مبانی فناوری بلاکچین و رمزارزها

نیم‌سال اول ۱۳۹۹-۱۴۰۰

مدرس: دکتر محمد علی مداح علی
دانشکده مهندسی برق

تمرین عملی سری دوم

نام و نام‌خانوادگی: عرفان نصرتی
شماره دانشجویی: ۹۷۱۰۲۵۵۸

مقدمه

در این تمرین مبانی اسکریپت زدن را در بیت‌کوین بررسی می‌کنیم.

۱

در این سوال ۳ روش برای قفل کردن یک UTXO می‌پردازیم که با شرایط مختلفی باز می‌شوند. ۱. در این روش ابتدا چک می‌کنیم که آیا امضای فراز وحود دارد یا خیر اگر بود که تراکنش انجام می‌شود اگر نبود وارد ELSE می شویم و سپس چک می‌کنیم که امضای عطا همراه ۳ نفر دیگر از سهام داران درست هستند. برای این کار ابتدا امضای عطا را چک کرده سپس از ۵ سهام دار کلید ۳ سهام دار را چک می‌کنیم. البته برای باز کردن دستوری که چند امضا را چک می‌کند یک باگ دارد که استک آخر را اضافی بر میدارد.

```
######################################################################
# TODO: Complete the scriptPubKey implementation for Exercise 2
Q31a_txout_scriptPubKey = [OP_DUP, faraz_public_key, OP_CHECKSIG, OP_IF, OP_1, OP_ELSE, ata_public_key, OP_CHECKSIGVERIFY, OP_3,
    shar5_public_key, shar4_public_key, shar3_public_key, shar2_public_key, shar1_public_key,
    OP_5, OP_CHECKMULTISIG, OP_ENDIF]
######################################################################
```

۲. در این روش دو روش برای باز کردن وجود دارد یا عطا و فراز هر دو امضا کنند یا فراز یا عطا و ۳ سهام دار دیگر از ۵ سهام دار و یا ۵ سهام دار برای این کار یک IF تو در تو می‌زنیم و با روش .MULTISIG

```
Q32a_txout_scriptPubKey = [OP_2DUP ,OP_2 ,faraz_public_key , ata_public_key , OP_2 , OP_CHECKMULTISIGVERIFY, OP_IF ,OP_1 ,OP_ELSE,
                           OP_2OVER , OP_2OVER ,faraz_public_key , ata_public_key , OP_2 , OP_CHECKMULTISIGVERIFY , OP_3,
                           shar5_public_key, shar4_public_key, shar3_public_key, shar2_public_key, shar1_public_key,
                           OP_5, OP_CHECKMULTISIGVERIFY , OP_IF ,OP_1 ,OP_ELSE , OP_5,
                           shar5_public_key, shar4_public_key, shar3_public_key, shar2_public_key, shar1_public_key,
                           OP_5, OP_CHECKMULTISIGVERIFY , OP_ENDIF , OP_1 , OP_ENDIF , OP_1 ]
```

۳.در روش سوم که استفاده از شرط مجاز نیست برای انجام آن ابتدا با روش MULTISIG از بین فراز و عطا یکی را چک کرده و از بین ۵ سهام دار باقی مانده ۳ رمز را چک می‌کنیم .

اما چون ماینر ها تایید کردن چند رمز را انجام نمی‌دهند و آن تراکنش را در بلوک خود قرار نمی‌دهند تراکنش ها verify نمی شود. دو قسمت اول سوال همان طور که توضیح داده شد بر اساس چک کردن شرط و چند کلید و قسمت آخر فقط توسط MULTISIG زده شد است.

```
Q33a_txout_scriptPubKey = [OP_1, faraz_public_key, ata_public_key, OP_2 , OP_CHECKMULTISIGVERIFY , OP_3,
    shar5_public_key, shar4_public_key, shar3_public_key, shar2_public_key, shar1_public_key,
    OP_5, OP_CHECKMULTISIG]
```

۴ ۲

در این سوال اول استک رشته ای که می‌خواهیم در بلاکچین بماند را میزنیم سپس از دستور $OP - RETURN$ استفاده می‌کنیم که همواره False بر میگرداند و برای proof of burn استفاده می‌شود.

```
input_string = input(" Please send your test: ").encode('UTF-8')
################################################################################
# TODO: Complete the scriptPubKey implementation for Exercise 2
Qsm_txout_scriptPubKey = [ OP_RETURN ,input_string]
```

۵ ۳

برای این سوال از مفهوم nlocktime استفاده می‌کنیم در این صورت سعید می‌تواند پول را به حساب حامد واریز کند اما در ابتدای اسکریپ یک شرط وجود دارد که چک می‌کند آیا زمان آن فرا رسیده است یا خیر اگر زمان آن فرا نرسیده بود حتی اگر حامد هم بخواهد آن را باز کند نمی‌تواند اما اگر زمان آن فرا برسد (حالا این زمان می‌تواند بر اساس UNIX یا بر اساس تعداد بلوک های ماین شده یا به عبارت دیگر ارتفاع بلوک باشد) تراکنش قابل استفاده برای حامد خواهد بود و می تواند پول را برداشت کند. تنها فرقی مه در اسکریپت آن وجود دارد با یک اسکریپت استاندارد این است که تنها زمان به اول آن اضافه شده است همان طور که در شکل دیده می‌شود. همان طور که در قسمت اول توضیح داده شد این اسکریپ تنها زمان را در

```
hamed_secretkey = CBitcoinSecret(
    'cQsSAgAUL5wLtW65freyTcsTd4TQkodhLQwMgQogDZjZ8Qga2JLP')
hamed_publickey = hamed_secretkey.pub
hamed_address = P2PKHBitcoinAddress.from_pubkey(hamed_publickey)
################################################################################
# TODO: Complete the scriptPubKey implementation for Exercise 2
Q5a_txout_scriptPubKey = [
    1606925730, OP_CHECKLOCKTIMEVERIFY, OP_DROP, OP_DUP, OP_HASH160, hamed_address, OP_EQUALVERIFY, OP_CHECKSIG]
################################################################################
```

ابتدا چک می کند برای همین برای باز کردن آن تنها کافیست حامد پس از فرا رسیدن زمان کلید عمومی و امضای خود را ارائه کند تا بتواند UTXO را باز کند.

۶ ۴

در این سوال سعید یک ودیعه به حامد داده است. به این صورت که اگر هر دوی آنها موافق باشند تراکنش در هر زمانی باز می‌شود. اما اگر امضای هر دوی آنها وجود نداشت ابتدا چک می‌شود که زمانی که باید برای آزاد شدن UTXO سپری می‌شد سپری شده است یا خیر اگر نشده بود که سعید نمی‌تواند به پول ها دسترسی داشته باشد. اما اگر زمان آن نرسیده باشد سعید نمی‌تواند آن UTXO را باز کند . برای حل این سوال به این گونه عمل می‌کنیم که در ابتدا چک می‌کنیم که آیا هر دو موافق باز شدن UTXO هستند یا خیر . اگر هر دو موافق باز شدن بود که تراکنش باید باز شود و این شرط اولیه ماست اگر هردو موافق نبودن یک حالت دیگر نیز وجود دارد که باید UTXO باز شود و آن این است که از موعد قرار گذشته باشد و بخوایم سعید بتواند با ارائه کلید عمومی خود آن را باز کند. که برای این کار از شرط استفاده کردیم و یک زمان خیلی دور هم به آن دادیم که تنها در صورتی که هر دو امضا کنند UTXO باز شود. هماطور که در بالا می

```
Q6a_txout_scriptPubKey = [OP_2 , saeed_public_key , hamed_public_key , OP_2 , OP_CHECKMULTISIGVERIFY ,OP_1, OP_IF , OP_1 , OP_ELSE
                , 1607097608 , OP_CHECKLOCKTIMEVERIFY , saeed_private_key , OP_CHECKSIG ,OP_ENDIF]
```

بینیم بعد OP-CHECKMULTISIGVERIFY یک عدد ۱ قرار داده شده است زیرا اگر امضای هردو درست باشد هیچ شرطی برای اینکه IF آن را چک کند نمی ماند و اگر کلید ها غلط باشد که فیل می‌شود پس آن یک برای این است که اگر امضا ها به تراکنش می‌خورد شرط صادق باشد و تراکش باز شود عبارت بعد از IF هم که شرط رسیدن زمان باز شدن تراکنش و امضای سعید را چک می‌کند.

۷ ۵

در این قسمت باید کاری می‌کردیم که تراکنش ها به ۳ تراکنش تبدیل شوند در ابتدا مقدار amount را تقسیم بر ۳ کردیم و سپس ۳ تا Txout درست کردیم که در این صورت txout خودش لیست شد و دیگر به اکولاد دور آن نیازی نبود زیرا با این حرکت به لیست تبدیل شده بود و دیگر نیازی به لیست کردن آن نبود. و سپس بقیه توابعی که از txout استفاده کرده بودند را اصلاح کردم.

```
sender_public_key = sender_private_key.pub
sender_address = P2PKHBitcoinAddress.from_pubkey(sender_public_key)

txout = [create_txout(amount_to_send/3, txout_scriptPubKey)]*3

txin_scriptPubKey = P2PKH_scriptPubKey(sender_address)
txin = create_txin(txid_to_spend, utxo_index)
txin_scriptSig = P2PKH_scriptSig(txin, txout, txin_scriptPubKey, # need t change due to the change of the txout
    sender_private_key, sender_public_key)

new_tx = create_signed_transaction(txin, txout, txin_scriptPubKey,          # need t change due to the change of the txout
                                   txin_scriptSig)

return broadcast_transaction(new_tx, network)
```

۸ ۶

در سوال هشت که کمی پیچیده تر از بالا بود باید توابع را طوری تغییر میدادم که اولا بتوانند امضای لازم برای هر کدام از Index ها را فراهم کنند و به یک تراکنش آن را بفرستند.

۹ ۷

در سوال ۹ باید یک راه ایجاد می‌کردیم که به وسیله آن سالار فایل خود را یک کلید خصوصی سازد و کلید عمومی و آدرس مرتبط آن را بدست آورد. برای انجام دادن این کار ابتدا فایل data.hex را باز کرده و Hash می گیریم این کار به وسیله ی دستور و تابع

$$datafile = open('data.hex','rb')$$

$$defhashfile() :$$

$$file - hash = hash.sha256(datafile.read()).digest()$$

$$returnfile - hash$$

```
pri_key = CBitcoinSecret.from_secret_bytes(hashfile())
pub_key = pri_key.pub
address = P2PKHBitcoinAddress.from_pubkey(pub_key)
```

حال ما Hash مورد نیاز را بدست آورده ایم و کافیست از آن کلید عمومی و آدرس را بدست آوریم که این توسط کد زیر انجام می‌شود. تا به آدرس فوق فرستاده شود اسکریپت تراکنش یک اسکریپت ساده سوال یک است تنها آدرس در اینجا نحوه ی ساختی متفاوت دارد. و سالار بعد از منتشر شدن مقاله اش می‌تواند با دادن کلید خصوصی محمد و همچنین Hash گرفتن از مقاله نوشتن مقاله در آن زمان را ثابت کند.

قسمت دوم) در این قسمت برای اینکه چندین فایل را بتوان با هم تایید کرد از روش merkle root استفاده می‌کنیم و این بار به جا اینکه از هر یک از فایل ها جداگانه Hash بگیریم ابتدا merkle root را حساب کرده و با داشتن آن عملیات بالا را دوباره تکرار می‌کنیم. و از روش ثابت کردن merkle root سالار می تواند داشتن همه فایل ها را ثابت کند.

۱۰ ۸

در این سوال ما باز هم با سوالی مواجه هستیم که برای باز کردن دو شرط وجود دار پس در این باید از ساختار شرطی استفاده کنیم و باید دو شرط چک شود . در حالت اول Alice می‌تواند پول خود(BCT) را آزاد کند اما چون با آزاد کردن پول خود مقدار x لو می‌رود Bob نیز میتواند پول خود(BCY) بردارد. در حالت دوم امضای هردو لازم است تا تراکنش باز شود و در این صورت نیز باز هر دو می‌توانند پول تراکنش را در ارز دیگر باز کنند و پول خود را در این دو فضا جابه‌جا کنند. حال پس ابتدا مقدار داده شده با Hash مقدار محرمانه مقاسه میشود اگر برابر بود حال باید ببینم حود Alice است که می خواهد تراکنش را باز کند یا خیر پس امضای او را چک می‌کنیم در حالت بعدی اگر Hash برابر نبود باید امضای هر دو باشد پس امضای باب را چک می‌کنیم و امضای Alice نیز چون بیرون از If است چک میشود و روش دوم نیز ساخته می‌شود. حال برای باز کردن تراکنش یا باید مقدار X و امضا Alice موجود باشد یا امضای هردو که با عوض کردن تابع

```
# This is the ScriptPubKey for the swap transaction
def coinExchangeScript(public_key_sender, public_key_recipient, hash_of_secret):
    return [OP_DUP,OP_HASH160 , hash_of_secret , OP_EQUAL , OP_IF , OP_DROP , OP_ELSE ,  public_key_sender, OP_CHECKSIGVERIFY,OP_ENDIF,
        public_key_recipient , OP_CHECKSIG]
        # fill this in!
```

alice-redeems می‌توانیم بفهمیم که کدام روش باز کردن استفاده می‌شود در روش اول باید X و امضای عالیس باشد و در روش دوم امضای هر دو.

```
# This is the ScriptSig that the receiver will use to redeem coins
def coinExchangeScriptSig1(sig_recipient, secret):
    return [sig_recipient , secret]

# This is the ScriptSig for sending coins back to the sender if unredeemed
def coinExchangeScriptSig2(sig_sender, sig_recipient):
    return [sig_recipient , sig_sender ]
##################################################################
```