Base 58

Os endereços Bitcoin são strings de dígitos e caracteres que podem ser compartilhados com qualquer pessoa ou instituição que queira lhe enviar dinheiro. O endereço de remessa sempre começa com um dígito 1. Eis um exemplo

`1vru7YTbmkgA62Q4qaXVymmVMS2nhFU4v3`

Se fossemos comparar uma transação bitcoin a um cheque em papel, o endereço bitcoin é que escreveríamos no cheque bo campo "à ordem de" Um endereço bitcoin pode representar muita coisa: uma pessoa, uma instituição, ou um script. Comecemos examinando o caso simples: um endereço bitcoin que representa uma chave pública e é dela derivado.

Ele é obtido a partir da chave pública usando-se a função hash de mão única. Os algoritmos usados para gerar um endereço a partir de uma chave pública são o `SHA256` e o `RIPEMD160`.

É uma aplicação simples: aplica-se o SHA256 à chave pública (o que resulta em 256 bits) e daí aplica-se o RIPEMd160 ao resultado o que gera um número de 160 bits (20 bytes).

Os endereços bitcoin são apresentados ao usuário na codificação `Base58Check` que usa uma base de numeração 58 e um checksum para proteger a introdução de erros de transcrição.

A Base 58 é usada em muitos outros lugares do bitcoin (endereços, chaves privadas, chave encriptada ou hash de script). O objetivo desta codificação é diminuir o comprimento dos números associados. Lembrando a teoria, quanto maior é a base de um sistema de numeração (e maior a coleção de símbolos empregados) mais curtos são os números escritos nesta base.

Relembrando:

`O número binário (base 2) 0000.0010.0111.0011 pode ser escrito em hexadecimal (base 16) como 02.73. No primeiro caso usa 16 dígitos e no segundo apenas 4.`

O Base58 está derivado do sistema Base64 no qual foram feitas algumas mudanças:

- a boa: excluiram-se os dígitos 0 (zero), O (O maiúsculo) l (ele minúsculo) e I (I maíusculo) além dos `+` e `/`. Agiu-se assim, porque dependendo da família de fontes usadas na impressão pode haver confusão.
- a ruim: 64 é uma potência de 2 e 58 não é: a conversão para 64 pode se fazer apenas redistribuindo os bits do número. Para base 58 isto não é possível: há que fazer as contas envolvendo números gigantes, mas este tema (operar com números gigantescos) não assusta o bitcoin.

Para fornecer segurança contra erros de digitação, o formato `Base58Check` tem um código de verificação embutido: O checksum é formado por 4 bytes adicionais acrescentados ao final do que se quer codificar. Estes bytes são os 4 primeiros bytes da aplicação de um SHA256 duplo sobre o que se quer codificar.

A primeira coisa que um manipulador faz ao receber um código destes é aplicar o SHA256 duas vezes ao código original e compará-lo com os 4 bytes que aí estão. Havendo diferença isto sinaliza que um erro foi cometido e que o código é inválido.

Para converter um código qualquer, primeiro adicionamos um byte de versão que sinaliza o que este código é. Os valores possíveis são:

Tipo	Byte prefixo	Base58
Endereço Bitcoin	0x00	1
Endereço de um script para pagamento	0x05	3
Endereço Bitcoin Testnet	0x6F	m ou n
Chave privada WIF	0x80	5, K ou L
Chave privada criptografada em BIP38	0x0142	6P
Chave pública extendida em BIP32	0x0488B21E	xpub

Eis o processo completo de aplicar Base58 a um endereço de pagamento:

1. Acrescentar no início o(s) bytes(s) de versão
2. Aplicar 2 vezes a função HASH256 e obter os primeiros 4 bytes do resultado
3. Acrescentar estes 4 bytes ao final
4. Codificar o resultado obtido em Base 58

O resultado está codificado de maneira inequívoca, versionada e verificada.

Formatos de chaves A chave privada pode ser representada em diversos formatos, todos correspondendo ao mesmo número de 256 bits. Veja-se 3 comuns:

Tipo	Prefixo	Descrição
Hex	nenhum	64 digs hexadecimais
WIF	5	Base58Check
WIF-compr	K ou L	igual acima, com sufixo 0x01

Formatos de chave pública A chave pública é um ponto na curva elíptica com suas coordenadas (x, y). Geralmente é apresentado com um 04 no início seguido por 2 números de 256 bits. O prefixo 4 é usado para distinguir as chaves públicas não comprimidas das chaves públicas comprimidas (que começam por 02 ou 03).

As chaves públicas comprimidas começaram a ser usadas no bitcoin para reduzir o tamanho das transações (sobretudo no blockchain). Mas, como se viu antes, no par (x, y), conhecido x o y pode ser calculado. Há uma pequena dificuldade: como y aparece como y^2 ele pode ser positivo ou negativo. A coordenada y pode estar acima ou abaixo do eixo x. Então, para omitir a coordenada y precisamos guardar o sinal (positivo ou negativo) do y. Então o prefixo é 02 se y for par e 03 se for ímpar. As chaves públicas comprimidas estão se tornando padrão no bitcoin.

Aqui há alguma confusão: uma chave privada não pode ser comprimida e o termo chave privada comprimida significa que ela é um byte maior do que não comprimida: (o 01 adicional). O termo "chave privada comprimida" na realidade quer dizer "chave privada a partir da qual chaves públicas comprimidas devem ser derivadas". O termo "chave privada não comprimida" quer dizer chave privada a partir da qual chaves públicas não comprimidas devem ser derivadas. Seja um exemplo em Python 3

```
% incluir f:/p/n/n25/chaend.py
def chaend():
 import bitcoin as bc
 prk_valida = False
 while not prk_valida:
  prk = bc.random_key()
  prk_hex = bc.decode_privkey(prk, 'hex')
  prk_valida = 0 < prk_hex < bc.N
 print ("prk (hex): ", prk)
 print ("prk (decimal): ", prk_hex)
 prk_wif = bc.encode_privkey(prk_hex, 'wif')
 print ("prk (WIF): ", prk_wif)
 prk_comp = prk + '01'
 print ("prk compr (hex): ", prk_comp)
 wif_prk_comp = bc.encode_privkey
   (bc.decode_privkey(prk_comp, 'hex'), 'wif')
 print ("prk (WIF-Compr): ", wif_prk_comp)
 puk = bc.fast_multiply(bc.G, prk_hex)
 print ("puk (x,y):", puk)
 puk_hex = bc.encode_pubkey(puk,'hex')
 print ("puk (hex):", puk_hex)
 (puk_x, puk_y) = puk
 if (puk_y % 2) == 0:
  comp_pref = '02'
 else:
  comp_pref = '03'
 hex_c_puk = comp_pref + bc.encode(puk_x, 16)
 print ("puk comp (hex):", hex_c_puk)
 print ("endereço bitcoin (b58check):",
   bc.pubkey_to_address(puk))
chaend()

cujo resultado é:
=== RESTART: F:\p\n\n25\chaend.py ===
prk (hex):  53fcb29aa724709bafd786a99d...
prk (decimal):  3798844457301892175158...
prk (WIF):  5JTGvGey4W1hSvLtdotpiDfZJ1...
prk compr (hex):  53fcb29aa724709bafd7...
prk (WIF-Compr):  Kz2yG86aST45XQSnuTyj...
puk (x,y):  (114770241517656892525047 5...
puk (hex):  04fdbda799881ffae9ebf9f11a...
puk comp (hex):  02fdbda799881ffae9ebf...
endereço bitcoin  (b58check): 144SQ1Zq...
```

Os ... indicam que a resposta segue, mas foram excluídos apenas para que o interessado possa conferir a resposta completa diretamente no Python.

☞ Para você fazer

Você vai receber 2 chaves privadas e deve achar o endereço bitcoin gerado a partir de cada chave privada. Use o python como mostrado acima, ou qualquer ambiente de outra linguagem.

1.
`bf7483ef0c9df33618c5039acb2e43cf 9bf90f85213aaa4b494d1afc713323f0`

2.
`5fb7551ceaa446b221357cb57eeb50c8 737d1e85a3c08eea8f31f22b95d54b8a`

Responda aqui:

endereço bitcoin da chave 1
endereço bitcoin da chave 2