Verificación de una firma con curvas elípticas

José Luis Ruiz

Departament de Matemàtiques
Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

2022

Datos de la curva

- p: número primo.
- E: $Y^2 = X^3 - 3x + b \pmod{p}$.
- P: punto de la curva E (coordenadas en $\mathbb{Z}_p$).
- q: orden del punto P; es decir: q es el mínimo entero positivo tal que $q \cdot P = \mathcal{O}$, donde $\mathcal{O} \in E(\mathbb{Z}_p)$ es el elemento neutro.

Clave privada y clave pública

- Clave privada: r, entero módulo q.
- Clave pública: $Q = (Q_x, Q_y) = r \cdot P \in E(\mathbb{Z}_p)$.

Firma de un mensaje m (o de su *hash*)

- Escoger un entero aleatorio: $k \in \{2, \ldots, q-2\}$.
- Calcular: $k \cdot P = (x_1, y_1)$.
- La firma es: (f_1, f_2), donde:

$$f_1 = x_1 \pmod{q}$$
$$f_2 = k^{-1} \cdot (m + f_1 \cdot r) \pmod{q}$$

Verificación de una firma (f_1, f_2)

- Calcular (w_1, w_2), donde:

$$w_1 = m \cdot f_2^{-1} \pmod{q}$$
$$w_2 = f_1 \cdot f_2^{-1} \pmod{q}$$

- Calcular el punto de E: $(x_0, y_0) = w_1 \cdot P + w_2 \cdot Q$.
- Aceptar la firma si y solo si: $x_0 = f_1 \pmod{q}$.

Conexión TSL 1.3 y captura de datos

Conexión a `wikipedia.org` con protocolo TSL 1.3 y captura de datos con Wireshark.

Hay que averiguar:

1. Datos de la curva.
2. Clave pública $Q = (Q_x, Q_y)$ (punto de la curva).
3. Firma (f_1, f_2).
4. Mensaje m que se ha firmado.

Protocolo TSL 1.3 (Handshake)

- `Client Hello` (en claro).
- `Server Hello` (en claro y cifrado).
- Resto de intercambios cifrados.
- En "`Hello`" se establece un secreto común (Diffie-Hellman con curvas elípticas).
- También se establece protocolo de cifrado, por ejemplo `TLS_AES_128_CGM_SHA256`.
- Para poder leer los certificados y otros datos, Wireshark necesita acceder a las claves entre el Client (Firefox) y el Server (Wikipedia).
- Firefox puede guardar las claves en un fichero y Wireshark puede acceder a él.

Cómo hacer que Wireshark lea las claves de Firefox

1. En una ventana de comandos:

   ```
   > export SSLKEYLOGFILE = ‘‘./keys.txt’’
   > firefox &
   ```

2. Abrir Wireshark.
3. En Wireshark:

   `Preferences -> Protocols -> TLS -> (Pre)-Master-Secret log filename`

   e introducir ruta y nombre de keys.txt
4. Wireshark: empezar a capturar.
5. Firefox: conectarse a `wikipedia.org`
6. Wireshark: dejar de capturar.

Paquetes de datos a explorar

Establecer conexión con `wikipedia.org` y buscar los paquetes de datos:

- `Client Hello` (conexión ejemplo: 246).
- `Server Hello` (conexión ejemplo: 248).
- `Certificate...` (conexión ejemplo: 250 y 251. Pueden estar en el mismo paquete).

Observación: a veces Wireshark intercambia el orden de los paquetes. Para leerlos en orden:

```
Preferences -> Protocols -> TCP
    -> Reassemble out-of-order segments
```

Clave pública $Q = (Q_x, Q_y)$

En el paquete `Certificate`:

- TSLv1.3 Record Layer: Handshake Protocol Certificate
- Handshake Protocol: Certificate
- Certificates (hay varios)
- Certificate: ...wikimedia...
- signedCertificate
- subjectPublicKeyInfo
- SubjectPublicKey:
- $04de...5a17...b3$

$04de\ldots5a17\ldots b3$ quiere decir:

- 04: siguen dos enteros en hexadecimal en formato no comprimido.
- $Q_x = 0xde\ldots5a$ (32 bytes).
- $Q_y = 0x17\ldots b3$ (32 bytes).

Datos de la curva (Certificate Verify, 251)

- TSLv1.3 Record Layer: Handshake Protocol: Certificate Verify
- Handshake Protocol: Certificate Verify
- Signature Algorithm: edcsa_secp256r1_sha256
- Signature: 3046...*b*4

- Curva elíptica: p256 (consultar documento NIST).
- La firma está en la secuencia de bytes 3046...*b*4 (en notación ASN.1)

Firma en notación ASN.1 (*Abstract Syntax Notation*):

- 30: secuencia de objetos.
- 46: longitud (en hexadecimal) en bytes de la secuencia.
- 02: siguiente objeto es un entero.
- 21: longitud (en hexadecimal) en bytes del entero.
- f_1 es este entero: $f_1 = 0x00 \ldots 66$.
- 02: siguiente objeto es un entero.
- 21: longitud (en hexadecimal) en bytes del entero.
- f_2 es este entero: $f_2 = 0x00 \ldots b4$.

Mensaje que se ha firmado

- El mensaje tiene dos partes:

  preámbulo + mensaje de la conexión

- El *preámbulo* es la concatenación de:
  - **64** veces el byte ‘20’
  - cadena de texto ‘TLS 1.3, server CertificateVerify’ (en ASCII)
  - el byte ‘00’

Mensaje de la conexión

Extraer las 4 partes siguientes de la conexión en ficheros binarios (raw binary):

1. Handshake Protocol: `Client Hello` → `1.bin`
2. Handshake Protocol: `Server Hello` → `2.bin`
3. Handshake Protocol: `Encrypted Extensions` → `3.bin`
4. Handshake Protocol: `Certificate` → `4.bin`

- Concatenar estos ficheros:

  `cat 1.bin 2.bin 3.bin 4.bin > mensaje.bin`

- Calcular el *hash* de este fichero (hexadecimal) con el tipo de *hash* indicado en el `Sever Hello CipherSuite`:

  `mensaje384 = sha384(mensaje.bin)`

- Finalmente, el mensaje firmado m es sha256 (viene de edcsa_secp256r1_sha256) de la concatenación del preámbulo y mensaje384:

  m = `sha256(preámbulo + mensaje384)`