By_i = a_i = \gcd(A,B)$.

Remaining RSA Basics

- Why is Y^X mod PQ = Y whenever
  X mod (P-1)(Q-1) = 1, 0 ≤Y<PQ,
  and P and Q are distinct primes?
- How can Alice can select integers E and D such that E•D mod (P-1)(Q-1) = 1?

Fermat's Little Theorem

If p is prime,
then $x^{p-1} \bmod p = 1$ for all $0 < x < p$.

Equivalently …

If p is prime,
then $x^p \bmod p = x \bmod p$ for all integers x.

Proof of Fermat's Little Theorem

The Binomial Theorem

$(x + y)^p = x^p + \binom{p}{1} x^{p-1}y + \ldots + \binom{p}{p-1} xy^{p-1} + y^p$

If p is prime, then $\binom{p}{i} \bmod p = 0$ for $0 < i < p$.

Thus, $(x + y)^p \bmod p = (x^p + y^p) \bmod p$.

Proof of Fermat's Little Theorem

By induction on x…

Basis

If $x = 0$, then $x^p \bmod p = 0 = x \bmod p$.
If $x = 1$, then $x^p \bmod p = 1 = x \bmod p$.

Proof of Fermat's Little Theorem

Inductive Step

Assume that $x^p \bmod p = x \bmod p$.
Then $(x + 1)^p \bmod p = (x^p + 1^p) \bmod p$
$= (x + 1) \bmod p$.
Hence, $x^p \bmod p = x \bmod p$ for integers $x \geq 0$.

Also true for negative x, since $(-x)^p = (-1)^p x^p$.

Proof of RSA

We have shown …

$Y^P \bmod P = Y$ whenever $0 \le Y < P$
and P is *prime*!

You will show …

$Y^{K(P-1)(Q-1)+1} \bmod PQ = Y$ when $0 \le Y < PQ$
P and Q are distinct primes and $K \ge 0$.

Authentication

How can I use RSA to *authenticate* someone's identity?

If Alice's public key E_A, just pick a random message *m* and send $E_A(m)$.

If *m* comes back, I must be talking to Alice.

Authentication

Should Alice be happy with this method of authentication?

Bob sends Alice the authentication string *y* = "I owe Bob $1,000,000 - signed Alice."

Alice dutifully authenticates herself by decrypting (putting her signature on) *y*.

Authentication

What if Alice only returns authentication queries when the decryption has a certain format?

RSA Cautions

Is it reasonable to sign/decrypt something given to you by someone else?

Note that RSA is multiplicative. Can this property be used/abused?

RSA Cautions

$$D(Y_1) \bullet D(Y_2) = D(Y_1 \bullet Y_2)$$

Thus, if I've decrypted (or signed) Y_1 and Y_2, I've also decrypted (or signed) $Y_1 \bullet Y_2$.

The Hastad Attack

Given

$E_1(x) = x^3 \bmod n_1$

$E_2(x) = x^3 \bmod n_2$

$E_3(x) = x^3 \bmod n_3$

one can easily compute x.

The Bleichenbacher Attack

PKCS#1 Message Format:

$$00\ 01\ \underbrace{XX\ XX\ ...\ XX}_{\text{random non-zero bytes}}\ 00\ \underbrace{YY\ YY\ ...\ YY}_{\text{message}}$$

“Man-in-the-Middle” Attacks

Alice ⟷ Bob

Alice ⟷ Eve ⟷ Bob

The Practical Side

- ♦ RSA can be used to encrypt any data.
- ♦ Public-key (asymmetric) cryptography is very inefficient when compared to traditional private-key (symmetric) cryptography.

The Practical Side

For efficiency, one generally uses RSA (or another public-key algorithm) to transmit a private (symmetric) key.

The private *session* key is used to encrypt and authenticate any subsequent data.

Digital signatures are only used to sign a *digest* of the message.

Symmetric Ciphers

Private-key (symmetric) ciphers are usually divided into two classes.

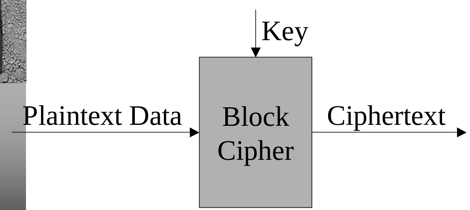
- ♦ Block ciphers
- ♦ Stream ciphers

Symmetric Ciphers

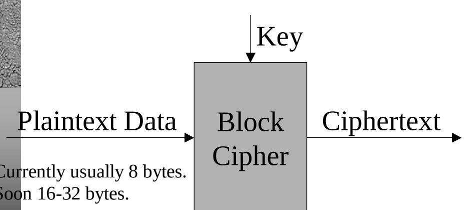
Private-key (symmetric) ciphers are usually divided into two classes.

- Block ciphers
- Stream ciphers

Block Ciphers



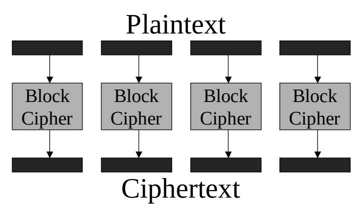
Block Ciphers



Currently usually 8 bytes.
Soon 16-32 bytes.

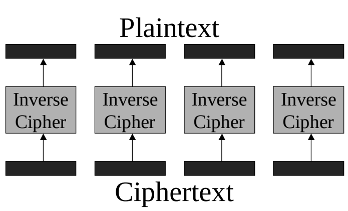
Block Cipher Modes

Electronic Code Book (ECB) Encryption:



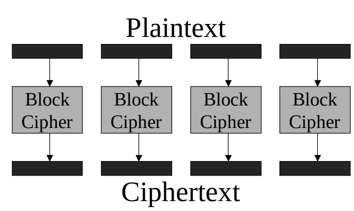
Block Cipher Modes

Electronic Code Book (ECB) Decryption:



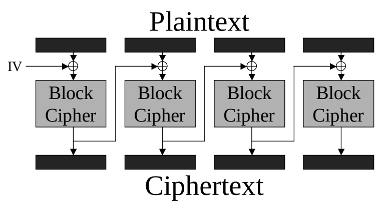
Block Cipher Modes

Electronic Code Book (ECB) Encryption:



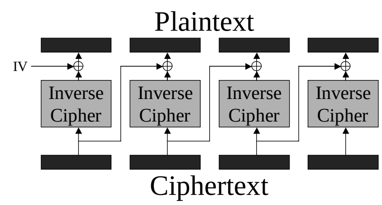
Block Cipher Modes

Cipher Block Chaining (CBC) Encryption:



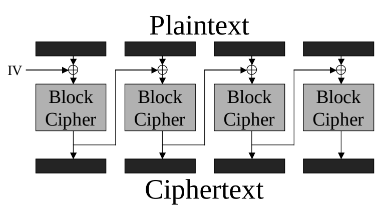
Block Cipher Modes

Cipher Block Chaining (CBC) Decryption:

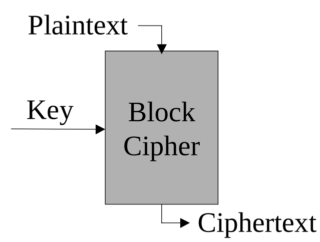


Block Cipher Modes

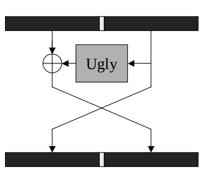
Cipher Block Chaining (CBC) Encryption:



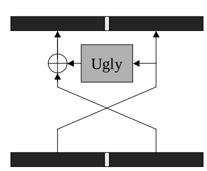
How to Build a Block Cipher



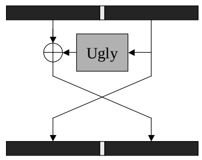
Feistel Ciphers



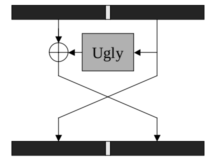
Feistel Ciphers



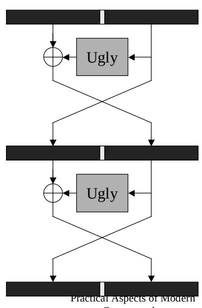
Feistel Ciphers



Feistel Ciphers



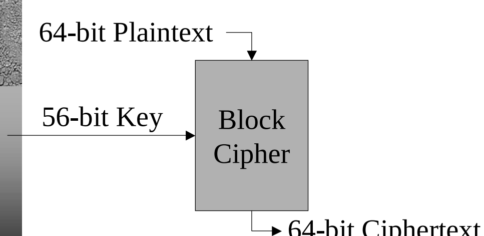
Feistel Ciphers



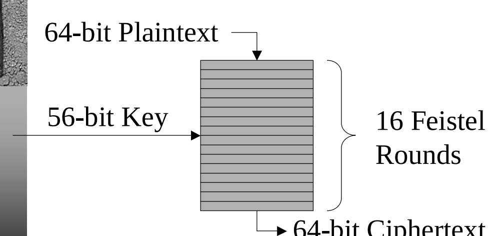
Feistel Ciphers

- Typically, most Feistel ciphers are iterated for about 16 rounds.
- Different “sub-keys” are used for each round.
- Even a weak round function can yield a strong Feistel cipher if iterated sufficiently.

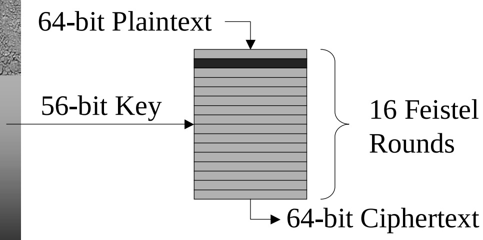
Data Encryption Standard (DES)



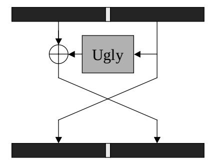
Data Encryption Standard (DES)



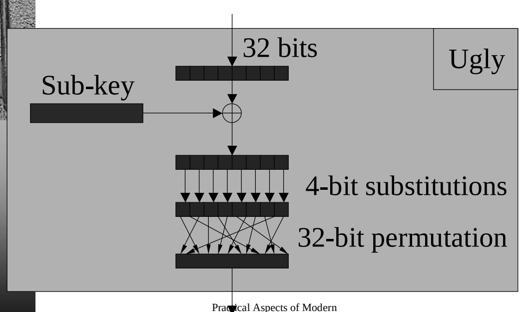
Data Encryption Standard (DES)



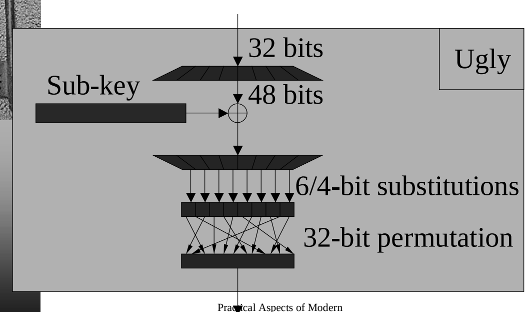
DES Round



Simplified DES Round Function



Actual DES Round Function



Symmetric Ciphers

Private-key (symmetric) ciphers are usually divided into two classes.

- Block ciphers
- Stream ciphers

Stream Ciphers

- Use the key as a seed to a pseudo-random number-generator.
- Take the stream of output bits from the PRNG and XOR it with the plaintext to form the ciphertext.

Stream Cipher Encryption

Plaintext: ■■■■■■■■■■■■■■■■
⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕
PRNG(seed): □□□□□□□□□□□□□□□□
↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓
Ciphertext: ■■■■■■■■■■■■■■■■

Stream Cipher Decryption

Plaintext: ■■■■■■■■■■■■■■■■
↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑
PRNG(seed): □□□□□□□□□□□□□□□□
⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕
Ciphertext: ■■■■■■■■■■■■■■■■

A PRNG: Alleged RC4

<u>Initialization</u>

```
S[0..255] = 0,1,…,255
K[0..255] = Key,Key,Key,…
for i = 0 to 255
    j = (j + S[i] + K[i]) mod 256
    swap S[i] and S[j]
```


A PRNG: Alleged RC4

<u>Iteration</u>

```
i = (i + 1) mod 256
j = (j + S[i]) mod 256
swap S[i] and S[j]
t = (S[i] + S[j]) mod 256
Output S[t]
```


Stream Cipher Integrity

- It is easy for an adversary (even one who can’t decrypt the ciphertext) to alter the plaintext in a known way.

Bob to Bob’s Bank:
Please transfer $0,000,002.00 to the account of my good friend Alice.

Stream Cipher Integrity

- It is easy for an adversary (even one who can’t decrypt the ciphertext) to alter the plaintext in a known way.

Bob to Bob’s Bank:
Please transfer $1,000,002.00 to the account of my good friend Alice.

Stream Cipher Integrity

- It is easy for an adversary (even one who can't decrypt the ciphertext) to alter the plaintext in a known way.

Bob to Bob's Bank:
Please transfer $1,000,002.00 to the account of my good friend Alice.

- This can be protected against by the careful addition of appropriate redundancy.

One-Way Hash Functions

The idea of a *check sum* is great, but it is designed to prevent accidental changes in a message.

For cryptographic integrity, we need an integrity check that is resilient against a smart and determined adversary.

One-Way Hash Functions

Generally, a *one-way hash function* is a function $H : \{0,1\}^* \rightarrow \{0,1\}^k$ (typically k is 128 or 160) such that given an input value x, one cannot find a value $x' \neq x$ such $H(x) = H(x')$.

One-Way Hash Functions

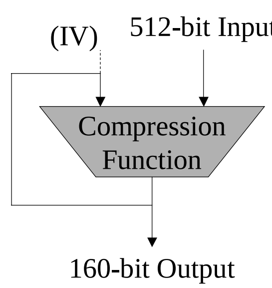
There are many measures for one-way hashes.

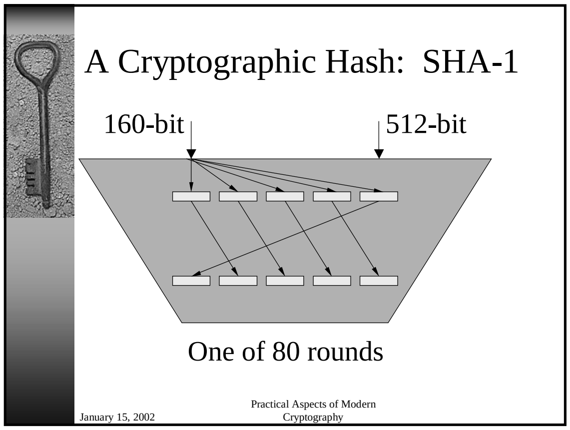
- Non-invertability: given y, it's difficult to find any x such that H(x) = y.
- Collision-intractability: one cannot find a pair of values $x' \neq x$ such that $H(x) = H(x')$.

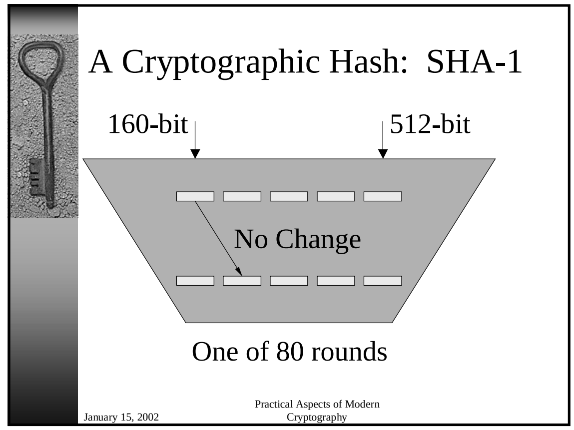
One-Way Hash Functions

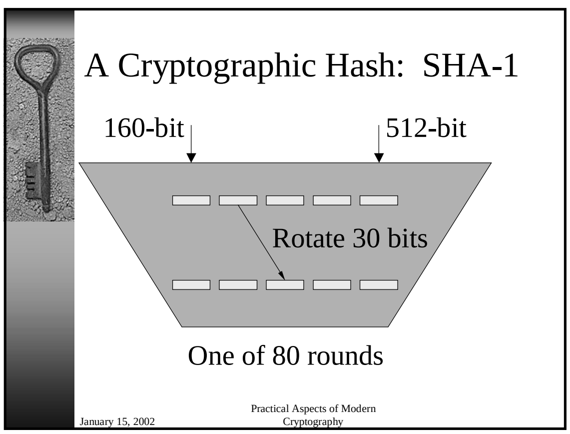
- When using a stream cipher, a hash of the message can be appended to ensure integrity. [Message Authentication Code]
- When forming a digital signature, the signature need only be applied to a hash of the message. [Message Digest]

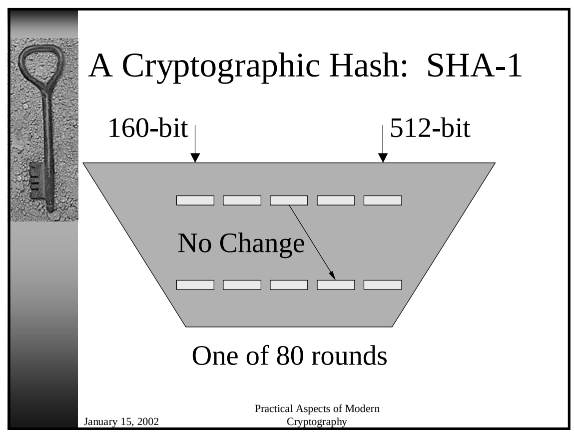
A Cryptographic Hash: SHA-1

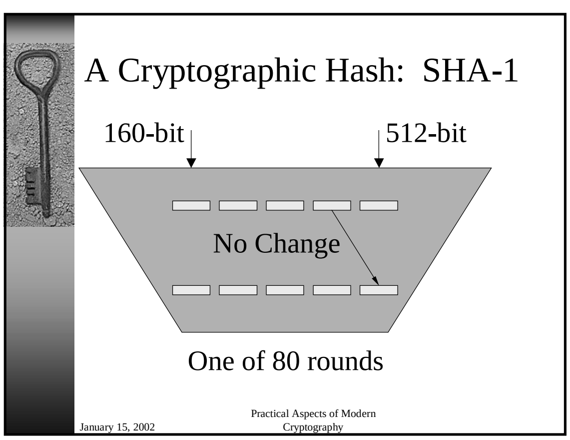


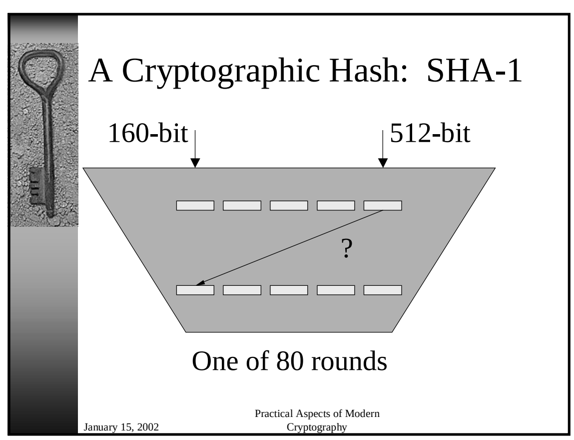

A Cryptographic Hash: SHA-1
160-bit
512-bit
One of 80 rounds
January 15, 2002
Practical Aspects of Modern Cryptography


A Cryptographic Hash: SHA-1
160-bit
512-bit
No Change
One of 80 rounds
January 15, 2002
Practical Aspects of Modern Cryptography


A Cryptographic Hash: SHA-1
160-bit
512-bit
Rotate 30 bits
One of 80 rounds
January 15, 2002
Practical Aspects of Modern Cryptography


A Cryptographic Hash: SHA-1
160-bit
512-bit
No Change
One of 80 rounds
January 15, 2002
Practical Aspects of Modern Cryptography


A Cryptographic Hash: SHA-1
160-bit
512-bit
No Change
One of 80 rounds
January 15, 2002
Practical Aspects of Modern Cryptography

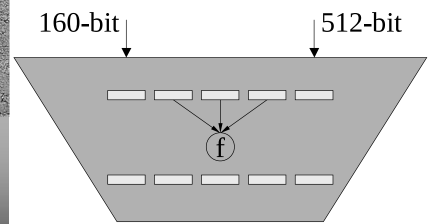

A Cryptographic Hash: SHA-1
160-bit
512-bit
?
One of 80 rounds
January 15, 2002
Practical Aspects of Modern Cryptography

A Cryptographic Hash: SHA-1

What's in the final 32-bit transform?

- Take the rightmost word.
- Add in the leftmost word rotated 5 bits.
- Add in a round-dependent function f of the middle three words.

A Cryptographic Hash: SHA-1



One of 80 rounds

A Cryptographic Hash: SHA-1

Depending on the round, the "non-linear" function f is one of the following.

$$f(X,Y,Z) = (X \wedge Y) \vee ((\neg X) \wedge Z)$$
$$f(X,Y,Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$$
$$f(X,Y,Z) = X \oplus Y \oplus Z$$

A Cryptographic Hash: SHA-1

What's in the final 32-bit transform?

- Take the rightmost word.
- Add in the leftmost word rotated 5 bits.
- Add in a round-dependent function f of the middle three words.

A Cryptographic Hash: SHA-1

What's in the final 32-bit transform?

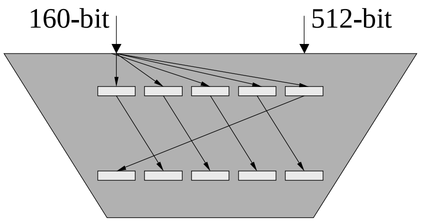
- Take the rightmost word.
- Add in the leftmost word rotated 5 bits.
- Add in a round-dependent function f of the middle three words.
- Add in a round-dependent constant.

A Cryptographic Hash: SHA-1

What's in the final 32-bit transform?

- Take the rightmost word.
- Add in the leftmost word rotated 5 bits.
- Add in a round-dependent function f of the middle three words.
- Add in a round-dependent constant.
- Add in a portion of the 512-bit message.

A Cryptographic Hash: SHA-1



One of 80 rounds

Cryptographic Tools

One-Way Trapdoor Functions
Public-Key Encryption Schemes
One-Way Functions
One-Way Hash Functions
Pseudo-Random Number-Generators
Secret-Key Encryption Schemes
Digital Signature Schemes