به نام خدا

تمرین کامپیوتری شماره یک **مرضیه باقری‌نیا امیری / ۸۱۰۱۹۷۶۸۲**

• بخش اول

سوال ۱) کدهای مربوط به این بخش در فولدر Part#01 و فایل Q#1.py قرار دارد؛ یک نمونه از اجرای تصادفی این الگوریتم به شکل زیر است:

```
marzieh@marzieh:~/Desktop/Crypto/Part#01$ python3 Q#1.py
Private key in WIF format is:  92o2ga3hC8Uez45ZWA1kpQNbK9C7JX6gCwWsmcruzVVu6S6aEpy
Address in Base58 format is:  n4SHYqKSoSBqXou6M1GHTmV3cuVhNvDCem
```

تفاوت آدرس‌های mainnet و testnet:

testnet	mainnet	
b'\x6F	b'\x00	prefix_A
b'\x04	b'\x04	prefix_B
b'\xC4	b'\x05	SEGWIT_PREFIX
tb	bc	BECH32_PREFIX

سوال ۲) کدهای مربوط به این بخش در فولدر Part#01 و فایل Q#2.py قرار دارد؛ یک نمونه از اجرای تصادفی این الگوریتم به شکل زیر است؛ توجه شود که حروف اختصاصی آدرس، mba (مخفف اسم خودم)است:

```
marzieh@marzieh:~/Desktop/Crypto/Part#01$ python3 Q#2.py
Private key in WIF format is:  92SD5rXSKgXvtzUN45nsM9qGzTCYx6NdxxJmcsnj3EZmNNjza49
Address in Base58 format is:  mmbaoiXy4WWtYmG9emYqUXv7YeaLsa2mfD
```

سوال ۳) کدهای مربوط به این بخش در فولدر Part#01 و فایل Q#3.py قرار دارد؛ یک نمونه از اجرای تصادفی این الگوریتم به شکل زیر است:

```
marzieh@marzieh:~/Desktop/Crypto/Part#01$ python3 Q#3.py
Private key in WIF format is: 92N68ook415zuGAvNjQFWDoeVf5MRjoLkgfzo44AT9sj9qhLBkU
Public key is: 02e61341f46b529b0fac2c5e15a67af7affceb2be7544af18d14206fff041c02c0
Segwit address is: tb1qsnwc0y43fpljyl2ep0e2gtsqa55utcj4edeay6
```

مزایای آدرس Segwit:

- قابلیت malleability معامله: از آنجا که داده های witness تنها بخشی از معامله است که می تواند توسط شخص ثالث تغییر پیدا کند، حذف آن فرصت حملات malleability معامله (تغییر امضا با حفظ اصالت آن) را از بین می‌برد.
- Script Versioning: با معرفی segwit، که در آن قبل از هر locking script یک شماره نسخه اسکریپت وجود دارد، این امکان را می‌دهد که زبان اسکریپت را به روشی سازگار برای معرفی متغیرها، Syntax یا Semantic جدید ارتقا دهید.
- Scaling شبکه و ذخیره‌سازی: داده های witness کمک بزرگی به اندازه تراکنش‌ها می‌کنند؛ با انتقال داده‌های witness به خارج از تراکنش، segwit مقیاس‌پذیری BTC را بهبود می‌بخشد.
- بهینه‌سازی تأیید امضا: Segwit پیچیدگی توابع تایید امضا را از O(N^2) به O(N) کاهش می‌دهد.

- بهبود امضای آفلاین: امضاهای Segwit دارای مقدار (مقدار) ارجاع شده توسط هر ورودی در هش امضا شده هستند. از آنجا که این مبلغ اکنون بخشی از Commitment hash است که امضا شده است، یک دستگاه آفلاین نیازی به معاملات قبلی ندارد.

• بخش دوم

سوال۱) ابتدا با کمک کتابخانه bitcoinaddress به صورت تصادفی private key تولید می‌کنیم؛ سپس به کمک توابع کتابخانه bitcoinlib، public_key و آدرس معتبر شبکه btc را تولید می‌کنیم. با استفاده از آدرس تولید شده و نیز سایت https://coinfaucet.eu/en/btc-testnet، بیتکوین قابل استفاده در testnet، Hash تراکنش مربوطه و ... را دریافت می‌کنیم. سپس script_public_key مربوط به خروجی تراکنش‌ها را می‌سازیم؛ در این سوال، تراکنش اول (توسط هیچکس قابل خرج شدن نباشد) را به وسیله OP_RETURN می‌سازیم و تراکنش دوم را(توسط همه قابل خرج کردن باشد) را خالی می‌گذاریم چرا که نباید وابسته به داشتن یک آدرس (چه از نوع خصوصی و چه از نوع عمومی) باشد. سپس با استفاده از این script_public_keyها دو txout می‌سازیم. در مرحله بعدی باید script_public_key مربوط به ورودی تراکنش را بسازیم؛ که با توجه به اینکه تراکنش ما (از ظرف سایت به آدرس ما) از نوع p2pk می‌باشد، یک اسکریپت مربوط به آن که به شکل زیر است را تولید می‌کنیم:

OP_DUP, OP_HASH160, Hash160(src_public_key), OP_EQUALVERIFY, OP_CHECKSIG

پس از آن یک txin می‌سازیم؛ لازم است تا با ارائه کلید خصوصی نشان دهیم که تراکنش مربوطه متعلق به ماست. پس باید scriptSig مربوط به آن را تهیه کنیم؛ این کار را به کمک تابع از قبل آماده شده create_OP_CHECKSIG_signature انجام می‌دهیم. پس از آماده شدن امضا، لازم است تا آن را راستی‌آزمایی کنیم. بنابراین، تراکنشی با استفاده از txin و txout می‌سازیم و امضای این تراکنش را set می‌کنیم؛ سپس script_public_key را با txin، verify می‌کنیم که از معتبر بودن آن اطمینان حاصل کنیم.

اگر scriptها بدون هیچ مشکلی اجرا شوند، یعنی تراکنش از نظر ساختاری درست است؛ سپس آن را در شبکه با استفاده از API سایت BlockCypher در شبکه testnet، broadcast می‌کنیم. اگر تراکنش بدون هیچ مشکلی در شبکه منتشر شود، خروجی rawTtransaction به ما می‌دهد؛ خروجی تراکنش این سوال، هم به صورت text و هم به صورت تصویر در فولدر Part#01/tx_1/Docs در فایل‌های Image_TX و Text_TX قابل مشاهده است. همچنین می‌توان با جست‌وجوی Hash این تراکنش که هم در فایل Text_TX موجود هست و هم در ادامه قرار داده شده است، در سایت‌های مربوط به Blockchain از confirm شدن آن اطمینان یافت.

TX_Hash: 3bad52d57adc6e25c9489ed07e9e9a1ad2f4036bbfd9c9efd5038bfb08e7ef77

در قسمت دوم سوال لازم است تا خروجی دوم تراکنش را که توسط همه قابل خرج کردن است را خرج کنیم؛ بدین منظور برای تهیه script_public_key خروجی، از script مربوط به تراکنش‌های P2PK استفاده می‌کنیم و یک txtout برای آن می‌سازیم؛ در مرحله بعدی باید script_public_key مربوط به ورودی تراکنش (که هروجی تراکنش قسمت قبل است) را بسازیم؛ که با توجه به اینکه خروجی تراکنش ما هیچگونه script_public_key نداشت تا برای همه قابل خرج کردن باشد، ورودی تراکنش جدید نیز نیازی به script_public_key ندارد. پس از آن یک txin

می‌سازیم؛ با توجه به اینکه تراکنش قبلی برای همه قابل دسترس است، پس نیازی به ارائه کلید خصوصی نداریم اما برای آنکه script ما مقدار true برگرداند، scriptSig ورودی را برابر با OP_TRUE قرار می‌دهیم. در ادامه مانند سایر تراکنش‌های ScriptVerify را انجام داده و در صورتی که با خطایی مواجه نشویم، تراکنش را در testnet، broeadcast می‌کنیم؛ خروجی تراکنش این سوال، هم به صورت text و هم به صورت تصویر در فولدر Part#01/tx_1/Docs در فایل‌های Image_Spend و Text_Spend قابل مشاهده است. همچنین می‌توان با جست‌وجوی Hash این تراکنش که هم در فایل Text_ Spend موجود هست و هم در ادامه قرار داده شده است، در سایت‌های مربوط به Blockchain از confirm شدن آن اطمینان یافت.

TX_Hash: 5af210ff22d6eab33d25c77922d68ae4d17ead91f228fba8b11a4063b6c8a77a

سوال ۲) ابتدا با کمک کتابخانه bitcoinaddress به صورت تصادفی چهار private key (یکی برای ورودی و سه تا برای خروجی multiSig) تولید می‌کنیم؛ سپس به کمک توابع کتابخانه bitcoinlib، public_key و آدرس معتبر شبکه btc را تولید می‌کنیم. با استفاده از آدرس ورودی تولید شده و نیز سایت https://coinfaucet.eu/en/btc-testnet، بیتکوین قابل استفاده در testnet، Hash تراکنش مربوطه و ... را دریافت می‌کنیم. سپس script_public_key مربوط به خروجی تراکنش‌ها را می‌سازیم؛ در این سوال، تراکنش ما باید به صورت multiSig باشد بنابراین، script_public_key آن باید به شکل زیر باشد:

OP_2, dest_public_key_1, dest_public_key_2, dest_public_key_3, OP_3, OP_CHECKMULTISIG

که در آن، publickey کسانی که امضای آن‌ها برای خرج کردن تراکنش لازم است را ست می‌کنیم. سپس با استفاده از این script_public_key یک txout می‌سازیم. در مرحله بعدی باید script_public_key مربوط به ورودی تراکنش را بسازیم؛ که با توجه به اینکه تراکنش ما (از ظرف سایت به آدرس ما) از نوع p2pk می‌باشد، یک اسکریپت مربوط به آن که به شکل زیر است را تولید می‌کنیم:

OP_DUP, OP_HASH160, Hash160(src_public_key), OP_EQUALVERIFY, OP_CHECKSIG

پس از آن یک txin می‌سازیم؛ لازم است تا با ارائه کلید خصوصی نشان دهیم که تراکنش مربوطه متعلق به ماست. پس باید scriptSig مربوط به آن را تهیه کنیم؛ این کار را به کمک تابع از قبل آماده شده create_OP_CHECKSIG_signature انجام می‌دهیم. پس از آماده شدن امضا، لازم است تا آن را راستی‌آزمایی کنیم. بنابراین، تراکنشی با استفاده از txin و txout می‌سازیم و امضای این تراکنش را set می‌کنیم؛ سپس script_public_key را با txin، verify می‌کنیم که از معتبر بودن آن اطمینان حاصل کنیم.

اگر scriptها بدون هیچ مشکلی اجرا شوند، یعنی تراکنش از نظر ساختاری درست است؛ سپس آن را در شبکه با استفاده از API سایت BlockCypher در شبکه testnet، broadcast می‌کنیم. اگر تراکنش بدون هیچ مشکلی در شبکه منتشر شود، خروجی rawTtransaction به ما می‌دهد؛ خروجی تراکنش این سوال، هم به صورت text و هم به صورت تصویر در فولدر Part#01/tx_2/Docs در فایل‌های Image_TX و Text_TX قابل مشاهده است. همچنین می‌توان با جست‌وجوی Hash این تراکنش که هم در فایل Text_TX موجود هست و هم در ادامه قرار داده شده است، در سایت‌های مربوط به Blockchain از confirm شدن آن اطمینان یافت.

TX_Hash: c0947841f5babc8a692db269c907db8a03709b327a128f483da974bb889f9513

در قسمت دوم سوال لازم است تا خروجی تراکنش را که از نوع multiSig است را خرج کنیم؛ بدین منظور برای تهیه script_public_key خروجی، از script مربوط به تراکنش‌های P2PK استفاده می‌کنیم و یک txtout برای آن می‌سازیم؛ در مرحله بعدی باید script_public_key مربوط به ورودی تراکنش (که خروجی تراکنش قسمت قبل است) را بسازیم؛ که با توجه به اینکه خروجی تراکنش ما از نوع mutliSig بود، برای script_public_key آن از اسکریپت زیر استفاده می‌کنیم (که همان script_public_key خروجی تراکنش قبل است):

OP_2, my_public_key_1, my_public_key_2, my_public_key_3, OP_3, OP_CHECKMULTISIG

. پس از آن یک txin می‌سازیم؛ با توجه به اینکه تراکنش قبلی از نوع multiSig است، برای خرج کردن آن باید به تعدادی که در آن ذکر شده امضای معتبر ارائه دهیم؛ این کار را به کمک تابع از قبل آماده شده create_OP_CHECKSIG_signature انجام می‌دهیم و برای این مورد خاص، باید دو امضا از سه امضای ممکن را با استفاده از این تابع بسازیم و scriptSig را با استفاده از اسکریپت‌های زیر تهیه کنیم:

OP_0, signature_1, signature_2

در ادامه مانند سایر تراکنش‌های ScriptVerify را انجام داده و در صورتی که با خطایی مواجه نشویم، تراکنش را در testnet، broeadcast می‌کنیم؛ خروجی تراکنش این سوال، هم به صورت text و هم به صورت تصویر در فولدر Part#01/tx_2/Docs در فایل‌های Image_Spend و Text_Spend قابل مشاهده است. همچنین می‌توان با جست‌وجوی Hash این تراکنش که هم در فایل Text_ Spend موجود هست و هم در ادامه قرار داده شده است، در سایت‌های مربوط به Blockchain از confirm شدن آن اطمینان یافت.

TX_Hash: db1454e95d430f88aabbe1a8f29137333636da56df48714cb3f6883d9931e063

سوال ۳) ابتدا با کمک کتابخانه bitcoinaddress به صورت تصادفی دو private key (یکی برای ورودی و یکی برای خروجی P2SH) تولید می‌کنیم؛ سپس به کمک توابع کتابخانه bitcoinlib، public_key و آدرس معتبر شبکه btc را تولید می‌کنیم. با استفاده از آدرس ورودی تولید شده و نیز سایت https://coinfaucet.eu/en/btc-testnet، بیتکوین قابل استفاده در testnet، Hash تراکنش مربوطه و ... را دریافت می‌کنیم. سپس script_public_key مربوط به خروجی تراکنش‌ها را می‌سازیم؛ در این سوال، تراکنش ما باید به صورت P2SH باشد بنابراین، script_public_key آن باید به شکل زیر باشد:

OP_HASH160, Hash160(dest_public_key, OP_CHECKSIG), OP_EQUAL

که در آن، publickey خروجی را ست می‌کنیم. سپس با استفاده از این script_public_key یک txout می‌سازیم.

در مرحله بعدی باید script_public_key مربوط به ورودی تراکنش را بسازیم؛ که با توجه به اینکه تراکنش ما (از ظرف سایت به آدرس ما) از نوع p2pk می‌باشد، یک اسکریپت مربوط به آن که به شکل زیر است را تولید می‌کنیم:

OP_DUP, OP_HASH160, Hash160(src_public_key), OP_EQUALVERIFY, OP_CHECKSIG

پس از آن یک txin می‌سازیم؛ لازم است تا با ارائه کلید خصوصی نشان دهیم که تراکنش مربوطه متعلق به ماست. پس باید scriptSig مربوط به آن را تهیه کنیم؛ این کار را به کمک تابع از قبل آماده شده create_OP_CHECKSIG_signature انجام می‌دهیم. پس از آماده شدن امضا، لازم است تا آن را راستی‌آزمایی

کنیم. بنابراین، تراکنشی با استفاده از txin و txout می‌سازیم و امضای این تراکنش را set می‌کنیم؛ سپس script_public_key را با txin، verify می‌کنیم که از معتبر بودن آن اطمینان حاصل کنیم.
اگر scriptها بدون هیچ مشکلی اجرا شوند، یعنی تراکنش از نظر ساختاری درست است؛ سپس آن را در شبکه با استفاده از API سایت BlockCypher در شبکه testnet، broadcast می‌کنیم. اگر تراکنش بدون هیچ مشکلی در شبکه منتشر شود، خروجی rawTtransaction به ما می‌دهد؛ خروجی تراکنش این سوال، هم به صورت text و هم به صورت تصویر در فولدر Part#01/tx_3/Docs در فایل‌های Image_TX و Text_TX قابل مشاهده است. همچنین می‌توان با جست‌وجوی Hash این تراکنش که هم در فایل Text_TX موجود هست و هم در ادامه قرار داده شده است، در سایت‌های مربوط به Blockchain از confirm شدن آن اطمینان یافت.
TX_Hash: 5bf6358f7b3077c65eefa205623cfa66d94f4cc9417e18d563b18c77e0939440

در قسمت دوم سوال لازم است تا خروجی تراکنش را که از نوع P2SH است را خرج کنیم؛ بدین منظور برای تهیه script_public_key خروجی، از script مربوط به تراکنش‌های P2PK استفاده می‌کنیم و یک txtout برای آن می‌سازیم؛ در مرحله بعدی باید script_public_key مربوط به ورودی تراکنش (که خروجی تراکنش قسمت قبل است) را بسازیم؛ که با توجه به اینکه خروجی تراکنش ما از نوع P2SH بود، برای script_public_key آن از اسکریپت زیر استفاده می‌کنیم (که همان script_public_key خروجی تراکنش قبل است):
OP_HASH160, Hash160(src_public_key, OP_CHECKSIG), OP_EQUAL
. پس از آن یک txin می‌سازیم؛ با توجه به اینکه تراکنش قبلی از نوع P2SH است، برای خرج کردن آن باید امضای معتبر ارائه دهیم؛ این کار را به کمک تابع از قبل آماده شده create_OP_CHECKSIG_signature انجام می‌دهیم و برای این مورد خاص، باید امضای signature را با استفاده از این تابع بسازیم و scriptSig را با استفاده از اسکریپت‌های زیر تهیه کنیم:
signature, CScript([src_public_key, OP_CHECKSIG])
در ادامه مانند سایر تراکنش‌های ScriptVerify را انجام داده و در صورتی که با خطایی مواجه نشویم، تراکنش را در testnet، broeadcast می‌کنیم؛ خروجی تراکنش این سوال، هم به صورت text و هم به صورت تصویر در فولدر Part#01/tx_3/Docs در فایل‌های Image_Spend و Text_Spend قابل مشاهده است. همچنین می‌توان با جست‌وجوی Hash این تراکنش که هم در فایل Text_ Spend موجود هست و هم در ادامه قرار داده شده است، در سایت‌های مربوط به Blockchain از confirm شدن آن اطمینان یافت.
TX_Hash: 4348b2f15ce5a01ac9caa44bcda495d8a37e1ddc823c2f5933fe8fc94b1044e7

سوال ۴) ابتدا با کمک کتابخانه bitcoinaddress به صورت تصادفی دو private key (یکی برای ورودی و یکی برای خروجی segWit) تولید می‌کنیم؛ سپس به کمک توابع کتابخانه bitcoinlib، public_key و آدرس معتبر شبکه btc را تولید می‌کنیم. با استفاده از آدرس ورودی تولید شده و نیز سایت https://coinfaucet.eu/en/btc-testnet، بیتکوین قابل استفاده در testnet، Hash تراکنش مربوطه و ... را دریافت می‌کنیم و همچنین برای خروجی یک آدرس segWit تولید می‌کنیم؛ سپس script_public_key مربوط به خروجی تراکنش‌ها را می‌سازیم؛ در این سوال، تراکنش ما باید به صورت segWit باشد بنابراین، script_public_key آن باید به شکل زیر باشد:

OP_0, Hash160(dest_public_key)

که در آن، publickey خروجی را ست می‌کنیم. سپس با استفاده از این script_public_key یک txout می‌سازیم.

در مرحله بعدی باید script_public_key مربوط به ورودی تراکنش را بسازیم؛ که با توجه به اینکه تراکنش ما (از ظرف سایت به آدرس ما) از نوع p2pk می‌باشد، یک اسکریپت مربوط به آن که به شکل زیر است را تولید می‌کنیم:

OP_DUP, OP_HASH160, Hash160(src_public_key), OP_EQUALVERIFY, OP_CHECKSIG

پس از آن یک txin می‌سازیم؛ لازم است تا با ارائه کلید خصوصی نشان دهیم که تراکنش مربوطه متعلق به ماست. پس باید scriptSig مربوط به آن را تهیه کنیم؛ این کار را به کمک تابع از قبل آماده شده create_OP_CHECKSIG_signature انجام می‌دهیم. پس از آماده شدن امضا، لازم است تا آن را راستی‌آزمایی کنیم. بنابراین، تراکنشی با استفاده از txin و txout می‌سازیم و امضای این تراکنش را set می‌کنیم؛ سپس script_public_key را با txin، verify می‌کنیم که از معتبر بودن آن اطمینان حاصل کنیم.

اگر scriptها بدون هیچ مشکلی اجرا شوند، یعنی تراکنش از نظر ساختاری درست است؛ سپس آن را در شبکه با استفاده از API سایت BlockCypher در شبکه testnet، broadcast می‌کنیم. اگر تراکنش بدون هیچ مشکلی در شبکه منتشر شود، خروجی rawTtransaction به ما می‌دهد؛ خروجی تراکنش این سوال، هم به صورت text و هم به صورت تصویر در فولدر Part#01/tx_4/Docs در فایل‌های Image_TX و Text_TX قابل مشاهده است. همچنین می‌توان با جست‌وجوی Hash این تراکنش که هم در فایل Text_TX موجود هست و هم در ادامه قرار داده شده است، در سایت‌های مربوط به Blockchain از confirm شدن آن اطمینان یافت.

TX_Hash: 26cae739160ddd0027fd97af54c084cafe9ba047a3e8346e3932380b932b05d6

در قسمت دوم سوال لازم است تا خروجی تراکنش را که از نوع segWit است را خرج کنیم؛ بدین منظور برای تهیه script_public_key خروجی، از script مربوط به تراکنش‌های P2PK استفاده می‌کنیم و یک txtout برای آن می‌سازیم؛ در مرحله بعدی باید script_public_key مربوط به ورودی تراکنش (که خروجی تراکنش قسمت قبل است) را بسازیم؛ که با توجه به اینکه خروجی تراکنش ما از نوع segWit بود، برای script_public_key آن از اسکریپت زیر استفاده می‌کنیم (که همان script_public_key خروجی تراکنش قبل است):

OP_0, Hash160(src_public_key)

پس از آن یک txin می‌سازیم؛ در این قسمت دیگر نیازی به ارائه امضا در ساخت تراکنش نداریم بلکه آن را در witness اضافه می‌کنیم. در ادامه تراکنش را در testnet، broeadcast می‌کنیم؛ خروجی تراکنش این سوال، هم به صورت text و هم به صورت تصویر در فولدر Part#01/tx_4/Docs در فایل‌های Image_Spend و Text_Spend قابل مشاهده است. همچنین می‌توان با جست‌وجوی Hash این تراکنش که هم در فایل Text_ Spend موجود هست و هم در ادامه قرار داده شده است، در سایت‌های مربوط به Blockchain از confirm شدن آن اطمینان یافت.

TX_Hash: de6db2fb248f1c5ebb865fc294b90e4cc09a8fce8e21cb2927e0a9f63768440a

- **بخش سوم**

ابتدا باید تراکنش Coinbase را بسازیم؛ همانند آنچه در بخش دوم دیدیم، تراکنش را با خروجی از نوع P2PK به آدرس خودمان می‌سازی. همچنین متن خواسته شده را که HEX آن به شکل زیر است در scriptSig می‌نویسیم. سپس از hash این تراکنش که در واقع همان هش merkle root است + hash بلوک 7682 (چهار رقم شماره دانشجویی بنده) + را به همراه یک nonce متغیر هربار Hash می‌گیریم تا از target مورد نظر ما کمتر شود. در پیاده‌سازی، Difficultyt متغیر در نظر گرفته شده است تا به مقدار دلخواه تغییر یابد. مستندات اجرای این کد، در folder مربوط به Part#03 در فایل output و نیز تصویر output قابل مشاهده است؛ همانطور که مشاهده می‌کنید، استخراج با Difiiculty = ۸، نزدیک به دو ساعت زمان برده است بنابراین بنده اصلا نتوانستم با ۱۶ صفر در ابتدا آن را اجرا کنم.

```
marzieh@marzieh:~/Desktop/Crypto/Part#03$ python3 spend.py
start mining
end mining. Mining took: 6811.108483076096 seconds

Hash: 00000000b964bcb661cab7c86814cdc314c5cefedaf82c08deed0dbad6ed4045
Timestamp: 2021-05-28 19:38:11.008186
Miner: MBA
Number of Transactions: 1
Difficulty: 8
Merkle root: b3f060e3fd437805e1675770db152caf117c0b120fc5ac487804a4361ead3029
Nonce: 2873122540
Block Reward: 6.25 BTC
Fee Reward: 0 BTC
marzieh@marzieh:~/Desktop/Crypto/Part#03$
```