

Macedo Madrigal Rodrigo

Temas Selectos de Computación

No. Cuenta: 314676797

Generador de Tokens.

Nota: Se entrega la documentación del proyecto, dentro del cual se indica la ubicación del código dentro de un repositorio de Github: <https://github.com/Macmaad/RSA-SecureID>. El código no se manda debido a la limitación que tiene la plataforma en la cantidad de archivos.

Investigación:

RSA SecureID es un mecanismo de autenticación de usuarios que consiste en un generador de códigos que tienen una vida útil de cierto tiempo, por lo general es de 60 segundos. Este generador de tokens puede ser digital o físico.

Con este generador se puede agregar una capa mas de seguridad a los usuarios, siendo este mecanismo difícil de descifrar, pero no imposible.

Algoritmo RSA:

“Se eligen dos números primos, por ejemplo, **p=3** y **q=11**”

Calcula **$n=p*q$** , en este caso, $n=3*11=33$

Calcula **$z=(p-1)*(q-1)$** , en nuestro caso: $z=(3-1)*(11-1)=20$

A esta función se la denomina Función Phi o Fi de Euler.

Elige un número primo k, tal que k sea **co-primo** a z, por ejemplo, z no es divisible por k.

Tenemos varias opciones aquí, valores de k como pueden ser 7, 11, 13, 17 o 19 son

válidos. 5 es primo, pero no es co-primo de k puesto que 20 (z) es divisible por 5.

Elegimos $k=7$ para simplificar los cálculos con un número pequeño.

La **clave pública** va a ser el conjunto de los números (n,k) , es decir, $(33,7)$.

Ahora se calcula la clave privada. Para ello, se elige un **número j** que verifique la siguiente ecuación:

$k*j$ congruente con 1 (mod z)

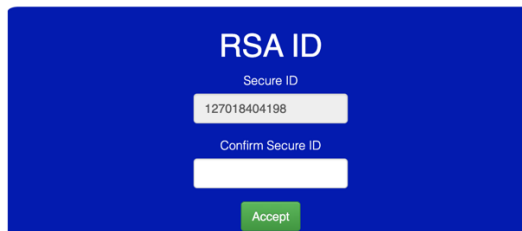
Diseño de solución.

Para crear este generador de tokens aleatorios bajo un mismo PIN de seguridad, se utilizará el algoritmo de encriptación RSA. Una vez logrando encriptar el mismo valor con diferentes resultados cada cierto tiempo, solamente se creará una aplicación web como lo mínimo utilizado para usar como método de autenticación.

Prototipo.

Página para autenticar al usuario, se genera los tokens en el servidor.

Secure ID

A blue rectangular interface for RSA ID authentication. At the top, it says "RSA ID" in white. Below that, "Secure ID" is written in small white text. There is a white input field containing the number "127018404198". Below this field, the text "Confirm Secure ID" is written in small white text. At the bottom, there is a green button with the word "Accept" in white.

El token cambia cada 60 segundos y el anterior deja de ser válido.

Secure ID

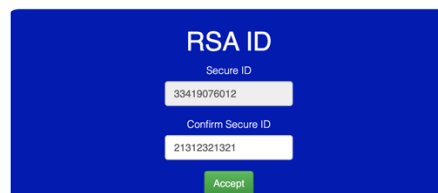


A blue rectangular login screen for RSA ID. At the top, it says "RSA ID" in white. Below that, "Secure ID" is written in smaller white text. There are two white input fields: the first contains the number "23642789114" and is labeled "Secure ID" above it; the second is empty and labeled "Confirm Secure ID" above it. At the bottom center is a green button with the word "Accept" in white.

Si se ingresa un token equivocado o no válido, se mostrará el mensaje de error.

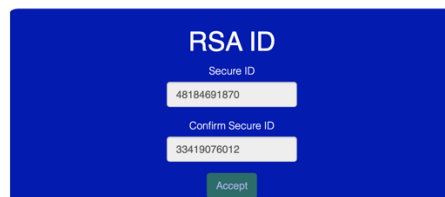
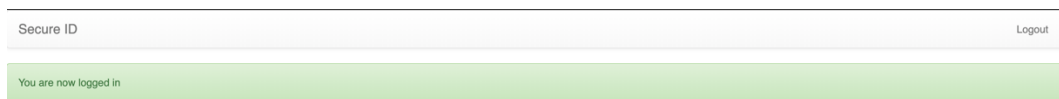
Secure ID

Error in the pin

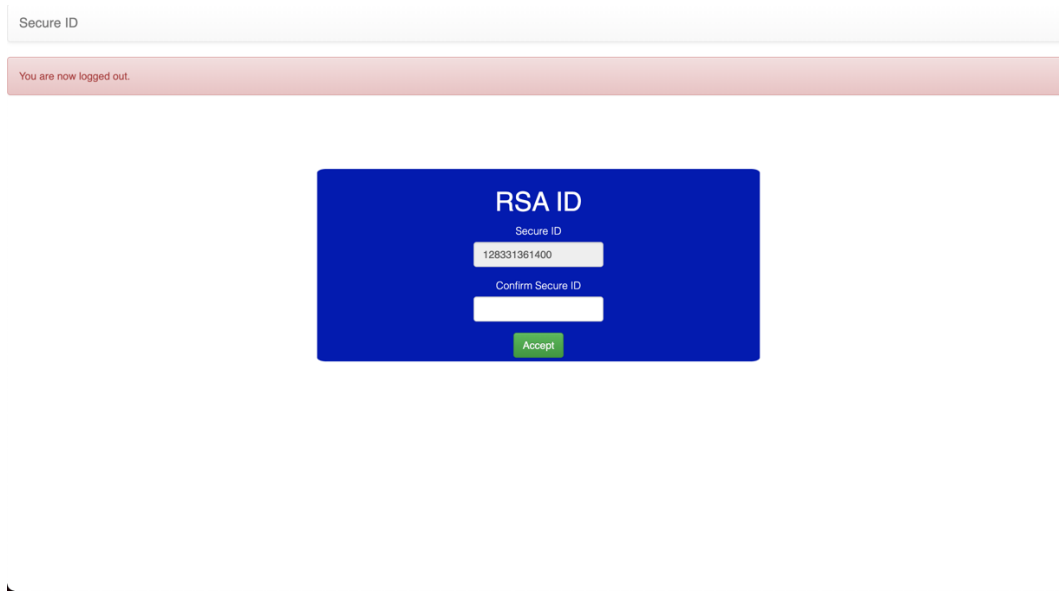


A blue rectangular login screen for RSA ID, identical to the one above. It shows "RSA ID" and "Secure ID" at the top. The first white input field contains "33419076012" and is labeled "Secure ID". The second white input field contains "21312321321" and is labeled "Confirm Secure ID". A green "Accept" button is at the bottom. This screen is part of a sequence showing an error state.

Si el token es válido mostrará el mensaje que hemos ingresado y nos abrirá la opción de hacer logout en el navbar. En esta pantalla no podremos mandar tokens ni escribirlos, será necesario hacer logout.



Al hacer click en el botón de logout, nos regresará a la pantalla y ya podremos volver a autenticarnos.



Documentación de uso y diseño.

El código de la aplicación corre usando Python como código del lado del servidor, un poco de HTML y CSS para general la aplicación en el navegador web y JS para el lado lo que se necesita en el lado del cliente.

En Python hacemos uso de Flask, el cual es una librería para hacer micro aplicaciones web creando una API con 2 verbos.

1. GET: Sirve para la generación de tokens y poder desplegarlos en el navegador.
2. POST: Valida el token ingresado con el PIN que se encuentra hardcoado en el código.

El código se encuentra en esta liga: <https://github.com/Macmaad/RSA-SecureID>.

Para poder hacer uso de la aplicación es necesario tener Python instalado y los siguientes requerimientos: flask versión 1.1.2 y flask-cors versión 3.0.9 .

Una vez teniendo eso, necesitamos tener las variables de entorno en su lugar con los siguientes comandos: export FLASK_APP=app/app.py y export

FLASK_ENV=development. Una vez que todo esto esta corriendo podemos correr la aplicación de la siguiente forma Python app/app.py y acceder a nuestro navegador en la siguiente dirección <http://127.0.0.1:5000/>

El archivo app/final_rsa.py guarda el algoritmo RSA y tiene todo lo necesario para la generación de tokens.

La aplicación por el momento solo se usa con un único PIN (4567) pero puede ser modificada para obtener PINS de forma dinámica y trabajar con cualquier número.

Bibliography

Wikiwand. (n.d.). Retrieved from https://www.wikiwand.com/en/RSA_SecurID

Wikipedia. (n.d.). Retrieved from

[https://simple.wikipedia.org/wiki/RSA_algorithm#:~:text=RSA%20\(Rivest%E2%80%93Shamir%E2%80%93Adleman,can%20be%20given%20to%20anyone.](https://simple.wikipedia.org/wiki/RSA_algorithm#:~:text=RSA%20(Rivest%E2%80%93Shamir%E2%80%93Adleman,can%20be%20given%20to%20anyone.)

Wikipedia. (n.d.). Retrieved from https://es.wikipedia.org/wiki/RSA#Algoritmo_RSA

Wikipedia. (n.d.). Retrieved from

https://en.wikipedia.org/wiki/Miller%E2%80%93Rabin_primality_test

Geekdforgeeks. (n.d.). Retrieved from <https://www.geeksforgeeks.org/rsa-algorithm-cryptography/>

Wikipedia. (n.d.). Retrieved from

https://en.wikipedia.org/wiki/Primality_test#Fermat_primality_test

Wikipedia. (n.d.). Retrieved from

https://es.wikipedia.org/wiki/Ecuaci%C3%B3n_diof%C3%A1ntica