Cryptographic Hash

Göran Andersson

goeran@kth.se

Public Key Encryption

Cont'd

RSA Review

Alice and Bob want to communicate. They each create two large primes $p, q \geq 1024$ b and then compute

$$n = p\, q,\ \ \phi = (p-1)\,(q-1),\ \ e:\ \gcd(e, \phi) = 1,\ \ d \equiv_\phi e^{-1}$$

Both Alice and Bob publish their (e, n) but keep d secret.

Now, Alice wants to send m to Bob. She encrypts with Bob's public key (e_B, n_B).

$$c \equiv_{n_B} m^{e_B}$$

Bob decrypts the cipher c using his private key d_B.

$$m \equiv_{n_B} c^{d_B}$$

Explanation: if m and n_B are coprimes (most likely) we have, using Euler's theorem

$$c^{d_B} = (m^{e_B})^{d_B} = m^{e_B\, d_B} = m^{1+k\,\phi_B} = m\left(m^{\phi_B}\right)^k \equiv_{n_B} m\, 1^k = m$$

RSA Review

If $m \neq 0$ and n_B are not coprimes:

$$d \equiv_\phi e^{-1} \Leftrightarrow d\, e = 1 + k\, \phi = 1 + k' \operatorname{lcm}(p-1, q-1)$$

$$\therefore d\, e \equiv_{p-1} 1,\; d\, e \equiv_{q-1} 1$$

$$\Rightarrow \begin{cases} m^{d\,e} \equiv_p m \\ m^{d\,e} \equiv_q m \end{cases}$$

By the Chinese remainder theorem we have (also in this case) $m^{d\,e} \equiv_{p\,q} m$.

Elgamal Cryptosystem

- Public key cryptosystem
- Invented by Taher Elgamal
- Built on modulo arithmetic
- Security built on discrete logarithm problem

Elgamal Cryptosystem

Alice and Bob want to communicate. They each create a large prime p and find a generator (primitive root) g in $\mathbf{Z}_p$. Then pick a random number $x < p - 1$.

$y = g^x \bmod p$

Both Alice and Bob publish their (p, g, y) but keep x secret.

Now, Alice wants to send m to Bob. She finds a session random number $k < p_B - 1$. She encrypts with Bob's public key (p_B, g_B, y_B).

$a \equiv_{p_B} g_B^k$

$b \equiv_{p_B} m\, y_B^k$

The cipher is then the pair (a, b). Bob decrypts the cipher using his private key x_B.

$m \equiv_{p_B} b\,(a^{x_B})^{-1}$

Explanation:

$b\,(a^{x_B})^{-1} = m\, y_B^k \left(g_B^{k\,x_B}\right)^{-1} = m \left(g_B^{k\,x_B}\right)\left(g_B^{k\,x_B}\right)^{-1} \equiv_{p_B} m$

Problem

- Bob publish $(p, g, y) = (13, 6, 2)$ and keeps $x = 5$ secret. Alice generates $k = 5$ and wants to send $m = 5$. Help her!

Diffie-Hellman Key Exchange

- Key exchange protocol
- Invented by Whitfield Diffie and Martin Hellman
- Built on modulo arithmetic
- Security built on discrete logarithm problem
- Vulnerable to man-in-the-middle attack

Diffie-Hellman Key Exchange

Alice and Bob want to exchange a key over an unsecure channel. Alice and Bob agree on a large prime p and a generator g (primitive root) to $\mathbf{Z}_p$. These are public.

Alice picks a random number x in $\mathbf{Z}_p$ and sends

$$X = g^x \bmod p$$

to Bob. Bob picks a random number y in $\mathbf{Z}_p$ and sends

$$Y = g^y \bmod p$$

to Alice. Now, Alice computes her secret key

$$k_A = Y^x \bmod p$$

Bob computes his secret key

$$k_B = X^y \bmod p$$

They both have got the same key. Explanation:

$$k_A = Y^x = (g^y)^x = g^{yx} = g^{xy} = (g^x)^y = X^y = k_B$$

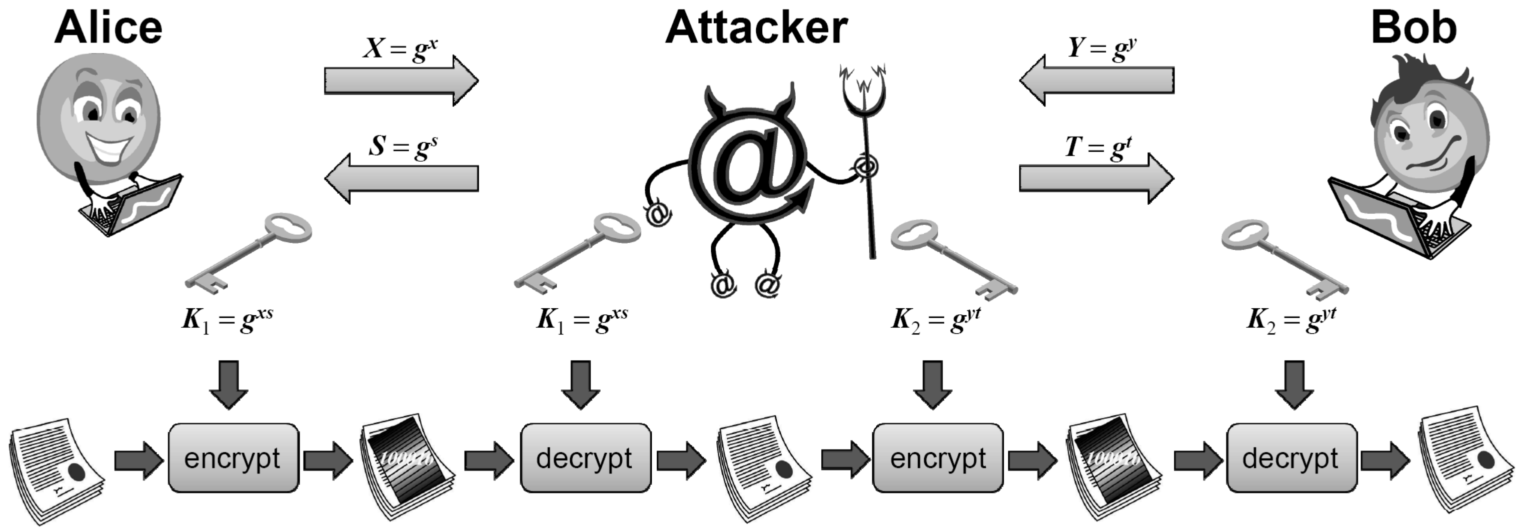


Figure 8.11: The man-in-the-middle attack against the DH protocol. First, by intercepting and modifying the messages of the DH protocol, the attacker establishes a secret key, K_1, with Alice and secret key, K_2, with Bob. Next, using keys K_1 and K_2, the attacker reads and forwards messages between Alice and Bob by decrypting and reencrypting them. Alice and Bob are unaware of the attacker and believe they are communicating securely with each other.

Problem

- With $p = 13$ and $g = 2$, find the common key if Alice and Bob generates the random numbers 3 and 7 respectively.

Pseudoprimality Testing

- The number of primes less than or equal to $\boldsymbol{n}$ is about $\boldsymbol{n} / \ln(\boldsymbol{n})$
- Thus, we expect to find a prime among $\boldsymbol{O}(\boldsymbol{b})$ randomly generated numbers with $\boldsymbol{b}$ bits each
- Testing whether a number is prime (primality testing) is a difficult problem, though polynomial-time algorithms exist
- Example: Rabin-Miller algorithm

Probability of Hitting a Prime

#primes $\leq x$ is denoted by $\pi(x)$. There are continuous estimates of $\pi(x)$, e.g. Riemann prime counting function $R(x)$. A very simple and well-known estimate is $x/\ln(x)$, which slightly under-estimates $\pi(x)$.

Using this simple estimate the probability of hitting a prime can be computed. Assume the interval is $[0, n) = \left[0, 2^b\right)$, where n consists of b bits. The probability of hitting a prime with an odd random integer is then

$$p_1 = 2\,\frac{\pi(n)}{n} \approx 2\,\frac{n/\ln(n)}{n} = \frac{2}{\ln(n)} = \frac{2}{b\ln(2)}$$

With b independent odd trials we have the following probability of hitting at least one prime:

$$p_b = 1 - (1-p_1)^b \approx 1 - \left(1 - \frac{2}{b\ln(2)}\right)^b \to 1 - e^{-2/\ln(2)} \approx 0.944$$

Probability of Hitting a Prime

Now if we focus on the interval $[n/2, n) = \left[2^{b-1}, 2^{b}\right)$ we get the prime hitting probability

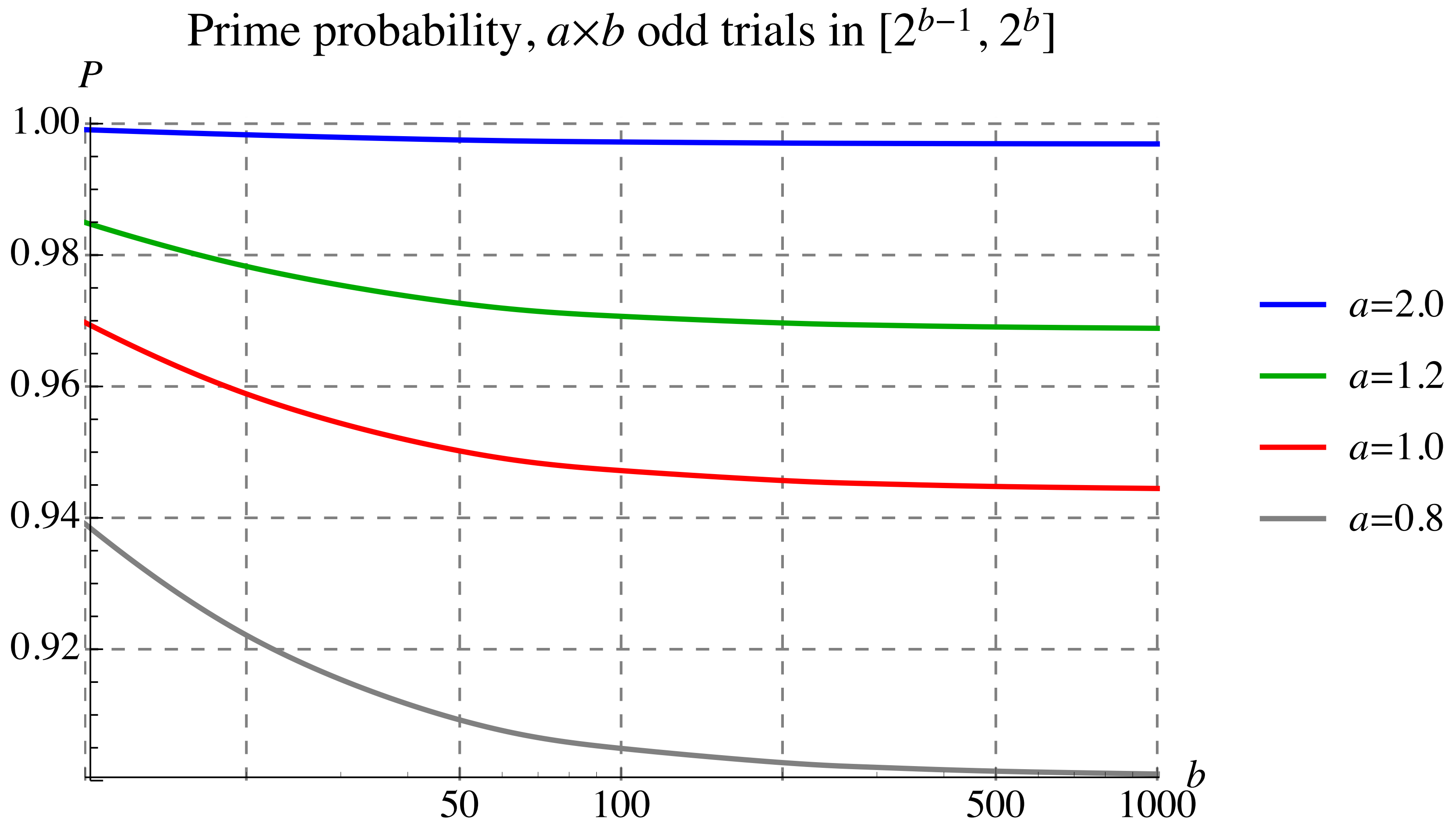
$$p_2 = 2\,\frac{\pi(n) - \pi(n/2)}{n - n/2} \approx 2\,\frac{n/\ln(n) - (n/2)/\ln(n/2)}{n - n/2} = \frac{2}{b\ln(2)}\left(1 - \frac{1}{b-1}\right)$$

With b independent odd trials we have the following probability of hitting at least one prime:

$$p'_b = 1 - (1 - p_2)^b \approx 1 - \left(1 - \frac{2}{b\ln(2)}\left(1 - \frac{1}{b-1}\right)\right)^b \to 1 - e^{-2/\ln(2)} \approx 0.944$$

The limit as $b \to \infty$ will be the same as in the $\left[0, 2^{b}\right)$ case. This means that we have a around 94 % probability of hitting at least one prime with b trials of b bits odd random numbers in the interval $\left[2^{b-1}, 2^{b}\right)$.

Probability of Hitting a Prime



Cryptographic Hash

Hash Functions

- A hash function h maps a plaintext x to a fixed-length value $x = h(P)$ called hash value or digest of P
  - A collision is a pair of plaintexts P and Q that map to the same hash value, $h(P) = h(Q)$
  - Collisions are unavoidable
  - For efficiency, the computation of the hash function should take time proportional to the length of the input plaintext

Cryptographic Hash Functions

- A cryptographic hash function satisfies additional properties
  - Preimage resistance (aka one-way)
    - Given a hash value x, it is hard to find a plaintext P such that $h(P) = x$
  - Second preimage resistance (aka weak collision resistance)
    - Given a plaintext P, it is hard to find a plaintext Q such that $h(P) = h(Q)$
  - Collision resistance (aka strong collision resistance)
    - It is hard to find a pair of plaintexts P and Q such that $h(P) = h(Q)$
- Collision resistance implies second preimage resistance
- Hash values of at least 256 bits recommended to defend against brute-force attacks

Birthday Attack

Message birthday, $m = 365$ days in a year (all messages). No duplicates.

$H_i =$ choice i doesn't collide

$$P(H_0) = \frac{365}{365} = 1$$

$$P(H_0 \cap H_1) = P(H_0)\, P(H_1 \mid H_0) = \frac{365}{365}\,\frac{365-1}{365} = 1 - \frac{1}{365}$$

$$P(H_0 \cap H_1 \cap H_2) = P((H_0 \cap H_1) \cap H_2) = P(H_0 \cap H_1)\, P(H_2 \mid H_0 \cap H_1)$$

$$= P(H_0)\, P(H_1 \mid H_0)\, P(H_2 \mid H_0 \cap H_1) = \frac{365}{365}\,\frac{365-1}{365}\,\frac{365-2}{365} = \left(1 - \frac{1}{365}\right)\left(1 - \frac{2}{365}\right)$$

Birthday Attack

The no collision probability after n random messages, without duplicates, is then

$$\therefore P\left(\bigcap_{i=0}^{n-1} H_i\right) = \prod_{i=0}^{n-1}\left(1-\frac{i}{m}\right) = \prod_{i=1}^{n-1}\left(1-\frac{i}{m}\right) = \frac{m!}{m^n\,(m-n)!} = m^{-n}\,m^{(n)}$$

The probability $p_c(n)$ of at least one collision is $p_c(n) = 1 - m^{-n}\,m^{(n)}$. Here we usually have $m > 2^{100} \approx 10^{30}$ and $n \ll m$ so the approximation

$$\frac{i}{m} = x \approx 0 \Rightarrow e^{-x} \approx 1 - x$$

comes in handy.

$$\therefore P\left(\bigcap_{i=0}^{n-1} H_i\right) = \prod_{i=1}^{n-1}\left(1-\frac{i}{m}\right) \approx \prod_{i=1}^{n-1} e^{-\frac{i}{m}} = e^{-\frac{1}{m}\sum_{i=1}^{n-1} i} = e^{-\frac{1}{m}(n-1)\frac{1+n-1}{2}} = e^{-\frac{n(n-1)}{2m}}$$

$$\Rightarrow p_c(n) \approx 1 - e^{-\frac{n(n-1)}{2m}}$$

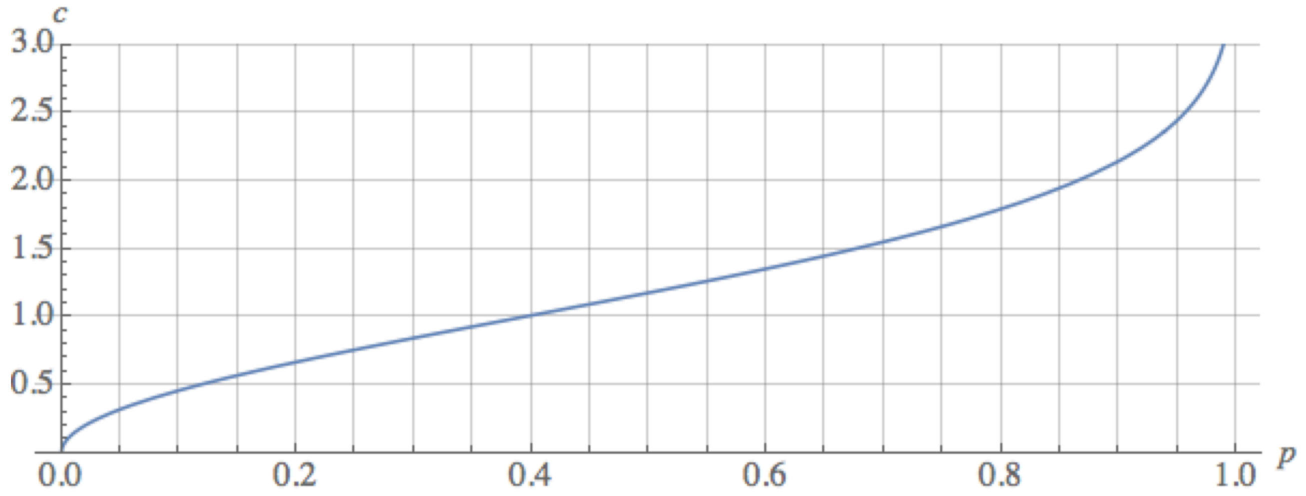
Birthday Attack

Solving for n we have:

$$-\ln(1-p) = \frac{n\,(n-1)}{2\,m} \Rightarrow n = \sqrt{\frac{1}{4} - 2\,m\ln(1-p)} + \frac{1}{2}$$

$$\approx \sqrt{-2\ln(1-p)\,m} \approx c\sqrt{m}$$

p	c
0.01	0.14
0.1	0.46
0.2	0.67
0.5	1.18
0.9	2.15



Birthday Attack

This means that $n \propto \sqrt{m}$ for reasonable collision probability. So if we have a b bits message digest we have to generate $2^{b/2}$ different messages in order to get a reasonable collision probability.

Example $m = 2^{32}$ and demanding $p = 0.9$ gives

$$n \approx 2.15\sqrt{2^{32}} \approx 1.4 \times 10^3$$

Notice that $n \ll m$.With $m = 2^{128}$ we get $n \approx 4 \times 10^{19}$ (inconvenient) and with $m = 2^{256}$ we get $n \approx 7 \times 10^{38}$ (unrealistic).

Problem

- Compute the birthday collision probability in this student group.

$$\text{Let's say } n = 25 \Rightarrow p = 1 - 365^{-25}\, 365^{(25)} \approx 1 - e^{-\frac{25\times 24}{2\times 365}} \approx 0.56$$

- If we want 90% collision probability, how many messages are needed with a 64 bits digest?
- 512 bits?

$$p = 0.9 \Rightarrow c = \sqrt{-2\ln(0.1)} \approx 2.15$$

$$m = 2^{64} \Rightarrow n \approx 2.15\sqrt{2^{64}} = 2.15 \times 2^{32} \approx 9.2 \times 10^{9}$$

$$m = 2^{512} \Rightarrow n \approx 2.15 \times 2^{256} \approx 2.5 \times 10^{77}$$

Message-Digest Algorithm 5 (MD5)

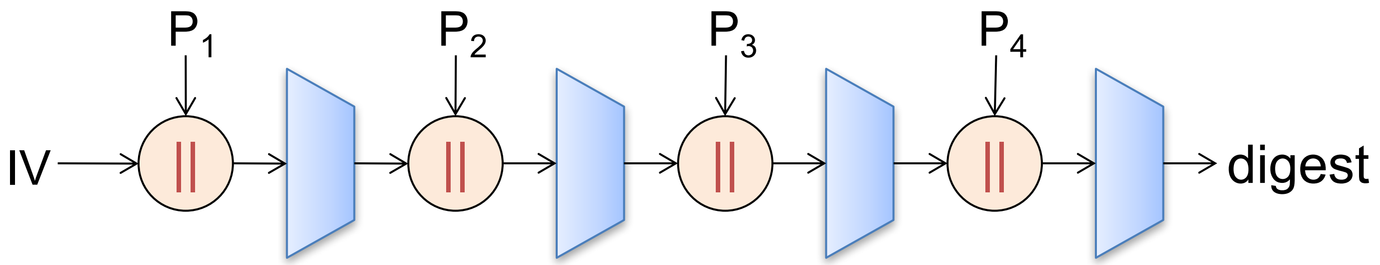
- Developed by Ron Rivest in 1991
- Uses 128-bit hash values
- Still widely used in legacy applications although considered insecure
- Various severe vulnerabilities discovered
- Chosen-prefix collisions attacks found by Marc Stevens, Arjen Lenstra and Benne de Weger
  - Start with two arbitrary plaintexts P and Q
  - One can compute suffixes S_1 and S_2 such that $P \,||\, S_1$ and $Q \,||\, S_2$ collide under MD5 by making 250 hash evaluations
  - Using this approach, a pair of different executable files or PDF documents with the same MD5 hash can be computed

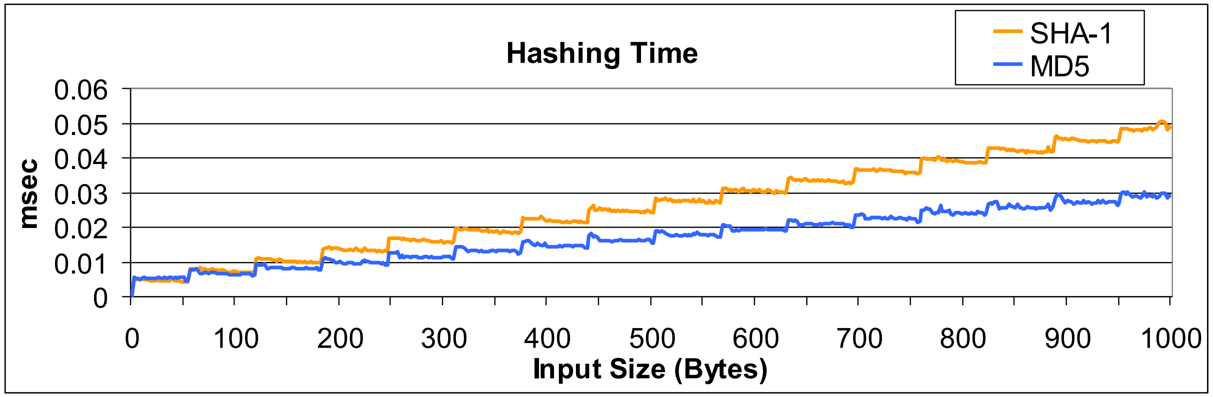
Secure Hash Algorithm (SHA)

- Developed by NSA and approved as a federal standard by NIST
- SHA-0 and SHA-1 (1993)
  - 160-bits
  - Considered insecure
  - Still found in legacy applications
  - Vulnerabilities less severe than those of MD5
- SHA-2 family (2002)
  - 256 bits (SHA-256) or 512 bits (SHA-512)
  - Still considered secure despite published attack techniques
- SHA-3 released 2015

Iterated Hash Function

- A compression function works on input values of fixed length
- An iterated hash function extends a compression function to inputs of arbitrary length
  - padding, initialization vector, and chain of compression functions
  - inherits collision resistance of compression function
- MD5 and SHA are iterated hash functions


P1
P2
P3
P4
IV
||
digest


Hashing Time
SHA-1
MD5
msec
Input Size (Bytes)