CURVAS ELÍPTICAS EN CRIPTOGRAFÍA Y SU APLICACIÓN EN BITCOIN

Mauricio Odreman Vera
Moisés Sánchez Adame
Adrián Chouza Delgado

Introducción

- La criptografía estudia las formas en que se puede transformar un texto legible a uno ilegible y viceversa.

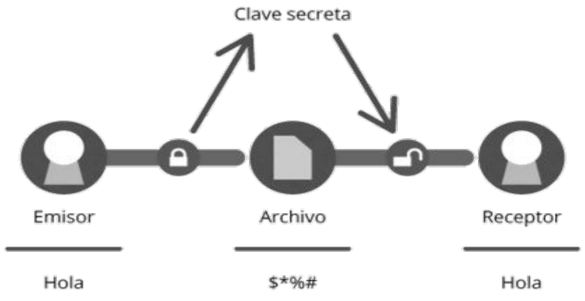

Figura 1. Representación de un sistema criptográfico [3]

- Entre las técnicas más utilizadas se encuentran DSA, RSA, AES y Diffie-Hellman, donde la mayoría de estos métodos tienen una implementación con curvas elípticas.
- Una curva elíptica es de la forma $y^2=x^3+Ax+B$, y posee sus propia definición para la suma de dos puntos y multiplicación (suma continua de puntos).

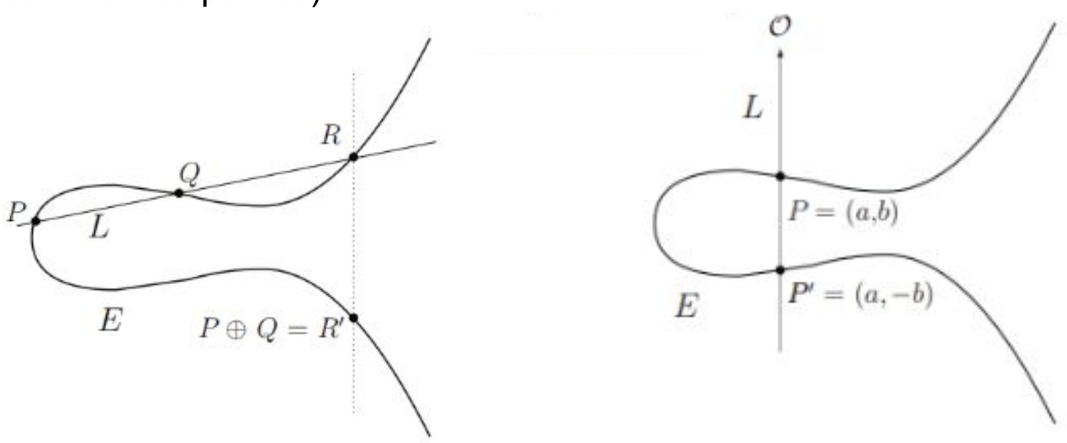

Figura 2. Adición de curvas elípticas con enfoque gráfico. [4]

Objetivo

- Realizar el estudio matemático-teórico sobre criptografía y curvas elípticas
- Conocer los principales algoritmos y protocolos basados en curvas elípticas.
- Comprender cómo funcionan las monedas digitales bitcoins mediante las curvas elípticas

Metodología

- Evaluación de algoritmos de encriptado empleando curvas elípticas.
- Aplicación de curvas elípticas en Bitcoin.
- Implementación de algoritmos criptográficos en Python para ser comparados a partir de distintas métricas como tiempo de ejecución y memoria.

Resultados

ECDSA
Alice quiere firmar un mensaje *m*, por lo que utiliza los valores públicos (*C,G,n*) donde *C* es una curva, *G* un punto y *n* el orden de este punto. Ella cuenta con d_A (llave privada) y $Q_A=d_A \bullet G$ (llave pública). El procedimiento es el que sigue para firmar y verificar es:

1. $e = HASH(m)$
2. $k = random(1,n-1)$
3. $(x_1,y_1) = k \bullet G$
4. $r = x_1 \bmod n$
5. $s = k^{-1}(e + rd_A) \bmod n$
6. La firma es (r,s)

1. $e=HASH(m)$
2. $w=s^{-1} \bmod n$
3. $u_1=ew \bmod n$
4. $u_2=rw \bmod n$
5. $(x_1,y_1) = u_1G + u_2Q_A$
6. Si $r=x_1$, la firma es válida

Cualquier otra persona puede verificar (sin necesidad de conocer d_A que un documento firmado con (r,s) pertenece a Alice.

En la siguiente tabla se muestran los resultados del programa creado con la librería cryptography de Python.

Comparación entre DSA y ECDSA (SECP384R1)		
Algoritmo	ECDSA	DSA
Tiempo de ejecución (seg)	0.10498	0.40548
Tamaño de llave (bits)	384	2048

Aplicación en Bitcoin

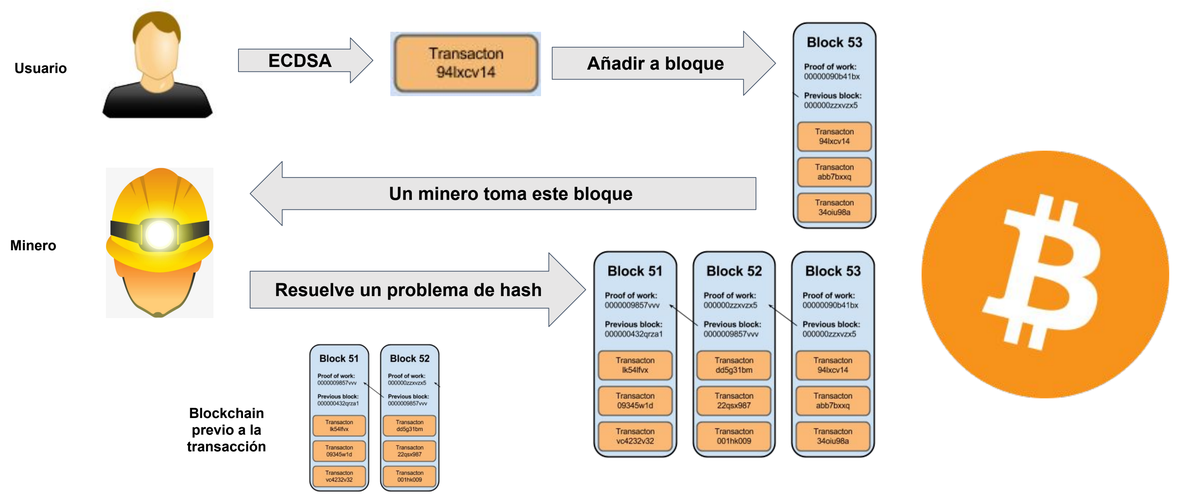

Fig 3. Ciclo de vida de una transacción en Bitcoin.

Conclusiones

El uso de curvas elípticas en algoritmos criptográficos mejora la implementación de los sistemas que lo requieren, debido a la reducción en la longitud de las llaves utilizadas sin reducir su seguridad. DSA requiere de una llave de 2048 bits, mientras que las curvas elípticas reducen en 81% este número. De no tener este tipo de tecnología, la blockchain de Bitcoin sería aún más lenta.

Referencias:
[1] Abidi, A, Bouallegue, B. y Kahri, F. (2013) Implementation of Elliptic Curve Digital Signature Algorithm (ECDSA).
[2] Nakamoto, S. (2009). Bitcoin: A Peer-to-Peer Electronic Cash System.
[3] Imagen recuperada de https://i.blogs.es/7bbb1d/criptografia-simetrica-20-copia-/450_1000.png
[4] Imagen recuperada de Hoffstein J., Pipher J., Silverman J.H. An Introduction to Mathematical Cryptography. Undergraduate Texts in Mathematics. Springer, New York, NY, 2014.

Contacto:
Moisés Sánchez: moises.sanchez@cetys.mx
Mauricio Odreman: mauricio.odreman@cetys.mx
Adrián Chouza: adrian.chouza@cetys.edu.mx

Performance comparison study between digital signature algorithm and its counterpart using elliptic curves

Adrián Chouza Delgado - adrian.chouza@cetys.edu.mx
Mauricio Odreman Vera - mauricio.odreman@cetys.mx
Moisés Sánchez Adame - moises.sanchez@cetys.mx

Introduction

- Cryptography studies the way you transform legible text into unreadable text.

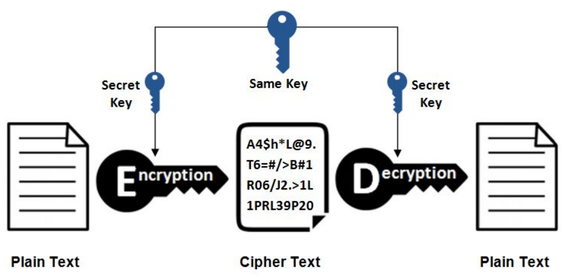

Figure 1. Representación de un sistema criptográfico [2]

- Some of the most common algorithms used are DSA, RSA, AES and Diffie-Hellman, where many of these has a counterpart using elliptic curves.
- An elliptical curve is of the form $y^2=x^3+Ax+B$, and has its own definition for point addition and scalar multiplication.

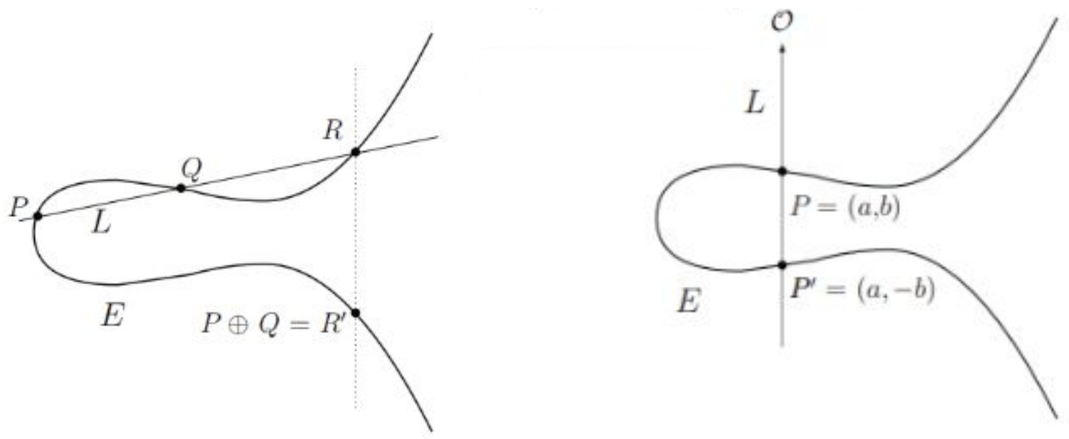

Figure 2. Adición de curvas elípticas con enfoque gráfico. [3]

Objective

- Perform the mathematical-theoretical study on cryptography and elliptic curves.
- Make a performance comparison between digital signature algorithm and its counterpart with elliptical curves.

Methodology

- A 100000 bytes text document is signed using DSA and ECDSA (with curve SECP384R1) and key sizes of 2048 and 381, respectively.
- Sixteen different samples are taken from this text, where each represents a document with size corresponding to a power of 2, from 2^1 to 2^{16}.
- For comparison, this experiment is repeated using key sizes of 1024 and 571 bits for DSA and ECDSA, respectively.

References:
[1] Abidi, A, Bouallegue, B. y Kahri, F. (2013) Implementation of Elliptic Curve Digital Signature Algorithm (ECDSA).
[2] Image recovered from https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences
[3] Image recovered from Hoffstein J., Pipher J., Silverman J.H. An Introduction to Mathematical Cryptography. Undergraduate Texts in Mathematics. Springer, New York, NY, 2014.

ECDSA

Alice wants so sign a message m, so she uses the public values (C,G,n) where C is an elliptical curve, G a point in it and n de order of this point. She chooses d_A (private key) and $Q_A=d_A \bullet G$ (public key). The next algorithm shows how to sign and verify the document.

1. $e = HASH(m)$
2. $k = random(1,n-1)$
3. $(x_1,y_1) = k \bullet G$
4. $r = x_1 \bmod n$
5. $s = k^{-1}(e + rd_A) \bmod n$
6. The signature is (r,s)

1. $e=HASH(m)$
2. $w=s^{-1} \bmod n$
3. $u_1=ew \bmod n$
4. $u_2=rw \bmod n$
5. $(x_1,y_1) = u_1G + u_2Q_A$
6. If $r=x_1$, the signature is valid

Any other person can verify (without knowledge of the value of d_A) that a document with signature equal to (r,s) belongs to Alice this way.

Results

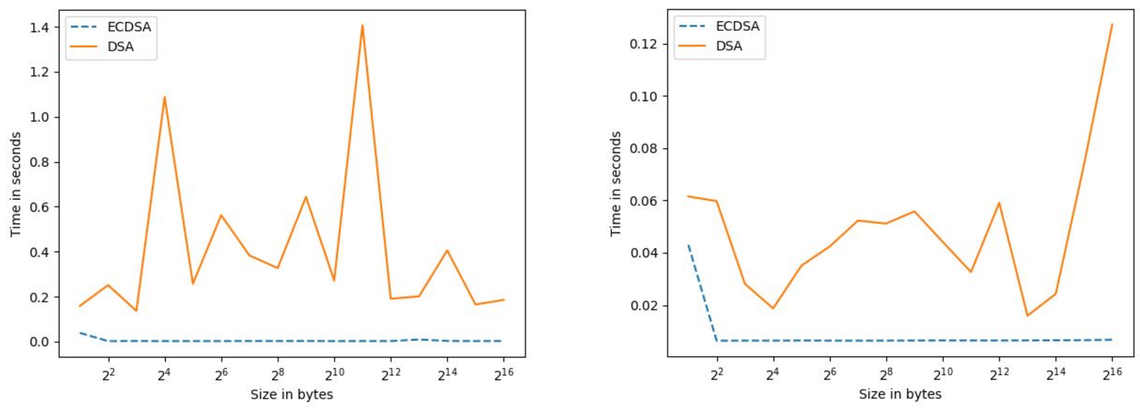

Figure 3. Left: running time (in seconds) of DSA and ECDSA given the size of the document (in bytes) using key sizes of 2048 and 381, respectively. Right: running time (in seconds) of DSA and ECDSA given the size of the document (in bytes) using key sizes of 1024 and 571, respectively. Both plots use a logarithmic scale for the x-axis.

- ECDSA is superior to DSA significantly, never going past the tenth of a second mark, while the last one goes over the whole second.
- It is important to see that the size of the document is not as important as the size of the key, as both plots show.

Conclusion

- The key size has a greater impact on the running time of ECDSA/DSA than the size of the document to sign.
- Currently, the recommended key size for DSA is 2048 bits while ECDSA is 381, 81% smaller.
- ECDSA is preferred over DSA not only by its greater security but its smaller running time.