1. **Lý thuyết**

**Two-Factor Authentication**(thường viết tắt là ****2FA**** hoặc ****TFA****) là một phương pháp xác thực người dùng dựa trên 2 yếu tố, một là ****mật khẩu**** (thứ phổ biến nhất) và thứ hai là một thứ mà người dùng sở hữu, có quyền truy cập đến, ví dụ như ****dấu vân tay****, ****tin nhắn SMS****, ****gửi mã token tới Email**** hoặc tốt hơn nữa là****One-time Password (OTP)****(*mật khẩu một lần có giới hạn hiệu lực theo thời gian*).

**One-Time Password** sẽ là thứ mà chúng ta bàn đến trong ngày hôm nay, ****SMS**** và ****Email**** thì mình sẽ làm ở những bài riêng biệt khác sau. Và đúng như cái tên của nó, ****One-time Password**** hay còn được viết tắt là ****OTP**** là một loại mã ****token**** mà chỉ có thể được sử dụng một lần rồi sau đó nó sẽ bị hủy, không được phép sử dụng tới lần thứ hai.

Lúc này, câu hỏi tiếp theo mà chúng ta đặt ra ở đây là, làm sao để đảm bảo chúng ta có thể tạo một ****mã token mà thỏa mãn điều kiện**** ****là duy nhất (unique)?****  
Và câu trả lời sẽ tiếp tục dẫn dắt chúng ta đến với một khái niệm khác đó là: **[Thuật toán HMAC-based One-Time Password, hay còn được viết tắt là HOTP.](https://en.wikipedia.org/wiki/HMAC-based_One-time_Password_algorithm" \t "/home/sonnguyen/Documents\\x/_blank)**

****HMAC-based One-Time Password**** – ****HOTP**** là một thuật toán sinh mã OTP dựa trên hàm băm ****HMAC\_SHA-1****, nó sử dụng 2 thành phần: thứ nhất là một ****Chuỗi Secret cố định****, còn thành phần thứ hai là ****một bộ đếm (Counter)**** bộ đếm này dùng một cái là ****“Moving-Factor”**** (mình tạm dịch ra là ****một yếu tố di chuyển****, các bạn cũng có thể coi nó tương tự một chuỗi random ngẫu nhiên cho dễ hiểu cũng được.)

– Tiếp theo, vì kết quả output của hàm băm ****HMAC\_SHA-1**** ở trên là một giá trị có ****độ dài 160 bit = 20 bytes**** nên chúng ta sẽ ****cần thêm một bước làm ngắn gọn output**** để mắt người dùng có thể dễ dàng đọc được. Việc cắt ngắn này sẽ dẫn chúng ta đến với **[thuật toán TOTP – Time-based One-time Password algorithm](https://en.wikipedia.org/wiki/Time-based_One-time_Password_algorithm" \t "/home/sonnguyen/Documents\\x/_blank)** để tạo ra chuỗi có độ dài như chúng ta mong muốn, ví dụ "170 795" mà các bạn thường thấy ở app **Google Authenticator** hoặc **Authy**.

**Time-based One-time Password – TOTP** về cơ bản chỉ khác ****HOTP**** ở chỗ là ****TOTP sẽ sử dụng “thời gian” (Time) để làm bộ đếm**** (****Counter****) thay vì ****“Moving Factor”**** như ****HOTP****. Chính vì việc sử dụng ****counter**** là ****thời gian****nên phía ****Server**** lẫn ****Client**** khi đã có chung ****Secret Key**** rồi thì ****không cần có sự tương tác qua lại nữa****. Vì cả 2 phía đều có quyền truy cập vào ****thời gian****. Điều này cũng trả lời luôn cho ****một thắc mắc khá thú vị**** mà lâu nay mình vẫn tự hỏi, đó là tại sao khi mình thử tắt mạng, ****không kết nối**** ****internet**** cho cái điện thoại vậy mà ****Token**** sinh ra của mấy cái app ****Google**** ****Authenticator**** hay ****Authy**** vẫn sử dụng được ngon lành chả vấn đề gì =))

****Giải thích cụ thể hơn cho các bạn**** đó là: phía ****Server**** sẽ so sánh ****giá trị token mà người dùng submit từ phía client lên**** với ****tất cả các token được sinh ra trong cùng một khoảng thời gian nhất định trên Server****. *(thường là 30 giây cho đến 1 phút)*, và dĩ nhiên là nếu trùng nhau thì bạn sẽ pass qua vòng xác thực 2 lớp này. Đọc đến đây nhiều bạn có thể sẽ thắc mắc tiếp là: Ủa thế ****server và client khác múi giờ (Time zone****) thì làm sao mà khoảng thời gian của 2 phía có thể đồng nhất được nhỉ?

Giải pháp là chúng ta có thể ****convert**** thời gian của cả 2 phía về dạng ****Unix Timestamp**** (hay còn gọi với tên khác là ****Epoch Time****) rồi so sánh chúng. Hiểu một cách đơn giản thì ****Unix Timestamp**** là số giây đếm tăng dần từ một điểm thời gian cố định trong quá khứ đó là ngày ****01/01/1970 (UTC) 00:00:00****  
Mình sẽ để link tham khảo về Unix Time cho các bạn tìm hiểu thêm ở đây **[https://www.unixtimestamp.com/](https://www.unixtimestamp.com/" \t "/home/sonnguyen/Documents\\x/_blank)**

1. **Phân tích và triển khai code ứng dụng 2FA**

