علی آراسته ۹۶۱۰۱۱۶۵

مبانی فناوری بلاکچین و رمزارزها – گزارش تمرین عملی سری اول

در این تمرین از Sha-256 به عنوان تابع چکیده‌ساز (Hash Function) استفاده می‌کنیم.

برای این کار از پکیج hashlib کمک می‌گیریم. در این پکیج، کلاسی به اسم sha256 وجود دارد که نیاز ما را بر طرف می‌کند. نحوه استفاده از این کلاس بدین صورت است که در هنگام نمونه‌سازی از آن، پیامی را که قرار است چکیده شود، به صورت bytes به عنوان پارامتر ورودی به سازنده کلاس می‌دهیم. همچنین برای اضافه‌کردن متنی به پیام، می‌توان از تابع update استفاده کرد. برای گرفتن hash پیام، از تابع hexdigest بهره می‌گیریم. این تابع چکیده پیامی را که تاکنون ذخیره شده است، در مبنای hex محاسبه می‌کند و به صورت یک string تحویل می‌دهد.

سوال ۱:

A) در این سوال برای یافتن t، از حمله جستجوی فراگیر کمک می‌گیریم. برای این کار ابتدا تابع get_random_string را تعریف می‌کنیم. این تابع ابتدا به طور تصادفی یک عدد صحیح بین ۱ تا ۱۰ انتخاب و سپس stringای به طول عدد انتخاب شده تولید می‌کند. برای تولید string، هر بار به طور تصادفی یک کاراکتر از بین حروف کوچک و بزرگ و اعداد انگلیسی انتخاب می‌نماید.

سپس با استفاده از یک حلقه while با شرط True، آنقدر stringهای تصادفی تولید کرده، از آن‌ها hash می‌گیریم تا stringای پیدا کنیم که مقدار hash آن با مقدار hash شماره دانشجویی مدنظر برابر باشد.

در آخر پس از پیدا کردن t، فایلی با نام x.txt تولید می‌کنیم و شماره دانشجویی و t را به ترتیب در خط اول و دوم آن می‌نویسیم.

برای شماره دانشجویی ۹۶۱۰۱۱۶۵، یکی از tهای ممکن eCil6fWi است. شایان ذکر است بعد از هر بار اجرای الگوریتم، t جدیدی پیدا می‌شود که در شرایط خواسته شده صدق می‌کند.

B) ابتدا توضیح کوتاهی در مورد مسئله یا پارادوکس روز تولد ارائه می‌کنیم. صورت مسئله بدیت صورت است که در مجموعه‌ای از n نفر که به طور تصادفی انتخاب شده اند، احتمال اینکه حداقل دو نفر در یک روز یکسان متولد شده باشند، چقدر است؟ (فرض کنید احتمال به دنیا آمدن در روزهای مختلف سال یکسان است) طبق اصل لانه کبوتر این احتمال به ازای ۳۶۶ نفر برابر ۱ است (فرض کنید سال ۳۶۵ روز است). اما پارادوکس این مسئله است که این احتمال به ازای ۲۳ نفر بیش از ۰/۵ و به ازای ۷۶ نفر بیش از ۰/۹۹۹ است. برای اینکه این نتیجه قابل‌پذیرش باشد باید توجه داشت که در واقع، مقایسه روز تولد بین هر ۲ نفری انجام می‌شود که در حالت ۲۳ نفر، $\frac{23 \times 22}{2} = 253$ حالت ممکن دارد که به خوبی بیشتر از نصف روزهای سال است.

از این مسئله برای کاهش پیچیدگی پیدا کردن تصادم برای یک تابع چکیده‌ساز استفاده می‌شود. در واقع می‌توان ثابت کرد اگر تابعی یک خروجی m بیتی تولید کند و خروجی‌های آن هم‌احتمال باشند (همه خروجی‌های ممکن با احتمال 2^{-m} انتخاب شوند)، با انتخاب $\sqrt{2^m} = 2^{\frac{m}{2}}$ ورودی تصادفی متفاوت و محاسبه خروجی هریک، با احتمال بیش از $\frac{1}{2}$، حداقل دو ورودی متفاوت

با خروجی یکسان وجود خواهد داشت. از آنجایی که تابع چکیده‌ساز مدنظر، ۲۰ بیتی است، با انتخاب $2^{10} = 1024$ ورودی تصادفی متفاوت، پیدا کردن دو ورودی متفاوت با خروجی یکسان محتمل خواهد بود (احتمال آن بیشتر از $\frac{1}{2}$ خواهد بود).

در این سوال برای پیدا کردن تصادم از یک حاقه while با شرط True استفاده می‌کنیم؛ بدین صورت که در هر بار تکرار حلقه، ابتدا یک string تصادفی با استفاده از get_random_string تولید می‌کنیم، سپس در صورتی که در گذشته همچین stringای تولید نشده باشد، مقدار hash آن را محاسبه می‌کنیم و در آخر بررسی می‌کنیم که آیا در گذشته برای string دیگری این مقدار بدست آمده است یا خیر. اگر جواب این سوال مثبت باشد، دو stringای را که مقدار hash آن‌ها یکسان شده است، در خروجی چاپ می‌کنیم.

برای مثال خروجی تابع چکیده‌ساز برای GutBQ و gnPRTe یکسان است. شایان ذکر است بعد از هر بار اجرای الگوریتم، دو string جدید (یک تصادم جدید) پیدا می‌شود و در خروجی چاپ می‌گردد.

C) برای مسئله احراز اصالت روش‌های بسیاری بر حسب شرایط وجود دارد. در شرایط صورت مسئله (وجود کلید مخفی s بین فرستنده و گیرنده) می‌توان از روش زیر استفاده کرد. فرستنده ابتدا کلید مخفی s را به پیام m می‌چسباند و مقدار hash حاصل را محاسبه می‌کند. سپس پیام m را به همراه مقدار hash بدست آمده، برای گیرنده ارسال می‌کند. گیرنده پس از دریافت آنچه فرستنده ارسال کرده است، ابتدا کلید مخفی s را به پیام m می‌چسباند و مقدار hash حاصل را محاسبه می‌کند. سپس اگر این مقدار با مقدار hash ارسالی توسط فرستنده، یکسان باشد، مطمئن می‌شود که پیام واقعا از طرف فرستنده ارسال شده است. این اطمینان از آن جهت است که کلید مخفی s تنها در اختیار فرستنده و گیرنده بوده است و برای دیگران، بدون دانستن کلید مخفی s، امکان ارسال پیام m به همراه مقدار درست وجود ندارد.

در این سوال به صورتی که توضیح داده شد، عمل می‌کنیم. در تابع sender، ابتدا کلید مخفی s به پیام m چسبانده می‌شود و مقدار hash حاصل محاسبه می‌گردد. سپس یک list حاوی دو مقدار، یکی پیام m و دیگری مقدار hash محاسبه شده، برای reciever ارسال می‌شود. در تابع reciever، ابتدا کلید مخفی s به پیام m چسبانده می‌شود و مقدار hash حاصل محاسبه می‌گردد. سپس اگر این مقدار با مقدار hash ارسالی توسط فرستنده، یکسان باشد، پیام m در خروجی چاپ می‌شود. اگر چنین نشود، عبارت !The message has been changed چاپ می‌گردد.

سوال ۲:

برای ساخت درخت Merkle، ابتدا به روش گفته شده در صورت سوال، تعداد برگ‌ها را آنقدر افزایش می‌دهیم تا برابر نزدیک‌ترین توانی از ۲ شود که از تعداد واقعی برگ‌ها بیشتر است. سپس hash تمام برگ‌ها را محاسبه می‌کنیم. در ادامه در هر مرحله hashهایی را که کنار هم هستند، دو تا دو تا به هم می‌چسبانیم، مقدار hash حاصل را محاسبه می‌کنیم و حاصل را جایگزین در مقدار قبلی می‌کنیم. در آخر وقتی به ریشه درخت رسیدیم، مقدار بدست آمده را به عنوان خروجی برمی‌گردانیم.

در این سوال ابتدا تعداد برگ‌ها (n) را از کاربر می‌پرسیم، سپس نزدیک‌ترین توانی از ۲ را که از n بزرگتر است، محاسبه می‌کنیم. در ادامه در یک حلقه for، مقدار i را از ۱ تا توانی از ۲ که در قسمت قبل محاسبه شد، تغییر می‌دهیم. برای $i \leq n$ در هر تکرار یک string از کاربر گرفته می‌شود و به یک list اضافه می‌شود. برای $i > n$ فرآیند padding به صورتی که در قسمت قبل

توضیح داده شد، انجام می‌شود. در ادامه در یک حلقه for دیگر، مقدار i را از توان ۲ که در قسمت قبل محاسبه شد تا صفر، تغییر می‌دهیم و در هر مرحله با توجه به روش توضیح داده شده در قسمت قبل، hashهای مورد نیاز را محاسبه می‌کنیم. در آخر ریشه درخت Merkle را، در خروجی چاپ می‌کنیم.

به عنوان مثال داریم:

Enter number of leaves: 3

Enter first string: a

Enter next string: b

Enter next string: c

7b82f931fab28cb9fa422900a5e9e4fa7d55ef33d4aad339485544f1ed93a91f

سوال ۳:

برای اثبات وجود یک مقدار در درخت Merkle، باید آن مقدار، مکان آن در درخت، hashهایی که در هر مرحله در مسیر آن مقدار از برگ تا ریشه، به hash آن مقدار چسبانده شده است و ریشه درخت در اختیار باشد. در این صورت برای اثبات وجود مقدار مورد نظر، فرآیند طی شده در ساخت درخت Merkle با در نظر گرفتن مکان مقدار مورد نظر (چپ یا راست بودن در فرآیند چسباندن دو hash به یکدیگر)، دوباره طی می‌شود تا ریشه درخت بدست بیاید. اگر این مقدار با مقدار اصلی ریشه درخت برابر باشد، وجود مقدار مورد نظر اثبات می‌شود.

در این سوال ابتدا تعداد برگ‌ها (n) را از کاربر می‌پرسیم، سپس نزدیک‌ترین توانی از ۲ را که از n بزرگتر است، محاسبه می‌کنیم. سپس مکان مقدار مورد نظر در درخت (k) را از کاربر می‌پرسیم. سپس در یک حلقه for، مقدار i را از صفر تا یکی بیشتر از توان ۲ که در قسمت قبل محاسبه شد، تغییر می‌دهیم و ورودی‌های مورد نیاز را از کاربر دریافت می‌کنیم. سپس در یک حلقه for دیگر، مقدار i را از صفر تا توان ۲ که در قسمت قبل محاسبه شد، تغییر می‌دهیم و فرآیند طی شده در ساخت درخت Merkle را با در نظر گرفتن مکان مقدار مورد نظر (چپ یا راست بودن در فرآیند چسباندن دو hash به یکدیگر)، دوباره طی می‌کنیم تا ریشه درخت بدست بیاید. اگر این مقدار با مقدار اصلی ریشه درخت برابر باشد، عبارت Exsited و در غیر این صورت، عبارت Did not exsit! در خروجی چاپ می‌گردد.

به عنوان مثال داریم:

Enter number of leaves: 3

Enter place of message: 1

Enter message: a

Enter first hash value:
3e23e8160039594a33894f6564e1b1348bbd7a0088d42c4acb73eeaed59c009d

Enter next hash value:
272b58b1eba7bdabab033442be84078360506b6fd1fad7b6e0585f1a3739ce4f

Enter Merkle root value:
7b82f931fab28cb9fa422900a5e9e4fa7d55ef33d4aad339485544f1ed93a91f

Exsited