گزارش کار

HW2

دانش عبداللهی

9723053

- فایل server.cpp/h

در Constructor کلاس Server ، با هر بار تعریف کردن یک Object از کلاس Server اجرا می-شود و مقادیر متغیر clients را Null Pointer و صفر می‌گذارد.

- add_client

این تابع ، به عنوان ورودی ID کاربر جدید را می‌گیرد و یک اشاره‌گر به کلاس Client به عنوان خروجی می‌دهد.

اگر ID داده شده به این تابع ، تکراری باشد (یعنی یک کاربر با این ID در سرور وجود داشته باشد)

تابع یک عدد 4 رقمی رندوم تولید می‌کند و به انتهای ID داده شده اضافه می‌کند و کاربر را با ID جدید در سرور ثبت می‌کند. برای انجام این امر ، در ابتدای تابع ، با یک for loop ، ID های موجود در سرور را با ID ورودی مطابقت می‌دهیم.

هم‌چنین این تابع در هنگام ثبت کاربر جدید ، به صورت پیش‌فرض 5 عدد سکه ، به کیف پول کاربر جدید اضافه می‌کند.

- get_client

این تابع ID را به عنوان ورودی می‌گیرد و با سرچ کردن در متغیر clients ، اگر ID در سرور وجود داشت ، اشاره گر به آن کاربر مخصوص را برمی‌گرداند. اگر هم ID ورودی در سرور ثبت نشده بود،

یک **NULL Pointer** به عنوان خروجی می‌دهد.

- **get_wallet**

این تابع **ID** را به عنوان ورودی می‌گیرد و با سرچ کردن در متغیر **clients** ، اگر **ID** در سرور وجود داشت ، مقدار پول کیف پول مخصوص به کاربر را برمی‌گرداند. اگر هم **ID** ورودی در سرور ثبت نشده بود ، با ارور **logic_error** مواجه می‌شویم.

- **parse_trx**

این تابع یک رشته ورودی به عنوان **Transaction** و رفرنس‌هایی از رشته‌های **Sender** و **Receiver** و هم‌چنین رفرنس از یک عدد **double** را به می‌گیرد و با قسمت‌بندی رشتهٔ **Transaction** ، مقادیر **Sender** ، **Receiver** و **value** را مشخص می‌کند.

هم‌چنین این تابع بررسی می‌کند که فرم رشتهٔ **Transaction** استاندارد است یا نه . این کار را با شماردن '-' ، در رشتهٔ **Transaction** انجام می‌دهد. اگر تعداد کاراکتر '-' برابر با **2** نبود ، یعنی فرم رشتهٔ **Transaction** استاندار نیست و تابع **false** را برمی‌گرداند. و اگر فرم رشته استاندارد بود ، تابع **true** را برمی‌گرداند.

برای قسمت بندی رشتهٔ **Transaction** ، اندیس کاراکترهای '-' را در رشته با تابع **()find** پیدا می‌کنیم. **ID** ارسال کننده از اولین کاراکتر رشته شروع می‌شود و طول آن برابر با اندیس اولین کاراکتر '-' است. (از تابع **()substr** استفاده کردیم).**ID** دریافت‌کننده از اولین کاراکتر بعد از اولین '-' شروع می‌شود و طول آن برابر با تفاضل دومین و اولین اندیس کاراکتز '-' منهای یک است.

مقدار **Value** هم از کاراکتر به از دومین '-' شروع می‌شود و تا آخر رشته ادامه دارد. همچنین برای ذخیرهٔ **Value** در یک متغیر **double** ، از تابع **()stod** استفاده کردیم.

- **add_pending_trx**

این تابع رشتهٔ **Transaction** و امضای دیجیتال آن را به عنوان ورودی می‌گیرد. ابتدا با استفاده از تابع **()parse_trx** ، **ID** ارسال کننده ، دریافت کننده و مقدار پول جابه‌جایی را مشخص می‌کند. سپس با استفاده از **ID** ارسال‌کننده ، پول کیف‌پول او و **public_key** او را با استفاده از توابع

get_wallet() و **get_publickey()** می‌گیرد و سپس با استفاده از تابع **crypto::verifySignature** ، **Transaction** را احرازهویت می‌کند و همچنین چک می‌کند که مقدار پول ارسالی از مقدار دارایی کیف‌پول ارسال کننده بیشتر نباشد.

اگر هر کدام از مراحل بالا درست نباشد ، تابع **false** را برمی‌گرداند در غیر این صورت **true** برمی‌گرداند.

- **mine()**

این تابع ، ابتدا از تمام کاربرهای سرور یه عدد رندوم می‌گیرد و با استفاده از **mempool** ساخته شده و این عدد رندوم، یک **Hash sha256** تولید می‌کند. حال اگر در **10** کاراکتر اول این رشتهٔ تولیدی ، **3** تا صفر پشت‌سرهم وجود داشته باشند ، بلوک **Mine** می‌شود و تمام **Transaction**ها انجام می‌شودو هم‌چنین به کاربر ماینر **6.25** سکه پاداش داده می‌شود.

- در فایل **server.h** ، بردار **pending_trx** را به صورت **inline** تعریف می‌کنیم که تا آخر اجرای برنامه وجود داشته باشد.

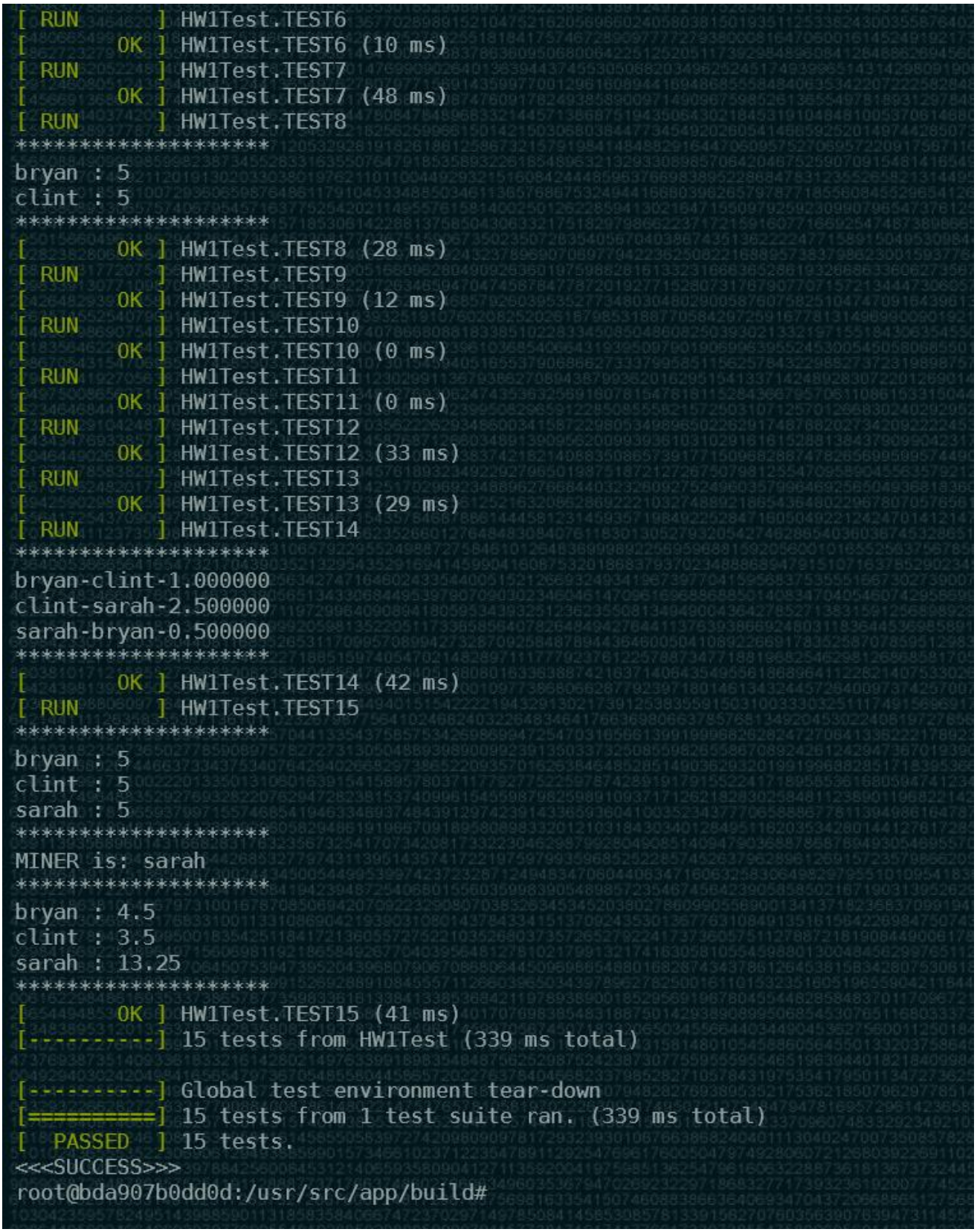
- فایل **client.cpp/h**

در **Constructor** کلاس **Client** ، با گرفتن **ID** کاربر و سرور آن ، متغیرهای کلاس **Client** را مقدار-دهی می‌کنیم و همچنین یک **public_key** و **private_key** با استفاده از تابع **crypto::generate_key()** برای کاربر جدید می‌سازیم.

- سایر توابع این کلاس دقیقا طبق توضیحات سوال نوشته شده‌اند و پیچیدگی خاصی ندارند. (توضیحات لازم در کد ، کامنت شده‌اند.)

در تابع **generate_nonce()** ، یک عدد در بازهٔ **0** تا **1000** تولید می‌شود.

تصویر اجرای تست‌ها :



- لینک git