

بلاک چین

مقدمه

ساختمان داده‌ی بلاک چین عبارت است از یک لیست مرتب (زنجیره) از بلاک‌هایی که هر یک به بلاک قبلی خود لینک می‌کنند. بلاک چین را می‌توان به صورت یک فایل تخت یا یک پایگاه داده‌ی ساده ذخیره کرد. مشتری هسته‌ی بتکوین فراداده‌های بلاک چین را با استفاده از پایگاه داده‌ی LevelDB گوگل ذخیره می‌کند. هر بلاک به بلاک قبل از خود در این زنجیره اشاره (لینک) می‌کند. بلاک چین را می‌توان به صورت یک پشته‌ی عمودی (مثل یک دسته کاغذ) نصور کرد که هر بلاک روی بلاک قبلی قرار می‌گیرد و اولین بلاک به عنوان زیربنای این پشته عمل می‌کند. اصطلاح «ارتفاع بلاک» از همین تصویر ریشه می‌گیرد و به عنوان فاصله‌ی هر بلاک با اولین بلاک تعریف می‌شود؛ بلاک «رأس» یا «نوک» نیز آخرین بلاک اضافه شده به پشته است.

در داخل بلاک چین، هر بلاک با یک دارم‌شناسایی می‌شود که حاصل‌الگوریتم دارم‌سازی رمزنگاری SHA256 روی سرآیند آن بلاک است. هر بلاک از طریق فیلد «دارم بلاک قبلی» در سرآیندش به بلاک قبل از خود، موسوم به بلاک مادر (والد)، ارجاع می‌کند. به بیان دیگر، هر بلاک حاوی دارم بلاک مادر خود در سرآیندش است. این دنباله‌ی دارم‌ها که هر بلاک را به بلاک مادر آن لینک می‌کند، یک زنجیره می‌سازد که تا اولین بلاک بلاک چین، موسوم به بلاک زرینده، عقب می‌رود. اگر چه هر بلاک فقط یک مادر دارد، ولی می‌تواند موقتاً چندین فرزند داشته باشد. همه‌ی این بلاک‌های فرزند در فیلد «دارم بلاک قبلی» خود به یک بلاک (مادر) اشاره می‌کنند. این وضعیت زمانی اتفاق می‌افتد که یک بلاک چین در حال «انشعاب» باشد؛ وقتی دو یا چند معدنچی تقریباً همزمان بلاک‌های متفاوتی کشف می‌کنند، انشعاب رُخ می‌دهد (فصل بعد را ببینید). با گذشت زمان، فقط یکی از این بلاک‌های فرزند در بلاک چین ثبت شده و «انشعاب» برطرف می‌شود. هر چند ممکن است یک بلاک بیش از یک فرزند داشته باشد، ولی هر بلاک یک (و فقط یک) مادر می‌تواند داشته باشد، چون هر بلاک فقط یک فیلد «دارم بلاک قبلی» دارد که به یک مادر واحد اشاره می‌کند.

فیلد «دارم بلاک قبلی» در داخل سرآیند بلاک قرار دارد و بنابراین روی دارم بلاک تأثیر می‌گذارد. اگر هویت یک بلاک مادر تغییر کند، هویت فرزند آن هم عوض خواهد شد. اگر یک بلاک مادر (به هر صورتی) تغییر کند، دارم SHA256 آن تغییر خواهد کرد. تغییر دارم بلاک مادر ایجاب می‌کند که فیلد «دارم بلاک قبلی» بلاک فرزند آن هم تغییر کند. با بروز این تغییر در بلاک فرزند، بلاکی که فرزند آن است (بلاک نوهی بلاک مادر اولیه)، نیز باید تغییر کند، والی آخر. این تغییر آبشاری به معنای آن است که با بروز تغییر در یک بلاک، تمام نسل‌های بعد از آن نیز باید دارم رمزنگاری SHA256 خود را

از نو محاسبه کرده و تغییر دهنده، از آنجاکه این محاسبات بسیار سنگین (و در نتیجه از نظر مصرف انرژی) بسیار پُر هزینه است، وجود یک زنجیره‌ی طولانی از بلاک‌ها تضمین می‌کند که تاریخچه‌ی بلاک چین تغییرناپذیر باشد؛ و این یک ویژگی کلیدی در امنیت بیت‌کوین است.

بلاک چین را می‌توان تا حدودی شبیه لایه‌های زمین‌شناسی یا یخچال‌های طبیعی دانست. لایه‌های سطحی می‌توانند با تغییر فصل دستخوش تغییر شوند، یا حتی (قبل از این که فرصت ثبت شدن پیدا کنند) از بین بروند، ولی همین که لایه‌های بعدی روی آنها را پوشانند (حتی به مقدار چند سانتی‌متر)، ثبات بیشتر و بیشتری خواهند یافت. وقتی عمق یک لایه‌ی زمین‌شناسی به چند صد متر برسد، چنان ثبت می‌شود که دیگر با گذشت میلیون‌ها سال هم تغییر نخواهد کرد. در بلاک چین هم بلاک‌های جدید می‌توانند به خاطر وجود پدیده‌ی انشعاب که به آن اشاره کردیم، دستخوش تغییر و بازنگری شوند. شش بلاک بالای بلاک چین در واقع مثل لایه‌ی سطحی خاک هستند، ولی همین که بلاک‌های بیشتری (بیشتر از شش بلاک) روی یک بلاک قرار بگیرند، احتمال تغییر در آن کمتر و کمتر خواهد شد. بعد از اضافه شدن ۱۰۰ بلاک ثبات آن به چنان حدی می‌رسد که می‌توان تراکنش پایگاه‌سکه آن [تراکنش حاوی بیت‌کوین‌هایی که به تازگی استخراج شده‌اند] را خرج کرد. وقتی چند هزار بلاک جدید اضافه شوند (فرآیندی که حدود یک ماه زمان می‌برد)، آن بخش از بلاک چین دیگر عملأً به تاریخ پیوسته است. هر چند پروتکل بیت‌کوین همچنان اجازه می‌دهد یک زنجیره به وسیله‌ی زنجیره‌ای طولانی تر شکسته شود، و هر چند احتمال نامعتبر شدن هر بلاکی وجود دارد، احتمال چنین رویدادی با گذشت زمان کاهش می‌یابد تا جایی که بی‌نهایت کوچک [یا عملأً غیرممکن] شود.

ساختار بلاک

بلاک یک ساختمان داده است که تراکنش‌ها را برای قرار گرفتن در دفتر کل (بلاک چین) مجتمع می‌کند. هر بلاک یک سرآیند دارد که محتوی مقداری فراداده (metadata) است و به دنبال آن فهرستی طولانی از تراکنش‌هایی که در این بلاک قرار دارند، می‌آید. سرآیند بلاک ۸۰ بایت طول دارد، در حالی که یک تراکنش متوسط حداقل ۲۵۰ بایت داشته، و یک بلاک به طور میانگین شامل ۵۰۰ تراکنش است. بنابراین، یک بلاک کامل (با تمامی تراکنش‌ها) ۱۰۰۰ بار بزرگتر از سرآیند آن است. ساختار بلاک را در جدول ۱-۹ مشاهده می‌کنید.

جدول ۱-۹ ساختار بلاک

فیلد	اندازه	توضیح
اندازه‌ی بلاک	۴ بایت	اندازه‌ی بلاک (بر حسب بایت)، بعد از این فیلد.
سرآیند بلاک	۸۰ بایت	چندین فیلد که سرآیند بلاک را تشکیل می‌دهند.
شمارنده‌ی تراکنش	۱ تا ۹ بایت (متغیر)	تعداد تراکنش‌هایی که در یک بلاک وجود دارد.
تراکنش‌ها	متغیر	تراکنش‌های ثبت شده در این بلاک.

سرآیند بلاک

سرآیند بلاک از سه مجموعه فراداده تشکیل می‌شود. اولین فراداده، فیلد ارجاع به ڈرهم بلاک قبلی است که این بلاک را در بلاک چین به بلاک قبل از آن متصل می‌کند. دومین مجموعه فراداده، شامل فیلدهای دشواری، برچسب زمانی و نوئس است که به رقابت معدنکاری (استخراج) بیت‌کوین مربوط می‌شوند (فصل ۱۰ را ببینید). سومین بخش از فراداده ریشه‌ی درخت مرکل است، یک ساختمان داده که تمامی تراکنش‌های موجود در یک بلاک را به گونه‌ای کارآمد خلاصه می‌کند، فیلدهای مختلف سرآیند بلاک را در جدول ۲-۹ می‌بینید.

فیلد	اندازه	توضیح
ویرایش	۴ بایت	عدد ویرایش نرم‌افزار/پروتکل (که در ارتقای سیستم کاربرد دارد).
درَهم بلاک قبلی	۳۲ بایت	ارجاع به دَرَهم بلاک قبلی (مادر) در بلاک چین.
ریشه‌ی مرکل	۳۲ بایت	دَرَهم ریشه‌ی درخت مرکل تراکنش‌های این بلاک.
برچسب زمانی	۴ بایت	زمان تقریبی ایجاد این بلاک (بر حسب ثانیه از اول ژانویه ۱۹۷۰).
هدف دشواری	۴ بایت	هدف دشواری برای الگوریتم اثبات-کار (PoW) این بلاک.
نونس	۴ بایت	شمارنده‌ای که در الگوریتم اثبات-کار استفاده می‌شود.

شناسه‌ی بلاک: دَرَهم سرآیند بلاک و ارتفاع بلاک

شناسه‌ی اصلی هر بلاک دَرَهم رمزگاری آن است، یک اثranگشت دیجیتال که با دوبار دَرَهم‌سازی سرآیند بلاک توسط الگوریتم SHA256 به دست می‌آید. به دَرَهم ۳۲-بایتی حاصل از این فرآیند دَرَهم بلاک (block hash) گفته می‌شود، در حالی که دَرَهم سرآیند بلاک درست‌تر است، چون برای تولید این دَرَهم فقط از سرآیند بلاک استفاده می‌شود. برای مثال، ۰00000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f دَرَهم بلاک اولین بلاک بیت‌کوین تاریخ است. دَرَهم بلاک شناسه‌ی یکتا و تردیدناپذیر آن است چون هر کسی می‌تواند با (دوبار) اعمال الگوریتم SHA256 روی سرآیند یک بلاک به آن برسد.

توجه کنید که دَرَهم بلاک واقعاً در ساختمان داده‌ی بلاک گنجانده نمی‌شود، هنگام مبادله‌ی بلاک روی شبکه ارسال نمی‌شود، و در بلاک چین هم ذخیره نمی‌شود. به جای آنها، گره‌ها با دریافت هر بلاک از شبکه دَرَهم آن را محاسبه می‌کنند. دَرَهم بلاک را می‌توان در یک جدول پایگاه داده‌ی جداگانه و به عنوان بخشی از فراداده‌ی آن بلاک (برای تسهیل در مرتب‌سازی و بازیابی سریع‌تر بلاک از دیسک) ذخیره کرد.

دومین روش شناسایی یک بلاک استفاده از مکان آن در بلاک چین، موسوم به ارتفاع بلاک (block height)، است. اولین بلاکی که ایجاد شد (همان بلاک ۰00000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f)، ارتفاع ۰ (صفر) داشت. بنابراین، یک بلاک را می‌توان به دوروش شناسایی کرد: با ارجاع به دَرَهم بلاک یا با ارجاع به ارتفاع بلاک. هر بلاکی که به بلاک چین اضافه می‌شود، ارتفاع آن یک واحد بیشتر از بلاک قبلی است. اولین بلاک بیت‌کوین (بلاک ۰) در ژانویه ۱۹۷۰ تولید شد، و ده سال بعد (اوایل ۲۰۱۹) ارتفاع بلاک به بیش از ۵۷۰,۰۰۰ رسید.

بر خلاف دَرَهم بلاک، ارتفاع بلاک یک شناسه‌ی یکتا نیست. اگر چه یک بلاک همواره ارتفاع مشخص و ثابتی دارد، عکس این موضوع درست نیست: یک ارتفاع مشخص همیشه یک بلاک واحد را مشخص نمی‌کند. ممکن است دو یا چند بلاک ارتفاع یکسانی داشته باشند و بر سر به دست آوردن این مکان (ارتفاع) مشخص در بلاک چین رقابت کنند. در فصل آینده (مبحث «انشعاب‌های بلاک چین») بیشتر در این باره صحبت خواهیم کرد. همچنین، ارتفاع بلاک بخشی از ساختمان داده‌ی یک بلاک نیست، و در آن بلاک ذخیره نمی‌شود. هر گره با دریافت یک بلاک از شبکه، مکان (ارتفاع) آن را در بلاک چین به صورت پویا شناسایی می‌کند. البته برای دسترسی و بازیابی سریع‌تر بلاک‌ها می‌توان ارتفاع آنها را به عنوان فراداده در یک جدول پایگاه داده ذخیره کرد.

درهم بلاک همیشه یک بلاک واحد را به طور یکتا مشخص می‌کند. همچنین، هر بلاک ارتفاع کاملاً مشخصی دارد. با این حال، عکس آن درست نیست، یعنی یک ارتفاع معین همیشه به یک بلاک واحد ارجاع نمی‌کند، چون ممکن است دو یا چند بلاک در حال رقابت بر سر به دست آوردن این مکان (ارتفاع) در بلاکچین باشند.



بلاک زاینده

اولین بلاک در بلاکچین بیتکوین که در ژانویه ۹ ۲۰۰۰ ایجاد شد، بلاک زاینده (genesis block) نامیده می‌شد. این بلاک جد تمامی بلاک‌های بلاکچین است، یعنی اگر از هر بلاک دلخواه شروع کنید و در زمان به عقب بروید، سرانجام به این بلاک خواهد رسید.

هر گره بیتکوین کار خود را با حداقل یک بلاک شروع می‌کند، چون بلاک زاینده به صورت ثابت در نرم‌افزار مشتری بیتکوین درج شده است و نمی‌توان آن را تغییر داد. به عبارت دیگر، هر گره بیتکوین درهم بلاک زاینده، زمان ایجاد آن، و حتی تنها تراکنش موجود در این بلاک را «می‌شناسد». بنابراین، همه‌ی گره‌های بیتکوین دارای یک «ریشه» وزیریتی امن و مستحکم برای تولید یک بلاک چین قابل اعتماد هستند. برای دیدن بلاک زاینده‌ی بلاکچین در گذشتی هسته‌ی بیتکوین به فایل `chainparams.cpp` [http://bit.ly/1x6rcwP] نگاه کنید. همان طور که قبل گفتیم، شناسه‌ی درهم متعلق به این بلاک (شناخته شده) می‌تواند بلاک زاینده‌ی بلاکچین را در سایت‌های کاویشگر بلاک جستجو کنید:

<https://blockchain.info/block/000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f>
یا

<https://blockexplorer.com/block/>

000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f

و یا از فرمان `getblock` مشتری هسته‌ی بیتکوین استفاده کنید تا اطلاعات زیر را به دست آورید:

```
$ bitcoin-cli getblock  
000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f
```

```
|  
"hash" : "000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f",  
"confirmations" : 308321,  
"size" : 285,  
"height" : 0,  
"version" : 1,  
"merkleroot" :  
    "4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b",  
"tx" : [  
    "4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b"  
],  
"time" : 1231006505,  
"nonce" : 2083236893,  
"bits" : "1d00ffff",
```

```

"difficulty" : 1.00000000,
"nextblockhash" :
"00000000839a8e6886ab5951d76f411475428afc90947ee320161bbf18eb6048"
]

```

در بلاک زاینده یک پیام مخفی وجود دارد. ورودی تراکنش پایگاه سکه‌ی این بلاک حاوی پیام متني «روزنامه تایمز، ۳ زانویه ۲۰۰۹؛ نخست وزیر در آستانه‌ی دومین استیضاح به دلیل بحران بانکی» است. این پیام علاوه بر اشاره به زمان ایجاد این بلاک (با ارجاع به تیتر روزنامه‌ی انگلیسی تایمز)، با کنایه به بحران پولی آن دوران که تمام دنیا را در بر گرفته بود، اهمیت یک سیستم پولی مستقل را یادآوری می‌کند. گنجاندن این پیام در اولین بلاک بیت‌کوین ایده‌ی ساتوشی ناکاموتو (حالق بیت‌کوین) بود.

اتصال بلاک‌های دارای بلاک چین

گره‌های کامل بیت‌کوین یک کپی کامل از بلاک چین نزد خود نگه می‌دارند، که از بلاک زاینده شروع می‌شود. با پیداشدن بلاک‌های جدید، بلاک چین محلی به طور پیوسته به روز-رسانی شده و زنجیره‌ی بلاک‌ها توسعه می‌یابد. با دریافت هر بلاک جدید از شبکه، گره کامل آن را اعتبارسنجی کرده و سپس این بلاک را به بلاک چین موجود متصل (لینک) می‌کند. برای برقراری این اتصال، گره دریافت کننده فیلد «درهم بلاک قبلی» در سرآیند بلاک وارد را بررسی می‌کند. برای مثال، فرض کنید یک گره دارای ۲۷۷,۳۱۴ بلاک در بلاک چین محلی خود است. آخرین بلاکی که این گره می‌شناسد، بلاک شماره‌ی ۲۷۷,۳۱۴# با سرآیند زیر است:

000000000000000027e7ba6fe7bad39faf3b5a83daed765f05f7d1b71a1632249

کمی بعد، این گره یک بلاک جدید با مشخصات زیر از شبکه دریافت می‌کند:

```

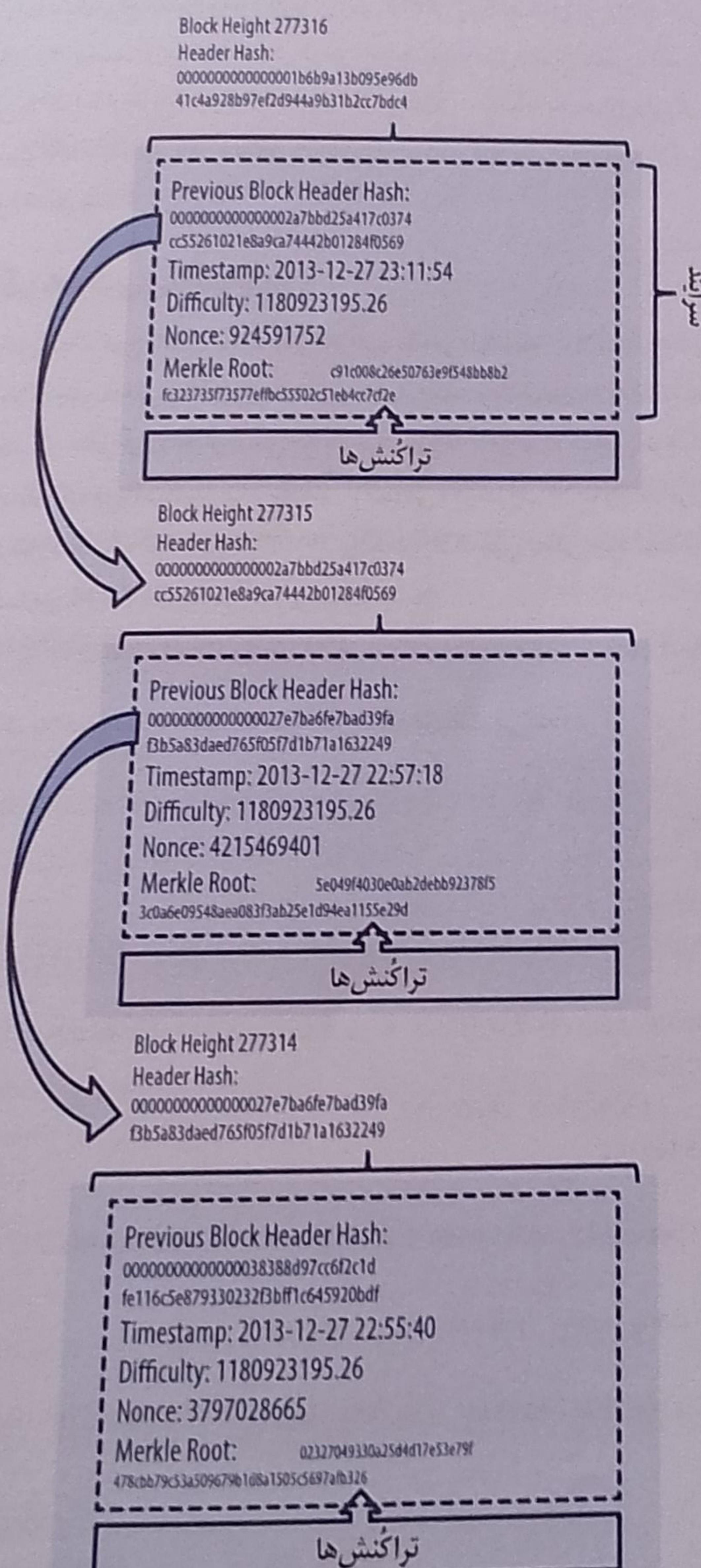
"size" : 43560,
"version" : 2,
"previousblockhash" :
"000000000000000027e7ba6fe7bad39faf3b5a83daed765f05f7d1b71a1632249",
"merkleroot" :
"5e049f4030e0ab2debb92378f53c0a6e09548aea083f3ab25e1d94ea1155e29d",
"time" : 1388185038,
"difficulty" : 1180923195.25802612,
"nonce" : 4215469401,
"tx" : [
"257e7497fb8bc68421eb2c7b699dbab234831600e7352f0d9e6522c7cf3f6c77",
#
# [... many more transactions omitted ...]
]

```

"05cf38f6ae6aa83674cc99e4d75a1458c165b7ab84725eda41d018a09176634"

با بررسی فیلد previousblockhash بلاک جدید که حاوی درهم بلاک مادر آن است، گره فرضی ما متوجه می‌شود که مادر این بلاک جدید همان آخرین بلاک دارای ارتفاع ۲۷۷,۳۱۴# است. بنابراین،

بلاک جدید فرزند آخرین بلاک این زنجیره است و بلاکچین موجود را توسعه می‌دهد. پس، گره ما بلاک جدید را به انتهای بلاکچین خود اضافه می‌کند و ارتفاع آن را به ۲۷۷,۳۱۵ می‌رساند. در شکل ۱-۹ سه بلاک این زنجیره را که از طریق فیلد previousblockhash به یکدیگر متصل شده‌اند، مشاهده می‌کنید.



شکل ۱-۹ سه بلاک که از طریق فیلد «درهم سرآیند بلاک قبلی» در یک زنجیره به یکدیگر متصل شده‌اند.

درخت مرکل

هر بلاک در بلاک‌چین بیت‌کوین حاوی خلاصه‌ای از تمامی تراکنش‌های آن بلاک به صورت یک درخت مرکل (merkle tree) است. درخت مرکل، که به آن درخت دَرْهَم باینری (binary hash tree) نیز گفته می‌شود، یک ساختمان داده است که برای خلاصه‌سازی و سنجش یکپارچگی مجموعه‌های عظیم داده به کار می‌رود. درخت مرکل یک درخت باینری حاوی دَرْهَم‌های رمزنگاری است. در علوم کامپیوتر از واژه‌ی «درخت» برای توصیف ساختمان‌های داده‌ی منشعب و شاخه‌دار استفاده می‌شود، ولی (همان طور که در مثال‌های این قسمت خواهد دید) این درخت‌ها معمولاً به صورت معکوس، «ریشه» در بالا و «شاخ و برگ» در پایین، نمایش داده می‌شوند.

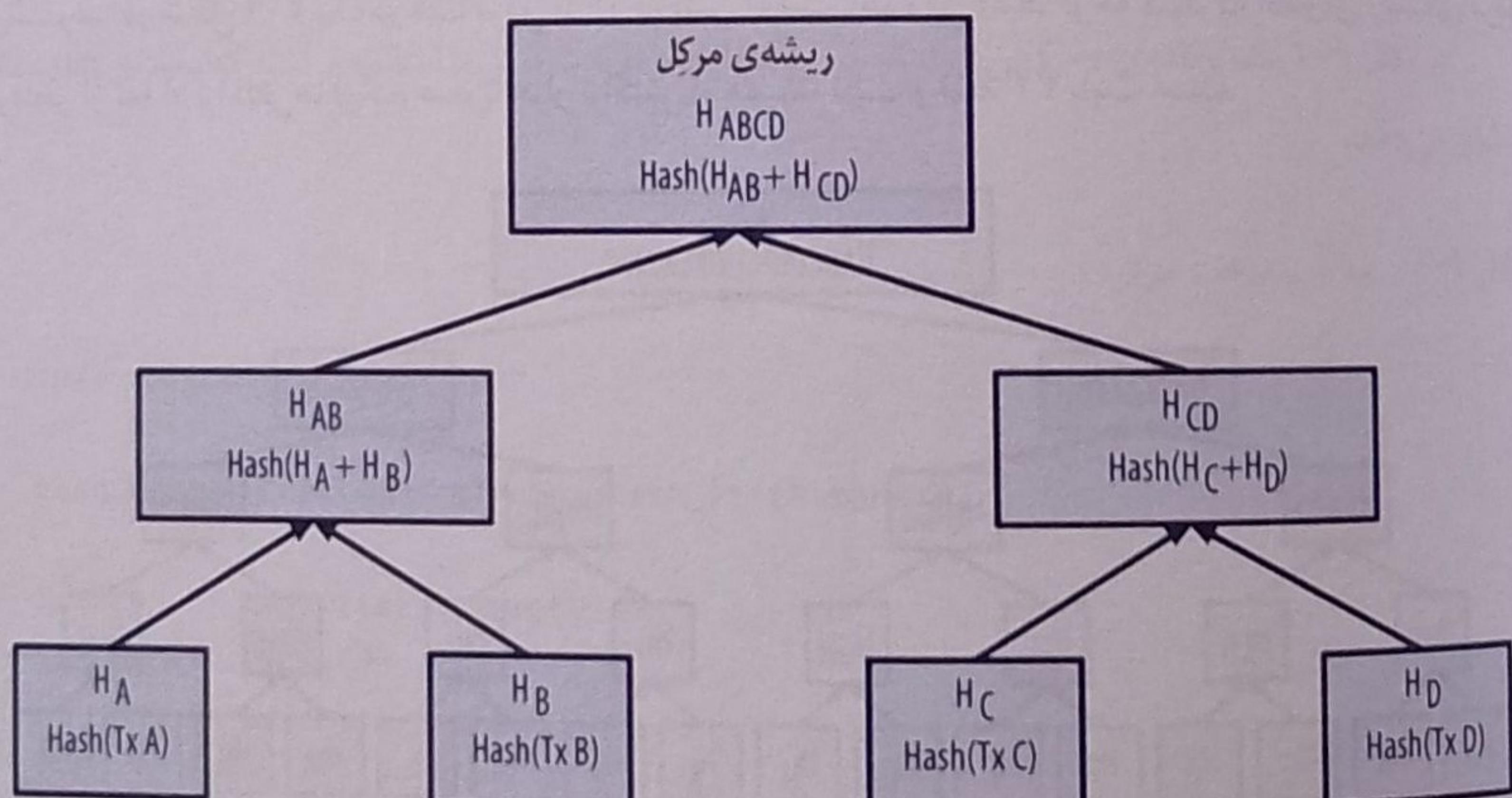
درخت مرکل با خلاصه‌سازی (فسرده کردن) تمام تراکنش‌های یک بلاک، یک اثranگشت تولید می‌کند، که با آن می‌توان وجود یک تراکنش خاص در این بلاک را به سرعت و با کارایی بسیار بالا تشخیص داد. یک درخت مرکل با دَرْهَم‌سازی چفت گره‌های به صورت بازگشته، تا جایی که در نهایت فقط یک گره موسوم به ریشه یا ریشه‌ی مرکل (merkle root) مرکل باقی بماند، ساخته می‌شود. الگوریتم رمزنگاری به کار رفته در تولید درخت مرکل همان SHA256-دوگانه (دوبار اعمال الگوریتم SHA256) است. برای مثال، اگر N عنصر داده (در اینجا، تراکنش) دَرْهَم‌سازی شده و در یک درخت مرکل فشرده شده باشند، با حداقل $2 \times \log_2(N)$ محاسبه می‌توان تشخیص داد که آیا یک عنصر داده (تراکنش) معین در این درخت وجود دارد یا خیر؛ همان طور که می‌بینید، این ساختمان داده بسیار کارآمد است.

درخت مرکل از پایین به بالا ساخته می‌شود. در مثال این قسمت با چهار تراکنش (A, B, C, D) که برگ‌های درخت مرکل را تشکیل می‌دهند، شروع می‌کنیم؛ شکل ۲-۹ را ببینید. البته خود این تراکنش‌ها در درخت مرکل ذخیره نمی‌شوند، بلکه با فرمول زیر دَرْهَم‌سازی شده و دَرْهَم‌های حاصل (H_A, H_B, H_C و H_D) در هر گره برگ (leaf node) ذخیره خواهند شد:

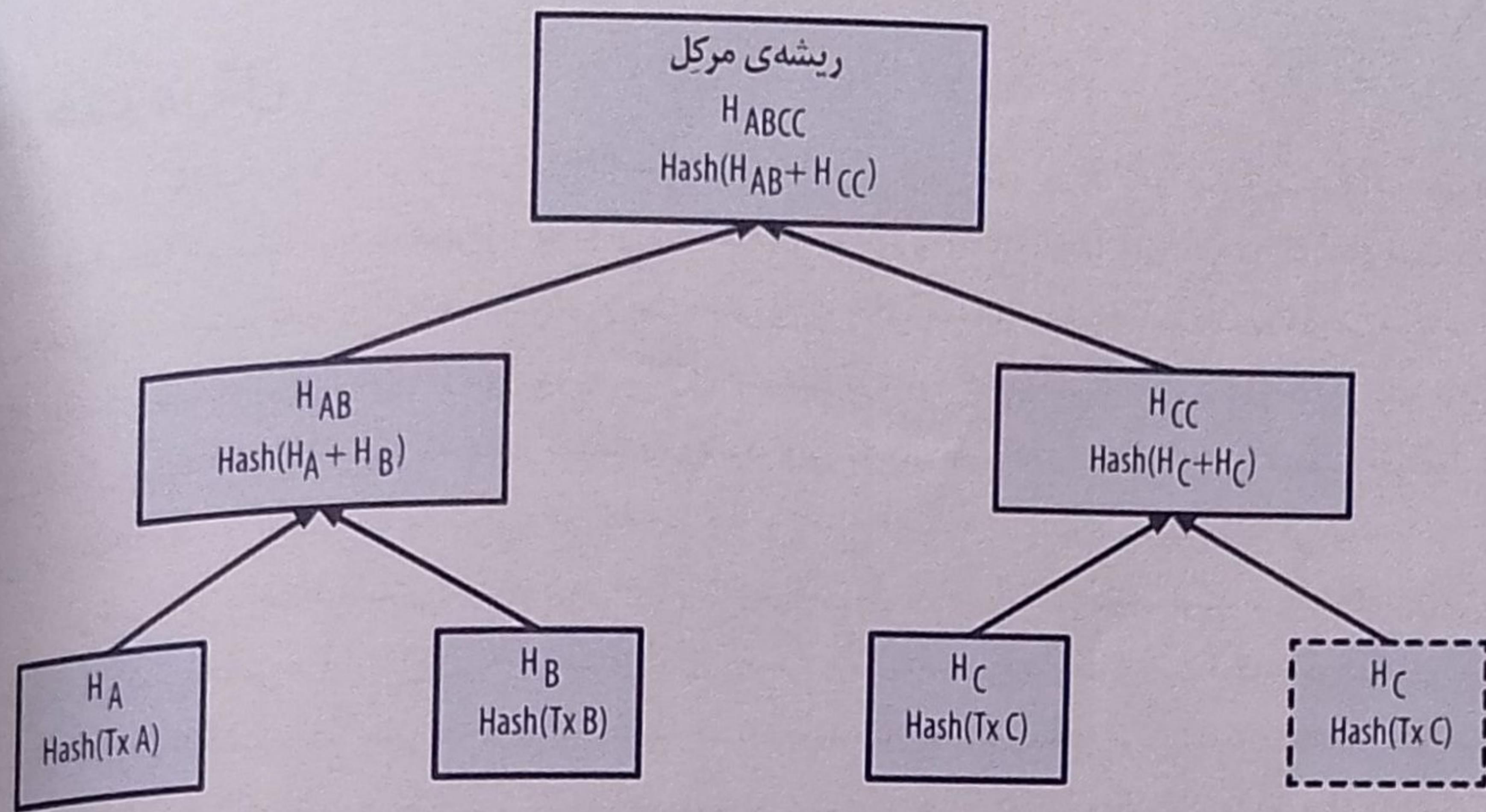
$$H_A = \text{SHA256}(\text{SHA256}(\text{Transaction A}))$$

سپس، چفت گره‌های متوالی به یکدیگر چسبانده شده، با همان فرمول بالا دَرْهَم‌سازی می‌شوند، و این دَرْهَم جدید به عنوان گره مادر در درخت مرکل ذخیره می‌شود. مثلاً، برای تولید گره مادر H_{AB} ، دو دَرْهَم ۳۲-بایتی گره‌های فرزند (یعنی H_A و H_B) به صورت یک رشته ۶۴-بایتی به یکدیگر چسبانده شده، و سپس الگوریتم SHA256-دوگانه روی این رشته اعمال می‌شود:

$$H_{AB} = \text{SHA256}(\text{SHA256} (H_A + H_B))$$



شکل ۲-۹ محاسبه گره‌ها در یک درخت مرکل.

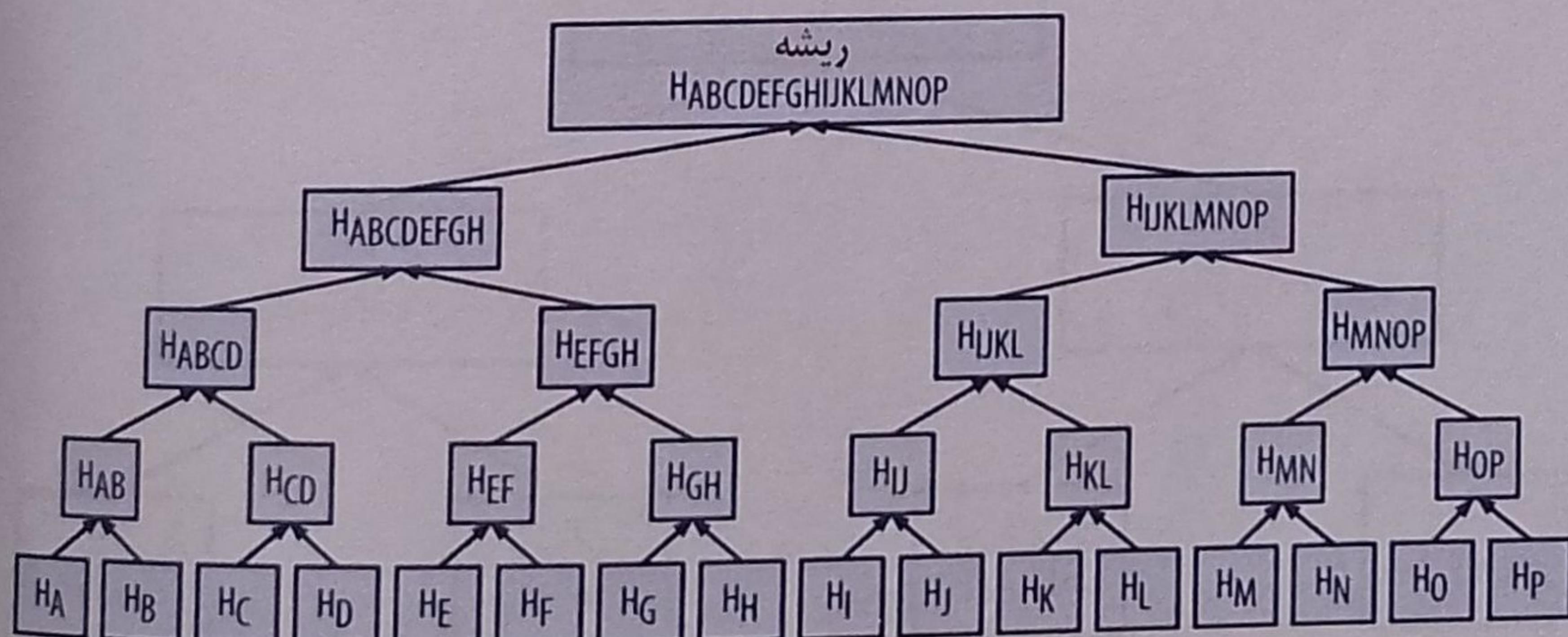


شکل ۳-۹ تکرار یک عنصر داده برای زوج کردن تعداد گره‌های درخت مرکل.

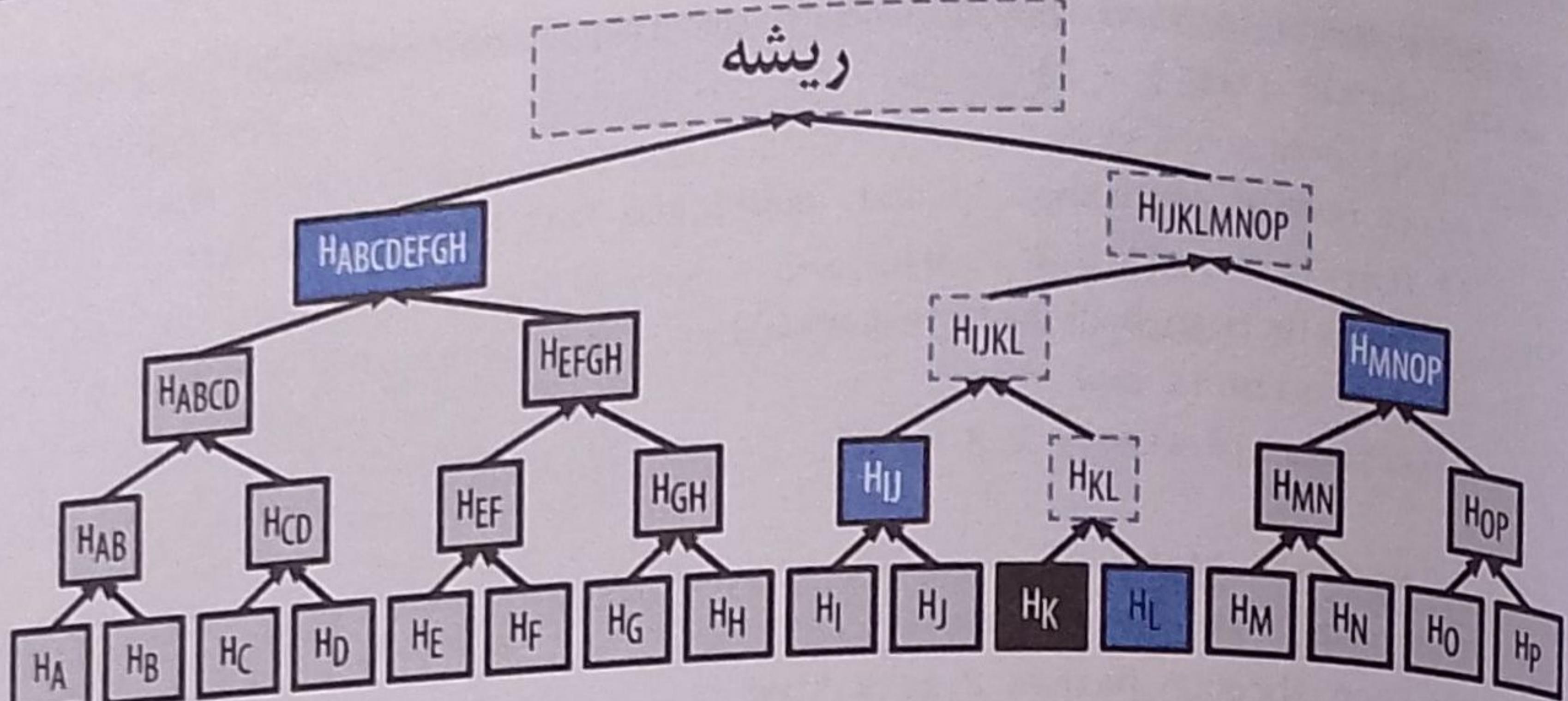
این فرآیند آنقدر تکرار می‌شود تا فقط یک گره، موسوم به ریشه‌ی مرکل، در بالای درخت باقی بماند؛ شکل ۲-۹ چگونگی محاسبه‌ی گره‌های مادر و گره ریشه‌ی مرکل را نشان داده است. این ۳۲-بایتی که فشرده‌ای از تمامی تراکنش‌های موجود در یک بلاک است، در سرآیند بلاک ذخیره می‌شود.

از آنجاکه درخت مرکل یک درخت باینری است، تعداد گره‌های برگ آن باید زوج باشد. به درختی که تعداد گره‌های برگ آن زوج است، درخت متعادل گفته می‌شود. اگر تعداد تراکنش‌های یک بلاک زوج نباشد، آخرین گره (درهم تراکنش) تکرار می‌شود تا تعداد گره‌های برگ زوج شده و درخت مرکل متعادل شود. این وضعیت را در شکل ۳-۹ که در آن تراکنش C تکرار شده است، مشاهده می‌کنید.

روش ساخت درختی با چهار تراکنش را می‌توان به هر تعداد تراکنش (گره برگ) تعمیم داد. بلاک‌های بیت‌کوین معمولاً صدها (و گاه بیش از هزار) تراکنش دارند، که به همین ترتیب به یک ریشه‌ی مرکل واحد ۳۲-بایتی تبدیل می‌شوند. در شکل ۴-۹ یک درخت مرکل با ۱۶ تراکنش را مشاهده می‌کنید. توجه کنید که اندازه‌ی ریشه‌ی مرکل، صرفنظر از تعداد تراکنش‌ها (یک، صد یا هزار تراکنش)، همیشه ثابت و فقط ۳۲ بایت است.



شکل ۴-۹ درخت مرکل حاصل از خلاصه کردن تعداد زیادی عنصر داده.



شکل ۵-۹ یک مسیر مرکل برای اثبات وجود یک عنصر داده (درهم تراکنش) در بلاک.

مهم‌ترین ویژگی درخت مرکل کارایی جستجوی آن است. همان‌طور که قبلاً گفتیم، برای تشخیص وجود یک تراکنش خاص در درخت مرکل، فقط به محاسبه $\log_2(N)$ درهم ۳۲-بایتی نیاز است: این درهم‌ها روی هم رفته یک سیر احراز هویت (authentication path) یا مسیر مرکل (merkle path) می‌سازند که ریشه‌ی درخت را به آن تراکنش خاص متصل می‌کند. این ویژگی به خصوص وقتی تعداد تراکنش‌ها افزایش می‌یابد، اهمیت زیادی دارد، چون آهنگ رشد تعداد محاسبه‌های مورد نیاز (که از مرتبه‌ی لگاریتم مبنای-۲ است) خیلی کمتر از آهنگ رشد تعداد تراکنش‌ها خواهد بود. به عبارت دیگر، گره‌های بیت‌کوین می‌توانند با محاسبه‌ی مسیرهایی که حداقل از ۱۰ تا ۱۲ درهم (۳۲۰ تا ۳۸۴ بایت) تشکیل می‌شوند، وجود (یا عدم وجود) یک تراکنش خاص در میان هزاران تراکنش یک بلاک (که اندازه‌ی آن به یک مگابایت می‌رسد) را اثبات کنند.

برای نمونه، در مثال شکل ۵-۹، برای اثبات وجود تراکنش K در این بلاک فقط به محاسبه‌ی یک مسیر مرکل با چهار درهم ۳۲-بایتی (در مجموع ۱۲۸ بایت) نیاز داریم. این مسیر از چهار درهم H_{ABCDEF} , $H_{IJKLMNOP}$, H_{IJKL} و H_K (کادرهای دارای زمینه‌ی تیره در شکل ۵-۹) تشکیل می‌شود. فقط با در اختیار داشتن همین چهار درهم، هر گرهی می‌تواند با محاسبه چهار درهم جفتی H_{IJKL} , H_{IJKL} , $H_{IJKLMNOP}$ و ریشه‌ی درخت مرکل (کادرهای دارای محیط خط‌چین در شکل ۵-۹) ثابت کند که درهم K در درخت مرکل وجود دارد (و در نتیجه، تراکنش K عضو این بلاک است).

در مثال ۱-۹ یک برنامه‌ی C++ می‌بینید که چگونگی ایجاد درخت مرکل با استفاده از توابع کتابخانه‌ی libbitcoin را نشان می‌دهد.

مثال ۱-۹ ساخت درخت مرکل

```
#include <bitcoin/bitcoin.hpp>

bc::hash_digest create_merkle(bc::hash_list& merkle)
{
    // Stop if hash list is empty.
    if (merkle.empty())
        return bc::null_hash;
    else if (merkle.size() == 1)
        return merkle[0];
    else
        return create_merkle(bc::hash_list{ merkle[0], create_merkle(merkle.begin() + 1, merkle.end()) });
}
```


در مثال ۲-۹ هم چگونگی کامپایل کردن و اجرای این برنامه را می‌بینید.

مثال ۲-۹ کامپایل و اجرای برنامه‌ی ساخت درخت مرکل

```
$ # Compile the merkle.cpp code
$ g++ -o merkle merkle.cpp $(pkg-config --cflags --libs libbitcoin)
$ # Run the merkle executable
$ ./merkle
Current merkle hash list:
32650049a0418e4380db0af81788635d8b65424d397170b8499cdc28c4d27006
30861db96905c8dc8b99398ca1cd5bd5b84ac3264a4e1b3e65afalbcee7540c4

Current merkle hash list:
d47780c084bad3830bcdaf6eace035e4c6cbf646d103795d22104fb105014ba3

Result: d47780c084bad3830bcdaf6eace035e4c6cbf646d103795d22104fb105014ba3
```

کارایی درخت مرکل با افزایش تعداد دَرَهم‌ها (تراکنش‌ها) آشکار می‌شود. جدول ۳-۹ مقدار داده‌ی مورد نیاز برای نمایش (ومبادله‌ی) یک مسیر مرکل [برای اثبات وجود یک تراکنش خاص در یک بلاک] را نشان می‌دهد.

جدول ۲-۹ کارایی درخت مرکل

تعداد تراکنش	اندازهی تقریبی بلاک	اندازهی مسیر (درهم)	اندازهی مسیر (بايت)
۱۶ تراکنش	۴ کیلو بايت	۴ درهم	۱۲۸ بايت
۵۱۲ تراکنش	۱۲۸ کیلو بايت	۹ درهم	۲۸۸ بايت
۲۰۴۸ تراکنش	۵۱۲ کیلو بايت	۱۱ درهم	۳۵۲ بايت
۶۵,۵۳۵ تراکنش	۱۶ مگابايت	۱۶ درهم	۵۱۲ بايت

همان طور که از این جدول می بینید، با آن که وقتی تعداد تراکنش های یک بلاک از ۱۶ به ۶۵,۵۳۵ می رسد، اندازهی آن از KB ۴ به ۱۶ MB افزایش می یابد، ولی رشد اندازهی مسیر مرکل (که برای اثبات وجود یک تراکنش خاص در یک بلاک لازم است) بسیار آهسته تر (از ۱۲۸ بایت به فقط ۵۱۲ بایت) است. به کمک درخت و مسیر مرکل، یک گره برای فهمیدن این تراکنش مورد نظر در یک بلاک معین وجود دارد یا خیر، نیازی به بارگیری کل آن بلاک ندارد و فقط کافی است یک مسیر مرکل کوچک را از سر آیند آن بازیابی کند؛ این ویژگی گره هارا از ذخیره سازی و مبادلهی کل بلاک چین روی شبکه (که اندازهی آن امروزه به چند صد گیگابایت بالغ می شود) بی نیاز می کند. همان طور که در فصل قبل اشاره کردیم، گره های موسوم به SPV (اعتبار منجی پرداخت ساده) با استفاده از همین مسیر های مرکل تراکنش هارا اعتبار منجی می کنند.

درخت مرکل و اعتبارسنجی پرداخت ساده (SPV)

مسیر مرکل کاربرد گسترده‌ای در گره‌های SPV دارد. یک گره SPV تمام تراکنش‌ها را در اختیار ندارد و حتی وقتی لازم باشد، فقط سرآیند بلاک‌ها (نه کل آنها) را از شبکه می‌گیرد. در واقع، گره‌های SPV برای اعتبارسنجی یک تراکنش از مسیر مرکل (یا مسیر احراز هویت) استفاده می‌کنند.

برای مثال، فرض کنید یک گره SPV می‌خواهد یک تراکنش پرداخت به یکی از آدرس‌هایی که در کیف‌پول وی وجود دارد، را اعتبارسنجی کند. این گره SPV ابتدا با اعمال یک فیلتر بلوم روی اتصال‌هایی که به گره‌های همتا دارد (فصل ۹ را ببینید)، تراکنش‌های دریافتی را به آنها بیان که حاوی آدرس‌های مورد نظر وی هستند، محدود می‌کند. وقتی یکی از گره‌های همایه به تراکنشی برخورد می‌کند که با این فیلتر بلوم منطبق است، آن بلاک را با استفاده از یک پیام merkleblock به گره در خواست‌کننده بر می‌گرداند. البته این پیام merkleblock فقط حاوی سرآیند آن بلاک و یک مسیر مرکل است که [درهم] تراکنش‌های این بلاک را به ریشه‌ی مرکل آن متصل می‌کند. به کمک این اطلاعات، گره SPV می‌تواند وجود (یا عدم وجود) تراکنش مورد نظر در داخل این بلاک را تشخیص دهد. گره SPV همچنین برای متصل کردن این بلاک به بقیه‌ی بلاک‌چین فقط به سرآیند آن نیاز دارد. وجود این دو اتصال (اتصال بین تراکنش و بلاک، و اتصال بین بلاک و بلاک‌چین) ثابت می‌کند که تراکنش مزبور واقعاً در بلاک‌چین ثبت و ضبط شده است. بدین ترتیب، گره SPV برای اعتبارسنجی یک تراکنش [به جای بلاک کامل (حدود یک مگابایت)] فقط به سرآیند بلاک و مسیر مرکل آن (که کمتر از یک کیلوبایت است) نیاز خواهد داشت.

بلاک‌چین‌های آزمایشی بیت‌کوین

تعجب می‌کنید اگر بدانید در بیت‌کوین بیش از یک بلاک‌چین وجود دارد. بلاک‌چین «اصلی» بیت‌کوین [همان بلاک‌چینی که در سوم ژانویه ۲۰۰۹ توسط ساتوشی ناکاموتو ایجاد شد و حاوی بلاک زاینده است] «شبکه‌ی اصلی»، mainnet، نام دارد. بیت‌کوین در حال حاضر سه بلاک‌چین دیگر به نام‌های «شبکه‌ی آزمایشی»، testnet، «شبکه‌ی تفکیکی»، segnet، و «آزمایش بازگشتی»، regtest، برای اهداف آزمایشی و توسعه‌ی برنامه‌های بیت‌کوین نیز دارد، که در این قسمت نگاهی به آنها می‌اندازیم.

میدان آزمایش بیت‌کوین: testnet

بیت‌کوین یک بلاک‌چین، شبکه و پول آزمایشی به نام testnet دارد که برای اهداف آزمایشی در نظر گرفته شده است. شبکه‌ی testnet یک شبکه‌ی P2P تمام و کمال (با کیف‌پول، سکه‌های آزمایشی، فرآیند معدنکاری و استخراج، و تمامی دیگر ویژگی‌های mainnet) است. در واقع، شبکه‌ی testnet فقط دو فرق با شبکه‌ی اصلی بیت‌کوین دارد: سکه‌های testnet ارزش مادی ندارند، و سطح دشواری فرآیند استخراج در آن به گونه‌ای است که استخراج سکه در این شبکه نسبتاً ساده باشد (بی‌ارزش بودن سکه‌های testnet هم به همین دلیل است).

هر برنامه و نرم‌افزاری که قرار است در شبکه‌ی mainnet کاربرد تولیدی داشته باشد، باید ابتدا در testnet و با سکه‌های testnet آزمایش شود. به این ترتیب سازندگان نرم‌افزار در مراحل توسعه‌ی برنامه و به خاطر بائی‌های آن (و همچنین مشکلات ناخواسته‌ای که ممکن است روی شبکه‌ی بیت‌کوین پیش آید) ضرر مالی نخواهند کرد.

با این حال، بی‌ارزش نگه داشتن سکه‌ها و ساده نگه داشتن استخراج آنقدر که در حرف به نظر می‌رسد، ساده نیست. با وجود درخواست‌های مکرر و دانمی برنامه‌نویسان، برخی افراد در شبکه‌ی testnet هم از تجهیزات پیشرفته (ASIC و GPU) برای استخراج سکه استفاده می‌کنند. این کار سطح دشواری فرآیند استخراج را افزایش می‌دهد و استخراج سکه‌ی testnet با یک CPU را تقریباً غیرممکن می‌کند، در نتیجه استخراج سکه‌های testnet چنان دشوار

می‌شود که افراد شروع می‌کنند به ارزش قائل شدن برای آنها؛ به عبارت دیگر، سکه‌ای که باید بی ارزش باشد، دیگر بی ارزش نلفی نمی‌شود. به همین دلیل، لازم است هر از جندهای گاهی شبکه‌ی testnet به کلی پاکسازی (ریست) شده و دوباره از یک بلاک زاینده‌ی جدید (و دشواری پایین) شروع به کار کند. این اتفاق تاکنون سه بار افتاده است: جدیدترین شبکه‌ی testnet که در فوریه ۲۰۱۱ ریست شده، testnet3 نام دارد.

بلاکچین3 testnet بسیار بزرگ است (۱۴ گیگابایت در اوایل ۲۰۱۹)، همگام‌سازی کامل آن بیش از یک روز زمان می‌برد، و بخش زیادی از منابع کامپیوتر را می‌بلعد. البته این بلاکچین به بزرگی بلاکچین اصلی بیت‌کوین بنت [برای مقایسه، اندازه‌ی بلاکچین اصلی بیت‌کوین در فصل اول ۲۰۱۹ بیش از ۲۱ گیگابایت بوده است]. ولی «بک‌وزن» هم محسوب نمی‌شود! بهترین روش پیاده‌سازی گره testnet استفاده از یک ماشین مجازی (مثل Cloud Server، VirtualBox، VMware)

کار با testnet

همه‌ی بیت‌کوین (مثل تقریباً تمامی نرم‌افزارهای دیگر بیت‌کوین) به طور کامل از testnet پشتیبانی می‌کند؛ به عبارت دیگر، می‌توانید یک گره کامل testnet راه بیندازید و سکه‌ی testnet استخراج کنید. برای اجرای هسته‌ی بیت‌کوین با testnet (به جای mainnet) فقط کافی است از پرچم testnet استفاده کنید:

```
$ bitcoind -testnet
```

با اجرای این فرمان، هسته‌ی بیت‌کوین (bitcoind) یک بلاکچین جدید در زیردايركتوري3 testnet (در دايركتوري پيش‌فرض بیت‌کوین) می‌سازد:

```
bitcoind: Using data directory /home/username/.bitcoin/testnet3
```

برای ارتباط با این بلاکچین نیز می‌توانید از فرمان‌های مشتری هسته‌ی بیت‌کوین (bitcoin-cli) که قبل دیده‌اید (البته با همان پرچم testnet)، استفاده کنید:

```
$ bitcoin-cli -testnet getinfo
```

```
|
```

```
"version": 130200,
"protocolversion": 70015,
>walletversion": 130000,
"balance": 0.00000000,
"blocks": 416,
"timeoffset": 0,
"connections": 3,
"proxy": "",
"difficulty": 1,
"testnet": true,
"keypoololdest": 1484801486,
"keypoolsize": 100,
"paytxfee": 0.00000000,
"relayfee": 0.00001000,
"errors": ""
```

```
|
```

با فرمان `getblockchaininfo` نیز می‌توانید اطلاعاتی دربارهٔ بلاک‌چین testnet³ و فرآیند همگام‌سازی به دست آورید:

```
$ bitcoin-cli -testnet getblockchaininfo
```

اگر به تجربه کردن با چارچوب‌ها و زبان‌های برنامه‌نویسی دیگر علاقمند هستید، می‌توانید شبکه‌ی testnet3 را روی دیگر پیاده‌سازی‌های گره-کامل بیت‌کوین [مانند `btc` (که به زبان Go نوشته شده) و `bcoing` (که با جاوا‌اسکریپت نوشته شده)] نیز به خوبی اجرا کنید. نکته‌ی جالب این که `testnet3`، علاوه بر پشتیبانی از تمامی ویژگی‌های mainnet، از «شاهد تفکیک شده» نیز پشتیبانی می‌کند، در حالی که این ویژگی هنوز به طور کامل روی شبکه‌ی اصلی بیت‌کوین فعال نشده است.

شبکه‌ی آزمایش «شاهد تفکیک شده»: segnet

در سال ۲۰۱۶، یک شبکه‌ی آزمایشی تک-منظوره برای آزمایش شاهد تفکیک شده (Segregated Witness)، یا `segwit`، راه‌اندازی شد. این بلاک‌چین آزمایشی `segnet` نام دارد و با اجرای ویرایش (شاخه) خاصی از هسته‌ی بیت‌کوین می‌توان به آن ملحق شد.

از آنجا که segwit اکنون به testnet³ اضافه شده است، برای آزمایش «شاهد تفکیک شده» دیگر نیازی به segnet ندارید. شاید در آینده شاهد ظهور دیگر شبکه‌های آزمایشی تک-منظوره باشیم که برای آزمایش یک ویژگی خاص با تغییر اساسی در معماری بیت‌کوین (مانند segnet) طراحی شده‌اند.

regtest: بلاک چین خصوصی

شبکه‌ی regtest، آزمایش بازگشتی (Regression Testing)، یک ویژگی هسته‌ی بیت‌کوین است که به شما اجازه می‌دهد یک بلاک چین خصوصی برای مقاصد آزمایشی ایجاد کنید. برخلاف testnet³، که یک بلاک چین آزمایشی عمومی است و همگان به آن دسترسی دارند، بلاک چین regtest برای اجرا روی یک سیستم بسته و آزمایش‌های محلی در نظر گرفته شده است. وقتی یک بلاک چین regtest راه‌اندازی می‌کنید، بلاک زاینده‌ی آن را هم خودتان به وجود می‌آورید. این بلاک چین که ععمولاً فقط با یک گره اجرامی شود، ولی می‌توان گره‌های دیگری را نیز به شبکه‌ی آن اضافه کرد، برای آزمایش خصوصی نرم‌افزارهای بیت‌کوین بسیار مناسب است.

برای اجرای هسته‌ی بیت‌کوین در حالت regtest فقط کافی است از پرچم regtest استفاده کنید:

```
s bitcoind -regtest
```

با اجرای این فرمان، هسته‌ی بیت‌کوین (bitcoind) یک بلاک‌چین جدید در زیردايركتوري regtest (در دايركتوري لرض بیت‌کوین) ايجاد می‌کند:

bitcoind: Using data directory /home/username/.bitcoin/regtest

برای ارتباط با این بلاک چین نیز می‌توانید از فرمان‌های مشتری هسته‌ی بیت‌کوین (bitcoinc11) که قبلاً دیده‌اید اهمان بر جم regtest)، استفاده کنید:

```
$ bitcoin-cli -regtest getblockchaininfo
```

[...]

همان طور که می بینید، در این بلاک چین هنوز هیچ بلاکی وجود ندارد. اجازه دهید چند بلاک (مثلاً ۵۰۰ بلاک) حظر است؟) استخراج کرده و جایزه‌ی آنها را تصاحب کنیم:

```
$ bitcoin-cli -regtest generate 500
```

1

```
"7afed70259f22c2bf11e406cb12ed5c0657b6e16a6477a9f8b28e2046b5balca",
"1aca2f154a80a9863a9aac4c72047a6d3f385c4eec5441a4aaafa6acaaldada14",
"4334ecf6fb022f30fbe764c3ee778fabbd53b4a4d1950eae8a91f1f5158ed2d1",
"5f951d34065efeaf64e54e91d00b260294fcdfc7f05dbb5599aec84b957a7766",
"43744b5e77c1dfce9d05ab5f0e6796ebe627303163547e69e27f55d0f2b9353",
[...]
"6c31585a48d4fc2b3fd25521f4515b18aefb59d0def82bd9c2185c4ecb754327"
```

استخراج این ۵۰ بلاک فقط چند ثانیه طول می کشد، که مسلم است که آزمایش بسیار خوب است. اگر تراز کنف پیول [آزمایشی] خود را بررسی کنید، متوجه خواهید شد که جایزه استخراج ۴۰ بلاک اول به حساب شما واریز شده است (جوایز پایگاه سکه باستی حداقل ۱۰۰ بلاک عمق داشته باشند تا بتوانید آنها را خرج کنید):

```
$ bitcoin-cli -regtest genbalance  
12462.50000000
```

استفاده از بلاکچین‌های آزمایشی برای توسعهٔ نرم‌افزار

بلاکچین‌های مختلف بیت‌کوین (regtest، segnet، testnet، mainnet) طیف گسترده‌ای از محیط‌های آزمایشی برای توسعهٔ نرم‌افزارهای بیت‌کوین فراهم می‌کنند. اگر در حال توسعهٔ نرم‌افزار برای هستهٔ بیت‌کوین یا یک مشتری گره-کامل دیگر هستید، یا می‌خواهید برای بیت‌کوین برنامه‌های کاربردی (کیف‌پول، صرافی، سایت تجارت-الکترونیک، یا حتی قراردادهای هوشمند و اسکرپت‌های پیچیده) بنویسید، از این بلاکچین‌های آزمایشی استفاده کنید.

بلاکچین‌های آزمایشی بیت‌کوین ابزاری مناسب برای زدودن هر گونه باگ از برنامه قبل از ارنهی آن به بازار واقعی هستند. برنامه‌های خود را ابتدا روی یک بلاکچین regtest توسعه دهید. همین که برای عرضهٔ عمومی برنامه آماده شدید، آن را به testnet که محیطی متعدد و پویاتر است، منتقل کنید. سرانجام، وقتی از همهٔ جواب برنامه مطمئن شدید، آن را روی mainnet (شبکهٔ اصلی بیت‌کوین) منتشر کنید. برای هر تغییر مهم و اساسی که در برنامهٔ خود می‌دهید، همین مراحل (از regtest به testnet، و از آنجا به mainnet) را یک بار دیگر تکرار کنید.