Ayudantía 6

IIC3253 - Criptografía y Seguridad Computacional

Christian Klempau

1 HMAC

1.1 JWT

JSON Web Token (JWT) es un estándar del Internet, utilizado para compartir información y autenticar usuarios, de forma segura. Se construye de la siguiente manera:

Header:

```
{
    "alg": "HS256",
    "typ": "JWT"
}
```

Body:

```
{
  "lat": " -19.949156",
  "lng": "-69.633842",
  "name": "Chris Klempau",
  "url_base64": "aHR0cHM6Ly93d3cueW91dHViZS5jb2
                 0vd2F0Y2g/dj1kUXc0dzlXZ1hjUQ=="
}
```

Signature:

$$Message = base64(Header) + \text{"."} + base64(Body)$$

$$Signature = \text{HMACSHA256}(Message, Secret)$$

Y el JWT final es:

$$Header_{b64}.Body_{b64}.Signature_{b64}$$

Por ejemplo, algo del estilo:


```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```


¿Qué hace el receptor? Verifica que la firma (*signature*) del JWT sea efectivamente: $Signature_{JWT} = \text{HMACSHA256}(Message_{JWT}, Secret)$. En ese caso, puede asumir que el mensaje fue construido por alguien con acceso a la llave $Secret$ y se considera válido.

En el caso de un atacante, al no tener acceso a la llave *Secret*, no puede construir un JWT válido.

Puedes probar JWTs en: `https://jwt.io/`

2 Teoría de números

2.1 Máximo común divisor

1. Defina un algoritmo para calcular el máximo común divisor de dos números enteros positivos a y b, dado que:

$$MCD(a,b) = \begin{cases} a & \text{si } b = 0 \\ MCD(b, a \bmod b) & \text{si } b > 0 \end{cases}$$

2. ¿Cuál es la complejidad temporal (peor caso) del algoritmo?

2.2 Algoritmo extendido de Euclides

Dado el algoritmo extendido de Euclides:

```
# input:    dos números a y b
# output 1:       ?
# output 2 y 3:   ?

ExtendedGCD(a, b):
    r_old, r := a, b
    s_old, s := 1, 0
    t_old, t := 0, 1

    while r != 0 do
        q  := old_r // r

        s_old, s := s, s_old - q * s
        t_old, t := t, t_old - q * t
        r_old, r := r, s * a + t * b
    end

    return r_old, s_old, t_old
```

Nota: recuerde la identidad de Bézout:

$$MCD(a,b) = s \cdot a + t \cdot b$$

1. ¿Qué significan los valores retornados?

   Output 1: $r\_old = GCD(a,b)$

Output 2: s_old = coeficiente de a en la identidad de Bézout

Output 3: t_old = coeficiente de b en la identidad de Bézout

2. ¿Cuál es el resultado de ejecutar el algoritmo, con $a = 240$ y $b = 46$?

2.3 Invertibilidad dado GCD

Demuestre el siguiente teorema:

Un número a es invertible en módulo n si y solo si $GCD(a, n) = 1$.

2.4 Desafío: demostración alternativa complejidad GCD

Demuestre que el algoritmo de Euclides tiene complejidad temporal $O(\log b)$.

Nota:

Teorema de Lamé: La cantidad de pasos que GCD debe realizar es exactamente n, donde $a > b > 0$, tal que:

$$a = F_{n+2}$$

$$b = F_{n+1}$$

Donde F_k es el k-ésimo número de Fibonacci.

Fórmula de Binet:

$$F_k = \frac{\phi^k - \varphi^k}{\sqrt{5}}$$

donde ϕ es el *golden ratio*:

$$\phi = \frac{1 + \sqrt{5}}{2}$$

$$\varphi = \frac{1 - \sqrt{5}}{2} \approx -0.61$$