

تراکنش

مقدمه

تراکنش (transaction) مهم‌ترین بخش از بیت‌کوین است. هر چیزی که در سیستم بیت‌کوین وجود دارد، برای آن طراحی شده تا ایجاد، انتشار، اعتبارسنجی و سرانجام اضافه شدن تراکنش‌ها به دفتر کل (بلاک‌چین) را تسهیل و نضمین کند. تراکنش یک ساختمان داده است که انتقال یک مبلغ معین را بین دو طرف مبادله در سیستم بیت‌کوین کُردگاری می‌کند. هر تراکنش یک مدخل (درایه) عمومی در بلاک‌چین بیت‌کوین (دفتر کل حسابداری دوبل جهانی) است.

در این فصل آشکال مختلف تراکنش‌های بیت‌کوین را تشریح می‌کنیم، نشان می‌دهیم هر یک از آنها چه محتویاتی دارند، چگونه ایجاد می‌شوند، چگونه اعتبارسنجی می‌شوند، و چگونه به یک رکورد دائمی در بلاک‌چین (دفتر کل تراکنش‌ها) تبدیل می‌شوند. وقتی در این فصل از اصطلاح «کیف پول» استفاده می‌کنیم، منظورمان نرم‌افزاری است که تراکنش‌ها را ایجاد می‌کند، نه یک پایگاه داده از کلیدها (آن گونه که در فصل قبل دیدیم).

بررسی تشریحی یک تراکنش

در فصل ۲ تراکنش خرید یک فنجان قهوه توسط آليس از یک کاوشگر بلاک (block explorer) مشاهده کردیم؛ شکل ۱-۶ را بینید. کاوشگر بلاک نشان می‌دهد که یک تراکنش از «آدرس» آليس به «آدرس» باب صورت گرفته است. اما این فقط یک نمای ساده از آن چیزی است که در واقع اتفاق افتاده است. در حقیقت (همان طور که در این فصل خواهیم دید)، قسمت اعظم اطلاعاتی که توسط کاوشگر بلاک نمایش داده می‌شود، توسط خود کاوشگر استخراج شده و جزئی از این تراکنش نیست.

پشت صحنه‌ی تراکنش

در پشت صحنه، یک تراکنش واقعی با آنچه در کاوشگر بلاک (شکل ۱-۱) می‌بینید، بسیار متفاوت است. در حقیقت، قسمت اعظم اطلاعات تراکنش که در برنامه‌های بیت‌کوین مشابه می‌بینیم، واقعاً در سیستم بیت‌کوین وجود ندارند.

Transaction

View information about a bitcoin transaction

0627052b626912f2703066a912ea577f20e4da4caa5a5fbdb8a57286c345c2f2

1Cdd9KFAaaatwczBwBttQcwXYCpvK8h7FK (0.1 BTC - Output)

1GdK9UzpHBzqzX2A9JFP3Dl4weBwqgmoQA - (Unspent)	0.015 BTC
1Cdd9KFAaaatwczBwBttQcwXYCpvK8h7FK - (Unspent)	0.0845 BTC

97 Confirmations

0.0995 BTC

Summary

Size 258 (bytes)

Received Time 2013-12-27 23:03:05

Included In Blocks 277316 (2013-12-27 23:11:54 +9 minutes)

Inputs and Outputs

Total Input 0.1 BTC

Total Output 0.0995 BTC

Fees 0.0005 BTC

Estimated BTC Transacted 0.015 BTC

شکل ۱-۶ تراکنش آليس به مغازه‌ی باب.

اگر برای بازیابی، کدگشایی و مشاهده‌ی اطلاعات درونی تراکنش آليس از واسطه خط-فرمان هسته‌ی بیت‌کوین (فرمان‌های getrawtransaction و decoderawtransaction) استفاده کنیم، چیزی شبیه کد زیر خواهیم دید:

```

"version": 1,
"locktime": 0,
"vin": [
  {
    "txid": "7957a35fe64f80d234d76d83a2...8149a41d81de548f0a65a8a999f6f18",
    "vout": 0,
    "scriptSig":
      "3045022100884d142d86652a3f47ba4746ec719bbfb040a570b1deccbb6498c75c4ae24cb02204b
      9f039ff08df09cbe9f6addac960298cad530a863ea8f53982c09db8f6e3813[ALL]0484ecc0d46f19
      18b30928fa0e4ed99f16a0fb4fde0735e7ade8416ab9fe423cc5412336376789d172787ec3457eee4
      lc04f4938de5cc17b4a10fa336a8d752adf",
    "sequence": 4294967295
  },
  {
    "vout": [
      {
        "value": 0.01500000,
        "scriptPubKey": "OP_DUP OP_HASH160
          ab68025513c3dbd2f7b92a94e0581f5d50f654e7 OP_EQUALVERIFY OP_CHECKSIG",
      },
      {
        "value": 0.08450000,
        "scriptPubKey": "OP_DUP OP_HASH160
          7f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a8 OP_EQUALVERIFY OP_CHECKSIG",
      }
    ]
  }
]

```

اگر به دقت به این تراکنش نگاه کنید، متوجه نکات مهمی خواهید شد؛ ولی از همه مهم‌تر اطلاعاتی است که در آین تراکنش وجود ندارد! آدرس آلیس کجاست؟ آدرس باب کجاست؟ مبلغ ورودی ۱۰ بیت‌کوین که آلیس از جو ادریافت کرده، کجاست؟ باید بدانید که در بیت‌کوین چیزی به نام سکه (پول)، فرستنده، گیرنده، تراز، حساب، و آدرس وجود ندارد. همه‌ی اینها در یک سطح بالاتر و برای راحتی و درک بهتر کاربر ساخته می‌شوند. در این تراکنش تعداد زیادی فبلدها و رشته‌های هگزادرسی‌مال غیرقابل فهم نیز به چشم می‌خورد. نکران باشد؛ در این فصل تمام آنها را به تفصیل توضیح خواهیم داد.

ورودی و خروجی تراکنش

اساسی‌ترین بخش ساختاری در تراکنش‌های بیت‌کوین خروجی تراکنش (transaction output) است. خروجی تراکنش یک تکه‌ی تقسیم‌ناپذیر از پول بیت‌کوین است، که در بلاک‌چین ثبت شده و اعتبار آن در سرتاسر شبکه‌ی بیت‌کوین به رسمیت شناخته می‌شود. گره‌های کامل بیت‌کوین یک نسخه از تمامی خروجی‌های موجود و قابل خرج کردن، موسوم به خروجی تراکنش خرج‌نشده (unspent transaction output)، یا UTXO، نگه می‌دارند. به مجموع تمامی خروجی‌های تراکنش خرج‌نشده مجموعه‌ی UTXO گفته می‌شود، که در حال حاضر تعداد آنها به چندین میلیون بالغ شده است. مجموعه‌ی UTXO با ایجاد UTXO‌های جدید بزرگتر، و با مصرف آنها کوچکتر می‌شود. هر تراکنش به یک تغییر در مجموعه‌ی UTXO منجر خواهد شد.

وقتی می‌کوییم کیف‌پول کاربر بیت‌کوین دریافت کرده است، منظور این است که این کیف‌پول یک UTXO پیدا کرده که می‌تواند آن را با یکی از کلیدهایی که در اختیار دارد، خرج کند. بنابراین، «تراز» بیت‌کوین یک کاربر عبارت است از مجموع تمام UTXO‌هایی که کیف‌پول وی می‌تواند خرج کند؛ این UTXO‌ها می‌توانند بین صدها تراکنش و صدها بلاک پراکنده باشند. مفهوم «تراز» توسط نرم‌افزار کیف‌پول خلق می‌شود. برنامه‌ی کیف‌پول با جستجوی کل بلاک‌چین می‌کند. اکثر برنامه‌های کیف‌پول از یک پایگاه داده‌ی محلی یا یک سرویس پایگاه داده‌ی آنلاین برای ذخیره کردن یک

ارجاع سریع به مجموعه‌ی تمام UTXO‌هایی که می‌توانند با کلیدهای خود خرج کنند، استفاده می‌کنند.

مبلغ یک خروجی تراکنش می‌تواند تعداد (صحیح) دلخواهی ساتوشی (یک-صد میلیونیم بیت‌کوین) باشد، هر چند یک تراکنش می‌تواند هر مبلغی داشته باشد، ولی بعد از ایجاد تراکنش، این مبلغ دیگر قابل تقسیم نخواهد بود. این یکی از ویژگی‌های مهم تراکنش‌های بیت‌کوین است که باید بر آن تأکید کنیم: خروجی تراکنش بیت‌کوین بک مقدار صحیح (بر حسب ساتوشی) و تقسیم‌ناپذیر است. یک خروجی خرج‌نشده را فقط می‌توان در یک تراکنش و به صورت کامل خرج کرد.

اگر مقدار یک UTXO بزرگتر از مبلغ مورد نیاز برای یک تراکنش باشد، تمامی آن مقدار بایستی در این تراکنش خرج شده، و باقیمانده‌ی آن به صورت یک تراکنش تتمه (change transaction) به کاربر برگردانده شود. به عبارت دیگر، اگر یک UTXO به ارزش ۲۰ بیت‌کوین داشته باشد و بخواهد در یک معامله فقط ۱ بیت‌کوین پردازد، تراکنش شما باید تمامی این UTXO ۲۰-بیت‌کوینی را خرج کرده و دو خروجی تولید کند: یک خروجی برای پرداخت ۱ بیت‌کوین به گیرنده‌ی مورد نظر، و یک خروجی دیگر برای بازگرداندن ۱۹ بیت‌کوین باقیمانده به کیف‌پول خودتان. به حاظر همین ویژگی تقسیم‌ناپذیر بودن خروجی‌های تراکنش، اکثر تراکنش‌های بیت‌کوین دارای تتمه هستند.

برای مثال، فرض کنید به معازه‌ای رفته‌اید و یک بطری نوشابه به قیمت ۲۵۰۰ تومان خریده‌اید. برای پرداخت بهای این بطری نوشابه، فروشنده هر ترکیبی از اسکناس‌ها و سکه‌های رایج کشور را که مجموع آنها ۲۵۰۰ تومان باشد (دو اسکناس ۱۰۰۰ تومانی + یک اسکناس ۵۰۰ تومانی؛ یک اسکناس ۲۰۰۰ تومانی + دو سکه‌ی ۲۰۰ تومانی + یک سکه‌ی ۱۰۰ تومانی؛ پنج اسکناس ۵۰۰ تومانی؛ یک اسکناس ۱۰۰۰ تومانی + سه سکه‌ی ۵۰۰ تومانی؛ وغیره)، از شما می‌پذیرد. البته اگر پول خرد نداشته باشید، فروشنده یک اسکناس ۱۰۰۰۰ تومانی را هم خواهد پذیرفت. در این حالت فروشنده تمهی این مبلغ را (باترکیب‌های مختلف) به شما بر می‌گرداند که می‌توانید از آن برای خریدهای بعدی استفاده کنید.

تراکنش‌های بیت‌کوین نیز به همین صورت انجام می‌شوند، یعنی کاربر باید یک UTXO (با هر مبلغی) را به طور کامل خرج کند. همان‌طور که شما نمی‌توانید برای پرداخت بهای خریدهای خود اسکناس‌های درشت را پاره کنید، کاربران بیت‌کوین هم نمی‌توانند UTXO‌ها را تقسیم کنند. البته برنامه‌های کیف‌پول معمولاً تلاش می‌کنند با ترکیب UTXO‌های ریزی که در اختیار دارند، مقداری برابر یا کمی بیشتر از مبلغ تراکنش فراهم کنند.

UTXO‌های زندگی واقعی، برنامه‌های کیف‌پول بیت‌کوین نیز از راهبردهای مختلفی برای تأمین مبلغ خرید استفاده می‌کنند؛ ترکیب UTXO‌های با مبالغ ریز، یافتن یک UTXO که دقیقاً برابر با مبلغ مورد نیاز باشد، یا استفاده از یک UTXO با مبلغ بالاتر و پس گرفتن تمهی آن. ناگفته پیداست که کیف‌پول این کارهای پیچیده را به طور خودکار و دور از چشم کاربر انجام می‌دهد. شما فقط زمانی باید نگران این چیزها باشید که بخواهید خودتان برنامه‌ای برای ایجاد تراکنش از UTXO‌های موجود در یک کیف‌پول بنویسید.

در یک تراکنش بیت‌کوین، خروجی‌های موجود (UTXO‌ها) خرج شده و در مواردی یک خروجی جدید تولید می‌شود که می‌توان آن را در تراکنش‌های آینده خرج کرد. بدین ترتیب، مبالغ بیت‌کوین (در زنجیره‌ای از تراکنش‌های مصرف و تولید UTXO) بین کاربران رد و بدل می‌شود.

تتها استناد این زنجیره‌ی ورودی/خروجی نوع خاصی از تراکنش موسوم به تراکنش پایگاه سکه (coinbase transaction) است، که اولین تراکنش هر بلاک محسوب می‌شود. این تراکنش به وسیله‌ی معدنچی «برنده» در ابتدای یک بلاک قرار می‌گیرد و جایزه‌ی او برای تولید یک بیت‌کوین کاملاً جدید قابل خرج است. تراکنش پایگاه سکه نیازی به مصرف UTXO ندارد، و به جای آن از یک ورودی خاص موسوم به پایگاه سکه (coinbase) استفاده می‌کند. این همان فرآیند استخراج بیت‌کوین جدید (موسوم به معدن‌کاوی) است که در فصل ۱۰ به تفصیل تشریح خواهیم کرد.

کدام زودتر به وجود آمده است؟ مرغ یا تخم مرغ؟ ورودی یا خروجی؟ به بیان دقیق فنی، خروجی زودتر به وجود آمده است، چون خروجی‌ها حاصل تراکنش‌های «پایگاه سکه» هستند که بیت‌کوین جدید تولید می‌کنند. تراکنش‌های «پایگاه سکه» ورودی ندارند و خروجی‌های خود را از «هیچ» خلق می‌کنند.



خروجی تراکنش

هر تراکنش بیت‌کوین یک یا چند خروجی تولید می‌کند، که در دفتر کل بیت‌کوین (بلاک‌چین) ثبت و ضبط می‌شوند. تقریباً تمامی این خروجی‌ها، به استنای یک خروجی خاص (موسوم به «خروجی ثبت داده»)، که در فصل ۷ خواهد دید، تعدادی قطعه‌ی بیت‌کوین قابل خرج موسوم به UTXO تولید می‌کنند که در تمامی شبکه‌ی بیت‌کوین به رسمیت شناخته می‌شوند و کاربران (مالک این UTXO‌ها) می‌توانند آنها را در تراکنش‌های بعدی مصرف کنند. در هر گره کامل بیت‌کوین پایگاه داده‌ای به نام «مجموعه‌ی UTXO» نگهداری می‌شود که شامل تمامی UTXO‌های موجود در شبکه می‌باشد. هر تراکنش جدید فقط مجاز است خروجی‌های موجود در این مجموعه را خرج (مصرف) کند.

هر خروجی تراکنش از دو بخش تشکیل می‌شود:

- یک مبلغ بیت‌کوین، بر حسب ساتوشی (کوچکترین واحد بیت‌کوین)، و

- یک معما رمزنگاری که شرایط لازم برای خرج کردن آن خروجی را مشخص می‌کند.

با این معما رمزنگاری گاهی اوقات اسکریپت قفل‌کننده (locking script)، اسکریپت شاهد (witness script) یا `scriptPubKey` نیز گفته می‌شود. در ادامه‌ی همین فصل درباره‌ی «زبان اسکریپتنویسی تراکنش» (که برای نوشن اسکریپت قفل‌کننده به کار می‌رود) به تفصیل صحبت خواهیم کرد.

اکنون، اجازه دهید دوباره به تراکنش آلیس (قسمت قبل) نگاه کنیم و بینیم آیا می‌توانیم خروجی‌های آن را تشخیص دهیم. در کدگذاری JSON (نمادگذاری شیء جاوا اسکریپت: JavaScript Object Notation)، در یک آرایه (لیست) به نام `vout` ذخیره می‌شوند:

```
"vout": [
    {
        "value": 0.01500000,
        "scriptPubKey": "OP_DUP OP_HASH160
ab68025513c3dbd2f7b92a94e0581f5d50f654e7 OP_EQUALVERIFY OP_CHECKSIG",
    },
    {
        "value": 0.08450000,
        "scriptPubKey": "OP_DUP OP_HASH160
7f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a8 OP_EQUALVERIFY OP_CHECKSIG",
    }
]
```

همان طور که می‌بینید، این تراکنش دو خروجی دارد، و هر خروجی با یک مبلغ (پارامتر `value`) و یک معما رمزنگاری (پارامتر `scriptPubKey`) تعریف شده است. در کدگذاری JSON (که توسط مشتری هسته‌ی بیت‌کوین) عدد صحیح بر حسب ساتوشی ثبت خواهد شد. بخش دوم هر خروجی یک معما رمزنگاری است که شرایط خرج کردن این خروجی را مشخص می‌کند. مشتری هسته‌ی بیت‌کوین این معما را با نام `scriptPubKey` و با فرمتی قابل خواندن نمایش می‌دهد.

در ادامه‌ی همین فصل درباره‌ی قفل کردن و باز کردن قفل UTXO، و همچنین زبان اسکریپتنویسی تولید `scriptPubKey` بیشتر صحبت خواهیم کرد، ولی قبل از پرداختن به آن مباحثت لازم است ساختار کلی ورودی‌ها و خروجی‌های یک تراکنش را بشناسیم.

سerial‌سازی تراکنش: خروجی

برای مبادله‌ی تراکنش‌ها بین برنامه‌های کاربردی و ارسال آنها روی شبکه، باید تراکنش را سerial‌سازی (serialization) کرد. سerial‌سازی فرآیندی است که در آن فرمت داخلی یک ساختمان داده به فرمتی تبدیل می‌شود که بتوان آن را بایت منتقل کرد، که به آن استریم بایت نیز گفته می‌شود. اصلی‌ترین و رایج‌ترین کاربرد سerial‌سازی ذخیره کردن یک ساختمان داده در فایل یا مبادله‌ی آن روی شبکه است. فرمت سerial‌سازی خروجی تراکنش در جدول ۱-۶ نشان داده شده است.

برای مثال، فرض کنید به معازه‌ای رفته‌اید و یک بطری نوشابه به قیمت 2500 تومان خریده‌اید. برای پرداخت بهای این بطری نوشابه، فروشنده هر ترکیبی از اسکناس‌ها و سکه‌های رایج کشور را که مجموع آنها 2500 تومان باشد (دو اسکناس 1000 تومانی + یک اسکناس 500 تومانی؛ یک اسکناس 2000 تومانی + دو سکه‌ی 200 تومانی + یک سکه‌ی 100 تومانی؛ پنج اسکناس 500 تومانی؛ یک اسکناس 1000 تومانی + سه سکه‌ی 50 تومانی؛ وغیره)، از شما می‌پذیرد. اگر پول خرد نداشته باشید، فروشنده یک اسکناس 10000 تومانی را هم خواهد پذیرفت. در این حالت فروشنده تهمه‌ای این مبلغ را (باترکیب‌های مختلف) به شما بر می‌گرداند که می‌توانید از آن برای خریدهای بعدی استفاده کنید.

تراکنش‌های بیت‌کوین نیز به همین صورت انجام می‌شوند، یعنی کاربر باید یک UTXO (با هر مبلغی) را به طور کامل خرج کند. همان طور که شما نمی‌توانید برای پرداخت بهای خریدهای خود اسکناس‌های درشت را پاره کنید، کاربران بیت‌کوین هم نمی‌توانند UTXO ها را تقسیم کنند. البته برنامه‌های کیف‌پول معمولاً تلاش می‌کنند با ترکیب UTXO های ریزی که در اختیار دارند، مقداری برابر یا کمی بیشتر از مبلغ تراکنش فراهم کنند.

مانند زندگی واقعی، برنامه‌های کیف‌پول بیت‌کوین نیز از راهبردهای مختلفی برای تأمین مبلغ خرید استفاده می‌کنند: ترکیب UTXO های با مبالغ ریز، یافتن یک UTXO که دقیقاً برابر با مبلغ مورد نیاز باشد، یا استفاده از یک UTXO با مبلغ بالاتر و پس گرفتن تتمه‌ی آن. ناگفته‌ی پیداست که کیف‌پول این کارهای پیچیده را به طور خودکار و دور از چشم کاربر انجام می‌دهد. شما فقط زمانی باید نگران این چیزها باشید که بخواهید خودتان برنامه‌ای برای ایجاد تراکنش از UTXO های موجود در یک کیف‌پول بنویسید.

در یک تراکنش بیت‌کوین، خروجی‌های موجود (UTXO ها) خرج شده و در مواردی یک خروجی جدید تولد می‌شود که می‌توان آن را در تراکنش‌های آینده خرج کرد. بدین ترتیب، مبالغ بیت‌کوین (در زنجیره‌ای از تراکنش‌های مصرف و تولید UTXO) بین کاربران رد و بدل می‌شود.

تها استثناد این زنجیره‌ی ورودی/خروجی نوع خاصی از تراکنش موسوم به تراکنش پایگاه سکه (coinbase transaction) است، که اولین تراکنش هر بلاک محسوب می‌شود. این تراکنش به وسیله‌ی معدنچی «برنده» در ابتدای یک بلاک قرار می‌گیرد و جایزه‌ی او برای تولید یک بیت‌کوین کاملاً جدید قابل خرج است. تراکنش پایگاه سکه نیازی به مصرف UTXO ندارد، و به جای آن از یک ورودی خاص موسوم به پایگاه سکه (coinbase) استفاده می‌کند. این همان فرآیند استخراج بیت‌کوین جدید (موسوم به معدن‌کاوی) است که در فصل ۱۰ به تفصیل تشریح خواهیم کرد.

کدام زودتر به وجود آمده است؟ مرغ یا تخم مرغ؟ ورودی یا خروجی؟ به بیان دقیق فنی، خروجی زودتر به وجود آمده است، چون خروجی‌ها حاصل تراکنش‌های «پایگاه سکه» هستند که بیت‌کوین جدید تولید می‌کنند. تراکنش‌های «پایگاه سکه» ورودی ندارند و خروجی‌های خود را از «هیچ» خلق می‌کنند.



خروجی تراکنش

هر تراکنش بیت‌کوین یک یا چند خروجی تولید می‌کند، که در دفتر کل بیت‌کوین (بلاک چین) ثبت و ضبط می‌شوند. تقریباً تمامی این خروجی‌ها، به استثنای یک خروجی خاص (موسوم به «خروچی ثبت داده»، که در فصل ۷ خواهد دید)، تعدادی قطعه‌ی بیت‌کوین قابل خرج موسوم به UTXO تولید می‌کنند که در تمامی شبکه‌ی بیت‌کوین به رسمیت شناخته می‌شوند و کاربران (مالک این UTXO ها) می‌توانند آنها را در تراکنش‌های بعدی مصرف کنند. در هر گره کامل بیت‌کوین پایگاه داده‌ای به نام «مجموعه‌ی UTXO» نگهداری می‌شود که شامل تمامی UTXO های موجود در شبکه می‌باشد. هر تراکنش جدید فقط مجاز است خروجی‌های موجود در این مجموعه را خرج (مصرف) کند.

هر خروجی تراکنش از دو بخش تشکیل می‌شود:

- یک مبلغ بیت‌کوین، بر حسب ساتوشی (کوچکترین واحد بیت‌کوین)، و

- یک معما رمزنگاری که شرایط لازم برای خروج کردن آن خروجی را مشخص می‌کند.

به این معما رمزنگاری گاهی اوقات اسکریپت قفل‌کننده (locking script) یا witness script نیز گفته می‌شود. در ادامه همین فصل درباره «زبان اسکریپتنویسی تراکنش» (که برای نوشتan scriptPubKey اسکریپت قفل‌کننده به کار می‌رود) به تفصیل صحبت خواهیم کرد.

اکنون، اجازه دهید دوباره به تراکنش آلبس (قسمت قبل) نگاه کنیم و بینیم آیا می‌توانیم خروجی‌های آن را تشخیص دهیم. در گذاری JSON (نمادگذاری شیء جاوا اسکریپت: JavaScript Object Notation)، خروجی‌های تراکنش در یک آرایه (لیست) به نام vout ذخیره می‌شوند:

```
"vout": [
    {
        "value": 0.01500000,
        "scriptPubKey": "OP_DUP OP_HASH160
ab68025513c3dbd2f7b92a94e0581f5d50f654e7 OP_EQUALVERIFY OP_CHECKSIG",
    },
    {
        "value": 0.08450000,
        "scriptPubKey": "OP_DUP OP_HASH160
7f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a8 OP_EQUALVERIFY OP_CHECKSIG",
    }
]
```

همان طور که می‌بینید، این تراکنش دو خروجی دارد، و هر خروجی با یک مبلغ (پارامتر value) و یک معما رمزنگاری (پارامتر scriptPubKey) تعریف شده است. در گذاری JSON (که توسط مشتری هسته‌ی بیت‌کوین) مبلغ خروجی بر حسب بیت‌کوین نمایش داده می‌شود، ولی در خود تراکنش این مبلغ به صورت یک به کار می‌رود) مبلغ خروجی بر حسب بیت‌کوین ثبت خواهد شد. بخش دوم هر خروجی یک معما رمزنگاری است که شرایط خرج عدد صحیح بر حسب ساتوشی ثبت خواهد شد. مشتری هسته‌ی بیت‌کوین این معما را با نام scriptPubKey و با فرمتی کردن این خروجی را مشخص می‌کند. مشتری هسته‌ی بیت‌کوین این معما را با نام vout قابل خواندن نمایش می‌دهد.

در ادامه همین فصل درباره قفل کردن و باز کردن قفل UTXO، و همچنین زبان اسکریپتنویسی تولید scriptPubKey یثتر صحبت خواهیم کرد، ولی قبل از پرداختن به آن مباحثت لازم است ساختار کلی ورودی‌ها و خروجی‌های یک تراکنش را بشناسیم.

سربال‌سازی تراکنش: خروجی

برای مبادله‌ی تراکنش‌ها بین برنامه‌های کاربردی و ارسال آنها روی شبکه، باید تراکنش را سربال‌سازی (serialization) کرد. سربال‌سازی فرآیندی است که در آن فرمت داخلی یک ساختمان داده به فرمتی تبدیل می‌شود که بتوان آن را بایت به بایت منتقل کرد، که به آن استریم بایت نیز گفته می‌شود. اصلی‌ترین و رایج‌ترین کاربرد سربال‌سازی ذخیره کردن یک ساختمان داده در فایل یا مبادله‌ی آن روی شبکه است. فرمت سربال‌سازی خروجی تراکنش در جدول ۱-۶ نشان داده شده است.

جدول ۱-۶ سریال‌سازی خروجی تراکنش

فیلد	اندازه	توضیح
مبلغ	۸ بایت (راست-نویس)	مقدار بیت کوین بر حسب ساتوشی (10^{-8} بیت کوین)
اندازه اسکریپت قفل کننده	۹ بایت (متغیر)	طول اسکریپت قفل کننده بر حسب بایت
اسکریپت قفل کننده	متغیر	اسکریپتی که شرایط خرج کردن این خروجی را مشخص می‌کند

اکثر کتابخانه‌ها و چارچوب‌های بیت کوین در داخل خود تراکنش‌های را به صورت استریم بایت ذخیره نمی‌کنند، چون در این صورت پردازش آنها که مستلزم دسترسی فوری به فیلدهای مختلف است، بسیار پیچیده و وقت‌گیر خواهد بود آنها برای این منظور از ساختمان‌های داده‌ی خاص (که معمولاً بر اساس فناوری شیء‌گرا طراحی شده‌اند) استفاده می‌کنند. فرآیند تبدیل استریم بایت تراکنش به ساختمان داده‌ی داخلی مورد نیاز کتابخانه‌ی بیت کوین ناسریال‌سازی (deserialization) یا نجزیه‌ی تراکنش (transaction parsing) نامیده می‌شود، که دقیقاً عکس سریال‌سازی است. اکثر کتابخانه‌های بیت کوین دارای توابع داخلی برای سریال‌سازی و ناسریال‌سازی تراکنش‌ها هستند. تشخیص بخش‌های مختلف یک تراکنش از روی استریم بایت آن کمی دشوار است ولی با کمی دقت می‌توان عنصر آن را تشخیص داد. در مثال ۱-۶ رشته‌ی سریال‌سازی شده‌ی تراکنش آليس را مشاهده می‌کنید:

مثال ۱-۶ تراکنش سریال‌سازی شده‌ی آليس با فرمت هگزادسیمال

```
0100000001186f9f998a5aa6f048e51dd8419a14d8a0f1a8a2836dd734d2804fe65fa357790000000
08b483045022100884d142d86652a3f47ba4746ec719bbfb040a570b1deccbb6498c75c4ae24cb02
204b9f039ff08df09cbe9f6addac960298cad530a863ea8f53982c09db8f6e381301410484ecc0d46
f1918b30928fa0e4ed99f16a0fb4fde0735e7ade8416ab9fe423cc5412336376789d172787ec3457e
ee41c04f4938de5cc17b4a10fa336a8d752adffffffffff0260e31600000000001976a914ab6802551
3c3dbd2f7b92a94e0581f5d50f654e788acd0ef8000000000001976a9147f9b1a7fb68d60c536c2fd
Baeea53a8f3cc025a888ac00000000
```

نکات زیر می‌توانند در گذگشایی این رشته‌ی هگزادسیمال و استخراج اطلاعات خروجی (های) تراکنش به شما کمک کنند:

- در بخش برجسته شده‌ی مثال ۱-۶ دو خروجی وجود دارد که بر اساس قواعد جدول ۱-۶ سریال‌سازی شده‌اند.
- مبلغ این تراکنش (فیلد `value`) $15\text{,}000\text{,}000$ ساتوشی است که در دستگاه عددنوبی هگزادسیمال معادل $60\text{,}000\text{,}000$ خواهد بود.
- در تراکنش سریال‌سازی شده، مقدار $60\text{,}000\text{,}000$ با فرمت راست-نویس (little-endian) است: فرمتی که در آن بایت‌های یک رشته از راست به چپ در خروجی قرار داده می‌شوند و ترتیب آنها معکوس می‌شود) گذگذاری می‌شود، بنابراین به صورت $16\text{,}000\text{,}000\text{,}000$ در می‌آید.
- طول `scriptPubKey` در این مثال ۲۵ بایت است که در دستگاه عددنوبی هگزادسیمال معادل ۱۹ خواهد بود.

ورودی تراکنش

ورودی (های) یک تراکنش مشخص می‌کند کدام UTXO باید مصرف شود و از طریق باز کردن قفل اسکریپت، مالکیت خود بر آنها را اثبات می‌کند. برای ایجاد یک تراکنش، برنامه‌ی کیف‌پول از میان UTXO هایی که در اختیار دارد، یک یا چند UTXO که مجموع آنها برای پرداخت مورد نظر کافی باشد، انتخاب می‌کند. به ازای هر UTXO که برای این پرداخت مصرف می‌شود، کیف‌پول یک ورودی ایجاده کرده و با یک اسکریپت بازکننده قفل آن را باز می‌کند.

اجازه دهد نگاهی عمیق‌تر به اجزای یک ورودی بیندازیم. اولین بخش از یک ورودی اشاره‌گری است به یک UTXO علاوه بر ارجاع درهم تراکنش، یک شماره ترتیب هم دارد که نشان می‌دهد این UTXO در کجای بلاک چین ثبت شده است. دومین بخش از ورودی تراکنش یک اسکریپت بازکننده‌ی قفل است که کیف‌پول برای برآورده کردن شرط خرج کردن این UTXO ارائه می‌کند. در اکثر مواقع، این اسکریپت بازکننده‌ی قفل چیزی نیست جزیک امضای دیجیتال و کلید عمومی که مالکیت ییت کوین را ثبات می‌کند. با این حال، همه‌ی اسکریپت‌های بازکننده‌ی قفل حاوی امضای دیجیتال نیستند. سومین بخش از ورودی تراکنش یک شماره ترتیب (sequence number) است که در ادامه بیشتر درباره آن صحبت خواهیم کرد. یک پاره‌یگر به تراکنش آلیس (قسمت قبل) نگاه کنید. ورودی‌های تراکنش در یک آرایه (لیست) به نام vin ذخیره می‌شوند:

```
"vin": [
  {
    "txid": "7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18",
    "vout": 0,
    "scriptSig":
      "3045022100884d142d86652a3f47ba4746ec719bbfb040a570b1deccbb6498c75c4ae24cb02204b
      9f039ff08df09cbe9f6addac960298cad530a863ea8f53982c09db8f6e3813[ALL]0484ecc0d46f19
      18b30928fa0e4ed99f16a0fb4fde0735e7ade8416ab9fe423cc5412336376789d172787ec3457eee4
      1c04f4938de5cc17b4a10fa336a8d752adf",
    "sequence": 4294967295
  }
],
```

همان طور که می‌بینید، این تراکنش فقط یک ورودی دارد (چون همین یک UTXO دارای مبلغ کافی برای پرداخت مورد نظر هست). این ورودی چهار فیلد دارد:

- یک شناسه‌ی تراکنش، ارجاع به تراکنشی که حاوی UTXO مورد نیاز برای خرج کردن است.
- یک آندیس خروجی (vout)، که مشخص می‌کند کدام UTXO موجود در آن تراکنش باید خرج شود (آن‌دیس صفر یعنی اولین UTXO آن تراکنش).
- یک scriptSig (امضای اسکریپت)، که شرط موجود در این UTXO را برآورده کرده و قفل خرج کردن آن را باز می‌کند.
- یک شماره ترتیب (که در ادامه توضیح خواهیم داد).

در تراکنش آلیس، ورودی به شناسه‌ی تراکنش زیر اشاره کرده است:

7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18

واندیس خروجی آن ۰ (یعنی، اولین UTXO ایجاد شده توسط آن تراکنش) است. برای تولید اسکریپت بازکننده‌ی قفل، کیف‌پول آلیس ابتدا UTXO اشاره‌شده را بازیابی کرده، اسکریپت قفل‌کننده‌ی آن را بررسی می‌کند، و سپس با استفاده از آن اسکریپت بازکننده‌ی قفل مورد نیاز را تولید خواهد کرد.

فقط با یک نگاه به این ورودی می‌توان متوجه شد که هیچ چیزی درباره این UTXO نمی‌دانیم، به غیر از یک ارجاع به تراکنشی که این UTXO در آن قرار دارد. به عبارت دیگر، نه مقدار این UTXO را (بر حسب ساتوشی) می‌دانیم، نه اسکریپت قفل‌کننده‌ای را می‌شناسیم که شرایط خرج کردن آن را بیان می‌کند. برای داشتن این اطلاعات، کیف‌پول باید ابتدا تراکنشی را که UTXO ارجاع شده در آن قرار دارد، بازیابی کند. توجه داشته باشید، از آنجا که مبلغ این ورودی به طور صریح قید نشده است، همچنین لازم است از این UTXO ارجاع شده برای محاسبه کارمزد تراکنش نیز استفاده کنیم (در ادامه درباره «کارمزد تراکنش» بیشتر صحبت خواهیم کرد).

البته این فقط آليس نیست که مجبور است UTXO ارجاع شده در ورودی تراکنش خود را بازیابی کند؛ همین‌که تراکنش آليس روی شبکه بیت‌کوین منتشر شود، تمام گره‌های اعتبارسنجی نیز باید UTXO ارجاع شده در ورودی این تراکنش را بازیابی کنند تا بتوانند صحت و اعتبار آن را بسنجند.

تراکنش‌ها به خودی خود ناقص هستند، چون محتوای مورد نیاز را در خود ندارند. آنها در ورودی‌های خود به یک یا چند UTXO ارجاع (اشارة) می‌کنند، ولی بدون بازیابی آن UTXO‌ها چیزی از مبلغ یا شروط قفل کننده‌گی آنها نخواهیم داشت. هنگام کُدگشایی یک تراکنش به هر منظوری که باشد (اعتبارسنجی، محاسبه‌ی مقدار کارمزد، یا پردازش اسکریپت بازکننده‌ی قفل آن)، مجبور هستیم قبل از هر کاری UTXO‌های ارجاع شده در ورودی (های) آن تراکنش را از بلاک چین بازیابی کنیم تا بتوانیم محتویات مورد نیاز برای کار خود را استخراج کنیم. برای مثال، محاسبه‌ی کارمزد یک تراکنش مستلزم دانستن مجموع مبلغ ورودی‌ها و خروجی‌های آن است؛ ولی دانستن مبلغ آنها بدون بازیابی UTXO‌های ارجاع شده در ورودی تراکنش از بلاک چین ممکن نخواهد بود. بنابراین، حتی یک عمل ساده مثل محاسبه‌ی کارمزد یک تراکنش واحد مستلزم مراحل متعدد و استخراج اطلاعات از چندین تراکنش دیگر است.

برای بازیابی آن UTXO‌ها می‌توان از همان فرمان‌های هسته‌ی بیت‌کوین که برای بازیابی، کُدگشایی و مشاهده اطلاعات تراکنش آليس به کار بردیم (`decoderawtransaction` و `getrawtransaction`)، استفاده کنیم. با این کار، اطلاعاتی شبیه زیر دریافت خواهیم کرد:

```
"vout": [
  {
    "value": 0.10000000,
    "scriptPubKey": "OP_DUP OP_HASH160
7f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a8 OP_EQUALVERIFY OP_CHECKSIG",
  },
]
```

همان‌طور که مشاهده می‌کنید، این UTXO دارای مبلغ ۰.۱ ره بیت‌کوین است، و یک اسکریپت قفل کننده (scriptPubKey) هم دارد. [که با `OP_DUP` `OP_HASH160` شروع می‌شود]

برای درک کامل تراکنش آليس بایستی تراکنش‌های (های) قبلی ارجاع شده در ورودی‌های آن را بازیابی کنیم. تابع مورد نیاز برای بازیابی تراکنش‌های قبلی و خروجی‌های خرج نشده چنان پُرکاربرد است که تقریباً در هر کتابخانه و API بیت‌کوین وجود دارد.



سربال‌سازی تراکنش: ورودی

وقتی یک تراکنش (برای ذخیره کردن ساختمان داده در فایل یا انتقال روی شبکه) سربال‌سازی می‌شود، ورودی (های) آن بر اساس جدول ۲-۶ به استریم بایت تبدیل خواهد شد.

جدول ۲-۶ سربال‌سازی ورودی تراکنش

فیلد	اندازه	توضیح
ذره‌م تراکنش	۳۲ بایت	اشاره‌گر به تراکنش حاوی UTXO که باید خرج شود
اندیس خروجی	۴ بایت	اندیس UTXO که باید خرج شود؛ شماره‌گذاری از ۰ شروع می‌شود
اسکریپت بازکننده‌ی قفل	۱ تا ۹ بایت (متغیر)	اندازه‌ی اسکریپت بازکننده‌ی قفل (بر حسب بایت)
شماره ترتیب	۴ بایت	اسکریپت که شرایط اسکریپت قفل کننده‌ی UTXO را برآورده می‌کند برای زمان قفل به کار می‌رود؛ یا می‌تواند غیرفعال باشد (FFFFFFFFFF)

مانند خروجی‌ها، تشخیص ورودی‌های یک تراکنش نیز از روی استریم بایت آن کمی دشوار است؛ اجازه دهدید یک بار دیگر بخش ورودی تراکنش آلیس را ببینیم:

```
"vin": [
  {
    "txid": "7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18",
    "vout": 0,
    "scriptSig":
      "3045022100884d142d86652a3f47ba4746ec719bbfb040a570b1deccbb6498c75c4ae24cb02204b
      9f039ff08df09fbe9f6addac960298cad530a863ea8f53982c09db8f6e3813[ALL]0484ecc0d46f19
      18b30928fa0e4ed99f16a0fb4fde0735e7ade8416ab9fe423cc5412336376789d172787ec3457eee4
      1c04f4938de5cc17b4a10fa336a8d752adf",
    "sequence": 4294967295
  },
],
]
```

اگرچه ببینیم آیا می‌توانیم فیلد‌های این ورودی را در رشته‌ی سریال‌سازی‌شده‌ی تراکنش آلیس (با فرمت هگزادسیمال)، مثال ۲-۶، تشخیص دهیم:

مثال ۲-۶ تراکنش سریال‌سازی‌شده‌ی آلیس با فرمت هگزادسیمال

```
0100000001186f9f998a5aa6f048e51dd8419a14d8a0f1a8a2836dd734d2804fe65fa357790000000
08b483045022100884d142d86652a3f47ba4746ec719bbfb040a570b1deccbb6498c75c4ae24cb02
204b9f039ff08df09fbe9f6addac960298cad530a863ea8f53982c09db8f6e381301410484ecc0d46
f1918b30928fa0e4ed99f16a0fb4fde0735e7ade8416ab9fe423cc5412336376789d172787ec3457e
ee41c04f4938de5cc17b4a10fa336a8d752adffffffffff0260e316000000000001976a914ab6802551
3c3dbd2f7b92a94e0581f5d50f654e788acd0ef8000000000001976a9147f9b1a7fb68d60c536c2fd
8aeaa53a8f3cc025a888ac00000000
```

در بخش بر جسته شده‌ی مثال ۲-۶ یک ورودی وجود دارد که بر اساس قواعد جدول ۲-۶ سریال‌سازی‌شده‌اند. نکات زیر می‌توانند در گذگشایی این رشته و استخراج اطلاعات ورودی‌های تراکنش به ما کمک کنند:

- شناسه‌ی این تراکنش (فیلد txid) با ۷۹ شروع و به ۱۸ ختم می‌شود، اما از آنجاکه فرآیند سریال‌سازی ترتیب بایت‌ها را معکوس می‌کند (فرمت راست-نویس)، این رشته در خروجی سریال با ۱۸ شروع و به ۷۹ ختم خواهد شد.
- اندیس خروجی (فیلد vout) یک دسته‌ی ۴-بایتی صفر (۰۰۰۰۰۰۰۰) است که تشخیص آنها چندان دشوار نیست.
- طول فیلد scriptSig در این مثال ۱۳۹ بایت است که در دستگاه عدد نویسی هگزادسیمال معادل ۸۵ خواهد بود.
- در این تراکنش به شماره ترتیب (فیلد sequence) نیز مقدار FFFFFFFF (زمان قفل غیرفعال) داده شده که تشخیص آن ساده است.

کارمزد تراکنش

اکثر تراکنش‌ها شامل کارمزد تراکنش (transaction fee) هستند، که به نوعی جبران هزینه‌ای است که معدن‌چیان بیت‌کوین برای آمن کردن این شبکه متحمل می‌شوند. کارمزد تراکنش به عنوان یک ساز و کار امنیتی نیز عمل می‌کند، چون سرازیر کردن انبوهی از تراکنش‌ها توسط نفوذگران و غرق کردن شبکه‌ی بیت‌کوین را از نظر اقتصادی ناممکن می‌سازد. همان‌طور که در فصل ۱۰ خواهید دید، استخراج بیت‌کوین و کارمزد تراکنش جوایزی هستند که به استخراج کنندگان بیت‌کوین تعلق می‌گیرند.

در این قسمت نشان می‌دهیم کارمزد تراکنش چگونه به یک تراکنش اضافه می‌شود. اکثر برنامه‌های کیف‌پول کارمزد تراکنش را به طور خودکار محاسبه و به آن اضافه می‌کنند. با این حال، اگر خودتان قصد دارید برنامه‌ای برای پردازش تراکنش‌های بیت‌کوین بنویسید یا از رابط خط-فرمان هسته‌ی بیت‌کوین برای این منظور استفاده کنید، باید کارمزد تراکنش را به صورت دستی محاسبه کرده و به آن اضافه کنید.

کارمزد تراکنش مشوفی است برای اضافه کردن (استخراج) یک تراکنش به بلاک بعدی، و همچنین با تعیین یک هزینه‌ی کوچک به هر تراکنش، مانع سوءاستفاده از سیستم بیت‌کوین می‌شود. کارمزد یک تراکنش نصیب کسی (معدنچی) می‌شود که بلاک مربوطه را استخراج کرده و این تراکنش را در بلاک چین ثبت کند.

کارمزد یک تراکنش هیچ ارتباطی با مبلغ آن ندارد، بلکه بر مبنای اندازه‌ی آن تراکنش بر حسب کیلویایت محاسب می‌شود. در حالت کلی، کارمزد تراکنش بر اساس نیروی بازار (عرضه و تقاضا) در داخل شبکه‌ی بیت‌کوین تعیین می‌شود معدنچیان بیت‌کوین تراکنش‌های ابرمبنای معیارهای مختلفی (از جمله کارمزد تراکنش) اولویت‌بندی می‌کنند، و شاید حتی در شرایط خاص تراکنش‌های ابرمجانی پردازش کنند. کارمزد تراکنش روی اولویت پردازش آن تأثیر می‌گذارد، به این معنا که احتمال اضافه شدن تراکنش‌های دارای کارمزد بالاتر در بلاک بعدی بیشتر است، در حالی که پردازش یک تراکنش با کارمزد ناقیز (یا بدون کارمزد) ممکن است به تأخیر بیفتند، یا اصلاً پردازش نشود. کارمزد تراکنش اجباری نیست، و تراکنش‌های بدون کارمزد هم ممکن است سرانجام پردازش شوند؛ با این حال، اضافه کردن کارمزد به یک تراکنش اولویت پردازش آن را بالا خواهد برد چگونگی محاسبه‌ی کارمزد تراکنش و تأثیر آن بر اولویت‌بندی پردازش تراکنش‌هادر طول زمان دستخوش تغییرات زیادی شده است. در ابتدا، کارمزد تراکنش در سرتاسر شبکه ثابت و مشخص بود. ولی به تدریج، ساختار کارمزد تراکنش انعطاف‌پذیرتر شده و تأثیر نیروهای بازار بر آن بیشتر شد، به طوری که در محاسبه‌ی کارمزد تراکنش به ظرفیت شبکه و مبلغ آن توجه یافتند مبدول شد. دست کم از آغاز سال ۲۰۱۶ به این سو، محدود شدن ظرفیت بیت‌کوین باعث ایجاد رقابت شدید بین تراکنش‌ها شد، و بالارفتن کارمزد تراکنش عملاً تراکنش‌های بدون کارمزد را به تاریخ سپرد. امروزه دیگر تراکنش‌های بدون کارمزد یا با کارمزد بسیار ناقیز به ندرت استخراج (پردازش) می‌شوند و گاهی اوقات حتی در شبکه‌ی بیت‌کوین منتشر نمی‌شوند.

در هسته‌ی بیت‌کوین، خط‌مشی محاسبه‌ی کارمزد تراکنش به وسیله‌ی گزینه‌ی minrelaytxfee تعیین می‌شود مقدار پیش‌فرض minrelaytxfee در حال حاضر 1 mili بیت‌کوین یا یک‌صدم یک میلی بیت‌کوین به ازای هر کیلویایت است. بنابراین، به طور پیش‌فرض، تراکنش‌هایی که کارمزد آنها کمتر از 1 mili بیت‌کوین باشد، به عنوان تراکنش بدون کارمزد (مجانی) تلقی شده و فقط در صورتی در شبکه‌ی بیت‌کوین منتشر می‌شوند که قضای خالی در مخزن حافظه (mempool) وجود داشته باشد؛ در غیر این صورت، به سادگی دور اندامه خواهد شد. گره‌های بیت‌کوین می‌توانند با تغییر دادن مقدار minrelaytxfee از مبالغه دیگری به عنوان کارمزد پیش‌فرض خود استفاده کنند.

هر سرویس بیت‌کوین که با تولید تراکنش سروکار داشته باشد (از جمله برنامه‌های کیف‌پول، صرافی، خرده‌فروشی و غیره)، باید کارمزد دینامیک (dynamic fee) را پیاده‌سازی کند. برای پیاده‌سازی کارمزد دینامیک می‌توانید از سرویس‌های تخمین کارمزد شخص-ثالث یا یک الگوریتم داخلی تخمین کارمزد استفاده کنید. اگر مطمئن نیستید، ابتدا با یک سرویس تخمین کارمزد شخص-ثالث شروع کرده و بعد از این که تجربه‌ی کافی به دست آوردید، الگوریتم تخمین کارمزد شخصی خود را طراحی و پیاده‌سازی کنید.

الگوریتم‌های تخمین کارمزد مقدار کارمزد مناسب را بر اساس ظرفیت شبکه و کارمزد پیشنهادی تراکنش‌های «رقبا» محاسبه می‌کنند. این الگوریتم‌ها می‌توانند کاملاً ساده‌نگر باشند و بر اساس میانگین (یا میانه) کارمزد در آخرین بلاک عمل کنند، یا ظرفات و پیچیدگی بیشتری به خرج دهند و از تحلیل‌های آماری استفاده کنند. یک الگوریتم تخمین کارمزد باید تواند کارمزدی را تخمین بزند که احتمال انتخاب شدن یک تراکنش و اضافه شدن آن به تعدادی بلاک مشخص را بینه کند. اکثر سرویس‌های تخمین کارمزد شخص-ثالث به کاربر اجازه می‌دهند اولویت دلخواه خود (بالا، متوسط، یا پایین) را برای تخمین کارمزد یک تراکنش انتخاب کند. اولویت بالا یعنی پرداخت کارمزد بیشتر، ولی در عین حال به معنای احتمال بالای

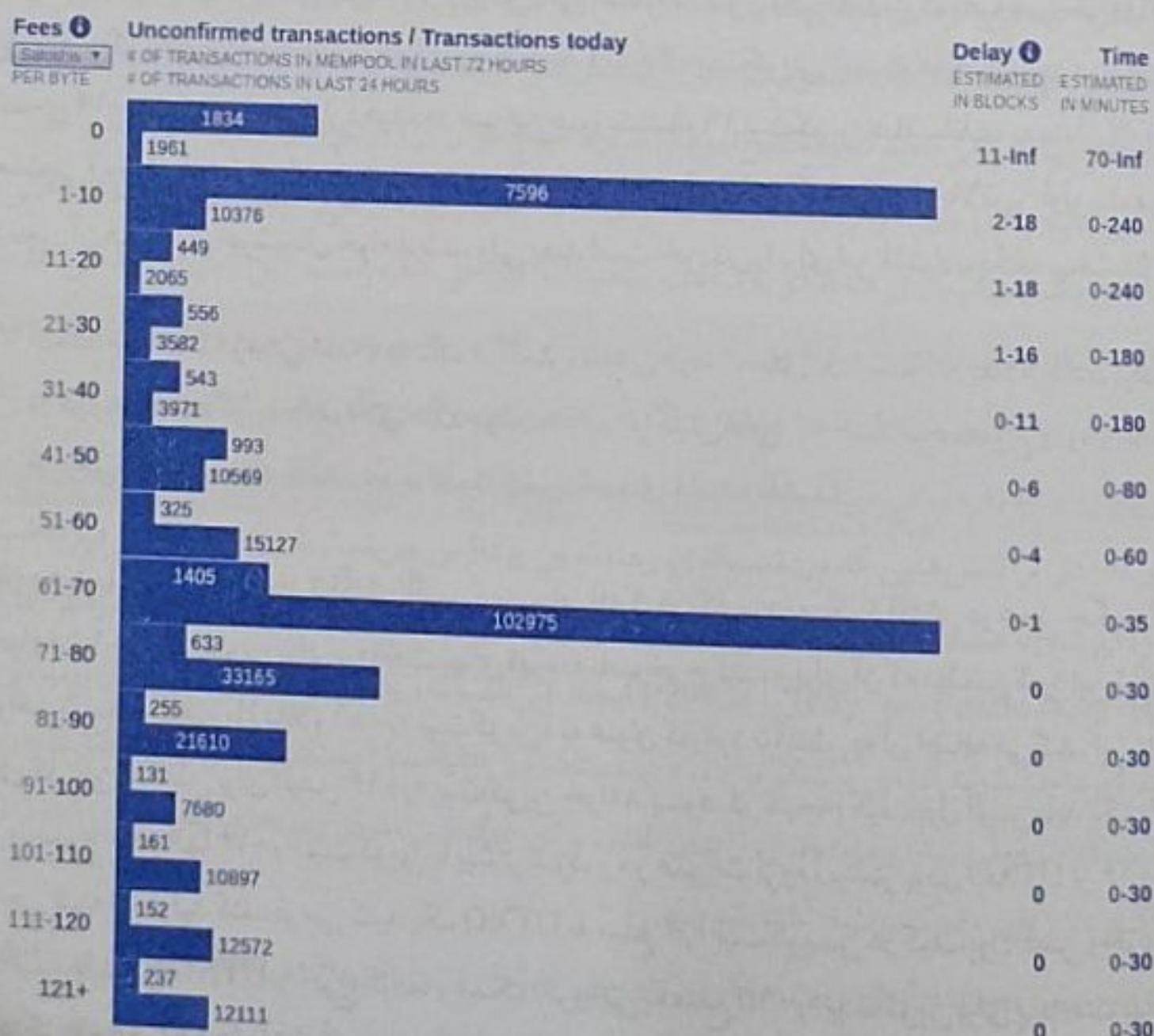
اصافه شدن آن تراکنش به بلاک بعدی نیز هست. پایین آوردن اولویت باعث کاهش کارمزد تراکنش می‌شود، ولی در ضمن تأیید آن را به تأخیر خواهد انداخت.

اغلب برنامه‌های کیف‌پول از سرویس‌های شخص-ثالث برای محاسبه کارمزد تراکنش استفاده می‌کند. بکی از این سرویس‌های پُر طرفدار <http://bitcoinfees.21.co> است، که علاوه بر فراهم آوردن API محاسبه کارمزد دینامیک، یک نمودار ساتوشی/بایت به ازای اولویت‌های مختلف را نیز به شما نمایش خواهد داد.

کارمزد ثابت دیگر امکان موفقیت در شبکه‌ی بیت‌کوین ندارد. کاربران کیف‌پول‌هایی که از کارمزد ثابت استفاده می‌کنند، اغلب از عملکرد آنها ناراضی هستند چون تراکنش‌های معمولاً «گیرمی‌کنند» و تأیید نمی‌شوند. کاربرانی که با ساز و کار تراکنش‌های بیت‌کوین و نقش کارمزد تراکنش در پردازش آنها آشنا نیستند، در مواجهه با تراکنش‌های «گیر کرده» نگران و دلسرد می‌شوند چون فکر می‌کنند پول خود را از دست داده‌اند.



در نمودار شکل ۲-۶ تخمین کارمزد تراکنش در زمان-واقعی در پله‌های ۱۰ ساتوشی/بایت و زمان تأیید مورد انتظار (بر حسب دقیقه و تعداد بلاک) برای تراکنش‌هایی با کارمزدهای مختلف را مشاهده می‌کنید. برای هر بازه‌ی کارمزد (مثلاً ۰-۶۱ ساتوشی/بایت)، دو ستون افقی می‌بینید که یکی تعداد تراکنش‌های تأیید نشده (۱۴۰۵)، و دیگری تعداد کل تراکنش‌های انجام شده با آن کارمزد در ۲۴ ساعت گذشته (۱۰۲,۹۷۵) را نشان می‌دهد. بر اساس این نمودار، کمترین کارمزد توصیه شده برای کسانی که می‌خواهند تراکنش‌های آنها بالا فاصله (وبدون تأخیر) در بلاک بعدی استخراج شود، ۸۰ ساتوشی/بایت است. برای مقایسه، میانگین اندازه‌ی تراکنش‌ها ۲۲۶ بایت است، بنابراین حداقل کارمزد مناسب برای آنها $18,080 \times 226 = 18,080,000$ BTC (۰,۰۸٪ خواهد بود).



شکل ۲-۶ سرویس تخمین کارمزد bitcoinfees.21.co

داده‌های تخمین کارمزد تراکنش را می‌توانید به سادگی و از طریق یک HTTP REST API ساده از صفحه‌ی <https://bitcoinfees.21.co/api/v1/fees/recommended> بازیابی کنید. برای مثال، اگر از فرمان `curl` استفاده کنید، با پاسخ ذیل روبرو خواهد شد:

```
$ curl https://bitcoinfees.21.co/api/v1/fees/recommended
```

```
{"fastestFee": 80, "halfHourFee": 80, "hourFee": 60}
```

این API یک شیء JSON بر می‌گرداند که حاوی آخرین تخمین برای سریع‌ترین تأییدیه (`fastestFee`) [پردازش در بالای بعدی]، تأییدیه در سه‌بلالک (`halfHourFee`)، و تأییدیه در شش‌بلالک (`hourFee`) بر حسب ساتوشی/بایت است.

اضافه کردن کارمزد به تراکنش

در ساختمان داده‌ی تراکنش هیچ فیلدی برای کارمزد وجود ندارد، چون کارمزد یک مقدار ضمنی است که از تفاصل مجموع ورودی‌ها و مجموع خروجی‌های تراکنش به دست می‌آید. بعد از کم کردن تمام خروجی‌ها از تمام ورودی‌ها، هر مقداری که باقی بماند، به عنوان کارمزد به معنی‌چی (گرهی که موفق شده این تراکنش را به بالا چین اضافه کند) تعلق خواهد گرفت:

$$\text{Fees} = \text{Sum(Inputs)} - \text{Sum(Outputs)}$$

این یکی از موارد گیج‌کننده در تراکنش‌های بیت‌کوین است که باید آن را به خوبی به خاطر بسپارید، چون اگر تراکنش بسازید که ورودی‌های آن به درستی خرج نشده باشند، ممکن است ناخواسته مبلغ زیادی را از دست بدهید. به عبارت دیگر، اگر یک ورودی بزرگ دارید (مانند مثال تراکنش آليس)، باید خروجی تتمه را به دقت محاسبه کنید، چون در غیر این صورت معنی‌چی هر آنچه را (پس از کسر خروجی‌ها) از ورودی باقی بماند، به عنوان یک انعام بزرگ تصاحب خواهد کرد! برای مثال، اگر برای پرداخت ۱ بیت‌کوین از یک UTXO ۲۰-بیت‌کوینی استفاده کنید، باید در تراکنش خود یک خروجی تتمه‌ی ۱۹-بیت‌کوینی هم قرار دهید، در غیر این صورت تمام ۱۹ بیت‌کوین «باقیمانده» به عنوان کارمزد تلقی شده و توسط معنی‌چی (خوشبخت) تصاحب خواهد شد. هر چند به احتمال زیاد تراکنش شما با بالاترین اولویت ممکن پردازش شده و معنی‌چی آن هم بسیار خوشحال خواهد شد، ولی بعید است خودتان را برای این اشتباه بچگانه بیخشد!

اضافه نکردن «خروچی تتمه» به یک تراکنش دستی درست مثل آن است که بعد از دادن یک اسکناس به فروشنده بگویید «بقيه اش مال خودت». در تراکنش‌هایی که اختلاف مجموع ورودی‌ها با مجموع خروجی‌ها بسیار زیاد باشد، بعید است چنین قصدی داشته باشد!



اجازه دهید با بررسی دوباره‌ی تراکنش آليس بینیم اضافه کردن کارمزد به یک تراکنش در عمل چگونه اتفاق می‌افتد. آليس می‌خواهد ۱۵٪ بیت‌کوین برای پرداخت بهای قهقهه به باب خرج کند، و برای آن که مطمئن شود این تراکنش بالا فاصله پردازش خواهد شد، مبلغی را (مثلاً ۱٪ ۰۰۰ بیت‌کوین) به عنوان کارمزد تراکنش به آن اضافه می‌کند. این بدان معناست که هزینه‌ی کل این تراکنش برای آليس ۱۶٪ ۰ بیت‌کوین خواهد بود. در نتیجه، کیف‌پول آليس باید یک یا چند UTXO پیدا کند که مجموع مبلغ آنها ۱۶٪ ۰ بیت‌کوین یا بیشتر شود، و در صورت لزوم (بیشتر بودن UTXO از ۱۶٪ ۰ بیت‌کوین) یک خروچی تتمه نیز تولید کند. فرض کنید یک UTXO به مبلغ ۲٪ ۰ بیت‌کوین در کیف‌پول آليس وجود دارد. بتایرانی کیف‌پول آليس باید این UTXO را خرج کرده، و یک خروچی به مبلغ ۱۵٪ ۰ بیت‌کوین برای پرداخت بهای قهقهه، و یک خروچی دیگر به مبلغ ۱۸٪ ۰ بیت‌کوین به عنوان تتمه تولید کند. مقدار باقیمانده، یعنی تفاصل مجموع این دو خروچی (۱۸٪ ۰ + ۱۵٪ ۰ = ۳۳٪ ۰) ورودی ۲٪ ۰-بیت‌کوینی، که معادل ۱٪ ۰۰۰ بیت‌کوین است، همان کارمزد تراکنش خواهد بود.

اجازه دهد مثالی متفاوت را بررسی کنیم. اوژنی (مدیر مؤسسه‌ی خیریه کودکان در فیلیپین) بعد از فراغت اجازه دهد که مجموع مبلغ آنها به ۵۰ بیت‌کوین می‌رسد. بنابراین، کیف‌پول اوژنی پُر است از پرداخت‌های (UTXO) دریافت می‌کند که محلی صدها دفترچه بخرد و پول آنها را با بیت‌کوین پرداخت کند.

بیار خرد، او اکنون می‌خواهد از یک تولیدکننده‌ی محلی پرداخت آن را با بیت‌کوین پرداخت کند. از آنجاکه بهای این خرید (خروجی تراکنش) بسیار زیاد است، کیف‌پول اوژنی مجبور است برای پرداخت آن از دهها و شاید صدها UTXO (ورودی) با مبالغ بسیار کوچک استفاده کند. تراکنشی با این تعداد زیاد ورودی بسیار بزرگتر از ۱ کیلو بایت خواهد بود، و در واقع اندازه‌ی آن حتی ممکن است به چندین کیلو بایت هم برسد. کارمزد چنین تراکنشی بسیار بیشتر از یک تراکنش معمولی (چند صد بایتی) است. همان طور که دیدیم، کیف‌پول کارمزد تراکنش را بر مبنای تعداد بایت‌های آن محاسبه می‌کند. بسیاری از برنامه‌های کیف‌پول در تراکنش‌های بزرگ حتی کارمزد برابر نسبت به کارمزد تخمینی در نظر می‌گیرند، تا مطمئن شوند تراکنش آنها بلا فاصله پردازش خواهد شد. این بیشتری نسبت به کارمزد تخمینی در نظر می‌گیرند، تا مطمئن شوند تراکنش آنها بلا فاصله پردازش خواهد شد. این کارمزد بزرگتر است؛ همان طور که قبل از گفتیم، کارمزد یک تراکنش هیچ ارتباطی با مبلغ آن ندارد. از نظر حجم بزرگتر است.

اسکریپت تراکنش و زبان «اسکریپت»

زبان اسکریپتنویسی تراکنش‌های بیت‌کوین، موسوم به اسکریپت (Script)، یک زبان اجرایی شبه-فورت پشت‌محور با نمادگذاری پسوندی است. اگر از این کلمات و اصطلاحات عجیب و نامفهوم سر در نمی‌آورید، مشکل از شمانیست [شانس آوردهای در دهه ۶۰ و ۷۰ برنامه‌نویسی نمی‌کنید]؛ نگران نباشید، در ادامه آنها را توضیح خواهیم داد. اسکریپت قفل کننده که در UTXO قرار داده می‌شود) و اسکریپت بازکننده‌ی قفل هر دو با این زبان اسکریپتنویسی نوشته می‌شوند. وقتی یک تراکنش اعتبارسنجی می‌شود، تک تک اسکریپت‌های بازکننده‌ی قفل در تمام ورودی‌های این تراکنش به موازات اسکریپت‌های

قابل‌متوجه آنها اجرامی شوند تا مشخص شود آیا شروط خرج کردن آن ورودی‌ها را برأورده می‌کنند یا خیر.

«اسکریپت» یک زبان بسیار ساده است که برای اجراشدن روی هر نوع سخت‌افزار (شاید حتی ساده‌ترین سخت‌افزارها مثل یک اسباب‌بازی سخنگو یا ساعت دیجیتال) طراحی شده است. این زبان به حداقل قدرت پردازش نیاز دارد و قادر به انجام بسیاری از کارهایی که در زبان‌های برنامه‌نویسی مدرن می‌توانند انجام دهند، نیست. زبان «اسکریپت» عامده‌انه چنین طراحی شده است، و از نظر امنیتی این یک ویژگی مثبت آن محسوب می‌شود، چون تنها

کاربرد فعلی این زبان اعتبارسنجی تراکنش‌ها در پول‌های قابل برنامه‌ریزی و ارزهای رمزبینیان است.

امروزه، اکثر تراکنش‌هایی که در شبکه‌ی بیت‌کوین پردازش می‌شوند، به صورت «پرداخت به آدرس بیت‌کوین ... [گیرنده]» هستند و بر اساس اسکریپتی موسوم به اسکریپت پرداخت-به-دَرهم-کلید-عمومی (Pay-to-Public-Key-Hash script) یا P2PKH یا ایجاد می‌شوند. با این حال، تراکنش‌های بیت‌کوین محدود به اسکریپت «پرداخت به آدرس بیت‌کوین ...» نیستند. در حقیقت، اسکریپت‌های قفل کننده را می‌توان به گونه‌ای نوشت که شروط بسیار پیچیده‌تری را بیان کنند. برای درک این اسکریپت‌های پیچیده‌تر، ابتدا باید با مبانی اسکریپت تراکنش و زبان «اسکریپت» آشنا شویم.

در این قسمت اصول اولیه‌ی زبان اسکریپتنویسی بیت‌کوین را تشریح می‌کنیم، و نشان می‌دهیم که چگونه می‌توان از آن برای بیان شروط ساده برای خرج کردن یک مبلغ بیت‌کوین استفاده کرد، و این شروط چطور می‌توانند توسط اسکریپت‌های قفل کننده برآورده شوند.

اعتبارسنجی تراکنش‌های بیت‌کوین بر اساس یک الگوی ثابت بنا نشده است، بلکه از طریق اجرای یک زبان اسکریپت‌نویسی حاصل می‌شود. با این زبان می‌توان شروط بسیار متنوعی (تقریباً بی‌نهایت) را بیان کرد. همین ویژگی است که بیت‌کوین را به یک پول قابل برنامه‌ریزی (programmable money) تبدیل کرده است.



ناتمامیت تورینگ

زبان اسکریپت‌نویسی تراکنش‌های بیت‌کوین، موسوم به «اسکریپت»، عملگرهای متعددی دارد، ولی از یک جنبه مهم به طور عامدانه محدود شده است: در این زبان از حلقه و ساختارهای کنترلی پیچیده خبری نیست و فقط دستورات شرطی ساده در آن وجود دارد. از این نظر زبان اسکریپت‌نویسی بیت‌کوین یک زبان «تورینگ-کامل» نیست، یعنی پیچیدگی چندانی ندارد و زمان اجرای یک اسکریپت تا حد زیادی قابل پیش‌بینی است. به عبارت دیگر، «اسکریپت» یک زبان برنامه‌نویسی همه-منظوره نیست. این محدودیت‌ها تضمین می‌کنند که از زبان «اسکریپت» نتوان برای جاگذاری «بمب‌های منطقی» (مثل حلقه‌های بی‌پایان) در تراکنش‌ها و حمله به شبکه‌ی بیت‌کوین و از کار انداختن آن استفاده کرد. به یاد داشته باشید که هر تراکنش بیت‌کوین توسط تمامی گره‌های کامل در سرتاسر شبکه می‌باشد. اسکریپت محدود مانع از آن می‌شود تا از ساز و کار اعتبارسنجی بیت‌کوین برای حمله و آسیب زدن به این شبکه استفاده شود.

اعتبارسنجی بدون حالت

زبان اسکریپت‌نویسی تراکنش‌های بیت‌کوین زبانی بدون حالت (stateless) است، یعنی قبل از اجرای یک اسکریپت هیچ حالت خاصی مفروض دانسته نمی‌شود، و بعد از اجرای آن هم هیچ چیزی ثبت نخواهد شد. به عبارت دیگر، هر اطلاعاتی که برای اجرای یک اسکریپت لازم است، در درون آن اسکریپت قرار دارد. یک اسکریپت روی هر سیستم تقریباً به یک صورت اجرا خواهد شد. اگر سیستم شما بتواند یک اسکریپت را اعتبارسنجی کند، می‌توانید مطمئن باشید که هر سیستم دیگری در شبکه‌ی بیت‌کوین نیز قادر به اعتبارسنجی آن خواهد بود؛ به عبارت دیگر، یک تراکنش معتبر در همه جا و برای همه کس معتبر است، و همه هم این موضوع را می‌دانند. این پیش‌بینی پذیر بودن نتایج اجرای اسکریپت‌ها یکی از مزایای اساسی سیستم بیت‌کوین است.

ساختار اسکریپت (قفل کردن + باز کردن قفل)

موتور اعتبارسنجی تراکنش بیت‌کوین برای اعتبارسنجی تراکنش‌ها به دو نوع اسکریپت منکی است: اسکریپت قفل‌کننده (locking script) و اسکریپت بازکننده‌ی قفل (unlocking script).

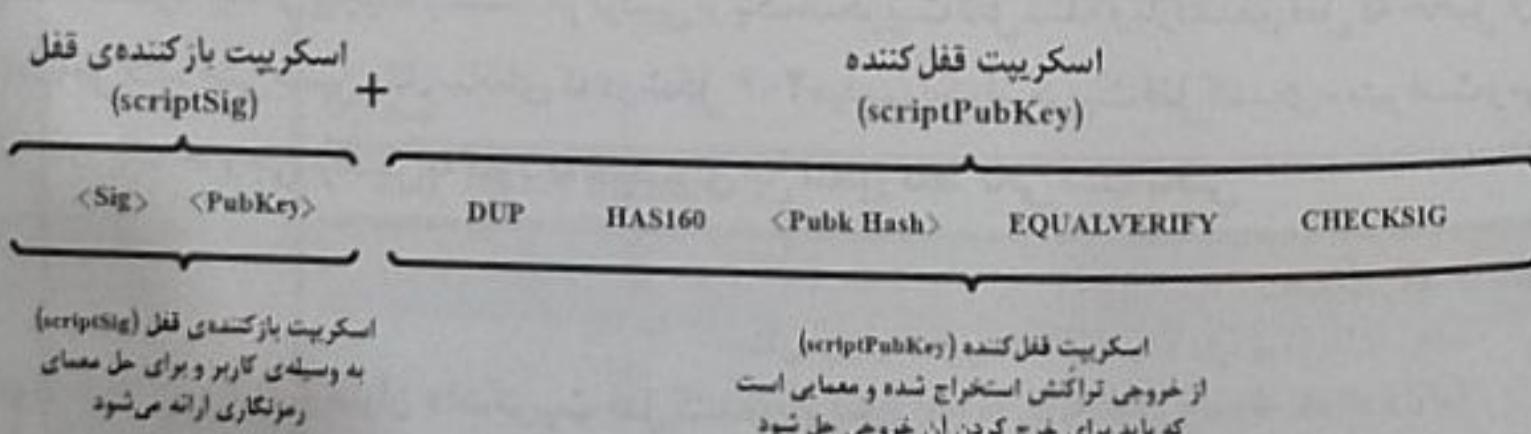
اسکریپت قفل‌کننده شرط خروج کردن یک خروجی را بیان می‌کند: این اسکریپت می‌گوید که برای خروج کردن آن خروجی در آینده چه شروطی باید برآورده شوند. از نظر تاریخی، به اسکریپت قفل‌کننده `scriptPubKey` می‌شود، چون معمولاً حاوی یک کلید عمومی یا آدرس بیت‌کوین (رشته‌ی درهم‌شده کلید عمومی) است. در این کتاب برای تأکید بر آن که فناوری اسکریپت‌نویسی بیت‌کوین حوزه‌ی احتمالات بسیار گسترده‌تری را پوشش می‌دهد از همان اصطلاح «اسکریپت قفل‌کننده» استفاده خواهیم کرد. در اکثر برنامه‌های کاربردی بیت‌کوین، آنچه به نام اسکریپت قفل‌کننده می‌شناسیم، در گذ منبع با نام `scriptPubKey` ظاهر خواهد شد؛ البته ممکن است در برخی

مراجع با اسمی اسکریپت شاهد (witness script) یا معماه رمزنگاری (cryptographic puzzle) نیز به آن اشاره شود؛ این نام‌ها همگی به یک معنا هستند. اسکریپت بازکننده‌ی قفل اسکریپتی است که شروط خرج کردن یک خروجی (که در اسکریپت قفل کننده بیان شده است) را برآورده می‌کند [یا به اصطلاح آن معما را «حل می‌کند»]، و اجازه می‌دهد کاربر آن خروجی را خرج کنند. اسکریپت‌های بازکننده‌ی قفل بخشی از ورودی هر تراکنش هستند. در اکثر مواقع، اسکریپت بازکننده‌ی قفل کنند. امضای دیجیتال تولیدشده به وسیله‌ی کیف‌پول کاربر است، امضایی که از کلید خصوصی او ساخته می‌شود. حاوی امضای دیجیتال اسکریپت بازکننده‌ی قفل **scriptSig** گفته می‌شود، چون این اسکریپت معمولاً حاوی یک از نظر تاریخی، به اسکریپت بازکننده‌ی قفل **scriptSig** می‌شود. در این اسکریپت بازکننده‌ی قفل می‌شانسیم، در کُد منبع با نام **scriptSig** ظاهر خواهد شد؛ البته گاهی از اسکریپت بازکننده‌ی قفل هم با عنوان اسکریپت شاهد نام برده می‌شود. در این کتاب برای تأکید بر محدوده‌ی بسیار گسترده‌تر ملزمات اسکریپت قفل کننده از اسکریپت بازکننده‌ی قفل «استفاده خواهیم کرد، چون همه‌ی اسکریپت‌های بازکننده‌ی قفل حاوی امضای دیجیتال نیستند.

هر گره کامل بیت‌کوین (گره‌هایی که اعتبارسنجی تراکنش‌هارا انجام می‌دهند)، برای اعتبارسنجی یک تراکنش، اسکریپت‌های قفل کننده و بازکننده‌ی قفل آن را به موازات هم اجرا می‌کند. در هر ورودی تراکنش یک اسکریپت بازکننده‌ی قفل وجود دارد و به یک UTXO موجود ارجاع می‌کند. برنامه‌ی اعتبارسنجی اسکریپت بازکننده‌ی قفل را از ورودی تراکنش کپی می‌کند، سپس UTXO اشاره‌شده در این ورودی را بازیابی کرده و اسکریپت قفل کننده را هم از آن UTXO کپی می‌کند. پس از آن اسکریپت‌های بازکننده‌ی قفل و قفل کننده به دنبال هم اجرا می‌شوند. اگر اسکریپت بازکننده‌ی قفل شروط بیان شده در اسکریپت قفل کننده را برآورده کند، آن ورودی معتبر است. برای تعیین اعتبار یک تراکنش، تمام ورودی‌های آن به طور مستقل اعتبارسنجی می‌شوند.

توجه کنید که این UTXO به طور دائمی در بلاک‌چین ثبت شده است، بنابراین تغییرناپذیر بوده و هر تعداد نلاش ناموفق برای خرج کردن آن تأثیری بر اعتبار این UTXO نخواهد داشت. فقط یک تراکنش معتبر (موفق) که شروط بیان شده در این خروجی را برآورده سازد، منجر به آن خواهد شد که این خروجی «خرج شده» تلقی شده و از مجموعه‌ی UTXO (خرجی‌های تراکنش خرج نشده) حذف شود.

نمونه‌ای از اسکریپت‌های بازکننده‌ی قفل و قفل کننده در رایج‌ترین نوع تراکنش یست‌کوین (تراکنش پرداخت به درهم کلید عمومی) را در شکل ۳-۶ مشاهده می‌کنید؛ همان طور که در این شکل می‌بینید، اسکریپت حاصل از به هم چسباندن اسکریپت بازکننده‌ی قفل و اسکریپت قفل کننده قبل از اعتبارسنجی اسکریپت به دست آمده است.



شکل ۳-۶ ترکیب **scriptPubKey** و **scriptSig** برای اعتبارسنجی یک اسکریپت تراکنش.

پشته‌ی اجرای اسکریپت

زبان اسکریپتنویسی بیت‌کوین یک زبان پشته-محور است، به این معنا که برای اجرای اسکریپت‌ها از ساختمان داده‌ای موسوم به پشته (stack) استفاده می‌کند. پشته یک ساختمان داده‌ی بسیار ساده است که می‌توان آن را مثل چند ورق کاغذ که روی یکدیگر چیده شده‌اند، تصور کرد. پشته فقط دو عملکرد دارد: داخل فرستادن (push) و بیرون کشیدن (pop). عمل داخل فرستادن یک درایه (مدخل) را به بالای پشته اضافه می‌کند (مثل ورقی که روی یک دسته ورق کاغذی قرار می‌دهید)؛ و عمل بیرون کشیدن یک درایه (مدخل) را از بالای پشته برمی‌دارد (مثل ورقی که از روی یک دسته ورق کاغذ برمی‌دارید). هر دو عملکرد پشته روی بالاترین درایه عمل می‌کنند، به همین دلیل به پشته «صف LIFO» (Last-In-First-Out) نیز گفته می‌شود زبان اسکریپتنویسی بیت‌کوین یک اسکریپت را از چپ به راست خوانده و پردازش می‌کند. اعداد (داده‌های ثابت) به داخل یک پشته فرستاده می‌شوند. عملگرهای ریاضی-منطقی یک یا چند پارامتر را از پشته بیرون کشیده یا به داخل آن می‌فرستند، و در برخی موارد حاصل آن عمل را به داخل پشته برمی‌گردانند. برای مثال، عملگر ADD_OP دو درایه (پارامتر) را از پشته بیرون می‌کشد، آنها را جمع می‌کند، و حاصل جمع را به داخل پشته برمی‌گرداند.

عملگرهای شرطی یک شرط را ارزیابی می‌کنند؛ حاصل ارزیابی یک عبارت شرطی می‌تواند یکی از دو مقدار TRUE (درست) یا FALSE (نادرست) باشد. برای مثال، عملگر EQUAL_OP دو درایه (پارامتر) را از پشته بیرون می‌کشد، و آنها را مقایسه می‌کند؛ اگر این دو پارامتر مساوی باشند، مقدار TRUE (که با عدد ۱ نشان داده می‌شود) به داخل پشته برگردانده می‌شود، و در غیر این صورت مقدار FALSE (که با ۰ نشان داده می‌شود) به داخل پشته برگردانده خواهد شد. اسکریپت تراکنش‌های بیت‌کوین معمولاً حاوی یک عملگر شرطی است، که اگر مقدار TRUE برگرداند، به معنای معتبر بودن آن تراکنش است.

یک اسکریپت ساده

اجازه دهید با بررسی دو مثال ساده طرز کار اسکریپت‌های بیت‌کوین و پشته را نشان دهیم.
در شکل ۴-۶ چگونگی اجرای اسکریپت اسکریپت ۳ ۲ ۵ ۰P_EQUAL ۰P_ADD ۳ ۰P_EQUAL ۵ ۰P_ADD ۲ را مشاهده می‌کنید. این اسکریپت در عدد را جمع کرده و سپس نتیجه را با یک عدد دیگر مقایسه می‌کند. عملگر ریاضی ADD_OP دو عملوند (پارامتر) ۲ و ۳ را از پشته بیرون می‌کشد (pop)، آنها را با یکدیگر جمع می‌کند و حاصل جمع را به داخل پشته برمی‌گرداند؛ سپس عملگر شرطی EQUAL_OP این حاصل جمع را با عملوند ۵ مقایسه کرده و نتیجه‌ی این مقایسه (که در این مثال مقدار TRUE دارد) را به داخل پشته می‌فرستد. در این شکل، برای رعایت اختصار و شفافیت، پیشوند OP را حذف کرده‌ایم. [برای اطلاعات بیشتر درباره عملگرهای توابع مجاز در اسکریپت‌های بیت‌کوین به پیوست‌های کتاب نگاه کنید.]

هر چند در اکثر تراکنش‌های بیت‌کوین اسکریپت قفل کننده به یک درهم کلید عمومی (که در واقع همان آدرس بیت‌کوین است) اشاره می‌کنند، و در نتیجه برای خرج کردن آنها لازم است مالکیت کاربر بر آن بیت‌کوین اثبات شود، این اسکریپت‌ها معمولاً چندان پیچیده نیستند. هر ترکیبی از یک اسکریپت قفل کننده و بازکننده قفل که حاصل آن باشد، کفایت می‌کند. حتی همین مثال ساده‌ای که در شکل ۴-۶ دیدیم، یک اسکریپت قفل کننده‌ی معتبر است و می‌تواند به عنوان «قفل» در یک خروجی تراکنش به کار برده شود. برای این منظور فقط کافی است بخش

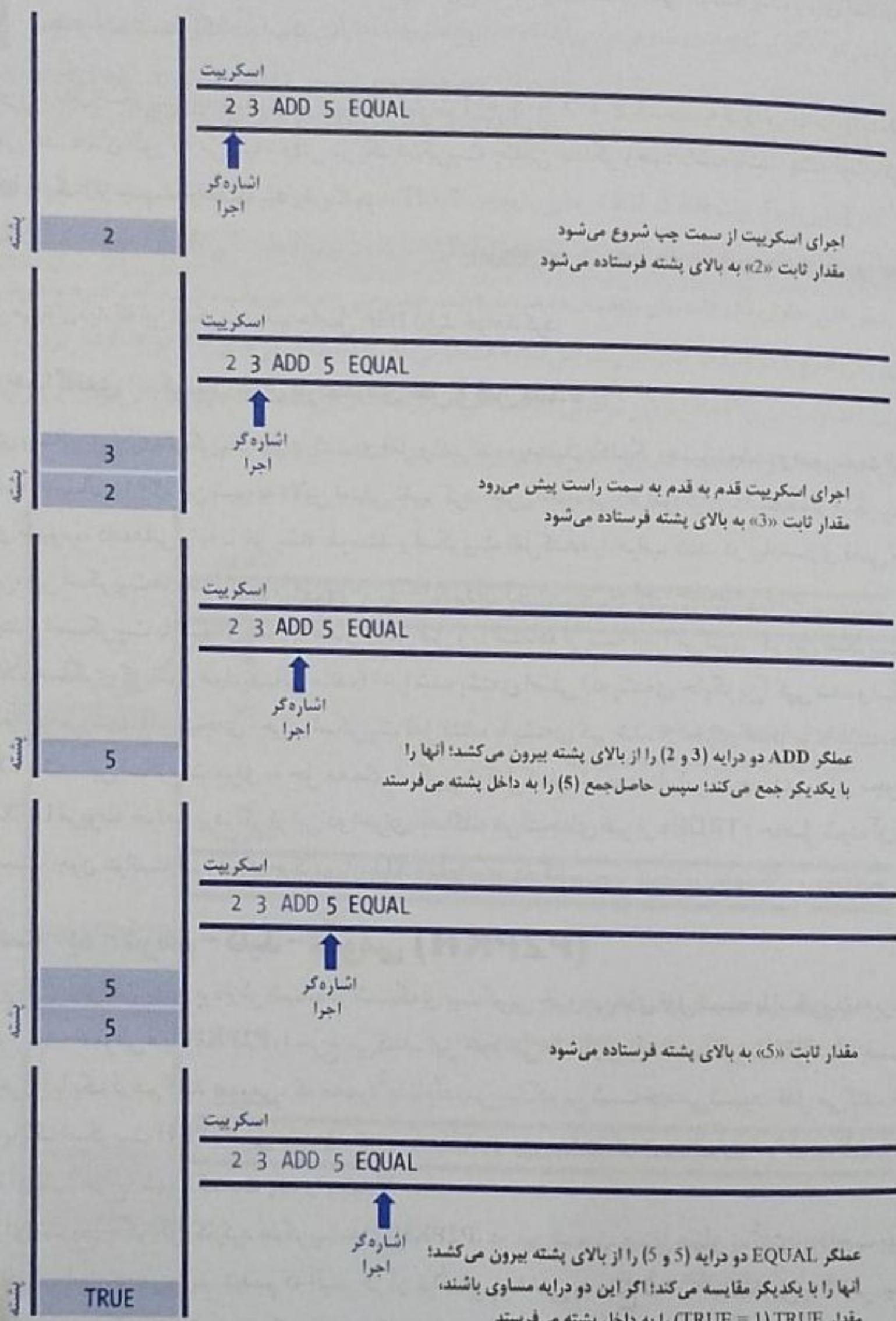
۳ ۰P_ADD ۵ ۰P_EQUAL

از اسکریپت شکل ۴-۶ را به عنوان «اسکریپت قفل کننده»، و بخش

آن را به عنوان «اسکریپت بازگشته‌ی قفل» به کار ببریم تا در ترکیب با یکدیگر یک اسکریپت «بازگشته‌ی قفل + قفل‌گشته» معتبر با حاصل OP_TRUE ایجاد کنند:

`2 3 OP_ADD 5 OP_EQUAL`

نه تنها اسکریپت `OP_ADD 5 OP_EQUAL` یک اسکریپت قفل‌گشته‌ی معتبر است، بلکه هر کسی با دانستن این که عدد ۲ جواب این معما است، می‌تواند این UTXO را مرزگشایی و خرج کندا



شکل ۴-۶ اعتبارسنجی اسکریپت بیت‌کوین چیزی جز چند عمل ساده‌ی ریاضی نیست.

تراکنش‌های معتبر تلقی می‌شوند که پس از اجرای اسکریپت «بازکننده قفل + قفل کننده»، آخرین مقدار باقیمانده در پشتۀ TRUE (که به صورت 0x01 نشان داده می‌شود) یا هر مقدار غیر صفر دیگر باشد؛ خالی بودن پشتۀ بعد از اجرای اسکریپت نیز به مثابه TRUE در نظر گرفته خواهد شد. تراکنش‌هایی که پس از اجرای اسکریپت «بازکننده قفل + قفل کننده»، مقدار بالای پشتۀ FALSE (که با رشته‌ی خالی ۱۱ نشان داده می‌شود) باشد، یا اجرای اسکریپت توسط یک عملگر مانند OP_RETURN, OP_VERIFY, یا یک عملگر پایان‌دهنده شرط مثل OP_ENDIF، به طور صریح قطع شود، نامعتبر تلقی خواهند شد. (برای اطلاعات بیشتر، پیوست‌های کتاب را ببینید.)

در زیر مثال اسکریپت پیچیده‌تری می‌بینید که عبارت $1 + 3 - 7 + 2$ را محاسبه کرده و حاصل را با مقدار ۷ مقایسه می‌کند. همان طور که می‌بینید، وقتی در یک اسکریپت چندین عملگر وجود داشته باشد، پشتۀ اجازه می‌ده آنها را یک به یک (از چپ به راست) پردازش کنیم:

2 7 OP_ADD 3 OP_SUB 1 OP_ADD 7 OP_EQUAL

به سادگی می‌توان دید که این اسکریپت هم حاصل TRUE تولید خواهد کرد.

اجرای جداگانه‌ی اسکریپت‌های بازکننده‌ی قفل و قفل کننده

در مشتری بیت‌کوین اولیه، اسکریپت‌های بازکننده‌ی قفل و قفل کننده به دنبال یکدیگر چسبانده شده و صورت متوالی اجرا می‌شوند. در سال ۲۰۱۰، این شیوه به دلایل امنیتی تغییر کرد، چون اجازه می‌داد تفوذگران با استفاده از اسکریپت‌های بازکننده‌ی معیوب، داده‌هایی را به داخل پشتۀ بفرستند و اسکریپت قفل کننده را خراب کنند. در پیاده‌سازی فعلی مشتری بیت‌کوین، این اسکریپت‌ها جداگانه اجرا می‌شوند و پشتۀ به روش زیر بین این دو اجرا جابجا می‌شود.

ابتدا، اسکریپت بازکننده‌ی قفل به همان روش قبل و با استفاده از پشتۀ اجرا می‌شود. اگر این اسکریپت بدون خطأ (مثلاً، عملگری که بدون عملوند باقی مانده) اجرا شد، پشتۀ اصلی (نه پشتۀ جایگزین) کپی شده و اسکریپت قفل کننده اجرا می‌شود. اگر نتیجه‌ی اجرای اسکریپت قفل کننده با پشتۀ کپی شده از اجرای اسکریپت بازکننده‌ی قفل قفل کننده اجرا می‌شود، این اسکریپت موفق به حل معما می‌شود. و بنابراین آن ورودی دارای مجوز خرج TRUE باشد، این اسکریپت موفق به حل معما می‌شود. اگر از این دو اجرای جداگانه هر نتیجه‌ای غیر از «TRUE» حاصل شود، آن ورودی کردن UTXO مربوطه خواهد بود. اگر از این دو اجرای جداگانه هر نتیجه‌ای غیر از «TRUE» حاصل شود، آن ورودی معتبر نیست، چون نتوانسته شروط خرج کردن آن UTXO را برآورده کند.

پرداخت-به-درهم-کلید-عمومی (P2PKH)

اکثریت بزرگی از تراکنش‌های پردازش شده در شبکه‌ی بیت‌کوین خروجی‌های قفل شده با اسکریپت «پرداخت به-درهم-کلید-عمومی» یا P2PKH را خرج می‌کنند. این خروجی‌ها حاوی یک اسکریپت قفل کننده هستند که آن خروجی را با یک درهم کلید عمومی، که معمولاً با نام آدرس بیت‌کوین شناخته می‌شود، قفل می‌کند. خروجی قفل شده با یک اسکریپت P2PKH می‌تواند با ارائه‌ی یک کلید عمومی و امضای دیجیتال ایجاد شده با کلید خصوصی متناظر با آن باز (خرج) شود (قسمت بعد را ببینید).

برای تشریح چگونگی کارکرد اسکریپت‌های P2PKH، در این قسمت هم از همان تراکنش پرداخت بهای فهره از آلیس به باب استفاده می‌کنیم. دیدیم که آلیس در آن تراکنش پرداختی به مبلغ ۱۵٪ بیت‌کوین به آدرس بیت‌کوین مغازه‌ی قهوه‌فروشی باب انجام داد. اسکریپت قفل کننده‌ی خروجی آن تراکنش چنین بود:

OP_DUP OP_HASH160 <Cafe Public Key Hash> OP_EQUALVERIFY OP_CHECKSIG

که در آن **Cafe Public Key Hash** معادل آدرس بیت‌کوین مغازه‌ی باب (بدون کدگذاری Base58Check) است. اکثر برنامه‌های کاربردی این دارم کلید عمومی را با فرمت هگزادسیمال نمایش می‌دهند، نه با فرمت آشنای Base58Check آدرس‌های بیت‌کوین که همیشه با «۱» شروع می‌شود.

این اسکریپت قفل‌کننده را می‌توان با یک اسکریپت بازکننده قفل به صورت زیر باز کرد:

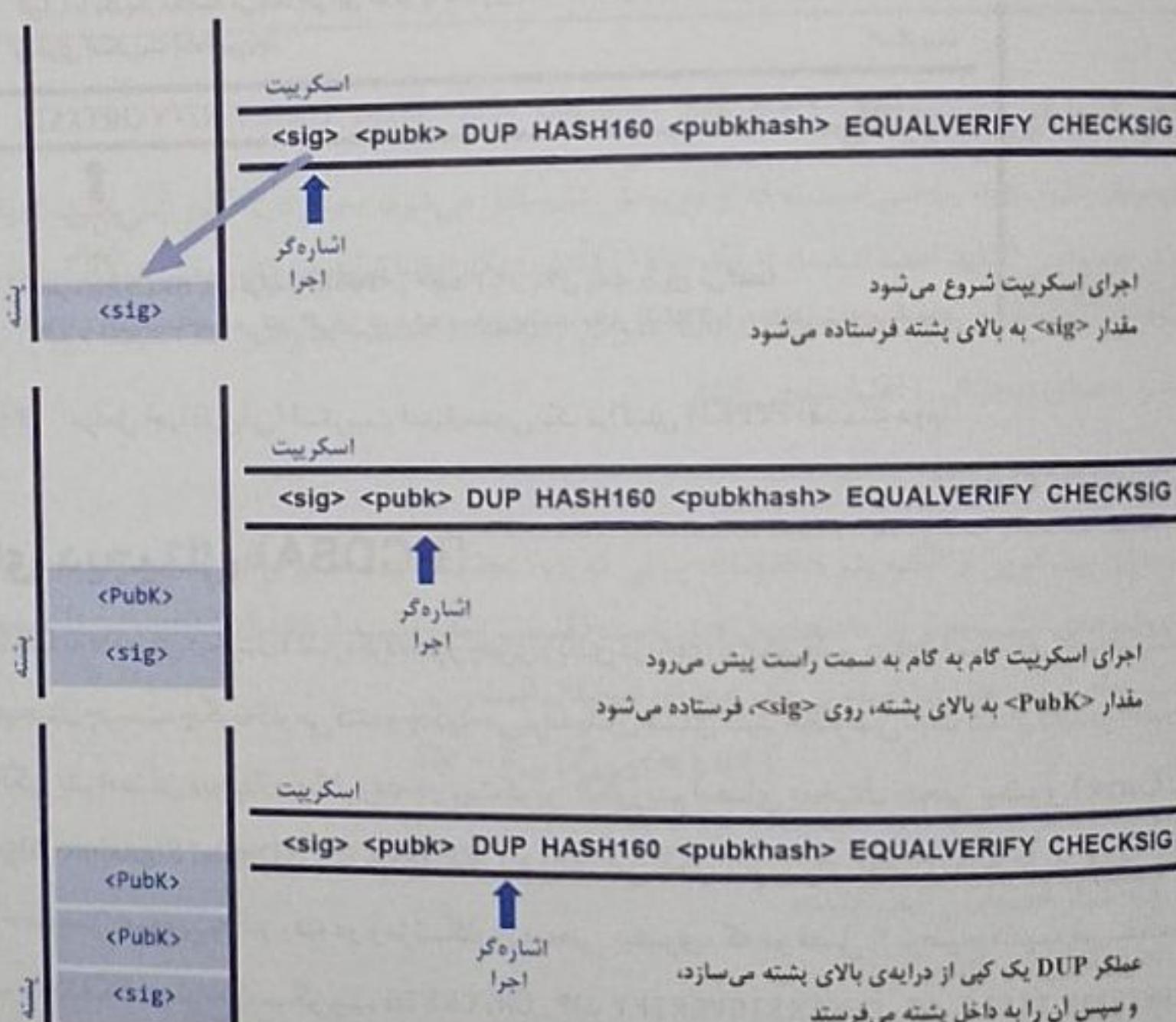
<Cafe Signature> <Cafe Public Key>

مجموع این دو اسکریپت به صورت زیر در می‌آید که همان اسکریپت اعتبارسنجی تراکنش آلیس خواهد بود:

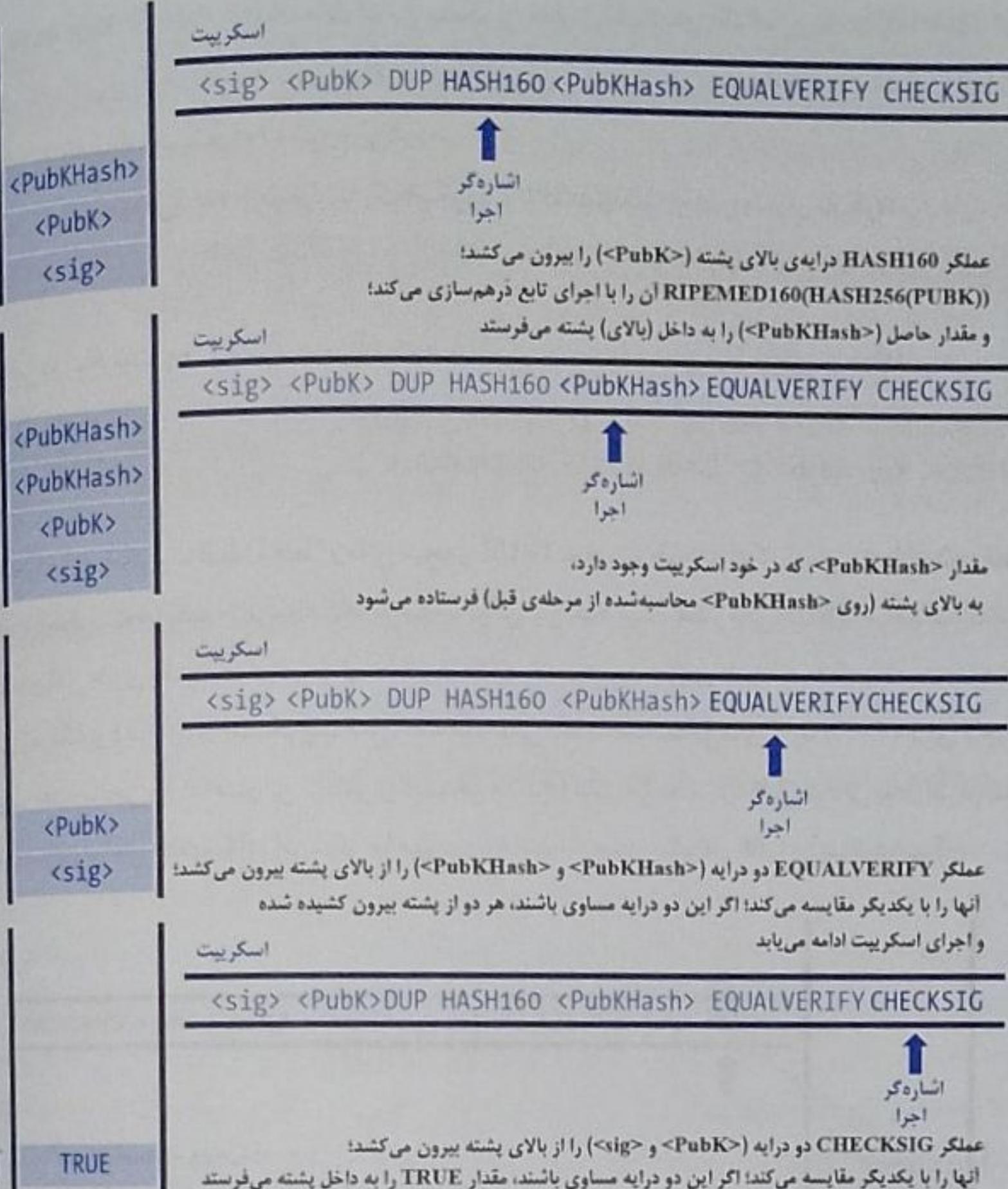
<Cafe Signature> <Cafe Public Key> OP_DUP OP_HASH160

<Cafe Public Key Hash> OP_EQUALVERIFY OP_CHECKSIG

اجرای این اسکریپت فقط (و فقط) زمانی نتیجه‌ی TRUE به دست خواهد داد که اسکریپت بازکننده قفل شروط یازشده در اسکریپت قفل‌کننده را برابر آورده کند. به عبارت دیگر، این اسکریپت فقط زمانی TRUE خواهد شد که اسکریپت بازکننده قفل حاوی یک «امضای دیجیتال» معتبر از «کلید خصوصی» مغازه‌ی باب باشد که با «دارم کلید عمومی» (معماًی رمزنگاری) موجود در اسکریپت قفل‌کننده همخوانی داشته باشد. شکل‌های ۶-۵ و ۶-۶ اجرای گام به گام این اسکریپت اعتبارسنجی را (که اعتبار این تراکنش را ثابت خواهد کرد) نشان می‌دهند. [به دلیل طولانی بودن این فرآیند آن را در دو شکل متوالی نمایش داده‌ایم.] [در این شکل‌ها هم، برای رعایت اختصار، پیشوند OP را حذف کرده‌ایم.]



شکل ۶-۵ مراحل اجرا (ارزیابی) اسکریپت اعتبارسنجی یک تراکنش P2PKH (قسمت اول).



شکل ۶-۶ مراحل اجرا (ارزیابی) اسکریپت اعتبارسنجی یک تراکنش P2PKH (قسمت دوم).

امضای دیجیتال (ECDSA)

تابه اینجا فقط به «امضای دیجیتال» اشاره کرده‌ایم و چیزی زیادی درباره‌ی آن نگفته‌ایم. در این قسمت می‌خواهیم نشان دهیم امضا دیجیتال چیست، چگونه کار می‌کند، و چگونه می‌تواند بدون افشای کلید خصوصی ثابت کند آن کلید در اختیار ماست. الگوریتم امضا دیجیتال به کار رفته در بیت‌کوین الگوریتم امضا دیجیتال منحنی بیضوی (Elliptic Curve)، یا ECDSA، نام دارد. این الگوریتم برای امضا دیجیتال از همان جهت‌های کلی خصوصی/عمومی به کار رفته در رمزگاری منحنی بیضوی، که در فصل ۳ توضیح دادیم، استفاده می‌کند. الگوریتم ECDSA در توابع اسکریپت OP_CHECKMULTISIG، OP_CHECKSIGVERIFY، OP_CHECKSIG و OP_CHECKMULTISIGVERIFY مورد استفاده قرار می‌گیرد. این توابع در اسکریپت قفل کننده به کار گرفته می‌شوند، و اسکریپت بازکننده قفل باید حاوی یک امضا ECDSA باشد.

امضای دیجیتال در بیت کوین سه هدف را برآورده می کند. اول، این امضان ثابت می کند که صاحب کلید خصوصی که امضا دیجیتال از آن مشتق شده (کسی که به طور ضمنی مالک آن بیت کوین نیز محسوب می شود)، اجازه خروج کردن آن پول را صادر کرده است. دوم، صدور این اجازه غیرقابل انکار است (انکارناپذیری). سوم، امضای دیجیتال ثابت می کند که این تراکنش (یا بخش های مشخصی از این تراکنش) بعد از امضان شدن دستکاری نشده است و نمی تواند دستکاری شود.

توجه کنید که هر یک از ورودی های تراکنش به طور مستقل امضامی شوند. این یک ویژگی کلیدی و بسیار مهم است، چون لزومی ندارد همهٔ ورودی های یک تراکنش متعلق به یک شخص واحد باشند. در حقیقت، نوع خاصی از تراکنش موسوم به «تنوع-سکه» وجود دارد که از همین ویژگی برای امنیت و محرومگی بیشتر تراکنش ها استفاده می کند.

هر ورودی یک تراکنش و هر امضایی که ممکن است در آن وجود داشته باشد، به طور کامل از تمامی ورودی ها و امضاهای دیگر آن تراکنش مستقل است. به عبارت دیگر، چند نفر می توانند در ساخت یک تراکنش واحد مشارکت کرده و هر کدام ورودی مخصوص خود را امضانند.

تعريف «امضای دیجیتال» در ویکی پدیا

امضای دیجیتال یک طرح یا تمهید ریاضی برای نشان دادن اصالت یک پیام یا سند دیجیتال است. یک امضای دیجیتال معتبر این اطمینان را به گیرنده می دهد که پیام به وسیلهٔ همان فرستنده ای که ادعامی شود، ایجاد شده است (اصالت پیام)، این که فرستنده نمی تواند ارسال آن را تکذیب کند (انکارناپذیری پیام)، و این که پیام در فرآیند ارسال دستکاری نشده است (یکپارچگی پیام).

منبع: https://en.wikipedia.org/wiki/Digital_signature

طرز کار امضای دیجیتال

امضای دیجیتال یک تمهید ریاضی است که از دو بخش تشکیل می شود. بخش اول الگوریتمی است که با استفاده از یک کلید خصوصی (کلید امضانکننده)، از یک پیام (تراکنش) یک امضا تولید می کند. بخش دوم الگوریتمی است که به همگان اجازه می دهد با استفاده از یک کلید عمومی (منتظر با کلید خصوصی امضانکننده) و پیام (تراکنش) دریافتی، این امضای دیجیتال را اعتبارسنجی کنند.

تولید یک امضای دیجیتال

در پیاده سازی بیت کوین از الگوریتم ECDSA، پیامی که باید امضا شود، همان تراکنش، یا به بیان دقیق تر رشتهٔ ذرهم زیر مجموعهٔ مشخصی از داده های تراکنش است (قسمت بعد را بینید). کلید امضانکننده نیز کلید خصوصی کاربر است. حاصل این عملیات ریاضی یک امضای دیجیتال است:

$$Sig = F_{sig}(F_{hash}(m), dA)$$

که در آن:

- dA کلید خصوصی امضانکننده،
- m تراکنش (یا بخشی از آن)،
- F_{hash} تابع ذرهم سازی،
- F_{sig} الگوریتم امضانکننده، و
- Sig امضای حاصل است.

امضای Sig ، خروجی تابع F_{Sig} ، از دو بخش تشکیل می‌شود که معمولاً آنها را R و S می‌نامند:

$$\text{Sig} = (R, S)$$

بعد از محاسبه R و S ، این دو مقدار بر اساس یک طرح جامع کُدگذاری استاندارد بین‌المللی، موسوم به قواعد متاز کُدگذاری (DER) (Distinguished Encoding Rules) یا **DER**، به یک استریم بایت سریال‌سازی می‌شوند.

سریال‌سازی امضا (DER)

باز هم به تراکنش آليس بر می‌گردیم. در ورودی این تراکنش یک اسکریپت بازکننده قفل حاوی امضا دیجیتال با کُدگذاری DER زیر از کیف‌پول آليس دیده می‌شود:

3045022100884d142d86652a3f47ba4746ec719bbfb040a570b1deccbb6498c75c4ae24cb02204b
9f039ff08df09cbe9f6addac960298cad530a863ea8f53982c09db8f6e381301

این امضا دیجیتال چیزی نیست جز استریم بایت حاصل از سریال‌سازی مقادیر R و S ، که توسط کیف‌پول آليس برای اثبات مالکیت او بر کلید خصوصی مجاز برای خرج کردن این خروجی تولید شده است. این استریم سریال از ۹ بخش تشکیل می‌شود:

- ۰x30 - نشانگر شروع دنباله‌ی DER
- ۰x45 - طول دنباله‌ی DER (۶۹ بایت)
- 0x02 - مقدار بعدی یک عدد صحیح است (بخش R)
- 0x21 - طول این عدد صحیح (۳۳ بایت)
- مقدار R - 00884d142d86652a3f47ba4746ec719bbfb040a570b1deccbb6498c75c4ae24cb
- ۰x02 - مقدار بعدی یک عدد صحیح است (بخش S)
- 0x20 - طول این عدد صحیح (۳۲ بایت)
- مقدار S - 4b9f039ff08df09cbe9f6addac960298cad530a863ea8f53982c09db8f6e3813
- ۰x01 - نشانگر نوع درهم‌سازی امضا (در اینجا، SIGHASH_ALL)

بینید آیا می‌توانید امضا آليس را با استفاده از فهرست بالا رمزگشایی کنید. بخش‌های مهم این استریم عبارتند از R و S ، و سایر داده‌ها فقط عناصر مورد نیاز برای کُدگذاری DER هستند.

اعتبارسنجی امضا دیجیتال

برای اعتبارسنجی یک امضا دیجیتال باید سه چیز را در اختیار داشته باشیم: امضا (مقادیر R و S)، استریم سریال تراکنش، و کلید عمومی (منتظر با کلید خصوصی به کار رفته برای تولید آن امضا). اعتبارسنجی یک امضا دیجیتال در اساس یعنی «فقط مالک کلید عمومی منتظر با این کلید خصوصی می‌تواند امضا این تراکنش را تولید کرده باشد.»

الگوریتم اعتبارسنجی امضا سه پارامتر می‌گیرد: پیام (درهم تراکنش یا بخشی از آن)، کلید عمومی امضاکننده، و خود امضا (مقادیر R و S)؛ اگر امضا واقعاً متعلق به آن پیام و کلید عمومی فرستنده باشد، این الگوریتم مقدار TRUE بر می‌گرداند.

نوع درهم امضا دیجیتال (SIGHASH)

امضا دیجیتال روی پیام (که در بیت کوین همان تراکنش‌ها هستند) اعمال می‌شود. امضا دیجیتال به معنای تعجب امضاکننده نسبت به بخش مشخصی از محتويات تراکنش است. در ساده‌ترین شکل، امضا دیجیتال کل تراکنش (همی‌ورودی‌ها، خروجی‌ها و دیگر فیلد‌های تراکنش) را شامل می‌شود. با این حال، یک امضا می‌تواند فقط بخشی از داده‌های یک تراکنش را تأیید کند، که در مواردی مفید باشد (در ادامه‌ی همین فصل بیشتر در این باره توضیح خواهیم داد).

امضاهای دیجیتال بیت کوین می‌توانند مشخص کنند کدام بخش از داده‌ی یک تراکنش شامل آن امضا می‌شود؛ آنها برای این منظور از پرچم **SIGHASH** استفاده می‌کنند. پرچم **SIGHASH** یک بایت واحد است که در انتهای امضا قرار می‌گیرد. هر امضا باید پرچم **SIGHASH** را داشته باشد، ولی این پرچم می‌تواند برای هر ورودی متفاوت باشد. برای مثال، در تراکنشی که سه ورودی امضا شده دارد، هر ورودی می‌تواند با یک پرچم **SIGHASH** متفاوت امضا شده باشد، و هر امضا بخشی از آن تراکنش را امضا (تعهد) می‌کند.

به یاد داشته باشید که هر ورودی می‌تواند حاوی یک امضا در اسکریپت بازکننده‌ی قفل خود باشد. در نتیجه، تراکنشی که چندین ورودی دارد، ممکن است هر ورودی آن با یک پرچم **SIGHASH** متفاوت امضا شده باشد. همچنین توجه کنید که بک تراکنش بیت کوین می‌تواند از تعدادی ورودی متعلق به «صاحبان مختلف» تشکیل شده باشد، که هر کدام از آنها فقط بکی از ورودی‌هارا امضا کرده‌اند، و موفقیت در اعتبارسنجی این تراکنش مستلزم معتبر بودن تمامی آن امضاها است. بسیاری از انواع پرچم **SIGHASH** فقط در این قبیل سناریوها (تراکنش‌های مشارکتی، که در آنها هر نفر تنها بخشی از تراکنش را امضا می‌کند) معنا پیدا می‌کنند. سه نوع پرچم **SIGHASH** وجود دارد که آنها را در جدول ۲-۶ مشاهده می‌کنید.

جدول ۲-۶ انواع **SIGHASH** و مفهوم آنها

پرچم SIGHASH	مقدار	توضیح
ALL	0x01	امضا به تمام ورودی‌ها و خروجی‌ها اعمال می‌شود.
NONE	0x02	امضا به تمام ورودی‌ها (نه هیچ یک از خروجی‌ها) اعمال می‌شود.
SINGLE	0x03	امضا به تمام ورودی‌ها و فقط یک خروجی با اندیس همسان با ورودی امضا شده اعمال می‌شود.

همچنین یک پرچم تغییردهنده به نام **SIGHASH_ANYONECANPAY** وجود دارد که می‌تواند با هر یک از پرچم‌های جدول ۲-۶ ترکیب شده و معنای آنها را عوض کند. وقتی از پرچم **ANYONECANPAY** استفاده می‌کنید، فقط یک ورودی امضا می‌شود و سایر ورودی‌ها (و شماره ترتیب آنها) را می‌توان به طور مستقل دستکاری کرد. پرچم **ANYONECANPAY** مقدار 0x80 دارد و با پرچم **SIGHASH** به صورت بیت به بیت OR می‌شود؛ جدول ۴-۶ نتیجه‌ی ترکیب این پرچم با پرچم **SIGHASH** را نشان می‌دهد.

جدول ۴-۶ انواع **SIGHASH** همراه با پرچم تغییردهنده مفهوم آنها

پرچم SIGHASH	مقدار	توضیح
ALL ANYONECANPAY	0x81	امضا به یک ورودی و تمام خروجی‌ها اعمال می‌شود.
NONE ANYONECANPAY	0x82	امضا به یک ورودی (نه هیچ یک از خروجی‌ها) اعمال می‌شود.
SINGLE ANYONECANPAY	0x83	امضا به یک ورودی و خروجی با اندیس همسان آن اعمال می‌شود.

روش اعمال پرچم **SIGHASH** در فرآیند امضا و اعتبارسنجی چنین است: یک کپی از تراکنش گرفته می‌شود؛ برخی از فیلد‌های خاص آن حذف می‌شوند (طول آنها صفر شده و خالی می‌شوند)؛ تراکنش حاصل سریال‌سازی می‌شود؛ پرچم **SIGHASH** به انتهای رشته‌ی سریال‌شده چسبانده می‌شود؛ رشته‌ی حاصل دارهم‌سازی می‌شود؛ و این رشته‌ی دارهم همان «پیام» امضا شده است. بسته به این که کدام پرچم **SIGHASH** استفاده شود، بخش‌های متفاوتی از تراکنش حذف می‌شوند و رشته‌ی دارهم حاصل به زیرمجموعه‌ای از داده‌های تراکنش که نگه داشته شده‌اند، بستگی خواهد داشت. با اضافه کردن پرچم **SIGHASH** در آخرین مرحله‌ی قبل از دارهم‌سازی، تراکنش امضا شده حاوی نوع **SIGHASH** نیز خواهد بود، و بنابراین امکان تغییر آن (مثلاً توسط یک معدنچی متقلب) وجود نخواهد داشت.

تمام انواع SIGHASH فیلد `nLocktime` (زمان قفل) تراکنش را امضا می کنند (فصل ۷ را ببینید). علاوه بر آن، قفل از امضاشدن، نوع SIGHASH نیز به تراکنش چسبانده می شود، به طوری که پس از امضای یک تراکنش دیگر امکان دستکاری آن وجود ندارد.



در مثال تراکنش آلیس (قسمت قبل) دیدیم که آخرین بخش از امضای DER این تراکنش مقدار `01dA37` که معادل پرچم `SIGHASH_ALL` است. این پرچم داده های تراکنش را قفل می کند، به طوری که امضای آلیس حالت تمام ورودی ها و خروجی های تراکنش را حفظ خواهد کرد. این رایج ترین شکل امها است. اجازه دهد نگاهی به دیگر انواع SIGHASH و کاربرد عملی آنها بیندازیم:

ALL | ANYONECANPAY

این پرچم می تواند برای ایجاد تراکنش های «چند منبعی» به کار رود. افرادی که قصد جمع آوری کمک های مردمی را دارند، می توانند به کمک این پرچم تراکنشی با یک خروجی واحد ایجاد کنند. این خروجی واحد دارای یک «مبلغ هدف» است که آن را به گیرنده [کسی که کمک های مردمی را جمع آوری می کند] می پردازد. از آنجا که چنین تراکنشی دارای ورودی نیست، قاعده تأثیرگذار نیست؛ ولی دیگران می توانند با اضافه کردن ورودی (به عنوان اعانه) آن را به یک تراکنش معتبر تبدیل کنند. این افراد [کمک کننده] ورودی خود را با پرچم `ANYONECANPAY` امضا می کنند. تازمانی که مجموع ورودی ها به «مبلغ هدف» خروجی نرسیده باشد، این تراکنش همچنان نامعتبر می ماند. هر اعانه یک «قول» یا تعهد است، که تاریخی مجموع آنها به «مبلغ هدف» نمی تواند از دهنده گرفته شده و به گیرنده پرداخت شود.

NONE

با این پرچم می توان یک «چک حامل» یا «چک سفید» به مبلغ مشخص ایجاد کرد. این پرچم به ورودی تراکنش اعمال شده و آن را قفل می کند، ولی اجازه می دهد اسکریپت قفل کننده خروجی تغییر داده شود. هر کسی می تواند آدرس بیت کوین خود را در اسکریپت قفل کننده خروجی این تراکنش بنویسد و آن را نقد کند. با این حال، مقدار این تراکنش قابل تغییر نیست، چون توسط امضا قفل شده است.

NONE | ANYONECANPAY

این پرچم به کاربر اجازه می دهد مقادیر بسیار خرد، موسوم به «غبار»، را جمع کرده و در یک تراکنش که به آن «غبارروب» گفته می شود، تجمع کند. موقعی پیش می آید که کاربر در گیف پول خود تعداد زیادی UTXO را مبلغ بسیار جزیی دارد که نمی تواند آنها را به صورت انفرادی خرج کند چون کارمزد تراکنش از مبلغ UTXO یشنر است. به کمک این پرچم می توان تعداد زیادی UTXO غبار را در یک تراکنش تجمع کرده و آنها را یکجا خرج کرد.

پیشنهادهایی برای اصلاح یا توسعه سیستم SIGHASH ارائه شده است. یکی از این پیشنهادها که توسط گلن ویلن (از بلاک استریم) به عنوان بخشی از پروژه Elements ارائه شده، «حالت های بیت ماسک Sighash» نام دارد که هدف آن جایگزینی انواع SIGHASH با گونه ای انعطاف پذیرتر است که اجازه می دهد با «بیت ماسک های ورودی و خروجی دلخواه و قابل بازنویسی توسط معدنچی» بتوانیم تراکنش هایی به صورت «الگوهای قرارداد پیش تعهد پیچیده تر، مثل پیشنهادهای امضا شده به همراه تمه در یک تبادل دارایی توزیع شده» تولید کنیم.

وقتی با یک برنامه‌ی کیف‌پول کار می‌کنید، انواع مختلف پرجم SIGHASH را به عنوان گزینه‌ی قابل انتخاب نخواهید دید. با چند استثنای محدود، همه‌ی برنامه‌های کیف‌پول اسکریپت P2PKH تولید کرده و آنها را با پرجم ALL_SIGHASH امضا می‌کنند. برای استفاده از دیگر پرجم‌های SIGHASH باید خودتان دست به کار شوید و برنامه‌ای برای ایجاد و امضا کردن تراکنش‌ها بنویسید. از همه مهمتر این که، با استفاده از پرجم‌های SIGHASH می‌توان برنامه‌های بیت‌کوین بسیار متنوع و بدیع و با کاربردهای ویژه نوشت.

چگونگی محاسبه‌ی ECDSA

همان طور که قبلاً گفتیم، امضای دیجیتال با استفاده از تابع ریاضی F_{sig} ایجاد می‌شود که امضایی مشکل از دو مقدار R و S تولید می‌کند. در این قسمت به تشریح تابع F_{sig} می‌پردازیم.

الگوریتم امضای دیجیتال کار خود را با تولید یک جفت کلید خصوصی/عمومی لحظه‌ای (موقتی) شروع می‌کند. این جفت کلید موقتی، بعد از یک عملیات تبدیل شامل کلید خصوصی امضایکنده و ذرهم تراکنش، برای محاسبه‌ی مقادیر R و S به کار گرفته می‌شود. کلید خصوصی به کار رفته در این جفت کلید یک عدد تصادفی، k ، است. از این کلید خصوصی (با استفاده از فرمول $P = k \times G$ که در فصل ۳ دیدیم) کلید عمومی P استخراج می‌شود. مقدار R امضای دیجیتال مختصه‌ی x کلید عمومی لحظه‌ای P است. سپس، الگوریتم امضای دیجیتال مقدار S را با فرمول زیر از R استخراج می‌کند:

$$S = k^{-1} (Hash(m) + dA \times R) \bmod p$$

که در آن:

- k کلید خصوصی لحظه‌ای،
- R مختصه‌ی x کلید عمومی لحظه‌ای،
- dA کلید خصوصی امضایکنده،
- m داده‌ی تراکنش، و
- P مرتبه‌ی اول منحنی بیضوی است.

برای اعتبارسنجی نیز معکوس تابع تولید امضا، به همراه مقادیر R و S و آن کلید عمومی برای محاسبه‌ی یک مقدار P ، که نقطه‌ای روی منحنی بیضوی است (کلید عمومی لحظه‌ای مورد استفاده در تولید امضا)، به کار گرفته می‌شود:

$$P = (S^{-1} \times Hash(m) \times G) + (S^{-1} \times R \times Qa)$$

که در آن:

- R و S مؤلفه‌های امضا،
- Qa کلید عمومی آليس،
- m داده‌ی امضاشده‌ی تراکنش، و
- G نقطه‌ی مولد منحنی بیضوی است.

گراینده‌ی P به دست آمده از این فرمول برابر با R باشد، آنگاه می‌توان نتیجه گرفت که این امضای معتبر است. توجه شود که در فرآیند اعتبارسنجی امضایه هیچ وجه از کلید خصوصی استفاده نمی‌شود، بنابراین خطر لورفتن آن نیز وجود ندارد.

ریاضیات ECDSA پیچیده و دشوار است، اما منابع زیادی در اینترنت هست که می‌توانید به آنها مراجعه کنید. برای مثال، لینک <http://bit.ly/2r0HhGB> را توصیه می‌کنیم.



اهمیت تصادفی بودن در امضای دیجیتال

همان‌طور که در قسمت قبل دیدیم، الگوریتم تولید امضای دیجیتال از یک کلید تصادفی، k ، برای تولید جفت کلید خصوصی عمومی لحظه‌ای (موقتی) استفاده می‌کند. مادامی که این کلید تصادفی باشد، مقدار آن اهمیتی ندارد. اگر از یک مقدار k برای امضای دوپیام (تراکنش) مختلف استفاده شود، آنگاه هر کسی می‌تواند این کلید خصوصی امضای تراکنش را محاسبه کند. به این دیگر، استفاده‌ی مجدد از یک مقدار k برای چند امضای دیجیتال مختلف می‌تواند منجر به لورفتن آن کلید خصوصی شود.

اگر از یک مقدار واحد k برای امضای دو (یا چند) تراکنش مختلف استفاده کنید، آن کلید خصوصی می‌تواند محاسبه شده و فاش شود.



این فقط یک احتمال نظری نیست، و در چندین پیاده‌سازی مختلف از الگوریتم‌های امضای تراکنش‌های بیت‌کوین دیده شده است. به عبارت دیگر، افرادی پول خود را از دست داده‌اند، فقط به این خاطر که (ناخواسته) از یک مقدار k چند بار استفاده کرده‌اند. رایج‌ترین علت استفاده‌ی مجدد از یک مقدار k نامناسب بودن ماثول مولده‌ای اعداد تصادفی است. بهترین رویکرد برای اجتناب از این آسیب‌پذیری آن است که به جای مولده‌ای اعداد تصادفی مبتنی بر آنتروپی، از فرآیندهایی استفاده کنید که برای تولید عدد تصادفی از داده‌های خود تراکنش به عنوان پدر استفاده می‌کنند. این رویکرد تضمین می‌کند که برای هر تراکنش یک k متفاوت خواهد داشت. این روش که به آن فرآیند تصادفی قطعی (deterministic random process) گفته می‌شود، در استاندارد RFC 6979 (صفحه <https://tools.ietf.org/html/rfc6979>) تعریف شده است. اگر قصد دارید خودتان اقدام به پیاده‌سازی الگوریتم امضای تراکنش‌های بیت‌کوین کنید، باید برای اطمینان از تولید مقادیر مختلف k برای هر تراکنش از RFC 6979 (یا یک الگوریتم تصادفی قطعی مشابه) استفاده کنید.

آدرس بیت‌کوین، تراز حساب، و سایر موارد تجریدی

این فصل را با کشف این واقعیت آغاز کردیم که تراکنش‌های بیت‌کوین در حقیقت با آنچه کاربران در برنامه‌های کیف‌پول، کاوشگر بلکچین، و دیگر برنامه‌های کاربردی بیت‌کوین می‌بینند، بسیار متفاوت هستند. به نظر می‌رسد در ساختار تراکنش‌های بیت‌کوین هیچ خبری از بسیاری مفاهیم ساده و آشنا که در فصل‌های قبل دیدیم، مثل آدرس بیت‌کوین و تراز حساب، نیست. دیدیم که تراکنش‌های بیت‌کوین به خودی خود حاوی آدرس هیچ بیت‌کوینی نیستند، بلکه به جای آن بر اساس اسکریپت‌هایی که وظیفه‌ی آنها قفل کردن و باز کردن قفل مبالغ معین بیت‌کوین است، عمل می‌کنند. تراز حساب نیز در هیچ کجای سیستم بیت‌کوین وجود ندارد، ولی همه‌ی برنامه‌های کیف پول تراز حساب کاربران را به گونه‌ای واضح به نمایش می‌گذارند. اکنون که با تراکنش‌های بیت‌کوین آشنایی داشتیم، می‌توانیم بینیم این اطلاعات تجریدی چگونه از داده‌های خام یک تراکنش استخراج می‌شوند. اجازه دهید بار دیگر نگاهی به تراکنش آلیس در یک برنامه‌ی کاوشگر بلکچین بیندازیم؛ شکل ۶-۲ را بینید. در سمت چپ این تراکنش، کاوشگر بلکچین آدرس بیت‌کوین آلیس را به عنوان «فرستنده» نشان می‌دهد. اما اکنون می‌دانیم که در حقیقت این اطلاعات در خود تراکنش وجود ندارد. وقتی کاوشگر بلکچین این تراکنش را بازیابی می‌کند، تراکنش قبلی ارجاع شده در ورودی آن را نیز بازیابی کرده و اولین خروجی را از این تراکنش [قبلی] استخراج خواهد کرد. در این خروجی یک اسکریپت قفل کننده (اسکریپت P2PKH) وجود دارد که UTXO مربوطه را به درهم کلید عمومی آلیس قفل

Transaction View information about a bitcoin transaction

Summary		Inputs and Outputs	
Size	258 (bytes)	Total Input	0.1 BTC
Received Time	2013-12-27 23:03:05	Total Output	0.0995 BTC
Included In Blocks	277316 (2013-12-27 23:11:54 +9 minutes)	Fees	0.0005 BTC
		Estimated BTC Transacted	0.015 BTC

97 Confirmations 1/11/2013 21:19

شکل ۴-۶ تراکنش آليس برای پرداخت بهای یک فنجان قهوه به باب.

کرد، است. کاوشگر بلاکچین این دارم کلید عمومی را از آن اسکریپت استخراج کرده و با استفاده از فرمت Base58Check کُدگذاری می‌کند تا آدرس بیت کوین متناظر با این کلید عمومی را به دست آورده و نمایش دهد.

به همین ترتیب در سمت راست، کاوشگر بلاکچین دو خروجی نشان داده است؛ خروجی اول یک پرداخت به آدرس بیت کوین باب، و خروجی دوم یک پرداخت (به عنوان تسمه) به آدرس بیت کوین آليس است. کاوشگر بلاکچین این آدرس هارانیز از اسکریپت قفل کننده‌ی هر یک از این دو خروجی (که باز هم یک اسکریپت P2PKH است)، و بیرون کشیدن دارم کلید عمومی از آنها، استخراج می‌کند. سرانجام، کاوشگر بلاکچین هر یک از این دو دارم کلید عمومی را با فرمت Base58Check کُدگذاری می‌کند تا بتواند آدرس بیت کوین متناظر با این کلیدها عمومی را به دست آورده و نمایش دهد.

اگر روی آدرس بیت کوین باب کلیک کنید، کاوشگر بلاکچین تراز حساب آدرس بیت کوین باب را نشان خواهد داد؛ شکل ۴-۶ را ببینید. ولی باز هم می‌دانیم که در سیستم بیت کوین مفهومی به نام «تراز حساب» وجود ندارد. در واقع، این اطلاعات را خود کاوشگر بلاکچین از تراکنش‌ها استخراج کرده و به نمایش می‌گذارد. اما چگونه؟

برای محاسبه‌ی مقدار «دریافتی کل»، کاوشگر بلاکچین با کُدگشایی رشته‌ی Base58Check این آدرس بیت کوین، دارم ۱۶۰-بیتی کلید عمومی باب را (که در داخل آدرس بیت کوین وی گنجانده شده) بازیابی کرده، و سپس پایگاه داده‌ی تراکنش‌های بیت کوین را به دنبال خروجی‌هایی با اسکریپت قفل کننده‌ی P2PKH که حاوی دارم کلید عمومی باب باشند، جستجو می‌کند. با جمع کردن مبلغ این خروجی‌ها، کاوشگر بلاکچین دریافتی کل باب را محاسبه خواهد کرد.

Bitcoin Address Addresses are identifiers which you use to send bitcoins to another person.

Summary		Transactions	
Address	1GdK9UzphBzqzX2A9JFP3D4weBwmgmoQA	No. Transactions	25
Hash 160	ab68025513c3dbd2f7b92a94e058115d50f654a7	Total Received	0.17579525 BTC
Tools	Taint Analysis - Related Tags - Unspent Outputs	Final Balance	0.17579525 BTC

شکل ۴-۶ تراز حساب آدرس بیت کوین باب.

محاسبه تراز جاری (که در شکل ۸-۶ با عنوان «تراز نهایی» نشان داده شده [سطر آخر، سمت راست]) کاربیتری می‌برد. گفته‌یم که کاوشگر بلاکچین همواره یک پایگاه داده‌ی مستقل از خروجی‌هایی که ناکنون خرج نشده‌اند، موسوم به «مجموعه UTXO»، نگه می‌دارد. برای به روز نگه داشتن این پایگاه داده، کاوشگر بلاکچین باید به طور مرتب شبکه‌ی بیت‌کوین را پایش کند، و به صورت لحظه‌ای (با ورود تراکنش‌های تأیید شده) UTXO های تازه ایجاد شده را به آن اضافه کرده، و UTXO های خرج شده را حذف کند. این یک فرآیند پیچیده است که به پایش مداوم تراکنش‌های منتشر شده در شبکه‌ی بیت‌کوین، و همچنین حفظ اجماع با شبکه‌ی بیت‌کوین برای اطمینان از دنبال کردن زنجیره‌ی صحیح، نیاز دارد. [گاهی اوقات، کاوشگر بلاکچین همگامی خود با شبکه‌ی بیت‌کوین را از دست می‌دهد و در نتیجه «مجموعه UTXO» آن ناقص یا نادرست می‌شود.]

کاوشگر بلاکچین از این «مجموعه UTXO» تمام خروجی‌های خرج شده‌ی متاظر با درهم کلید عمومی باب را استخراج می‌کند و با جمع کردن مبلغ آنها، تراز نهایی وی را محاسبه کرده و به کاربر نمایش می‌دهد. برای نمایش این دو «تراز» و نمایش همین صفحه‌ی ساده‌ای که در شکل ۸-۶ می‌بینید، کاوشگر بلاکچین مجبور است دهها، صدها و یا حتی هزاران تراکنش را مرتب (اندیس‌گذاری) کرده و جستجو کند.

به طور خلاصه، اطلاعاتی که کاربران در برنامه‌هایی مانند کیف‌پول، کاوشگر بلاکچین، یا دیگر برنامه‌های کاربردی بیت‌کوین می‌بینند، اغلب توسط لایه‌های بالاتر و از طریق جستجو در تراکنش‌های مختلف، بررسی محتویات آنها، و پردازش داده‌هایی که در این تراکنش‌ها وجود دارد، استخراج می‌شود. برای تولید و نمایش همین نمای ساده از یک تراکنش بیت‌کوین که شبیه برنامه‌های مدیریت حساب بانکی است، برنامه‌های کاربردی بیت‌کوین باید مقدار زیادی پردازش انجام دهند که بخش اعظم جزئیات آن از دید کاربران پنهان می‌ماند. این برنامه‌ها اکثر روى رايچ ترین نوع تراکنش تمرکز می‌کنند: تراکنش‌های P2PKH با پرچم SIGHASH_ALL روی تمام ورودی‌ها. بنابراین، برنامه‌های کاربردی بیت‌کوین می‌توانند محتویات بیش از ۸۰ درصد تمامی تراکنش‌های را به گونه‌ای قابل دری برای کاربران به نمایش بگذارند؛ ولی گاهی هم به تراکنش‌هایی بر می‌خورند که عادی نیستند: تراکنش‌هایی که اسکریپت‌های قفل‌کننده‌ی پیچیده‌تر یا پرچم‌های SIGHASH متفاوتی دارند، یا تعداد ورودی‌ها و خروجی‌های آنها بسیار زیاد است. این قبیل تراکنش‌های سادگی و نقاط ضعف مازو کار استخراج اطلاعات در سیستم بیت‌کوین را آشکار می‌کنند.

هر روز صدها تراکنش که حاوی خروجی‌های P2PKH نیستند، در بلاکچین بیت‌کوین تأیید می‌شوند. برنامه‌ها و سرویس‌های کاوشگر بلاکچین اغلب این قبیل تراکنش‌های را با یک علامت هشدار قرمز به نمایش می‌گذارند تا بگویند که قادر به گذشتایی آدرس‌های بیت‌کوین موجود در آنها نیستند. برای دیدن جدیدترین «تراکنش‌های ناشناخته» که برنامه‌های کاوشگر بلاکچین قادر به گذشتایی کامل آنها نبوده‌اند، سری به صفحه <https://blockchain.info/strange-transactions> بزنید.

همان طور که در فصل بعد خواهید دید، همه‌ی این تراکنش‌ها لزوماً تراکنش ناشناخته نیستند؛ آنها فقط تراکنش‌هایی هستند که اسکریپت قفل‌کننده‌ی آنها از اسکریپت رایج P2PKH پیچیده‌تر است. در فصل آینده با این نوع تراکنش‌ها، و چگونگی گذشتایی آنها و برنامه‌هایی که برای این کار لازم دارید، آشنا خواهید شد.