

Dominando as Carteiras no Bitcoin Core: Da Teoria à Prática na Signet

Anfitrião:
Rafael Penna



O Que Veremos!

Carteiras



Tipos de Endereços



UTXOs e Transações



SegWit na Signet

Chaves e Criptografia



HD Wallets e Descriptors



O Que Veremos!

Carteiras



Chaves e
Criptografia

Tipos de
Endereços



HD Wallets e
Descriptors



UTXOs e
Transações



SegWit na
Signet



Carteiras

O que é uma Carteira Bitcoin?

🔑 Guarda **chaves privadas** (não moedas)

📍 Gera **endereços** para receber BTC

✍️ Assina **transações** para gastar



💡 “Carteira = Cofre de chaves que controlam seus bitcoins.”

Carteiras

Carteiras no Bitcoin Core

- Cada carteira = arquivo `wallet.dat`
- Local padrão:
 - Linux → `~/.bitcoin/wallets/`
 - Windows → `%APPDATA%\Bitcoin\wallets\`
- Contém:
 - Chaves
 - Endereços
 - Metadados
 - UTXOs controlados pela carteira



Carteiras



Carteiras no Bitcoin Core

- Contém:

- Chaves

```
bitcoin-cli -datadir="." -rpcwallet=NovaCarteira listdescriptors true
```

- Endereços

```
bitcoin-cli -datadir="." -rpcwallet=NovaCarteira listreceivedbyaddress 0 true
```

- Metadados

```
bitcoin-cli -datadir="." -rpcwallet=NovaCarteira getwalletinfo
```

- UTXOs controlados pela carteira

```
bitcoin-cli -datadir="." -rpcwallet=NovaCarteira listunspent
```

Carteiras

Criando e Gerenciando Carteiras

- Criar:
 - `bitcoin-cli -datadir="." createwallet "MinhaCarteira"`
- Carregar:
 - `bitcoin-cli -datadir="." loadwallet "MinhaCarteira"`
- Descarregar:
 - `bitcoin-cli -datadir="." unloadwallet "MinhaCarteira"`
- Listar abertas:
 - `bitcoin-cli -datadir="." listwallets`
- Listar todas do diretório:
 - `bitcoin-cli -datadir="." listwalletdir`

Carteiras

Criando e Gerenciando Carteiras

```
penna@penna-H610M-K-DDR4: ~/AreaBitcoin/BitcoinCore/signet
penna@penna-H610M-K-DDR4: ~/AreaBitcoin/BitcoinCore/signet$ bitcoin-cli -datadir="." createwallet "NovaCarteira"
{
  "name": "NovaCarteira"
}
penna@penna-H610M-K-DDR4: ~/AreaBitcoin/BitcoinCore/signet$ bitcoin-cli -datadir="." unloadwallet "NovaCarteira"
{
}
penna@penna-H610M-K-DDR4: ~/AreaBitcoin/BitcoinCore/signet$ bitcoin-cli -datadir="." loadwallet "NovaCarteira"
{
  "name": "NovaCarteira"
}
penna@penna-H610M-K-DDR4: ~/AreaBitcoin/BitcoinCore/signet$ bitcoin-cli -datadir="." listwallets
[
  "demo-signet2",
  "MinhaCarteira2",
  "NovaCarteira"
]
penna@penna-H610M-K-DDR4: ~/AreaBitcoin/BitcoinCore/signet$ bitcoin-cli -datadir="." listwalletdir
{
  "wallets": [
```

O Que Veremos!

Carteiras



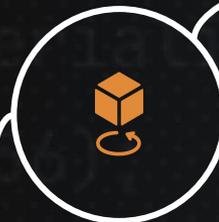
Chaves e
Criptografia

Tipos de
Endereços



HD Wallets e
Descriptors

UTXOs e
Transações



SegWit na
Signet

Chaves e Criptografia



Chaves no Bitcoin

- **Privada** → segredo absoluto
- **Pública** → derivada da privada
- **Endereço** → versão amigável da chave pública

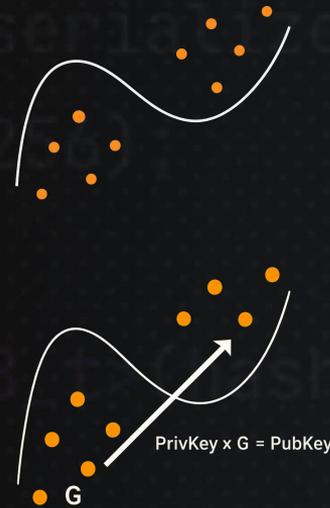
💡 “Quem controla a chave privada controla o Bitcoin.”

Chaves e Criptografia



Criptografia de Chaves

- **Privada** = número aleatório grande
- Baseada na curva elíptica **secp256k1**
- Pública = **PrivKey** × **G**
- **Endereço** = hash da chave pública



Chaves e Criptografia



Na **Prática** (Bitcoin Core)

- Gerar endereço → **getnewaddress**

```
bitcoin-cli -datadir="." -rpcwallet="NovaCarteira" getnewaddress
```

- Ver detalhes → **getaddressinfo <addr>**

```
bitcoin-cli -datadir="." -rpcwallet="NovaCarteira" getaddressinfo  
tb1q7kwvtav5t8ds49fnv9d87lqw9dhkycnhvjahl
```

- Exportar chave privada em carteiras antigas → **dumpprivkey <addr>**

Hoje em dia é diferente... **Chegaremos lá**

O Que Veremos!

Carteiras



Tipos de Endereços



UTXOs e Transações



SegWit na Signet



Chaves e Criptografia



HD Wallets e Descriptors



Tipos de Endereços

Primeiros Endereços (P2PK)

- Usados nos ~70 primeiros blocos
- Pagamento direto para a chave pública
- Problema: chave exposta cedo



Tipos de Endereços

Legacy (P2PKH)

- Endereços começando com 1
- Paga ao **hash da chave pública**

- Script:

```
OP_DUP OP_HASH160 <hash160(pubkey)> OP_EQUALVERIFY OP_CHECKSIG
```



Tipos de Endereços

P2SH

- Endereços 3...
- Criado para encapsular **scripts complexos**
- Guarda só o **hash** do script
- Depois usado no **SegWit compatível**



Tipos de Endereços

SegWit

- Separou **assinaturas** da transação
- Resolveu:
 - **Maleabilidade**
 - **Espaço** em bloco
- **Formatos**: P2SH-SegWit (**3...**), Bech32 (**bc1q...**)



Tipos de Endereços

Taproot (P2TR)

- Endereços `bc1p...`
- Assinaturas **Schnorr**
- Scripts privados e flexíveis (MAST)
- Ativado em **2021**



🔑 Com Taproot e MAST, você pode ter várias regras possíveis, mas só precisa revelar a que realmente usou, ganhando privacidade e eficiência.

Tipos de Endereços



Criando os endereços no Bitcoin Core

- Legacy (P2PKH):

```
bitcoin-cli -datadir="." -rpcwallet="NovaCarteira" getnewaddress "" legacy
```

- P2SH:

```
bitcoin-cli -datadir="." -rpcwallet="NovaCarteira" getnewaddress "" p2sh-segwit
```

- SegWit Nativo (bech32):

```
bitcoin-cli -datadir="." -rpcwallet="NovaCarteira" getnewaddress "" bech32
```

- Taproot (bech32m):

```
bitcoin-cli -datadir="." -rpcwallet="NovaCarteira" getnewaddress "" bech32m
```

O Que Veremos!

Carteiras



Tipos de Endereços



UTXOs e Transações



SegWit na Signet

Chaves e Criptografia



HD Wallets e Descriptors



HD Wallets e Descriptors



Problema das **carteiras antigas**

- Cada endereço **independente**
- Backup exigia salvar **todas** as chaves
- Perda de 1 chave = perda de fundos
- Difícil de **organizar**

HD Wallets e Descriptors



HD Wallets (BIP32/BIP44)

- **Seed única** → infinitos endereços
- Caminhos de **derivação**
- Estrutura **hierárquica**:

`m / purpose' / coin_type' / account' / change / address_index`

- Ex.: `m/84' / 0' / 0' / 0 / 0` (SegWit nativo)
- Backup = **só a seed** (ou xprv)

HD Wallets e Descriptors



HD Wallets (BIP32/BIP44)

`m / purpose' / coin_type' / account' / change / address_index`

m

- master private key (raiz, derivada da seed).

purpose'

- 44' → Legacy (P2PKH)
- 49' → P2SH-SegWit
- 84' → Native SegWit (Bech32)
- 86' → Taproot

coin_type'

- 0' → Bitcoin Mainnet
- 1' → Bitcoin Testnet/Signet/Regtest

HD Wallets e Descriptors



HD Wallets (BIP32/BIP44)

`m / purpose' / coin_type' / account' / change / address_index`

account'

- Dividir contas lógicas (normalmente 0').
- Semântica na organização das contas.

change

- 0 → endereços externos (recebimento)
- 1 → endereços internos (troco)

address_index

- O índice incremental (0, 1, 2, ...) que gera cada endereço.

HD Wallets e Descriptors



HD Wallets (BIP32/BIP44)

`m / purpose' / coin_type' / account' / change / address_index`

Exemplo

`m/84'/1'/0'/0/0` → primeiro endereço externo (recebimento)

`m/84'/1'/0'/1/0` → primeiro endereço interno (troco)

HD Wallets e Descriptors



Descriptors

- Formalizam a derivação da carteira
- Incluem:
 - Tipo de script (**pkh**, **wpkh**, **tr**)
 - Caminho (**84' / 0' / 0'**)
 - Chave estendida (**xpub** / **tpub**)
 - Checksum
- Torna carteiras portáteis e auditáveis

HD Wallets e Descriptors



Exemplo de Descriptor

```
wpkh([F23ABCD1/84h/1h/0h]tpub.../0/*)#abc123
```

- `wpkh()` = tipo SegWit
- `[F23ABCD1/84h/1h/0h]` = fingerprint + caminho
- `tpub...` = chave estendida pública
- `/0/*` = endereços externos
- `#abc123` = checksum

HD Wallets e Descriptors



Exemplo de Descriptor

```
wpkh([F23ABCD1/84h/1h/0h]tpub.../0/*)#abc123
```

- `wpkh()` = tipo SegWit
- `[F23ABCD1/84h/1h/0h]` = fingerprint + caminho
- `tpub...` = chave estendida pública
- `/0/*` = endereços externos
- `#abc123` = checksum

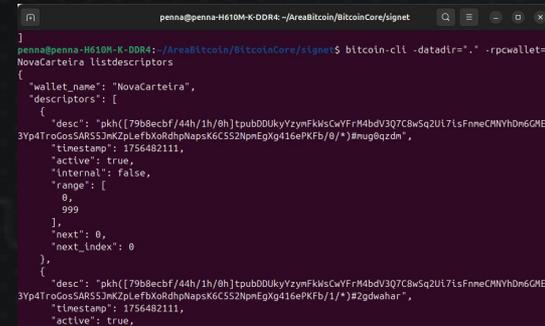
identifica a
chave-mestra de
origem

HD Wallets e Descriptors

Exemplo de Descriptor

```
bitcoin-cli -datadir="." -rpcwallet=NovaCarteira listdescriptors
```

```
{ "wallet_name": "NovaCarteira",  
  "descriptors": [...,  
    {  
      "desc":  
"wpkh([79b8ecbf/84h/1h/0h]tpubDCQKd5an5BNaDKy7XA6d4jMx6bBJhHACTZ7gsNb55Sk2U3NEkp1APuHHmpa  
Hsk1tygC5dLeARAvnh7nzqo4RElyzUKYJiWXbxkbSRP4GJyo/0/*)#2ly5y3sf",  
      "timestamp": 1756482111,  
      "active": true,  
      "internal": false,  
      "range": [  
        0,  
        1000  
      ],  
      "next": 1,  
      "next_index": 1  
    },...]  
}
```



```
penna@penna-H610M-K-DDR4: ~/AreaBitcoin/BitcoinCore/signed  
] penna@penna-H610M-K-DDR4: ~/AreaBitcoin/BitcoinCore/signed$ bitcoin-cli -datadir="." -rpcwallet=NovaCarteira listdescriptors  
{  
  "wallet_name": "NovaCarteira",  
  "descriptors": [  
    {  
      "desc": "pkh([79b8ecbf/44h/1h/0h]tpubDDUkyZynFkMsCwYFrM4bdV3Q7C8wSq2U7LsFnneCMWYhdM6GME3Yp4TroGos5ARs5jKzPLefBxOrdhpNapsK6C52NpnEgKg416ePKFB/0/*)#nug0qzdm",  
      "timestamp": 1756482111,  
      "active": true,  
      "internal": false,  
      "range": [  
        0,  
        999  
      ],  
      "next": 0,  
      "next_index": 0  
    },  
    {  
      "desc": "pkh([79b8ecbf/44h/1h/0h]tpubDDUkyZynFkMsCwYFrM4bdV3Q7C8wSq2U7LsFnneCMWYhdM6GME3Yp4TroGos5ARs5jKzPLefBxOrdhpNapsK6C52NpnEgKg416ePKFB/1/*)#2gdwaha",  
      "timestamp": 1756482111,  
      "active": true,  
    }  
  ]  
}
```

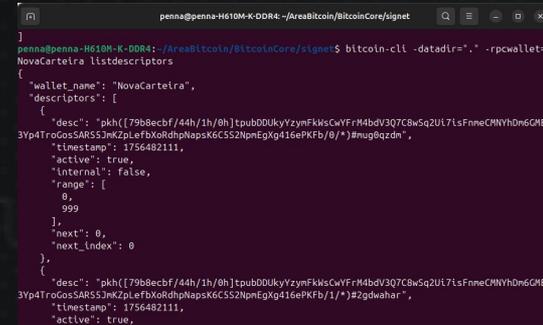
HD Wallets e Descriptors

Exemplo de Descriptor

```
bitcoin-cli -datadir="." -rpcwallet=NovaCarteira listdescriptors
```

```
{ "wallet_name": "NovaCarteira",  
  "descriptors": [...,  
    {  
      "desc":  
"wpkh([79b8ecbf/84h/1h/0h]tpubDCQKd5an5BNaDKy7XA6d4jMx6bBJhHACTZ7gsNb55Sk2U3NEkp1APuHHmpa  
Hsk1tygC5dLeARAvnh7nzqo4RElyzUKYJiWXbxbkSRP4GJyo/0/*)#2ly5y3sf",  
      "timestamp": 1756482111,  
      "active": true,  
      "internal": false,  
      "range": [  
        0,  
        1000  
      ],  
      "next": 1,  
      "next_index": 1  
    },...]  
}
```

Por padrão
8 descritores



```
penna@penna-H610M-K-DDR4: ~/AreaBitcoin/BitcoinCore/signed  
] penna@penna-H610M-K-DDR4:~/AreaBitcoin/BitcoinCore/signed$ bitcoin-cli -datadir="." -rpcwallet=NovaCarteira listdescriptors  
{  
  "wallet_name": "NovaCarteira",  
  "descriptors": [  
    {  
      "desc": "pkh([79b8ecbf/44h/1h/0h]tpubDDUkyZynFkMsCwYFrM4bdV3Q7C8wSq2U7LsFnneCMWYhdM6GME3Yp4TroGos5ARs5jKzPLefbXordhpNapsK6C52NpnEgKg416ePKFB/0/*)#nug0qzdm",  
      "timestamp": 1756482111,  
      "active": true,  
      "internal": false,  
      "range": [  
        0,  
        999  
      ],  
      "next": 0,  
      "next_index": 0  
    },  
    {  
      "desc": "pkh([79b8ecbf/44h/1h/0h]tpubDDUkyZynFkMsCwYFrM4bdV3Q7C8wSq2U7LsFnneCMWYhdM6GME3Yp4TroGos5ARs5jKzPLefbXordhpNapsK6C52NpnEgKg416ePKFB/1/*)#2gdwahar",  
      "timestamp": 1756482111,  
      "active": true,  
    }  
  ]  
}
```

O Que Veremos!

Carteiras



Tipos de Endereços



UTXOs e Transações



SegWit na Signet



Chaves e Criptografia



HD Wallets e Descriptors



UTXO e Máquina de Script

UTXO: O Modelo do Bitcoin

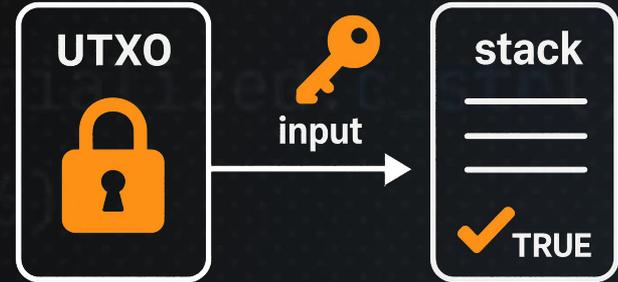
- **UTXO = Unspent Transaction Output**
- Transações criam outputs (condições de gasto)
- Para gastar → txid + índice + assinatura
- UTXOs não gastos = saldo disponível



UTXO e Máquina de Script

Máquina de Script

- Cada UTXO carrega o **contrato**: `scriptPubKey`
- **Input** fornece a prova: `scriptSig` ou `witness`
- Core executa na **stack**
- Se resultado = **TRUE** → gasto válido



Exemplo P2PKH:

- `scriptPubKey: OP_DUP OP_HASH160 <hash160(pubkey)> OP_EQUALVERIFY OP_CHECKSIG`
- `scriptSig: <assinatura> <pubkey>`

O Que Veremos!

Carteiras



Tipos de Endereços



UTXOs e Transações



SegWit na Signet

Chaves e Criptografia



HD Wallets e Descriptors

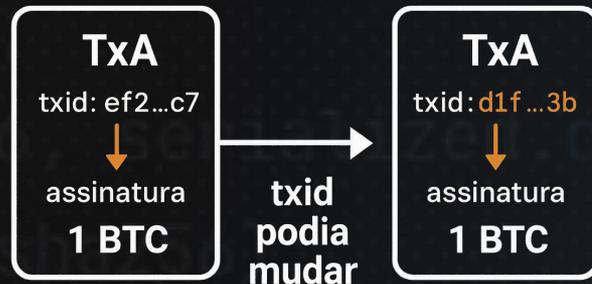


SegWit



Problemas Antes do SegWit

- **Maleabilidade** → txid podia mudar



- **Assinaturas** ocupavam **muito espaço**



- **Taxas altas**, poucos txs por bloco

SegWit



O que é SegWit

- Segregated Witness = **separar assinaturas**
- Resolveu **maleabilidade** + **escalabilidade**
- Permitiu **Lightning** e novos scripts



SegWit



Formatos SegWit

- **P2SH-SegWit** (3...) → compatível
- **Bech32 (nativo)** (bc1q...) → mais eficiente
- **Taproot** (Bech32m) (bc1p...) → Schnorr + MAST

SegWit



Como Funciona

- **Geração** de Endereço SegWit
 - Tudo começa na **seed aleatória** (128/256 bits)
 - Seed → **master key + chain code** (BIP32)
 - Caminho de **derivação** (ex.: `m/84'/0'/0'/0/0`) define:
 - Tipo de endereço → SegWit (BIP84)
 - Da chave privada derivada → chave pública → endereço específico

SegWit



Como Funciona

- **Witness Program e scriptPubKey**
 - A partir da pubkey → `hash160(pubkey)` (20 bytes)
 - Monta o **witness program**:
`0x00 0x14 <20-byte pubkeyhash>`
 - Esse witness program entra no `scriptPubKey` do output
 - O endereço **Bech32** é só a **codificação amigável** do witness program

💡 Resumindo: “O endereço Bech32 é a versão legível do script SegWit.”

SegWit



Como Funciona

- **Gastando o UTXO SegWit**
 - **scriptSig:** **vazio**
 - **txinwitness:** [assinatura, pubkey]
 - O nó valida:
 - i. $\text{hash160}(\text{pubkey}) = \text{pubkeyhash}$ no witness program
 - ii. Assinatura **é válida** com essa pubkey?
 - Se tudo confere → UTXO pode ser gasto

💡 Resumindo: “*scriptPubKey* define a regra, o *witness* traz a prova.”

SegWit



SegWit na Prática (Signet)

- Criar carteira:

```
bitcoin-cli -datadir="." createwallet CarteiraExemplo
```

- Criar endereço SegWit:

```
bitcoin-cli -datadir="." -rpcwallet="CarteiraExemplo"  
getnewaddress "endereco1" bech32
```

- Receber faucet

<https://alt.signetfaucet.com/>

- **getaddressinfo** → witness program

```
bitcoin-cli -datadir="." -rpcwallet="CarteiraExemplo"  
getaddressinfo <ADDR>
```

SegWit



```
penna@penna-H610M-K-DDR4: ~/AreaBitcoin/BitcoinCore/signet
penna@penna-H610M-K-DDR4:~/AreaBitcoin/BitcoinCore/signet$ bitcoin-cli -datadir="." -rpcwallet="demo-sig
net" getaddressinfo tb1q5x6vqpjsk2535z9dfpmpmg0yj4argc49uf8nh3
{
  "address": "tb1q5x6vqpjsk2535z9dfpmpmg0yj4argc49uf8nh3",
  "scriptPubKey": "0014a1b4c00650b2a91a08ad48761da1e4957a3462a5",
  "ismine": true,
  "solvable": true,
  "desc": "wpkh([7746de77/84h/1h/0h/0/0]02bad300bbdbb7fb813e13bff2762d2dc92353335ba9c8974ad3e2295f6e99c2
b2)#dmfv8gul",
  "parent_desc": "wpkh([7746de77/84h/1h/0h]tpubDDoa2ZZbm9SK9pwgK1A77UQpLUXG9h1aYtbnbM3bS8HwfXEUSVYCEvU5n
79xAUQv7zx32Zu48hTGWudQeZtLF5ZYC7itdUpoDwDgDhT7v6i/0/*)#vuehxn9j",
  "iswatchonly": false,
  "isscript": false,
  "iswitness": true,
  "witness_version": 0,
  "witness_program": "a1b4c00650b2a91a08ad48761da1e4957a3462a5",
  "pubkey": "02bad300bbdbb7fb813e13bff2762d2dc92353335ba9c8974ad3e2295f6e99c2b2"
  "ischange": false,
  "timestamp": 1756649907,
  "hdkeypath": "m/84h/1h/0h/0/0",
  "hdseedid": "0000000000000000000000000000000000000000000000000000000000000000",
  "hdmasterfingerprint": "7746de77",
  "labels": [
    "endereco1"
  ]
}
```

SegWit



Transação SegWit (Demo)

1. Criar um novo endereço de **destinatário**:

```
bitcoin-cli -datadir="." -rpcwallet="CarteiraExemplo" getnewaddress "destinatario" bech32
```

2. Criar a transação bruta (sem inputs):

```
bitcoin-cli -datadir="." -rpcwallet="CarteiraExemplo" createrawtransaction  
"[]" "[{\ "<DEST_ADDR>\":0.0005}]"
```

3. Completar:

```
bitcoin-cli -datadir="." -rpcwallet="CarteiraExemplo" fundrawtransaction  
"<HEX_CRAWT>" | jq -r .hex
```

💡 Os retornos dos passos 1 e 2 são uma **string em hexadecimal**

SegWit



Transação SegWit (Demo)

4. Assinar:

```
bitcoin-cli -datadir="." -rpcwallet="CarteiraExemplo" signrawtransactionwithwallet  
"<HEX_FRAWT>" | jq -r .hex
```

5. Enviar:

```
bitcoin-cli -datadir="." -rpcwallet="CarteiraExemplo" sendrawtransaction  
"<HEX_SRAWT>"
```

6. Examinar:

```
bitcoin-cli -datadir="." -rpcwallet="CarteiraExemplo" listunspent
```

```
bitcoin-cli -datadir="." -rpcwallet="CarteiraExemplo" decoderawtransaction  
"<SIGNRAWT>"
```

Conclusão

Carteiras



Tipos de Endereços



UTXOs e Transações



Chaves e Criptografia



HD Wallets e Descriptors



SegWit na Signet

Conclusão



- **Carteiras** = cofres de chaves
- **Endereços** = diferentes scripts
- **HD Wallets** e **Descriptors** = organização moderna
- **UTXO** = saldo real do Bitcoin
- **SegWit** = escalabilidade + Lightning
- **Próximos passos**: Taproot & scripts avançados

```
uint8_t hash[SHA256_DIGEST_LENGTH],  
SHA256_CTX sha256;  
SHA256_Init(&sha256);  
SHA256_Update(&sha256, serialized.c_str(),  
SHA256_Final(hash, &sha256);  
  
return std::vector<uint8_t>(hash, hash + S
```

FIM

Obrigado!

Grupo Whatsapp - Bitup Coders

https://chat.whatsapp.com/IB3rBamPe6QlvfncMBb3nF?mode=ems_copy_c

Rafael Penna
rapennas@gmail.com

bitcoinCoders bitups