

کلید و آدرس

شاید شنیده باشد که بیت‌کوین ارزی مبتنی بر رمزنگاری (cryptography) است، شاخه‌ای از ریاضیات که کاربرد گسترده‌ای در امنیت کامپیوتری دارد. رمزنگاری از واژه‌ی یونانی «مخفي نويسي» ریشه گرفته، ولی امروزه علم رمزنگاری محدوده‌ای بسیار وسیع‌تر از مخفی نویسی (که به آن رمزگذاری گفته می‌شود) دارد. رمزنگاری همچنین می‌تواند برای اثبات دانستن یک راز بدون آشکار کردن آن راز (امضای دیجیتال)، یا اثبات اصالت یک پیغام (اثر انگشت دیجیتال) به کار رود. این نوع از آزمون‌های رمزنگاری نقشی کلیدی در بیت‌کوین دارند و مهمترین ابزارهای ریاضی آن به شمار می‌روند. طرفه اینجاست که خود فرآیند رمزگذاری بخشی کلیدی از بیت‌کوین نیست، چون مبادلات و تراکنش‌های بیت‌کوین رمزگذاری نمی‌شوند و برای حفاظت از دارایی افراد نیازی به رمزگذاری آنها نیست. در این فصل به معرفی اجمالی آن دسته از مفاهیم رمزنگاری که در بیت‌کوین برای کنترل مالکیت دارایی‌ها به کار گرفته شده‌اند (مثل کلید، آدرس و کیف‌پول)، می‌پردازیم.

مقدمه

در بیت‌کوین، مالکیت یک دارایی از طریق کلید دیجیتال (digital key)، آدرس بیت‌کوین (bitcoin address) و امضا دیجیتال (digital signature) به اثبات رسیده و محقق می‌شود. کلیدهای دیجیتال در شبکه‌ی بیت‌کوین نگهداری نمی‌شوند، بلکه خود کاربران مستنول ایجاد و نگهداری آنها در یک فایل، یا پایگاه داده‌ی ساده، موسوم به کیف‌پول (wallet) هستند. امضاهای دیجیتال موجود در کیف‌پول کاربر به کلی مستقل از پروتکل بیت‌کوین هستند و نرم‌افزار کیف‌پول کاربر می‌تواند آنها را بدون ارجاع به بلاک‌چین یا حتی بدون دسترسی به اینترنت ایجاد کرده و نگهداری (مدیریت) کند. بسیاری از ویژگی‌های جالب بیت‌کوین، از جمله اعتماد و کنترل غیرمت مرکز، گواهی مالکیت، و مدل امنیتی اثبات-رمزنگاری، ناشی از همین خصوصیت کلیدهای دیجیتال است.

در اکثر تراکنش‌های بیت‌کوین لازم است یک امضا دیجیتال معتبر به بلاک‌چین اضافه شود، امضا کی که فقط به کمک یک کلید سری (secret key) می‌توان آن را تولید کرد؛ بنابراین، هر کسی که این کلید را داشته باشد، می‌تواند آن بیت‌کوین را کنترل کند. به امضا دیجیتال به کار رفته برای خرج کردن بیت‌کوین شاهد (witness)، که یکی از اصطلاحات رمزنگاری است، نیز گفته می‌شود. امضا دیجیتال (شاهد) گنجانده شده در یک تراکنش بیت‌کوین مالکیت فرد بر آن مبلغ را تصدیق کرده و به وی اجازه‌ی خرج کردن آن را می‌دهد.

کلیدهای صورت جفت تولید می‌شوند: یک کلید خصوصی (private key) [یا سرّی] و یک کلید عمومی (public key). کلید عمومی را می‌توان معادل شماره‌ی یک حساب بانکی و کلید خصوصی را معادل رمز یا امضای صاحب آن حساب (که با آن می‌تواند حساب را کنترل کند) دانست. کاربران بیت‌کوین به ندرت با این کلیدهای دیجیتال سروکار دارند، چون آنها همیشه در داخل فایل کیف‌پول ذخیره می‌شوند و مدیریت آنها بر عهده‌ی نرم‌افزار کیف‌پول است.

در بخش پرداخت یک تراکنش بیت‌کوین، فرد گیرنده کلید عمومی (یا اثر انگشت دیجیتال) خود را که به آدرس بیت‌کوین معروف است، ارائه می‌کند. این کلید در واقع معادل همان شماره حساب گیرنده است (یا نام واقعی گیرنده در زمانی که پرداخت با چک انجام می‌شود). در اکثر موارد، آدرس بیت‌کوین از یک کلید عمومی تولید می‌شود و متاظر با آن است. با این حال، همه‌ی آدرس‌های بیت‌کوین معادل یک کلید عمومی نیستند؛ انواع دیگری از گیرنده نیز وجود دارند (مانند اسکریپت) که در ادامه‌ی همین فصل به آنها خواهیم پرداخت. با این روش، آدرس بیت‌کوین جای گیرنده را می‌گیرد و انعطاف‌پذیری تراکنش‌هارا افزایش می‌دهد، درست مثل پرداخت با چک که می‌توان آن را در وجه یک فرد حقیقی یا حقوقی، یا در وجه حامل (برای نقد کردن) صادر کرد. آدرس بیت‌کوین تنها نمای بیرونی کلیدهای عمومی است که کاربران به طور معمول با آن سروکار دارند، چون این تنها چیزی است که باید برای مبادلات بیت‌کوین در اختیار دیگران قرار دهد.

در ابتدا، اصول رمزنگاری به کار رفته در بیت‌کوین و مبانی ریاضی آن را را به اختصار تشریح می‌کنیم، سپس نگاهی به چگونگی تولید، ذخیره‌سازی و مدیریت کلیدها می‌اندازیم، و فرمتهای مختلف نمایش کلیدهای عمومی و خصوصی، آدرس‌های بیت‌کوین و آدرس‌های اسکریپت را بررسی می‌کنیم. در آخر، نگاهی می‌اندازیم به کاربردهای پیشرفته‌ی کلید و آدرس: پوچی، چندامضایی، آدرس اسکریپت، و کیف‌پول کاغذی.

رمزنگاری کلید عمومی و ارز رمزبنیان

رمزنگاری کلید عمومی که در دهه ۱۹۷۰ اختراع شد، مبنای ریاضی امنیت کامپیوتر و اطلاعات را تشکیل می‌دهد. بعد از ابداع رمزنگاری کلید عمومی، چندین تابع ریاضی مناسب برای این منظور کشف شده است، مثل توان رسانی اعداد اول و ضرب منحنی بیضوی. این توابع ریاضی عملاً معکوس ناپذیر هستند، یعنی محاسبه‌ی آنها در یک جهت ساده و در جهت مخالف غیرممکن است. رمزنگاری بر اساس این توابع ریاضی اجازه می‌دهد کلیدهای امضاهای دیجیتالی تولید کنیم که جعل کردن آنها غیرممکن باشد. بیت‌کوین از ضرب منحنی بیضوی به عنوان مبتنای برای رمزنگاری خود استفاده می‌کند. در سیستم بیت‌کوین از رمزنگاری کلید عمومی برای تولید جفت کلیدهایی استفاده می‌شود که دسترسی به بیت‌کوین‌ها را کنترل می‌کنند. این جفت کلید از یک کلید خصوصی و یک کلید عمومی یکتا (که از آن کلید خصوصی مشتق شده) تشکیل می‌شود. کلید عمومی برای دریافت بیت‌کوین به کار می‌رود، و با کلید خصوصی می‌توان تراکنش‌های خرج کردن (مصرف) بیت‌کوین را امضای کرد.

بین این کلید عمومی و خصوصی یک رابطه‌ی ریاضی وجود دارد که اجازه می‌دهد از کلید خصوصی برای تولید امضای دیجیتال پیغام‌ها استفاده کنیم. سپس این امضای دیجیتال برای اعتبارسنجی (راست‌آزمایی) آن کلید عمومی به کار گرفته می‌شود، بدون این که نیازی به فاش کردن کلید خصوصی یا سپردن آن به دست دیگران باشد.

هنگام خرج کردن یک مقدار بیت‌کوین، مالک فعلی آن کلید عمومی و امضای دیجیتال تولید شده از کلید خصوصی خود را (که هر بار متفاوت است ولی از همان کلید خصوصی ساخته می‌شود) در تراکنش خرج کردن بیت‌کوین قرار می‌دهد. با ارائه‌ی این کلید عمومی و امضای دیجیتال متاظر با آن، هر کسی در شبکه‌ی بیت‌کوین می‌تواند این تراکنش را اعتبارسنجی کرده (تأثید کند که فرد ارسال‌کننده‌ی بیت‌کوین همان مالک آن است) و آن را پذیرد.



شکل ۴-۴ کلید خصوصی، کلید عمومی، و آدرس بیت کوین.

در اکثر برنامه های کیف پول، برای راحتی کار، کلید های عمومی و خصوصی به صورت جفت کلید ذخیره می شوند. ولی این اجباری نیست، چون کلید عمومی را همیشه می توان از کلید خصوصی محاسبه کرد؛ به عبارت دیگر، کیف پول می تواند فقط کلید خصوصی را ذخیره کند و هر بار کلید عمومی را از روی آن بسازد.



کلید های خصوصی و عمومی

یک کیف پول بیت کوین مجموعه ای است از تعدادی جفت کلید، که هر جفت از یک کلید خصوصی و یک کلید عمومی تشکیل شده است. کلید خصوصی (k) یک عدد است، عددی که معمولاً به صورت تصادفی انتخاب می شود. با استفاده از ضرب منحنی بیضوی (elliptic curve multiplication)، که یک تابع رمز نگاری یک-طرفه است، از این کلید خصوصی یک کلید عمومی (K) تولید می شود. سپس یک تابع درهم ساز (hash function)، که آن هم یک تابع رمز نگاری یک-طرفه است، از این کلید عمومی یک آدرس بیت کوین (A) می سازد. در این قسمت کار خود را بانمایش چگونگی تولید کلید خصوصی شروع می کنیم، مبانی ریاضی منحنی های بیضوی (برای تبدیل این کلید خصوصی به یک کلید عمومی) را توضیح می دهیم، و در آخر چگونگی تولید آدرس بیت کوین از این کلید عمومی را تشریح می کنیم. رابطه ای بین کلید خصوصی، کلید عمومی، و آدرس بیت کوین در شکل ۴-۱ نشان داده شده است.

چرا رمز نگاری نامتقارن (کلید عمومی / خصوصی)؟

چرا در بیت کوین از رمز نگاری نامتقارن استفاده می شود؟ در بیت کوین، رمز نگاری برای «مخفي کردن» (رمزنگاری) تراکنش ها به کار نمی رود، بلکه از یک خصوصیت مفید دیگر رمز نگاری نامتقارن، یعنی توانایی آن برای ایجاد امضای دیجیتال، استفاده شده است. به کمک یک کلید خصوصی می توان از اثر انگشت دیجیتال تراکنش ها برای تولید به امضای دیجیتال (عددی) استفاده کرد. این امضای دیجیتال را فقط کسی می تواند تولید کند که کلید خصوصی مربوطه را در اختیار داشته باشد، ولی برای اعتبارسنجی (تعیین درستی) یک امضای دیجیتال فقط به کلید عمومی متناظر با آن کلید خصوصی و اثر انگشت تراکنش نیاز است (نه به خود کلید خصوصی). این ویژگی سودمند رمز نگاری نامتقارن به همگان اجازه می دهد امضای دیجیتال تراکنش ها را بررسی کنند، در حالی که مطمئن هستیم فقط مالک واقعی کلید خصوصی می تواند یک امضای معترض تولید کند.

کلید خصوصی

کلید خصوصی چیزی نیست جز یک عدد که به طور تصادفی انتخاب شده است. مالکیت و کنترل این کلید خصوصی هسته ای کنترل دارای های بیت کوین و آدرس های متناظر با آنها را تشکیل می دهد. کلید خصوصی به منظور تولید امضای دیجیتال به کار می رود و برای اثبات مالکیت بر یک دارایی بیت کوین و خروج کردن آن در یک تراکنش ضروری است. کلید خصوصی هرگز نباید فاش شود، چون با داشتن یک کلید خصوصی همه می توانند بر دارایی بیت کوین متناظر با این کلید

کنترل داشته باشند و آن منابع را خرج کنند. داشتن نسخه‌ی پشتیبان از کلیدهای خصوصی نیز یک ضروری است، چون اگر بک کلید خصوصی را از دست بدھید (گم کنید)، دیگر هرگز نخواهد توانست منابع قفل شده با این کلید را بازیابی کنید (درست مثل گاوصدوقی که کلید آن را گم کرده یا رمز آن را فراموش کنید).

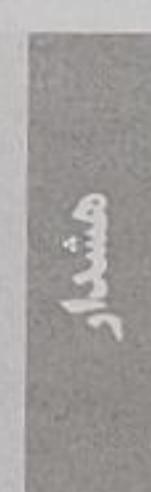
کلید خصوصی بیت کوین فقط یک عدد ۲۵۶-رقمی است. هر عددی که این تعداد رقم داشته باشد، می‌تواند به عنوان کلید خصوصی به کار برده شود: یک سکه را ۲۵۶ بار بیندازید، و هر بار عدد آن را (مثلث شیر = ۱، خط = ۰) یادداشت کنید تا یک کلید خصوصی باینتری داشته باشد که می‌توانید از آن در کیف‌پول بیت کوین خود استفاده کنید. کلیدهای عمومی از این کلید خصوصی ساخته می‌شوند.



تولید کلید خصوصی از یک عدد تصادفی

اولین و مهمترین مرحله در تولید کلیدهای خصوصی یافتن منبعی مطمئن از بی‌نظمی (entropy)، یا تصادفی بودن، است. ایجاد یک کلید بیت کوین اساساً چیزی نیست جز «انتخاب تصادفی یک عدد از ۱ تا ۲۵۶». تازمانی که انتخاب این عدد قابل پیش‌بینی یا تکرار پذیر نباشد، روش آن چندان اهمیتی ندارد. ترم افزار بیت کوین از ماژول مولد عدد تصادفی سیستم عامل برای تولید ۲۵۶ بیت آنتروپی (بی‌نظمی) استفاده می‌کند. ماژول مولد اعداد تصادفی سیستم عامل معمولاً توسط کاربر با یک مقدار اولیه (موسوم به بذر) آماده‌سازی می‌شود، به همین دلیل است که شاید برخی برنامه‌های تولید کلید خصوصی بیت کوین از شما بخواهند برای چند ثانیه ماوس خود را بی‌هدف تکان دهید.

به بیان دقیق‌تر، کلید خصوصی می‌تواند هر عددی بین ۱ تا ۱۱۱۱۰۷۷ باشد، که در آن ۱۱۱ یک عدد ثابت است ($1/158 \times 10^{77}$ ، کمی کوچکتر از ۲۵۶) که به عنوان مرتبه‌ی منحنی بیضوی به کار رفته در بیت کوین تعریف می‌شود (نگران نباشید؛ در ادامه آن را بیشتر توضیح خواهیم داد). برای تولید این کلید، یک عدد تصادفی ۲۵۶-بیتی انتخاب کرد و بررسی می‌کنیم که از ۱-۱۱۱ کوچکتر باشد. از نظر برنامه‌نویسی، این کار معمولاً با تولید یک رشته‌ی طولانی تراز بیت‌های تصادفی (که از یک مولد اعداد تصادفی آمن شده بارمزنگاری گرفته شده‌اند) و دادن آنها به الگوریتم درهم‌سازی SHA256 که خروجی آن همیشه یک عدد ۲۵۶-بیتی است، انجام می‌گیرد. اگر خروجی این الگوریتم (تابع) کوچکتر از ۱-۱۱۱ باشد، کلید خصوصی مناسب به دست آمده است؛ در غیر این صورت، یک عدد تصادفی دیگر را امتحان می‌کنیم.



از ماژول‌های مولد عدد تصادفی ساده‌ای که در زبان‌های برنامه‌نویسی می‌بینید (یا خودتان گذنویسی می‌کنید)، استفاده نکنید؛ به جای آن از یک مولد عدد شبه-تصادفی آمن شده با رمزنگاری (cryptographically secure pseudorandom number generator: CSPRNG) به همراه بذری که از یک منبع با آنتروپی کافی گرفته‌اید، استفاده کنید. برای اطمینان از امنیت کافی یک کتابخانه‌ی مولد عدد تصادفی به مستندات آن مراجعه کنید. پیاده‌سازی صحیح CSPRNG برای امنیت کلیدهای خصوصی ضرورت تام دارد.

در زیر یک کلید خصوصی تصادفی (k) با فرمت هگزادسیمال (که هر رقم آن معادل ۴ بیت باینری است) مشاهده می‌کنید:

1E99423A4ED27608A15A2616A2B0E9E52CED330AC530EDCC32C8FFC6A526AEDD



بزرگی فضای کلیدهای خصوصی بیت کوین (۲۵۶) باورنکردنی است. این عدد تقریباً معادل $10^{۷۷}$ دهدهی است. برای مقایسه، جالب است بدانید تعداد اتم‌های موجود در کل کیهان قابل مشاهده از مرتبه‌ی $10^{۸۰}$ است.

برای تولید یک جفت کلید جدید در مشتری هسته‌ی بیت‌کوین (که در فصل ۳ به طور مفصل درباره‌ی آن صحبت کردیم) از فرمانی به نام `getnewaddress` استفاده می‌کنیم. برای حفظ امنیت، این برنامه فقط کلید عمومی را نمایش می‌دهد، و کلید خصوصی متناظر با آن را مخفی نگه می‌دارد. اگر می‌خواهید `bitcoind` کلید خصوصی متناظر با یک کلید عمومی را نمایش دهد، از فرمان `dumpprivkey` استفاده کنید. فرمان `dumpprivkey` این کلید خصوصی را با گونه‌ای از فرمت Base58 با گذاری جمع‌تطبیقی موسوم به فرمت واردات کیف پول (WIF) (که در ادامه بیشتر درباره‌ی آن توضیح خواهیم داد) نمایش می‌دهد. در زیر یک جفت کلید که با فرمان `getnewaddress` تولید شده‌اند، مشاهده می‌کنید:

```
$ bitcoin-cli getnewaddress
1J7mdg5rbQyUHENYdx39WVWK7fsLpEoXZy
$ bitcoin-cli dumpprivkey 1J7mdg5rbQyUHENYdx39WVWK7fsLpEoXZy
KxFc1jmwwCoACiCAWZ3eXa96mBM6tb3TYzGmf6YwgdGWZgawvrtJ
```

فرمان `dumpprivkey` کیف پول را باز کرده و کلید خصوصی تولید شده توسط فرمان `getnewaddress` را استخراج می‌کند. اگر `bitcoind` این جفت کلید خصوصی-عمومی را در کیف پول خود نداشته باشد، غیرممکن است بتواند کلید خصوصی متناظر با یک کلید عمومی را استخراج کند.

فرمان `dumpprivkey` این کلید خصوصی را از کلید عمومی داده شده تولید نمی‌کند؛ این کار غیرممکن است. این فرمان فقط کلید خصوصی تولید شده توسط فرمان `getnewaddress` را (که از قبل در کیف پول ذخیره شده) نمایش می‌دهد.

برای تولید و نمایش کلیدهای خصوصی از فرمان‌های `seed`, `ec-new`, و `ec-to-wif` ابزار خط-فرمان کاوشگر بیت‌کوین (bx) نیز می‌توان استفاده کرد (پیوست‌های کتاب را ببینید):

```
$ bx seed | bx ec-new | bx ec-to-wif
5J3mBbAH58CpQ3Y5RNJpUKPE62SQ5tfcvU2JpbnkeyhfsYB1Jcn
```

کلید عمومی

کلید عمومی با استفاده از ضرب منحنی بیضوی که تابعی معکوس ناپذیر است، از کلید خصوصی استخراج (محاسبه) می‌شود: به عبارت دیگر، $G \times k = K$ ، که در آن k یک کلید خصوصی معلوم، G یک نقطه‌ی ثابت موسوم به نقطه‌ی مولد (generator point)، و K کلید عمومی متناظر با کلید خصوصی داده شده هستند. عکس این عمل [محاسبه] k با در دست داشتن K ، که به آن «یافتن لگاریتم گستته» گفته می‌شود، کاری به غایت دشوار است چون مستلزم امتحان کردن تک تک مقادیر k خواهد بود. قبل از این که نشان دهیم کلید عمومی چگونه از یک کلید خصوصی ساخته می‌شود، اجازه دهید قدری بیشتر درباره‌ی رمزگاری منحنی بیضوی توضیح دهیم.

ضرب منحنی بیضوی نوعی تابع است که متخصصان رمزگاری به آن تابع «تله موش» می‌گویند: محاسبه‌ی این تابع [ورود به تله موش] از یک طرف (ضرب) آسان، ولی از طرف دیگر (تقسیم) غیرممکن است. کسی که کلید خصوصی را در اختیار دارد، به آسانی می‌تواند کلید عمومی متناظر با آن را تولید کرده و در اختیار دیگران قرار دهد. در حالی که اطمینان دارد هیچ کس قادر به انجام عکس این عمل (استخراج کلید خصوصی از کلید عمومی) نخواهد بود. این ترفند ریاضی مبنای تولید امضاهای دیجیتال غیرقابل جعلی است که مالکیت دارایی‌های بیت‌کوین را اثبات می‌کنند.

آشنایی با رمزگاری منحنی بیضوی

رمزنگاری منحنی بیضوی نوعی رمزگاری نامتقارن یا کلید عمومی بر مبنای مساله‌ی لگاریتم گستته (discrete logarithm) است که به صورت جمع و ضرب نقاط یک منحنی بیضوی بیان می‌شود. در شکل ۲-۴ نمونه‌ای از یک منحنی بیضوی، مشابه آنچه در بیت‌کوین به کار می‌رود، مشاهده می‌کنید.

آنچه در بیت‌کوین از یک منحنی بیضوی و مجموعه‌ی از ثابت‌های ریاضی خاص، که توسط مؤسسه‌ی ملی استاندارد فناوری ایالات متحده (NIST) تحت استاندارد secp256k1 تعریف شده، استفاده می‌کند. منحنی secp256k1 با تابع زیر (که یک منحنی بیضوی تولید می‌کند) تعریف می‌شود:

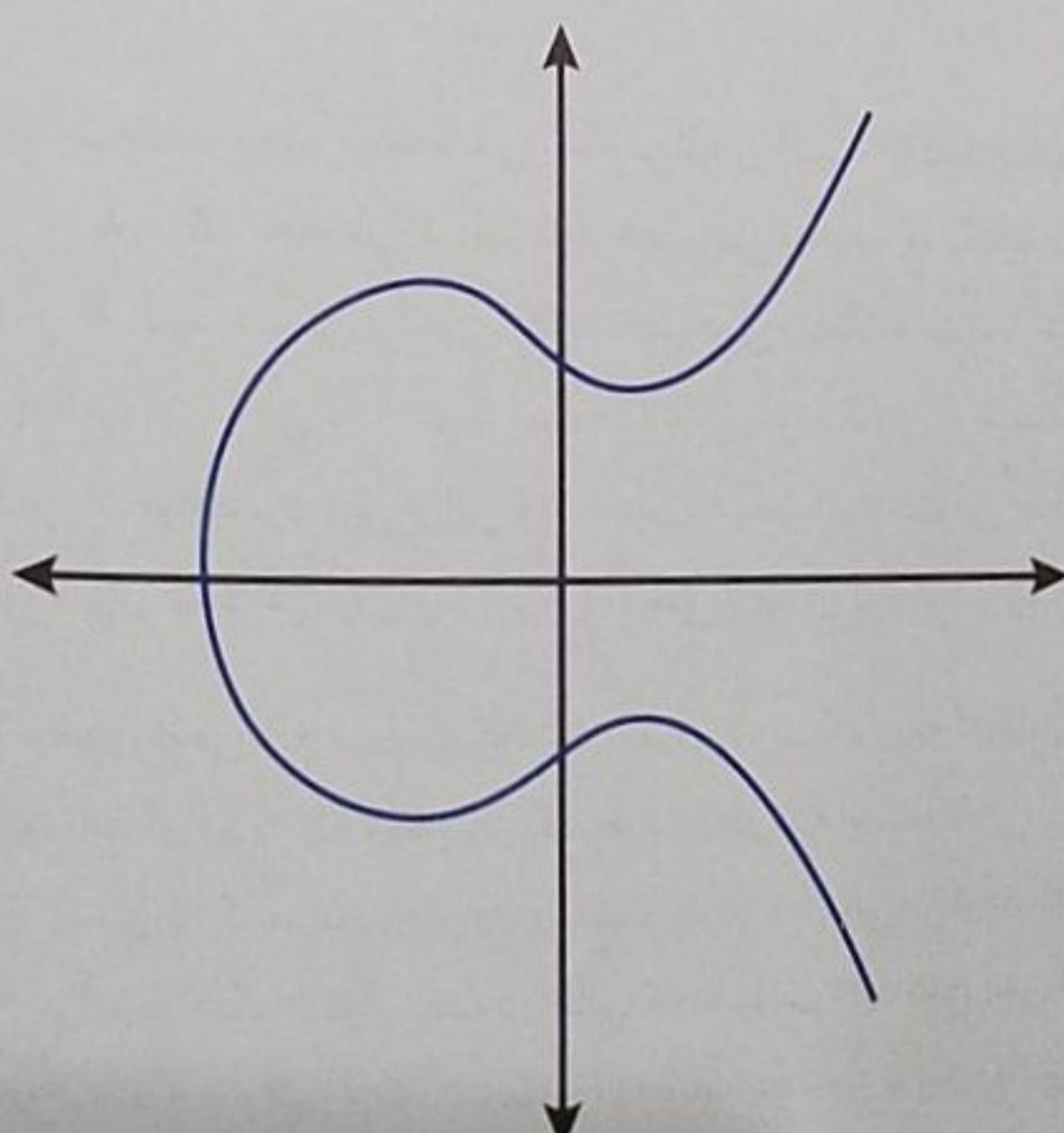
$$y^2 = (x^3 + v) \text{ over } (F_p)$$

$$y^2 \bmod p = (x^3 + v) \bmod p$$

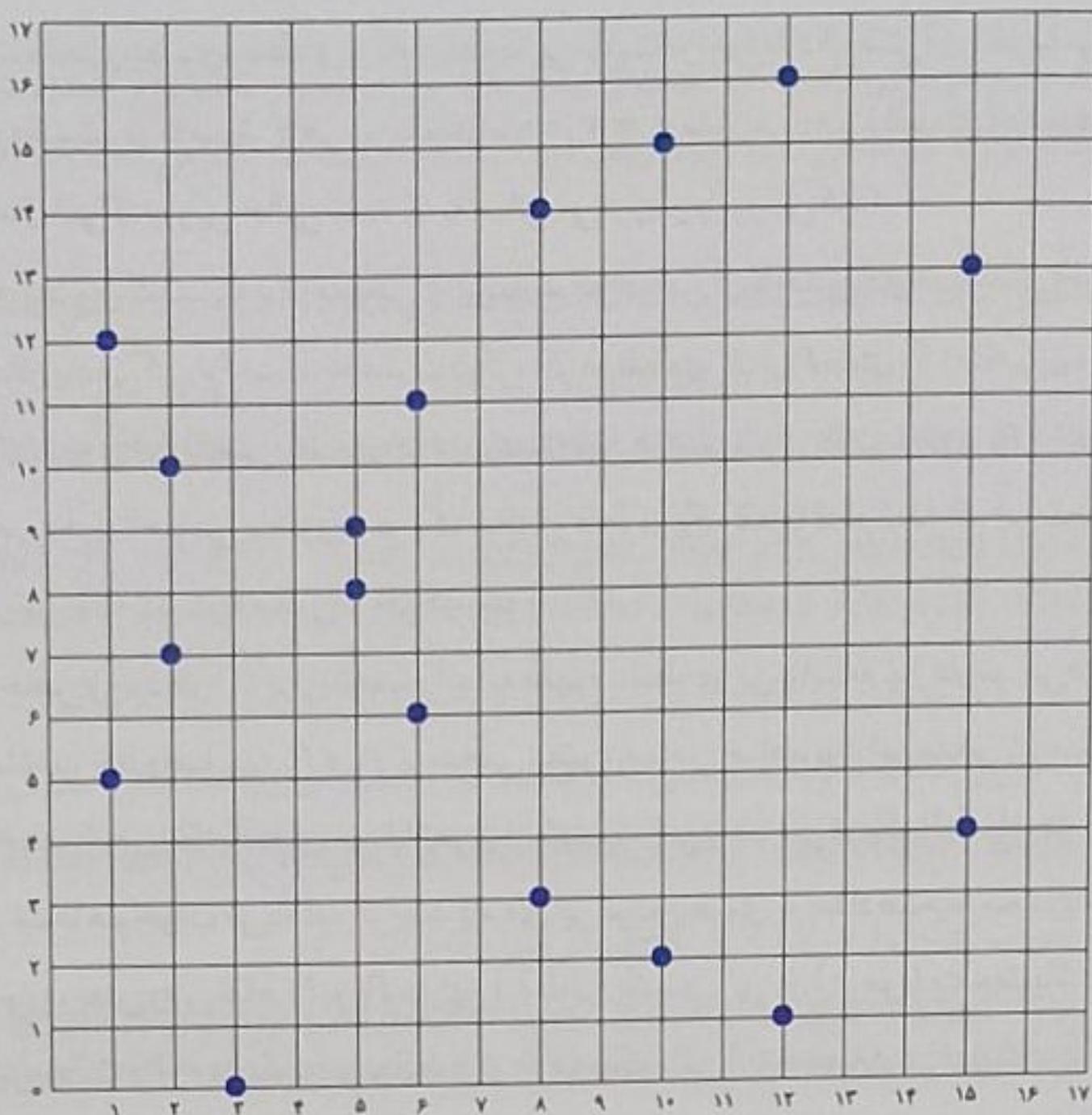
یا

عبارت $p \bmod p$ (پیمانه‌ی عدد اول p) نشان می‌دهد که این منحنی روی یک مجموعه‌ی متناهی از مرتبه‌ی اول p (که صورت F_p هم نوشته می‌شود) تعریف شده، که در آن $1 - 2^4 - 2^6 - 2^7 - 2^8 - 2^9 - 2^{32} - 2^{56} = 2^{256} - 1 = p$ یک عدد اول بسیار بزرگ است.

از آنجاکه این منحنی روی دامنه‌ای متناهی از مرتبه‌ی اول تعریف شده است (نه روی تمام اعداد حقیقی)، نمودار آن از تعدادی نقطه‌ی پراکنده در صفحه‌ی مختصات تشکیل می‌شود که رسم آن را دشوار می‌کند. با این حال، ریاضیات این منحنی کاملاً شبیه ریاضیات یک منحنی بیضوی روی دامنه‌ی اعداد حقیقی است. در شکل ۲-۴ این تابع بیضوی روی یک دامنه متناهی بسیار کوچک از مرتبه‌ی اول ۱۷ ترسیم شده است. منحنی بیضوی secp256k1 به کار گرفته شده در بیت‌کوین را می‌توان مجموعه‌ای از نقاط مشابه این نمودار، ولی با دامنه‌ی بسیار بزرگتر و الگویی بسیار پیچیده‌تر، تصور کرد.



شکل ۲-۴ یک منحنی بیضوی.



شکل ۳-۴ رمزگاری منحنی بیضوی: نمودار یک منحنی بیضوی روی دامنه‌ی $F(p)$ ، با $p = 17$.

برای مثال، نقطه‌ی P با مختصات (x, y) زیر نقطه‌ای واقع بر منحنی secp256k1 است:

```
P =
(55066263022277343669578718895168534326250603453777594175500187360389116729240,
32670510020758816978083085130507043184471273380659243275938904335757337482424)
```

در مثال ۱-۴ یک برنامه‌ی پایتون می‌بینید که به کمک آن خودتان می‌توانید صدق کردن این نقطه در منحنی بیضوی secp256k1 را بررسی کنید:

مثال ۱-۴ برنامه‌ی پایتون برای نشان دادن این که نقطه‌ی P روی منحنی secp256k1 قرار دارد

```
Python 3.4.0 (default, Mar 30 2014, 19:23:13)
[GCC 4.2.1 Compatible Apple LLVM 5.1 (clang-503.0.38)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> p =
115792089237316195423570985008687907853269984665640564039457584007908834671663
>>> x =
55066263022277343669578718895168534326250603453777594175500187360389116729240
>>> y =
32670510020758816978083085130507043184471273380659243275938904335757337482424
>>> (x ** 3 + 7 - y**2) % p
```

در ریاضیات منحنی بیضوی، نقطه‌ای به نام «نقطه در بی‌نهایت» وجود دارد که نقش آن را تقریباً می‌توان متناظر با نقش صفر در عمل جمع دانست. در کامپیوتر گاهی این نقطه را با $y = x$ نشان می‌دهند (نقطه‌ای که مختصات آن در معادله منحنی بیضوی صدق نمی‌کند، ولی حالتی است که به آسانی می‌توان آن را بررسی کرد).

منحنی‌های بیضوی گسته یک عملگر $+$ نیز دارند که خواص آن تا حد زیادی شبیه عمل جمع در حساب اعداد حقیقی (که در دبستان یاد می‌گیریم) است. به عبارت دیگر، اگر دونقطه‌ی P_1 و P_2 روی یک منحنی بیضوی باشند، نقطه‌ی سوم $P_3 = P_1 + P_2$ نیز روی این منحنی بیضوی خواهد بود. به روش هندسی، برای یافتن نقطه‌ی P_3 ، خطی از P_1 به P_2 رسم کرده و آن را امتداد می‌دهیم تا منحنی بیضوی را در یک نقطه (و فقط در یک نقطه) قطع کند؛ اگر این نقطه را $(x, y) = P_3'$ بنامیم، P_3 قرینه‌ی این نقطه نسبت به محور- x خواهد بود.

دو حالت خاص وجود دارد که برای توضیح آنها به مفهوم «نقطه در بی‌نهایت» نیاز داریم. می‌دانیم که اگر P_1 و P_2 یک نقطه‌ی واحد باشند، خط واصل بین P_1 و P_2 بر منحنی بیضوی مماس خواهد بود. این مماس نیز منحنی بیضوی را دقیقاً در یک نقطه قطع خواهد کرد. با تکنیک‌های حسابان می‌توان شبیه مماس بر منحنی در یک نقطه را محاسبه کرد، و جالب این که آن تکنیک‌های در اینجا هم جواب می‌دهند، هر چند دامنه‌ی این توابع بیضوی به اعداد صحیح محدود شده است! در برخی موارد، یعنی وقتی نقاط P_1 و P_2 دارای x (طول) یکسان ولی y (عرض) متفاوت باشند، خط مماس قائم خواهد بود که در این حالت $P_3 = \text{«نقطه در بی‌نهایت»}$. از طرف دیگر، اگر $P_1 = \text{«نقطه در بی‌نهایت»}$ ، آنگاه $P_2 = P_1 + P_2$ به همین ترتیب، اگر $P_2 = \text{«نقطه در بی‌نهایت»}$ ، آنگاه $P_1 = P_1 + P_2$. همان طور که می‌بینید، این «نقطه در بی‌نهایت» نقش «صفرا» (در عمل جمع) را بازی می‌کند.

همچنین معلوم می‌شود که عملگر $+$ خاصیت شرکت پذیری دارد، یعنی $(A + B) + C = A + (B + C)$. به عبارت دیگر، در جمع چند نقطه روی یک منحنی بیضوی می‌توان پراتزه‌ها را حذف کرد و به سادگی نوشت $A + B + C = A + (B + C)$. اکنون که عمل جمع روی منحنی بیضوی را تعریف کردیم، می‌توانیم عمل ضرب را هم به سادگی (مانند حساب معمولی) به عنوان تعمیم عمل جمع از آن نتیجه بگیریم: به بیان دیگر، اگر P نقطه‌ای روی یک منحنی بیضوی و k یک عدد صحیح باشد، آنگاه (k بار) $P + P + P + \dots + P = kP$. توجه کنید که گاهی (به گونه‌ای گیج‌کننده) به k «توان» گفته می‌شود.

تولید کلید عمومی

برای تولید کلید عمومی از یک کلید خصوصی معلوم (k) آن را در یک نقطه‌ی از قبل مشخص شده روی منحنی بیضوی موسوم به نقطه‌ی مولد G (generator point) ضرب می‌کنیم تا به نقطه‌ی دیگری روی همان منحنی برسیم، نقطه‌ای که متناظر با کلید عمومی K است. نقطه‌ی مولد به عنوان بخشی از استاندارد secp256k1 مشخص شده و مقدار آن همواره برای تمامی کلیدهای بیت‌کوین یکسان است، به عبارت دیگر:

$$K = k \times G$$

که در آن k کلید خصوصی، G نقطه‌ی مولد، و K کلید عمومی حاصل (نقطه‌ای روی منحنی بیضوی) هستند. از آنجا که نقطه‌ی مولد همیشه برای تمام کاربران بیت‌کوین یکسان است، ضرب یک کلید عمومی (k) در G همیشه همان کلید عمومی (K) را تولید خواهد کرد. به بیان دیگر، رابطه‌ی بین k و K ثابت است، ولی محاسبه‌ی آن فقط در یک جهت (از k به K) ممکن است. به همین دلیل است که آدرس‌های بیت‌کوین را (که از K استخراج می‌شوند) می‌توان به همگان اعلام کرد، بدون آن که نیازی به فاش کردن کلید خصوصی کاربر (K) باشد.

یک کلید خصوصی را می‌توان به کلید عمومی تبدیل کرد، ولی عکس آن (استخراج کلید خصوصی متناظر با یک کلید عمومی) غیرممکن است، چون ریاضیات آن فقط در یک جهت کار می‌کند.



مقدار نقطه‌ی مولد G در استاندارد secp256k1 به دو صورت فشرده و غیرفشرده داده شده است.
این دو مقدار G را به ترتیب در زیر می‌بینید:

$G^* = 02\ 79BE667E\ F9DCBBAC\ 55A06295\ CE870B07\ 029BFCDB\ 2DCE28D9\ 59F2815B\ 16F81798$

$G^{**} = 04\ 79BE667E\ F9DCBBAC\ 55A06295\ CE870B07\ 029BFCDB\ 2DCE28D9\ 59F2815B\ 16F81798\ 483ADA77\ 26A3C465\ 5DA4FBFC\ 0E1108A8\ FD17B448\ A6855419\ 9C47D08F\ FB10D4B8$

برای تولید کلید عمومی متناظر با کلید خصوصی که در قسمت قبل به دست آوردیم، فقط کافی است آن را در G ضرب کنیم:

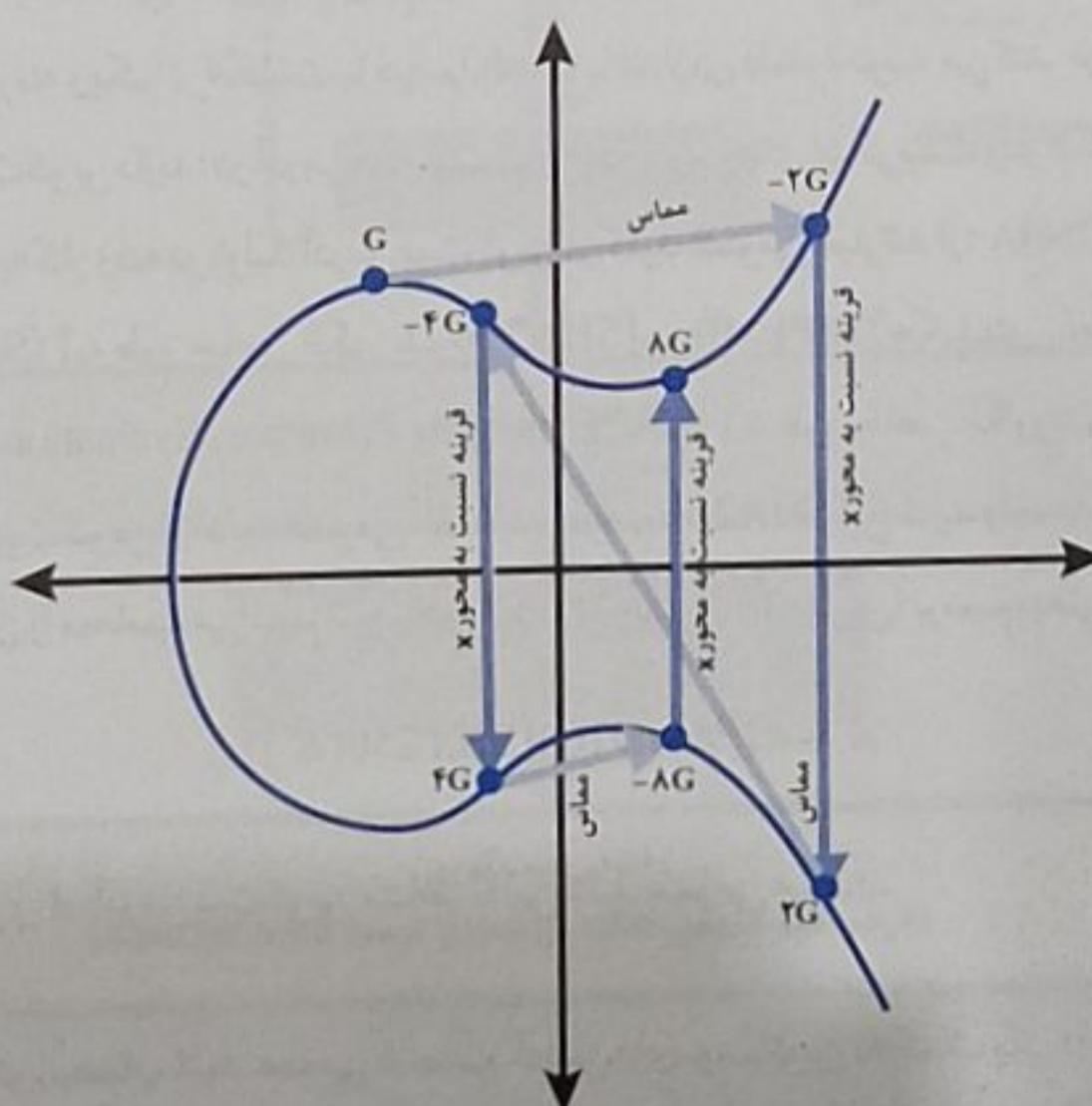
$K = 1E99423A4ED27608A15A2616A2B0E9E52CED330AC530EDCC32C8FFC6A526AEDD * G$

کلید عمومی K به صورت نقطه‌ی $(x, y) = K$ تعریف می‌شود، که در آن

$x = F028892BAD7ED57D2FB57BF33081D5CFCF6F9ED3D3D7F159C2E2FFF579DC341A$

$y = 07CF33DA18BD734C600B96A72BBC4749D5141C90EC8AC328AE52DDFE2E505BDB$

برای نمایش ضرب یک نقطه (روی منحنی بیضوی) در یک عدد صحیح از گونه‌ی ساده‌تر منحنی‌های بیضوی (با دامنه‌ی اعداد حقیقی) استفاده می‌کنیم؛ به یاد داشته باشید که اصول ریاضی آنها یکسان است. در واقع هدف ما یافتن مقدار kG است، و برای این منظور G را k بار با خودش جمع می‌کنیم. همان طور که قبلاً گفتیم، در منحنی‌های بیضوی، جمع کردن یک نقطه با خودش معادل است بارسم مماس بر منحنی در آن نقطه، امتداد دادن این مماس، پیدا کردن نقطه‌ی برخورد خط مماس با منحنی (که نقطه‌ای یکتا است)، و تعیین قرینه‌ی آن نسبت به محور- x . در شکل ۴-۴ فرآیند هندسی تعیین نقاط G ، $2G$ ، $4G$ و $8G$ روی یک منحنی بیضوی نشان داده شده است.



شکل ۴-۴ رمزگاری منحنی بیضوی: ضرب نقطه‌ی G در عدد صحیح k روی یک منحنی بیضوی.

اکثر پیاده‌سازی‌های بیت‌کوین برای انجام محاسبات منحنی بیضوی از کتابخانه‌ی رمزگاری OpenSSL استفاده می‌کنند. برای مثال، برای استخراج کلید عمومی به کمک این کتابخانه می‌توان تابع `(EC_POINT_mul)` را فراخوانی کرد.



آدرس‌های بیت‌کوین

یک آدرس بیت‌کوین رشته‌ای از حروف و ارقام است که می‌توان آن را در اختیار کسانی که می‌خواهند به شما پول حواله کنند، قرار داد. آدرس‌های بیت‌کوین از کلیدهای عمومی استخراج شده و همیشه با رقم «۱» شروع می‌شوند. نمونه‌ای از یک آدرس بیت‌کوین را در زیر می‌بینید:

1J7mdg5rbQyUHENYdx39WVWK7fsLpEoXZy

این آدرس بیت‌کوین همان چیزی است که در پرداخت‌های بیت‌کوین به عنوان «گیرنده» ظاهر می‌شود. آدرس بیت‌کوین را می‌توان معادل «پرداخت شود در وجه ...» در چک‌های بانکی دانست که می‌تواند یک فرد حقیقی یا حقوقی باشد. در چک‌های کاغذی هم نیازی به مشخص کردن شماره حساب گیرنده نیست و می‌توان آن را به یک نام انتزاعی صادر کرد، و همین است که چک را به یک ابزار پرداخت بسیار انعطاف‌پذیر تبدیل کرده است. تراکنش‌های بیت‌کوین نیز برای انعطاف‌پذیری بیشتر از همین نوع انتزاع (که همان آدرس بیت‌کوین است) استفاده می‌کنند. یک آدرس بیت‌کوین می‌تواند نشان‌دهنده‌ی صاحب یک جفت کلید خصوصی/عمومی باشد، یا چیزی دیگر، مثل یک اسکریپت پرداخت (قسمت P2SH را در فصل ۷ ببینید). در حال حاضر فقط حالت ساده‌تر را در نظر می‌گیریم و فرض می‌کنیم آدرس بیت‌کوین نشان‌دهنده‌ی یک کلید عمومی بوده و از آن مستقیم شده است.

آدرس به وسیله‌ی یک تابع ذره‌مسازی رمزگاری یک-طرفه از کلید عمومی استخراج می‌شود. الگوریتم ذره‌مسازی (hashing algorithm)، یا به سادگی الگوریتم ذره (hash algorithm)، یک تابع یک-طرفه است که یک کلید عمومی را به عنوان آرگومان ورودی گرفته و یک اثر انگشت یا ذره (hash) با اندازه‌ی دلخواه تولید می‌کند. تابع ذره‌مسازی رمزگاری کاربرد بسیار وسیعی در بیت‌کوین دارد: در آدرس‌های بیت‌کوین، در آدرس‌های اسکریپت، و در الگوریتم اثبات-کار استخراج بیت‌کوین. الگوریتم‌های به کار رفته در تولید آدرس بیت‌کوین از کلید عمومی عبارتند از: SHA (الگوریتم ذره‌مسازی امن: Secure Hash Algorithm) [به طور خاص الگوریتم SHA256]، و RIPEMD (چکیده‌ی پیام ارزیابی یکپارچگی مبانی رس: RACE Integrity Primitives Evaluation Message Digest) [به طور خاص الگوریتم RIPEMD160]. برای تولید آدرس بیت‌کوین از کلید عمومی K ، ابتدا ذره SHA256 این کلید را محاسبه کرده و سپس ذره RIPEMD160 حاصل آن را محاسبه می‌کنیم تا به یک عدد ۱۶۰-بیتی (۲۰-بایتی) برسیم، یعنی:

$$A = \text{RIPEMD160}(\text{SHA256}(K))$$

که در آن K کلید عمومی و A آدرس بیت‌کوین متناظر با این کلید عمومی هستند.

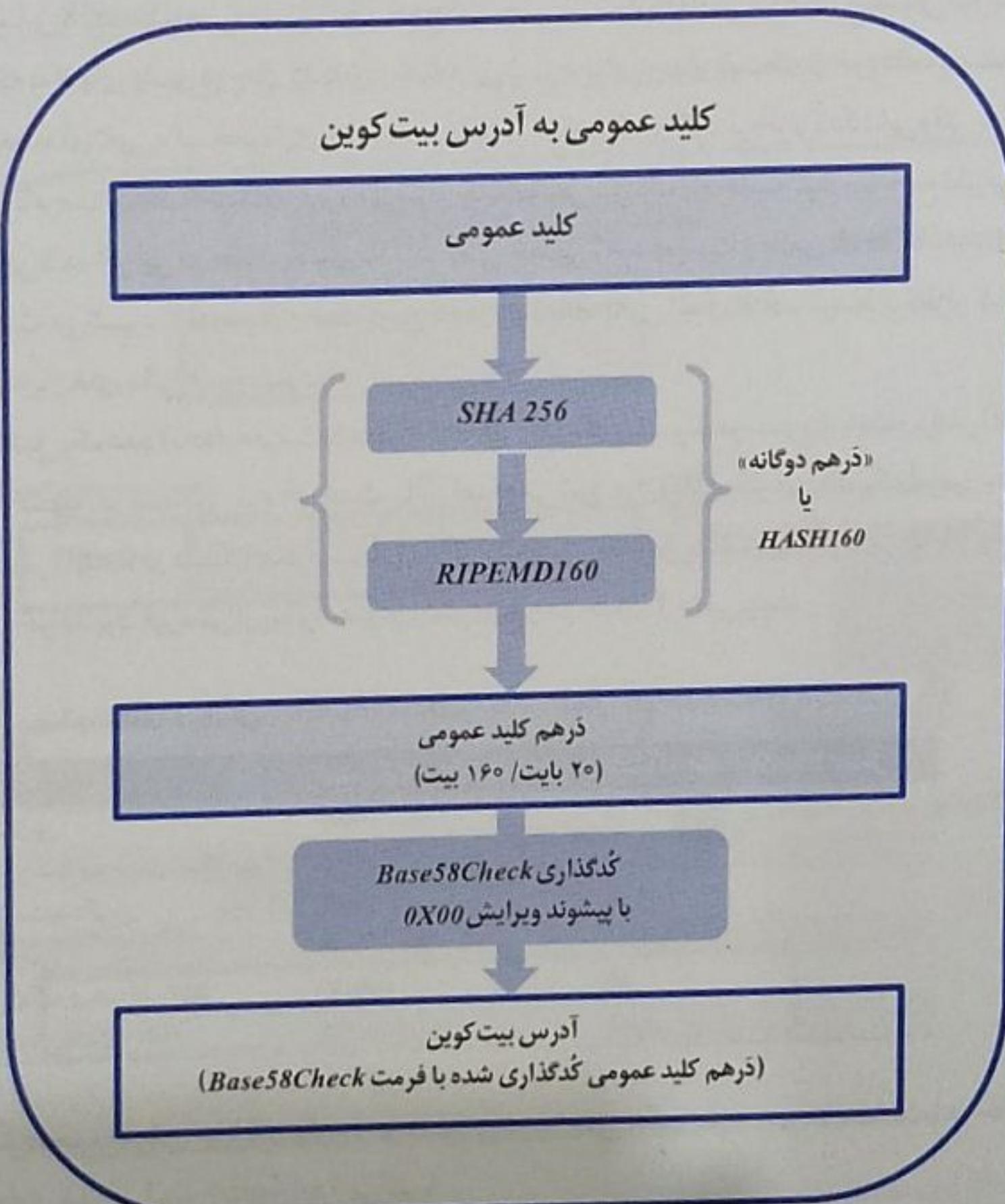
آدرس بیت‌کوین همان کلید عمومی نیست. آدرس‌های بیت‌کوین به کمک یک تابع رمزگاری یک-طرفه از کلیدهای عمومی استخراج می‌شوند.



آدرس‌های بیت‌کوین تقریباً همیشه با فرمت Base58Check گذاری می‌شوند، که از ۵۸ کاراکتر [دستگاه عددنويسي در مبني ۵۸] و یک جمع‌تطبيقي (checksum) برای خوانايي، اجتناب از ابهام، و جلوگيري از بروز خطأ در تراکنش‌های بیت‌کوين استفاده می‌کند. گذاری Base58Check در بسیاري از بخش‌های سистем بیت‌کوين (در واقع هر جا که لازم باشد کاربر خودش اعداد را بخواند یا بنويسد، مثل آدرس‌ها، اسکريپت‌ها و کلید‌های رمزگاري شده) نيز کاربرد دارد. (در قسمت آينده ساز و کار گذاري و گذاري Base58Check را به تفصيل تريح خواهيم کرد). شكل ۴-۵ فرآيند تبديل يك کلید عمومي به آدرس بیت‌کوين را نشان مي‌دهد.

دستگاه عددنويسي Base58Check و گذاري Base58

بسيرى از سیستم‌های کامپیوتري برای نمایش فشرده اعداد بسیار بزرگ (با استفاده از نمادهای کمتر) از دستگاه‌های عددنويسي حروفی-عددی با مبنای بزرگتر از ۱۰ استفاده می‌کنند. برای مثال، در حالی که دستگاه عددنويسي سنتي دهدۀ (مبني ۱۰) از ارقام ۰ تا ۹ برای نمایش اعداد استفاده می‌کند، دستگاه هگزادسيمال (مبني ۱۶) حروف A تا F را نيز به عنوان شش رقم اضافه به کار می‌گيرد. وقتی يك مقدار عددی را در دستگاه هگزادسيمال می‌نويسيد، از حروف کمتری استفاده می‌کنيد و نمایش آن عدد فشرده‌تر خواهد بود. يك دستگاه عددنويسي باز هم فشرده‌تر دستگاه Base64 (مبني ۶۴)



شکل ۴-۵ تبديل کلید عمومي به آدرس بیت‌کوين متناظر با آن.

است که از ۲۶ حرف کوچک الفبا (a تا z)، ۲۶ حرف بزرگ الفبا (A تا Z)، ۱۰ رقم (۰ تا ۹)، و دو علامت دیگر (مثلاً + و //) برای مبادله اطلاعاتی باینری از طریق رسانه های نوشتاری (مانند ایمیل) استفاده می کند؛ در واقع، همان Base64 دستگاه گذاری است که برای پیوست کردن فایل های باینری به کار می رود. دستگاه Base58 دستگاهی است که اولین بار برای استفاده در بیت کوین توسعه داده شد، ولی امروزه در بسیاری از ارزهای رمز بینیان نیز کاربرد دارد. این دستگاه توان زیر مجموعه ای از Base58 است که در آن فقط از حروف بزرگ الفبا (A تا Z)، حروف کوچک الفبا (a تا z)، و ۱۰ رقم (۰ تا ۹) استفاده می شود؛ همچنین، به دلیل احتمال بروز خطای انسانی در کار با کاراکترهایی که ظاهر آنها (مخصوصاً در برخی فونت های خاص) بسیار شبیه یکدیگر است، کاراکترهای ۰ (رقم صفر)، O (اول بزرگ)، I (آی بزرگ)، و علامت «+» و «//» از دستگاه کنار گذاشته شده اند. در مثال ۲-۴ مجموعه کامل کاراکترهای دستگاه عدد نویسی Base58 را مشاهده می کنید.

مثال ۲-۴ الفبای دستگاه عدد نویسی بیت کوین: Base58

123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

برای امنیت بیشتر در مقابل خطاهای تایپی یا نسخه برداری، بیت کوین اغلب از Base58Check که نوعی فرمت گذاری بر اساس Base58 به همراه گذاری تشخیص-خطای داخلی است، استفاده می کند؛ جمع تطبیقی عبارت است از چهار بایت اضافی که به انتهای داده در حال گذاری اضافه می شود. با این چهار بایت که از خود داده ها استخراج می شوند، می توان بروز خطاهای تایپی یا نسخه برداری را تشخیص داده و از آنها جلوگیری کرد. نرم افزار گذگشایی وقتی بایک رشته داده گذاری شده با فرمت Base58Check روبرو می شود، جمع تطبیقی این داده را محاسبه کرده و با جمع تطبیقی گنجانده شده در آن مقایسه می کند؛ اگر این دو مقدار یکسان نباشند، یعنی خطایی رُخ داده است و داده Base58Check نامعتبر است. این تمہید باعث می شود تا آدرس های بیت کوین که در آنها غلط تایپی اتفاق افتاده، توسط نرم افزار کیف پول پذیرفته نشوند و خطر زیان های مالی از بین برود.

برای تبدیل یک عدد (داده) به فرمت Base58Check، ابتدا یک پیشوند موسوم به بایت ویرایش (version byte) که وظیفه آن تسهیل در شناسایی نوع داده است، به آن اضافه می شود. برای مثال، اگر این داده یک آدرس بیت کوین باشد، پیشوند آن صفر (0x00 در دستگاه هگزادسیمال)، و اگر یک کلید خصوصی باشد، پیشوند آن ۱۲۸ (0x80 در دستگاه هگزادسیمال) خواهد بود. فهرستی از پیشوندهای ویرایش رایج را در جدول ۴-۱ می بینید.

جدول ۱-۴ پیشوندهای ویرایش Base58Check و مقدار حاصل آنها در دستگاه Base58

نوع داده	آدرس hex بیت کوین	پیشوند ویرایش (hex)	پیشوند مقدار Base58 حاصل
آدرس P2SH (پرداخت به-درهم-اسکریپت)	0x00	1	
آدرس شبکه‌ی تست بیت کوین	0x05	3	
WIF کلید خصوصی	0x6F	n یا m	
کلید خصوصی گذگاری شده BIP-38	0x80	K یا L	
کلید عمومی گسترش یافته BIP-32	0x0142	6P	
	0x0488B21E	xpub	

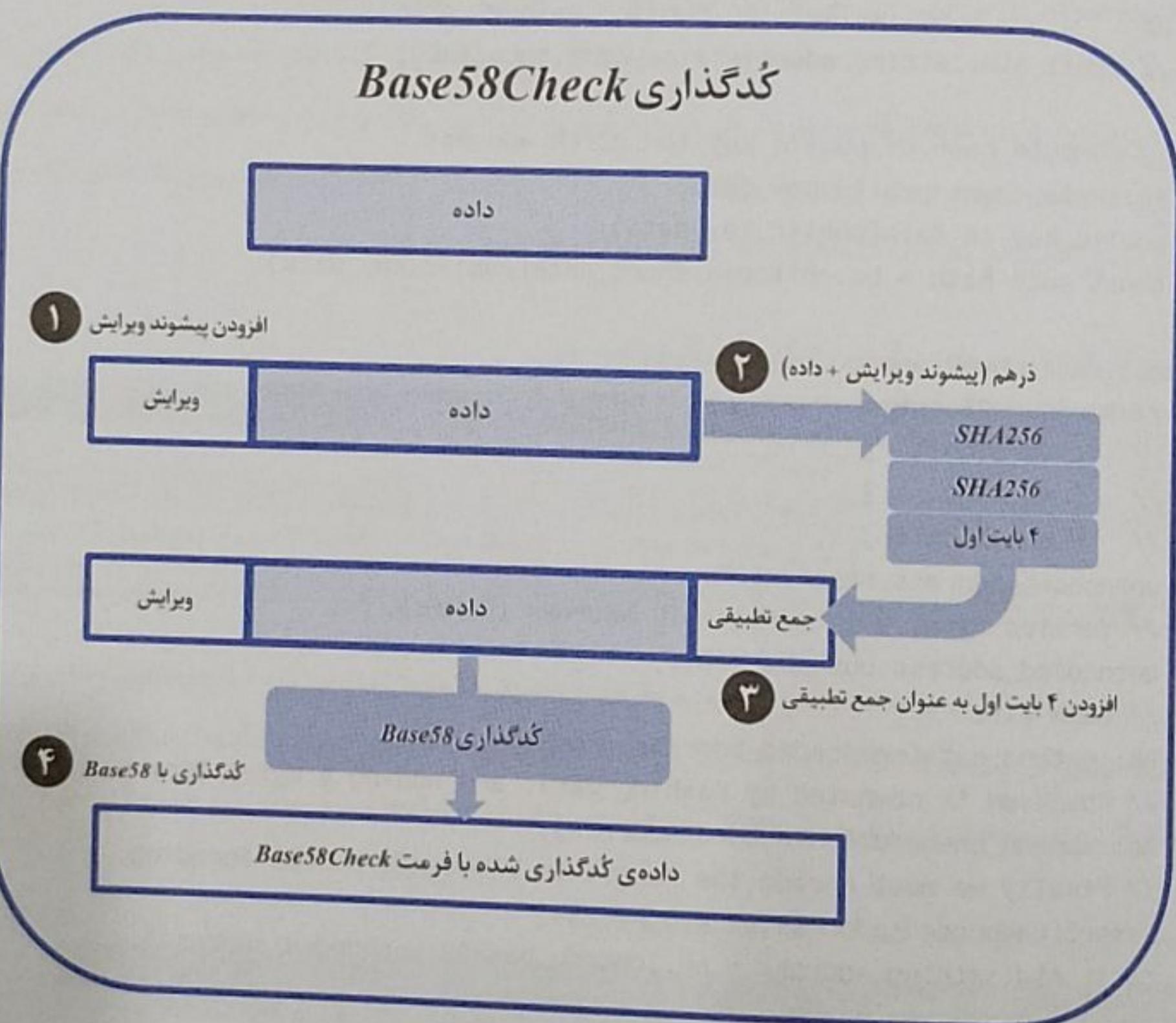
سپس، جمع تطبیقی «SHA دوگانه» محاسبه می شود، یعنی الگوریتم دَرهم SHA256 را دو بار متوالی روی نتیجه هی مرحله‌ی قبل (پیشوند + داده) اجرا می کنیم:

$$\text{checksum} = \text{SHA256}(\text{SHA256}(\text{prefix+data}))$$

حاصل این عبارت یک دَرَهم-دَرَهم (درهم دوگانه) ۳۲-بایتی است که فقط ۴ بایت اول آن به عنوان جمع‌تطبیقی نگه داشته شده و ۲۸ بایت بقیه دور انداخته می‌شود. این جمع‌تطبیقی به انتهای داده چسبانده خواهد شد. حاصل کار تا به اینجا از سه بخش تشکیل شده است: پیشوند، داده‌ی اصلی، و جمع‌تطبیقی. در مرحله‌ی بعد این عبارت با استفاده از الفبای Base58 (که در مثال ۲-۴ می‌بینید) گُدگذاری خواهد شد. شکل ۶-۴ فرآیند گُدگذاری Base58Check را نشان می‌دهد.

به خاطر فشردگی، خوانایی و سهولت در تشخیص خطاهای اکثر داده‌هایی که در بیت‌کوین به کاربر عرضه می‌شوند، با فرمت Base58Check گُدگذاری شده‌اند. پیشوند ویرایش در Base58Check برای سهولت در تشخیص انواع مختلف داده به کار می‌رود؛ در نتیجه وقتی بعد از افزودن پیشوند ویرایش به یک عدد (داده) آن را با فرمت Base58Check گُدگذاری می‌کنیم، خروجی آن رشته‌ای خواهد بود که با یک کاراکتر خاص شروع می‌شود (جدول ۱-۴ را بینید) و تشخیص آن را برای کاربر ساده می‌کند. برای مثال، آدرس‌های بیت‌کوین گُدگذاری شده با فرمت Base58Check همیشه با کاراکتر «۱» شروع می‌شوند که تشخیص آنها را از یک WIF (فرمت واردات کیف‌پول) کلید خصوصی که با کاراکتر «۵» شروع می‌شود، ساده خواهد کرد. اجازه دهید نگاهی بیندازیم به فرآیند تولید آدرس بیت‌کوین، که از یک کلید خصوصی (عدد تصادفی) شروع شده، از آنجا به یک کلید عمومی (نقطه‌ای روی منحنی یضوی secp256k1)، سپس به آدرسی با درهم‌سازی دوگانه، و سرانجام به یک مقدار گُدگذاری شده با فرمت Base58Check ختم می‌شود. در مثال ۳-۴ برنامه‌ای به زبان C++ می‌بینید که این فرآیند را به طور کامل از ابتدا (کلید خصوصی) تا انتها (آدرس بیت‌کوین) پیاده‌سازی کرده است. در این گُد از توابع کتابخانه‌ی libbitcoin (که در قسمت‌های قبل توضیح دادیم) استفاده کرده‌ایم.

گُدگذاری Base58Check



شکل ۶-۴ گُدگذاری Base58Check: افزودن بایت ویرایش و جمع‌تطبیقی به داده‌ی اصلی ابهام در خرابی داده را از بین می‌برد.

مثال ۲-۴ تولید آدرس بیت‌کوین با فرمت Base58Check از یک کلید خصوصی

```
#include <bitcoin/bitcoin.hpp>
int main()
{
    // Private secret key string as base16
    bc::ec_secret decoded;
    bc::decode_base16(decoded,
        "038109007313a5807b2ecc082c8c3fbb988a973cacf1a7df9ce725c31b14776");

    bc::wallet::ec_private secret(
        decoded, bc::wallet::ec_private::mainnet_p2kh);

    // Get public key.
    bc::wallet::ec_public public_key(secret);
    std::cout << "Public key: " << public_key.encoded() << std::endl;

    // Create Bitcoin address.
    // Normally you can use:
    //     bc::wallet::payment_address payaddr =
    //         public_key.to_payment_address(
    //             bc::wallet::ec_public::mainnet_p2kh);
    // const std::string address = payaddr.encoded();

    // Compute hash of public key for P2PKH address.
    bc::data_chunk public_key_data;
    public_key.to_data(public_key_data);
    const auto hash = bc::bitcoin_short_hash(public_key_data);

    bc::data_chunk unencoded_address;
    // Reserve 25 bytes
    // [ version:1 ]
    // [ hash:20 ]
    // [ checksum:4 ]
    unencoded_address.reserve(25);
    // Version byte, 0 is normal BTC address (P2PKH).
    unencoded_address.push_back(0);
    // Hash data
    bc::extend_data(unencoded_address, hash);
    // Checksum is computed by hashing data, and adding 4 bytes from hash.
    bc::append_checksum(unencoded_address);
    // Finally we must encode the result in Bitcoin's base58 encoding.
    assert(unencoded_address.size() == 25);
    const std::string address = bc::encode_base58(unencoded_address);

    std::cout << "Address: " << address << std::endl;
    return 0;
}
```

در این مثال کلید خصوصی در داخل برنامه تعریف شده و مقداری ثابت دارد، بنابراین هر بار که برنامه را اجرا کنید، خروجی یکسانی دریافت خواهید کرد (مثال ۴-۴ را ببینید).

مثال ۴-۴ کامپایل کردن و اجرای برنامه‌ی addr.cpp

```
# Compile the addr.cpp code
$ g++ -o addr addr.cpp $(pkg-config --cflags --libs libbitcoin)
# Run the addr executable
$ ./addr
Public key: 0202a406624211f2abbdc68da3df929f938c3399dd79fac1b51b0e4ad1d26a47aa
Address: 1PRTTaJesdNovgne6EhcdulfpEdX7913CK
```

فرمت کلید

کلیدهای خصوصی و عمومی هر دو می‌توانند با فرمتهای مختلفی نمایش داده شوند. این فرمتهای هر چند در ظاهر متفاوت هستند، همگی یک عدد واحد را کدگذاری می‌کنند. هدف اصلی از به کار گیری فرمتهای مختلف افزایش خوانایی کلیدهای افراد معمولی و همچنین کاهش احتمال بروز خطای خواندن یا نسخه برداری آنها است.

فرمت کلید خصوصی

بک کلید خصوصی را می‌توان به چندین فرمت مختلف نمایش داد؛ توجه داشته باشید که همه‌ی این فرمتهای متناظر با یک عدد ۲۵۶-بیتی واحد هستند. در جدول ۴-۴ سه فرمت رایج برای نمایش کلیدهای خصوصی را مشاهده می‌کنید. این فرمتهای در موارد متفاوت کاربرد دارند. کاربرد فرمتهای هگزادسیمال و باینری خام در داخل برنامه‌ها است و کاربران به ندرت با این دو فرمت برخورد می‌کنند. فرمت WIF برای واردات/صادرات کلید بین برنامه‌های کیف پول به کار می‌رود و اغلب در کد QR کلیدهای خصوصی استفاده می‌شود.

جدول ۴-۴ فرمتهای نمایش (کدگذاری) کلید خصوصی

نوع فرمت	توضیح	پیشوند
raw (خام)	۳۲ بایت (۲۵۶ بیت)	نیازد
(هگز)	۶۴ رقم هگزادسیمال	نیازد
WIF	۵	
-WIF	۰x01 با پسوند K پایا	فشرده

در جدول ۴-۴ یک کلید خصوصی که با این سه فرمت کدگذاری شده، مشاهده می‌کنید. با وجود تفاوت ظاهری بین این فرمتهای آنها یک مقدار (عدد) واحد را نمایش می‌دهند. فرمتهای سادگی قابل تبدیل به یکدیگر هستند. از آنجاکه در این کتاب هرگز از فرمت باینری خام برای نمایش اعداد (کلیدهای، آدرس‌ها، وغیره) استفاده نمی‌کنیم، آن را در جدول ۴-۴ نشان نداده‌ایم.

جدول ۴-۴ مثال: یک کلید با فرمتهای مختلف

فرمت	کلید خصوصی
hex	1e99423a4ed27608a15a2616a2b0e9e52ced330ac530edcc32c8ffc6a526ae
WIF	5J3mBbAH58CpQ3Y5RNJpUKPE62S05tfcvU2JpbnkeyhfsYB1Jcn
-WIF	KxFCljmwwCoAC1CAWZ3eXa96mBM6tb3TYzGmf6YwgdGWZgawvrtJ

برای نمایش کلید خصوصی متناظر با فرمتهای WIF و WIF-فشرده از فرمان `wif-to-ec` کاوشگر:

بیت کوین استفاده می کنیم (پوستهای کتاب را ببینید):

```
$ bx wif-to-ec 5J3mBbAH58CpQ3Y5RNJpUKPE62SQ5tfcvU2JpbnkeyhfsYB1Jcn  
1e99423a4ed27608a15a2616a2b0e9e52ced330ac530edcc32c8ffc6a526aedd  
$ bx wif-to-ec KxFc1jmwwCoACiCAWZ3eXa96mBM6tb3TYzGmf6YwgdGWZgawvrtJ  
1e99423a4ed27608a15a2616a2b0e9e52ced330ac530edcc32c8ffc6a526aedd
```

کُدگشایی از Base58Check

با فرمانهای کاوشگر بیت کوین (bx) به سادگی می توان برای پردازش کلیدها، آدرس ها و تراکنش های بیت کوین اسکریپت پوسته و خط-لوله (pipe) خط-فرمان نوشت. برای مثال، با فرمان `base58check-decode` کاوشگر بیت کوین می توانید یک کلید خصوصی غیرفشرده را به طور کامل کُدگشایی کنید:

```
$ bx base58check-decode 5J3mBbAH58CpQ3Y5RNJpUKPE62SQ5tfcvU2JpbnkeyhfsYB1Jcn  
wrapper  
|  
checksum 4286807748  
payload 1e99423a4ed27608a15a2616a2b0e9e52ced330ac530edcc32c8ffc6a526aedd  
version 128
```

خروجی این فرمان داده اصلی (کلید خصوصی)، پیشوند ویرایش WIF، و جمع تطبیقی آن را نشان می دهد. توجه کنید وقتی از این فرمان برای کُدگشایی یک کلید خصوصی با فرمت WIF-فشرده استفاده می کنید، پسوند ۰۱ را به انتهای آن اضافه می کند، که نشان می دهد کلید عمومی متناظر با این کلید خصوصی باید فشرده شود:

```
$ bx base58check-decode KxFc1jmwwCoACiCAWZ3eXa96mBM6tb3TYzGmf6YwgdGWZgawvrtJ  
wrapper  
|  
checksum 2339607926  
payload 1e99423a4ed27608a15a2616a2b0e9e52ced330ac530edcc32c8ffc6a526aedd01  
version 128
```

کُدگذاری از کلید hex به Base58Check

برای تبدیل یک کلید خصوصی به فرمت Base58Check (معکوس عمل قسمت قبل)، کلید خصوصی مورد نظر را با فرمت hex به فرمان `base58check-encode` داده و پیشوند ویرایش WIF (۱۲۸) را نیز مشخص می کنیم:

```
$ bx base58check-encode  
1e99423a4ed27608a15a2616a2b0e9e52ced330ac530edcc32c8ffc6a526aedd --version 128  
5J3mBbAH58CpQ3Y5RNJpUKPE62SQ5tfcvU2JpbnkeyhfsYB1Jcn
```

کُدگذاری از کلید hex (فشرده) به Base58Check

برای تبدیل یک کلید خصوصی فشرده به فرمت Base58Check، پسوند ۰۱ را به انتهای کلید خصوصی با فرمت hex اضافه کرده و آن را به همراه پیشوند ویرایش WIF (۱۲۸) به فرمان `base58check-encode` می دهیم:

```
$ bx base58check-encode
1e99423a4ed27608a15a2616a2b0e9e52ced330ac530edcc32c8ffc6a526aedd01 --version 128
KxFc1jmwwCoAC1CAWZ3eXa96mBM6tb3TYzGmf6YwgdGWZgawvrtJ
```

حاصل کار یک کلید خصوصی با فرمت WIF-فشرده است که با کاراکتر «K» شروع می‌شود. این کاراکتر نشان می‌دهد که کلید خصوصی کُدگذاری شده دارای پسوند ۰۱ است و فقط باید برای تولید کلیدهای عمومی فشرده به کار بردشود. (در ادامه همین فصل درباره کلیدهای خصوصی و عمومی فشرده صحبت خواهیم کرد.)

فرمت کلید عمومی

کلیدهای عمومی هم با فرمتهای مختلف (فُشرده و غیرفُشرده) ذخیره و نمایش داده می‌شوند. همان طور که قبل‌آمدید، کلیدهای عمومی نقطه‌ای است با مختصات (x, y) روی یک منحنی بیضوی. کلیدهای عمومی معمولاً با پیشوند (هگز) ۰۴ شروع شده و بعد از آن دو عدد ۲۵۶-بیتی (مختصه x و مختصه y این نقطه) می‌آیند. پیشوند ۰۴ برای تمایز کردن کلیدهای عمومی غیرفشرده از کلیدهای عمومی فشرده (که با پیشوند ۰۲ یا ۰۳ شروع می‌شوند) به کار می‌رود. در زیر مختصه‌های x و y کلید عمومی متاظر با کلید خصوصی قسمت قبل را مشاهده می‌کنید:

```
x = F028892BAD7ED57D2FB57BF33081D5CFCF6F9ED3D3D7F159C2E2FFF579DC341A
y = 07CF33DA18BD734C600B96A72BBC4749D5141C90EC8AC328AE52DDFE2E505BDB
```

کلید عمومی متاظر با این مختصه‌های x و y ، به اضافه‌ی پیشوند ۰۴، یک عدد ۵۲۰-بیتی (۶۵ بایت یا ۱۳۰ رقم هگزاده‌یم) خواهد بود ($K = 04xy$):

```
K = 04F028892BAD7ED57D2FB57BF33081D5CFCF6F9ED3D3D7F159C2E2FFF579DC341A
    07CF33DA18BD734C600B96A72BBC4749D5141C90EC8AC328AE52DDFE2E505BDB
```

کلید عمومی فشرده

کلیدهای عمومی فشرده برای اولین بار در بیت‌کوین معرفی شدند تا حجم تراکنش‌هایی را که باید در پایگاه داده‌ی بلاک چین بیت‌کوین ذخیره شوند، کم کنند و مصرف منابع سخت‌افزاری در گره‌ها را کاهش دهند. در اکثر تراکنش‌های بیت‌کوین یک کلید عمومی (که برای اعتبارسنجی مالک بیت‌کوین و خرج کردن آن لازم است) وجود دارد. دیدیم که هر کلید عمومی $(x + y)$ ۵۲۰ بیت است، که وقتی در تعداد تراکنش‌های هر بلاک (که معمولاً به صدها تراکنش بالغ می‌شود) یا تعداد تراکنش‌هایی که در هر روز انجام می‌شود (و تعداد آنها به دهها و صدها هزار تراکنش می‌رسد)، ضرب شود، مقدار آن بسیار چشمگیر خواهد بود.

همان‌طور که در قسمت قبل دیدیم، یک کلید عمومی چیزی نیست جز نقطه‌ای با مختصات (x, y) روی یک منحنی بیضوی. از آنجاکه این منحنی یک تابع ریاضی است، هر نقطه روی آن در معادله‌ی منحنی صدق می‌کند، بنابراین اگر مختصه‌ی x یک نقطه (کلید عمومی) را بدانیم، مختصه‌ی y آن را می‌توانیم با حل معادله $y \equiv x^3 + 7 \pmod{p}$ به دست آوریم. در نتیجه فقط کافی است مختصه‌ی x کلید عمومی را ذخیره کنیم، و مختصه‌ی y آن را با محاسبه به دست آوریم، که این باعث صرفه‌جویی ۲۵۶ بیتی خواهد شد، که تقریباً معادل ۵٪ صرفه‌جویی فضای ذخیره‌سازی در هر تراکنش است!

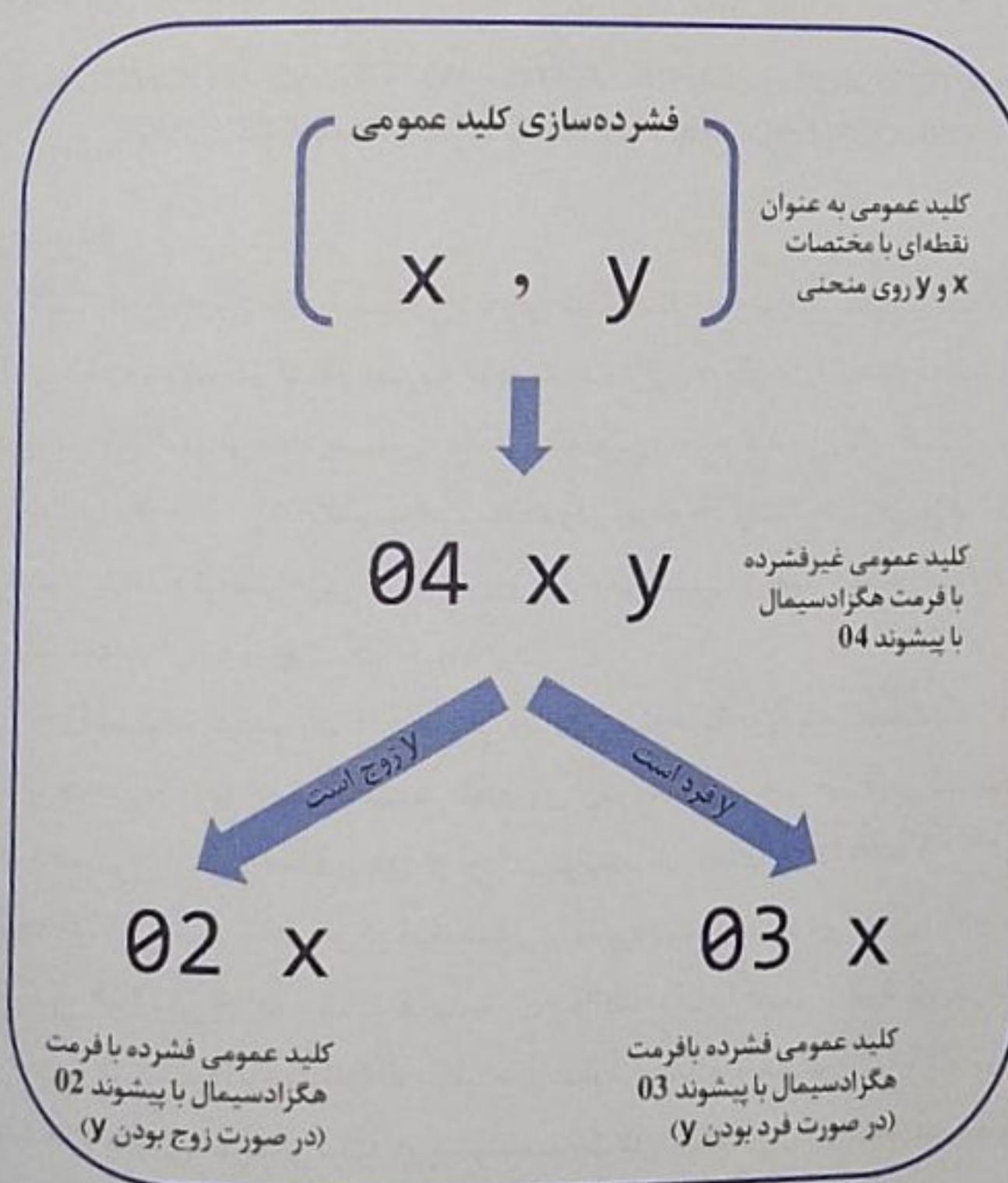
کلیدهای عمومی غیرفشرده با پیشوند ۰۴، و کلیدهای عمومی فشرده با پیشوند ۰۲ یا ۰۳ شروع می‌شوند. اجازه دهد بینیم چرا کلیدهای عمومی فشرده می‌توانند دو پیشوند متفاوت داشته باشند: از آنجاکه سمت چپ معادله‌ی منحنی بیضوی شامل y است، برای یافتن y باید از آن جذر بگیریم، که می‌تواند دو مقدار مثبت (بالای محور- x) یا منفی (پایین محور- x) داشته باشد. همان‌طور که از نمودار منحنی بیضوی (شکل ۲-۴) می‌توانید بیننید، این منحنی نسبت به محور- x

متقارن است. بنابراین، هر چند می‌توان مقدار مختصهٔ x را حذف کرد، ولی علامت آن (مثبت یا منفی) باید ذخیره شود؛ به عبارت دیگر، باید بدانیم این نقطه (کلید عمومی) بالای محور- x بوده است یا پایین آن، چون اینها دو نقطهٔ متقارن و در نتیجه دو کلید عمومی متقارن هستند. وقتی (در ریاضیات باینری) منحنی بیضوی را روی دامنهٔ متاهی از اعداد اول مرتبهٔ P محاسبه می‌کنیم، مختصهٔ x یا زوج است یافتد، که متاظر است با علامت مثبت/منفی این مختصه. بنابراین برای تعیین علامت این مختصه در کلید عمومی فشرده، اگر x زوج باشد، کلید عمومی را با پیشوند ۰۲، و اگر فرد باشد، آن را با پیشوند ۰۳ ذخیره می‌کنیم تا برنامه بتواند مختصهٔ x در کلید فشرده استخراج کرده و کلید عمومی غیرفسرده را به دست آورد. فرآیند فشرده‌سازی کلید عمومی در شکل ۷-۴ نشان داده شده است.

کلید عمومی فشرده (۰۴-بیتی) کلید خصوصی قسمت قبل را در زیر می‌بینید (توجه کنید که به دلیل فرد بودن مختصهٔ x این کلید پیشوند ۰۳ دارد) :

$x = 03F028892BAD7ED57D2FB57BF33081D5CFCF6F9ED3D3D7F159C2E2FFF579DC341A$

با آن که این کلید عمومی متاظر با همان کلید خصوصی قبلی است (یعنی از همان کلید خصوصی استخراج شده)، ولی ظاهری متقارن دارد. از آن مهم‌تر، اگر این کلید عمومی را به کمک تابع درهم دوگانه (یعنی K) (RIPEMD160(SHA256)) به آدرس بیت‌کوین تبدیل کنیم، یک آدرس بیت‌کوین متقارن تولید خواهد شد. این شاید کیج‌کننده باشد، چون به معنای آن



شکل ۷-۴ فرآیند فشرده‌سازی کلید عمومی.

است که با یک کلید خصوصی واحد می‌توان دو کلید عمومی در فرمات‌های مختلف (فشرده و غیرفشرده) ساخت که دو آدرس بیت‌کوین متفاوت تولید می‌کنند. با این حال، (آنچه اهمیت دارد این است که) کلید خصوصی متناظر با این آدرس‌های بیت‌کوین همچنان یکسان است.

کلیدهای عمومی فشرده به تدریج در حال تبدیل شدن به کلیدهای پیش‌فرض در میان مشتری‌های بیت‌کوین هستند، چون اندازه‌ی تراکنش‌ها (و در نتیجه اندازه‌ی بلاک‌چین) را به مقدار زیادی کاهش می‌دهند. با این حال، همه مشتری‌های بیت‌کوین هنوز از این فرمات پشتیبانی نمی‌کنند. مشتری‌های جدیدی که از کلیدهای عمومی فشرده پشتیبانی می‌کنند، همچنان باید بتوانند تراکنش‌های متعلق به مشتریان قدیمی را که در آنها از کلیدهای عمومی غیرفشرده استفاده شده، پردازش کنند. این به ویژه هنگام وارد کردن کلیدهای خصوصی از دیگر برنامه‌های کیف‌پول بیت‌کوین اهمیت بیشتری می‌یابد، چون کیف‌پول جدید باید کل بلاک‌چین را جستجو کرده و تراکنش‌های متناظر با این کلیدهای واردشده را پیدا کند. این کیف‌پول بیت‌کوین باید دنبال چه آدرس‌هایی بگردد؟ آدرس‌های بیت‌کوین تولید از کلیدهای عمومی غیرفشرده، یا آدرس‌های بیت‌کوین تولید از کلیدهای خصوصی فشرده؟ این دو فرمت هر دو معتر هستند، و می‌توان از آنها برای امضای تراکنش‌ها استفاده کرد، ولی آدرس‌های تولیدشده از آنها یکسان نیستند! برای حل این مشکل، وقتی یک کیف‌پول جدید بیت‌کوین قصد دارد کلیدهای خصوصی خود را صادر کند، فرمات WIF متفاوتی برای این منظور به کار می‌گیرد، تا نشان دهد که از این کلیدهای خصوصی برای تولید کلیدهای عمومی فشرده (و در نتیجه، آدرس‌های بیت‌کوین فشرده) استفاده شده است. این ویژگی اجازه می‌دهد تا کیف‌پول واردکننده بین کلیدهای خصوصی قدیمی و جدید تمایز قائل شده و (متناوب با آن) بلاک‌چین را به دنبال آدرس‌های بیت‌کوین متناظر با کلیدهای عمومی غیرفشرده یافشوده جستجو کند. در قسمت بعد این فرآیند را با جزئیات بیشتر تشریح خواهیم کرد.

کلید خصوصی فشرده

جالب است بدانید که واژه‌ی «کلید خصوصی فشرده» غلط انداز است: وقتی یک کلید خصوصی را با فرمات WIF-فشرده کدگذاری می‌کنید، در واقع طول آن یک بایت از یک کلید خصوصی «غیرفشرده» بیشتر می‌شود، چون یک پسوند ۱-باپتی به این کلید خصوصی افزوده خواهد شد؛ جدول ۴-۴ را ببینید. این پسوند ۱-باپتی (مقدار ۰۱ در انتهای کلید خصوصی با فرمات hex-فشرده) به برنامه‌های کیف‌پول جدید می‌گوید این کلید خصوصی فقط باید برای تولید کلیدهای عمومی فشرده به کار برد شود. کلیدهای خصوصی خود فشرده نیستند و نمی‌توان آنها را فشرده کرد. در حقیقت، اصطلاح «کلید خصوصی فشرده» فقط به این معنا است که «از این کلید خصوصی فقط باید برای تولید کلیدهای عمومی فشرده استفاده شود»، و «کلید خصوصی غیرفشرده» یعنی «این کلید خصوصی را فقط باید برای تولید کلیدهای عمومی غیرفشرده به کار برد». برای اجتناب از سردگمی، صفت «فشرده» برای خود کلیدهای خصوصی به کار برد نمی‌شود، و فقط فرمت واردات/صادرات آنها را با «WIF» و «WIF-فشرده» متمایز می‌کنیم. جدول ۴-۴ یک کلید خصوصی واحد را با فرمات‌های WIF و WIF-فشرده نشان داده است.

جدول ۴-۴ مثال: یک کلید خصوصی با فرمات‌های مختلف

کلید خصوصی

فرمت

hex	WIF	hex-فشرده	WIF-فشرده
1E99423A4ED27608A15A2616A2B0E9E52CED330AC530EDCC32C8FFC6A526AEDD			
5J3mBbAH58CpQ3Y5RNJpUKPE62S05tfcvU2JpbnkeyhfsYB1Jcn			
1E99423A4ED27608A15A2616A2B0E9E52CED330AC530EDCC32C8FFC6A526AEDD01			
KxFc1jmwwCoAC1CAWZ3eXa96mBM6tb3TYzGmf6YwgdGWZgawvrtJ			

همان طور که در این جدول می‌بینید، کلید خصوصی با فرمت hex-فشرده یک بایت (مقدار ۱^۰ در انتهای کلید)، اضافه تر دارد. با آن که پیشوند ویرایش کُدگذاری Base58 برای هر دو فرمت WIF و WIF-فشرده یکسان است (0x80)، اضافه شدن یک بایت به انتهای کلید hex-فشرده باعث می‌شود تا اولین کاراکتر آن (بعد از کُدگذاری Base58) از ۱۵۱ به «K» یا «L» تغییر کند. این را می‌توان معادل تفاوت اعداد ۱۰۰ و ۹۹ در کُدگذاری دهد. دهنده دانست: با این که ۱۰۰ فقط یک واحد از ۹۹ بزرگتر است، ولی پیشوند آن از ۹ به ۱ تغییر کرده است. با تغییر طول کلید، پیشوند آن هم تغییر خواهد کرد در Base58، اضافه شدن یک بایت به عدد (کلید خصوصی) پیشوند آن را از «۵» به «K» یا «L» تغییر می‌دهد.

به یاد داشته باشید که این فرمت‌ها یکسان نیستند و نباید آنها را به جای یکدیگر به کار برد. در یک کیف‌پول جدید که از کلیدهای عمومی فشرده استفاده می‌کند، کلیدهای خصوصی فقط با فرمت WIF-فشرده (با پیشوند «K» یا «L») وارد می‌شوند. اگر این کیف‌پول قدیمی بوده و استفاده از کلیدهای عمومی فشرده در آن پیاده‌سازی نشده باشد، کلیدهای خصوصی فقط با فرمت WIF (با پیشوند «۵») وارد خواهد شد. در اینجا هدف آن است که به کیف‌پول وارد کننده‌ی این کلیدهای خصوصی گفت شود که بلاک‌چین را باید چگونه (با استفاده از کلیدهای عمومی و آدرس‌های بیت‌کوین فشرده یا غیر فشرده) جستجو کند.

اگر یک کیف‌پول بیت‌کوین قادر به پیاده‌سازی کلیدهای عمومی فشرده باشد، از این فرمت در تراکنش‌ها استفاده خواهد کرد. کلیدهای خصوصی موجود در این کیف‌پول برای محاسبه کلیدهای عمومی فشرده (روی منحنی یضوی) به کار خواهند رفت، و این کلیدهای عمومی فشرده به نوبه‌ی خود برای تولید آدرس‌های بیت‌کوین فشرده به کار گرفته خواهد شد. هنگام صادر کردن کلیدهای خصوصی از یک کیف‌پول جدید که از کلیدهای عمومی فشرده پشتیبانی می‌کند، کُدگذاری WIF آنها تغییر داده شده و یک پیشوند ۱-بایتی ۱^۰ به تمامی آنها اضافه می‌شود. بعد از کُدگذاری Base58Check، این کلیدهای خصوصی به فرمت «WIF-فشرده» (با پیشوند «K» یا «L») تبدیل خواهند شد.

«کلید خصوصی فشرده» غلط انداز است! کلیدهای خصوصی فشرده نمی‌شوند؛ صفت «WIF-فشرده» یعنی از این کلید فقط باید برای استخراج کلیدهای عمومی فشرده (و آدرس‌های بیت‌کوین متناظر) استفاده کرد. حال این که اگر یک کلید خصوصی را با فرمت WIF-فشرده کُدگذاری کنید، طول آن یک بایت بیشتر خواهد شد. چون پسوند ۰۱ به انتهای این کلید افزوده می‌شود تا آن را از کلیدهای غیر فشرده متمایز کند.



پیاده‌سازی کلید و آدرس در پایتون

جامع‌ترین کتابخانه‌ی بیت‌کوین به زبان پایتون کتابخانه‌ی pybitcointools (ساخته‌ی ویتالیک بوترین) است. در مثال ۴-۵ یک برنامه‌ی پایتون به نام key-to-address-ecc-example.py می‌بینید که از این کتابخانه (که با نام «bitcoin» به برنامه پیوست شده) برای تولید و نمایش چند کلید و آدرس با فرمت‌های گوناگون استفاده کرده است.

مثال ۴-۵ تولید و فرمت کلید و آدرس با کتابخانه‌ی pybitcointools

```
import bitcoin

# Generate a random private key
valid_private_key = False
while not valid_private_key:
    private_key = bitcoin.random_key()
    decoded_private_key = bitcoin.decode_privkey(private_key, 'hex')
```

```

valid_private_key = 0 < decoded_private_key < bitcoin.N

print "Private Key (hex) is: ", private_key
print "Private Key (decimal) is: ", decoded_private_key

# Convert private key to WIF format
wif_encoded_private_key = bitcoin.encode_privkey(decoded_private_key, 'wif')
print "Private Key (WIF) is: ", wif_encoded_private_key

# Add suffix "01" to indicate a compressed private key
compressed_private_key = private_key + '01'
print "Private Key Compressed (hex) is: ", compressed_private_key

# Generate a WIF format from the compressed private key (WIF-compressed)
wif_compressed_private_key = bitcoin.encode_privkey(
    bitcoin.decode_privkey(compressed_private_key, 'hex'), 'wif')
print "Private Key (WIF-Compressed) is: ", wif_compressed_private_key

# Multiply the EC generator point G with the private key to get a public key point
public_key = bitcoin.fast_multiply(bitcoin.G, decoded_private_key)
print "Public Key (x,y) coordinates is:", public_key

# Encode as hex, prefix 04
hex_encoded_public_key = bitcoin.encode_pubkey(public_key, 'hex')
print "Public Key (hex) is:", hex_encoded_public_key

# Compress public key, adjust prefix depending on whether y is even or odd
(public_key_x, public_key_y) = public_key
if (public_key_y % 2) == 0:
    compressed_prefix = '02'
else:
    compressed_prefix = '03'
hex_compressed_public_key = compressed_prefix + bitcoin.encode(public_key_x, 16)
print "Compressed Public Key (hex) is:", hex_compressed_public_key

# Generate bitcoin address from public key
print "Bitcoin Address (b58check) is:", bitcoin.pubkey_to_address(public_key)

# Generate compressed bitcoin address from compressed public key
print "Compressed Bitcoin Address (b58check) is:", \
    bitcoin.pubkey_to_address(hex_compressed_public_key)

```

مثال ۶-۴ اجرای برنامه‌ی key-to-address-ecc-example.py را در مثال ۶-۴ مشاهده می‌کنید.

مثال ۶-۴ اجرای برنامه‌ی key-to-address-ecc-example.py

```
$ python key-to-address-ecc-example.py
Private Key (hex) is:
```

```
3aba4162c7251c891207b747840551a71939b0de081f85c4e44cf7c13e41daa6
Private Key (decimal) is:
26563230048437957592232553826663696440606756685920117476832299673293013768870
Private Key (WIF) is:
5JG9hT3beGTJuUAmCQEmNaxAuMacCTfXuw1R3FCXig23RQHMr4K
Private Key Compressed (hex) is:
3aba4162c7251c891207b747840551a71939b0de081f85c4e44cf7c13e41daa601
Private Key (WIF-Compressed) is:
KyBsPxxTuVD82av65KZkrGrWi5qlMah5SdNq6uftawDbgKa2wv6S
Public Key (x,y) coordinates is:
(41637322786646325214887832269588396900663353932545912953362782457239403430124
16388935128781238405526710466724741593761085120864331449066658622400339362166L
Public Key (hex) is:
045c0de3b9c8ab18dd04e3511243ec2952002dbfad864b9628910169d9b9b00ec
243bcfed4347074d44bd7356d6a53c495737dd96295e2a9374bf5f02ebfc176
Compressed Public Key (hex) is:
025c0de3b9c8ab18dd04e3511243ec2952002dbfad864b9628910169d9b9b00ec
Bitcoin Address (b58check) is:
1thMirt546nngXqyPEz532S8fLwbozud8
Compressed Bitcoin Address (b58check) is:
14cxpo3MBCYYWCgF74SWTdcmxipnGUspw3
```

در مثال ۷-۴ اسکریپت دیگری به نام `ec-math.py` می‌بینید که در آن (به جای یک کتابخانه‌ی تخصصی) از کتابخانه‌ی ریاضی `ECDSA` پایتون برای محاسبات منحنی بیضوی استفاده شده است.

مثال ۷-۴ تولید کلیدهای خصوصی و عمومی و آدرس بیت کوین با محاسبات مستقیم منحتی بیضوی

```

def random_secret():
    convert_to_int = lambda array: int("".join(array).encode("hex"), 16)

# Collect 256 bits of random data from the OS's cryptographically secure random generator
byte_array = os.urandom(32)

return convert_to_int(byte_array)

def get_point_pubkey(point):
    if point.y() & 1:
        key = '03' + '%064x' % point.x()
    else:
        key = '02' + '%064x' % point.x()
    return key.decode('hex')

def get_point_pubkey_uncompressed(point):
    key = '04' +
          '%064x' % point.x() +
          '%064x' % point.y()
    return key.decode('hex')

# Generate a new private key.
secret = random_secret()
print "Secret: ", secret

# Get the public key point.
point = secret * generator
print "EC point:", point

print "BTC public key:", get_point_pubkey(point).encode("hex")

# Given the point (x, y) we can create the object using:
point1 = ecdsa.ellipticcurve.Point(curve, point.x(), point.y(), ec_order)
assert point1 == point

```

خروجی اسکریپت ec-math.py را در مثال ۸-۴ می‌بینید.

مثال ۸-۴ اجرای اسکریپت ec-math.py

```

$ # Install Python PIP package manager
$ sudo apt-get install python-pip
$ # Install the Python ECDSA library
$ sudo pip install ecdsa
$ # Run the script
$ python ec-math.py

Secret:
38090835015954358862481132628887443905906204995912378278060168703580660294000
EC point:

```

(70048853531867179489857750497606966272382583471322935454624595540007269312627,
 10526220647868674319106080026347958932992020952728580393573602168604554235380)
 BTC public key: 029ade3effb0a67d5c8609850d797366af428f4a0d5194cb221d807770a1522873

در مثال ۴-۷ از تابع `os.urandom` استفاده شده، که یک «مولد عدد تصادفی آمن شده با رمزنگاری» (CSRNG) ارائه شده توسط سیستم عامل است. در سیستم‌های شبه-یونیکس (مانند لینوکس) نام این تابع `/dev/urandom` است. اگر این تابع نتواند یک منبع آنتروپی (تصادفی بودن) مناسب پیدا کند، با خطای `CryptGenRandom` `NotImplementedError` متوقف خواهد شد. تابع مولد عدد تصادفی به کار رفته در مثال ۴-۷ فقط برای اهداف آموزشی است؛ برای تولید کلیدهای بیت‌کوین در برنامه‌های حرفه‌ای باید از یک مولد عدد تصادفی با امنیت کافی استفاده کنید.

کلیدها و آدرس‌های پیشرفتی

در ادامه‌ی این فصل نگاهی خواهیم انداشت به آشکال پیشرفتی کلید و آدرس، مانند کلید خصوصی رمزگذاری شده، آدرس اسکریپت، آدرس چندامضایی، آدرس ویژه، و کیف‌پول کاغذی.

کلید خصوصی رمزگذاری شده (BIP-38)

کلیدهای خصوصی باید مخفی و محروم‌انه بمانند، اما برآورده کردن این نیاز امنیتی در عمل بسیار دشوار است، چون در تضاد مستقیم با دیگر خصوصیت امنیتی ضروری کلیدهای خصوصی یعنی سهولت دسترسی قرار دارد. مخفی نگه داشتن کلیدهای خصوصی به ویژه زمانی دشوارتر می‌شود که بخواهیم (برای جلوگیری از فراموشی یا گم شدن) یک نسخه‌ی پشتیاز از آنها بگیریم. ذخیره کردن کلیدهای خصوصی در یک کیف‌پول که با استفاده از گذرواژه رمزگذاری شده، احتمالاً از امنیت کافی برخوردار است، ولی از خود این کیف‌پول هم باید نسخه‌ی پشتیاز گرفته شود. علاوه بر آن، گاهی اوقات لازم می‌شود کلیدهای خصوصی خود را (مثلًا برای ارتقا یا عوض کردن برنامه‌ی کیف‌پول) از یک کیف‌پول به کیف‌پول دیگر منتقل کنیم. نسخه‌ی پشتیاز کلیدهای خصوصی را می‌توان در یک کیف‌پول کاغذی یا دستگاه‌های ذخیره‌سازی خارجی (مثل حافظه‌ی فلاش USB) ذخیره کرد. ولی اگر این نسخه‌ها مفقود یا دزدیده شوند، چطور؟ این نیازهای امنیتی منقاد منجر به ارائه یک استاندارد ساده و قابل حمل برای رمزگذاری کلیدهای خصوصی شده که برای کیف‌پول‌ها و مشتری‌های مختلف بیت‌کوین قابل استفاده است؛ این استاندارد BIP-38 نام دارد.

BIP-38 یک استاندارد مشترک برای رمزگذاری کلیدهای خصوصی با یک گذرجمله (passphrase) و کُدگذاری آنها با فرمت Base58Check پیشنهاد می‌کند، به طوری که بتوان از آنها با امنیت کافی نسخه‌ی پشتیاز گرفت و ذخیره کرد. آنها را بین کیف‌پول‌های مختلف جابجا کرد، یا در محیط‌هایی که احتمال فاش شدن آنها هست، نگهداری کرد. BIP-38 از استاندارد AES (استاندارد رمزگذاری پیشرفتی: Advanced Encryption Standard) برای رمزگذاری استفاده می‌کند، استانداردی که توسط NIST (مؤسسه ملی استانداردهای ایالات متحده) وضع شده و کاربرد گسترده‌ای در رمزگذاری داده‌های صنعتی و نظامی دارد.

الگوریتم رمزگذاری BIP-38 کار خود را با گرفتن یک کلید خصوصی بیت‌کوین به عنوان ورودی شروع می‌کند، کلیدی که معمولاً فرمت WIF (رشته‌ی کُدگذاری شده با Base58Check با پیشوند «5») دارد. این الگوریتم همچنین یک گذرجمله می‌گیرد؛ گذرجمله گذرواژه‌ای است طولانی تراز گذرواژه‌های معمولی، که معمولاً ترکیبی پیچیده از حروف و ارقام

مختلف است و از چندین کلمه تشکیل می‌شود. خروجی الگوریتم رمزگذاری ۳۸-BIP یک کلید خصوصی کُدگذاری شده با فرمت Base58Check است که با پیشوند «6P» شروع می‌شود. اگر یک کلید خصوصی دیدید که با ۶P شروع شده، باید بدانید که این کلید رمزگذاری شده است و قبل از این که بتوانید از آن در هر کیف پولی استفاده کنید، باید آن را رمزگشایی کرده و به یک کلید خصوصی با فرمت WIF (با پیشوند «5») تبدیل کنید. اغلب برنامه‌های کیف پول جدید کلیدهای خصوصی ۳۸-BIP را می‌شناسند و بلافاصله از کاربر درخواست یک گذر جمله می‌کنند تا بتوانند آنها را رمزگشایی کنند. برنامه‌های کاربردی شخص-ثالث، مانند برنامه‌ی فوق العاده سودمند مرورگر-محور Bit Address (به آدرس <https://www.bitaddress.org>)، از کلیدهای خصوصی ۳۸-BIP پشتیبانی کرده و می‌توانند آنها را رمزگشایی کنند.

رایج‌ترین کاربرد کلیدهای ۳۸-BIP در کیف پول‌های کاغذی است، کیف پول‌هایی که می‌توان از آنها برای ذخیره کردن کلیدهای خصوصی روی یک تکه کاغذ استفاده کرد. به شرط انتخاب گذر جمله‌ی قوی، یک کیف پول کاغذی با رمزگذاری ۳۸-BIP وسیله‌ای بسیار آمن و روشی بسیار عالی برای نگهداری آفلاین بیت‌کوین (موسوم به «انباره‌ی سرد») است. در جدول ۴-۵ یک کلید خصوصی WIF و کلید ۳۸-BIP معادل آن را (که رمزگذاری آن با استفاده از امکانات سایت bitaddress.org انجام شده) مشاهده می‌کنید.

جدول ۴-۵ مثال: کلید خصوصی رمزگذاری شده با استاندارد ۳۸-BIP

فرمت کلید	مقدار کلید
WIF	5J3mBbAH58CpQ3Y5RNJpUKPE62SQ5tfcvU2JpbnkeyhfsYB1Jcn
گذر جمله	MyTestPassphrase
BIP-38	6PRTHL6mWa48xSopbU1cKrVjpKbBZxcLRRCdctLJ3z5yxE87MobKoXdtSJ

آدرس P2SH و آدرس چندامضایی

همان طور که قبلاً گفتیم، آدرس‌های سنتی بیت‌کوین با عدد «۱» شروع شده از کلیدهای عمومی (که خود از کلیدهای خصوصی مشتق شده‌اند) استخراج می‌شوند. با آن که هر کسی می‌تواند یک آدرس بیت‌کوین جعلی که با «۱» شروع می‌شود، بسازد ولی این بیت‌کوین فقط زمانی قابل خرج کردن است که آدرس آن همراه با امضای کلید خصوصی و درهم کلید عمومی متناظر باشد.

آدرس‌های بیت‌کوین که با عدد «۳» شروع می‌شوند، آدرس پرداخت-به-درهم اسکرپت (P2SH) هستند، که گاه به اشتباه آدرس چندامضایی گفته می‌شوند. این آدرس‌ها از گیرنده‌ی (ذینفع) تراکنش بیت‌کوین، به جای کلید عمومی مالک بیت‌کوین، به عنوان درهم یک اسکرپت نام می‌برند. این ویژگی اولین در سال ۲۰۱۲ با استاندارد ۱۶-BIP معرفی شد (پیوست‌های کتاب را ببینید) و به سرعت مورد قبول قرار گرفت، چون فرصت اضافه کردن کارکرد به خود آدرس بیت‌کوین را ایجاد می‌کند. بر خلاف تراکنش‌هایی که مبالغ بیت‌کوین را به آدرس‌های سنتی بیت‌کوین (که با «۱» شروع می‌شوند) حوله می‌کنند، تراکنش‌هایی که بانام پرداخت-به-درهم-کلید-عمومی (P2PKH) نیز شناخته می‌شوند، مبالغی که به آدرس‌های P2SH (که با «۳» شروع می‌شوند) فرستاده شده‌اند، برای اثبات مالکیت به چیزی بیش از یک درهم کلید عمومی و یک امضای کلید خصوصی نیاز دارند. این الزامات در زمان تولید آدرس و در داخل اسکرپت مشخص می‌شود، و تمامی درودی‌های این آدرس باید این الزامات را برابرده کنند.

آدرس P2SH از یک اسکرپت تراکنش ساخته می‌شود، که می‌گوید چه کسی می‌تواند یک خروجی تراکنش را خرج کند (در فصل ۷ بیشتر درباره‌ی P2SH صحبت خواهیم کرد). در کُدگذاری یک آدرس P2SH از همان تابع درهم-دوگانه (که در فرآیند تولید آدرس‌های بیت‌کوین دیدیم) استفاده می‌شود، فقط در اینجا ورودی این تابع (به جای کلید عمومی) یک اسکرپت است:

```
script hash = RIPEMD160(SHA256(script))
```

«درهم اسکریپت» حاصل سپس با فرمت Base58Check (با پیشوند ویرایش «5») گذاری می‌شود، که نتیجه‌ی آن آدرس است که با عدد «۳» خواهد شد. در مثال زیر چگونگی تولید یک آدرس P2SH از یک اسکریپت با استفاده از فرمان‌های base58check-encode، script-encode، sha256، ripemd160 کاوشگر بیت‌کوین (bx) نشان داده شده است:

```
$ echo dup hash160 [ 89abcdefabbaabbaabbaabbaabbaabba ] equalverify
checksig > script
$ bx script-encode < script | bx sha256 | bx ripemd160 | bx base58check-encode
--version 5
3F616KwKevjR7AsAd4te2YB2zZyASEm1HM
```

P2SH لزوماً همیشه معادل تراکنش استاندارد چندامضایی نیست. یک آدرس P2SH در اکثر موقعیت‌ها نشان‌دهنده‌ی یک اسکریپت چندامضایی است. ولی می‌تواند حاصل گذاری دیگر انواع تراکنش نیز باشد.



آدرس چندامضایی و P2SH

در حال حاضر، اسکریپت آدرس چندامضایی (multisignature address script) رایج‌ترین پاده‌سازی تابع P2SH است. همان‌طور که از نام آن بر می‌آید، اسکریپت اولیه برای اثبات مالکیت (و در نتیجه خرج کردن مبلغ بیت‌کوین) به بیش از یک امضانیاز دارد. ویژگی چندامضایی بیت‌کوین طوری طراحی شده که از مجموع N کلید به M/امضا (که به آن «آستانه» نیز گفته می‌شود) نیاز دارد، به همین دلیل به آن «چندامضایی M-از-N» (که در آن M کوچکتر یا مساوی N است) می‌گویند. برای مثال، باب (صاحب معازه‌ی قهوه‌فروشی) که در فصل ۱ با او آشنا شدیم) می‌تواند از یک آدرس چندامضایی ۱-از-۲ که امضای آن از میان دو کلید متعلق به خودش و همسرش انتخاب می‌شود، استفاده کند تا هر دو بتوانند تراکنش‌های قفل شده با این آدرس را امضاء و خرج کنند. این در واقع مشابه یک «حساب بانکی مشترک» است که چند نفر حق امضاء برداشت از آن حساب را دارند. یا ممکن است گوپش (طراحی وب‌هندی که قرار است برای سابت معازه‌ی باب را طراحی کند) با دونفر دیگر شریک بوده و یک آدرس چندامضایی ۲-از-۳ داشته باشد، به این معنا که تراکنش‌های قفل شده با آدرس آنها را فقط با دو امضای از سه امضای این شرکا می‌توان امضاء و خرج کرد. در فصل ۶ درباره چگونگی ایجاد تراکنش برای خرج کردن بیت‌کوین از آدرس‌های P2SH (و چندامضایی) بیشتر صحبت خواهیم کرد.

آدرس ویژه

آدرس ویژه (vanity addresses) یک آدرس معتبر بیت‌کوین است که در آن می‌توان کلمات یا عبارات معناداری را تشخیص داد. برای مثال، یک آدرس 1LoveBPzzD72PUXLzCKYAtGFYmK5vYNR33 یک آدرس بیت‌کوین معتبر است که چهار کاراکتر ابتدایی آن (البته بعد از عدد «۱») که تمام آدرس‌های بیت‌کوین با آن شروع می‌شوند) کلمه‌ی آشنا «Love» است. برای تولید یک آدرس ویژه گاه لازم است میلیاردها میلیارد کلید خصوصی تولید و آزمایش شوند، ناگزیر است که یک آدرس ویژه با الگوی کلمات مورد نظر ظاهر شود. اگر چه الگوریتم‌های تولید آدرس ویژه از بهینه‌سازی‌های متعددی استفاده می‌کنند، ولی اساساً تفاوت چندانی با فرآیند تولید آدرس‌های معمولی بیت‌کوین (انتخاب تصادفی یک کلید خصوصی، استخراج کلید عمومی از این کلید خصوصی، و استخراج آدرس بیت‌کوین از آن کلید عمومی) ندارند. همین که یک آدرس ویژه یافته شد، خرج کردن بیت‌کوین با این آدرس هیچ تفاوتی با آدرس‌های معمولی نخواهد داشت امنیت آدرس‌های ویژه نه بیشتر از آدرس‌های دیگر است، نه کمتر از آنها، چون این آدرس‌ها هم با همان الگوریتم‌های رمزگاری

منحنی پیضوی (ECC) و SHA تولید می‌شوند. شناس شکستن یک آدرس ویژه و رسیدن به کلید خصوصی متناظر با آن همان اندازه‌ی آدرس‌های دیگر است.

در فصل ۱ با اوژنی، مدیر یک مؤسسهٔ خیریه کودکان در فیلیپین، آشنا شدیم که قصد دارد می‌خواهد از بیت‌کوین برای جمع‌آوری کمک‌های خیریه استفاده کند. برای این منظور، اوژنی به یک آدرس بیت‌کوین نیاز دارد که معکس‌کنندهٔ اهداف مؤسسهٔ متبوع‌وی باشد، مثلاً آدرسی که با «1Kids» شروع شود. اجازه دهید بیینیم اوژنی چطور می‌تواند چنین آدرسی تولید کند و این کار چه تاثیری بر امنیت مبادلات مالی مؤسسهٔ وی دارد.

تولید آدرس ویژه

از آنجاکه در الفبای فرمت Base58 از ۵۸ نماد مختلف استفاده می‌شود، در این بازه چیزی حدود ۵۸^{۲۹} (نحویاً ۱۰^{۱۴}) آدرس وجود دارد که همگی با «1Kids» شروع می‌شوند؛ جدول ۶-۴ را بینید.

جدول ۶-۴ بازه‌ی آدرس‌های ویژه که با «I Kids» شروع می‌شوند

اجازه دهید الگوی «1Kids» را به عنوان یک عدد فرض کرده و بینیم احتمال یافتن این الگو در یک آدرس بیت‌کوین چقدر است؛ فراوانی ترکیب‌های مختلف الگوی «1Kids» در جدول ۴-۷ نشان داده شده است. یک کامپیوتر شخصی معمولی (بدون هیچ سخت‌افزار خاصی) می‌تواند در هر ثانیه تقریباً 10^6 هزار کلید را جستجو کند.

جدول ٤-٤ فراوانی یک الگوی ویژه (KidsCharity) و میانگین زمان جستجوی آن با یک کامپیوتر شخصی

طول الكو	فراوانى	ميانگين زمان جستجو
١	١در ٥٨٥ كيليد	<١ ملي ثانية
٢	١در ٣٣٦٤ كيليد	٥٠ ميلي ثانية
٣	١در ١٩٥ هزار	<٢ ثانية
٤	١در ١١ ميليون	١ دقيقه
٥	١در ٦٥٦ ميليون	١ ساعت
٦	١در ٣٨٠ ميليارد	٢ روز
٧	١در ٢٢٠ تريليون	٣ نا٤ ماہ
٨	١در ١٢٨٠ تريليون	١٢ نا١٨ سال
٩	١در ٧٠ هزار تريليون	٨٠٠ سال
١٠	١در ٤٠٠ هزار تريليون	٤٦ هزار سال
١١	١در ٢٣٠ ميليون تريليون	٢٥ ميليون سال

همان طور که می بینید، اوژنی حتی با در اختیار داشتن صد هزار کامپیوتر، هیچ شناسنامه برای تولید آدرس ویژه «KidsCharity» ندارد. هر کاراکتری که به الگوی مطلوب یک آدرس ویژه اضافه کنید، زمان (دشواری) جستجوی آن را ۵۸۵ کامپیوتر هایی که امروزه برای تولید آدرس های ویژه به کار گرفته می شوند، اغلب کامپیوتر هایی هستند که در گذشته ای نه چنان دور برای استخراج بیت کوین به کار می رفته اند ولی اکنون دیگر برای این منظور مقرن به صرفه نیستند. جستجوی آدرس های ویژه باسیستم های مبتنی بر GPU صد ها بار سریع تر از سیستم های معمولی (کامپیوتر های همه-منظوره مجهز به CPU) است. روش دیگر برای تهیه یک آدرس ویژه خریدن یا بروزرسانی سپاری عملیات استخراج آن به یک شرکت استخراج آدرس های ویژه، مانند سایت مخزن آدرس ویژه (به آدرس <http://vanitypool.appspot.com>)، است. این شرکت ها از سخت افزار های استخراج بیت کوین خود برای یافتن آدرس های ویژه نیز استفاده کرده و آنها را به بهایی اندک می فروشنند. برای مثال، اوژنی به جای چند ماه انتظار برای یافتن یک آدرس ویژه هفت کاراکتری با کامپیوتر شخصی، می تواند آن را با قیمت ناجیز^۱، بیت کوین (نحوی ۱ میلیون تومان در زمان چاپ این کتاب) بخرد.

تولید آدرس های ویژه اساساً یک جستجوی زور-ورزانه (brute-force) است: یک کلید خصوصی تصادفی انتخاب کنید، ببینید آیا آدرس بیت کوین تولید شده از این کلید با الگوی مطلوب منطبق است یا خیر، و این کار را تا رسیدن به آدرس دلخواه دنبال کنید. در مثال ۹-۴ یک برنامه کوچک به زبان C++ به نام vanity-miner می بینید که با همین روش یک آدرس ویژه با الگوی چهار کاراکتری را جستجو می کند. در این برنامه از کتابخانه libbitcoin (که در قسمت های قبلی همین فصل معرفی کردیم) استفاده کرده ایم.

مثال ۹-۴ برنامه vanity-miner: استخراج آدرس ویژه

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

// The string we are searching for
const std::string search = "1kid";

// Generate a random secret key. A random 32 bytes.
bc::ec_secret random_secret(std::default_random_engine& engine);
// Extract the Bitcoin address from an EC secret.
std::string bitcoin_address(const bc::ec_secret& secret);
// Case insensitive comparison with the search string.
bool match_found(const std::string& address);

int main()
{
    // random_device on Linux uses "/dev/urandom"
    // CAUTION: Depending on implementation this RNG may not be secure enough!
    // Do not use vanity keys generated by this example in production
    std::random_device random;
    std::default_random_engine engine(random());

    // Loop continuously...
    while (true)
    {
        std::string address = bitcoin_address(random_secret(engine));
        if (match_found(address))
            break;
    }
}
```

```

// Generate a random secret.
bc::ec_secret secret = random_secret(engine);
// Get the address.
std::string address = bitcoin_address(secret);
// Does it match our search string? (1kid)
if (match_found(address))
{
    // Success!
    std::cout << "Found vanity address! " << address << std::endl;
    std::cout << "Secret: " << bc::encode_base16(secret) << std::endl;
    return 0;
}
// Should never reach here!
return 0;
}

bc::ec_secret random_secret(std::default_random_engine& engine)
{
    // Create new secret...
    bc::ec_secret secret;
    // Iterate through every byte setting a random value...
    for (uint8_t& byte: secret)
        byte = engine() % std::numeric_limits<uint8_t>::max();
    // Return result.
    return secret;
}

std::string bitcoin_address(const bc::ec_secret& secret)
{
    // Convert secret to payment address
    bc::wallet::ec_private private_key(secret);
    bc::wallet::payment_address payaddr(private_key);
    // Return encoded form.
    return payaddr.encoded();
}

bool match_found(const std::string& address)
{
    auto addr_it = address.begin();
    // Loop through the search string comparing it to the lower case
    // character of the supplied address.
    for (auto it = search.begin(); it != search.end(); ++it, ++addr_it)
        if (*it != std::tolower(*addr_it))
            return false;
    // Reached end of search string, so address matches.
    return true;
}

```

مثال ۴-۹ از تابع `std::random_device` استفاده می‌کند. بسته به پیاده‌سازی آن، این تابع می‌تواند یک «مولد عدد تصادفی آمن شده با رمزنگاری» (CSRNG) ارائه شده توسط سیستم عامل است. در سیستم‌های شبه-یونیکس (مانند لینوکس) این تابع از ماژول `/dev/urandom` مشتق می‌شود. تابع مولد عدد تصادفی به کار رفته در این مثال فقط برای اهداف آموزشی است: برای تولید کلیدهای بیت‌کوین در برنامه‌های حرفه‌ای باید از یک مولد عدد تصادفی با امنیت کافی استفاده کنید.

مثال ۴-۱۰ چگونگی کامپایل و چند بار اجرای برنامه‌ی `vanity-miner` و خروجی‌های آن را مشاهده می‌کنید. در اجرای سوم، زمان اجرای برنامه توسط ماژول `time` نیز اندازه‌گیری شده است. [برای کامپایل و اجرای این برنامه کتابخانه‌های C++ و libbitcoin باید از قبل در سیستم شما نصب شده باشند.]

مثال ۴-۱۰-۱ کامپایل و اجرای برنامه‌ی `vanity-miner`

```
$ # Compile the code with g++
$ g++ -o vanity-miner vanity-miner.cpp $(pkg-config --cflags --libs libbitcoin)
$ # Run the example
$ ./vanity-miner
Found vanity address! 1KiDzKG4MxmovZryZRj8tK81oQRhbZ46YT
Secret: 57cc268a05f83a23ac9d930bc8565bac4e277055f4794cbd1a39e5e71c038f3f
$ # Run it again for a different result
$ ./vanity-miner
Found vanity address! 1Kidxr3wsmMzzouwXibKfwTYs5Pau8TUFn
Secret: 7f65bbbbe6d8caaee74a0c6a0d2d7b5c6663d71b60337299a1a2cf34c04b2a623
$ # Use "time" to see how long it takes to find a result
$ time ./vanity-miner
Found vanity address! 1KidPWhKgGRQWD5PP5TAnGfDyfWp5yceXM
Secret: 2a802e7a53d8aa237cd059377b616d2bfcfa4b0140bc85fa008f2d3d4b225349
real    0m8.868s
user    0m8.828s
sys     0m0.035s
```

این برنامه الگوهای چهار کاراکتری «1Kids» را (بدون حساسیت به نوع حروف) در چند ثانیه پیدا می‌کند. برای جستجوی الگوهای دیگر متغیر `search` را تغییر داده و زمان آن را با فرمان `time` اندازه‌گیرید.

امنیت آدرس‌های ویژه

آدرس‌های ویژه هم می‌توانند برای تقویت تمہیدات امنیتی به کار گرفته شوند، هم برای شکست آنها؛ آدرس ویژه‌در حقیقت یک شمشیر دولبه است. آدرس‌های ویژه می‌توانند با آسان کردن شناسایی آدرس‌های بیت‌کوین یک فردیا سازمان برای مشتریان، و دشوار کردن جایگزینی آنها با آدرس رقبا و سارقان (و گول زدن کاربران ساده با این آدرس‌ها)، امنیت را تقویت کنند. اما متأسفانه، آدرس‌های ویژه همچنین می‌توانند باعث تضعیف امنیت شوند، چون جعل کردن آدرس‌هایی که شبیه آدرس‌های ویژه باشند (و کاربران ساده گول آنها را بخورند)، آسان‌تر است.

اوژنی می‌تواند یک آدرس بیت‌کوین تصادفی (مثلاً `1J7mdg5rbQyUHENYdx39WVWK7fsLpEoXZy`) برای خود تولید کرده و از مردم بخواهد کمک‌های خیریه‌ی خود را به این آدرس حواله کنند؛ یا این که می‌تواند یک آدرس ویژه (مثلاً آدرسی که با عبارت «`1Kids`» شروع می‌شود) برای خود دست و پا کرده و شناسایی مؤسسه‌ی خود را برای افراد خیر ساده‌تر کند.

در هر دو حالت، خطر استفاده از یک آدرس ثابت واحد (به جای یک آدرس تصادفی به ازای هر اهداکننده) این است که سارقان می‌توانند به سایت این مؤسسه‌ی خیریه نفوذ کرده و آدرس ثابت آن را با آدرس خود عوض کنند، و پول‌های اهدایی را به چیزی بزنند. افرادی که می‌خواهند به این مؤسسه کمک مالی کنند، معمولاً آدرس بیت‌کوین آن را از طریق بروشورهای چاپی، ایمیل، سایت خود مؤسسه، یا سایت‌های دیگری که آگهی آن را منتشر کرده‌اند، به دست می‌آورند. اگر این مؤسسه از یک آدرس تصادفی مثل `1J7mdg5rbQyUHENYdx39WVWK7fsLpEoXZy` استفاده کرده باشد، اغلب کاربران (اهداکنندگان) قبل از واریز کردن پول، با مراجعه به سایت مؤسسه فقط چند کاراکتر اول آن (مثلاً `1J7mdg5`) را چک می‌کنند تا خیالشان راحت شود که این آدرس واقعی و معتبر است. در این وضعیت، سارقان می‌توانند با یک برنامه‌ی تولید آدرس‌های ویژه (مانند آنچه در مثال ۴-۹ دیدید) به سادگی آدرس‌هایی تولید کنند که چند کاراکتر ابتدایی آنها شبیه آدرس تصادفی مؤسسه باشد؛ جدول ۴-۴ را بینید.

جدول ۴-۴ تولید آدرس‌های ویژه مشابه یک آدرس تصادفی

آدرس	آدرس تصادفی اصلی
<code>1J7mdg5rbQyUHENYdx39WVWK7fsLpEoXZy</code>	آدرس ویژه (۴-کاراکتر مشابه)
<code>1J7md10qU4LpctBetHS2ZoyLV5d6dShhEy</code>	آدرس ویژه (۵-کاراکتر مشابه)
<code>1J7mdgYqyNd4ya3UEcq31Q7sqRMXw2XZ6n</code>	آدرس ویژه (۶-کاراکتر مشابه)
<code>1J7mdg5WxGENmwyJP9xuGhG5KRzu99BBCX</code>	

بس، آیا آدرس‌های ویژه واقعاً امنیت را بالا می‌برند؟ اگر اوژنی از آدرس ویژه `1Kids33q44erFfpeXrmDSz7zEqG2FesZEN` استفاده کند، کاربران (اهداکنندگان) به احتمال زیاد علاوه بر «الگوی ویژه» (عبارت «`1Kids`») به چند کاراکتر بعد از آن هم توجه می‌کنند، و مثلاً عبارت «`1Kids33`» را در نظر می‌گیرند. این باعث می‌شود تا کار سارقان برای جعل این آدرس نسبت به «الگوی ویژه» ۴-کاراکتری (`1Kids`) (۳۳۶۴ × ۵۸ = ۲۳۶۴) مرتبه دشوارتر شود. به بیان دیگر، هر قدر تلاش بیشتری صرف تولید یک آدرس ویژه با الگوی طولانی تر کنید (یا پول بیشتری برای خرید یک آدرس ویژه با الگوی طولانی تر پردازید)، مانع بلندتری بر سر راه سارقان ایجاد کرده‌اید. در واقع، اگر اوژنی پول کافی برای خرید یک آدرس ویژه ۸-کاراکتری خرج کند، سارقان مجبور می‌شوند وارد قلمروی آدرس‌های ویژه ۱۰-کاراکتری شوند، قلمرویی که برای کامپیوترهای شخصی معمولی به کلی غیرقابل دسترس، و حتی برای بهترین سخت‌افزارهای استخراج بیت‌کوین یا آدرس ویژه نیز بسیار پُرهزینه و زمان بر است. آنچه با پولی اندک برای اوژنی قابل وصول است، به روایی دست‌نیافتنی برای سارقان تبدیل می‌شود؛ به خصوص که آنچه از این راه به می‌آورند، به هیچ وجه برای پوشش دادن هزینه‌ی تولید آدرس‌های ویژه کافی نخواهد بود.

کیف‌پول کاغذی

اگر کلیدهای خصوصی را روی کاغذ چاپ کنید، یک کیف‌پول کاغذی (paper wallet) خواهید داشت. در یک کیف‌پول کاغذی همچنین اغلب آدرس بیت‌کوین متاظر با آن کلید خصوصی نیز چاپ می‌شود، ولی این الزامی نیست چون آدرس بیت‌کوین را همیشه می‌توان از این کلید خصوصی استخراج کرد. کیف‌پول کاغذی وسیله‌ای بسیار کارآمد برای ذخیره کردن

بیت‌کوین به صورت آفلاین است، و به آن ابزارهای سرد (cold storage) هم گفته می‌شود. وقتی از کلیدهای خصوصی خود روی کاغذ نسخه‌ی پشتیبان می‌گیرید، آنها را در مقابل خرابی دستگاه‌های الکترونیکی، سرقت توسط قوژگران، و همچنین حذف اتفاقی محافظت کرده‌اید. از آنجاکه کیف‌پول کاغذی یک ابزار کاملاً آفلاین است، امنیت بسیار خوبی در مقابل انواع تهدیدات کامپیوتری و اینترنتی دارد.

کیف‌پول کاغذی در اشکال، اندازه‌ها و طرح‌های مختلفی وجود دارد، ولی اساساً چیزی نیست جزیک کلید خصوصی و آدرس بیت‌کوین متناظر با آن که روی یک تکه کاغذ چاپ شده‌اند. جدول ۹-۴ ساده‌ترین شکل یک کیف‌پول کاغذی را نشان می‌دهد.

جدول ۹-۴ ساده‌ترین کیف‌پول کاغذی: چاپ آدرس بیت‌کوین و کلید خصوصی متناظر با آن روی کاغذ

کلید خصوصی (WIF)

آدرس بیت‌کوین

5J3mBbAH58CpQ3Y5RNJpUKPE62SQ5tfcvU2JpbnkeyhfsYB1Jcn 1424C2F4bC9J1dNjjTUZCbUxv6Sa1Mt62x

تولید کیف‌پول کاغذی با ابزارهایی مانند مولد جواوسکریپت سمت مشتری <https://www.bitaddress.org> بسیار ساده است. این صفحه تمامی کد لازم برای تولید کلید و کیف‌پول را (حتی وقتی به طور کامل از اینترنت قطع هستید) در بر دارد. برای استفاده از این برنامه، فقط کافی است صفحه‌ی HTML این سایت را در کامپیوتر خود (یاروی فلاش USB) ذخیره کنید، سپس کامپیوتر را از اینترنت قطع کرده، و این فایل HTTP را در مرورگر وب باز کنید. [یا از آن آمن‌تر، این صفحه‌ی HTTP را روی کامپیوتری که بادیسک بوت یک سیستم عامل متفاوت (مثل لینوکس) راهاندازی شده، باز کنید]. پس از تولید کلید و آدرس بیت‌کوین با این برنامه‌ی جواوسکریپت، می‌توانید آن را روی کاغذ چاپ کنید. این بیت‌کوین‌های کاغذی را می‌توان به دیگران داد و یا مثل پول نقد خرج کرد. در شکل ۸-۴ تصویری از یک کیف‌پول کاغذی تولید شده در سایت bitaddress.org را مشاهده می‌کنید.

عیب اصلی این کیف‌پول‌های چاپی ساده آن است که به سادگی می‌توان آنها را سرقت کرد. در واقع حتی نیازی به سرقت نسخه‌ی چاپی هم نیست و فقط کافی است سارقین از آنها عکس بگیرند تا بتوانند کنترل بیت‌کوین‌های قفل شده با این کلیدها را در اختیار داشته باشند. کیف‌پول‌های کاغذی پیشرفته‌تر از کلیدهای خصوصی رمزنگاری شده‌ی BIP-38 استفاده می‌کنند. همان‌طور که می‌دانید، کلیدهای خصوصی BIP-38 با یک گذر جمله (که کاربر به خاطر می‌سپارد) محافظت می‌شوند. مزیت کیف‌پول کاغذی با رمزنگاری BIP-38 بر کیف‌پول‌های آنلاین (که برای دسترسی به آنها هم گذرواژه نیاز است) این است که گذرواژه‌ی این نوع از کیف‌پول در جایی آمن (یک گاوصندوق یا مغازه کاربر) قرار دارد و هرگز روی اینترنت قرار نمی‌گیرد. در شکل ۹-۴ تصویری از یک کیف‌پول کاغذی رمزنگاری شده (BIP-38) از سایت bitaddress.org را می‌بینید.



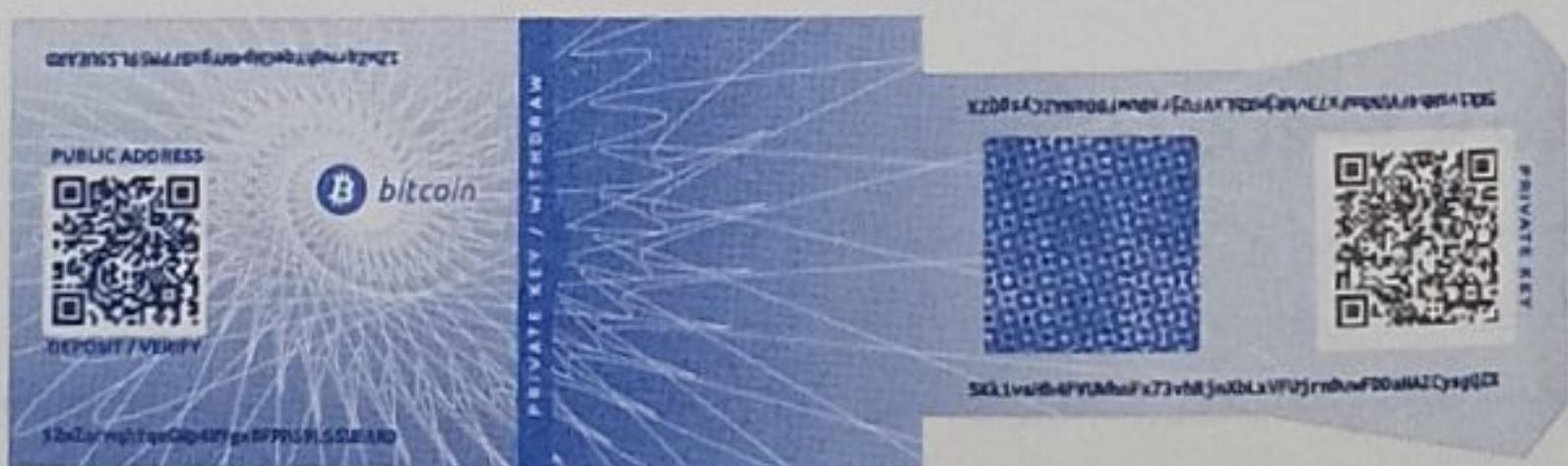
شکل ۸-۴ نمونه‌ی کیف‌پول کاغذی تولید شده در سایت bitaddress.org.



شکل ۹-۴ نمونه‌ی کیف‌پول کاغذی رمزنگاری شده (BIP-38) با گذر جمله‌ی «test» از سایت bitaddress.org

بزرگترین عیب کیف‌پول کاغذی این است که تمامی مبلغ آن را باید در یک تراکنش خرج کرد، چون اگر مبلغ پرداخت از مبلغ این کیف‌پول کاغذی کمتر باشد، تراکنش خرج کردن این بیت‌کوین یک آدرس تتمه تولید خواهد کرد. علاوه بر آن، اگر کامپیوتری که از آن برای امضای کردن این تراکنش استفاده می‌کنید، مورد حمله قرار گیرد، ممکن است این کلید خصوصی در معرض فاش شدن قرار گیرد؛ با خرج کردن کل مبلغ کیف‌پول کاغذی در یک تراکنش واحد، خطر لورفتن این کلید خصوصی منتفی می‌شود. اگر می‌خواهید فقط بخشی از یک کیف‌پول کاغذی را خرج کنید، تتمه‌ی این تراکنش را بلافاصله به یک کیف‌پول کاغذی دیگر منتقل کنید.

کیف‌پول‌های کاغذی در اندازه‌ها و طرح‌های گوناگون، و با ویژگی‌های مختلف عرضه می‌شوند. برخی از این کیف‌پول‌ها با طرح‌های گرافیکی فانتزی یا مناسبتی (مثل عید، روز تولد، سالگرد ازدواج، وغیره)، و برخی دیگر به صورت اوراق قرضه‌ی قابل نگهداری در گاوصدوق شخصی یا صندوق امانات بانک (با پوشش آلومینیمی خراشیدنی، یادربسته‌بندی مهر و موم شده) ارائه شده‌اند. در شکل‌های ۱۰-۴ تا ۱۲-۴ چندین نمونه از این کیف‌پول‌ها مشاهده می‌کنید.



شکل ۱۰-۴ یک نمونه کیف‌پول کاغذی با کلید خصوصی تاشو از سایت bitcoinpaperwallet.com



شکل ۱۱-۴ نمونه‌ی تاشده‌ی کیف‌پول کاغذی از سایت bitcoinpaperwallet.com



شکل ۱۲-۴ یک نمونه کیف‌پول کاغذی با چند کلید و آدرس اضافی.

کیف‌پول کاغذی شکل ۱۲-۴ که دارای تعدادی کلید و آدرس یدکی است، به خصوص برای نگهداری بیت‌کوین در مکان‌های مختلف و حفظ آنها از خطرات احتمالی (مثل، سیل، آتش‌سوزی، یا بلاجای طیعی) ایده‌آل است.