



دانشگاه صنعتی امیرکبیر  
(پلی تکنیک تهران)  
دانشکده مهندسی کامپیوتر و فناوری اطلاعات

پایان نامه کارشناسی  
گرایش فناوری اطلاعات

سیستم تصدیق اصالت غیرمتمرکز مبتنی بر زنجیره بلوک برای اینترنت  
اشیاء

نگارش  
مریم ابراهیم زاده

استاد راهنما  
دکتر بابک صادقیان

استاد داور  
دکتر حمیدرضا شهریاری

شهریور ۱۳۹۸



اینجانب مریم ابراهیم زاده متعهد می‌شوم که مطالب مندرج در این پایان نامه حاصل کار پژوهشی اینجانب تحت نظارت و راهنمایی اساتید دانشگاه صنعتی امیرکبیر بوده و به دستاوردهای دیگران که در این پژوهش از آن‌ها استفاده شده است مطابق مقررات و روال متعارف ارجاع و در فهرست منابع و مآخذ ذکر گردیده است. این پایان نامه قبلاً برای احراز هیچ مدرک هم‌سطح یا بالاتر ارائه نگردیده است.

در صورت اثبات تخلف در هر زمان، مدرک تحصیلی صادر شده توسط دانشگاه از درجه اعتبار ساقط بوده و دانشگاه حق پیگیری قانونی خواهد داشت.

کلیه نتایج و حقوق حاصل از این پایان نامه متعلق به دانشگاه صنعتی امیرکبیر می‌باشد. هرگونه استفاده از نتایج علمی و عملی، واگذاری اطلاعات به دیگران یا چاپ و تکثیر، نسخه‌برداری، ترجمه و اقتباس از این پایان نامه بدون موافقت کتبی دانشگاه صنعتی امیرکبیر ممنوع است. نقل مطالب با ذکر مآخذ بلامانع است.

مریم ابراهیم زاده

امضا

## تقدیر و تشکر

"ویری الذین اوتوا العلم الذی أنزل إلیک من ربّک هو الحق ویهدی إلی صراط العزیز الحمید"

ماحصل آموخته‌هایم را تقدیم می‌کنم به همسر من که تکیه‌گاه رویاهایم است؛ پدر و مادر من که همدم روزهای تلخ و شیرین این راه بوده‌اند؛ خواهرانم که شادی‌بخش لحظه‌های زندگی‌ام می‌باشند و تمام کسانی که در این راه مشوق من بوده‌اند. از استاد گرانقدرم دکتر بابک صادقیان که در این پژوهش با سعه‌ی صدر و حسن خلق راهنمای این حقیر بوده‌اند، و هم‌چنین دکتر شهریار که زحمت داوری این پژوهش را متقبل شده‌اند، کمال تشکر و قدردانی را دارم.

تو خشنود باشی و ما رستگار

خدایا چنان کن سرانجام کار

## چکیده

امروزه اینترنت اشیاء نقش مهمی در زندگی روزمره ما دارد و در این سیستم اشیاء به طور کاملاً اتوماتیک داده‌های زیادی را با یکدیگر تبادل می‌کنند، به همین دلیل این اشیاء نیاز دارند علاوه بر این که صحت داده‌ها را حفظ می‌کنند، یکدیگر را تصدیق اصالت کنند زیرا این اشیاء همواره در معرض هدف قرار گرفتن برای سوءاستفاده هستند.

با توجه به ماهیت اینترنت اشیاء تقریباً غیرممکن است که یک سیستم متمرکز برای این مشکل طراحی شود. بدین‌منظور یک سیستم غیرمتمرکز به نام حباب‌های قابل اعتماد، ارائه شده است که برای ایجاد صحت و دردسترس بودن داده‌ها، از زنجیره‌بلوک استفاده می‌کند. این سیستم بستری امن از نظر حفظ صحت داده‌ها است و دارای ویژگی‌های پایداری و انعطاف‌پذیری است. در این سیستم از ایجاد مناطق مجازی امنیتی به نام حباب، که در آن اشیاء می‌توانند به هم اعتماد کنند و با یکدیگر تبادل اطلاعات امن انجام دهند، استفاده می‌شود. اشیائی که متعلق به حباب‌های غیریکسان هستند نمی‌توانند با یکدیگر تبادل اطلاعات داشته باشند. هر حباب یک مدیر دارد که به همه اعضای آن حباب یک بلیط می‌دهد و اشیاء می‌توانند با استفاده از آن بلیط هویت خود را به زنجیره‌بلوک ثابت کنند.

در این سیستم از الگوریتم‌های رمزنگاری نامتقارن برای تصدیق اصالت و رمزکردن پیام‌ها استفاده می‌شود.

## واژه‌های کلیدی:

زنجیره‌بلوک، رمزنگاری نامتقارن، تصدیق اصالت

فصل اول مقدمه.....	۱
۱-۱ معرفی موضوع.....	۲
۲-۱ تعریف پروژه.....	۳
۳-۱ ساختار پایان نامه.....	۵
فصل دوم مفاهیم و تعاریف مقدماتی.....	۶
۱-۲ اینترنت اشیاء.....	۷
۲-۲ زنجیره بلوک.....	۷
۳-۲ روش اثبات کار برای استخراج بلوک در زنجیره بلوک.....	۱۱
۴-۲ رمز نگاری نامتقارن.....	۱۲
فصل سوم معماری و رویکرد مورد استفاده.....	۱۴
۱-۳ الزامات امنیتی.....	۱۵
۲-۳ مدل پیشنهادی.....	۱۶
۱-۲-۳ فاز آغازین.....	۱۷
۲-۲-۳ عملکرد سیستم.....	۱۸
۳-۳ معماری سیستم پیشنهادی.....	۲۴
فصل چهارم پیاده سازی.....	۲۶
۱-۴ نیازمندی های پیاده سازی.....	۲۷
۲-۴ پیاده سازی بخش زنجیره بلوک.....	۲۷
۳-۴ پیاده سازی بخش رابط کاربری.....	۳۴
فصل پنجم راه اندازی و آزمون سامانه.....	۴۳
۱-۵ راه اندازی سامانه.....	۴۴
۲-۵ آزمون سامانه.....	۴۸
۳-۵ بررسی عملکرد سامانه بر بستر اینترنت اشیاء.....	۵۰
فصل ششم جمع بندی و پیشنهادات.....	۵۲
۱-۶ جمع بندی و پیشنهادات.....	۵۳
منابع و مراجع.....	۵۴

صفحه	فهرست اشکال
شکل ۱-۱	درخواست تشکیل گروه به زنجیره بلوک..... ۴
شکل ۱-۲	مقایسه ساختار ثبت تراکنش در شبکه همتا به همتا زنجیره بلوک با روش سنتی .... ۹
شکل ۲-۲	نمونه‌های از یک زنجیره بلوک ساده..... ۱۰
شکل ۳-۲	رمزنگاری نامتقارن..... ۱۳
شکل ۱-۳	اشیاء..... ۱۸
شکل ۲-۳	تشکیل حباب توسط مدیر حباب..... ۱۹
شکل ۳-۳	اعطای بلیط به دنبال‌کننده‌ها توسط مدیر حباب..... ۲۰
شکل ۴-۳	درخواست اضافه شدن به حباب برای دنبال‌کننده..... ۲۱
شکل ۵-۳	ارتباط بین اشیاء در یک حباب اعتماد..... ۲۲
شکل ۶-۳	کنترل دسترسی در سیستم..... ۲۳
شکل ۷-۳	نمای کلی از اکوسیستم..... ۲۴
شکل ۸-۳	معماری کلی سیستم..... ۲۵
شکل ۱-۴	کلاس بلوک..... ۲۷
شکل ۲-۴	تابع اضافه کردن بلوک..... ۲۸
شکل ۳-۴	تابع اثبات کار..... ۲۸
شکل ۴-۴	تابع is_valid_proof..... ۲۹
شکل ۵-۴	تابع بررسی صحت زنجیره..... ۲۹
شکل ۶-۴	بخش اول تابع check_contract..... ۳۰
شکل ۷-۴	بخش دوم تابع check_contract..... ۳۱
شکل ۸-۴	بخش سوم تابع check_contract..... ۳۱
شکل ۹-۴	تابع mine..... ۳۲

شکل ۴-۱۰ تابع اجماع.....	۳۳
شکل ۴-۱۱ تابع announce_new_block.....	۳۳
شکل ۵-۱ پنجره راهاندازی اولیه رابط کاربری.....	۴۵
شکل ۵-۲ صفحه ثبت نام.....	۴۵
شکل ۵-۳ صفحه گرفتن بلیط از مدیر گروه.....	۴۶
شکل ۵-۴ صفحه ثبت نام به عنوان دنبالکننده.....	۴۷
شکل ۵-۵ صفحه ارسال پیام.....	۴۸
شکل ۵-۶ خروجی سامانه برای ارسال پیام به شیء در حباب دیگر.....	۴۸
شکل ۵-۷ صفحه ارسال پیام بعد نفرسادن پیام به گروه دیگر.....	۴۹
شکل ۵-۸ خروجی سامانه برای ثبت نام بدون بلیط.....	۴۹
شکل ۵-۹ خروجی سامانه در صورت تکراری بودن نام گروه.....	۵۰



## فصل اول

### مقدمه

امروزه با گسترش استفاده از اینترنت اشیاء در زندگی، مسائلی از جمله امنیت این شبکه بسیار مورد توجه است. بدین منظور با ارائه یک سیستم غیرمتمرکز برای تصدیق اصالت اشیاء به این مسئله می-پردازیم. هدف این پروژه طراحی یک سیستم غیرمتمرکز برای تصدیق اصالت در اینترنت اشیاء است.

## ۱-۱ معرفی موضوع

در حال حاضر اینترنت اشیاء با همه جنبه‌های زندگی انسان‌ها درگیر است. طبق مطالعات اخیر در سال ۲۰۲۰ حدود ۵۰ میلیارد دستگاه به اینترنت متصل می‌شوند بنابراین شهروندها به تدریج خانه‌های خود را با وسایل هوشمند از جمله تلویزیون هوشمند، سیستم گرمایشی هوشمند و ... مجهز می‌کنند. در کارخانه ها و محیط‌های صنعتی همکاری بین ربات‌ها و دیگر تجهیزات هوشمند کارایی را افزایش داده است و باعث تولید محصولات بهتر شده است. تکنولوژی اینترنت اشیاء وارد حوزه‌های نظامی، کشاورزی و شهر هوشمند هم شده است.

اینترنت اشیاء نقش اساسی را در هوشمند کردن شهرها ایفا می‌کند. اشیاء متصل به اینترنت پتانسیل کاهش تولید گاز  $CO_2$  را دارند. علاوه بر این اینترنت اشیاء کاربردهای دیگری هم برای هوشمند ساختن شهرها از جمله : مدیریت زباله هوشمند، مدیریت محیط، سیستم حمل‌ونقل هوشمند، مدیریت ترافیک، سیستم مسیریابی هوشمند را دارد.

ایده مورد استفاده در اینترنت اشیاء، حضور تعداد زیادی شیء است که به منظور ارائه طیف وسیعی از خدمات و سرویس‌ها با هم در تعامل و ارتباط هستند. هر شیء فیزیکی یا مجازی باید در دسترس بقیه باشد و بتواند محتوایی تولید کند که همه صرف نظر از مکان خود، بتوانند به آن دسترسی داشته باشند. از طرفی این که فقط اشیائی که تصدیق اصالت شده‌اند بتوانند از این سیستم استفاده کنند بسیار مهم است. در غیر این صورت این سیستم در معرض ریسک‌های امنیتی زیادی از جمله: سرقت اطلاعات، جایگزینی اطلاعات و غصب هویت است. بنابراین مسائل امنیتی مهم‌ترین مانع در برابر گسترش اینترنت اشیاء است زیرا اینترنت اشیاء به شدت در برابر حملات آسیب‌پذیر است. از علل آسیب‌پذیری اینترنت اشیاء می‌توان بی‌سیم بودن اکثر اتصالات و محدود بودن منابع انرژی، حافظه و توانایی پردازش در اکثر دستگاه‌های متصل به این سیستم نام برد که باعث رخ دادن حملاتی از جمله استراغ‌سمع پیام و تغییر پیام می‌شود.

محققان زیادی اینترنت اشیاء را به عنوان یک سیستم از سیستم‌ها توصیف می‌کنند. در مواردی که از این سیستم استفاده می‌شود باید فقط کاربرهای مورد اعتماد حضور داشته باشند. بنابراین الزامات امنیتی متعارف از جمله: تصدیق اصالت، محرمانگی و صحت داده‌ها برای همه قسمت‌های این سیستم شامل اشیاء، شبکه و برنامه‌های نرم‌افزاری مهم است. با این حال، به دلیل محدودیت و ناهمگونی منابع و دستگاه‌ها، راه حل‌های امنیتی موجود کاملاً با چنین سیستمی سازگار نیستند. علاوه بر این، ترکیب چندین فناوری امنیتی مورد نیاز است، که منجر به هزینه‌های اضافی می‌شود. علاوه بر این، راه حل‌های امنیتی کارآمد اغلب متمرکز هستند، به عنوان مثال زیرساخت کلید عمومی (PKI)، که می‌تواند باعث ایجاد مقیاس‌پذیری عظیم در محیطی شود که از هزاران کاربر تشکیل شده باشد. سرانجام، هر مورد استفاده از این سیستم، رویکرد امنیتی و معماری متفاوتی را اعمال می‌کند، که باعث ایجاد مشکلات متعددی در ادغام خدمات و سناریوهای جدید می‌شود. در نتیجه، ارائه راه حل‌های امنیتی جدید برای این سیستم به طور کلی ضروری است. راه حل ارائه شده باید: (۱) امکان ادغام آسان دستگاه‌های جدید و همچنین خدمات جدید را فراهم کند. (۲) کاملاً با نیازهای اینترنت اشیاء سازگار باشد. و (۳) به نوع دستگاه‌ها و همچنین نوع معماری مورد استفاده و طراحی آن بستگی نداشته باشد.

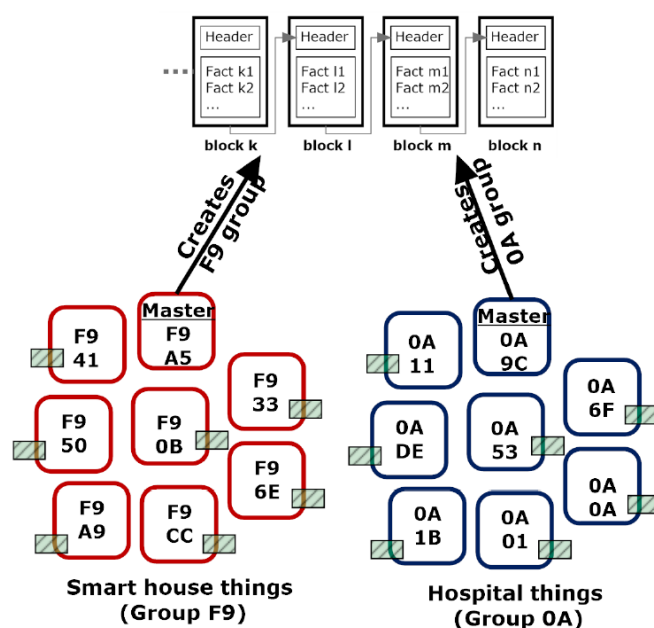
در مقاله "مدیریت دستگاه‌های اینترنت اشیاء با استفاده از زنجیره‌بلوک" [۲] برای برقراری امنیت در اینترنت اشیاء، بستری معرفی شده‌است که هر شیء با یک قرارداد هوشمند مستقل که نحوه رفتار آن را مشخص می‌کند به زنجیره‌بلوک متصل می‌شود که این بستر اجازه ورود هر شیء مشکوک را هم می‌دهد. در اغلب نمونه‌های قبلی که اقدام به ایجاد امنیت در اینترنت اشیاء کرده‌اند، اشیاء از کلید برای رمزنگاری استفاده می‌کنند که این روش از حضور اشیاء مشکوک جلوگیری نمی‌کند و روشی برای تصدیق اصالت نیست.

## ۲-۱ تعریف پروژه

بسیاری از محققان بیان کرده‌اند، زنجیره‌بلوک یک فناوری بسیار امیدوارکننده برای برآورده کردن الزامات امنیتی در زمینه اینترنت اشیاء است. در این پروژه، با استفاده از مزایای قدرت و انعطاف پذیری در زنجیره‌بلوک، ما یک مکانیزم تصدیق اصالت غیرمتمرکز کارآمد به نام حباب اعتماد پیشنهاد می‌کنیم.

هدف از استفاده از زنجیره‌بلوک ایجاد مناطق مجازی امن است، جایی که اشیاء می‌توانند با امنیت با یکدیگر ارتباط برقرار کنند.

سیستم مورد نظر شامل تعدادی حساب است که هر حساب یک مدیر دارد. هر عضو در این شبکه یک جفت کلید عمومی/خصوصی دارد که مدیر حساب، کلید عمومی همه اعضای آن حساب را با کلید خود امضا می‌کند و یک بلیط برای آن‌ها تولید می‌کند. این بلیط شامل شناسه گروه، شناسه شیء و نتیجه تابع درهم‌ساز برای کلید عمومی شیء است که این بلیط با کلید خصوصی مدیر حساب امضا و تایید شده است. اعضا با استفاده از بلیط خود ابتدا به زنجیره‌بلوک می‌پیوندند و سپس می‌توانند به بقیه اعضای که باهم در یک حساب قرار دارند پیام بفرستند.



شکل ۱-۱ درخواست تشکیل گروه به زنجیره‌بلوک

فرستنده برای ارسال پیام به گیرنده، ابتدا پیام خود را به زنجیره‌بلوک ارسال می‌کند، سپس زنجیره‌بلوک اقدام به تصدیق اصالت فرستنده می‌کند و اگر تایید شد پیام در زنجیره‌بلوک قرار می‌گیرد و گیرنده می‌تواند پیام را بخواند.

زنجیره‌بلوک مورد استفاده، یکتا بودن شناسه اشیائی که درخواست اضافه شدن به زنجیره دارند و همچنین یکتا بودن نام گروه به هنگام ایجاد آن را بررسی می‌کند. اشیائی که در حساب‌های متفاوت

قراردارند نمی‌توانند با یکدیگر تبادل اطلاعات کنند زیرا اگر چنین بود یک شیء مشکوک می‌توانست یک حباب تکی بسازد و با بقیه اشیاء ارتباط برقرار کند.

## ۳-۱ ساختار پایان نامه

در این بخش ساختار کلی سایر فصل‌های این پایان‌نامه توضیح داده می‌شود.

در فصل دوم مفاهیم مقدماتی و تعاریفی که برای درک مسئله موردنیاز است، از جمله اینترنت اشیاء، زنجیره‌بلوک و رمزنگاری نامتقارن، مطرح می‌شود. در فصل سوم معماری و رویکرد مورد استفاده بیان می‌شود و همچنین الزامات امنیتی سیستم، مورد بررسی قرار می‌گیرد. در فصل چهارم به نحوه پیاده‌سازی سیستم، شامل دو بخش زنجیره‌بلوک و رابط کاربری می‌پردازیم. در نهایت در فصل پنجم به نحوه راه‌اندازی و آزمون سیستم و نتایج حاصل از انجام تست‌های آزمایشی پرداخته می‌شود. جمع‌بندی و پیشنهاداتی در زمینه فعالیت انجام شده، نیز در فصل ششم ارائه می‌شود.

## فصل دوم

### مفاهیم و تعاریف مقدماتی

برای حل مسأله به کمک مدلی که در فصل سوم به شرح توضیح داده می‌شود، باید درک مناسبی از سیستمی که در آن به حل مسأله می‌پردازیم به دست آورد. در ادامه اجزای این سیستم و مفاهیم آن به تفصیل بیان می‌شود.

## ۱-۲ اینترنت اشیاء

نظریه IOT<sup>۱</sup> یا اینترنت اشیاء، برای نخستین بار در سال ۱۹۹۹ توسط کوین اشتون<sup>۲</sup> بیان شده است اما در حال حاضر اکثر کسب‌وکارها در حال حرکت به سمت استفاده وسیع از این تکنولوژی هستند.

مفهوم اینترنت اشیاء اتصال دستگاه‌های مختلف به یکدیگر از طریق اینترنت است. به کمک اینترنت اشیاء، برنامه‌ها و دستگاه‌های مختلف می‌توانند از طریق اتصال اینترنت با یکدیگر و حتی انسان تعامل و صحبت کنند. برای نمونه می‌توان به یخچال‌های هوشمند که به اینترنت متصل هستند و شما را از موجودی و تاریخ انقضا مواد خوراکی داخل یخچال با خبر می‌سازند اشاره نمود. در واقع، اینترنت اشیاء شما را قادر می‌سازد تا اشیاء مورد استفاده خود را از راه دور و به کمک زیرساخت‌های اینترنتی مدیریت و کنترل کنید.

اینترنت اشیاء فرصت‌هایی ایجاد می‌کند برای ادغام مستقیم دنیای فیزیکی و سیستم‌های مبتنی بر کامپیوتر، سیستم‌هایی مانند؛ خودروهای هوشمند، یخچال‌های هوشمند و خانه‌های هوشمند که این روزها در مباحث و مجالس مختلفی به آن‌ها اشاره می‌شود.

## ۲-۲ زنجیره بلوک

زنجیره بلوک یک دفترکل<sup>۳</sup> توزیع شده و همتابه‌همتا<sup>۴</sup> است که کار فرایند ثبت تراکنش‌ها و ردگیری دارایی‌ها را در یک شبکه‌ی کسب و کار ساده می‌کند. زنجیره بلوک نام خود را وام‌دار روشی است که

---

<sup>۱</sup> Internet of thing

<sup>۲</sup> Kevin Ashton

<sup>۳</sup> Ledger

<sup>۴</sup> Peer-to-Peer

داده‌های تراکنش را در بلوک‌هایی که به یکدیگر پیوند خورده‌اند تا یک زنجیر را تشکیل دهند، ذخیره می‌کند. با افزایش تعداد تراکنش‌ها، زنجیره‌بلوک هم رشد می‌کند.

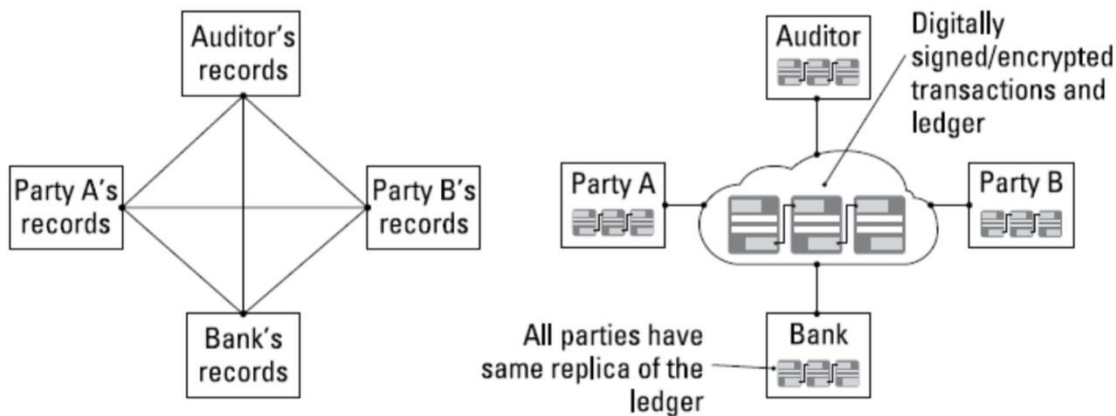
با استفاده از شبکه هم‌تا به هم‌تا، یک زنجیره‌بلوک کاملاً غیرمتمرکز است. به طور دقیق‌تر، هر گره<sup>۵</sup> از شبکه یک نسخه از دفترکل را نگه می‌دارد تا از خرابی در زنجیره جلوگیری کند. در حال حاضر، تعداد بیشماری برنامه‌های کاربردی، زنجیره‌بلوک را در موارد استفاده چندگانه مورد بررسی قرار داده و از آن‌ها به عنوان روشی مطمئن برای ایجاد و مدیریت یک بانک اطلاعاتی توزیع شده و حفظ سوابق برای انواع معاملات دیجیتالی استفاده می‌کند.

همانطور که در سمت چپ شکل ۱-۲ نشان داده شده‌است، در شیوه‌های سنتی ثبت تراکنش‌ها و ردگیری دارایی‌ها، هریک از مشارکت‌کنندگان در شبکه، دفترکل خودش را به همراه دیگر رکوردها نگهداری می‌کند. این شیوه‌ی سنتی می‌تواند گران باشد، تا حدی به این دلیل که به واسطه‌هایی نیاز است که برای خدمات خود پول دریافت می‌کنند. این کار به دلیل بروز تاخیرها در اجرای توافق‌نامه‌ها و دوباره کاری‌های لازم برای نگهداری تعداد زیادی دفترکل، آشکارا ناکارآمد است. این کار آسیب‌پذیر هم هست، چرا که اگر یک سامانه‌ی مرکزی (برای مثال، یک بانک) به دلیل کلاهبرداری، حمله‌ی سایبری یا اشتباهی کوچک به خطر بیفتد، کل شبکه‌ی کسب‌وکار از آن متاثر خواهد شد. در سمت راست شکل ۲-۱ شبکه‌های کسب‌وکاری که از زنجیره‌بلوک استفاده می‌کنند نشان داده شده‌است. ساختار زنجیره‌بلوک به مشارکت‌کنندگان این توانایی را می‌دهد تا به صورت مشارکتی از دفترکلی استفاده کنند که هرگاه تراکنشی رخ دهد، از طریق همانندسازی هم‌تا به هم‌تا به‌روز می‌شود. همانندسازی هم‌تا به هم‌تا یعنی این که هر مشارکت‌کننده (گره) در شبکه هم به‌عنوان یک ناشر و هم به‌عنوان یک مشترک کار می‌کند. هر گره می‌تواند تراکنش‌ها را به دیگر گره‌ها بفرستد یا از آن‌ها دریافت کند و داده‌های انتقال یافته در سراسر شبکه هماهنگ شده است.

---

<sup>۵</sup> Node





شکل ۱-۲ مقایسه ساختار ثبت تراکنش در شبکه همتا به همتا زنجیره بلوک با روش سنتی

- یک شبکه‌ی زنجیره بلوک دارای ویژگی‌های کلیدی زیر است:
- توافق نظر<sup>۶</sup>: برای این که تراکنشی معتبر باشد، همه‌ی طرف‌ها باید روی اعتبار آن توافق نظر داشته باشند.
- اصل بودن<sup>۷</sup>: مشارکت کنندگان می دانند دارایی از کجا می‌آید و مالکیت آن در طول زمان چگونه تغییر کرده است .
- تغییرناپذیری: هیچ مشارکت کننده‌ای نمی‌تواند پس از این که تراکنشی در دفترکل ثبت شد، آن را دستکاری کند. اگر تراکنشی خطا داشته باشد، باید از تراکنش جدیدی برای معکوس کردن آن خطا استفاده شود و در این صورت هر دوی این تراکنش‌ها روئیت پذیر است.
- قطعیت<sup>۸</sup>: یک دفترکل مشترک جایی است که برای مشخص شدن مالکیت یک دارایی یا کامل شدن یک تراکنش به آن مراجعه می‌شود .

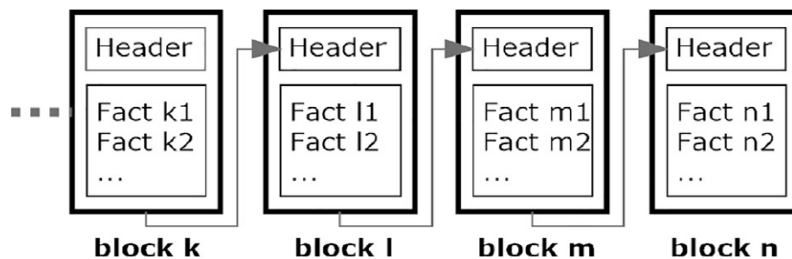
<sup>۶</sup> Consensus

<sup>۷</sup> Provenance

<sup>۸</sup> finality

دفترکل در زنجیره بلوک از بلوک‌های مختلف تشکیل شده است، هر بلوک از دو قسمت تشکیل شده است. اولین مورد، معاملات یا اطلاعات را نشان می‌دهد (که بانک اطلاعاتی<sup>۹</sup> باید آن را ذخیره کند)، که می‌تواند از هر نوع مانند معاملات پولی، داده‌های بهداشتی، اطلاعات سیستم، اطلاعات مربوط به ترافیک و غیره باشد. مورد دوم سربرگ<sup>۱۰</sup> است و شامل اطلاعاتی در مورد آن بلوک است از جمله زمان<sup>۱۱</sup>، نتیجه تابع درهمساز برای تراکنش خود و همچنین نتیجه تابع درهمساز برای بلوک قبلی.

بنابراین، مجموعه بلوک‌های موجود زنجیره‌ای از بلوک‌های مرتبط و مرتب را تشکیل می‌دهد. زنجیره‌ای طولانی‌تر، جعل آن سخت‌تر است. در واقع، اگر یک کاربر مخرب می‌خواهد تراکنش را روی یک بلوک تغییر دهد، (۱) باید تمام بلوک‌های بعد از آن را تغییر دهد، زیرا آن‌ها بوسیله نتایج تابع درهمساز برای بلوک قبلی خود، با یکدیگر در ارتباط هستند. (۲) سپس باید نسخه زنجیره‌بلوکی را که هر گره شرکت کننده ذخیره می‌کند، تغییر دهد. شکل زیر نمونه‌ای از زنجیره‌بلوک ساده را نشان می‌دهد.



شکل ۲-۲ نمونه‌ای از یک زنجیره‌بلوک ساده

دو نوع گره شرکت کننده وجود دارد: (۱) گره‌هایی که فقط می‌توانند اطلاعات را بخوانند (حالت منفعل<sup>۱۲</sup>). و (۲) گره‌هایی که می‌توانند حقایق را بخوانند و بنویسند (حالت فعال<sup>۱۳</sup>) که معمولاً استخراج کننده<sup>۱۴</sup> نامیده می‌شوند. برای اضافه کردن یک تراکنش جدید به زنجیره‌بلوک، مراحل زیر انجام می‌شود:

<sup>۹</sup> Database

<sup>۱۰</sup> Header

<sup>۱۱</sup> Timestamp

<sup>۱۲</sup> Passive

<sup>۱۳</sup> Active

۱. تراکنش با سایر معاملات به صورت بلوک، گروه‌بندی می‌شود.
  ۲. استخراج‌کننده‌ها تأیید می‌کنند که معاملات درون بلوک قوانین تعریف شده را رعایت می‌کنند.
  ۳. استخراج‌کننده‌ها برای تأیید اعتبار بلوک اضافه شده یک مکانیزم توافق نظر انجام می‌دهند.
  ۴. پاداش به استخراج‌کننده / استخراج‌کننده‌هایی که اعتبار این بلوک را تأیید کردند، تعلق می‌گیرد.
  ۵. معاملات تأیید شده در زنجیره بلوک ذخیره می‌شوند.
- برای اثبات اعتبار صحیح بلوک‌ها، روش‌های بی شماری وجود دارد. بیشترین کاربرد آن‌ها روش اثبات کار (PoW)<sup>۱۵</sup> است.

## ۲-۳ روش اثبات کار برای استخراج بلوک در زنجیره بلوک

در روش اثبات کار، یک استخراج‌کننده باید کارهایی از پیش تعریف شده را انجام دهد، که اغلب یک معمای ریاضی یا چالشی است که محاسبه آن دشوار است اما به راحتی قابل بررسی است. تولید یک اثبات کار یک فرآیند تصادفی با احتمال پایین است. بنابراین برای تولید یک اثبات کار تعداد زیادی آزمایش و خطا نیاز است. اثبات کار برای اعتبارسنجی هر یک از بلوک‌ها استفاده می‌شود. دشواری این چالش ریاضی می‌تواند با توجه به زمان مورد نیاز برای اعتبارسنجی یک بلوک و قدرت محاسبات استخراج‌کننده‌ها تعدیل شود.

از یک طرف، روش اثبات کار از این مزیت برخوردار است که معاملات و بلوک‌ها را در برابر تغییر محافظت می‌کند، زیرا مهاجم باید بخشی از زنجیره بلوک را تغییر دهد و نسخه زنجیره‌ای خود را در همه گره‌ها به‌روز کند، که نیاز به یک قدرت و انرژی محاسباتی عظیم دارد. از طرف دیگر، روش اثبات کار از برخی کاستی‌ها رنج می‌برد که می‌تواند عواقب فاجعه بار داشته باشد. در واقع، اثبات کار در محاسبات پازل به اتلاف انرژی زیادی احتیاج دارد.

---

<sup>۱۴</sup> Miner

<sup>۱۵</sup> Proof of work

در واقع، با گذشت زمان، پاداش استخراج کردن کاهش می‌یابد، که منجر به کاهش تعداد استخراج‌کننده‌ها می‌شود، زیرا استخراج‌کننده‌ها تنها از طریق استخراج معاملات درآمد کسب می‌کنند، که همچنین با گذشت زمان این مقدار به علت تصمیم کاربران که هزینه کمتری برای معاملات بپردازند، کاهش می‌یابد. کاهش تعداد استخراج‌کننده باعث می‌شود سیستم زنجیره‌بلوک در برابر حمله ۵۱٪ آسیب پذیر باشد. این حمله وقتی اتفاق می‌افتد که یک استخراج‌کننده مخرب یا تعدادی از استخراج‌کننده‌های مخرب حداقل ۵۱٪ قدرت محاسباتی شبکه را کنترل می‌کنند. بنابراین، او می‌تواند ضمن اینکه اعتبار معاملات دیگران در شبکه را باطل می‌کند، بلوک معاملات متقلبانه برای خود یا نهاد دیگری ایجاد کند.

## ۴-۲ رمز نگاری نامتقارن

در بحث رمزنگاری دو الگوریتم رمزنگاری مهم وجود دارد: متقارن<sup>۱۶</sup> و نامتقارن<sup>۱۷</sup> در الگوریتم رمزنگاری نامتقارن از دو کلید متفاوت استفاده می‌شود و یک رابطه ریاضی بین آن‌ها وجود دارد. که این کلیدها را تحت عنوان کلید خصوصی<sup>۱۸</sup> و کلید عمومی<sup>۱۹</sup> می‌شناسیم که با یکدیگر یک جفت کلید را تشکیل می‌دهند.

نحوه کار با این دو کلید بدین صورت است که اگر متن مورد نظر برای رمزنگاری را با کلید عمومی رمز کنیم متن رمز شده را تنها می‌توان با کلید خصوصی از حالت رمز خارج کرد و اگر متن را با کلید خصوصی رمز کنیم تنها می‌توان با کلید عمومی آن را رمزگشایی کرد.

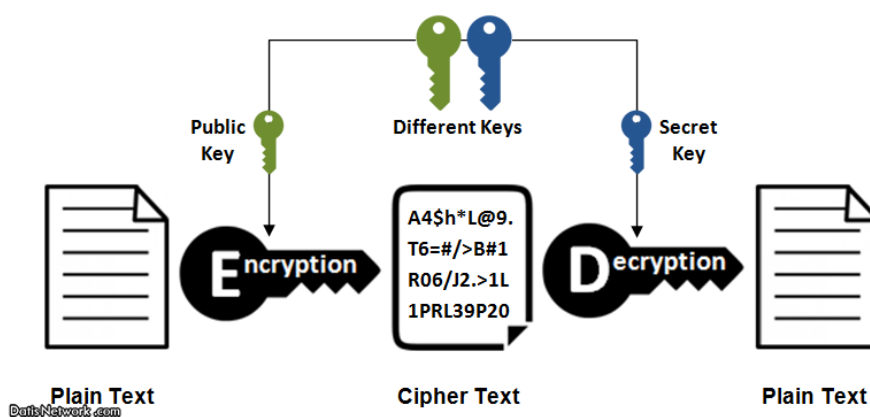
<sup>۱۶</sup> Symmetric

<sup>۱۷</sup> Asymmetric

<sup>۱۸</sup> Private Key

<sup>۱۹</sup> Public Key

## Asymmetric Encryption



شکل ۲-۳ رمزنگاری نامتقارن

استفاده از الگوریتم های نامتقارن پردازش زیادی را برای CPU در زمان رمزنگاری و رمزگشایی داده ایجاد می کند. در نتیجه بجای استفاده از این روش برای رمزنگاری کل اطلاعات، از آن برای مراحل خاصی از ارتباط استفاده می کنیم مانند تصدیق اصالت

یکی از دلایلی که این روش را رمزنگاری کلید عمومی می نامند این است که ما این کلید را به صورت عمومی در اختیار همه قرار می دهیم. کلید دیگری که در این روش استفاده می شود کلید خصوصی نامیده می شود و این کلید در اختیار کسی قرار نمی گیرد و مخصوص خود دستگاه است.

یک نمونه از الگوریتم های نامتقارن RSA است نام این الگوریتم برگرفته از سازندگان آن است یعنی Rivest ، Shamir و Adleman است. امروزه استفاده اصلی از این روش برای تصدیق اصالت است. طول کلید می تواند از ۵۱۲ تا ۴۰۹۶ باشد و حداقل مقدار برای داشتن امنیت مناسب ۱۰۲۴ می باشد. هرچه طول کلید بیشتر باشد امن تر خواهد بود.

## فصل سوم

### معماری و رویکرد مورد استفاده

در فصل‌های قبل به ساختار کلی اینترنت اشیاء و زنجیره بلوک پرداختیم و با تعریف حباب‌های اعتماد به برقراری امنیت برای اشیاء در سیستم اینترنت اشیاء پرداخته شد. در این فصل قصد داریم با بررسی الزامات امنیتی مورد نیاز به بررسی دقیق سیستم حباب‌های اعتماد بپردازیم.

### ۳-۱ الزامات امنیتی

یک طرح اینترنت اشیاء برای اطمینان از پایداری و مقاومت در برابر محیط باید الزامات امنیتی بیشتری را برآورده کند. بنابراین، در این بخش اهداف اصلی امنیتی را شرح می‌دهیم و معیارهای مورد نیاز برای ارزیابی مناسب بودن برنامه‌های تصدیق اصالت برای ایمن‌سازی موارد استفاده اینترنت اشیاء را معرفی می‌کنیم.

#### صحت

حفظ صحت نیاز اساسی است که هر طرح باید از آن اطمینان داشته باشد. در این حوزه، صحت به دو بخش تقسیم می‌شود:

۱. صحت پیام‌ها (معاملات / ارتباطات): یک پیام نباید در حین انتقال شبکه تغییر کند.
۲. صحت داده‌ها: شامل حفظ صحت داده‌ها در کل چرخه زندگی آن است. بنابراین، فقط کاربران مجاز می‌توانند داده‌های ذخیره شده را تغییر دهند.

#### در دسترس بودن

در دسترس بودن به معنای این است که منابع باید در صورت تقاضا افراد قانونی در دسترس آن‌ها قرار گیرد. بنابراین، یک سیستم باید در برابر انکار حملات مقاوم باشد.

#### مقیاس پذیری

مقیاس پذیری توانایی اطمینان به این که اندازه سیستم در عملکرد آن تأثیری ندارد، تعریف می‌شود. به عنوان مثال، اگر تعداد موارد استفاده شده خیلی زیاد شود، زمان لازم برای عملکرد سیستم مانند سرویس اطلاع رسانی، تحت تأثیر قرار نمی‌گیرد.

## عدم تکذیب<sup>۱</sup>

این به توانایی اطمینان از اینکه یک شیء نمی‌تواند منکر انجام یک عمل خاص باشد، اشاره دارد، به عنوان مثال. یک دستگاه نمی‌تواند پیام ارسال کند و آن را انکار کند.

## شناسایی<sup>۲</sup>

شناسایی در اکثر موارد استفاده اینترنت اشیاء یک نیاز اصلی است. این نشان دهنده خلاف ناشناس بودن است. به عنوان مثال، در یک پارکینگ هوشمند، هنگامی که یک سنسور یک محل پارکینگ یک اعلان ارسال می‌کند، سیستم مدیریت باید دقیقاً بداند که کدام سنسور در حال برقراری ارتباط است تا بتواند وضعیت نقاط پارکینگ را به‌روز کند.

## احراز هویت متقابل<sup>۳</sup>

احراز هویت روش اثبات هویت است. احراز هویت متقابل بیانگر درخواستی است که هر دو طرف ارتباط دهنده، یکدیگر را تصدیق کنند.

## ۲-۳ مدل پیشنهادی

هدف اصلی رویکرد ما ایجاد مناطق مجازی ایمن در محیط اینترنت اشیاء است. هر دستگاه فقط باید با دستگاه‌های منطقه خود ارتباط برقرار کند و هر دستگاه دیگر را مخرب قلمداد کند. ما این مناطق را حباب اعتماد می‌نامیم. بنابراین، حباب اعتماد منطقه‌ای است که تمام اعضای آن می‌توانند به یکدیگر اعتماد کنند. هر حباب برای دستگاه‌هایی که عضو حباب نیستند، محافظت شده و غیرقابل دسترس است. برای دستیابی به چنین سیستمی، ما از یک زنجیره‌بلوک، استفاده می‌کنیم.

ارتباطات موجود در سیستم به عنوان تراکنش انجام می‌شود و برای تأمین اعتبار باید از طریق این زنجیره‌بلوک تأیید شود. به عنوان مثال، اگر دستگاه A پیامی را به دستگاه B ارسال می‌کند، (۱) A این

<sup>۱</sup> Non-repudiation

<sup>۲</sup> Identification

<sup>۳</sup> Mutual authentication



پیام را به زنجیره بلوک ارسال می کند، (۲) اگر زنجیره بلوک A را تأیید کند، آن تراکنش تایید می شود. سرانجام، (۳) B می تواند پیام را بخواند. در ادامه کلیه چرخه حیات یک دستگاه را در سیستم اینترنت اشیاء که حساب های اعتماد را پیاده سازی می کند، شرح می دهیم.

### ۳-۲-۱ فاز آغازین

در هر حساب یک شیء به عنوان مدیر حساب<sup>۱</sup> (صاحب یک جفت کلید خصوصی / عمومی) طراحی شده است که شبیه به یک فرد دارای مجوز صدور گواهی نامه است. هر دستگاه می تواند مدیر باشد. علاوه بر این، هر شیء، که بخشی از سیستم را تشکیل می دهد، دنبال کننده<sup>۲</sup> نامیده می شود. مدیر برای هر دنبال کننده یک جفت کلید خصوصی / عمومی ایجاد می کند. سپس، به هر دنبال کننده ساختاری به نام بلیط ارائه می شود، که نشان دهنده یک گواهی شامل: (۱) اسم گروه (group)، که حسابی را نشان می دهد که جسم بخشی از آن خواهد بود، (۲) اسم شیء (name)، که معرف شناسه دنبال کننده در حساب است، (۳) Pubaddress، که نشان دهنده کلید عمومی دنبال کننده است و (۴) یک ساختار امضایی که امضای مدیر حساب با الگوریتم امضای دیجیتال RSA با استفاده از کلید خصوصی است. ساختار بلیط به شرح زیر است:

=====

Group : XX

Name : YY

Pubaddress : @@

=====

Sign( RSA(Group : XX , Name : YY , Pubaddress : @@) )

<sup>۱</sup> Master of bubble

<sup>۲</sup> Follower

### ۳-۲-۲ عملکرد سیستم

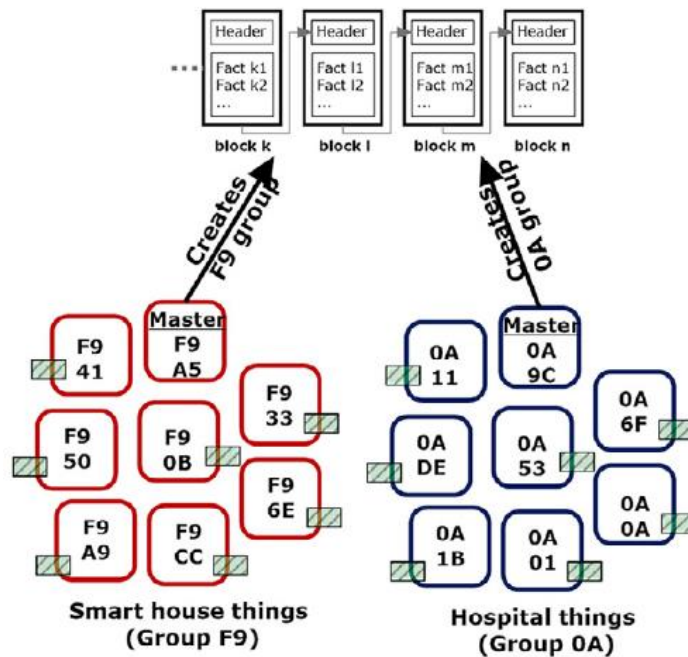
در این بخش فازهای متفاوت از این سیستم را دقیق بررسی می‌کنیم.

در ابتدا در شکل ۳-۱ نشان داده شده است که اشیائی که به سیستم متصل می‌شوند می‌توانند از هر نوعی باشند.



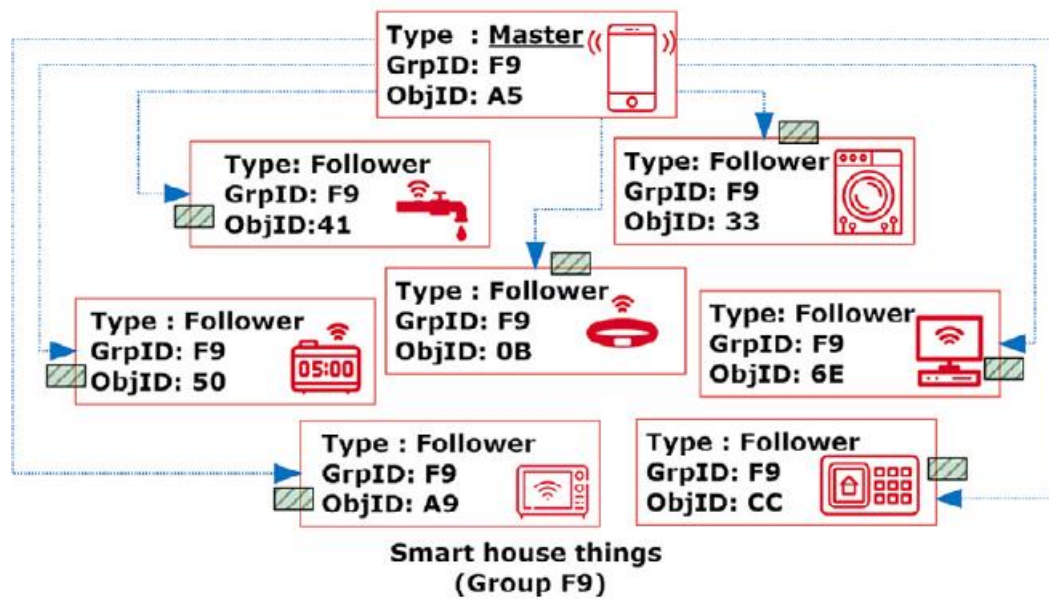
شکل ۳-۱ اشیاء

در شکل ۳-۲ مدیر گروه درخواست تشکیل حباب را به زنجیره‌بلوک ارسال می‌کند. تراکنش درخواست ساخت حباب شامل نام مدیر و نام حباب است. زنجیره‌بلوک ابتدا بررسی می‌کند که قبلاً گروهی با این نام وجود نداشته باشد و همچنین بررسی می‌کند فردی با این نام قبلاً گروهی در گروه دیگری عضو نباشد. سپس تراکنش ساخت حباب در زنجیره ذخیره می‌شود.



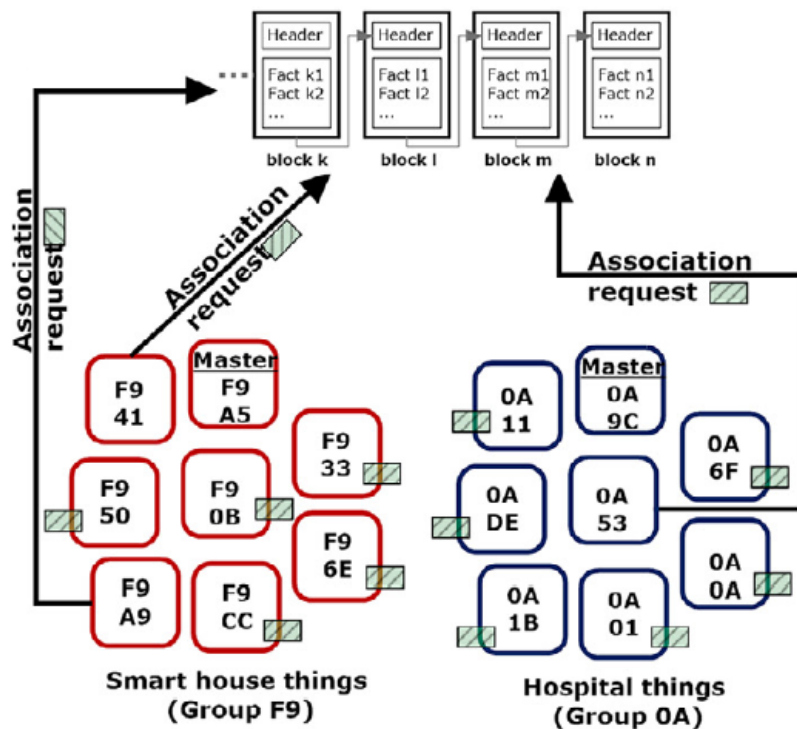
شکل ۳-۲ تشکیل حساب توسط مدیر حساب

افرادی که قصد عضو شدن در هر حسابی را دارند ابتدا از مدیر آن حساب درخواست صدور بلیط را می‌کنند سپس مدیر برای آن‌ها بلیطی صادر می‌کند و بلیط را با کلید خصوصی خودش امضا می‌کند. در نهایت بلیط یکبار دیگر با کلید خصوصی خود دنبال‌کننده امضا می‌شود و به دنبال‌کننده تحویل داده می‌شود.



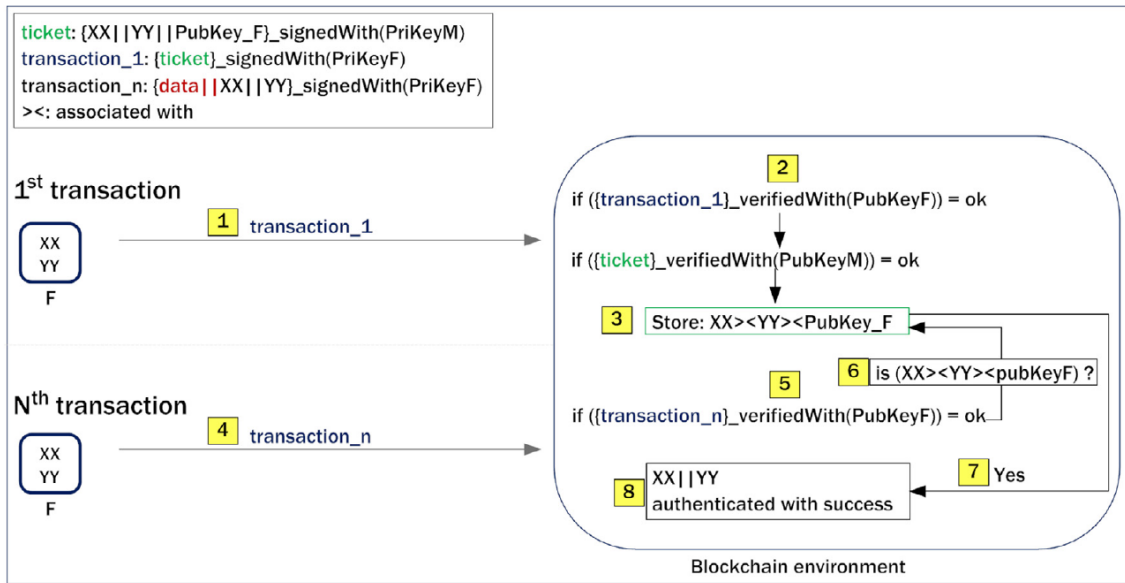
شکل ۳-۳ اعطای بلیط به دنبال کننده‌ها توسط مدیر حساب

دنبال کننده پس از این که بلیط را از مدیر دریافت کرد اقدام به عضو شدن در آن حساب می‌کند. زنجیره بلوک ابتدا به وسیله کلید عمومی مدیر حساب بررسی می‌کند که بلیط ارائه شده معتبر باشد سپس بررسی می‌کند که این نام قبلاً در گروهی عضو نشده باشد و نام آن منحصر به فرد باشد. اگر این شرایط برقرار نباشد شیء به حساب اضافه نمی‌شود.



شکل ۳-۴ درخواست اضافه شدن به حباب برای دنبال کننده

هنگامی که اولین تراکنش (درخواست اضافه شدن به حباب) یک دنبال کننده موفق شد، در درخواست‌های بعدی دیگر نیازی به استفاده از بلیط خود برای تأیید اعتبار ندارد. برای جزئیات بیشتر، یک مثال در شکل ۳-۵ به تفصیل بیان شده است. در این مثال، ما یک دستگاه دنبال کننده به نام F را توصیف می‌کنیم که بلیط صادر شده توسط مدیر را تهیه کرده است. بلیط شامل  $group = XX$ ،  $name = YY$  و کلید عمومی  $pubKey\_F$  است.



شکل ۳-۵ ارتباط بین اشیاء در یک حباب اعتماد

مراحل ارتباط به شرح زیر است:

- اولین تراکنش یک دنبال کننده درخواست برای اضافه شدن به حباب است. پیام ارسالی با کلید خصوصی دنبال کننده امضا شده است و حاوی بلیط دنبال کننده است.
- هنگامی که زنجیره بلوک تراکنش را دریافت می کند، صحت آن را با تأیید امضا با کلید عمومی دنبال کننده تأیید می کند. سپس، بلیط دنبال کننده با استفاده از کلید عمومی مدیر تأیید می شود.
- اگر بلیط معتبر بود، آنگاه زنجیره بلوک یک مجموعه از name, group و کلید عمومی را ذخیره می کند. (XX, YY و PubKey\_F)
- مرحله چهارم حالتی را توضیح می دهد که F تراکنش دیگری را پس از درخواست اضافه شدن به حباب بفرستد. این تراکنش حاوی موارد زیر است: (۱) داده های رد و بدل شده ، (۲) XX, YY (۳) و (۴) امضای RSA از قسمت های قبلی تراکنش با استفاده از کلید خصوصی دنبال کننده.
- هنگامی که زنجیره بلوک تراکنش را دریافت کرد، با تأیید امضا با کلید عمومی دنبال کننده، صحت آن را تأیید می کند.

۸. دستگاه با موفقیت تأیید شده است.

The diagram illustrates the process of creating a new group and associating it with existing groups in a distributed ledger system. It shows four categories of things: Factory things (Group 39), Smart house things (Group F9), School things (Group OA), and Hospital things (Group 0A). Each category has a Master node and several data nodes. The process involves creating a new group (Group 39) and associating it with existing groups (Group F9, Group OA, Group 0A). The diagram shows the flow of data and the creation of new blocks in the ledger.

**Factory things (Group 39):** Master 39 5E. Data nodes: F9 41, F9 50, F9 A9, F9 A5, F9 0B, F9 CC, F9 33, F9 6E.

**Smart house things (Group F9):** Master F9 BB. Data nodes: F9 41, F9 50, F9 A9, F9 A5, F9 0B, F9 CC, F9 33, F9 6E.

**School things (Group OA):** Master OA BB. Data nodes: OA 11, OA DE, OA 0A, OA 01, OA 9C, OA 53, OA 6F, OA 1B.

**Hospital things (Group 0A):** Master 0A 9C. Data nodes: OA 11, OA DE, OA 0A, OA 01, OA 9C, OA 53, OA 6F, OA 1B.

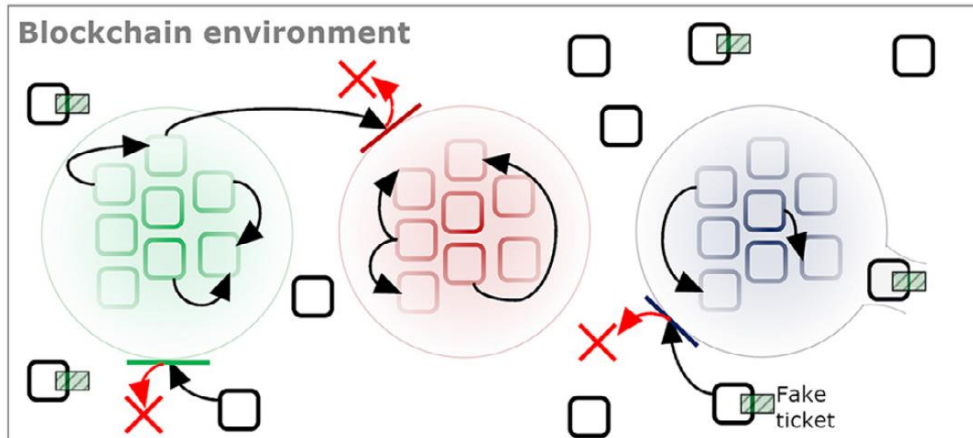
**Process Flow:**

- Creates 39 group:** A black arrow labeled "1" points from the Factory things Master to the ledger.
- Transaction failed (F9 ≠ 0A):** A red dashed arrow labeled "2" points from the Factory things Master to the ledger.
- Transaction failed (0A exists):** A red dashed arrow labeled "2" points from the Smart house things Master to the ledger.
- Creates OA group:** A black arrow labeled "1" points from the Smart house things Master to the ledger.
- Transaction failed (authentication error):** A red dashed arrow labeled "2" points from the Hospital things Master to the ledger.
- Creates 41 group:** A black arrow labeled "1" points from the Hospital things Master to the ledger.

**Blockchain Structure:** The ledger consists of blocks (block k, block l, block m, block n) with headers and facts. The facts are: Fact k1, Fact k2, ..., Fact l1, Fact l2, ..., Fact m1, Fact m2, ..., Fact n1, Fact n2, ...

شکل ۳-۷ نمای کلی از اکوسیستم را توصیف می‌کند. موارد مجاز (داشتن بلیط) را می‌توان در هر زمان به گروه‌های آن‌ها اضافه کرد. از نظر تئوریک، تعداد اشیاء گروه نامحدود است، زیرا به معماری کاملاً غیرمتمرکز متکی است. اشیاء بدون بلیط و یا موارد جعلی نمی‌توانند با حباب در ارتباط باشند، بنابراین نمی‌توانند با گره‌های حباب ارتباط برقرار کنند. به خاطر امضای تراکنش‌ها، تأیید صحت شیء و

صحت داده‌های تغییر یافته تضمین می‌شود. سرانجام حباب‌ها کاملاً از هم جدا می‌شوند و گره‌های حباب‌های مختلف نمی‌توانند اطلاعات یکدیگر را ارسال یا دریافت کنند.



شکل ۳-۷ نمای کلی از اکوسیستم

### ۳-۳ معماری سیستم پیشنهادی

معماری کلی این سیستم دارای بخش‌های مدیر حباب، دنبال‌کننده‌ها در هر حباب و زنجیره‌بلوک می‌شود بدین صورت که شامل تعدادی حباب است که هر حباب شامل تعدادی شیء است و زنجیره‌بلوکی که ساخته می‌شود دارای اطلاعات تمام حباب‌ها و تمام اشیاء است و این زنجیره غیر قابل دست‌کاری است.

مدیر، حباب را با یک نام یکتا تشکیل می‌دهد و اطلاعات حباب را در زنجیره‌بلوک ثبت می‌کند. مدیر برای اشیائی که قصد اضافه شدن به گروه را دارند بلیط صادر می‌کند. دنبال‌کننده‌ها با ارائه بلیط فقط می‌توانند عضو یک حباب شوند و نمی‌توانند یک حباب جدید را ایجاد کنند. دنبال‌کننده‌ها برای برقراری اولین تراکنش، نیاز به تصدیق اصالت با استفاده از بلیط دارند تا به زنجیره‌بلوک اضافه شوند.

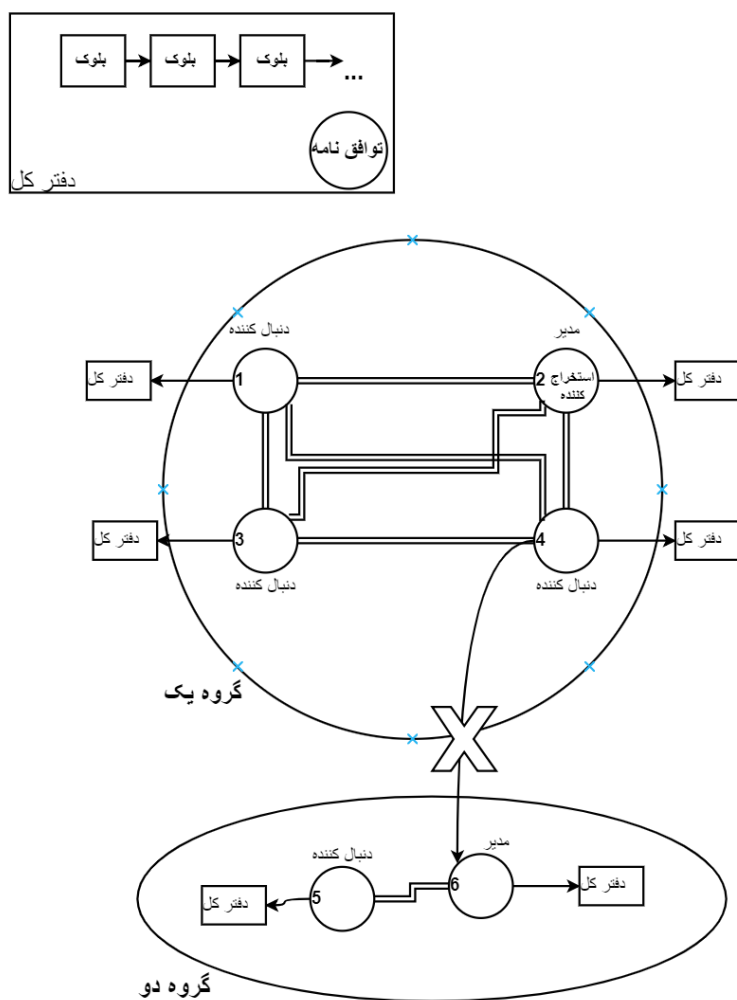
یک سیستم بر مبنای زنجیره‌بلوک شامل قسمت‌های زیر می‌شود :

- گره(شیء)
- تراکنش
- بلوک



- زنجیره
- استخراج کننده
- توافق نامه

هر گره در این سیستم یک نسخه کامل از دفترکل زنجیره بلوک را دارد.



شکل ۳-۸ معماری کلی سیستم

## فصل چهارم

### پیاده‌سازی

در فصل قبل نحوه کار سیستم در فازهای متفاوت را مشاهده کردیم. در این فصل به پیاده‌سازی مراحل قبل می‌پردازیم.

## ۴-۱ نیازمندی‌های پیاده‌سازی

برای پیاده‌سازی از زبان برنامه نویسی پایتون<sup>۱</sup> و چارچوب وب Flask استفاده می‌کنیم. کتابخانه‌های متفاوتی از python نیاز داریم از جمله: Crypto، tkinter، requests و zipfile ...

## ۴-۲ پیاده‌سازی بخش زنجیره‌بلوک

برای پیاده‌سازی کلاس زنجیره‌بلوک ابتدا کلاسی برای هر بلوک تعریف می‌کنم به نام Block که هر بلوک به وسیله شماره بلوک، تراکنش‌ها، زمان، نتیجه تابع درهم‌ساز برای بلوک قبل و یک عدد به نام nonce ساخته می‌شود. هر شیء بلوک یک تابع درهم‌ساز هم دارد.

```
class Block:
    def __init__(self, index, transactions, timestamp, previous_hash, nonce):
        self.index = index
        self.transactions = transactions
        self.timestamp = timestamp
        self.previous_hash = previous_hash
        self.nonce = nonce

    def compute_hash(self):
        """
        A function that return the hash of the block contents.
        """
        block_string = json.dumps(self.__dict__, sort_keys=True)
        return sha256(block_string.encode()).hexdigest()
```

### شکل ۴-۱ کلاس بلوک

کلاس دیگری که داریم کلاس Blockchain است که شامل توابع زیر است:

- create\_genesis\_block برای ایجاد کردن بلوک اولیه برای زنجیر
- add\_block این تابع پس از تایید بلوک آن را به زنجیره اضافه می‌کند

<sup>۱</sup> Python

```
def add_block(self, block, proof):
    """
    A function that adds the block to the chain after verification.
    Verification includes:
    * Checking if the proof is valid.
    * The previous_hash referred in the block and the hash of latest block
    in the chain match.
    """
    previous_hash = self.last_block.hash
    if previous_hash != block.previous_hash:
        print("in add block false because previous_hash")
        return False
    if not Blockchain.is_valid_proof(block, proof):
        print("in add block not valid proof ")
        return False

    block.hash = proof
    self.chain.append(block)
    return True
```

#### شکل ۴-۲ تابع اضافه کردن بلوک

- proof\_of\_work با تغییر مقدار nonce، نتیجه تابع درهم‌ساز را با شرایط داده شده مطابق می‌کند.

```
# @classmethod
def proof_of_work(self, block):
    """
    Function that tries different values of nonce to get a hash
    that satisfies our difficulty criteria.
    """
    block.nonce = 0
    computed_hash = block.compute_hash()
    while not computed_hash.startswith('0' * Blockchain.difficulty):
        block.nonce += 1
        computed_hash = block.compute_hash()

    return computed_hash
```

#### شکل ۴-۳ تابع اثبات کار

- is\_valid\_proof برای تشخیص تایید یک بلوک قبل از اضافه شدن به زنجیره

```

@classmethod
def is_valid_proof(cls, block, block_hash):
    """
    Check if block_hash is valid hash of block and satisfies
    the difficulty criteria.
    """
    f1 = block_hash.startswith('0' * Blockchain.difficulty)
    f2 = block_hash == block.compute_hash()
    return (f1 and f2)

```

#### شکل ۴-۴ تابع is\_valid\_proof

- check\_chain\_validity بررسی معتبر بودن کل زنجیره بلوک

```

@classmethod
def check_chain_validity(cls, chain):
    result = True
    previous_hash = "0"
    for block in chain:
        block2 = Block(block["index"],
                        block["transactions"],
                        block["timestamp"],
                        block["previous_hash"],
                        block["nonce"])
        block_hash = block['hash']
        c1 = not cls.is_valid_proof(block2, block_hash)
        c2 = previous_hash != block2.previous_hash
        if c1 or c2:
            print("condition in check_chain_validity is false ")
            result = False
            break
        previous_hash = block_hash

    return result

```

#### شکل ۴-۵ تابع بررسی صحت زنجیره

- check\_contract موارد قراردادی سامانه را برای هر بلوک بررسی می‌کند شامل سه بخش است بخش اول قراردادهای لازم برای ثبت‌نام به عنوان مدیر را بررسی می‌کند، بخش دوم قراردادها برای ثبت‌نام به عنوان دنبال‌کننده و بخش سوم قراردادها برای ارسال پیام را بررسی می‌کند.

---

```
def check_contract(self, tr):
    #-----mine master-----
    if tr[0]['type'] == "register" and tr[0]['category'] == "master":
        for block in self.chain:
            for t in block.transactions:
                if t['type'] == "register" and (t['name'] == tr[0]['name']
                or t['group'] == tr[0]['group']) :
                    print("in check_contract this was repetitive")
                    return False
            return True
    "
```

### شکل ۴-۶ بخش اول تابع `check_contract`

---

```
elif tr[0]['type'] == "register" and tr[0]['category'] == "follower":
    exist = False
    ip = ''
    port = ''
    for block in self.chain:
        for t in block.transactions:
            if t['type'] == "register" and t['category'] == "master"
            and t['group'] == tr[0]['group'] :
                exist = True
                ip = t['ip']
                port = t['port']
                break
    if not exist:
        print("in check_contract group not exist")
        return False
    else:
        for block in self.chain:
            for t in block.transactions:
                if t['type'] == "register" and t['name'] == tr[0]['name'] :
                    print("in check_contract this was repetitive")
                    return False
        ticket_string = tr[0]['ticket']
        ticket = json.loads(ticket_string)

        follower_sign = ticket['follower_sign']
        del ticket['follower_sign']
```

---

```

dump_ticket = json.dumps(ticket, sort_keys=True)
h = SHA256.new(dump_ticket.encode())
follower_publickey = ticket['Pubaddress'].encode()
Fpubkey = RSA.importKey(follower_publickey)
v = Fpubkey.verify(h.digest(), follower_sign)
print("in check_contract verify is " + str(v))
if v :
    master_address = str(ip)+":"+str(port)
    headers = {'Content-Type': "application/json"}
    response = requests.post(master_address + "/verify",
                             data=dump_ticket, headers=headers)
    if response.json() == "True":
        return True
    else :
        print("this key is not verified with master public key")
        return False
else :
    print("this key is not verified with follower public key")
    return False

```

#### شکل ۴-۷ بخش دوم تابع `check_contract`

```

if tr[0]['type'] == "pm" :
    flag = False
    for block in self.chain:
        for t in block.transactions:
            if t['type'] == "register" and t['name'] == tr[0]['receiver']
            and t['group'] == tr[0]['group'] :
                flag = True
    if not flag :
        print("in check_contract you are not in the same group")
        return False
    ticket_string = ''
    for block in self.chain:
        for t in block.transactions:
            if t['type'] == "register" and t['group'] == tr[0]['group']
            and t['name'] == tr[0]['sender'] :
                ticket_string = t['ticket']
    sign = tr[0]["sign"]
    del tr[0]["sign"]
    dump_post_obj = json.dumps(tr[0], sort_keys=True)
    h = SHA256.new(dump_post_obj.encode())
    ticket = json.loads(ticket_string)
    pubkey_str = ticket["Pubaddress"]
    pubkey = RSA.importKey(pubkey_str.encode())
    v = pubkey.verify(h.digest(), sign)
    return v

```

#### شکل ۴-۸ بخش سوم تابع `check_contract`

- mine پس از محاسبه اثبات کار و بررسی قراردادها برای بلوک آن را به زنجیره اضافه می‌کند

```
def mine(self):
    """
    This function serves as an interface to add the pending
    transactions to the blockchain by adding them to the block
    and figuring out Proof Of Work.
    """
    if not self.unconfirmed_transactions:
        return False
    if not self.check_contract(self.unconfirmed_transactions):
        return False
    last_block = self.last_block
    new_block = Block(index=last_block.index + 1,
                      transactions=self.unconfirmed_transactions,
                      timestamp=time.time(),
                      previous_hash=last_block.hash , nonce=0)
    proof = self.proof_of_work(new_block) #nonce avaz shod
    res = self.add_block(new_block, proof)
    print("in mine func res for mine block " +str(res))
    self.unconfirmed_transactions = []
    announce_new_block(new_block, proof)
    return True
```

#### شکل ۴-۹ تابع mine

با استفاده از چارچوب Flask مسیرهای برای این سرویس تعریف شده است که از این قبیل است:

- new\_transaction / اضافه کردن یک تراکنش به تراکنش‌های تایید نشده
- chain / ابتدا برای زنجیره بلندتر با هم گروه‌های خود توافق می‌کند سپس زنجیره را برمی‌گرداند
- chain۲ / زنجیره را برمی‌گرداند
- mine / برای استخراج بلوک‌هایی که هنوز به زنجیره بلوک اضافه نشده‌اند
- register\_node و register\_with / برای اضافه کردن عضو به شبکه

دیگر توابع مورد نیاز :

- Consensus برای ایجاد توافق بین اعضا شبکه به منظور زنجیره بلوک یکتا



```
def consensus():
    global blockchain
    longest_chain = None
    current_len = len(blockchain.chain)
    for node in peers:
        print("in consensus function "+'{}'.format(node))
        response = requests.get('{}'.format(node))
        res = response.json()
        length = res['length']
        chain = res['chain']
        if length > current_len and blockchain.check_chain_validity(chain):
            current_len = length
            longest_chain = chain
    if longest_chain:
        chain2 = []
        for block in longest_chain:
            block2 = Block(block["index"],
                           block["transactions"],
                           block["timestamp"],
                           block["previous_hash"],
                           block["nonce"])
            block2.hash = block['hash']
            chain2.append(block2)
        blockchain.chain = chain2
    return True
return False
```

#### شکل ۴-۱۰ تابع اجماع

- announce\_new\_block برای اعلام به اعضا شبکه که بلوک تازه استخراج شده را به زنجیره-

بلوک خود اضافه کنند

```
def announce_new_block(block, proof):
    """
    A function to announce to the network once a block has been mined.
    Other blocks can simply verify the proof of work and add it to their
    respective chains.
    """
    print("announce new block")
    for peer in peers:
        print(str("{}".format(peer)))
        url = " {}".format(peer)
        headers = {'Content-Type': "application/json"}
        block.hash = proof
        requests.post(url, data=json.dumps(block.__dict__), headers=headers)
```

#### شکل ۴-۱۱ تابع announce\_new\_block

## ۳-۴ پیاده‌سازی بخش رابط کاربری

این بخش رابط مورد نیاز که کاربر برای ساخت حساب‌ها و تعامل با دیگر اشیاء به آن نیاز دارد طراحی شده‌است. ابتدا از هر شیء برای ایجاد رابط کاربری برای او نام، پورت<sup>۱</sup> مورد نظر و آدرس سرور<sup>۲</sup> زنجیره-بلوک را درخواست می‌کند سپس رابط وب را برای اون ایجاد می‌کند. مسیرهایی که در این سرویس برای ادامه کار وجود دارد از این قبیل است:

- /registerform برای ثبت نام به عنوان مدیر با دنبال‌کننده

```
@app.route('/registerform')
```

```
def regform():
```

```
    return render_template('reg.html',
                           title='REGISTER')
```

```
@app.route('/register', methods=['POST','Get'])
```

```
def reg():
```

```
    global category,ip,group
```

```
    ip = request.form["ip"]
```

```
    category = request.form["category"]
```

```
    group = request.form["group"]
```

```
#-----register master-----
```

```
if request.method == 'POST' and ip!="" and category!="" and group!="":
```

```
    global port,name
```

```
    if(category == "master"):
```

```
        ticket = {
```

```
            "group" : group,
```

---

<sup>۱</sup> Port

<sup>۲</sup> Server

```
"name" : name,
"Pubaddress": publickey_master.exportKey('PEM').decode()
}
f = open("pubkey.pem" , 'wb')
public_key = publickey_master.exportKey('PEM')
f.write(public_key)
f.close()
ff = open("prkey.pem" , 'wb')
private_key = privatekey_master.exportKey('PEM')
ff.write(private_key)
ff.close()
fff = open("ticket.txt" , 'w')
fff.write(json.dumps(ticket))
fff.close()
zf = zipfile.ZipFile("master_keys.zip", mode='w')
zf.write("pubkey.pem")
zf.write("prkey.pem")
zf.write("ticket.txt")
zf.close()

post_object = {
    'type' : "register",
    'ip' : ip,
    'port' : port,
    'category': category,
```

```
'group': group,
'name' : name,
'ticket': json.dumps(ticket)
}

new_tx_address = "{}/new_transaction".format(miner)
requests.post(new_tx_address,
              json=post_object,
              headers={'Content-type': 'application/json'})
#-----register follower-----
elif (category == "follower"):
    if 'ticket' not in request.files:
        return redirect('/registerform')
    ticket_txt = request.files.get('ticket')
    ticket = ticket_txt.read()
    post_object = {
        "type" : "register",
        "ip" : ip,
        "port" : port,
        "category": category,
        "group": group,
        "name" : name,
        "ticket" : ticket.decode()
    }
    new_tx_address = "{}/new_transaction".format(miner)
```

```

requests.post(new_tx_address,
              json=post_object,
              headers={'Content-type': 'application/json'})
#request to mine
mine_address = "{ }/mine".format(miner)
response = requests.get(mine_address)
if response.json()=="True" :
    if category == "master":
        return redirect('/master')
    else:
        return redirect('/follower')
else :
    return redirect('/registerform')

```

- `/getticket` برای تولید بلیط برای دنبال کننده توسط مدیر گروه

```

@app.route('/getticket')
def get_ticket():
    return render_template('ticket.html',
                          title='GET TICKET')
@app.route('/ticket', methods=['POST','Get'])
def ticket():
    name = request.form["name"]
    global category
    if name == "" :
        return render_template('show_ticket.html',
                              title='please type your name')

```

---

```

if category != "master" :
    return render_template('show_ticket.html',
                           title='I AM NOT ROOT')

modulus_length = ۲۵۶*۸

privatekey = RSA.generate(modulus_length, Random.new().read)
publickey = privatekey.publickey()

ticket = {
    "group" : group,
    "name" : name,
    "Pubaddress": publickey.exportKey('PEM').decode()
}

json_dumps = json.dumps(ticket,sort_keys=True)
h = SHA۲۵۶.new(json_dumps.encode())

K = "
master_sign = privatekey_master.sign(h.digest(),K)
ticket["master_sign"] = master_sign
json_dumps۲ = json.dumps(ticket,sort_keys=True)

h۲ = SHA۲۵۶.new(json_dumps۲.encode())

K = "
follower_sign = privatekey.sign(h۲.digest(),K)
ticket["follower_sign"] = follower_sign

f = open("pubkey.pem" , 'wb')

```

```
public_key = publickey.exportKey('PEM')
f.write(public_key)
f.close()
```

```
ff = open("prkey.pem", 'wb')
private_key = privatekey.exportKey('PEM')
ff.write(private_key)
ff.close()
```

```
fff = open("ticket.txt", 'w')
fff.write(json.dumps(ticket))
fff.close()
```

```
zf = zipfile.ZipFile("keys.zip", mode='w')
zf.write("pubkey.pem")
zf.write("prkey.pem")
zf.write("ticket.txt")
zf.close()
return send_file("keys.zip")
```

• `/verify` برای تایید بلیط یک دنبال کننده توسط مدیر

```
@app.route('/verify', methods=['POST', 'Get'])
def verify_ticket():
    ticket = request.get_json()
    master_sign = ticket['master_sign']
    del ticket['master_sign']
```

```
dump_ticket = json.dumps(ticket,sort_keys=True)
h = SHA۲۵۶.new(dump_ticket.encode())

Mpubkey = RSA.importKey(publickey_master.exportKey('PEM'))
v = Mpubkey.verify(h.digest(),master_sign)
return json.dumps(str(v))
```



• `/submit` برای ارسال پیام به دیگر اشیاء

```
@app.route('/submit', methods=['POST','Get'])
```

```
def submit():
```

```
    content = request.form["content"]
```

```
    receiver = request.form["receiver"]
```

```
    global category,name,group
```

```
    if 'prkey' not in request.files or content=="" or receiver=="":
```

```
        if category == "master" :
```

```
            return redirect('/master')
```

```
        else :
```

```
            return redirect('/follower')
```

```
    post_object = {
```

```
        "type" : "pm",
```

```
        "sender" : name,
```

```
        "group" : group,
```

```
        "receiver" : receiver,
```

```
        "content" : content
```

```
    }
```

```
    prkey = request.files.get('prkey')
```

```
    privatekey = RSA.importKey(prkey.read())
```

```
    json_dumps = json.dumps(post_object,sort_keys=True)
```

```
h = SHA۲۵۶.new(json_dumps.encode())

K = "

sign = privatekey.sign(h.digest(),K)
post_object["sign"] = sign

# Submit a transaction

new_tx_address = "{ }/new_transaction".format(miner)

requests.post(new_tx_address,
              json=post_object,
              headers={'Content-type': 'application/json'})

#request to mine
mine_address = "{ }/mine".format(miner )
response = requests.get(mine_address)
if response.json()=="True" :
    if category == "master":
        return redirect('/master')

    else:
        return redirect('/follower')
else :
    return render_template('pm.html',
                          title='Your pm did not mine successfully')
```

## فصل پنجم

### راه‌اندازی و آزمون سامانه

در این فصل به نحوه راه اندازی سامانه و آزمون عملکرد آن می پردازیم.

## ۵-۱ راه اندازی سامانه

برای راه اندازی از سیستم عامل لینوکس استفاده می کنیم. هر شیء برای عضو شدن در این سامانه ابتدا باید زنجیره بلوک خود را ایجاد کند. برای ایجاد زنجیره بلوک باید فایل `server.py` را با دستورات زیر اجرا کند. می تواند پورت دلخواه خود را وارد کند.

```
Export FLASK_APP=server.py
```

```
Flask run -port ۸۰۰۰
```

حالا باید به بقیه اعضا در سامانه متصل شود. به طور مثال اگر سرویس دیگر در آدرس `http://۱۲۷,۰,۰,۱:۸۰۰۱` فعال باشد :

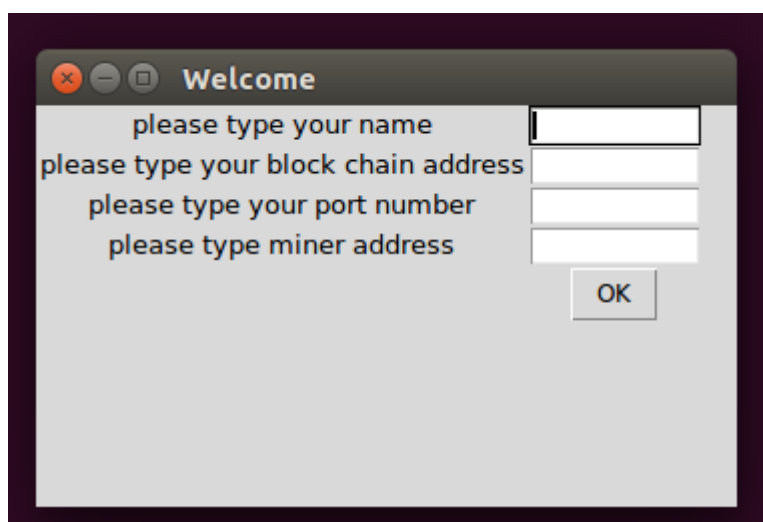
```
Curl -X post http://۱۲۷,۰,۰,۱:۸۰۰۱/register_with -H 'content-type:application/json' -d ' "node_address": "http://۱۲۷,۰,۰,۱:۸۰۰۰" '
```

منظور از این دستور این است که سرویس در آدرس `node_address` آدرسی به بعد از `post` آمده را به اعضا اضافه می کند. یکبار هم برعکس این دستور باید اجرا شود.

حال برای این سرویس زنجیره بلوک به یک رابط کاربری احتیاج داریم که بدین منظور با استفاده از دستور زیر به اجرای دستور زیر می پردازیم.

```
Python۳ node.py
```

با اجرای این دستور پنجره زیر باز می شود.



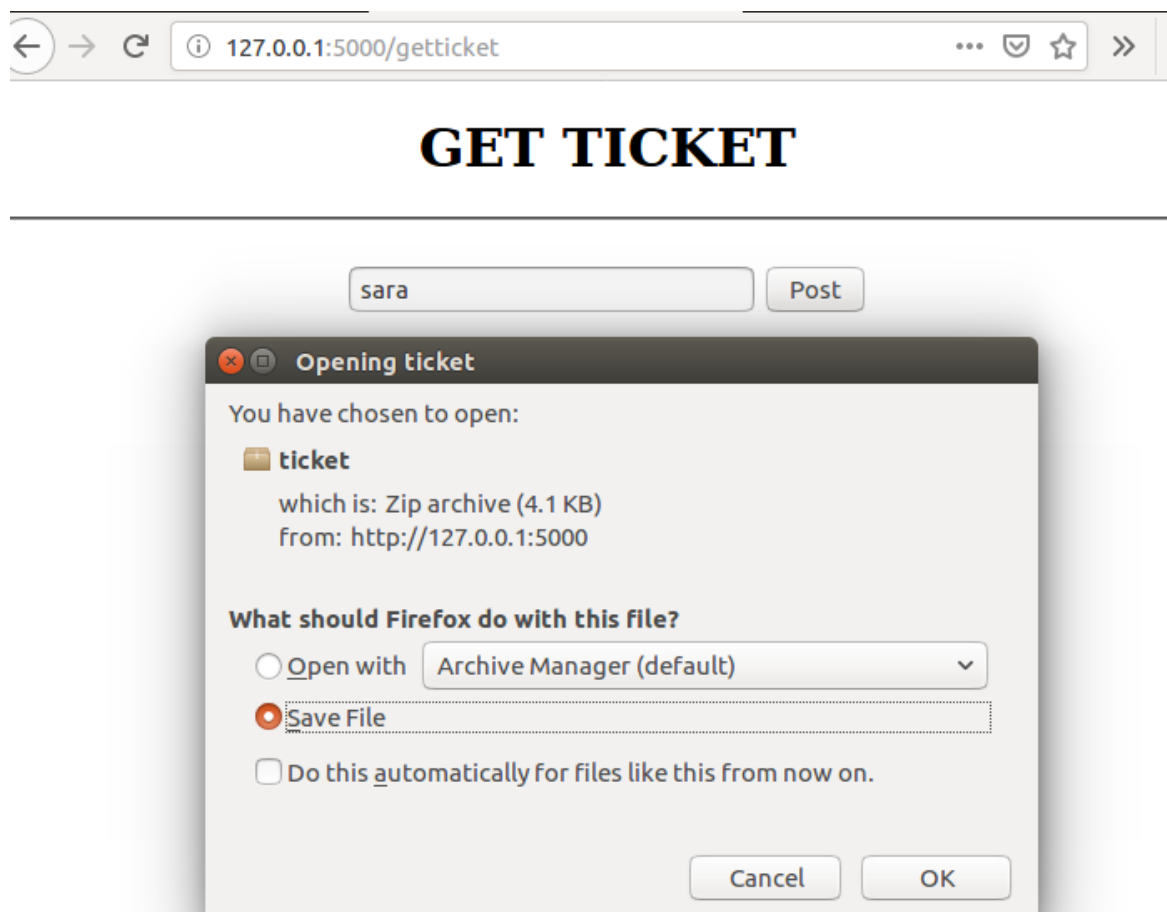
شکل ۵-۱ پنجره راه اندازی اولیه رابط کاربری

حالا در مرورگر خود به آدرس `/registerform` می رویم و به عنوان مدیر یا دنبال کننده ثبت نام می-کنیم.

شکل ۵-۲ صفحه ثبت نام

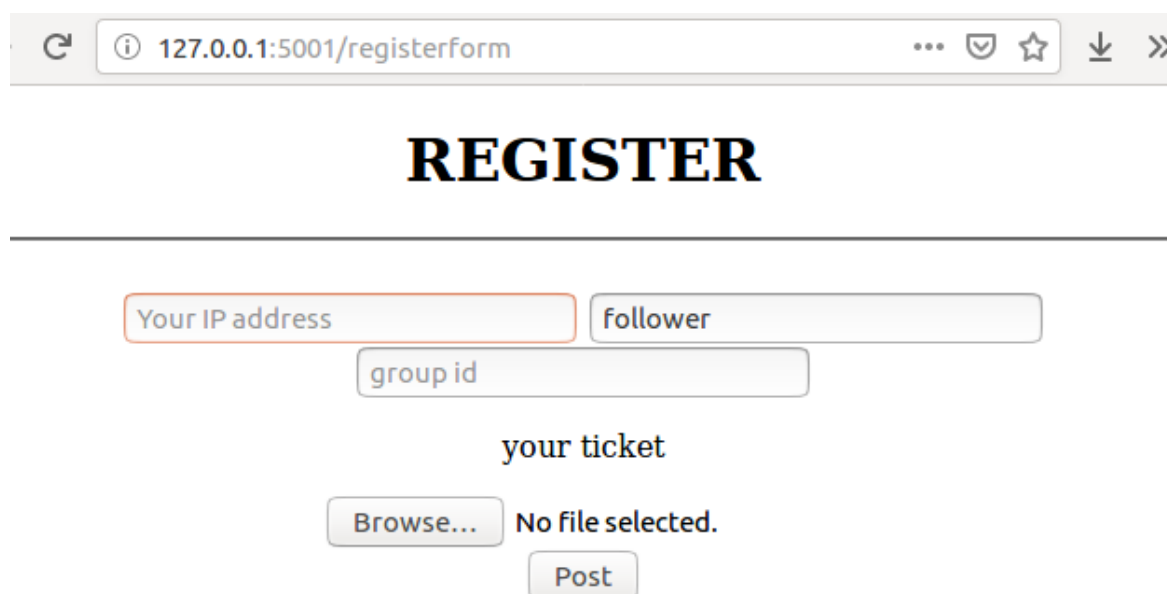
برای ثبت نام باید ip را وارد کرد سپس مشخص کرد که به عنوان مدیر ثبت نام شود یا دنبال کننده و در آخر نام گروه مورد نظر باید وارد شود. اگر به عنوان مدیر ثبت نام انجام شود یک فایل با نام `master_keys.zip` در کامپیوتر ذخیره می شود. به منظور ثبت نام به عنوان دنبال کننده ابتدا باید از مدیر آن گروه بلیط ورود دریافت کرد که برای دریافت باید به آدرس `/getticket` برویم. با وارد کردن

نام خود در این آدرس یک فایل فشرده شامل بلیط و کلید خصوصی و عمومی در اختیار فرد قرار می-گیرد.



شکل ۵-۳ صفحه گرفتن بلیط از مدیر گروه

حال می توان به عنوان دنبال کننده ثبت نام کرد.



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5001/registerform". The page title is "REGISTER". The form contains the following elements:

- A text input field labeled "Your IP address" with a red border.
- A text input field labeled "follower".
- A text input field labeled "group id".
- A label "your ticket" below the "group id" field.
- A "Browse..." button next to the text "No file selected."
- A "Post" button at the bottom.

شکل ۴-۵ صفحه ثبت نام به عنوان دنبال کننده

پس از ثبت نام، هر شیء می تواند به عنوان مدیر یا دنبال کننده به دیگر اعضای که در یک گروه قرار دارند پیام بدهد.

## FOLLOWER

Just write whatever you want to...

No file selected.

submit yout private key to sign your message

Resync

send:

S

sender : sara  
 receiver : maryam

hello master maryam i am sara

شکل ۵-۵ صفحه ارسال پیام

## ۲-۵ آزمون سامانه

آزمون کلی راه اندازی سامانه در بخش قبل توضیح داده شد. در این بخش به بررسی برخی موارد جزئی می پردازیم.

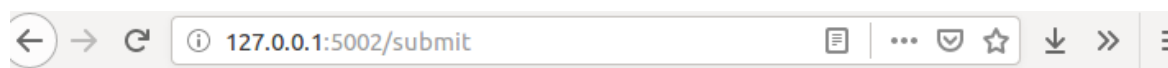
### □ بررسی عدم برقراری ارتباط برای اشیاء در حساب های غیر یکسان

سه شیء به نام های bb , aa و cc در نظر بگیرید. شیء aa یک حساب با نام a دارد و دو شیء دیگر در حسابی به نام b هستند. مشاهده می شود شیء cc نمی تواند به شیء aa پیام بفرستد.

```
127.0.0.1 - - [17/Aug/2019 01:45:43] "GET /chain2 HTTP/1.1" 200 -
127.0.0.1 - - [17/Aug/2019 01:45:43] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [17/Aug/2019 01:47:26] "POST /new transaction HTTP/1.1" 201 -
in check contract you are not in the same group
in /mine result of mine is False
127.0.0.1 - - [17/Aug/2019 01:47:26] "GET /mine HTTP/1.1" 200 -
```

شکل ۵-۶ خروجی سامانه برای ارسال پیام به شیء در حساب دیگر





## Your pm did not mine successfully

Just write whatever you want to...

type receiver name submit your private key to sign your message

Browse... No file selected. Post

Resync

send:

receive:

شکل ۵-۷ صفحه ارسال پیام بعد نفرسادن پیام به گروه دیگر

□ بررسی عدم ثبت نام به عنوان دنبال کننده در صورت نداشتن بلیط

برای این قسمت مشاهده می شود که اگر درخواست برای عضو در یک حساب را ثبت کنیم در صورتی که بلیط ورود نداریم.

```
this key is not verified with master public key
in /mine result of mine is False
127.0.0.1 - - [17/Aug/2019 02:15:10] "GET /mine HTTP/1.1" 200 -
```

شکل ۵-۸ خروجی سامانه برای ثبت نام بدون بلیط

□ بررسی عدم ثبت گروه در صورت تکراری بودن نام

اگر یک شیء گروهی را به یک نام ثبت نام کند شیء دیگر نمی تواند با همان نام گروهی ثبت کند.

```

in check_contract this was repetetive
in /mine result of mine is False
127.0.0.1 - - [17/Aug/2019 02:41:08] "GET /mine HTTP/1.1" 200 -

```

شکل ۵-۹ خروجی سامانه در صورت تکراری بودن نام گروه

### ۵-۳ بررسی عملکرد سامانه بر بستر اینترنت اشیا

در این بخش به ارزیابی سامانه از منظرهای زمان اجرا و مصرف انرژی می پردازیم.

با بررسی آزمون های انجام شده، برای پیاده سازی این سیستم با استفاده از زنجیره بلوک اتریوم<sup>۲۸</sup> نتایج زیر بدست آمده است. این آزمون ها با استفاده از یک لبتاب و یک برد Raspberry PI<sup>۲۹</sup> صورت گرفته است.

جدول ۵-۱ نتایج آمارگیری ها

Table 4 – Statistics (Average and Standard Deviation) of the obtained results.												
Node type	Assoc time (ms)		Data msg time (ms)		CPU pow assoc (mWatt)		CPU pow data msg (mWatt)		NIC pow assoc (mWatt)		NIC pow data msg (mWatt)	
	Av.	SD	Av.	SD	Av.	SD	Av.	SD	Av.	SD	Av.	SD
Raspberry PI	28.03	0.045	0.82	0.029	64.16	8.19	16.29	1.10	89.24	14.01	31.22	15.11
Laptop	1.56	0.13	0.04	0.001	9.76	2.04	3.35	0.87	16.14	2.69	12.54	4.51

○ زمان مصرفی

میانگین زمان لازم برای تحقق درخواست ثبت نام برای لپ تاپ ۱,۵۶ ms است. Raspberry Pi به زمان بیشتری معادل با ۲۸,۰۳ ms نیاز دارد. در مقایسه با این نتایج، ارسال پیام زمان کمتری را صرف می کند. دلیل این تفاوت در پیچیدگی درخواست ثبت نام در مقایسه با عملیات ارسال پیام است.

<sup>۲۸</sup> Ethereum

<sup>۲۹</sup> Board

## ○ مصرف انرژی

Raspberry Pi برای تحقق درخواست ثبت نام ۶۴,۱۶ مگاوات ساعت مصرف می کند در حالی که تنها ۹,۷۶ مگاوات از طریق لپ تاپ محاسبه می شود. برای ارسال پیام، Raspberry Pi به ۱۶,۲۹ مگاوات ساعت نیاز دارد در حالی که لپ تاپ به ۳,۳۵ مگاوات نیاز دارد. این تفاوت ها به دلیل پیچیدگی درخواست ثبت نام در مقایسه با ارسال پیام ساده است.

ستون های ۱۰-۱۳ جدول بالا توصیفی از انحراف متوسط و استاندارد مصرف انرژی مورد نیاز کنترلر رابط شبکه<sup>۳۰</sup> برای تحقق درخواست ثبت نام و برای ارسال پیام داده است. Raspberry Pi برای اجرای درخواست انجمن، ۸۹,۲۴ مگاوات ساعت نیاز دارد در حالی که لپ تاپ به ۱۶,۱۴ مگاوات نیاز دارد. برای ارسال پیام، Raspberry Pi ۳۱,۲۲ مگاوات و لپ تاپ ۱۲,۵۴ مگاوات مصرف می کند. این امر تأیید می کند که ارتباطات شبکه پرهزینه ترین عملیات برای یک سیستم است.

<sup>۳۰</sup> Network Interface Controller (NIC)

## فصل ششم

### جمع‌بندی و پیشنهادات

در این فصل به جمع‌بندی و ارائه پیشنهاداتی برای ادامه کار می‌پردازیم.

## ۶-۱ جمع‌بندی و پیشنهادات

اینترنت اشیاء و برنامه‌های کاربردی آن به سرعت بخشی از زندگی روزمره ما می‌شوند. در واقع، استفاده از آن رو به افزایش است، که منجر به ظهور بسیاری از دستگاه‌ها و خدمات اینترنت اشیاء می‌شود. هر دستگاه باید قابل دسترسی باشد و محتوایی تولید کند که بدون توجه به موقعیت مکانی خود، توسط هر کاربر مجاز قابل بازیابی باشد. در بسیاری موارد، دسترسی به این دستگاه‌ها و مبادلات ارتباطی آن‌ها باید ایمن باشد. در این مقاله، ما یک رویکرد اصلی به نام حباب اعتماد را پیشنهاد دادیم که در آن مناطق مجازی ایمن ایجاد می‌شود، در جایی که دستگاه‌ها می‌توانند به روشی کاملاً مطمئن ارتباط برقرار کنند. حباب‌های اعتماد می‌توانند در زمینه‌ها، خدمات و سناریوهای بی‌شمار اینترنت اشیاء اعمال شوند. این سیستم متکی به یک زنجیره‌بلوک است، از این رو از تمام خصوصیات امنیتی بهره‌مند است. علاوه بر این، ما الزامات امنیتی را تعریف کردیم که یک طرح تصدیق اصالت در اینترنت اشیاء باید از آن مطمئن شود. ارزیابی رویکرد ما توانایی آن در تحقق الزامات امنیتی درخواستی و همچنین مقاومت آن در برابر حملات را نشان می‌دهد.

برای کارهای بعدی، ما قصد داریم (۱) سیستم را تحول دهیم تا امکان برقراری ارتباط کنترل شده بین یک مجموعه حباب انتخابی فراهم شود. (۲) پیاده سازی مکانیسم ابطال برای دستگاه‌های به خطر افتاده. و (۳) مطالعه و طراحی پروتکل با هدف بهینه سازی تعداد استخراج‌کننده در یک سیستم مشخص.

## منابع و مراجع

- [١] Hammi, M.T., Hammi, B., Bellot, P. and Serhrouchni, A., ٢٠١٨. Bubbles of Trust: A decentralized blockchain-based authentication system for IoT. Computers & Security, ٧٨, pp.١٢٤-١٤٢.
- [٢] Huh, S., Cho, S. and Kim, S., ٢٠١٧, February. Managing IoT devices using blockchain platform. In ٢٠١٧ ١٩th international conference on advanced communication technology (ICACT)(pp. ٤٤٤-٤٤٧). IEEE.
- [٣] Flask Tutorial, viewed ٢٧ July ٢٠١٩,  
<https://www.tutorialspoint.com/flask/index.htm>
- [٤] How to Become a Blockchain Developer, viewed ٢ May ٢٠١٩,  
<https://www.bitdegree.org/tutorials/blockchain-developer/>



**Amirkabir University of Technology  
(Tehran Polytechnic)**

**Computer Engineering and Information Technology Department**

**BSc Thesis**

**A decentralized blockchain-based authentication system  
for IoT**

**By  
Maryam Ebrahimzadeh**

**Supervisor  
Dr. Babak Sadeghian**

**August ۲۰۱۹**