OOwiki.com
OpencryptO wiki

"Las matemáticas para no matemáticos dentro del Bitcoin".

¿Qué es, como funciona y cómo crear una cuenta (dirección) en Bitcoin?

Guillermo Vidal | OpencryptO wiki
vidal@oowiki.com

La mayoría de nosotros hemos oído de la moneda digital (criptomoneda) Bitcoin y más últimamente por su valor económico que está incrementándose minuto a minuto hasta llegar a alcanzar cifras de $50,000 dólares por unidad (un Bitcoin).

Para hacer transacciones necesitamos una cuenta o conocida común y normalmente como una dirección (address) en la red de Bitcoin para enviar y recibir (bitcoins).

Pero empecemos por entender que es una dirección en Bitcoin, esto es una cadena de números y letras única, una similitud es como tener una cuenta en tu banco preferido desde tu cuenta tu puedes enviar y recibir depósitos es igual en la red de Bitcoin sin embargo en este caso tu no recibes dinero como lo conoces físicamente, sino es dinero digital que por la demanda que tiene lo puedes cambiar por dinero físico para comprar o pagar algún bien físico.

En la actualidad hay un gran número de programas (software) en internet que se llaman carteras (wallet) y únicamente se tienen que instalar en tu computadora o tu smarphone y en estos puedes crear las direcciones que desees, la creación de una dirección de Bitcoin es totalmente "GRATIS" los costos es cuando deseas realizar una operación en la red de Bitcoin, en esta red que está formada por miles de computadoras distribuidas en todo el mundo, para ser más exactos unas 9,700 computadoras, cuando envías una operación; está operación forma aleatoria y siguiendo unas reglas internas de la red una de las estas computadoras procesa tu operación y cobra una comisión ya establecida en conjunto por todas las computadoras que componen la red, aclaremos solo una de las integrantes de la red cobra la comisión, no todas, esta comisión puede variar dependiendo de varios parámetros internos de la red, sin embargo para darse una idea puede la comisión rondar entre $1 y $5 dólares la comisión, no importa el monto de transferencia. Recordando que esta comisión puede variar.

Si deseas saber más a detalle de como "Pagar menos comisión por transacción" puedes consultar nuestro artículo:

https://github.com/bankoin/OOwiki/blob/main/ES-%20Pagar%20menos%20tarifas%20en%20Bitcoin.pdf

Con el anterior antecedente, empecemos el proceso de cómo se crea una dirección Bitcoin desde cero y revisemos los pasos internos para obtener una dirección para usarla de forma común.

Cada vez que se crea una dirección Bitcoin se deben de ejecutar diferentes procedimiento y operaciones matemáticas.

Este es un claro ejemplo de cómo una serie de operación de nivel simple, medio y avanzado de matemáticas son usadas para un caso práctico en la vida real y esto se aplica en el sistema de Bitcoin.

En la red de Bitcoin actualmente se manejan tres tipos de direcciones, la que vamos a crear será la dirección de uso más común y mundial debido a que fue con este tipo de dirección salió en el 2009 a todo el mundo, **posteriormente** con el avance de la tecnología se crearon otros dos tipos de direcciones que tienen más características para usar en la red de Bitcoin.

La dirección que estaremos revisando es:

- P2PKH o Legacy-address (formato de dirección común) es la primera versión y la más común usada de una dirección de Bitcoin que comienza con el número "1" y tiene de 26 a 36 caracteres.

Ejemplo: **1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa**

Para entender un poco los pasos que aplicaremos necesitamos describir de forma general las siguientes funciones matemáticas.

ECC (Elliptic Curve Cryptography*) La criptografía de curva elíptica.*

La ECC es una serie de funciones algebraica matemáticas que como resultado nos proporciona dos cadenas de caracteres que están relacionados uno con el otro, una forma similar para entender esta relación de cadenas de caracteres es la similitud de tener una puerta “A” y su llave “B” que abre la puerta “A”, si pensamos que para abrir la puerta “A” que está cerrada con candado, siempre necesitaremos la llave “B” para abrir esa puerta “A”, es decir si tuviéramos una lleve “C” no podríamos abrir la puerta “A” ya que no está hecha para esa puerta.

Teniendo la relación anterior sería una llave “B”, solo abre la puerta “A”, no puede abrir una puerta “B” o “C” y una puerta “A” solo se puede abrir con su respectiva llave “B”, no se puede abrir con una llave “D”, o “E”.

Ahora ampliaremos el ejemplo de la puerta “A” y la llave “B”, imaginemos que alguien quiere entrar por la puerta “A” claramente sabemos que para entrar necesitara la llave “B”, si tiene una llave “D” o E” no podrá abrir la puerta y por lo cual no podrá entrar. A esta persona que quiere entrar la llamaremos “Transacción”.

Es algo similar con las claves o cadenas de caracteres que nos da la función matemática ECC, estas cadenas de caracteres se llaman llave privada (similar a puerta “A”) y llave publica (similar a llave “B”), y su relación como en el ejemplo de la puerta es única e irrepetible. Ahorra imaginemos que usamos la llave publica para enviar una “Transacción”.

Ahora viene lo interesando donde interviene la ECC en el Bitcoin. Como comentamos en la similitud de una dirección Bitcoin con una dirección de Banco para que alguien pueda enviar a través de Bitcoin necesita una dirección, pero también necesita una clave de seguridad para liberar y confirmar él envió de fondos, si no hubiera clave cualquiera podría sacar los fondos sin autorización del dueño de la cuenta, pues bueno la ECC nos proporciona estos dos componentes, la dirección del Bitcoin se crea partiendo de la llave pública, “no es la llave publica” pero esta es la semilla para crear una dirección en Bitcoin y la clave para liberar los fondos es la llave privada. Otro punto importante de saber es que la llave publica es creada a partir de la llave privada en la ECC de aquí que su relación única e irrepetible.

También es conveniente decir que la ECC es una familia curvas de funciones matemáticas, dependiendo de curva que elijamos nos dará un tipo (cadena de caracteres) de llave privada y llave publica diferente. En el caso de Bitcoin usa el tipo de curva llamada secp256k1.

wikiCOIN
By Bankoin.org

Banco vs Bitcoin

Banco (Tradicional)

12345678910XXXXXXXXXXX

Cuenta bancaria en donde se almacenan los depósitos o se reflejan retiros.

Token o password dinamico de seguridad para Transferir fondos a otra cuenta.

Red Bitcoin (Blockchain)

18Zcyxqna6h7Z7bRjhKvGpr8HSfieQWXqj

Cuenta o dirección Bitcoin común conocida como (address) donde se almacenan depósitos o retiros esta compuesta por 34 caracteres alfanuméricos.

5fa9c7c1756d250664fc5c0d2d65ef4fc07187d10546a0090078da196607a395

Cuando se crea una dirección (address) en Bitcoin siempre esta asociada a una clave (password) alfanumérico de 64 caracteres que se llama “Llave Privada” y sirve para autorizar las transferencias a otra cuenta de Bitcoin.

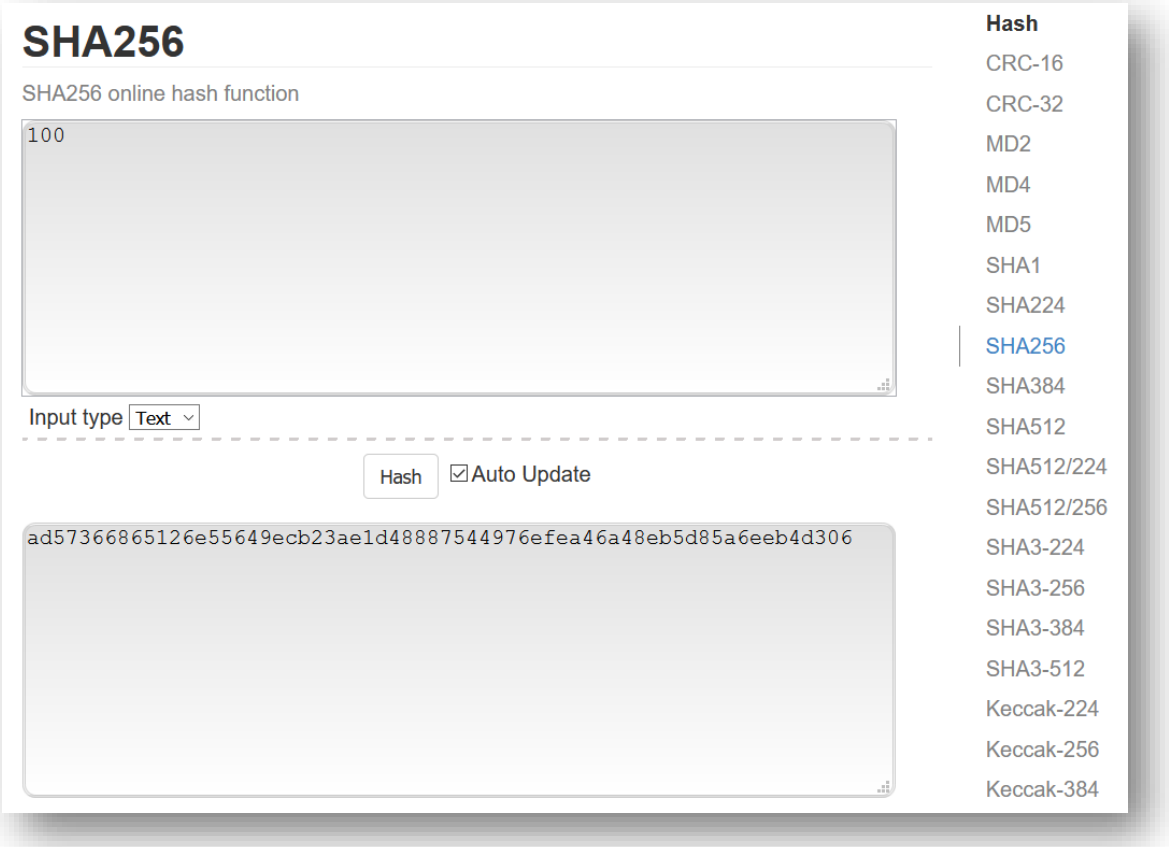
Ahora veremos otra función matemática que utilizaremos para la creación de nuestra dirección de Bitcoin, llamada “SHA-256” (**Secure Hash Algorithm 256)** esta función matemática es una firma digital que asegura que lo que estemos enviando no sea alterado en el camino, es una forma de verificar que la información que se envía es correcta, original y sin haber sido modificado de origen al destino.

La función SHA256 es también una cadena de caracteres y números, y siempre está compuesta por un total de 64 caracteres. Tiene dos características de gran importancia la función SHA-256, es única y es unidireccional que es esto de “unidireccional”, cuando usamos la función aplicada a una cadena de caracteres o archivo la firma entregada siempre será misma y única, nunca otra cadena diferente tendrá esa misma firma, esa es la primera característica, la segunda es que es “unidireccional” esto significa que no hay forma de saber teniendo la firma que cadena fue aplicada para su obtención, esto nos ayuda a que si alguien ajeno al círculo de confianza tiene la firma no podrá saber que cadena se usó para obtenerla, solo podrá verificar la información el destinatario ya que este al recibir los datos podrá aplicar la función SHA256 y comprarla con la original si son iguales los datos que recibió son originales y no han sido modificados. Ejemplo de un SHA-256.Supongamos que deseamos enviar 100 Bitcoins usando la firma digital de SHA256 seria, esto es una suposición muy simple, se necesitan más procesos para una operación dentro de la red de Bitcoin:

Aplicamos SHA256(“100”) = ad57366865126e55649ecb23ae1d48887544976efea46a48eb5d85a6eeb4d306

Lo anterior podemos comprobarlo usando el sitio de internet siguiente:

https://emn178.github.io/online-tools/sha256.html



Otro tipo de “SHA” que también usaremos es el tipo llamado Ripemd160 también es una firma digital.

Revisemos un último proceso es usar Base58, es un sistema de numeración posicional, en este caso también haremos una similitud para una explicación sencilla, imaginemos dos idiomas español e inglés cada palabra tiene su traducción en cada idioma consideremos el siguiente ejemplo.

ESPAÑOL	INGLES
Dueño	Owner
Niño	Little boy
Año	Year

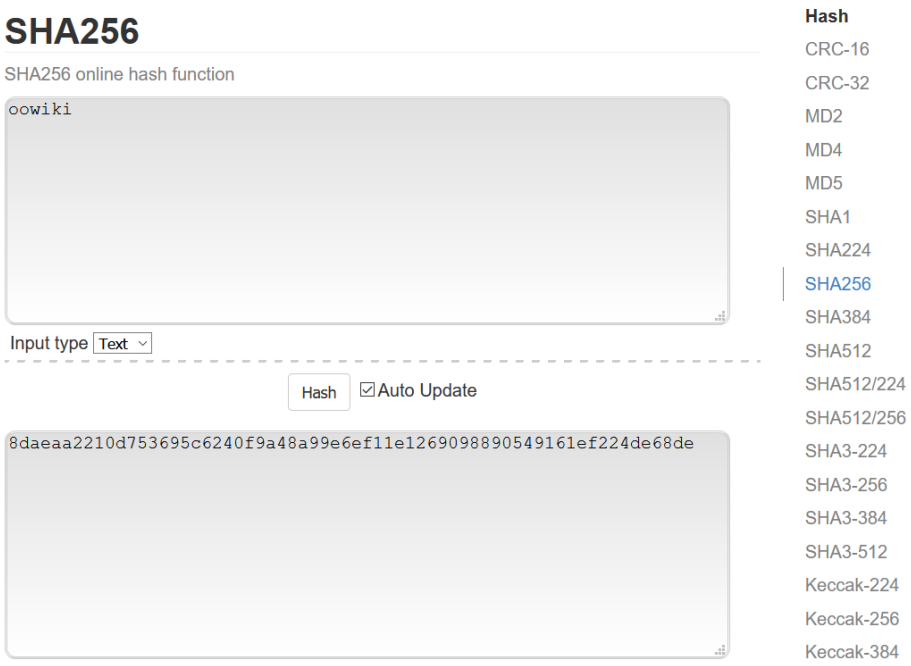
Vemos que en términos lingüísticos las palabras significan lo mismo en cada idioma, sin embargó la escritura (sintaxis) es totalmente diferente, si revisamos el alfabeto de cada idioma veremos que en el idioma ingles revisamos que no existe la letra "Ñ" sin embargo el idioma ingles contiene todas las palabras para interpretar cada palabra en español. El mismo significado, pero con sintaxis diferente.

En el caso de usar Base58 pasa exactamente lo mismo que en el ejemplo de los idiomas. Cuando se genera una dirección Bitcoin se aplican varios pasos (procesos matemáticos) el resultado final de esos procesos normalmente contiene algunos caracteres (sintaxis) que pueden causar errores al usar (compartir) las direcciones en el día a día en los usuarios, por ejemplo, se pueden confundir si una dirección contiene la letra mayúscula "O" con el uso del cero "0". Se usa Base58 para quitarle las sintaxis que pueda causar ese tipo de confusiones en concreto la Base58 se usa para evitar tanto caracteres no alfanuméricos (+ y /) como letras que pueden parecer ambiguas cuando se imprimen (0 - cero, I - i mayúscula, O - y l mayúscula - L minúscula).

Ya tenemos todos los elementos para realizar todos los pasos para la generación de una dirección de Bitcoin, empecemos.

1.- Generamos **llave privada** usando una palabra aleatoria aplicando SHA-256 (64 caracteres).

Nuestra palabra que usamos es (Input Type: **Text**): **oowiki**
Obtenemos la llave primaria: **8daeaa2210d753695c6240f9a48a99e6ef11e1269098890549161ef224de68de**

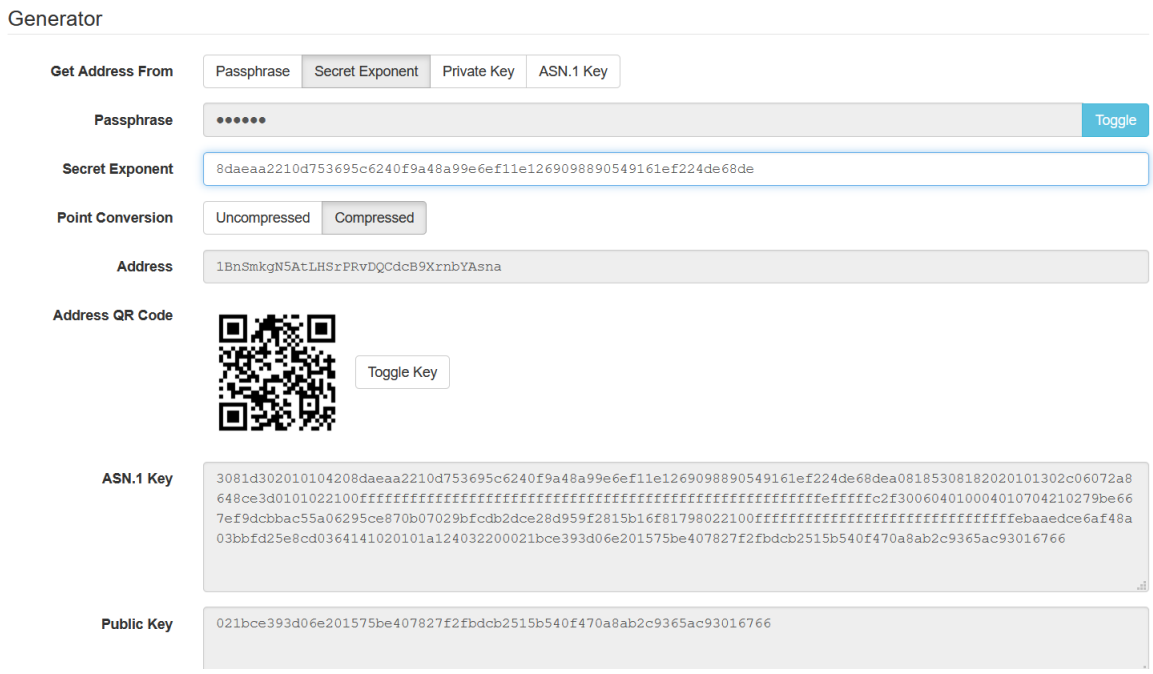


2.- Creamos llave publica usando la llave privada del punto 1.

Usamos el siguiente sitio: https://brainwalletx.github.io/#generator

Llave publica generada (Compressed) =

021bce393d06e201575be407827f2fbdcb2515b540f470a8ab2c9365ac93016766



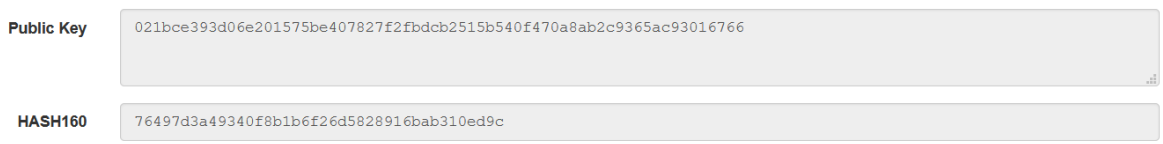
3.- Usaremos la llave publica para iniciar la generación de la dirección de Bitcoin. Sacamos el SHA256 de la llave pública, debemos usar es (Input Type: **Hex**), no Text.

Sacamos SHA256 de llave publica (compressed) =
SHA256 (021bce393d06e201575be407827f2fbdcb2515b540f470a8ab2c9365ac93016766) =
45150f1332942c5f16dfce0108d5b8a0adf15b0feea09f46393419fcd17d80cb

4.- Sacamos Ripemd160 (debe utilizar formato hexadecimal, no texto) del punto 3.

Ripemd160 (45150f1332942c5f16dfce0108d5b8a0adf15b0feea09f46393419fcd17d80cb) =
76497d3a49340f8b1b6f26d5828916bab310ed9c

Tomamos el resultado del sitio: https://brainwalletx.github.io/#generator en la opción de "HASH160" es lo mismo que el Ripemd160.



5.- Agregamos al inicio del resultado del punto 3 el prefijo (prefix) usado para identificar la red Bitcoin, los cuales son dos ceros consecutivos “00”.

00 76497d3a49340f8b1b6f26d5828916bab310ed9c

6.- Sacamos el SHA-256 dos veces (Input Type: **Hex**) del resultado del punto 5

SHA256 (0076497d3a49340f8b1b6f26d5828916bab310ed9c) =
7a878dca75d211584229cf4596c2d7b76240ca8740869152690c504f5406946c

SHA256(7a878dca75d211584229cf4596c2d7b76240ca8740869152690c504f5406946c) =
636944b3d547f4bdee7abdc27d6fb714a07552e933ea169a4d733e8640c5a9d6

Tomamos los primeros 8 caracteres: **636944b3**

7.- Ahora formaremos el formato de la dirección de Bitcoin.

Al resultado del punto 4 adicionamos el resultado del punto 6

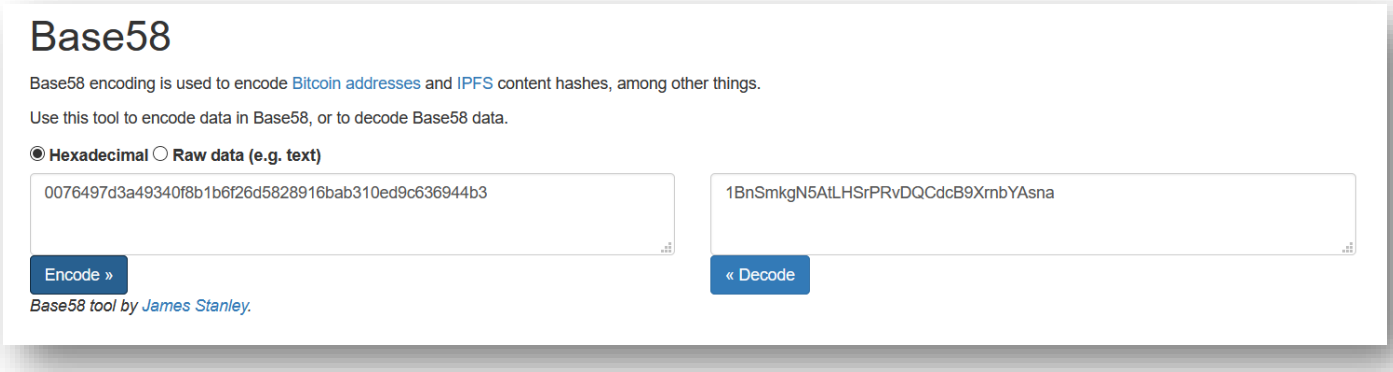
00 76497d3a49340f8b1b6f26d5828916bab310ed9c **636944b3**

8.- Al resultado del punto 6 aplicamos la operación de Base58.

Usaremos el sitio:

Base58 (0076497d3a49340f8b1b6f26d5828916bab310ed9c636944b3) =

1BnSmkgN5AtLHSrPRvDQCdcB9XrnbYAsna



Dirección Bitcoin lista para usar.

Codigo JAVA para sacar direccion Bitcoin,:

Utilizando librería Web3j

```
private static final char[] ALPHABET =
"123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz".toCharArray();
private static final int BASE_58 = ALPHABET.length;
private static final int BASE_256 = 256;

public void GenerateAddrBitcoinWeb3j(String hexPrivateKey) throws Exception {

String addressBitcoin;
String publicKeyCompress;
String addprefix;
String formatWIF;
String pubIni = "";
String addressHexadecimal;
String addressOAP;

BigInteger privKey = new BigInteger(hexPrivateKey, 16);

BigInteger pubKey = Sign.publicKeyFromPrivate(privKey);

String pubKeyHex = pubKey.toString(16);

publicKeyCompress = compressPubKeyCheckBit(pubKey) + pubKeyHex;

MessageDigest sha = MessageDigest.getInstance("SHA-256");

byte[] s1 = sha.digest(hexStringToByteArray(publicKeyCompress));

//MessageDigest rmd = MessageDigest.getInstance("RipeMD160");

byte[] r1 = getHashRipemd160bin(s1); //rmd.digest(s1);

byte[] r2 = new byte[r1.length + 1];

r2[0] = 0;

for (int i = 0; i < r1.length; i++) {

r2[i + 1] = r1[i];

}

byte[] s2 = sha.digest(r2);

byte[] s3 = sha.digest(s2);

byte[] a1 = new byte[25];
```

```
                    for (int i = 0; i < r2.length; i++) {
a1[i] = r2[i];
}
for (int i = 0; i < 4; i++) {
a1[21 + i] = s3[i];
}
        addressBitcoin = encodeBase58bin(a1);
        addprefix = "80" + hexPrivateKey + "01"; // Se agrega el "01" para sacar el checksum referente a la pubic key comprimida
        byte[] sha256priv1 = sha.digest(hexStringToByteArray(addprefix.trim()));
        byte[] sha256priv2 = sha.digest(sha256priv1);
        fourbytes = toHex(sha256priv2).substring(0,8);
        byte[] add4bytes = hexStringToByteArray((addprefix + fourbytes).trim());
        formatWIF = encodeBase58bin(add4bytes);

        addressHexadecimal = "13" + DecodeBase58(addressBitcoin).substring(0, 42);
        addressOAP = GetAddressAOP(addressHexadecimal.trim());

//return address1 = encodeBase58bin(a1) + "   :    " + bcPub.length();
OutPutGenerateAddrBitcoinWeb3j(addressBitcoin, hexPrivateKey, pubKeyHex, publicKeyCompress, formatWIF, addressOAP);

}
public static String compressPubKeyCheckBit(BigInteger pubKey) {
   String pubKeyYPrefix = pubKey.testBit(0) ? "03" : "02";
   //String pubKeyHex = pubKey.toString(16);
   //String pubKeyX = pubKeyHex.substring(0, 64);
   return pubKeyYPrefix;
   }
```

```
public static String encodeBase58bin(byte[] input) {
   //String str = "Example String";
   //byte[] input = str.getBytes();

      if (input.length == 0) {
         // paying with the same coin
         return "";
      }

      //
      // Make a copy of the input since we are going to modify it.
      //
      input = copyOfRange(input, 0, input.length);

      //
      // Count leading zeroes
      //
      int zeroCount = 0;
      while (zeroCount < input.length && input[zeroCount] == 0) {
         ++zeroCount;
      }

      //
      // The actual encoding
      //
      byte[] temp = new byte[input.length * 2];
      int j = temp.length;

      int startAt = zeroCount;
      while (startAt < input.length) {
         byte mod = divmod58(input, startAt);
         if (input[startAt] == 0) {
            ++startAt;
         }

         temp[--j] = (byte) ALPHABET[mod];
      }

      //
      // Strip extra '1' if any
      //
      while (j < temp.length && temp[j] == ALPHABET[0]) {
         ++j;
      }

      //
      // Add as many leading '1' as there were leading zeros.
      //
      while (--zeroCount >= 0) {
         temp[--j] = (byte) ALPHABET[0];
```

```
                }

        byte[] output = copyOfRange(temp, j, temp.length);

        return new String(output);
    }

private static final int[] INDEXES = new int[128];
    static {
        for (int i = 0; i < INDEXES.length; i++) {
            INDEXES[i] = -1;
        }
        for (int i = 0; i < ALPHABET.length; i++) {
            INDEXES[ALPHABET[i]] = i;
        }
    }
```

En OOwiki.com y Bankoin.org desarrollamos proyectos de ICO para todo tipo de sectores privados y públicos, con proyectos y experiencia demostrada.