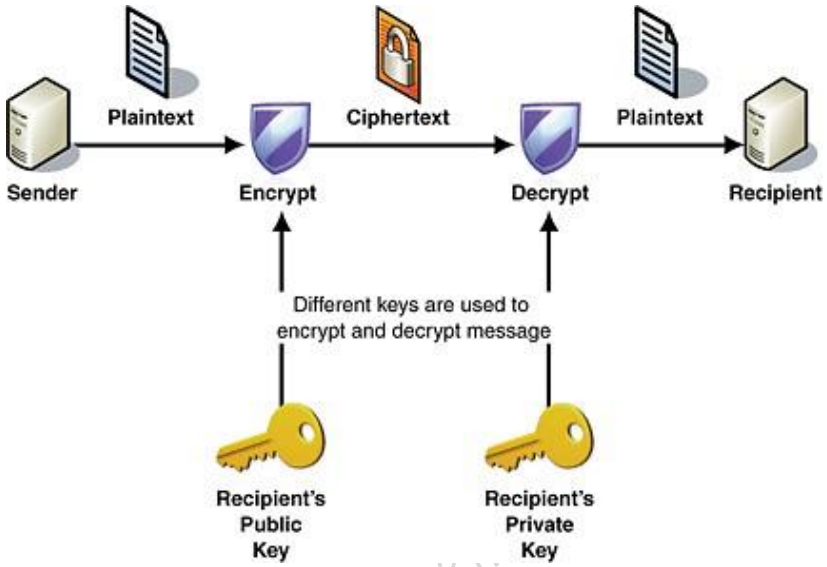
1. CÁC GIẢI THUẬT MÃ HÓA KHÓA BẤT ĐỐI XỨNG

1.1 Khái quát về mã hóa khóa bất đối xứng

Mã hóa khóa bất đối xứng, đôi khi được gọi là mã hóa khóa công khai sử dụng một cặp khóa cho quá trình mã hóa và giải mã. Trong cặp khóa, khóa công khai được sử dụng cho mã hóa và khóa riêng được sử dụng cho giải mã. Chỉ khóa riêng cần giữ bí mật, còn khóa công khai có thể phổ biến rộng rãi, nhưng phải đảm bảo tính toàn vẹn và xác thực chủ thể của khóa.

Hình 3.27 minh họa quá trình mã hóa (Encrypt) và giải mã (Decrypt) sử dụng mã hóa khóa bất đối xứng. Theo đó, người gửi (Sender) sử dụng khóa công khai (Public key) của người nhận (Recipient) để mã hóa bản rõ (Plaintext) thành bản mã (Ciphertext) và gửi nó cho người nhận. Người nhận nhận được bản mã sử dụng khóa riêng (Private key) của mình để giải mã khôi phục bản rõ.

Đặc điểm nổi bật của các hệ mã hóa khóa bất đối xứng là kích thước khóa lớn, lên đến hàng ngàn bit. Do vậy, các hệ mã hóa dạng này thường có tốc độ thực thi chậm hơn nhiều lần so với các hệ mã hóa khóa đối xứng với độ an toàn tương đương. Mặc dù vậy, các hệ mã hóa khóa bất đối xứng có khả năng đạt độ an toàn cao và ưu điểm nổi bật nhất là việc quản lý và phân phối khóa đơn giản hơn do khóa công khai có thể phân phối rộng rãi.



Hình 3.27. Mã hóa và giải mã trong hệ mã hóa bất đối xứng

Các giải thuật mã hóa khóa bất đối xứng điển hình bao gồm: RSA, Rabin, ElGamal, McEliece và Knapsack. Trong mục tiếp theo chúng ta tìm hiểu về giải thuật mã hóa RSA – một trong các giải thuật mã hóa khóa đối xứng được sử dụng rộng rãi nhất trên thực tế.

1.2 Giải thuật mã hóa RSA

1.2.1 Giới thiệu

Giải thuật mã hóa RSA được 3 nhà khoa học người Mỹ là Ronald Rivest, Adi Shamir và Leonard Adleman phát minh năm 1977, và tên giải thuật RSA lấy theo chữ cái đầu của tên 3 đồng tác giả. Độ an toàn của RSA dựa trên tính khó của việc phân tích số nguyên rất lớn, với độ lớn cỡ hàng trăm chữ số thập phân. Giải thuật RSA sử dụng một cặp khóa, trong đó khóa công khai dùng để mã hóa và khóa riêng dùng để giải mã. Chỉ khóa riêng RSA cần giữ bí mật. Khóa công khai có thể công bố rộng rãi. Hiện nay, các khóa RSA có kích thước nhỏ hơn 1024 bit được coi là không an toàn do tốc độ các hệ thống máy tính tăng nhanh. Để đảm bảo an toàn, khuyến nghị sử dụng khóa 2048 bit trong giai đoạn 2010-2020. Trong tương lai, cần sử dụng khóa RSA có kích thước lớn hơn, chẳng hạn 3072 bit.

1.2.2 Sinh khóa

RSA cung cấp một thủ tục sinh cặp khóa (khóa công khai và khóa riêng) tương đối đơn giản. Cụ thể, thủ tục sinh khóa gồm các bước như sau:

- Tạo 2 số nguyên tố p và q;
- Tính modulo $n = p \times q$
- Tính $\Phi(n) = (p-1) \times (q-1)$
- Chọn số e sao cho $0 < e < \Phi(n)$ và $\gcd(e, \Phi(n)) = 1$, trong đó hàm gcd() tính ước số chung lớn nhất của 2 số nguyên. Nếu $\gcd(e, \Phi(n)) = 1$ thì e và $\Phi(n)$ là 2 số nguyên tố cùng nhau.
- Chọn số d sao cho $d \equiv e^{-1} \bmod \Phi(n)$,

hoặc $(d \times e) \bmod \Phi(n) = 1$

hay d là modulo nghịch đảo của e.

- Ta có (n, e) là khóa công khai, (n, d) là khóa riêng và n còn được gọi là modulo.

1.2.3 Mã hóa và giải mã

- Mã hóa

+ Thông điệp bản rõ m đã được chuyển thành số, với $m < n$. Nếu thông điệp bản rõ m có kích thước lớn thì được chia thành các khối m_i, với $m_i < n$.

+ Bản mã $c = m^e \bmod n$

- Giải mã

+ Bản mã c, với $c < n$

+ Bản rõ $m = c^d \bmod n$

1.2.4 Ví dụ

- Sinh khóa:

+ Chọn 2 số nguyên tố $p = 3$ và $q = 11$

+ Tính $n = p \times q = 3 \times 11 = 33$

+ Tính $\Phi(n) = (p-1) \times (q-1) = 2 \times 10 = 20$

+ Chọn số e sao cho $0 < e < 20$, và e và $\Phi(n)$ là số nguyên tố cùng nhau ($\Phi(n)$ không chia hết cho e). Chọn $e = 7$

+ Tính (d x e) mod Φ(n) → $(d \times 7) \bmod 20 = 1$

$d = (20 \times k + 1)/7$ → $d = 3$ $(k=1)$

+ Ta có: khóa công khai là (33, 7) và khóa riêng là (33, 3)

- Mã hóa:

+ Với bản rõ $m = 6$,

+ $c = m^e \bmod n = 6^7 \bmod 33 = 279936 \bmod 33 = 30$

+ Vậy bản mã $c = 30$

- Giải mã:

+ Với bản mã $c = 30$

+ $m = c^d \bmod n = 30^3 \bmod 33 = 27000 \bmod 33 = 6$

+ Vậy bản rõ $m = 6$.

1.2.5 Một số yêu cầu với quá trình sinh khóa

Dưới đây liệt kê các yêu cầu đặt ra với các tham số sinh khóa và khóa để đảm bảo sự an toàn cho cặp khóa RSA. Các yêu cầu cụ thể gồm:

- Yêu cầu với các tham số sinh khóa p và q:

+ Các số nguyên tố p và q phải được chọn sao cho việc phân tích n ($n = p \times q$) là không khả thi về mặt tính toán. p và q nên có cùng độ lớn (tính bằng bit) và phải là các số đủ lớn. Nếu n có kích thước 2048 bit thì p và q nên có kích thước khoảng 1024 bit.

+ Hiệu số $p - q$ không nên quá nhỏ, do nếu $p - q$ quá nhỏ, tức $p \approx q$ và $p \approx \sqrt{n}$. Như vậy, có thể chọn các số nguyên tố ở gần $\sqrt{n}$ và thử. Khi có được p, có thể tính q và tìm ra d là khóa bí mật từ khóa công khai e và $\Phi(n) = (p - 1)(q - 1)$. Nếu p và q được chọn ngẫu nhiên và $p - q$ đủ lớn, khả năng hai số này bị phân tích từ n giảm đi.

- Vấn đề sử dụng số mũ mã hóa (e) nhỏ: Khi sử dụng số mũ mã hóa (e) nhỏ, chẳng hạn

$e = 3$ có thể tăng tốc độ mã hóa. Kẻ tấn công có thể nghe lén và lấy được bản mã, từ đó phân tích bản mã để khôi phục bản rõ. Do số mũ mã hóa nhỏ nên chi phí cho phân tích, hoặc vét cạn không quá lớn. Do vậy, nên sử dụng số mũ mã hóa e đủ lớn và thêm chuỗi ngẫu nhiên vào khối rõ trước khi mã hóa để giảm khả năng bị vét cạn hoặc phân tích bản mã.

- Vấn đề sử dụng số mũ giải mã (d) nhỏ: Khi sử dụng số mũ giải mã (d) nhỏ, có thể tăng tốc độ giải mã. Nếu d nhỏ và gcd(p-1, q-1) cũng nhỏ thì d có thể tính được tương đối dễ dàng từ khóa công khai (n, e). Do vậy, để đảm bảo an toàn, nên sử dụng số mũ giải mã d đủ lớn.

2. Các hàm băm

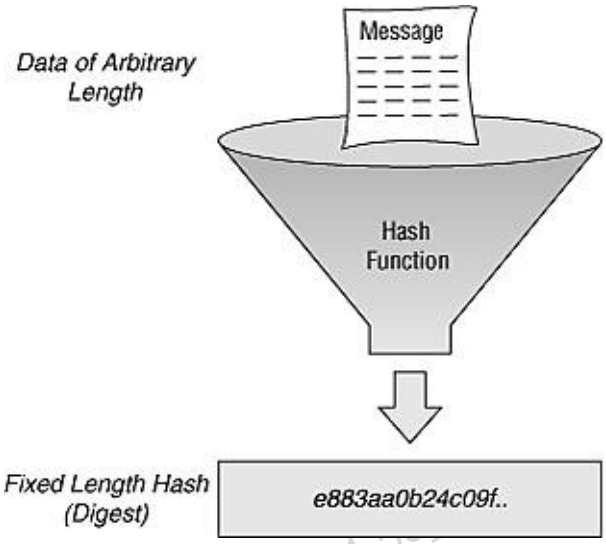
2.1 Khái quát về hàm băm

2.1.1 Giới thiệu

Hàm băm (hash function) là một hàm toán học h có tối thiểu 2 thuộc tính:

- Nén (Compression): h là một ánh xạ từ chuỗi đầu vào x có chiều dài bất kỳ sang một chuỗi đầu ra $h(x)$ có chiều dài cố định n bit;
- Dễ tính toán (Ease of computation): cho trước hàm h và đầu vào x, việc tính toán

$h(x)$ là dễ dàng.

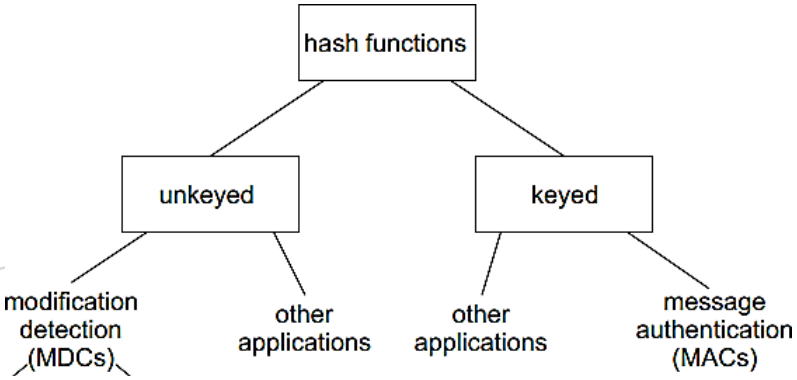


Hình 3.28. Mô hình nén thông tin của hàm băm

Hình 3.28 minh họa mô hình nén thông tin của hàm băm, theo đó thông điệp (Message) đầu vào với chiều dài tùy ý đi qua nhiều vòng xử lý của hàm băm để tạo chuỗi rút gọn, hay chuỗi đại diện (Digest) có kích thước cố định ở đầu ra.

2.1.2 Phân loại

Có thể phân loại các hàm băm theo khóa sử dụng hoặc theo chức năng. Theo khóa sử dụng, các hàm băm gồm 2 loại: hàm băm không khóa (unkeyed) và hàm băm có khóa (keyed), như biểu diễn trên Hình 3.29. Trong khi hàm băm không khóa nhận đầu vào chỉ là thông điệp (dạng $h(x)$, với hàm băm h và thông điệp x), hàm băm có khóa nhận đầu vào gồm thông điệp và khóa bí mật (theo dạng $h(x, K)$, với hàm băm h và thông điệp x và K là khóa bí mật). Trong các hàm băm không khóa, các mã phát hiện sửa đổi (MDC – Modification Detection Code) được sử dụng rộng rãi nhất, bên cạnh một số hàm băm không khóa khác. Tương tự, trong các hàm băm có khóa, các mã xác thực thông điệp (MAC - Message Authentication Code) được sử dụng rộng rãi nhất, bên cạnh một số hàm băm có khóa khác.



Hình 3.29. Phân loại các hàm băm theo khóa sử dụng

Theo chức năng, có thể chia các hàm băm thành 2 loại chính:

- Mã phát hiện sửa đổi (MDC - Modification Detection Code): MDC thường được sử dụng để tạo chuỗi đại diện cho thông điệp và dùng kết hợp với các kỹ thuật khác (như chữ ký số) để đảm bảo tính toàn vẹn của thông điệp. MDC thuộc loại hàm băm không khóa. MDC gồm 2 loại nhỏ:

+ Hàm băm một chiều (OWHF - One-way hash functions): Với hàm băm một chiều, việc tính giá trị băm là dễ dàng, nhưng việc khôi phục thông điệp từ giá trị băm là rất khó khăn;

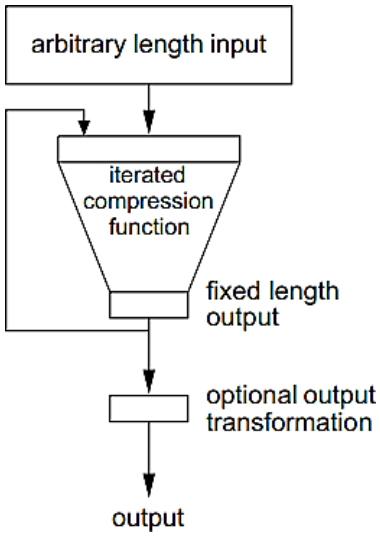
+ Hàm băm chống đụng độ (CRHF - Collision resistant hash functions): Với hàm băm chống đụng độ, sẽ là rất khó để tìm được 2 thông điệp khác nhau nhưng có cùng giá trị băm.

- Mã xác thực thông điệp (MAC - Message Authentication Code): MAC cũng được dùng để đảm bảo tính toàn vẹn của thông điệp mà không cần một kỹ thuật bổ sung nào khác. MAC là loại hàm băm có khóa như đã

đề cập ở trên, với đầu vào là thông điệp và một khóa bí mật.

2.1.3 Mô hình xử lý dữ liệu

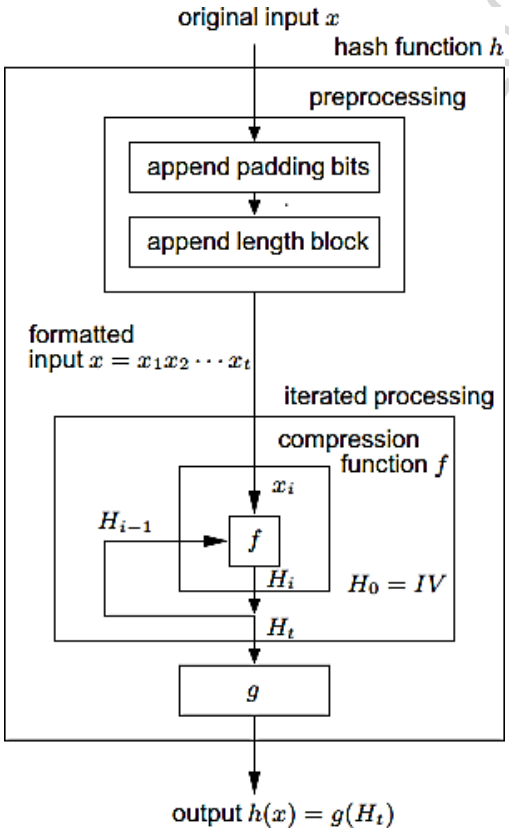
Hình 3.30 biểu diễn mô hình tổng quát xử lý dữ liệu của các hàm băm. Theo đó, thông điệp đầu vào với độ dài tùy ý (arbitrary length input) đi qua hàm nén lặp nhiều vòng (iterated compression function) để tạo chuỗi đầu ra có kích thước cố định (fixed length output). Chuỗi này đi qua một khâu chuyển đổi định dạng tùy chọn (optional output transformation) để tạo ra chuỗi băm kết quả (output).



Hình 3.30. Mô hình tổng quát xử lý dữ liệu của hàm băm

Hình 3.31 mô tả chi tiết quá trình xử lý dữ liệu của các hàm băm. Theo đó, quá trình xử lý gồm 3 bước chính: (1) tiền xử lý (preprocessing), (2) xử lý lặp (iterated processing) và (3) chuyển đổi định dạng. Trong bước tiền xử lý, thông điệp đầu vào x trước hết được nối đuôi thêm một số bit và kích thước khối, sau đó chia thành các khối có kích thước xác định. Kết quả của bước này là t khối dữ liệu có cùng kích thước có dạng $x = x_1x_2...x_t$ làm đầu vào cho bước 2. Trong bước 2, từng khối dữ liệu x_i được xử lý thông qua hàm nén f

để tạo đầu ra là H_i. Kết quả của bước 2 là chuỗi đầu ra H_t và H_t được chuyển đổi định dạng bởi hàm g để tạo chuỗi giá trị băm hết quả $h(x)$.



Hình 3.31. Mô hình chi tiết xử lý dữ liệu của hàm băm

2.2 Một số hàm băm thông dụng

Các hàm băm thông dụng giới thiệu trong mục này đều là các hàm băm không khóa, gồm các họ hàm băm chính như sau:

- Họ hàm băm MD (Message Digest) gồm các hàm băm MD2, MD4, MD5 và MD6.
- Họ hàm băm SHA (Secure Hash Algorithm) gồm các hàm băm SHA0, SHA1, SHA2 và SHA3.
- Một số hàm băm khác, gồm CRC (Cyclic redundancy checks), Checksums,...

Các mục con tiếp theo của mục này giới thiệu 2 hàm băm đã và đang được sử dụng rộng rãi nhất là hàm băm MD5 và SHA1.

2.2.1 Hàm băm MD5

* Giới thiệu

MD5 (Message Digest) là hàm băm không khóa được Ronald Rivest thiết kế năm 1991 để thay thế MD4. Chuỗi giá trị băm đầu ra của MD5 là 128 bit (16 byte) và thường được biểu diễn thành 32 số hexa. MD5 được sử dụng khá rộng rãi trong nhiều ứng dụng, như tạo chuỗi đảm bảo tính toàn vẹn thông điệp, tạo chuỗi kiểm

tra lỗi, hoặc kiểm tra tính toàn vẹn dữ liệu (Checksum) và mã hóa mật khẩu trong các hệ điều hành và các ứng dụng. MD5 hiện nay được khuyến nghị không nên sử dụng do nó không còn đủ an toàn.

Nhiều điểm yếu của MD5 đã bị khai thác, như điển hình MD5 bị khai thác bởi mã độc Flame vào năm 2012.

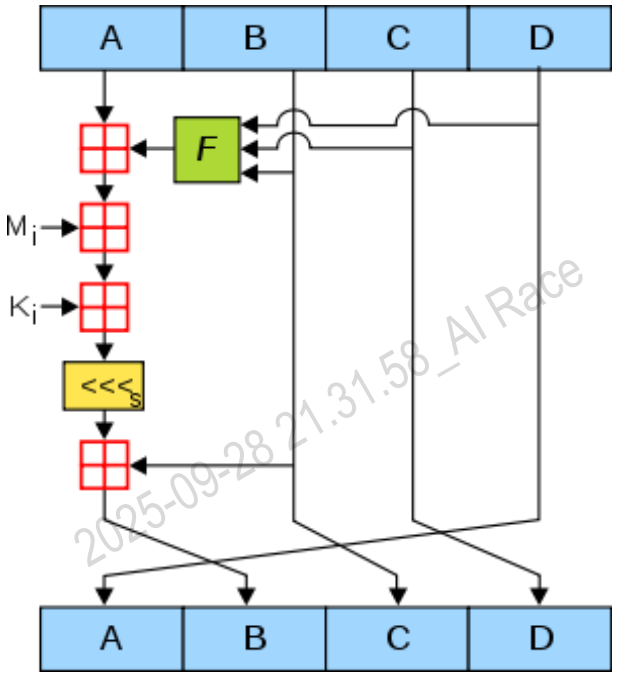
* Quá trình xử lý thông điệp

Quá trình xử lý thông điệp của MD5 gồm 2 khâu là *tiền xử lý* và *các vòng lặp xử lý*.

Cụ thể, chi tiết về các khâu này như sau:

- Tiền xử lý: Thông điệp được chia thành các khối 512 bit (16 từ 32 bit). Nếu kích thước thông điệp không là bội số của 512 thì nối thêm số bit còn thiếu.
- Các vòng lặp xử lý: Phần xử lý chính của MD5 làm việc trên *state* 128 bit, chia thành 4 từ 32 bit (A, B, C, D):

+ Các từ A, B, C, D được khởi trị bằng một hằng cố định;
+ Từng phần 32 bit của khối đầu vào 512 bit được đưa dần vào để thay đổi *state*;
+ Quá trình xử lý gồm 4 vòng, mỗi vòng gồm 16 thao tác tương tự nhau.
+ Mỗi thao tác gồm: Xử lý bởi hàm F (4 dạng hàm khác nhau cho mỗi vòng), Cộng modulo và Quay trái. Hình 3.32 biểu diễn lưu đồ xử lý của một thao tác của MD5, trong đó A, B, C, D là các từ 32 bit của *state*, Mi: khối 32 bit thông điệp đầu vào, Ki là 32 bit hằng khác nhau cho mỗi thao tác, <<<s là thao tác dịch trái *s* bit, ⊞ biểu diễn phép cộng modulo 32 bit và F là hàm phi tuyến tính.



Hình 3.32. Lưu đồ xử lý một thao tác của MD5

Hàm F gồm 4 dạng được dùng cho 4 vòng lặp. Cụ thể, F có các dạng như sau: $F(B, C, D) = (B \wedge C) \vee (\neg B \wedge D)$

$G(B, C, D) = (B \wedge D) \vee (C \wedge \neg D)$

$H(B, C, D) = B \oplus C \oplus D$

$I(B, C, D) = C \oplus (B \vee \neg D)$

trong đó, các ký hiệu $\oplus$, $\wedge$, $\vee$, $\neg$ biểu diễn các phép toán lô gíc XOR, AND, OR và NOT tương ứng.

2.2.2 Hàm băm SHA1

* Giới thiệu

SHA1 (Secure Hash Function) được Cơ quan mật vụ Mỹ thiết kế năm 1995 để thay thế cho hàm băm SHA0. Chuỗi giá trị băm đầu ra của SHA1 có kích thước 160 bit và thường được biểu diễn thành 40 số hexa. Tương tự MD5, SHA1 được sử dụng rộng rãi để đảm bảo tính xác thực và toàn vẹn thông điệp.

* Quá trình xử lý thông điệp

SHA1 sử dụng thủ tục xử lý thông điệp tương tự MD5, cũng gồm 2 khâu là *tiền xử lý*
và *các vòng lặp xử lý*. Cụ thể, chi tiết về các khâu này như sau:

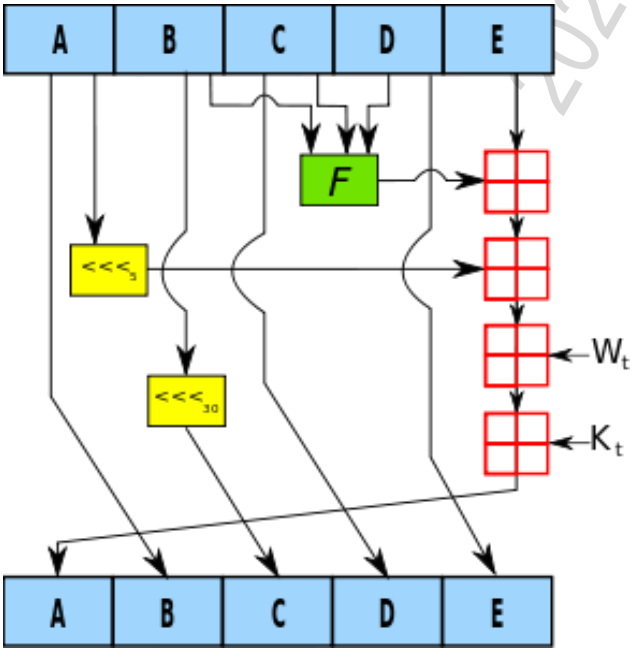
- Tiền xử lý: Thông điệp được chia thành các khối 512 bit (16 từ 32 bit). Nếu kích thước thông điệp không là bội số của 512 thì nối thêm số bit còn thiếu.
- Các vòng lặp xử lý: Phần xử lý chính của SHA1 làm việc trên *state* 160 bit, chia thành 5 từ 32 bit (A, B, C, D, E):

+ Các từ A, B, C, D, E được khởi trị bằng một hằng cố định;

+ Từng phần 32 bit của khối đầu vào 512 bit được đưa dần vào để thay đổi *state*;

+ Quá trình xử lý gồm 80 vòng, mỗi vòng gồm các thao tác: add, and, or, xor, rotate, mod.

+ Mỗi vòng xử lý gồm: Xử lý bởi hàm phi tuyến tính F (có nhiều dạng hàm khác nhau), Cộng modulo và Quay trái. Hình 3.33 biểu diễn lưu đồ một vòng xử lý của SHA1, trong đó A, B, C, D, E là các từ 32 bit của *state*, W_t: khối 32 bit thông điệp đầu vào, K_t là 32 bit hằng khác nhau cho mỗi vòng, <<<n là thao tác dịch trái *n* bit, ⊞ biểu diễn phép cộng modulo 32 bit và F là hàm phi tuyến tính.



Hình 3.33. Lưu đồ một vòng xử lý của SHA1