ĐẠI HỌC QUỐC GIA TP.HCM

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Đồ án:

Time-based one-time password (TOTP)

Thành viên nhóm: Bùi Hoàng Trúc Anh - 21521817

Nguyễn Ngọc Trà My - 21520353

Lê Hoàng Oanh - 21521253

I. Tổng quan

1. Ngữ cảnh ứng dụng

Để tăng tính bảo mật trong các ứng dụng chuyển tiền online, server sẽ yêu cầu người dùng nhập lại mật khẩu và tạo thêm mã TOTP để xác thực người dùng.

Rủi ro:

- Việc triển khai số lần đăng nhập không giới hạn cho phép kẻ tấn công thử nhiều lần để dò mã TOTP.
- Nếu kẻ tấn công đánh cắp được khóa bí mật K thì có thể tự do tạo mã TOTP mới hợp lệ.

Mục tiêu bảo mật:

- Tính xác thực: TOTP cần đảm bảo rằng mã xác thực được tạo ra chỉ có thể được sử dụng một lần và chỉ có thể được tạo ra bởi người dùng cụ thể đã được cấp quyền truy cập. Điều này giúp đảm bảo tính xác thực và ngăn chặn bất kỳ ai khác truy cập vào tài khoản.
- Đảm bảo tính bảo mật: TOTP cần đảm bảo rằng mã xác thực được tạo ra không thể bị đoán trước hoặc giả mạo bởi các kẻ xấu. Điều này được đạt được bằng cách sử dụng thuật toán bảo mật và các bộ mã hóa khác.

II. Thuật toán

1. SHA256

SHA-2 là một tập hợp các hàm băm mật mã do Cơ quan An ninh Quốc gia Hoa Kỳ (NSA) thiết kế và xuất bản lần đầu vào năm 2001. Chúng được xây dựng bằng cách sử dụng cấu trúc Merkle–Damgård, từ chính chức năng nén một chiều được xây dựng bằng cách sử dụng cấu trúc Davies–Meyer từ một mật mã khối chuyên dụng.

SHA-2 bao gồm những thay đổi đáng kể so với người tiền nhiệm của nó, SHA-1. Họ SHA-2 bao gồm nhiều hàm băn khác nhau. Chúng sử dụng lượng dịch chuyển và hằng số cộng khác nhau, nhưng cấu trúc của chúng hầu như giống hệt nhau, chỉ khác nhau về số vòng.

Giải thuật:

1. Nhồi thêm dữ liệu

Bắt đầu với thông điệp ban đầu có độ dài L bit.

Nối thêm một bit '1'. Nối thêm K bit '0', trong đó K là số nhỏ nhất >= 0 sao cho (L + 1 + K + 64) là bội số của 512.

Nối 64 bit vào cuối, trong đó 64 bit là số nguyên big-endian đại diện cho độ dài của đầu vào ban đầu ở dạng nhị phân, làm sao cho tổng độ dài được xử lý sau là bội số của 512 bit.

<Thông điệp ban đầu có độ dài L> 1 <K số không> <64 bit đại diện cho L>

2. Khởi tao giá tri băm

Bây giờ chúng ta tạo 8 giá trị băm. Đây là các hằng số được mã hóa cứng đại diện cho 32 bit đầu tiên của phần phân số của căn bậc hai của 8 số nguyên tố đầu tiên: 2, 3, 5, 7, 11, 13, 17, 19.

h0 = 0x6a09e667	h4 = 0x510e527f
h1 = 0xbb67ae85	h5 = 0x9b05688c
h2 = 0x3c6ef372	h6 = 0x1f83d9ab
h3 = 0xa54ff53a	h7 = 0x5be0cd19

3. Khởi tao hằng số tròn

Lần này chúng ta có 64 hằng số, mỗi giá trị (0-63) là 32 bit đầu tiên của các phần phân số của căn bậc hai của 64 số nguyên tố đầu tiên (2 - 311).

K[0..63] =0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5, 0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5, 0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3, 0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174, 0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc, 0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da, 0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7, 0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967, 0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13, 0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85, 0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3, 0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070, 0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5, 0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3, 0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208, 0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2

4. Xử lý các khối dữ liệu 512 bit

Chia thông điệp thành các khối 512-bit.

Tạo một mảng lịch trình nhắn tin w[0..63] với 64 đầu vào, trong đó mỗi mục nhập là một từ 32 bit. Sao chép khối dữ liệu đã được xử lý trước đó vào 16 từ đầu tiên w[0..15] của mảng lịch trình nhắn tin.

Mở rộng 16 từ đầu tiên thành 48 từ còn lại của w[16..63]:

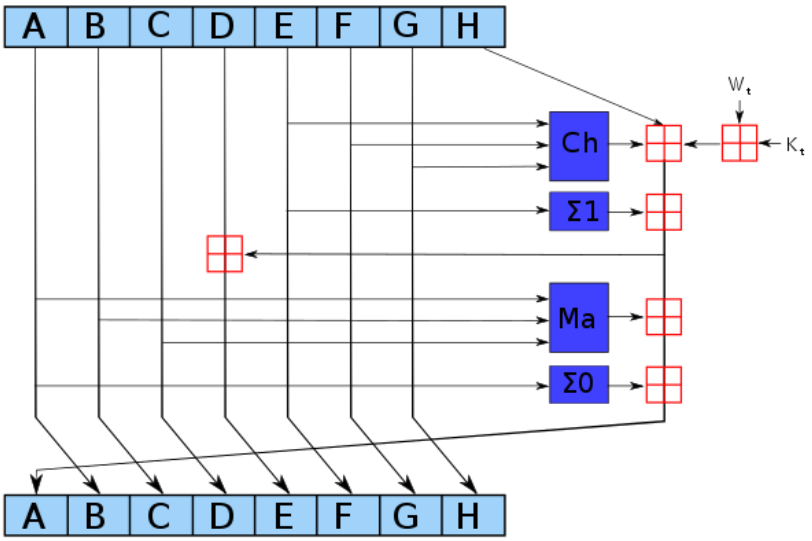
từ 16 đến 63

s0 = (w[i-15] rightrotate 7) xor (w[i-15] rightrotate 18) xor (w[i-15] rightshift 3)

s1 = (w[i- 2] rightrotate 17) xor (w[i- 2] rightrotate 19) xor (w[i- 2] rightshift 10)

w[i] = w[i-16] + s0 + w[i-7] + s1

5. Nén



Khởi tạo các biến a, b, c, d, e, f, g, h và đặt chúng bằng các giá trị băm hiện tại tương ứng. h0, h1, h2, h3, h4, h5, h6, h7

Chạy vòng lặp nén. Vòng lặp nén sẽ thay đổi các giá trị từ a…h. Tất cả phép cộng được tính theo modulo 2 ^ 32. Vòng lặp nén như sau:

Từ 0 đến 63

S1 = (e rightrotate 6) xor (e rightrotate 11) xor (e rightrotate 25)

ch = (e and f) xor ((not e) and g)

temp1 = h + S1 + ch + k[i] + w[i]

S0 = (a rightrotate 2) xor (a rightrotate 13) xor (a rightrotate 22)

maj = (a and b) xor (a and c) xor (b and c)

temp2 := S0 + maj

h = g	d = c
g = f	c = b
f = e	b = a
e = d + temp1	a = temp1 + temp2

6. Sửa đổi giá tri

h0 = h0 + a	h4 = h4 + e
h1 = h1 + b	h5 = h5 + f
h2 = h2 + c	h6 = h6 + g
h3 = h3 + d	h7 = h7 + h

7. Tao giá tri băm cuối cùng

Cuối cùng ghép tất cả lại với nhau, ta có một phép nối chuỗi đơn giản.

digest = hash = h0 **append** h1 **append** h2 **append** h3 **append** h4 **append** h5 **append** h6 **append** h7

2. HMAC

HMAC (Hash-based message authentication code) là cơ chế tính toán mã xác thực thông báo liên quan đến hàm băm như MD5 (Message Digest), SHA-1 hoặc hàm băm khác kết hợp với khóa bí mật.

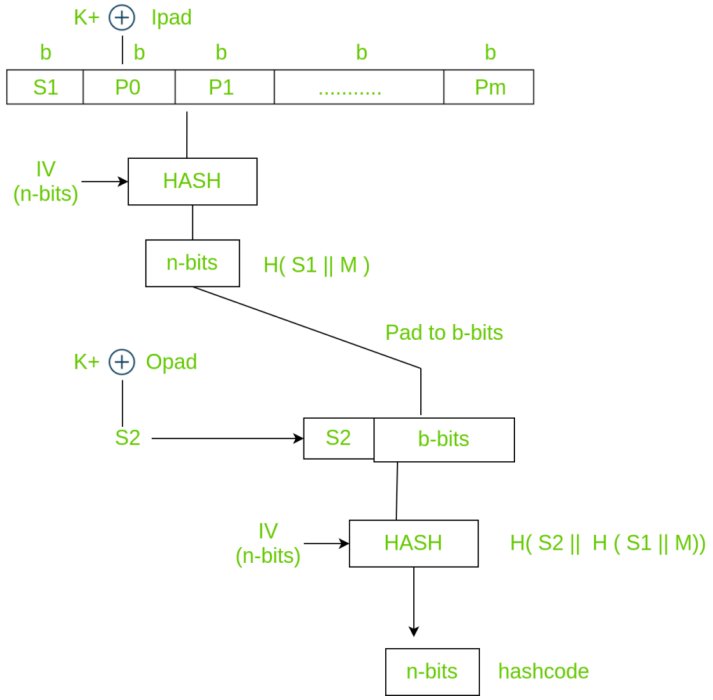
HMAC gần giống như chữ ký điện tử. Cả hai đều thực thi tính toàn vẹn và tính xác thực. Cả hai đều sử dụng các khóa mật mã và cả hai đều sử dụng các hàm băm. Sự khác biệt chính là chữ ký điện tử sử dụng khóa bất đối xứng, trong khi HMAC sử dụng khóa đối xứng.

HMAC sử dụng hai lần tính toán hàm băm. Khóa bí mật đầu tiên được sử dụng để lấy hai khóa – bên trong và bên ngoài. Lần đầu tiên của thuật toán tạo ra một hàm băm bên trong được lấy từ thông báo và khóa bên trong.

Bước thứ hai tạo ra mã HMAC cuối cùng bắt nguồn từ kết quả băm bên trong và khóa bên ngoài. HMAC được mô tả bằng cách sử dụng phương trình sau:

$$HMAC(K, M) = H((K^{+} \oplus opad) || H((K^{+} \oplus ipad) || M))$$

Thuật toán được thể hiện như sau:



Trong đó:

H: Là hàm băm nhúng (như MD5, SHA-1, ...).

b: số bits trong một khối.

n: độ dài của mã băm.

M: Là thông điệp đầu vào của HMAC

K: Là khóa bí mật

K^+: Là khóa đệm mở rộng của K; hoặc bằng cách đệm bên phải với 0 cho đến kích thước khối b hoặc bằng cách băm xuống nhỏ hơn hoặc bằng đến kích thước khối trước và sau đó đệm bên phải với số không

opad= 0x36, được lặp lại b/8 lần.

ipad= 0x5C, được lặp lại b/8 lần.

||: là phép nối

Tóm tắt giải thuật

1. Chọn K

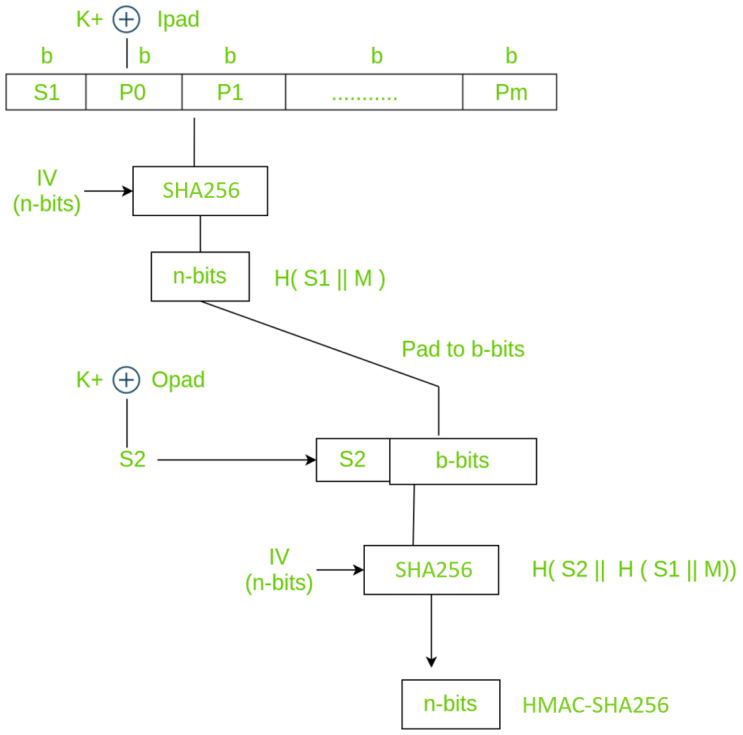
   Nếu K < b, thêm 0 ở bên trái cho đến khi k=b. K nằm giữa 0 và b (0 <K <b)
2. XOR K+ với ipad tương đương với b bit tạo ra bit S1.
3. Nối S1 với văn bản thuần túy M
4. Áp dụng hàm băm H trên (S1 || M)
5. Pad n-bit cho đến khi độ dài bằng b-bit
6. XOR K+ với opad tương đương với b bit tạo ra bit S2.
7. Nối S2 với đầu ra của bước 5.
8. Áp dụng hàm băm H ở bước 7 để xuất mã băm n-bit.

3. HMAC-SHA256

HMAC-SHA256 là một loại thuật toán băm có khóa được xây dựng từ hàm băm SHA256 và được sử dụng làm mã xác thực thông điệp băm dựa trên HMAC hoặc SHA256. HMAC-SHA256 chấp nhận các khóa có kích thước bất kỳ và tạo chuỗi băm dài 256 bit.

Công thức:

$$\text{HMAC-SHA256}(K, M) = H((K^+ \oplus \text{opad}) \,||\, H((K^+ \oplus \text{ipad}) \,||\, M))$$



Trong đó:

H: Là hàm băm nhúng (như MD5, SHA-1, ...).

b: số bits trong một khối.

n: độ dài của mã băm.

M: Là thông điệp đầu vào của HMAC

K: Là khóa bí mật

K^+: Là khóa đệm mở rộng của K; hoặc bằng cách đệm bên phải với 0 cho đến kích thước khối b hoặc bằng cách băm xuống nhỏ hơn hoặc bằng đến kích thước khối trước và sau đó đệm bên phải với số không

opad: Là phần đệm bên ngoài (0x36, lặp lại nếu cần thiết)

ipad: Là phần đệm bên trong (0x5C, lặp lại nếu cần thiết)

||: là phép nối

Tóm tắt giải thuật

1. Chọn K

   Nếu K < b, thêm 0 ở bên trái cho đến khi k=b. K nằm giữa 0 và b (0 <K <b)
2. XOR K+ với ipad tương đương với b bit tạo ra bit S1.

3. Nối S1 với văn bản thuần túy M
4. Áp dụng hàm băm SHA256 trên (S1 || M)
5. Pad n-bit cho đến khi độ dài bằng b-bit
6. XOR K+ với opad tương đương với b bit tạo ra bit S2.
7. Nối S2 với đầu ra của bước 5.
8. Áp dụng hàm băm SHA256 ở bước 7 để xuất mã băm n-bit.

4. OTP và HOTP

OTP (One-Time Password) là mật khẩu sử dụng một lần. Đây là một phương pháp xác thực bảo mật thông tin trong đó người dùng sẽ được cung cấp một mã OTP duy nhất để xác thực việc truy cập vào một tài khoản hoặc thực hiện một giao dịch cụ thể. Mã OTP này sẽ chỉ có thể sử dụng được một lần duy nhất và sẽ hết hạn sau một khoảng thời gian ngắn, từ đó nâng cao tính bảo mật cho quá trình xác thực. Phương pháp sử dụng OTP ngày nay được áp dụng rộng rãi trong các ứng dụng tài khoản trực tuyến, giao dịch ngân hàng, mạng xã hội và các dịch vụ trực tuyến khác.

HOTP (HMAC-based one-time password) được ra bởi tổ chức Initiative for Open Authentication (OATH), là thuật toán mật khẩu dùng một lần dựa trên cơ chế xác thực thông điệp bằng hàm băm HMAC (Hash-based Message Authentication Code) và hàm băm SHA-1 (Secure Hash Algorithm 1). Đối với giá trị HOTP, nó sử dụng thuật toán HMAC-SHA-1 chấp nhận một tập dữ liệu lớn tùy ý và trả về một giá trị có độ dài nhất định là 160 bit làm giá trị đầu ra:

$$HOTP(K,C) = Truncate(HMAC\_SHA\text{-}1(K,C))$$

Trong công thức :

- K: Là giá trị chia sẻ bí mật giữa Client và Server.

- C: Là bộ đếm đã được đồng bộ giữa Client và Server, C có độ dài 8 bytes.
- Truncate( ): Là hàm tách chuỗi, thực hiện việc trích xuất kết quả từ hàm Hash để có được mật khẩu OTP.

Kết quả đầu ra của hàm HMAC_SHA-1(K,C) cho ta một giá trị có độ dài là 160 bits = 20 bytes, chúng ta dùng hàm tách chuỗi (Truncate) để tách từ chuỗi 160 bits thành một chuỗi mới có độ dài 32 bit, sau đó tính modulo để được mật khẩu OTP. Cụ thể như sau:

Từ kết quả đầu ra 160 bit của hàm HMAC_SHA-1(K,C), ta lấy 4 bit thấp của byte cuối cùng chuyển sang cơ số 10 để tìm vị trí offset, sau đó ta được chuỗi 4 bytes = 32 bit tính từ vị trí offset.

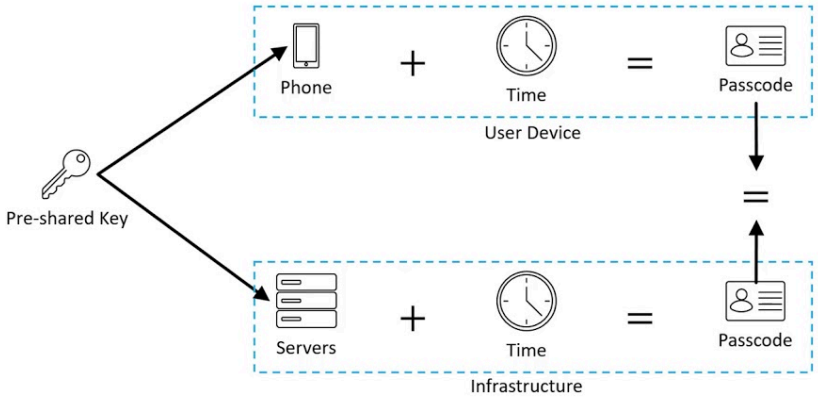
Giá trị mật khẩu được tính theo công thức sau:

$$HOTPvalue = HOTP(K,C) \bmod 10^d$$

Trong công thức : d là số chữ số của OTP, thông thường một mật khẩu OTP sinh ra có độ dài từ 6 đến 8 chữ số (ví dụ là: 123456,…).

5. TOTP

TOTP (Time-Based One-Time Password) là một thuật toán sử dụng để tạo ra mã xác thực một lần (OTP) dựa trên thời gian hiện tại và một khóa bí mật chia sẻ giữa người dùng và hệ thống. TOTP là một phiên bản mở rộng của HOTP (HMAC-based One-time Password).



a. Công thức

Bằng việc thay thế bộ đếm C của HOTP bằng thời gian thực T, TOTP đã giảm thời gian tồn tại của OTP từ đó nâng cao tính bảo mật của cho các đối tượng cần được bảo vệ.

TOTP = HOTP(K, T)

Giá trị thời gian thực (T) được tính như sau:

$$T = Floor(\frac{T_{current-unix-time} - T_0}{X})$$

Trong đó:

$T_{current-unix-time}$: giá trị thời gian hiện tại và được tính theo thời gian Unix (được tính từ thời điểm của Unix Epoch là ngày 01/01/1970 theo UTC - giờ chuẩn quốc tế).

T_0: giá trị thời gian ban đầu (thông thường sẽ chọn giá trị 0).

X: bước thời gian, tham số này được coi là yếu tố quyết định thời gian hợp lệ của mật khẩu OTP.

T: kết quả phép tính (lấy phần nguyên bằng cách dùng hàm floor để làm tròn dưới) và là giá trị cần tìm.

X: thời gian tồn tại của mật khẩu. *VD: X= 30 nghĩa là OTP tồn tại trong 30s*

Độ dài mật khẩu của thuật toán TOTP được tính như sau:

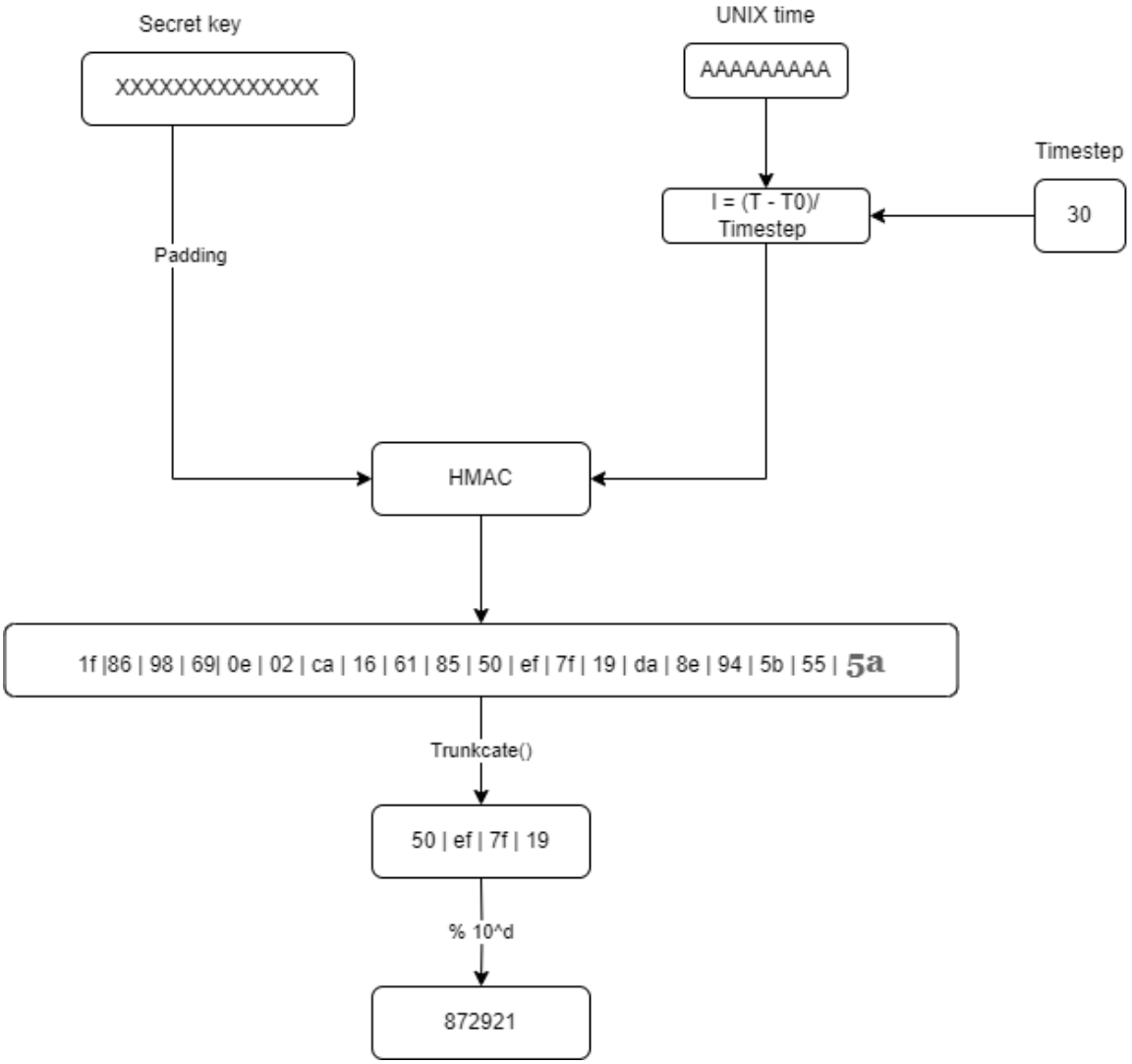
$$TOTPvalue = TOTP(K, T) \bmod 10^N$$

Trong đó:

N: là số chữ số của OTP, thông thường một mật khẩu OTP sinh ra có độ dài từ 6 đến 8 chữ số (ví dụ là: 123456,…).

Với thuật toán TOTP thì thời gian chuẩn hợp lệ của mật khẩu OTP là 30 giây (*X*=30), thời gian có hiệu lực của mỗi lần sử dụng mật khẩu OTP là 30 giây được chọn phù hợp với yêu cầu về bảo mật và khả năng sử dụng.

b. Quá trình tạo mã TOTP



Giả sử chúng ta có một secret key là "JBSWY3DPEHPK3PXP", độ dài mật khẩu là 6, thời gian dời (timestep) là 30 giây và đồng hồ của máy tính được đồng bộ hóa với server.

Tính số giây kể từ thời điểm UNIX epoch (tức là 1970-01-01 00:00:00 UTC) đến thời điểm hiện tại, gọi là T. Ví dụ, giả sử T = 1620892213.

Chia T cho timestep để tính toán số vòng lặp I. Trong trường hợp này, I = T /30 = 54029740.

Chuyển đổi I thành một chuỗi byte 8 byte (64 bit) bằng cách sử dụng big-endian byte ordering. Trong trường hợp này, I được chuyển đổi thành chuỗi byte:

00 00 00 00 03 3E C7 8C

Sử dụng thuật toán HMAC-SHA-1 để tính toán mã băm (hash) của I với secret key. Đầu vào cho HMAC-SHA-1 là secret key và chuỗi byte được chuyển đổi ở bước trước đó. Kết quả là một chuỗi byte 20 byte (160 bit), gọi là HMAC.

HMAC = 0x1D 3C 50 60 6A 35 0D 4E 66 E2 0D 7C 37 AA 8D B9 3E 43 1B 36

Lấy 4 byte cuối cùng (32 bit) của HMAC và chuyển đổi thành số nguyên không dấu IOTP. Để chuyển đổi thành số nguyên, ta sử dụng big-endian byte ordering. Trong trường hợp này, ta lấy 4 byte cuối cùng của HMAC là:

0x8D B9 3E 43

Khi được chuyển đổi sang số nguyên không dấu bằng cách sử dụng big-endian byte ordering, ta có:

IOTP = 2,383,492,035

Chia IOTP cho 10^6 và lấy phần dư để chỉ lấy 6 chữ số cuối cùng của IOTP. Kết quả là mã OTP:

OTP = IOTP % 10^6 = 492,035

Để đảm bảo mã OTP có độ dài là 6 chữ số, ta có thể bổ sung các số 0 vào đầu của OTP nếu cần thiết. Trong trường hợp này, mã OTP cuối cùng là 492035.

Vậy, với secret key là "JBSWY3DPEHPK3PXP", chúng ta đã tính toán được mã OTP tại thời điểm hiện tại là 492035.

III. Kết quả nghiên cứu

- Tìm hiểu thuật toán được sử dụng trong TOTP
- Xây dựng được chương trình mô phỏng lại thuật toán, sinh ra mã OTP để tăng tính xác thực

IV. Triển khai

1. Kịch bản ứng dụng

Xây dựng 1 local server và 1 ứng dụng cho client trên window. Khi client nhập mật khẩu để xác thực yêu cầu chuyển tiền, ứng dụng sẽ tự động sinh ra một mã OTP. Sau đó ứng dụng sẽ gửi mã OTP này cùng với mật khẩu đến server để xác thực client.

2. Chương trình demo

V. Tài liệu tham khảo

1. Balasta, D. U., Pelito, S. M. C., Blanco, M. C. R., Alipio, A. J., Mata, K. E., & Cortez, D. M. A. Enhancement of Time-Based One-Time Password for 2-Factor Authentication.
2. Seta, H., Wati, T., & Kusuma, I. C. (2019). Implement Time Based One Time Password and Secure Hash Algorithm 1 for Security of Website Login Authentication. 2019 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS).
3. Lumburovska, Lina & Dobreva, Jovana & Dimitrova, Vesna & Mihajloska, Hristina & Andonov, Stefan. (2021). A Comparative Analysis of HOTP and TOTP Authentication Algorithms. Which one to choose?. 5. 131-136.