بسم الله الرحمن الرحیم

استاد: محمد علی مداح علی

درس: بلاک چین

شرح محتوا: گزارش تمرین عملی سری ۲

نگارنده: امیرحسین رستمی

شماره دانشجویی: ۹۶۱۰۱۶۳۵

دانشگاه صنعتی شریف

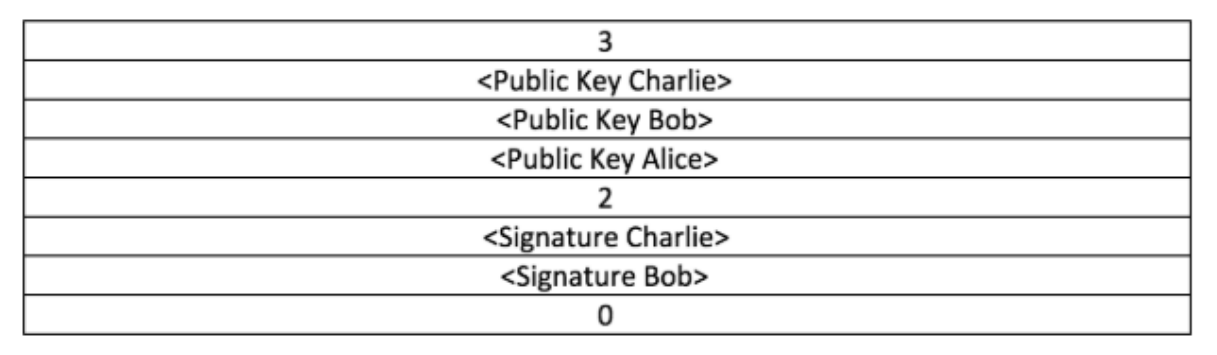
پاییز ۹۹

سوال سوم:

برای طراحی اسکریپت برای هندل کردن شرایط مطرحی صورت سوال از دستور OP\_IF استفاده می کنیم.از جمله دستورات مهمی که جهت کاهش تعداد خطوط اسکریپت استفاده کردیم دستور OP\_DEPTH است که بیانگر تعداد اعضا موجود در استک است و به کمک تعداد آیتم های موجود در استک(push شده توسط unlock script) متوجه می شویم که در کدام condition قرار داریم و طبق آن به بررسی احراز های لازم می پردازیم.

نکته مهمی که در استفاده از OP\_ CHECKMULTISIG وجود دارد این است که هنگام خواندن از استک با فرض اینکه به تعداد m از n کلید نیاز است،داریم که باید اولا m تا signature قرار بدهیم سپس عدد m را قرار بدهیم و سپس n تا کلید عمومی قرار بدهیم و روی آن نیز عدد n را بگذاریم،اما این کافی نیست و به خاطر وجود یک باگ در اجرا داریم که یک عدد دیگر از استک pop می شود لذا لازم است که در ابتدا هم یه x عه دلخواه در استک تزریق کنیم.یعنی در اصل برای احراز m عدد امضا توسط n عدد کلیدعمومی نیاز است تا m+n+2+1 عدد آیتم در استک قرار بدهیم.

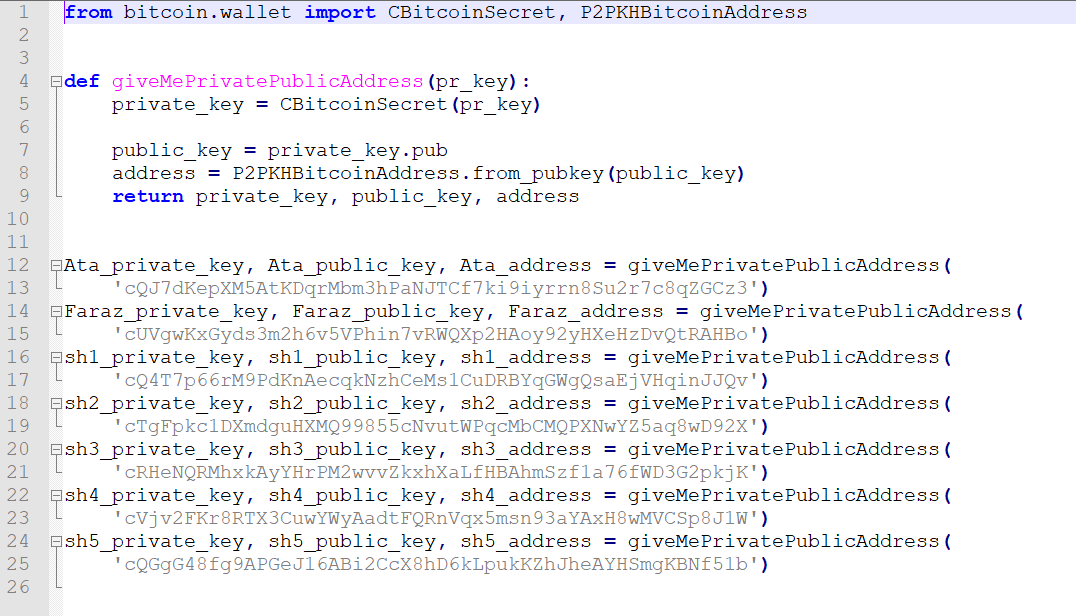
ساختاری مشابه تصویر زیر:(برای حالتی که m=2 و n=3 باشد).



آن x که پیشتر مطرح کردیم در تصویر فوق توسط OP\_0 در ابتدای استک تزریق می شود.

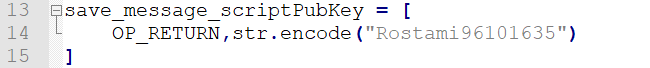
توجه:در این سوال برای هر کدام از سهامداران(عطا فراز و ۵ سهام دار دیگر) جفت کلید های private و public تولید کردیم و همگی در فایل ex3\_values قرار دارند که به صورت library در ابتدا import شده اند.

جفت کلید های (عمومی،خصوصی) به صورت زیر است:



سوال چهارم:

در این سوال جهت قرار دادن String دلخواه داخل بلاکچین از اسکریپت زیر استفاده می کنیم:



توجه: رشته جهت قراردهی روی شبکه نیاز دارد تا encode شود(به صورت byte) و قرار دادن خام آن ارسال را دچار اشکال می کند.

با جست و جو کردن Hash تراکنش حاصل در blockCypher محتوای نوشته شده را ملاحظه می کنیم:



سوال پنجم:

در این سوال از دستور CLTV استفاده کردیم(ClockLockTimeVerify) و داریم که خروجی تراکنش تا هنگامی که زمان موردنظر نرسیده است،قابل استفاده نیست.

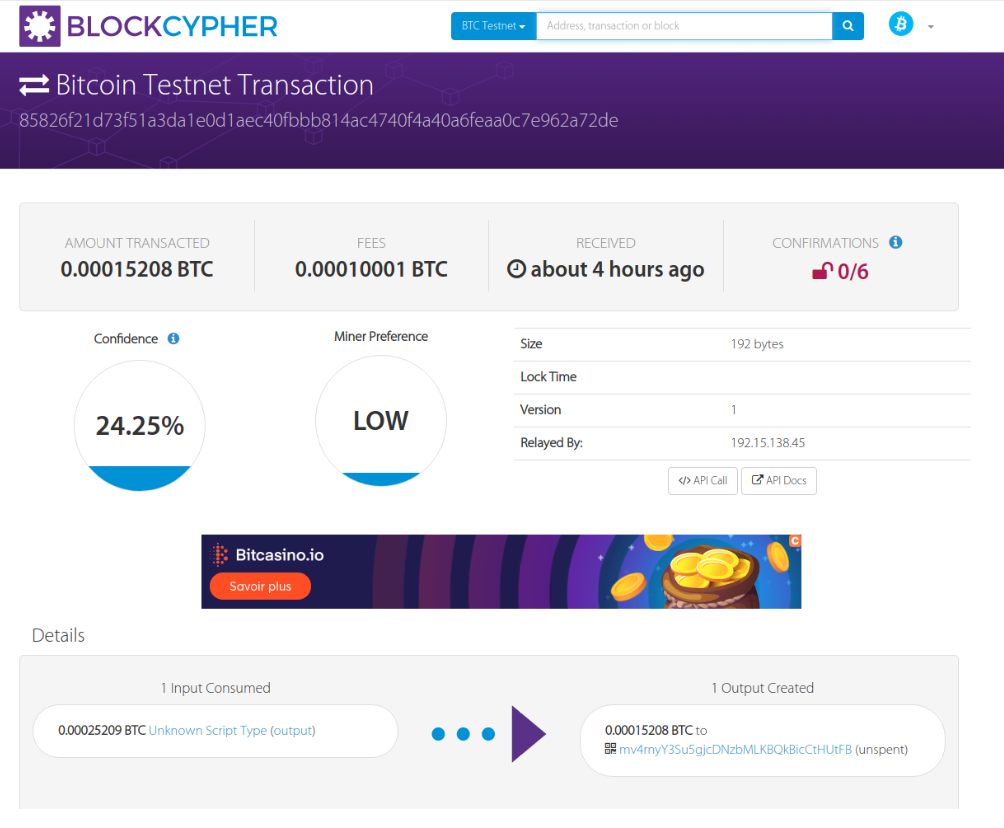
اگر

* زمان زیر 500M باشد -> "بلاک" مدنظر را نشان می دهد.
* زمان بالاتر از 500M باشد -> تاریخ مشخصی را نشان می دهد.

جهت تعیین زمان از سایتِ <https://www.unixtimestamp.com/index.php> استفاده کرده ام و همانطور که می دانید این عدد برابر است با تعداد ثانیه های گذشته شده از ابتدای سال 1970 میلادی است.

توجه کنید که تراکنش قسمت a در شبکه mine گشته و confirm می گردد اما تراکنش b به صورت unconfirmed باقی می ماند.

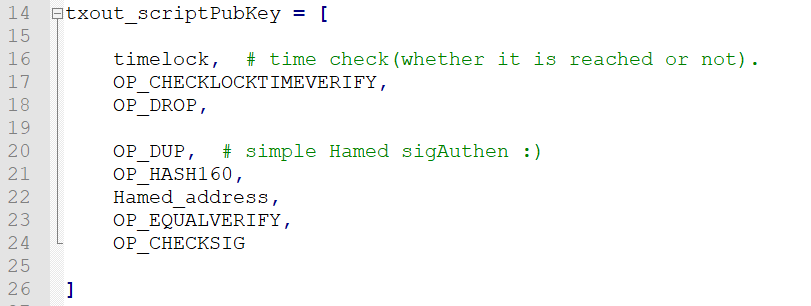
وضعیت تراکنش b:



توجه:در این سوال برای Hamed جفت کلید private و public تولید کردیم به همراه متغیر timelock در فایل ex5\_values قرار داده ایم که به صورت library در ابتدا import شده اند.

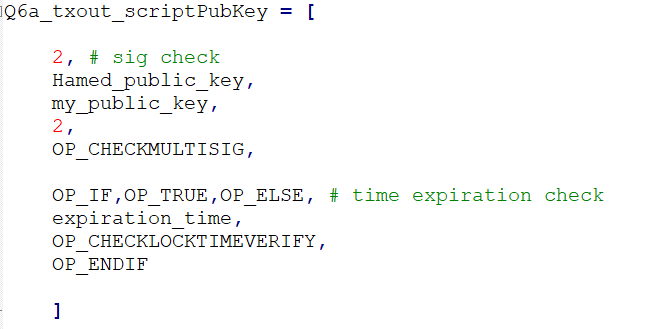
اسکریپت این قسمت از دو تکه تشکیل شده است:

1. احراز اینکه زمان بازگشایی فرارسیده است یا خیر
2. احراز هویت مصرف کننده



سوال ششم:

همانند سوال قبل در این سوال جهت احراز رخ داد expiration،از CLTV استفاده می کنیم.توجه کنید که داریم که اسکریپت این سوال از دو بخش تشکیل شده است:



ابتدا unlock script احراز اصالت می گردد.

* اگر اوکی بود تراکنش قابلیت redeem شدن می یابد.
* اگر اوکی نبود سراغ بررسی رخداد expiration می رویم:
  + 1. اگر rent period is over رخ داده بود:

تراکنش قابلیت redeem شدن می یابد.

* + 1. اگر rent period isn’t over:

تراکنش قابل redeem شدن ندارد.

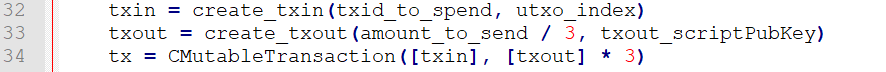
توجه:در این سوال برای Hamed جفت کلید private و public تولید کردیم و در کنار متغیر expiration\_time در فایل ex6\_values قرار داده ایم که به صورت library در ابتدا import شده اند.

سوال هفتم:

در این سوال تابع send\_from\_P2PKH\_transaction() را با پیاده سازی خود در فایل ex7.py قرار داده ایم،در این تابع که خواسته سوال را اجرا می کند مراحل زیر را پله پله انجام می دهیم:

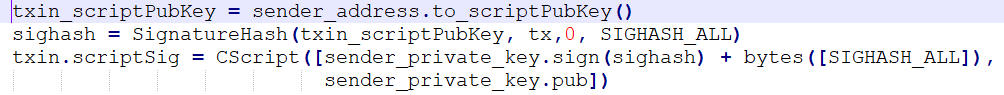
* ساختِ txin مشابه سوالات قبلی انجام می گیرد.
* یک عدد txout با amount\_to\_send ای برابر با کل مبلغ قابل انتقال ایجاد می کنیم.
* به تعداد 3 عدد از txout تولیدی مرحله قبل در tx قرار می دهیم.

سه مرحله ذکر شده در کد به صورت سه خط کد زیر اجرا می گردد.



* مشابه ایجاد تراکنش با یک عدد ورودی به یک عدد خروجی،به تزریق بخش های scriptPubKeyو sigHash به txin می پردازیم.

این مرحله به صورت قطعه کد زیر انجام می گیرد:



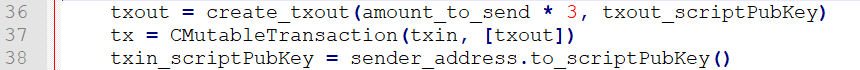
* در نهایت صحت تراکنش به وجود آمده شده را با تابع VerifyScript بررسی کرده و نتیجه را در شبکه broadcast می کنیم.

سوال هشتم:

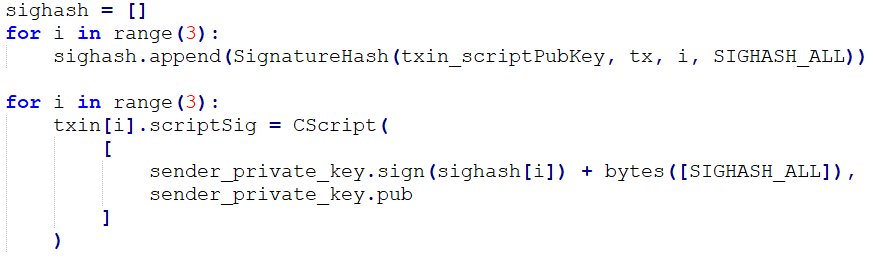
در این سوال تابع send\_from\_P2PKH\_transaction() را با پیاده سازی خود در فایل ex8.py قرار داده ایم،پیاده سازی کاملا مشابه با سوال هفتم است با این تفاوت که کار هایی که در سوال قبلی برای txout کردیم را اینجا برای txin انجام می دهیم،در این تابع که خواسته سوال را اجرا می کند مراحل زیر را پله پله انجام می دهیم:

* یک آرایه سه تایی از txin ایجاد می کنیم.
* یک عدد txout با amount\_to\_send ای برابر با مبلغ هر txin ایجاد می کنیم.
* به کمک دو مورد بالا tx را آماده می کنیم.

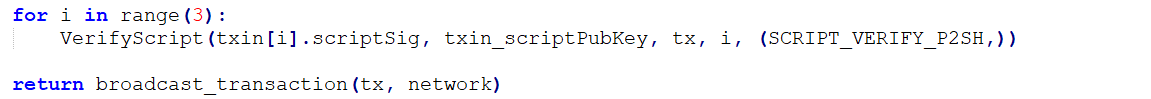
سه مرحله ذکر شده در کد به صورت سه خط کد زیر اجرا می گردد.



* برای هر سه txin موجود در آرایه txin به تزریق بخش های scriptPubKeyو sigHash به txin می پردازیم.این مرحله به صورت قطعه کد زیر انجام می گیرد:



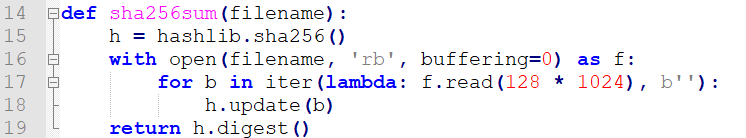
* در نهایت صحت تراکنش به وجود آمده شده را با تابع VerifyScript بررسی کرده و نتیجه را در شبکه broadcast می کنیم.(توجه کنید که Verify کردن script باید برای هر کدام از txin ها انجام گیرد).



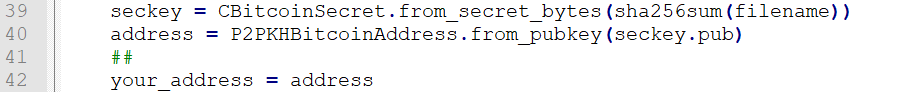
سوال نهم:

* fileverify.py

در حالت fileverify.py ابتدا به کمک تابع زیر مقدار Hash فایل data.hex موجود را خوانده و Hash آن را محاسبه می کنیم:



حال با درنظر گرفتن این محتوای Hash ِ به دست آمده به عنوان private\_key از روی آن کلید عمومی و آدرس کیف پول را محاسبه می کنیم:



باقی کار کاملا مشابه با issue کردنِ یک تراکنش ساده است.

* multifileverify.py

در این قسمت از merkleTree استفاده می کنیم و به root آن برسیم تا بتوانیم هرلحظه احراز های لازم را انجام بدهیم.(توجه کنید که واضحا چون سوال اشاره به میزانِ order زمان/پیغام نکرده است می توان از سناریو های دیگری استفاده کرد مثلا concat کردن همه فایل ها و استفاده از Hash آن ...) ،همانطور که به خاطر دارید این کار را در تمرین قبل مفصلا انجام داده ایم و لذا کد تولید merkleTree از تمرین قبل را در فایلی به نام merkle\_tree.py آورده ایم و در این حالت از آن استفاده می کنیم.

سوال دهم:

خواسته های سوال را به کمک OP\_IF ارضا می کنیم.

* سناریو اول:

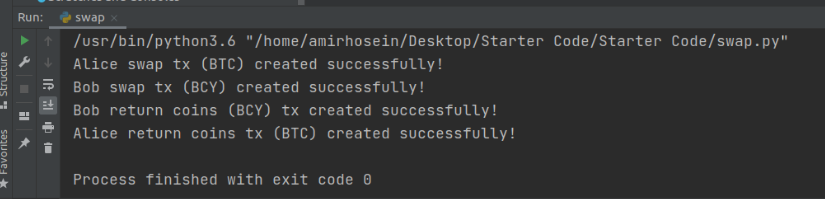
اگر شرط if درست بود،در ابتدا HASH160 مقدارX ورودی با secret مقایسه می شودکه اگر درست باشد،امضای نفر دوم با key\_public او مقایسه شده و verify می شود.

* سناریو دوم:

اگر شرط if درست نبود،به امضای هردو نفر نیاز است.بنابراین از CHECKMULTISIG استفاده می کنیم. همانطور که پیشتر گفتیم به علت باگ اجرایی CHECKMULTISIG نیاز است تا یک عدد متغیر به دلخواه(که در اینجا OP\_0 است) به ابتدای script اضافه شود.

خروجی های در حالت به شرح زیر گردید:

* Alice Redeems = False



* Alice Redeems = True

