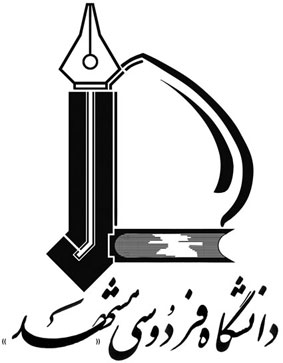
**دانشکده مهندسی**

**پایان‌نامه کارشناسی ارشد**

**گروه مهندسی کامپیوتر**

** تحلیل، طراحی و پیاده‌سازی وب سایت مربوط به زنجیره تامین و شبکه جهانی حمل و نقل- بخش بلاکچین**

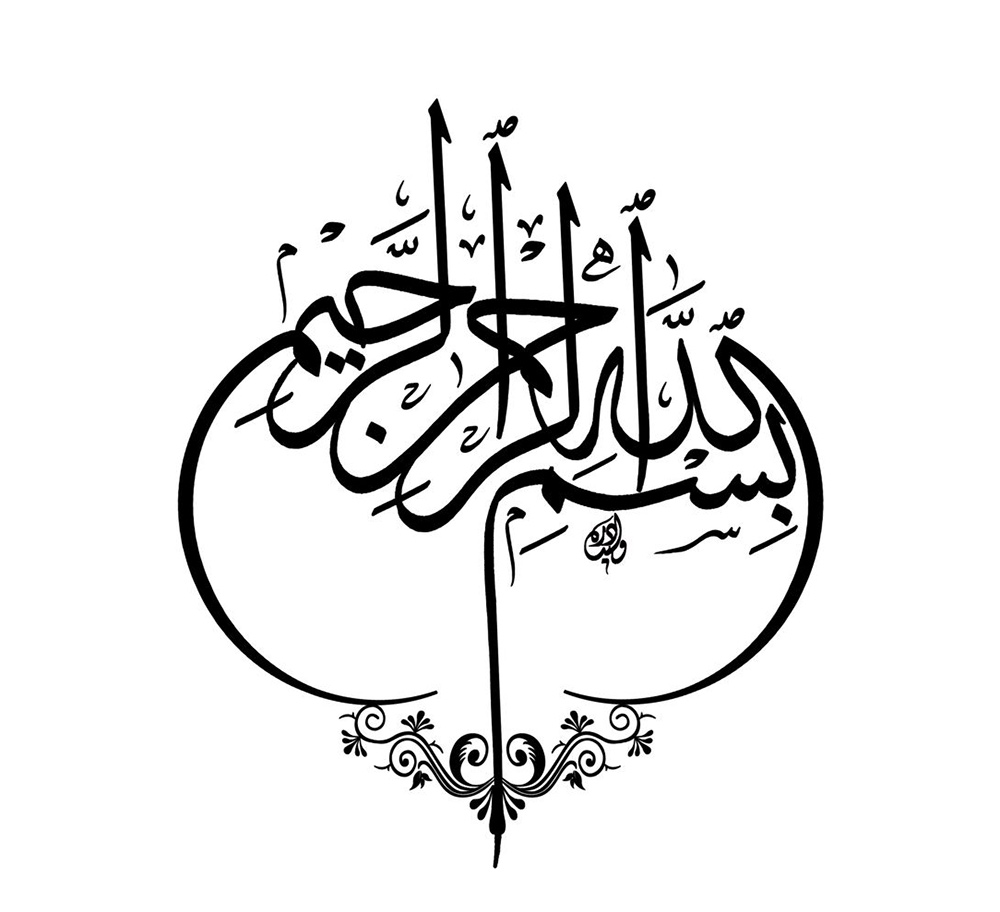
**نگارنده:**

سارا بلوری بزاز

**استاد راهنما:**

دکتر عباس رسول‌زادگان

پاییز 1401

****

اصالت نامه

فرم ارزشیابی

تقدیم به

مقدس­ترین واژه­ها در لغت­نامه دلم

مادر مهربانم که زندگیم را مدیون مهر و عطوفت او هستم

پدرم، مهربانی مشفق، بردبار و حامی

همسرم که نشانه لطف الهی در زندگی من است

**تقدیر و تشکر**

حمد و سپاس سزاوار خداوندی است که مرا نعمت هستی بخشید و در مسیر آموختن علم قرار داد. در این مسیر با اساتیدی فرهیخته، صبور و با اخلاق آشنایم ساخت. هر چند در مقام قدردانی از زحمات ایشان زبان قاصر و دست ناتوان است، اما بر خود لازم می­دانم از زحمات و راهنمایی­های استاد گرانقدر جناب آقای دکتر عباس رسول­زادگان قدردانی و تشکر نمایم چرا که بدون راهنمایی‌ها و دلسوزی‌های ایشان گردآوری این پایان‌نامه امکان‌پذیر نبود. همچنین از راهنمایی­ها و کمک­های همه اعضا و دوستان محترم آزمایشگاه کیفیت نرم­افزار نیز صمیمانه کمال تشکر و قدردانی را دارم.

|  |  |  |  |
| --- | --- | --- | --- |
| **arm1بسمه تعالي**  **مشخصات رساله/پايان نامه تحصيلي دانشجويان**  **دانشگاه فردوسي مشهد** | | | |
| **عنوان رساله/پايان نامه: تحلیل، طراحی و پیاده‌سازی وب سایت مربوط به زنجیره تامین و شبکه جهانی حمل و نقل- بخش بلاکچین** | | | |
| **نام نويسنده: سارا بلوری بزاز**  **نام استاد(ان) راهنما: جناب آقای دکتر عباس رسول­زادگان**  **نام استاد(ان) مشاور: --** | | | |
| **دانشكده: مهندسی** | **گروه:کامپیوتر** | | **رشته تحصيلي: نرم افزار** |
| **تاريخ تصويب:** | | | **تاريخ دفاع:** |
| **مقطع تحصيلي: كارشناسي ارشد** | | | **تعداد صفحات:** |
| **چکیده رساله/پایان‌نامه:**  امروزه جهان شاهد پیشرفت‌های رون افزونی در زمینه کامپیوتری است. فناوری بلاکچین یکی از فناوری‌های به روز دنیا است که مفاهیم اولیه از قرن 20 میلادی به وجود آمده است. فناوری بلاکچین یک سیستم توزیع‌شده و غیرمتمرکز است که در زمینه‌های مختلف علمی و صنعتی کاربرد بسیاری داشته است. به دلیل ویژگی غیرمتمرکز بودن این فناوری، به کمک سازوکارهای اجماع حالت نهایی سیستم توسط شرکت‌کنندگان سیستم مشخص خواهد شد. با اضافه شدن زیرساخت بلاکچین به موضوعات مختلفی مانند قرارداد هوشمند، مفاهیم جدیدی به علم کامپیوتر اضافه شده است. زبان‌های برنامه‌نویسی، ابزارها و چارچوب‌های متعددی ارائه شده است که برنامه نویسی در این زمینه را تسهیل بخشیده است. هدف از این پروژه رسیدن به یک نرم‌افزار جامع ذخیر‌ه‌سازی امن داده‌های سایر سیستم‌های نرم‌افزاری است. به این منظور، دو نمونه اولیه شبیه‌سازی بلاکچین طراحی و پیاده‌سازی شد. بعد از ارزیابی این دو نمونه، نرم‌افزار ذکر شده ارائه شد. | | | |
| **کلید واژه:**  **بلاکچین، ذخیره‌سازی امن داده، سیستم توزیع شده، قرارداد هوشمند، چارچوب Hyperledger Fabric** | | **امضای استاد راهنما: دکتر عباس رسول‎زادگان**  **امضا** | |

چکیده

امروزه جهان شاهد پیشرفت‌های رون افزونی در زمینه کامپیوتری است. فناوری بلاکچین یکی از فناوری‌های به روز دنیا است که مفاهیم اولیه از قرن 20 میلادی به وجود آمده است. فناوری بلاکچین یک سیستم توزیع‌شده و غیرمتمرکز است که در زمینه‌های مختلف علمی و صنعتی کاربرد بسیاری داشته است. به دلیل ویژگی غیرمتمرکز بودن این فناوری، به کمک سازوکارهای اجماع حالت نهایی سیستم توسط شرکت‌کنندگان سیستم مشخص خواهد شد. با اضافه شدن زیرساخت بلاکچین به موضوعات مختلفی مانند قرارداد هوشمند، مفاهیم جدیدی به علم کامپیوتر اضافه شده است. زبان‌های برنامه‌نویسی، ابزارها و چارچوب‌های متعددی ارائه شده است که برنامه نویسی در این زمینه را تسهیل بخشیده است. هدف از این پروژه رسیدن به یک نرم‌افزار جامع ذخیر‌ه‌سازی امن داده‌های سایر سیستم‌های نرم‌افزاری است. به این منظور، دو نمونه اولیه شبیه‌سازی بلاکچین طراحی و پیاده‌سازی شد.

**فهرست مطالب**

[1- مقدمه 14](#_Toc119397839)

[2- پیشینه 16](#_Toc119397840)

[2-1- فناوری بلاکچین 16](#_Toc119397841)

[2-1-1- مروری بر ساختار و معماری فناوری بلاکچین 17](#_Toc119397842)

[2-1-2- انوع بلاکچین 22](#_Toc119397843)

[2-1-3- نسخه‌های بلاکچین 25](#_Toc119397844)

[2-1-4- چالش‌های بلاکچین 26](#_Toc119397845)

[2-1-5- سیستم‌های رمزنگاری در بلاکچین 27](#_Toc119397846)

[2-1-6- سازوکار‌های اجماع در بلاکچین 31](#_Toc119397847)

[2-1-7- کاربردهای بلاکچین 34](#_Toc119397848)

[2-1-8- چارچوب‌ها و ابزارهای برنامه‌نویسی در بلاکچین 37](#_Toc119397849)

[2-2- Hyperledger Fabric 41](#_Toc119397850)

[3- راهکار پیشنهادی 45](#_Toc119397851)

[3-1- خروجی اول: شبیه‌سازی بلاکچین بر روی زیرخدمت‌های FCL و Chartering 46](#_Toc119397852)

[3-1-1- پیاده‌سازی سمت کاربر 46](#_Toc119397853)

[3-1-2- پیاده‌سازی سمت سرور 56](#_Toc119397854)

[3-2- خروجی دوم: استفاده از چارچوب Hyperledger Fabric بر روی زیرخدمت Air 58](#_Toc119397855)

[3-2-1- پیاده‌سازی سمت کاربر 58](#_Toc119397856)

[3-2-2- پیاده‌سازی سمت سرور 62](#_Toc119397857)

[3-3- خروجی سوم: نرم‌افزار جامع ذخیره‌سازی امن داده‌های سایر سیستم‌های نرم‌افزاری با استفاده از چارچوب Hyperledger Fabric 65](#_Toc119397858)

[4- ارزیابی 69](#_Toc119397859)

[5- نتیجه‌گیری و کارهای آتی 71](#_Toc119397860)

[6- راهنمای فنی 73](#_Toc119397861)

[6-1- راهنمای فنی – فرانت وبسایت Cayload 74](#_Toc119397862)

[6-2- راهنمای فنی – سمت سرور 74](#_Toc119397863)

[6-2-1- زیرخدمت FCL و Chartering – سمت سرور 74](#_Toc119397864)

[6-2-2- زیرخدمت Air – سمت سرور 75](#_Toc119397865)

[6-3- راهنمای فنی – سمت مشتری 79](#_Toc119397866)

[6-4- راهنمای فنی نرم‌افزار جامع ذخیره‌سازی داده 84](#_Toc119397867)

[مراجع 92](#_Toc119397868)

[پیوست 95](#_Toc119397869)

**فهرست شکل‌ها**

[شکل ‏2‑1: معماری توزیع شده نظیر به نظیر 18](#_Toc119397993)

[شکل ‏2‑2: معماری بلاکچین [7] 18](#_Toc119397994)

[شکل ‏2‑3: رمزنگاری متقارن 27](#_Toc119397995)

[شکل ‏2‑4: رمز با کلید خصوصی (امضا کردن) 29](#_Toc119397996)

[شکل ‏2‑5: رمز با کلید عمومی 29](#_Toc119397997)

[شکل ‏3‑1: نمایی از نرم‌افزار بلاکچین 46](#_Toc119397998)

[شکل ‏3‑2: نمونه‌ای از مفاد قرارداد FCL در وبسایت 47](#_Toc119397999)

[شکل ‏3‑3: نمودار مورد کاربرد ریزخدمت‌ FCL 48](#_Toc119398000)

[شکل ‏3‑4: صفحه ورود به نرم افزار 49](file:///C:\Users\Sara%20Bolouri\Desktop\Projects\General%20Framework\Thesis%20Doc\VS-05.docx#_Toc119398001)

[شکل ‏3‑5: بارگذاری/ذخیره‌سازی کلید خصوصی 53](#_Toc119398002)

[شکل ‏3‑6: صفحه نمایش لیست قراردادها 54](#_Toc119398003)

[شکل ‏3‑7: نمایی از اطلاعات قرارداد در قالب PDF 54](file:///C:\Users\Sara%20Bolouri\Desktop\Projects\General%20Framework\Thesis%20Doc\VS-05.docx#_Toc119398004)

[شکل ‏3‑8: نمودار کلاس زیرخدمت FCL 55](#_Toc119398005)

[شکل ‏3‑9: نمایی از نرم افزار – زیرخدمت Air 58](#_Toc119398006)

[شکل ‏3‑10: نمودار مورد کاربر زیرخدمت Air 59](#_Toc119398007)

[شکل ‏3‑11: نمودار کلاس زیرخدمت Air 61](#_Toc119398008)

[شکل ‏3‑12: شماتیک روابط بین اجزای سازنده سمت سرور 62](#_Toc119398009)

[شکل ‏3‑13: نمودار کاربرد برنامه جامع ذخیره‌سازی داده 66](#_Toc119398010)

[شکل ‏3‑14: نمودار کلاس مربوط به برنامه جامع ذخیره‌سازی داده 67](file:///C:\Users\Sara%20Bolouri\Desktop\Projects\General%20Framework\Thesis%20Doc\VS-05.docx#_Toc119398011)

[شکل ‏3‑15: نمودار پایگاه‌داده برنامه جامع ذخیره‌سازی داده 68](#_Toc119398012)

[شکل ‏4‑1: نمونه خروجی واسط برنامه‌نویسی کاربردی نرم افزار جامع ذخیره‌سازی داده 70](#_Toc119398013)

[شکل ‏6‑1: سازنده کلاس Cayload در قرارداد هوشمند 77](#_Toc119398014)

[شکل ‏6‑2: تابع configNetwork جهت برقرار ارتباط با شبکه بلاکچین 78](#_Toc119398015)

[شکل ‏6‑3: پوشه‌بندی سمت مشتری 80](#_Toc119398016)

[شکل ‏6‑4: سازنده کلاس Ui\_landing\_page() 81](#_Toc119398017)

[شکل ‏6‑5: تابع اجرایی بعد از اتمام عملیات ورود کاربر 81](#_Toc119398018)

[شکل ‏6‑6: تابع implement مربوط به زیرخدمت FCL و Chartering 83](#_Toc119398019)

[شکل ‏6‑7: تابع implement مربوط به زیرخدمت Air 84](#_Toc119398020)

[شکل ‏6‑8: صفحه ورود کاربران به نرم افزار 86](#_Toc119398021)

[شکل ‏6‑9: تابع signup\_page\_show جهت نمایش صفحه ثبت نام مدیر سایت 86](#_Toc119398022)

[شکل ‏6‑10: صفحه ورود کاربر و ثبت نام 86](file:///C:\Users\Sara%20Bolouri\Desktop\Projects\General%20Framework\Thesis%20Doc\VS-05.docx#_Toc119398023)

[شکل ‏6‑11: تابع signup جهت ثبت نام مدیر سایت 88](#_Toc119398024)

[شکل ‏6‑12: بخش‌های مختلف پنل مدیریت در برنامه جامع ذخیره‌سازی داده 89](file:///C:\Users\Sara%20Bolouri\Desktop\Projects\General%20Framework\Thesis%20Doc\VS-05.docx#_Toc119398025)

**فهرست جدول‌ها**

[جدول ‏2‑1: نیازمندی‌های پروژه جهت ذخیره‌سازی امن داده‌ها 39](#_Toc119398150)

[جدول ‏2‑2: مقایسه چارچوب‌های معروف در فناوری بلاکچین 40](#_Toc119398151)

[جدول ‏2‑3: مقایسه سه چارچوب از منظر فنی در فناوری بلاکچین 40](#_Toc119398152)

[جدول ‏3‑1: نگاشت توابع به سند 55](#_Toc119398153)

[جدول ‏3‑2: نگاشت توابع به سند 61](#_Toc119398154)

[جدول ‏3‑3: نگاشت توابع قرارداد هوشمند نصب شده بر روی بلاکچین 63](#_Toc119398155)

[جدول ‏3‑4: نگاشت توابع مربوط به برنامه واسط 63](#_Toc119398156)

[جدول ‏3‑5: نگاشت توابع در راه‌اندازی شبکه بلاکچین 64](#_Toc119398157)

[جدول ‏3‑6: نگاشت توابع مربوط به برنامه جامع ذخیره‌سازی داده 67](#_Toc119398158)

[جدول ‏4‑1: واسط برنامه‌نویسی کاربردی ارزیابی نرم‌افزار جامع ذخیره‌سازی امن داده 70](#_Toc119398159)

[جدول ‏6‑1: آدرس فایل فرانت زیرخدمت‌ها 74](#_Toc119398160)

[جدول ‏6‑2: آدرس فایل سرور زیرخدمت‌ها 74](#_Toc119398161)

[جدول ‏6‑3: نگاشت توابع سمت سرور 75](#_Toc119398162)

[جدول ‏6‑4: نگاشت توابع قرارداد هوشمند 77](#_Toc119398163)

[جدول ‏6‑5: نگاشت توابع سمت سرور زیرخدمت Air 79](#_Toc119398164)

[جدول ‏6‑6: نگاشت توابع اجرایی بعد از ورود موفقیت آمیز کاربر 82](#_Toc119398165)

[جدول ‏6‑7: نگاشت توابع مربوط به دکمه‌های نرم افزار تحت دستکتاپ 82](#_Toc119398166)

[جدول ‏6‑8: نگاشت توابع فرایند امضا کردن قرارداد در زیرخدمت FCL و Chartering 83](#_Toc119398167)

[جدول ‏6‑9: نگاشت توابع فرایند امضا کردن قرارداد و ارسال به شبکه بلاکچین Hyperledger Fabric 84](#_Toc119398168)

# مقدمه

امروزه جهان شاهد پیشرفت سریع علم کامپیوتر، کشفیات و اختراعات روز افزون است. یکی از فناوری‌های جدید اختراع شده، بلاکچین است که اساس آن از سال‌های 1980 میلادی بنیان گذاری شده و انقلابی در صنعت مالی و غیرمالی و هم در حوزه‌های مختلف علم کامپیوتر ایجاد کرده است.

فناوری بلاکچین به سیستم‌های توزیع شده و غیرمتمرکز گفته می‌شود. این فناوری کاربردهای بسیاری در زمینه‌های مالی، بهداشت و سلامت، اثبات مالکیت، اینترنت اشیاء، نگهداری و ذخیره‌سازی اطلاعات دارد و دارای ویژگی‌هایی از جمله غیرمتمرکز بودن، تغییرناپذیری، شفافیت و امنیت دارد. از آنجایی که بلاکچین یک سیستم توزیع شده است، لازم است شرکت‌کنندگان در این سیستم بر سر حالت سیستم به یک توافق یکسان برسند؛ چرا که هیچ عضو ثالثی جهت تعیین حالت سیستم وجود ندارد. از این رو در این فناوری از سازوکارهای اجماع جهت به توافق رسیدن اعضا، استفاده شده و کارایی این سیستم‌ها، با توجه به سازوکار انتخاب شده مشخص می‌شود.

از شروع اختراع بلاکچین، این فناوری پیشرفت‌هایی داشته است. از این زیرساخت در موضوعات مختلف از جمله قراردادهای هوشمند استفاده شده است. در راستای این کاربرد، مفاهیم جدیدی از جمله رمزهای غیرقابل معاوضه[[1]](#footnote-1) و برنامه‌های غیرمتمرکز[[2]](#footnote-2) که هر یک در صنایع مختلفی گسترش پیدا کرده‌اند، به وجود آمده است.

در این زمینه‌ها، زبان‌های برنامه‌نویسی، ابزارها و چارچوب‌های متعددی به وجود آمده است که توسعه‌دهنگان جهت تسهیل کار خود از آنها استفاده می‌کنند. همچنین در حال حاضر پروژه‌ها و سکو‌های مختلفی در زمینه بلاکچین در حال فعالیت و توسعه هستند که بسیاری از آنها در زمینه‌های مالی فعالیت دارند.

هدف این پروژه رسیدن به یک نرم‌افزار جامع ذخیره‌سازی امن داده است. به این منظور، ابتدا برای فهم بهتر مفاهیم و ایده‌های اصلی بلاکچین، یک نمونه اولیه بلاکچین جهت ذخیره‎‌سازی قراردادها، شبیه‌سازی شده است. این نمونه اولیه بر روی دو زیرخدمت FCL و Chartering در پروژه بین‌المللی حمل و نقل اعمال شده است.‌ سپس بعد از بررسی‌های انجام شده، ایرادهایی به آن وارد بود. به این منظور تصمیم گرفته شد تا از چارچوب‌های آماده جهت استفاده از فناوری بلاکچین استفاده شود. از این رو، بعد از مقایسه‌های انجام شده، چراچوب Hyperledger Fabric به عنوان چارچوب انتخاب شد و لازم بود تا نمونه اولیه‌ای دیگری پیاده‌سازی شود تا عملکرد این چارچوب مشخص شود. این نمونه اولیه بر روی زیرخدمت Air در پروژه بین‌المللی حمل و نقل اعمال شد و درستی آن ارزیابی شد. در انتها نرم‌افزار جامع ذخیره‌سازی امن داده‌های سایر سیستم‌های نرم‌افزاری ارائه شده است.

# پیشینه

در این بخش به بررسی پیشینه‌ و ادبیات حوزه فناوری بلاکچین پرداخته شده است. مطالعه مقالات مروری و پروژه‌های پیاده‌سازی شده در این حوزه نشان می‌دهد که طی چند سال اخیر فناوری بلاکچین رشد زیادی نه تنها در حوزه‌ی ادبیات، بلکه در صنایع مختلف داشته است. در ادامه به تاریخچه بلاکچین، نقاط مثبت و منفی آن، کاربردهای آن و ابزارهای موجود در این فناوری پرداخته خواهد شد.

## 2-1- فناوری بلاکچین

مفاهیم و ایده‌های اصلی بلاکچین از سال‌ها قبل به وجود آمده است. از ابتدای سال‌های 1980 ایده‌هایی از جمله پایگاه داده‌های توزیع شده، نگهداری سوابق داده‌ها، زنجیره تغییر ناپذیر اطلاعات در قالب درخت مرکل[[3]](#footnote-3)، اسناد زمانبند و رسیدن به اجماع دسته جمعی ارائه شده است [1]. بنابراین می‌توان گفت ریشه فناوری بلاکچین به قرن 20 میلادی برمی‌گردد و به مرور زمان با تجمیع این مفاهیم و ایده‌ها، بلاکچین رشد کرده است. مفهوم بلاکچین با به وجود آمدن ارز دیجیتال بیتکوین در سال 2008 توسط فردی ناشناس به نام مستعار ساتوشی ناکاموتو[[4]](#footnote-4) ارائه و در سال 2009 پیاده سازی شد [1] [2].

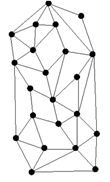
در اواخر سال 1990 اولین ایده‌ی ارز دیجیتال با کمک سرور مرکزی ایجاد شد؛ اما بدلیل مشکلات موجود در آن، ایده‌ی موفقیت آمیزی نبود. بعد از حدود 10 سال ارز دیجیتال بیتکوین ارائه شد و در سال 2008 کاغذ سفید[[5]](#footnote-5) ارائه شده توسط ساتوشی ناکاموتو باعت معروفیت آن شد. ارز دیجیتال معرفی شده مبتنی بر سازوکار اجماع بود. در سال‌های بین 2011 تا 2013 کاربرد بلاکچین در پرداخت‌های دیجیتالی و انتقال ارزهای دیجیتال بود؛ در حالی که امروزه بلاکچین در صنایع مختلفی با کاربردهای مختلف مورد استفاده قرار می‌گیرد [2].

از شروع اختراع بلاکچین، این فناوری پیشرفت‌هایی داشته است. ابتدا در سال 2009 با اختراع ارز دیجیتال بیتکوین، فناوری بلاکچین به عرصه علم کامپیوتر وارد شد. در سال 2013 به دلیل اختلاف نظرات پیش آمده در اضافه کردن قابلیت جدید به بیتکوین، یکی از توسعه دهندگان به نام ویتالیک بوترین[[6]](#footnote-6) با در نظر داشتن قابلیت جدید، ارز دیجیتال اتریوم را ارائه کرد؛ به عبارتی از زیرساخت بیتکوین که همان بلاکچین است ارز دیجیتال اتریوم را ارائه کرد، با این تفاوت که قابلیت جدیدی به نام قراردادهای هوشمند را به آن اضافه کرد.

تا به امروز بیتکوین به طرز چشم‌گیری مورد استقبال مردم قرار گرفته است و طبق ارقام اعلام شده، سایز زنجیره بیتکوین از سال 2009 تا به الان از صفر تا 240 گیگا­­بایت رشد داشته است [3]. همچنین طبق آخرین آمار اعلام شده در سال 2021، تا به امروز حدود 7000 ارز دیجیتال ارائه شده است که این تعداد خود نمایانگرد کاربردی بودن فناوری بلاکچین در حوزه مالی است.

### 2-1-1- مروری بر ساختار و معماری فناوری بلاکچین

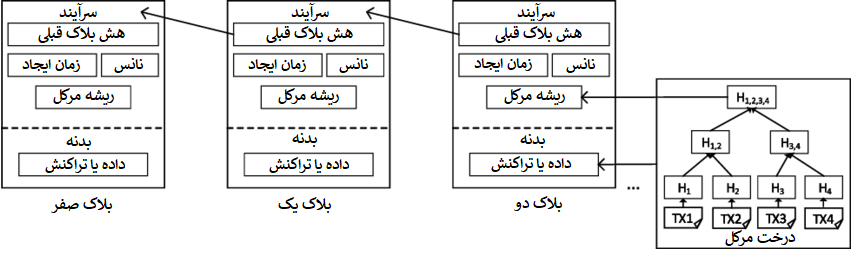
با بررسی فناوری بلاکچین در بستر ارز دیجیتال بیتکوین می‌توان آن را یک دفترکل عمومی[[7]](#footnote-7) غیرمتمرکز و توزیع شده‌ی نظیر به نظیر[[8]](#footnote-8) دانست که جهت نگهداری و مدیریت مجموعه‌ای از داده‌ها و تراکنش‌های مهم مورد استفاده قرار می‌گیرد. سیستم‌‌های توزیع شده‌ی نظیر به نظیر نوعی خاص از سیستم‌های توزیع شده هستند که در آن گره‌ها با نقشی یکسان، منابع خود را مانند توان پردازشی، حافظه و پهنای باند را در اختیار سایر گره‌های موجود در سیستم، قرار می‌دهند. در این نوع سیستم‌ها حقوق تمام گره‌ها یکسان است و هر گره می‌تواند هر دو نقش تامین‌کننده یا مصرف‌کننده را در سیستم داشته باشد.



**شکل ‏2‑1: معماری توزیع شده نظیر به نظیر**

برای نگهداری یک تراکنش لازم است تا تعداد زیادی از گره‌ها در شبکه، رضایت خود را نسبت به ثبت تراکنش مذکور اعلام کنند. تراکنش‌ها توسط فرد ایجاد کننده امضا می‌شود تا مشخص شود صاحب این تراکنش کدام گره در سطح شبکه است.

گروهی از تراکنش‌ها در یک بلوک نگهداری می‌شوند و این بلوک‌ها بصورت زنجیره‌ای به یکدیگر متصل هستند و در دفترکل ذخیره می‌شوند. برای ایجاد زنجیره و ارتباط دادن بلوک‌ها به یکدیگر لازم است که زمان ایجاد هر بلوک و هش بلوک قبلی در بلوک جدید قرار گیرد و به این صورت ارتباط بین بلوک‌ها ایجاد می‌شود [2] [4] [5] [6].



**شکل ‏2‑2: معماری بلاکچین** [7]

تابع هش یکپارچگی[[9]](#footnote-9) و غیر قابل انکار[[10]](#footnote-10) بودن بلوک را صحت سنجی می‌کند. همچنین برای اینکه زنجیره بلوک‌ها برای تمام کاربران بروز باشد، لازم است که تمام کاربران یک نسخه از زنجیره اصلی را همراه خود داشته باشند؛ همچنین با سایر کاربران همگام باشند تا با هر تغییر در زنجیره بروزرسانی را انجام دهند [2].

در ادامه به معماری بلاکچین پرداخته شده است. همانطور که گفته شد، این تکونولوژی پایگاه داده‌ای توزیع شده نظیر به نظیر و غیرمتمرکز است که هر فردی در هرکجای دنیا می‌تواند با کمک اینترنت به آن دسترسی داشته باشد. فناوری بلاکچین برخلاف مدل‌های سنتی پایگاه داده، به هیچ سازمان مرکزی وابسته نیست و از مشکل خرابی سیستم مرکزی[[11]](#footnote-11) رنج نمی‌برد؛ چرا که فناوری بلاکچین غیرمتمرکز است و توسط تعداد زیادی گره در شبکه مدیریت و نگهداری می‌شود؛ بنابراین اگر یک گره دچار اختلال در اینترنت یا خرابی سیستم شود، سیستم مبتنی بر بلاکچین دچار مشکل نخواهد شد. همچنین از آنجایی که بلاکچین توسط تعداد زیادی از افراد مدیریت و نگهداری می‌شود، ارسال داده و اسناد نامعتبر غیرممکن است زیرا که تعداد زیادی از گره‌ها، آن اسناد و مدارک را معتبر تشخیص نمی‌دهند؛ در نتیجه چنین داده و اسنادی وارد سیستم بلاکچین نخواهد شد [8].

ساختار بلاکچین از بخش‌های مختلفی تشکیل شده است. هر یک از این بخش‌ها به تفکیک در زیر به اختصار توضیح داده شده است.

* **ساختار بلاکچین**
* **تراکنش**

در بستر بیتکوین، تراکنش به اسناد مالی گفته می‌شود. کاربران این شبکه می‌توانند با ایجاد یک تراکنش، مبلغی بیتکوین بین یکدیگر مبادله کنند. در دیگر بسترهای مبتنی بر بلاکچین، تراکنش را می‌توان داده‌هایی در نظر گرفت که مقصود از این کار این است که داده‌ها در بلاکچین ذخیره شود تا از تغییر و دستکاری شدن آنها جلوگیری شود. به عنوان مثال می‌توان قرارداده‌های هوشمند را نیز تراکنش در نظر گرفت. در پروژه‌ی حمل و نقل، تراکنش‌ها را می‌توان همان قراردادهایی در نظر گرفت که می‌خواهیم آنها را در زنجیره بلوک‌ ذخیره و نگهداری کنیم.

* **بلوک**

در بلاکچین، بلوک دربردارنده‌ی مجموعه‌ای از تراکنش‌های معتبر است. تراکنش‌ها توسط شرکت‌کنندگان در شبکه بلاکچین تولید می‌شود و در سطح شبکه پخش می‌شود. این تراکنش‌ها در ابتدا در استخری از تراکنش‌های تایید نشده ذخیره می‌شوند. سپس شرکت‌کنندگان بعد از تایید هر تراکنش، آن را در یک بلوک قرار می‌دهند.

بعضی از شرکت‌کنندگان در شبکه از قدرت پردازشی سیستم خود جهت حل مسئله ریاضی استفاده می‌کنند که به آنها استخراج کننده[[12]](#footnote-12) گفته می‌شود. این افراد با حل مسئله ریاضی یا دیگر سازوکارهای مربوط به رسیدن به اجماع، بلوک جدیدی را تولید می‌کند و آن را به زنجیره بلوک‌ها اضافه می‌کنند.

هر بلوک از دو بخش تقسیم شده است:

* لیست تراکنش‌ها: در بدنه هر بلوک‌، لیست تمام تراکنش‌هایی که در این بلوک قرار گرفته بودند، وجود دارد. از هر تراکنشی هش گرفته می‌شود و به این ترتیب درخت مرکل مربوط به تراکنش‌ها نیز تولید می‌شود.
* سرآیند بلوک[[13]](#footnote-13): سرآیند بلوک شامل فراداده است. فراداده‌ها شامل اطلاعاتی نظیر هش بلوک قبلی، ریشه‌ی درخت مرکل مربوط به تراکنش‌ها، زمان ایجاد بلوک و مقدار عددی نانس[[14]](#footnote-14) می‌شود.
* **سازوکار** **اجماع**[[15]](#footnote-15)

سازوکار اجماع فرایندی در علم کامپیوتر است که در جهت به توافق رسیدن بر روی یک داده در سیستم‌های توزیع شده مورد استفاده قرار می‌گیرد. این سازوکار به جهت برقراری اعتماد در شبکه‌هایی که شامل گره‌های غیرقابل اعتماد است، طراحی شده است. در فناوری بلاکچین برای رسیدن به توافق بر سر صحت و درستی تراکنش‌ها، از سازوکارهای اجماع استفاده می‌شود [9]. در بخش چهارم به بررسی جزئی تعدادی از آنها پرداخته شده است.

* **تابع هش**

تابع هش تابعی است که به کمک آن یک مجموعه داده (مجموعه‌ای از اعداد و حروف) را به اعداد و حروف با طول ثابت مطابق با جدول هش تبدیل می‌کند [10]. تا به امروز تابع هش بسیاری اختراع شده است. در فناوری بلاکچین از تابع هش SHA-512 استفاده شده است که از امنیت بالایی برخوردار است. این تابع هش، در هش کردن تراکنش‌های موجود در هر بلاک مورد استفاده قرار میگیرد.

* **امضا دیجیتال**

از آنجایی که شبکه بلاکچین غیرمتمرکز است، گره‌های موجود در شبکه ناشناس هستند. بنابراین لازم است که صاحب هر داده یا تراکنش در شبکه مشخص باشد. برای حل این مسئله، از سازوکار امضای دیجیتال استفاده شده است. به عبارت دیگر، هر گره‌ای که قصد تولید داده یا تراکنش جدیدی را داشته باشد باید آن را با کلید خود، امضا کند. همانطور که می‌دانیم امضای دیجیتال یکی از کاربردهای سازوکار رمزنگاری می‌باشد که در بخش سوم به جزئیات آن پرداخته شده است.

* **نحوه کار بلاکچین در بیتکوین**

در شبکه بلاکچین بیتکوین، گره‌ها بصورت نظیر به نظیر با یکدیگر در ارتباط هستند. در این شبکه، ابتدا تراکنش‌های ایجاد شده در یک استخر قرار می‌گیرند، سپس این تراکنش‌ها توسط گره‌ها مورد بررسی و تایید قرار می‌گیرد. مجموعه از این تراکنش‌ها داخل بلوک قرار می‌گیرد و فرایند ساخت بلوک صورت می‌گیرد. اولین گره‌ای که موفق به تولید بلوک جدید شود آن را در سطح شبکه منتشر می‌کند و سایر گره‌ها درستی بلوک تولید شده را تایید می‌کنند و پاداش خود را دریافت می‌کند. بعد از دریافت تاییدیه 51 درصدی، بلوک مذکور به زنجیره بلوک اضافه می‌شود.

### 2-1-2- انوع بلاکچین

با توجه به روش‌های مختلف جهت پیاده‌سازی برنامه‌های متفاوت مبتنی بر بلاکچین، این فناوری به سه نوع عمومی، خصوصی و ترکیبی تقسیم می‌شود که جزئیات هر یک به شرح زیر می‌باشد [11]:

**بلاکچین عمومی:** این نوع بلاکچین بلاکچینی غیرمتمرکز است و هر فرد ناشناسی می‌تواند عضو شبکه بلاکچین شود. این شبکه برای همه نوع موجودیتی باز است و اجازه‌ی ارسال تراکش جدید، ارزیابی بلوک‌های جدید و خواندن محتوای موجود در شبکه را به آنها می‌دهد [2]. این نوع بلاکچین تراکنش‌ها را بصورت شفاف و قابل دسترس ذخیره می‌کند. به عنوان مثال، می‌توان به شبکه بلاکچین بیتکوین اشاره کرد. در این نوع بلاکچین‌ها، تنها سازوکار کلید خصوصی و عمومی استفاده می‌شود و از سایر ملاک‌های دسترسی جهت کنترل و شناسایی کاربر استفاده نمی‌شود [12].

از ضعف‌های این نوع بلاکچین می‌توان به امنیت کم، قدرت محاسباتی مورد نیاز زیاد و عدم حریم خصوصی برای تراکنش‌ها اشاره کرد. این سیستم بر اساس اجماع کاربران عمل می‌کند و هیچ نهاد مرکزی جهت به اجماع رسیدن وجود ندارد [11].

**بلاکچین خصوصی:** این نوع بلاکچین مانند بلاکچین عمومی شبکه‌ای غیرمتمرکز است. اما در این شبکه تنها یک سازمان مجاز به ارسال تراکنش جدید و شرکت در مکانیزم اجماع را دارد. در این شبکه افرادی که تمایل دارند تا در این شبکه فعالیت داشته باشند باید ابتدا توسط سازمان، احراز هویت شده باشند. در این صورت داده‌ها تنها در دسترس افراد قابل اعتماد می‌باشد نه عموم مردم. بنابراین اعتماد بین شرکت‌کنندگان بیشتر است [2] [11]. به عنوان مثال می‌توان به شبکه بلاکچین ریپل اشاره کرد. در مقایسه با بلاکچین عمومی، بلاکچین خصوصی زمان و قدرت پردازشی زیادی لازم ندارد؛ همچنین حریم خصوصی بیشتری در این شبکه مهیا می‌شود [2].

**بلاکچین ترکیبی:** مدل ترکیبی از بلاکچین عمومی و خصوصی می‌باشد. با راه اندازی این مدل، هر شرکت می‌تواند شبکه بلاکچین خصوصی خود را داشته باشد و داده‌های خود را در شبکه منتشر کنند و شرکت‌کنندگان دیگر بتوانند آن داده‌ها را مشاهده کنند. به عبارتی دیگر شرکت‌کنندگانی که ثبت نام شده‌اند می‌توانند در فرایند اجماع شرکت کنند و سایرین تنها می‌توانند داده‌های موجود در شبکه را مشاهده کنند. به عنوان مثال می‌توان به پروژه‌های Hyperledger[[16]](#footnote-16) اشاره کرد [13]. در مقایسه با بلاکچین عمومی، بلاکچین ترکیبی سریعتر و مقایس‌پذیری بیشتری دارد؛ همچنین حریم خصوصی بیشتر در این نوع شبکه مهیا می‌شود [2].

ساختار و معماری بلاکچین ویژگی‌های غیرمتمرکز بودن، تغییر ناپذیری[[17]](#footnote-17)، شفافیت[[18]](#footnote-18)، امنیت و کارایی[[19]](#footnote-19) را مهیا می‌سازد. با توجه به مرجع [6]، در ادامه با توجه به ویژگی‌های مختلف، به مقایسه مدل‌های سیستم بلاکچین که در بخش قبل توضیح داده شده است، پرداخته خواهد شد.

* نحوه رسیدن به اجماع:دربلاکچین عمومی، تمامی شرکت‌کنندگان می‌توانند در شبکه جهت رسیدن به اجماع مشارکت داشته باشند؛ در حالی که در بلاکچین ترکیبی، تنها افراد انتخابی قادر به مشارکت در فرایند رسیدن به اجماع هستند. در شبکه بلاکچین خصوصی رسیدن به اجماع تنها توسط سازمان مربوطه انجام می‌شود و تمام کنترل آن را بر عهده دارد.
* نحوه نمایش داده‌ها: در بلاکچین عمومی، تمام تراکنش‌ها و داده‌ها توسط تمام شرکت‌کنندگان در شبکه قابل مشاهده و خواندن است در حالی که این موضوع در بلاکچین ترکیبی و خصوصی بستگی به تصمیم سازمان مربوطه دارد. این سازمان می‌تواند تعیین کند که داده‌ها بصورت عمومی قابل مشاهده باشد یا محدودیت‌هایی را برای آن اعمال می‌کنند.
* تغییرناپذیری: فناوری بلاکچین این توانایی را دارد تا با کمک دفترکل تغییر ناپذیر، یکپارچگی داده را مهیا سازد. برخلاف سیستم‌های سنتی مرکزی که برای یکپارچگی داده نیاز به یک شخص سومی است، در فناوری بلاکچین به دلیل اینکه هر بلوک با اضافه شدن به سیستم دیگر قابلیت تغییر ندارد، در نتیچه یکپارچگی داده مهیا می‌شود [2].

از آنجایی که زنجیره بلوک در بلاکچین عمومی توسط گره‌های مختلف نگهداری می‌شود، دستکاری کردن تراکنش‌ها و داده‌ها امکان پذیر نخواهد بود. اما در بلاکچین ترکیبی و خصوصی، از آنجایی که زنجیره بلوک توسط سازمان مربوطه مدیریت می‌شود، در صورت تمایل حداکثری سازمان‌(ها) امکان تغییر آن توسط سازمان وجود خواهد داشت.

* غیرمتمرکز بودن: شبکه غیرمتمرکز به معنای عدم مدیریت کردن شبکه توسط یک سازمان مرکزی یا فرد خاص است. در این نوع شبکه، داده‌ها توسط تمام شرکت‌کنندگان نگهداری می‌شود. این ویژگی با کمک تکنولوژی‌های رمزنگاری، امضای دیجیتال (به کمک کلیدهای نامتقارن) و سازوکار اجماع توزیع شده فعال می‌شود [6]. این ویژگی مشکلاتی نظیر خرابی سیستم مرکزی و تاخیر را ندارد [2].

مدل بلاکچین عمومی به طور کامل یک سیستم غیرمتمرکز است؛ در حالی که بلاکچین ترکیبی را می‌توان نیمه متمرکز درنظر گرفت و بلاکچین خصوصی کاملا متمرکز است و توسط یک بخش مرکزی کنترل می‌شود.

* کارایی: بلاکچین معماری متمرکز سنتی را بهبود بخشیده است. در واقع این تکنولوژی از نظر هزینه، سرعت و مدیریت ریسک کارایی بهتری را ارائه کرده است [2].

از آنجایی که در بلاکچین عمومی، بلوک‌ها و تراکنش‌ها باید برای اکثر گره‌ها ارسال شود، در نتیجه تاخیر و محدودیت‌ها را به همراه دارد؛ در حالی که به دلیل تعداد کم افراد تاییدکننده[[20]](#footnote-20) در بلاکچین ترکیبی و بلاکچین خصوصی، کارایی بهتر از بلاکچین عمومی خواهند داشت.

* ورود به شبکه: در بلاکچین عمومی، تمام افراد از سراسر جهان اجازه ورود به این شبکه را بدون نیاز به احراز هویت دارند؛ در حالی که در بلاکچین ترکیبی و بلاکچین خصوصی لازم است کاربران احراز هویت شوند و بر اساس تصمیم سازمان، افراد امکان ورود به شبکه را داشته باشند.

### 2-1-3- نسخه‌های بلاکچین

بطور کلی تا به الان چهار نسخه از تکنولوژي بلاکچین شناخته شده است که با توجه به مراجع [4] [14] [15] هریک بطور خلاصه به شرح زیر می‌باشد:

**بلاکچین نسخه ۱:** بلاکچین‌ نسخه اول با پشتیبانی از ارزهای دیجیتال و نرم افزارهای مربوط به آنها شناخته شده است. ایده اصلی در این نسخه، حل یک مسئله ریاضیاتی بود. همچنین دستاورد حاصله از این نسخه غیرمتمرکز بودن و انتقال دارایی بین شرکت‌کنندگان غیر قابل اعتماد است. بارزترین مثال آن بیتکوین است.

**بلاکچین نسخه ۲:** بلاکچین نسخه دوم با قراردادهای هوشمند و برنامه‌های مالی خودکار غیرمتمرکز مبتنی بر ارزهای دیجیتال شناخته شده است. ایده اصلی در این نسخه، اجرای کدها بصورت اتوماتیک و بررسی شرایط موجود در آن و در قالب یک قرارداد بوده است. مهمترین مزیت تکنولوژي بلاکچین برای قرارداد هوشمند، مصون نگه داشتن قراردادها از دستکاری و تغییرات است. از بارزترین مثال‌ها می‌توان به شبکه بلاکچین اتریوم اشاره کرد.

**بلاکچین نسخه ۳:** بلاکچین نسخه سوم با طیف وسیع‌تری از برنامه‌های غیرمتمرکز غیرمرتبط با ارزهای دیجیتال شناخته شده است. ایده اصلی در این نسخه، استفاده از ارتباطات غیرمتمرکز است؛ بطوری که برنامه نویسی سمت سرور[[21]](#footnote-21) برنامه، بصورت غیرمتمرکز و نظیر به نظیر در سطح شبکه اجرا شود. این نسخه با Hyperledger شناخته شده است.

**بلاکچین نسخه 4:** بلاکچین نسخه چهارماستفاده بلاکچین در صنعت را به ارمغان می‌آوردن. این نسخه تمام فعالیت‌های مربوط به سازمان‌های مختلف را با شفافیت بیشتر، غیرقابل تغییر بودن سوابق و اطمینان در معاملات بین سازمان‌ها را تامین میکند. این نسخه بلاکچین با ترکیب هوش مصنوعی شناخته شده است. علاوه بر این، در این نسخه از اینترنت اشیا، ابر داده و محاسبات شناختی جهت برآورده کردن نیازمندی‌های موجود استفاده شده است.

### 2-1-4- چالش‌های بلاکچین

فناوری بلاکچین مانند سایر فناوری‌ها با چالش‌هایی روبرو است که نیاز است به آنها پرداخته شود. چالش‌های موجود در فناوری بلاکچین مقیاس پذیری، محرمانگی[[22]](#footnote-22)، تغییرناپذیری، استفاده مجدد، و یکپارچه‌سازی سیستم‌های قدیمی[[23]](#footnote-23) معرفی کرده است [2] [16].

همانطور که گفته شده است، تراکنش‌ها در استخری ذخیره می‌شوند. از آنجایی که روزانه به تعداد این تراکنش‌ها اضافه می‌شود؛ در نتیجه سایز این استخر افزایش می‌یابد. از طرفی روزانه تعدادی بلوک به زنجیره اضافه می‌شود که خود باعث افزایش سایز بلاکچین می‌شود. هردو مورد ذکر شده مشکل مقیاس‌پذیری بلاکچین را بیان می‌کنند؛ چرا که هر کاربر باید بتواند استخر و زنجیره بلوک خود را هر لحظه بروزرسانی کند و توانایی نگهداری آن را از نظر حجم داشته باشد.

در بستر بلاکچین هرکاربر می‌تواند با یک یا چند آدرس عمومی در شبکه بلاکچین فعالیت داشته باشد بدون آنکه هویت شخص فاش شود؛ چرا که هیچ سازمان متمرکزی عملیات احراز هویت را انجام نمی‌دهد. از این جهت شبکه بلاکچین حریم خصوصی برای کاربران خود مهیا می‌سازد و ویژگی ناشناس بودن را به خود اختصاص می‌دهد؛ اما از آنجایی که جزئیات تراکنش‌ها توسط تمامی شرکت‌کنندگان در شبکه بلاکچین قابل مشاهده است در نتیجه می‌توان گفت بلاکچین این سطح از حریم خصوصی را مهیا نمی‌کند.

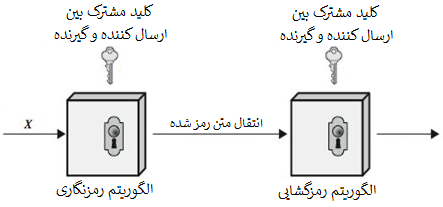
از دیگر چالش‌های بلاکچین‌های عمومی، مانند بیتکوین، می‌توان به مصرف انرژی بسیار بالا و هدر رفت منابع اشاره کرد. فناوری بلاکچین برای تامین امنیت لازم است پردازش محاسباتی سنگینی را جهت حل مسئله داشته باشد که این موضوع باعث مصرف بیش از اندازه از انرژی برق می‌شود. همچنین از آنجایی که واسط‌های برنامه‌ویسی کاربردی خوبی در این فناوری ارائه نشده است در نتیجه می‌توان گفت قابلیت استفاده مجدد در این فناوری ضعیف واقع شده است و یکپارچه شدن سیستم‌های قدیمی با این فناوری از دیگر مشکلات آن است.

### 2-1-5- سیستم‌های رمزنگاری در بلاکچین

سیستم‌های رمزنگاری از مدت‌ها قبل مورد بحث و بررسی جوامع کامپیوتر قرار گرفته است. این سیستم‌ها شامل فرایند تولید کلید، رمزنگاری و رمزگشایی هستند که هر کاربر می‌تواند با کمک کلید خود و الگوریتم رمزنگاری، محتوای خود را رمز کند. یکی از مباحث مهم در فناوری بلاکچین، فرایند امضا کردن می‌باشد. امضای دیجیتال را می‌توان به دیجیتالی بودن امضای کاغذی تشبیه کرد. امضای دیجیتال یکی از عملکردهای رمزنگاری کلید نامتقارن می‌‌باشد. بنابراین می‌توان گفت یکی از کاربردهای رمزنگاری نامتقارن در فناوری بلاکچین، امضای دیجیتال می‌باشد.

* **رمزنگاری کلید متقارن[[24]](#footnote-24)**

رمزنگاری متقارن یا رمزنگاری تک کلیدی، الگوریتم‌ها و پروتکل‌هایی هستند که تنها با استفاده از یک کلید محتوا را رمزنگاری و رمزگشایی می‌کنند. بنابراین واضح است که در رد و بدل کردن اطلاعات، هردو طرف باید این کلید را داشته باشند تا بتوانند فرایند رمزنگاری یا رمزگشایی را انجام دهند. از نمونه الگوریتم‌ها معروف در این زمینه می‌توان به DES، 3DES، AES، RC4 و RC5 اشاره کرد.



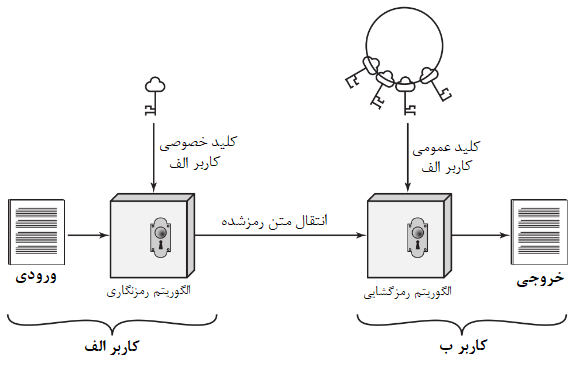
**شکل ‏2‑3: رمزنگاری متقارن**

* **رمزنگاری با کلید نامتقارن[[25]](#footnote-25)**

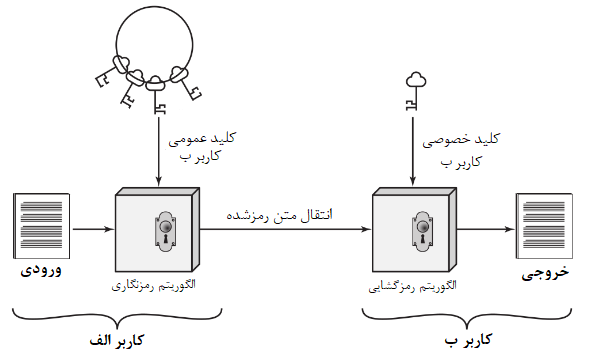
در ابتدا رمزنگاری کلید نامتقارن در راستای حل مشکل رمزنگاری کلید متقارن در قالب الگوریتم دیفی-هلمن[[26]](#footnote-26) پیشنهاد شد. در این نوع رمزنگاری، از یک جفت کلید، کلید عمومی و کلید خصوصی، جهت رمزنگاری و رمزگشایی استفاده می‌شود. کلید خصوصی همیشه در اختیار کاربر است و با سایر کاربران به اشتراک گذاشته نمی‌شود؛ در مقابل، کلید عمومی می‌تواند در اختیار سایرکاربران قرار گیرد.

در بعضی سیستم‌ها، کلید خصوصی جهت رمز کردن و کلید عمومی جهت رمزگشایی استفاده می‌شود و در بعضی از سیستم‌ها این عملیات برعکس خواهد بود. همانطور که در شکل 4 مشاهده می‌کنید زمانی که متن با کلید خصوصی رمز شود، هرکاربری که کلید عمومی را داشته باشد قادر به مشاهده متن می‌باشد؛ که این روش بیشتر زمانی استفاده می‌شود که ارسال کننده فرد دارای کلید خصوصی می‌باشد که آن با فرایند *امضا کردن* شناخته شده است و در فناوری بلاکچین نیز مورد استفاده قرار می‌گیرد. در این روش، یکپارچگی داده برقرار خواهد بود؛ زیرا این اطمینان وجود دارد که تنها کاربری که کلید خصوصی را دارد امکان دستکاری کردن داده را دارد در نتیجه داده به درستی به گیرنده رسیده است.

در شکل 5، اگر متن با کلید عمومی رمز شود، تنها کسی که کلید خصوصی را دارد می‌تواند آن را مشاهده کند؛ که این روش بیشتر زمانی استفاده می‌شود که کاربر می‌خواهد متن خصوصی برای دارنده کلید خصوصی ارسال کند. در این روش، حریم خصوصی کاربر برقرار خواهد بود، زیرا تنها کاربری که کلید خصوصی را داشته باشد قادر به مشاهده داده‌ی اصلی است.



**شکل ‏2‑4: رمز با کلید خصوصی (امضا کردن)**



**شکل ‏2‑5: رمز با کلید عمومی**

دو کلید خصوصی و عمومی از هم متفاوت هستند و با روش‌های ریاضی متفاوتی تولید می‌شود. این دو کلید به گونه‌ای طراحی شده‌اند که امکان بازیابی کلید خصوصی از روی کلید عمومی وجود نخواهد داشت؛ اما برعکس این موضوع امکان پذیر است. از نمونه الگوریتم‌های معروف در این زمینه می‌توان به RSA، ECC، ESDSA، DSA، El-Gamel و Diffie-Hellman اشاره کرد.

* **امضای دیجیتال در بلاکچین**

در فناوری بلاکچین کلید عمومی مشابه شناسه هر فرد عمل می‌کند اما هویت شخص را فاش نمی‌کند؛ زیرا بلاکچین یک سیستم غیرمتمرکز است و افراد در یک سیستم مرکزی احراز هویت نمی‌شوند. بنابراین هویت هیچ فردی مشخص نیست و حتی هر فرد می‌تواند چندین شناسه برای خود ایجاد کند.

در بستر بیتکوین هر فردی که تراکنشی را در بلاکچین ایجاد می‌کند، با کلید خصوصی خود آن را امضا می‌کند و با قرار دادن کلید عمومی خود در آن، آن تراکنش را با شناسه خود با سایر کاربران به اشتراک می‌گذارد تا آنها بتوانند درستی تراکنش را صحت سنجی کنند. همچنین استخراج کننده بلوک، کلید عمومی خود را (که با عنوان آدرس کیف پول شناخته می‌شود) در بلوک قرار می‌دهد تا به کمک آن نشان دهد بلوک مربوطه توسط او استخراج شده، در نتیجه جایزه بلوک تولید شده به او تعلق خواهد گرفت.

در شبکه بلاکچین، هر کاربر یک جفت کلید خصوصی و عمومی دارد. هر کاربر کلید خصوصی خود را باید به نحوه‌های مختلف نگهداری کند تا برای سایر کاربران قابل دسترس نباشد؛ از طرف دیگر لازم است کلید عمومی در سراسر شبکه پخش شوند و در دسترس تمام کاربران قرار گیرد. وجود امضای دیجیتال در روند فناوری بلاکچین کمک به عدم انکار پذیری در سطح شبکه را ایجاد میکند؛ به عبارت دیگر، اگر تراکنشی توسط فردی امضا شده باشد، آن فرد نمی‌تواند انکار کند که صاحب آن تراکنش نیست.

بطور معمول فرایند امضای دیجیتال شامل دو مرحله است: مرحله امضا کردن تراکنش و مرحله تایید. جزئیات هر یک از مراحل به شرح زیر است.

* مرحله امضا کردن: در این مرحله ابتدا تراکنش یا داده مربوطه به کمک تابع هش، مقدار هش آن تولید شده، سپس کاربر با کلید خصوصی خود مقدار هش شده داده را رمزگذاری میکند. با انجام این فرایند عملیات امضا کردن تراکنش با موفقیت انجام خواهد شد. در این مرحله لازم است کاربر داده اصلی خود را به همراه داده امضا شده و کلید عمومی خود را در سطح شبکه منتشر کند.
* مرحله تایید: در این مرحله لازم است دیگر کاربران درستی امضای انجام شده توسط فرستنده را بررسی و تایید کنند. برای انجام این مرحله، کاربران با استفاده از کلید عمومی فرد فرستنده درستی امضا را بررسی می‌کنند. در ابتدا مقدار هش داده اصلی را با کمک تابع هش تولید می‌کنند. همچنین با کمک کلید عمومی فرد فرستنده، داده هش ارسال شده را رمزگشایی می‌کنند. در صورت برابر بودن دو مقدار به دست آمده، امضای فرد فرستنده تایید خواهد شد.

با توجه به ویژگی‌هایی که هر یک از این الگوریتم‌ها دارند، در شبکه بلاکچین از الگوریتم ECDSA[[27]](#footnote-27) جهت امضای دیجیتال استفاده می‌شود. این الگوریتم با کمک کلیدهایی با سایز کمتر، امنیتی مشابه سایر الگوریتم‌ها از جمله RSA مهیا می‌کند.

### 2-1-6- سازوکار‌های اجماع در بلاکچین

شبکه عمومی بلاکچین یک سیستم غیرمتمرکز در سطح جهان است. از آنجایی که در این سیستم هیچ شخص ثالثی جهت تایید تراکنش‌ها فعالیت نمی‌کند، بنابراین باید سازوکاری وجود داشته باشد تا به کمک آن تمام شرکت‌کنندگان موجود در شبکه به کمک آن به یک توافق یکسان برسند. به کمک سازوکار‌های اجماع، تمام شرکت‌کنندگان بر اساس مجموعه‌ای از شرایط به یک تصمیم یکسان می‌رسند؛ بنابراین از الگوریتم‌های اجماع جهت رسیدن به توافق بین شرکت‌کنندگان استفاده می‌شود. به عبارت دیگر هدف از سازوکار اجماع، توافق بین شرکت‌کنندگان در محیطی که هیچ یک به یکدیگر اعتماد ندارند، است. از معروف ترین سازوکار‌های اجماع می‌توان به الگوریم اثبات کار[[28]](#footnote-28) اشاره کرد. تا به امروز الگوریتم‌های مختلف و زیادی ارائه شده است که در ادامه تنها به مواردی که در پایان‌نامه مورد استفاده قرار گرفته است اشاره خواهد شد [17].

* **اثبات کار**

همانطور که گفته شده، الگوریتم اثبات کار یکی از معروف‌ترین الگوریتم‌های اجماع می‌باشد که مورد استقبال قرار گرفته است. این الگوریتم اولین بار در ارز دیجیتال بیتکوین ارائه شده است. از دیگر ارزهای دیجیتالی که از الگوریتم اثبات کار استفاده می‌کنند می‌توان به اتریوم، لایت‌کوین، بیت‌کوین کش اشاره کرد. در این الگوریتم لازم است که هر گره یک مسئله پیچیده ریاضیاتی را حل کند. این مسئله یک مسئله هش با یک مقدار مشخص از میزان سختی است. میزان سختی این الگوریتم رابطه مستقیم با میزان زمانی که لازم است مسئله مورد نظر حل شود، دارد. به عبارت دیگر، هرچه میزان سختی مسئله بیشتر باشد، مدت زمان بیشتری لازم است صرف شود تا مسئله هش حل شود. اثبات کار جهت تایید و بررسی تراکنش‌ها و تولید بلوک مورد استفاده قرار می‌گیرد.

در بستر بیتکوین، از این الگوریتم جهت رسیدن به توافق بر سر صحت بلوک جدید ایجاد استفاده می‌شود. گره‌ها تلاش می‌کنند از اطلاعات موجود در بلاک هش تولید کنند. در صورتی هش مورد نظر صحیح است که از یک الگوی خاصی پیروی کند. زمانی که هش مورد نظر برای بلاک پیدا شود، میتوان گفت بلاک جدید تولید شده است.

* مزایا
  + برقراری امنیت در سطح بالا (در صورت بزرگ بودن قدرت محاسباتی شبکه)
  + کمک به جلوگیری از حمله‌ی دوبار مصرف کردن[[29]](#footnote-29)
* معایب
  + نیازمند دستگاه‌های قدرتمند جهت پردازش و حل مسئله ریاضی
  + مصرف بالای برق
  + احتمال بالای حمله 51% (در صورت کوچک بودن قدر محاسباتی شبکه)
  + زمان‌بر بودن مرحله تایید تراکنش‌ها
* **تحمل خطای بیزانس[[30]](#footnote-30)**

تحمل خطای بیزانس یک الگوریتم اجماع است که در اواخر دهه 90 ایجاد شده است. این الگوریتم با هدف ارائه راه حل‌هایی برای تحمل خطای بیزانس طراحی شده است و از مسئله‌ی ژنرال‌های بیزانس الهام گرفته است. تحمل خطای بیزانس مفهومی است که برای سیستم‌های توزیع شده به وجود آمده است. همانطور که گفته شد، سیستم‌های توزیع شده از جمله بلاکچین، نیاز دارند که شرکت‌کنندگان آن بر سر حالت سیستم به توافق برسند. حال مشکلی که وجود دارد این است که اگر بعضی از این شرکت‌کنندگان در پاسخ به آن، پاسخی ارسال نکنند یا اطلاعات اشتباه ارسال کنند در این صورت سیستم دچار اختلال و مشکل می‌شود. در اینجا بود که راه حل تحمل خطای بیزانس مطرح شد. هدف از این مفهوم ایجاد فضای امن در برابر خرابی سیستم است [18].

از کاربردهای آن می‌توان در سیستم‌های توزیع شده و بلاکچین اشاره کرد. در پروژه Hyperledger Fabric از این الگوریتم اجماع استفاده شده است. ساز و کار این الگوریتم به این صورت است که در هر مرحله از تایید تراکنش لازم است که از بین گره‌های موجود در شبکه یک گره انتخاب شود تا فرایند تایید تراکنش را انجام دهد. گره‌ی انتخابی باید توسط حداقل دو سوم گره‌های موجود در شبکه مورد تایید قرار گیرد. PBFT این تضمین را می‌دهد که با وجود یک سوم گره خراب در شبکه، سیستم دچار خرابی نمی‌شود [6].

* **مزایا**
  + بهره‌وری انرژی
  + کم بودن کارمزد تراکنش‌ها
* **معایب**
  + مناسب برای شبکه‌های کوچک
  + عدم مقیاس پذیری
  + مشکوک به حمله سیبل[[31]](#footnote-31)
* **الگوریتم RAFT**

سازوکار تحمل خطای خرابی، به حل مشکل خرابی گره‌ها پرداخته شده است. به عبارت دیگر، به کمک این سازوکار می‌توان اطمینان حاصل کرد که اگر در یک سیستم غیرمتمرکز گره‌ها دچار توقف[[32]](#footnote-32) یا از شبکه قطع شوند، سیستم می‌تواند حالت خود را نگه‌ دارد. تا به امروز دو الگوریتم مبتنی بر این سازوکار طراحی شده است. الگوریتم اجماع RAFT مبتنی بر تحمل خطای خرابی[[33]](#footnote-33) می‌باشد. در الگوریتم RAFT سه نوع موجودیت نقش دارند که عبارتند از:

* **رهبر[[34]](#footnote-34):** مدیریت تمام درخواست‌های دریافتی از کاربران
* **دنبال‌کننده[[35]](#footnote-35):** خود به تنهایی هیچ درخواستی ارسال نمی‌کنند و تنها موظف به پاسخ دادن به درخواست رهبران و نامزدها هستند (موجودیت منفعل)
* **نامزد[[36]](#footnote-36):** موجودیتی که از بین آنها رهبر جدید انتخاب می‌شود

یکی از فرایندهای مهم در این الگوریتم انتخاب رهبر می‌باشد، چرا که درصورتی که رهبر به هر دلیلی از شبکه خارج شود، نیاز است برای ادامه فرایند رهبر جدید انتخاب شود. نحوه‌ی انتخاب رهبر در این الگوریتم به شرح زیر می‌باشد:

1. زمانی که یک سرور جدید شروع به کار می‌کند، به عنوان دنبال‌کننده شناخته می‌شود و تا زمانی که فراخوانی رویه راه دور[[37]](#footnote-37) صحیحی از رهبر یا نامزد خود دریافت می‌کند، دنبال‌کننده باقی می‌ماند.
2. الگوریتم RAFT از سازوکار ضربان قلب استفاده می‌کند. به عبارت دیگر، تمام دنبال‌کنندگان در حال دریافت پیام پینگ از رهبر هستند. زمانی که رهبر به هر دلیلی از شبکه قطع شود (به عبارتی دیگر پینگ ارسال نکند)، دنبال‌کنندگان بعد انتظار به مدت معلوم، دنبال‌کنندگان مقدار زمانی خود را افزایش داده و خود را به نوع نامزد تغییر می‌دهند.
3. در زمان را‌ی‌گیری، یکی از نامزدها به عنوان رهبر انتخاب می‌شود و زمانی که مقدار زمانی رهبر با یکی از دنبال‌کنندگان یکی شد، رهبر قبلی به حالت دنبال کننده منتقل می‌شود [19] [20].

### 2-1-7- کاربردهای بلاکچین

با آمدن فناوری بلاکچین تحقیقات بسیار زیادی در این زمینه صورت گرفته است. همواره کاربردهای بلاکچین رو به افزایش است. در ابتدا کاربرد بلاکچین تنها در زمینه مالی و ارزهای دیجیتال بود. امروزه علاوه بر اینکه در زمینه‌های مالی بسیار کاربرد دارد، در دیگر زمینه‌های صنعتی نیز مورد توجه و استفاده قرار گرفته است. دامنه کاربردهای بلاکچین شامل بهداشت و درمان، رای گیری، مدیریت احراز هویت، آموزش، حقوق و قانون، رکوردهای دیجیتالی، سوابق دارایی‌ها، امنیت سایبری، بیمه، بازارهای مالی، انرژی و مباحث کامپیوتری است. در ادامه به چند مورد از کاربردهای مهم فناوری بلاکچین در صنایع مختلف پرداخته خواهد شد.

* **برنامه‌های مالی**

یکی از کاربردهای مهم بلاکچین در بحث مالی است. در ابتدا با به وجود آمدن بیتکوین، ارزهای دیجیتال به وجود آمدند. طبق آماری که در [21] اعلام شده است، امروزه بیش از هزاران ارز دیجیتال مبتنی بر فناوری بلاکچین به وجود آمده است .

همچنین از دیدگاهی دیگر، بلاکچین در مباحث خدمات بازرگانی، تسویه حساب‌های مالی، تراکنش‌های اقتصادی و پیش بینی بازار کاربرد دارد. این فناوری می‌تواند نقش اساسی در اقتصاد جهانی داشته باشد و منافع مشتریان و سیستم‌های بانکی را در برگیرد. همچنین از دیگر کاربردهای عملیاتی این فناوری می‌توان به اجرای انواع قراردادها، پرداخت‌های دیجیتالی، مدیریت وام، خدمات عمومی بانکی، حسابرسی مالی و پرداخت و تبدیل ارزهای دیجیتال اشاره کرد. پذیرش بلاکچین توسط بخش‌های مالی باعث صرفه جویی در هزینه در فعالیت‌هایی مانند گزارش مالی، معاملات تجاری و ... شده است.

* **قراردادهای هوشمند**

یکی دیگر از کاربردهای بلاکچین قراردادهای هوشمند است که از آن به عنوان نسخه دوم بلاکچین یاد شده است. قراردادها هوشمند مجموعه‌ای از شروطی‌ است که با توافق طرفین قرار داد نوشته شده است و به صورت خودکار پردازش می‌شود. بعد از اجرا شدن قرارداد، عملیات به اجماع رسیدن اعمال می‌شود و بعد از به اجماع رسیدن هیچ یک از طرفین قرارداد قادر به تغییر شرایط قرارداد یا انکار قرارداد نخواهند بود. قراردادهای هوشمند می‌توانند در بسیاری از کارهای تجاری و حقوقی مورد استفاده قرار گیرند.

یکی از معروف‌ترین نمونه‌های قرارداد هوشمند قرارداد هوشمند اتریوم می‌باشد. همچنین قرارداد هوشمند در رای گیری، مباحث پزشکی، خدمات مالی، خدمات حقوقی، بازی و ... مورد استفاده و استقبال قرار گرفته است. امروزه قراردادهای هوشمند رشد بزرگی داشته و باعث ایجاد فناوری‌های جدید شده است.

* **مدیریت بهداشت و درمان**

فناوری بلاکچین می‌تواند در صنعت بهداشت و درمان با برنامه‌هایی مانند مدیریت بهداشت و درمان عمومی، سوابق مراقبت‌های بهداشتی، دسترسی آنلاین بیمار، به اشتراک گذاری داده‌های پزشکی بیماران، جعل دارو، کارآزمایی بالینی و ارائه دارو مناسب نقش داشته باشد. از آنجایی که داده‌های پزشکی بیماران بسیار مهم است و هرگونه جعل و تغییر این داده‌ها منجر به نتایج جبران ناپذیری خواهد شد، بلاکچین در این زمینه بسیار کاربردی خواهد بود چرا که این فناوری می‌تواند امنیت و قابل دسترس بودن داده‌ها بصورت عمومی را تضمین کند.

بلاکچین در این حوزه می‌تواند مانند یک پروتکل عمل کند تا کاربران از طریق آن به داده‌های پزشکی خود با حفظ امنیت و حریم خصوصی، دسترسی پیدا کنند. از مزایا این فناوری در صنعت بهداشت می‌توان به توزیع سوابق پزشکی بیماران بین سازمان‌های مختلف بهداشتی در سراسر دنیا، عدم امکان دستکاری داده‌ها و بروزرسانی آن اشاره کرد.

* **آموزش**

بلاکچین در بحث آموزشی نیز مورد استفاده قرار گرفته است. این فناوری در جهت حریم خصوصی موسسات آموزشی، ذخیره سوابق آموزشی، ثبت و ذخیره دستاوردهای دانش‌آموزان و دانشجویان، جمع‌آوری و گزارش دهی و تجزیه و تحلیل داده‌های مربوط به مدارس و دانشگاه‌ها، انتشارات علمی نظیر ارسال مقاله و بررسی و تایید آن و مدیریت نگهداری گواهی‌های آموزشی کمک کرده است. در حال حاضر بلاکچین در این زمینه بسیار نوپا است و تا سال 2019 تنها 2% موسسه‌ها عالی از این فناوری استفاده کرده‌اند.

* **کاربردهای مربوط به زمینه‌های کامپیوتری**

بلاکچین کاربردهای بسیاری در زمینه‌های کامپیوتری دارد. در مرجع [22] به کاربرد بلاکچین در اینترنت اشیاء، رایانش ابری، محاسبات لبه‌ای، محاسبات تورین[[38]](#footnote-38)، استفاده در شبکه‌های اجتماعی اشاره شده است.

کاربرد بلاکچین در اینترنت اشیاء بسیار گسترده است. در شبکه سنسورهای بی‌سیم توزیع شده، معماری بلاکچین می‌تواند نقس بسیار موثری در بهبود اینترنت اشیا و به حداکثر رساندن پتانسیل آن داشته باشد و آن را توسعه دهد. ارتباط سیستم‌های مختلف بلاکچین می‌تواند سرویس پرداخت، تجارت سنتی، تجارت الکترونیکی یا سیستم‌های حمل و نقل عمومی و خروجی بصورت امن و غیر وابسته فعال کند.

### 2-1-8- چارچوب‌ها و ابزارهای برنامه‌نویسی در بلاکچین

امروزه جهت تسهیل برنامه‌نویسی در فناوری بلاکچین و قراردادهای هوشمند ابزارها و چارچوب‌های مختلفی عرضه شده است. بعضی از چارچوب‌ها با قرار دادن واسط‌های برنامه‌نویسی کاربردی مناسب، امکان استفاده از زیرساخت خود را مهیا ساخته‌اند. از آنجایی که هر یک از این ابزارها و چارچوب‌ها ویژگی‌ها و کاربردهای متفاوتی دارند، لازم است قبل استفاده از این ابزارها و چارچوب‌ها، با توجه به زمینه پروژه و ویژگی‌های مورد نیاز آن، ابزار وچارچوب مناسب انتخاب شود. در ادامه ابتدا تعدادی از ابزارهای برنامه‌نویسی معروف معرفی شده است؛ سپس تعدادی از چارچوب‌های معروف مورد مقایسه گرفته‌اند.

* **زبان برنامه‌نویسی Solidity**

طبق مستندات مربوط به ابزار Solidity در [23]، این ابزار یک زبان برنامه نویسی شی‌گرا و سطح بالا است که در جهت پیاده سازی قراردادهای هوشمند در بستر اتریوم استفاده می‌شود. با کمک این زبان می‌توان قراردادهایی برای رای‌گیری، تامین سرمایه‌ی جمعی، حراج و کیف پول‌های چند امضایی پیاده‌سازی کرد. Solidity ویژگی‌های ارث بری، کتابخانه‌ها و از انواع داده تعریف شده توسط کاربر در کنار سایر ویژگی‌ها پشتیبانی می‌کند.

این زبان برنامه نویسی متاثر از سه زبان برنامه نویسی پایتون، جاواسکریپت و C++ است و بر اساس ماشین مجازی اتریوم[[39]](#footnote-39) طراحی شده است. همچنین ابزاری به نام solc که مخفف Solidity Compiler است جهت کامپایل کردن کدهای زبانsolidity مورد استفاده قرار می‌گیرد. در واقع با کمک این ابزار اسکریپت‌ها نوشته شده به زبان solidity به یک فرمت خوانا برای ماشین مجازی اتریوم تبدیل خواهد شد. این ابزار با زبان C++ نوشته شده است.

* **Geth**

Geth یک برنامه متن باز است که به کمک آن می‌توان به شبکه اتریوم متصل شد. طبق مستندات مربوط به ابزار Geth یا Go Ethereum در [24] یکی از سه پیاده‌سازی اصلی پروتکل اتریوم است. این پروتکل بر اساس زبان برنامه نویسی Go پیاده سازی شده است و رابط خط فرمان برای اجرای گره‌های اتریوم می‌باشد. سایر پیاده‌سازی‌ها به زبان پایتون و C++ می‌باشد. همچنین می‌توان آن را به عنوان یک برنامه مستقل استفاده کرد یا به عنوان کتابخانه‌ای در پروژه‌های مبتنی بر Go، IOS و اندروید جاسازی کرد.

با استفاده از Geth امکان پیوستن به شبکه اتریوم، انتقال ارز دیجیتال اتریوم بین حساب‌ها، امکان استخراج و پیاده‌سازی قرارداد هوشمند نیز مهیا می‌شود. این ابزار برای پیاده سازی بلاکچین در هر سه سیستم عامل ویندوز، لینوکس و مک قابل استفاده است.

به کمک این ابزار هم می‌توان به شبکه اصلی اتریوم یا سایر شبکه‌های ساخته شده متصل شد، هم می‌توان شبکه خصوصی خود را ایجاد کرد. همچنین امکان انتخاب الگوریتم اجماع از بین اثبات کار و اثبات صلاحیت نیز وجود دارد.

* **Remix**

با مراجعه به مستندات [25]، ابزار remix یک محیط توسعه یکپارچه‌ی [[40]](#footnote-40)متن باز تحت وب و دسکتاپ است. این ابزار یک ابزار قدرتمند متن باز جهت پیاده سازی قراردادهای هوشمند با زبان solidity است که به زبان جاواسکریپت نوشته شده است. در این محیط توسعه یکپارچه ماژول‌هایی به زبان solidity برای آزمودن، اشکال زدایی و استقرار قراردادهای هوشمند وجود دارد.

* **Truffle Suite**

این ابزار یکی از معروف‌ترین چارچوب‌های توسعه اتریوم است که جهت توسعه برنامه‌های مبتنی بر بلاکچین طراحی شده است. این ابزار قدرتمند شامل سه بخش می‌شود که هر یک قابلیت‌هایی به کاربر ارائه می‌دهند. این سه بخش با نام‌های Truffle، Ganache و Dizzle است.

Truffle شامل کتابخانه‌ای وسیع است که جهت پیاده‌سازی قراردادهای هوشمند و پیاده‌سازی برنامه‌های توزیع شده تحت اتریوم مورد استفاده قرار می‌گیرد. این ابزار با استفاده از ماشین مجازی اتریوم یک محیط جهت توسعه و آزمودن بلاکچین مهیا می‌کند.

ابزار Ganache امکان ایجاد شبکه بلاکچین اتریوم خصوصی را برای توسعه‌دهندگان مهیا می‌کند تا به کمک آن برنامه‌های توزیع شده را آزمایش کنند و دستورات را اجرا کنند. این ابزار این کمک را به توسعه دهندگان بلاکچین می‌دهد تا قراردادهای هوشمند خود را بیاموزند.

Dizzle مجموعه‌ای از کتابخانه‌های front-end جهت پیاده‌سازی برنامه‌های توزیع شده را مهیا می‌کند. هسته این ابزار مبتنی بر Redux است.

در جدول ‏2‑1 نیازمندی‌های نرم‌افزار جامع ذخیره‌سازی امن داده‌ با استفاده از فناوری بلاکچین نشان داده شده است. با توجه به نیازمند‌ی‌های ذکر شده، چارچوب‌های مختلف در فناوری بلاکچین مورد مقایسه قرار گرفته است که در جداول جدول ‏***2‑2*** و جدول ***‏2‑3*** نشان داده شده است.

**جدول ‏2‑1: نیازمندی‌های پروژه جهت ذخیره‌سازی امن داده‌ها**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **نوع بلاکچین** | **مقیاس پذیری** | **امنیت** | **حریم خصوصی** | **متن باز** | **کاربرد** | **پشتیبانی از ارز دیجیتال** | **هزینه تراکنش** | **مشوق** |
| **نیازمندی‌های پروژه** | خصوصی - مجوزدار | بالا | بالا | بالا | بله | صنعتی | خیر | کم | خیر |

**جدول ‏2‑2: مقایسه چارچوب‌های معروف در فناوری بلاکچین**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **سکو بلاکچین** | **نوع بلاکچین** | **الگوریتم اجماع** | **زبان برنامه نویسی** | **متن باز** | **نیاز به استخراج کوین** | **پشتیبانی از قرارداد هوشمند** | **کاربرد** |
| **بیتکوین** | عمومی – بدون مجوز | اثبات کار | اسکریپت بیتکوین | بله | بله | خیر | مالی |
| **اتریوم** | عمومی – با مجوز | اثبات کار – اثبات بودجه | سالیدیتی | بله | بله | بله | صنعتی |
| **Zcash** | بدون مجوز | اثبات کار | اسکریپت بیتکوین | بله | بله | خیر | مالی |
| **لایت‌کوین** | بدون مجوز | اثبات کار | اسکریپت بیتکوین | بله | بله | خیر | مالی |
| **Dash** | بدون مجوز | اثبات کار – اثبات خدمت | اسکریپت بیتکوین | بله | بله | خیر | مالی |
| **Peercoin** | بدون مجوز | اثبات کار – اثبات بودجه | اسکریپت بیتکوین | بله | بله | خیر | مالی |
| **ریپل** | بدون مجوز | RPCA | نامشخص | بله | خیر | خیر | مالی |
| **Monero** | بدون مجوز | اثبات کار | نامشخص | بله | بله | خیر | مالی |
| **MultiChain** | مجوزدار | اثبات کار | اسکریپت بیتکوین | بله | خیر | خیر | صنعتی |
| **R3 Corda** | خصوصی | مبتنی بر رای، RAFT | جاوا | بله | بله | بله | مالی |
| **Hyperledger Fabric** | مجوزدار | BTF, CTF | Go، جاوا، NodeJS | بله | خیر | بله | صنعتی |

**جدول ‏2‑3: مقایسه سه چارچوب از منظر فنی در فناوری بلاکچین**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **سکو بلاکچین** | **حریم خصوصی** | **امنیت** | **سرعت** | **مقیاس پذیری** | **هزینه تراکنش** | **مشوق** |
| **بیتکوین** | عدم پشتیبانی از تراکنش‌های خصوصی | عمومی – مستعد حملات | 7 تراکنش در ثانیه | توان عملیاتی کم | بالا | ارز دیجیتال |
| **اتریوم** | پشتیبانی از تراکنش‌های خصوصی | عمومی – مجوزدار | 30 تراکنش در ثانیه | توان عملیاتی کم | بالا | ارز دیجیتال |
| **Hyperledger Fabric** | پشتیبانی از تراکنش‌های خصوصی، کانال شبکه‌ای خصوصی ای ایخنخنتسذدزیمسید | خصوصی– کمتر مورد حمله قرار می‌گیرد | 3 هزار تراکنش در ثانیه | توان عملیاتی بالا | کم | - |

با توجه به نیازمندی‌های ذکر شده و مقایسه‌ی انجام شده بین چارچوب‌های مختلف، چارچوب Hyperledger Fabric بیشتر نیازهای نرم‌افزار را تامین می‌کند. بنابراین، این چارچوب جهت پیاده‌سازی فناوری بلاکچین در نرم‌افزار جامع ذخیره‌سازی امن داده انتخاب و مورد استفاده قرار گرفته است.

## 2-2- Hyperledger Fabric

Hyperledger Fabric یک سکوی متن باز است که در آن فناوری دفترکل بصورت توزیع شده و در شبکه‌ی مجوزدار[[41]](#footnote-41) عمل می‌کند. منظور از شبکه‌ی مجوزدار شبکه‌ای است که کاربر جهت دسترسی به شبکه نیاز به احراز هویت خواهند داشت [26]. این چارچوب به دلیل مقیاس‌پذیر بودن و مقاوم بودن در برابر خطاها نسبت به سایر چارچوب‌های مجوزدار پیاده‌سازی شده محبوب‌تر است. از کاربردهای این چارچوب می‌توان به زنجیره تامین مالی، اینترنت اشیاء، مدیریت داده درمانی و ذخیره‌سازی داده اشاره کرد [19]. Hyperledger Fabric توسط بنیاد لینوکس مقرر شده است که خود تاریخچه بسیار موفقی در بین پروژه‌های متن باز دارد. این چارچوب توسط کمیته فنی متنوع (بیش از 200 نفر) از چندین سازمان (بیش از 35 سازمان) مدیریت می‌شود [12].

همانطور که در بخش‌های قبل گفته شد، اولین و شناخته شده‌ترین کاربرد بلاکچین، ارزدیجیتال بیتکوین است. همچنین از دیگر ارزهای شناخته شده می‌توان به اتریوم اشاره کرد که با اضافه کردن قابلیت قرارداد هوشمند جهت اجرای برنامه‌ها بصورت غیرمتمرکز، راهی متفاوت از یبتکوین را پیش گرفت. اما این دو ارز از نوع بلاکچین‌های عمومی بدون مجوز هستند. به عبارت دیگر، کاربران به صورت ناشناس می‌توانند در شبکه تعامل داشته باشند؛ که این نوع بلاکچین نیازمندی‌های پروژه‌های صنعتی و سازمانی را برآورده نمی‌کند. علاوه بر این، در بعضی از پروژه‌های سازمانی احراز هویت کاربران الزامی است؛ که در این نوع بلاکچین‌ امکان پذیر نمی‌باشد. در ادامه می‌توان به نیازمندی‌های پروژه‌های صنعتی و سازمانی جهت استفاده از بلاکچین اشاره کرد:

* قابلیت شناسایی شرکت‌کنندگان
* نیاز به شبکه مجوزدار
* عملکرد بالا تراکنش
* تاخیر کم در تایید تراکنش
* حریم خصوصی و محرمانه بودن معاملات و داده‎‌های مربوط به شرکت‌

در راستای سازگاری بلاکچین و کاربردهای صنعتی، چارچوب‌ها و سکوهای بسیاری به وجود آمده است. چارچوب Hyperledger Fabric از ابتدا در این راستا قدم برداشته است؛ که در ادامه به ویژگی‌های و تفاوت آن با سایر چارچوب‌ها پرداخته خواهد شد.

شبکه بلاکچین در چارچوب Hyperledger Fabric یک شبکه خصوصی و مجوزدار است. به عبارت دیگر، لازم است کاربر قبل از دسترسی به شبکه و برقراری ارتباط، فرایند شبکه احراز هویت را انجام دهد. همچنین، امکان تعریف نقش‌های مختلف با دسترسی‌های متفاوت برای کاربران وجود دارد. در این شبکه، چهار نقش متفاوت برای بازیگران تعریف شده است که هر یک دسترسی‌های متفاوتی دارند. این نقش‌ها عبارتند از گره[[42]](#footnote-42)، سفارش‌دهنده[[43]](#footnote-43)، مدیر و کاربر نهایی [12].‌

چارچوب Hyperledger Fabric راه حلی برای دفترکل توزیع شده با ویژگی‌هایی نظیر انعطاف‌پذیری[[44]](#footnote-44)، مقیاس پذیری و محرمانگی[[45]](#footnote-45) ارائه کرده است. از آنجایی که این چارچوب دارای معماری مادولار[[46]](#footnote-46) و قابل پیکربندی[[47]](#footnote-47) است، طیف گسترده‌ای از کاربردها را پوشش می‌دهد. این نوع معماری نوآوری، تطبیق‌پذیری و بهینه‌سازی را برای کاربردهایی مانند بانکداری، مالی، بیمه، منابع انسانی، زنجیره تامین، مراقبت‌های بهداشتی و آهنگ‌های دیجیتال مهیا می‌کند [12] [27]. همچنین لازم به ذکر است که Hyperledger Fabric ارزدیجیتال بومی ندارد. در نتیجه لازم نیست جهت اجرا قرارداد هوشمند هزینه‌ای (مانند کارمزد) داشته باشیم.

یکی دیگر از ویژگی‌های این سکو، قابلیت استفاده از سازوکارهای اجماع متفاوت است. به عبارت دیگر می‌توان بر اساس پروژه و کاربرد آن، سازوکار اجماع مناسب و کاراتر انتخاب شود. در این راستا، دو سازوکار اجماع تحمل خطای بیزانس و تحمل خطای خرابی قابل اجرا هستند. سازوکار تحمل خطای خرابی بصورت پیشفرض در این سکو قرار گرفته است؛ اما می‌توان آن را به تحمل خطای بیزانس تغییر داد.

عملکرد این چارچوب به متغیرهای مختلفی از جمله سایز تراکنش، سایز بلاک، سایز شبکه و محدودیت‌های سخت افزاری بستگی دارد. تحقیقات بسیاری بر روی قابلیت عملکردی این پلتفرم انجام شده است. آخرین مقیاس‌پذیری ذکر شده معادل پردازش 20 هزار تراکنش در ثانیه می‌باشد [12]. با توجه به ویژگی‌های ذکر شده، می‌توان گفت این چارچوب نسبت به سایر چارچوب‌ها از عملکرد بهتری در پردازش تراکنش‌ها و میزان تاخیر در تایید تراکنش‌ها دارد.

تمام تراکنش‌ها به کمک قرارداد هوشمند به نام chaincode مدیریت می‌شود. Chaincode برنامه‌ای که توسط برنامه نویس قابل طراحی و پیاده‌سازی می‌باشد. به کمک این برنامه، کاربران نهایی می‌توانند با شبکه بلاکچین ایجاد شده تعامل داشته باشند. این برنامه، زبان‌های برنامه‌نویسی عمومی جاوا، Go و NodeJS را پشتیبانی می‌کند.

امنیت چارچوب Hyperledger Fabric را میتوان از سه دیدگاه مورد بررسی قرار داد که شامل داده، شبکه، خرابی سیستم می‌باشد.

* داده: همانطور که در بخش ‏02-1-3- نسخه‌های بلاکچین اشاره شد، یکی از ویژگی‌های فناوری بلاکچین تغییرناپذیری داده می‌باشد. از آنجایی که در این فناوری از سیستم‌های رمزنگاری استفاده می‌شود و بلاک‌های ایجاد شده به کمک سازوکار هش به یکدیگر متصل می‌شوند، می‌توان این اطمینان را حاصل کرد که داده‌های ثبت شده در بلاکچین امکان تغییر یا حذف نخواهند داشت. با توجه به اینکه چارچوب Hyperledger Fabric از ساختار بلاکچین استفاده می‌کند، در نتیجه امنیت داده‌ها از منظر تغییرناپذیری تضمین خواهد شد.
* شبکه: همانطور که گفته شد، شبکه بلاکچین در Hyperledger Fabric از نوع خصوصی و مجوزدار است. خصوصی بودن شبکه آن کمک می‌کند تا محرمانگی داده‌ها حفظ شود و اطلاعات موجود در پروژه توسط عموم مردم قابل مشاهده نباشد. مجوزدار بودن شبکه آن کمک می‌کند تا هر بازیگر در این شبکه نیاز به احراز هویت داشته باشد و به کمک نقش تعریف شده برای آن حقوق و دسترسی‌ها مشخص شود. این نوع خصوصیات کمک میکند تا کاربر مخرب وارد سیستم نشود. حتی در صورت ورود فرد مخرب به سیستم، امکان شناسایی آن وجود دارد چرا که در این شبکه تمامی کاربران قابل شناسایی هستند. همچنین ارتباطات موجود در این شبکه بر اساس زیر ساخت TLS می‌باشد و تمامی اطلاعات بر بستر رمزنگاری منتقل می‌شود. بنابراین می‌توان گفت که امنیت شبکه این چارچوب بالا بوده و احتمال حمله به آن کم می‌باشد.
* خرابی سیستم: از آنجایی شبکه موجود در این چارچوب به صورت توزیع شده است؛ در نتیجه درصورت خرابی یک گره (یا چندین گره)، همچنان حالت سیستم حفظ خواهد شد. لازم به ذکر است که هرچه تعداد گره‌های موجود در این شبکه بیشتر باشد، سیستم مبتنی بر Hyperledger Fabric در برابر خرابی گره‌ها مقاوم‌تر خواهد بود.

دارای چالش‌هایی در زمینه حریم خصوصی و امنیت است که می‌توان آن را از چند زاویه مورد بررسی قرار داد که در ادامه بطور خلاصه به آنها پرداخته شده است.

* سازوکار اجماع: با توجه به ورژن‌های مختلف این سکو، از ابتدا تا به الان الگوریتم‌های اجماع متفاوتی استفاده شده است که به ترتیب Solo، Apache Kafka، Raft و BFT می‌باشند. در آخرین نسخه دو سازوکار تحمل خطای بیزانس (BFT) و تحمل خطای خرابی (CFT) قرار گرفته است. تصمیم‌گیری انتخاب سازوکار با توجه به حوزه می‌تواند بر امنیت تاثیر گذار باشند. به عنوان مثال، تحمل خطای خرابی تنها می‌تواند نگهداری حالت سیستم را درصورت خرابی گره‌ها تضمین کند و نمیتواند در تشخیص گره خرابکار عملکردی داشته باشد.
* قرارداد هوشمند: قراردادهای هوشمند به علت خطا در طراحی و کدنویسی می‌تواند آسیب‌پذیر باشند. همانطور که گفته شد، کد زنجیره‌ای (قرارداد هوشمند در Hyperledger Fabric) دارای زبان برنامه نویسی خاص منظوره نیست؛ در نتیجه ویژگی‌ها و محدودیت‌های مربوط به بلاکچین در این قرارداد کنترل نمی‌شود. این ویژگی خود می‌تواند باعث افزایش آسیب‌پذیری شبکه فابریک را به همراه داشته باشد [28] [29].

# راهکار پیشنهادی

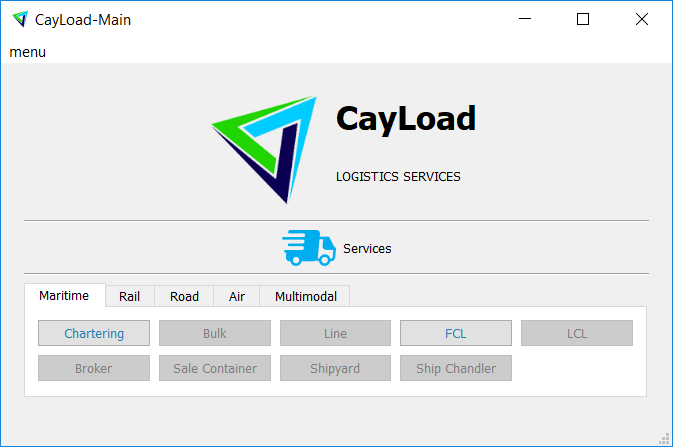
هدف روش پیشنهادی، ارائه نرم‌افزار جامع ذخیره‌سازی امن داده‌های سایر سیستم‌های نرم‌افزاری با استفاده از فناوری بلاکچین است. بدین منظور سعی شده است برای به دست آوردن روش مناسب جهت ذخیره‌سازی امن داده، دو رویکرد مختلف پیاده‌سازی شود. در ابتدا، ذخیره‌سازی داده‌های یک سیستم حمل و نقل جهانی تحت وب در شبکه بلاکچین شبیه‌سازی شده است و به عنوان خروجی اول در نظر گرفته شده است. با توجه به بررسی‌های انجام شده بر روی نمونه اولیه خروجی اول، مشکلاتی نظیر امکان تغییر داده، عدم پشتیبانی درست از توزیع‌پذیری و مشکلات امنیتی نمایان شد. به همین دلیل، با استفاده از چارچوب Hyperledger Fabric نمونه اولیه دیگری ارائه شد و به عنوان خروجی دوم در نظر گرفته شده است. با حل مشکلات ذکر شده در خروجی دوم، این نمونه اولیه در طراحی نرم‌افزار جامع ذخیره‌سازی امن داده‌های سایر سیستم‌های نرم‌افزاری استفاده شده است.

پیاده‌سازی رویکرد‌های پیشنهادی از دو بخش کاربر و سرور تشکیل شده که در ادامه، به جزئیات پیاده‌سازی هر یک از روش‌ها پرداخته شده است.

## 3-1- خروجی اول: شبیه‌سازی بلاکچین بر روی زیرخدمت‌های FCL و Chartering

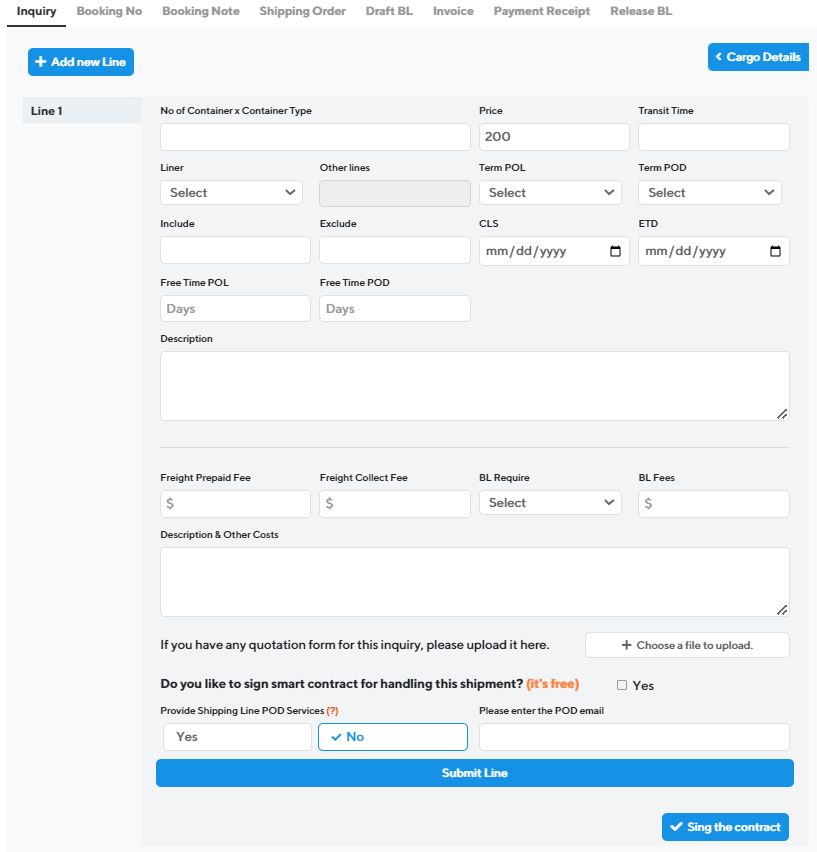
### 3-1-1- پیاده‌سازی سمت کاربر

از آنجایی که پیاده‌سازی این دو زیرخدمت مشابه یکدیگر است، تنها به جزئیات نحوه‌ی پیاده‌سازی زیرخدمت FCL پرداخته خواهد شد. در این برنامه، پیاده‌سازی بلاکچین با زبان پایتون زده شده است. همچنین برای بخش گرافیک نرم‌افزار از کتابخانه QTPython استفاده شده است. جهت پیاده‌سازی بلاکچین از کتابخانه‌های مختلفی استفاده شده که در ادامه به آن پرداخته می‌شود.



**شکل ‏3‑1: نمایی از نرم‌افزار بلاکچین**

همانطور که گفته شد، ابتدا باید طرفین قرارداد، قرارداد مورد نظر خود را تایید نهایی کنند. برای انجام این کار، هر کاربر می‌تواند قراردادها‌ی خود را در وب سایت مربوط به پروژه‌ی حمل و نقل مشاهده کند و در صورت تمایل آن را تایید نهایی کند. به عنوان مثال، در تصویر زیر نمونه قراردادی از زیرخدمت FCL آورده شده است که در انتهای آن کاربر می‌تواند با زدن دکمه sign the contract قرارداد مربوطه را تایید نهایی کند. در نتیجه زمانی که تمامی افراد داخل قرارداد، مفاد قرارداد را تایید نهایی کنند، قرارداد مربوطه در برنامه تحت دسکتاپ نمایش داده می‌شود.



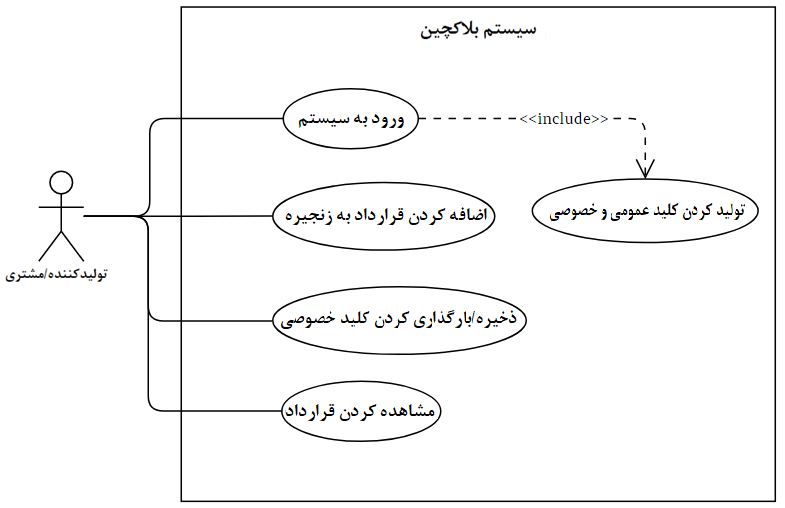
**شکل ‏3‑2: نمونه‌ای از مفاد قرارداد FCL در وبسایت**

با توجه به وجود احتمال افزایش کاربران، مشکلاتی در ذخیره‌سازی و انتقال بلاکچین بین کاربران و سرور به وجود می‌آید. دلیل بروز این مشکل این است که افزایش کاربران باعث افزایش حجم قراردادهای عقد شده بین کاربران می‌شود؛ در نتیجه احتمال افزایش حجم زنجیره بلاکچین وجود دارد. در جهت حل این مسئله تصمیم بر این گرفته شد که برای هر قرارداد یک شبکه بلاکچین جداگانه تشکیل شود و در نهایت شبکه مربوطه برای هر قرارداد در سرور ذخیره می‌شود. در نتیجه هر شبکه بلاکچین به تعداد طرفین قرارداد بعلاوه یک بلوک جنسیس، بلوک خواهد داشت. به عنوان مثال، برای قراردادی که طرفین قرارداد آن دو کاربر هستند، تعداد بلوک‌های شبکه بلاکچین مربوط به آن قرارداد سه بلوک خواهد بود. در نتیجه حجم کمی نیاز خواهد بود تا در سرور ذخیره شود. این مسئله میتواند امنیت شبکه بلاکچین را کاهش دهد؛ زیرا در این صورت یک کاربر مخرب می‌تواند با تغییر دادن کل بلوک‌ها (که تعداد آن بسیار کم است) متناسب با خواسته خود، داده‌های ذخیره شده را تغییر دهد.

در ادامه به کمک نمودار مورد کاربرد و نمودار کلاس به جزئیات هر بخش پرداخته شده است. همانطور که گفته شده، از آن‌جایی که پیاده سازی دو زیرخدمت ذکر شده مشابه یکدیگر است و تنها در واسط‌های برنامه‌نویسی کاربردی با هم متفاوت هستند، به توضیحات تنها یکی از این زیرخدمت‌ها، FCL، پرداخته شده است.

* **نمودار مورد کاربرد زیرخدمت FCL**

در پروژه حمل و نقل، بخش بلاکچین یک مورد کاربرد به حساب می‌آید؛ اما برای فهم بهتر جزئیات این بخش، نمودار مورد کاربر آن بصورت جزئی‌تر رسم شده که به شکل زیر می‌باشد. در این نمودار بخش‌های مهم سیستم در قالب نمودار مورد کاربرد بیان شده است.



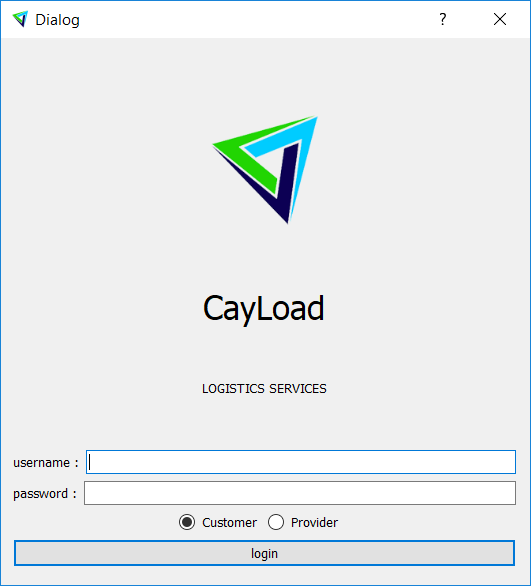
**شکل ‏3‑3:** **نمودار مورد کاربرد ریزخدمت‌ FCL**

1. **مورد کاربرد ورود به سیستم**

در این مورد کاربرد، برای ورود کاربر به نرم افزار از همان مکانیزم و واسط‌های برنامه‌نویسی کاربردیهای پیاده سازی شده مربوط به سایت استفاده شده است. در فرایند ورود به سیستم، عملیات تولید کلید نیز صورت می‌گیرد. در طی این فرایند سه حالت برای تولید کلید‌ها به وجود می‌آید که به شرح زیر می‌باشد:

**حالت اول:** کاربر برای اولین بار نرم افزار را نصب کرده باشد. در اینصورت فیلد کلید عمومی کاربر در سمت سرور خالی است و تنها نام کاربری و رمز عبور فرد وجود دارد. در اینصورت بعد از ورود موفقیت آمیز کاربر، سیستم کلید عمومی و کلید خصوصی تولید می‌کند و آنها را در سیستم کاربر ذخیره می‌کند؛ همچنین کلید عمومی او به سرور ارسال می‌شود و در آنجا نیز ذخیره می‌شود.

**حالت دوم:** کاربر نرم افزار را از قبل نصب کرده و مجدد از نرم افزار استفاده می‌کند. در اینصورت کاربر بعد از ورود موفقیت آمیز، از آنجایی که کلید عمومی و خصوصی از قبل تولید شده و در سیستم ذخیره شده است، کلید عمومی و خصوصی جدیدی تولید نمی‌شود و از کلیدهای قبلی استفاده می‎‌شود.

**حالت سوم:** کاربر مجبور به نصب مجدد نرم افزار شده است. در اینصورت کاربر بعد از ورود موفقیت آمیز، از آنجایی که کلید عمومی تولید شده او در سرور وجود دارد ولی در سیستم وجود ندارد، مشخص می‌شود کلیدهای عمومی و خصوصی برای او تولید شده است. در نتیجه لازم است کاربر کلید خصوصی خود را بارگذاری کند تا به کمک آن کلید عمومی بازیابی شود و آن را با کلید عمومی موجود در سرور مقایسه شود تا صحت آن تایید شود.

**شکل ‏3‑4: صفحه ورود به نرم افزار**

1. **مورد کاربرد تولید کلید عمومی و خصوصی**

در این مورد کاربرد به تولید کلید جهت رمزنگاری نامتقارن پرداخته می‌شود. رمزنگاری نامتقارن یک سیستم رمزنگاری است که از دو کلید جهت رمزگذاری و رمزگشایی استفاده می‌کند. جهت تولید کلید عمومی و خصوصی و قابلیت امضا کردن از الگوریتم رمزنگاری RSA موجود در کتابخانه‌ی آماده‌ی cryptography استفاده شده است. برای تولید کلید خصوصی از تابع rsa.generate\_private\_key() استفاده شده است. یکی از مهمترین پارامترهای این تابع سایز کلید است که معادل 2048 بیت قرار داده شده است. سایز کلید میزان امنیت کلید را مشخص می‌کند که هرچه این سایز بیشتر باشد از امنیت بالاتری برخوردار است. در حال حاضر از آنجایی که روز به روز سیستم‌های قوی‌تری به بازار عرضه می‌شوند در نتیجه حداقل سایز مورد نیاز جهت جعل نشدن کلید در الگوریتم RSA معادل 2048 بیت اعلام شده است. با کلید خصوصی تولید شده، به کمک تابع public\_key() کلید عمومی متناظر تولید می‌شود.

بعد از ساخت کلیدها، کلیدهای تولید شده در سیستم کاربر ذخیره می‌شود و کلید عمومی به سرور ارسال می‌شود و در سرور نیز ذخیره می‌شود. از آنجایی که کلیدهای گفته شده از اهمیت زیادی برخوردار هستند، قبل از ذخیره سازی کلید‌ها، آنها به کمک کلید متقارن رمزنگاری می‌شوند. برای رمزنگاری متقارن از کتابخانه PBKDF2HMAC و Fernet استفاده شده است.

1. **مورد کاربرد اضافه کردن قرارداد به زنجیره**

یکی از بخش‌های مهم بلاکچین در این پروژه، اضافه کردن قرارداد به زنجیره بلوک است. این مورد کاربرد از بخش‌های مختلفی جهت تکمیل فرایند خود استفاده می‌کند که هر یک به شرح زیر است.

* **ایجاد کردن بلوک**

هر بلوک شامل اطلاعات مختلفی است. این اطلاعات به شرح زیر می‌باشد:

* index: شماره بلوک در زنجیره بلوک
* timestamp: زمان ایجاد بلوک
* data: اطلاعات مربوط به قرارداد
* signature: شامل نام، امضا و کلید عمومی کاربر
* previuse\_hash: هش اطلاعات بلوک قبلی
* hash: هش اطلاعات بلوک فعلی
* proof\_of\_work: عدد نانس

برای ایجاد بلوک جدید ابتدا لازم است تمام اطلاعات مربوط به قرارداد از سرور دریافت و به رشته تبدیل شود؛ همچنین لازم است زمان ایجاد بلوک به این رشته اضافه شود. در نتیجه زمان ایجاد بلوک در قالب رشته به اطلاعات قرارداد اضافه می‌شود.

در فناوری بلاکچین، الگوریتم‌های مختلفی برای رسیدن به اجماع استفاده می‌شود. الگوریتم مورد استفاده در این پروژه، الگوریتم اثبات کار[[48]](#footnote-48) است. این الگوریتم برای رسیدن به اجماع جهت جلوگیری از حملات مربوط به شبکه رایانه‌ای استفاده می‌شود. در این مکانیزم، فرستنده با صرف هزینه پردازش محاسبات ریاضیاتی به یک مقدار عددی می‌رسد و گیرنده تنها با کمک آن عدد صحت کار را اثبات می‎کند.

مکانیزم مربوطه با صرف پردازش ریاضیاتی مسئله‌ای را حل میکند. این مسئله با کمک تابع هش SHA256 الگوی مشخص شده را تولید می‌کند. در پروژه الگو داده شده 00 در ابتدا مقدار هش تولید شده است. تا زمانی که هش مورد نظر با این الگو تولید نشده باشد، عملیات تولید هش ادامه خواهد داشت. مقدار ورودی تابع SHA256 رشته‌ داده‌ی قرارداد به همراه زمان و عدد نانس[[49]](#footnote-49) می‌باشد. در هر بار تولید مجدد هش، مقدار عددی نانس اضافه می‌شود. زمانی که هش با الگوی مورد نظر تولید شد آنگاه مقدار عددی نانس در اطلاعات بلوک با عنوان proof\_of\_work ذخیره می‌شود. در ابتدا مقدار عددی نانس برابر صفر است.

* **امضا کردن قرارداد**

در ادامه بعد از کامل شدن اطلاعات بلوک و بدست آمدن هش مورد نظر، عملیات امضای کاربر صورت می‌گیرد. کاربر با کلید خصوصی خود هش بدست آمده را امضا می‌کند تا عملیات انجام شده به نام او ثبت شود. با انجام عملیات امضا، امکان انکار کردن از کاربر گرفته می‌شود؛ چرا که تنها کسی که کلید خصوصی او را دارد خودش است؛ بنابراین امضا شدن با کلید خصوصی کاربر به منزله‌ی امضا کردن توسط خود کاربر می‌باشد.

* **جایگزین کردن زنجیره بلوک جدید در سرور**

در ابتدای ایجاد بلوک جدید، آخرین نسخه زنجیره بلوک از سرور دریافت می‌شود. بعد از ایجاد بلوک جدید و امضا شدن آن با کلید خصوصی کاربر، بلوک جدید به زنجیره بلوک دریافت شده اضافه می‌شود.

در ادامه لازم است زنجیره بلوک ایجاد شده جایگزین زنجیره بلوک موجود در سرور شود تا زنجیره بروز شده برای دیگر طرفین قرارداد قابل دسترس باشد. به همین دلیل لازم است قبل ارسال زنجیره بلوک جدید به سرور، شروطی مورد بررسی قرار گیرد؛ چرا که ممکن است در حین ایجاد بلوک جدید، کاربر(ان) دیگر زنجیره بلوک را بروزرسانی کرده باشد و کاربر مذکور از آن مطلع نباشد.

به این منظور در دو مرحله زنجیره بلوک از سرور دریافت می‌شود:

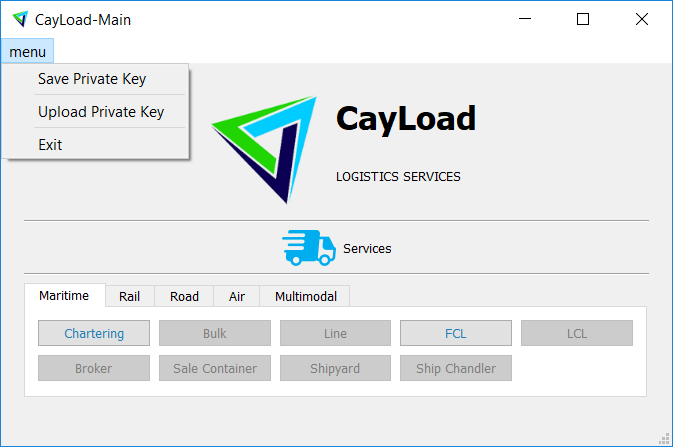
**مرحله اول:** دریافت آخرین نسخه زنجیره بلوک قبل از ایجاد بلوک جدید (زنجیره بلوک قدیمی)

**مرحله دوم:** دریافت آخرین نسخه زنجیره بلوک بعد از ایجاد بلوک جدید (زنجیره بلوک جدید)

در این مرحله دو شرط مورد بررسی قرار می‌گیرد. شرط اول بررسی طول زنجیره‌های قدیمی و جدید و شرط دوم بررسی زمان ایجاد آخرین بلوک در زنجیره‌ی قدیمی و جدید است. اگر طول زنجیره قدیمی از طول زنجیره جدید کمتر باشد یا زمان ایجاد آخرین بلوک ایجاد شده در زنجیره بلوک قدیمی جلوتر از زمان ایجاد آخرین بلوک ایجاد شده در زنجیره بلوک جدید باشد، در این صورت زنجیره بلوک طی ایجاد بلوک جدید بروزرسانی شده است. در نتیجه لازم است مجدد بلوک جدید ایجاد شود؛ چرا که بلوک جدید از نظر زمانی باید از آخرین بلوک موجود در زنجیره جلوتر باشد. اگر شرایط فوق برقرار نباشد بدین معناست که زنجیره‌ی ایجاد شده توسط کاربر جدیدترین زنجیره بلوک است و باید جایگزین زنجیره موجود در سرور شود.

1. **مورد کاربرد ذخیره/بارگذاری کلید خصوصی:**

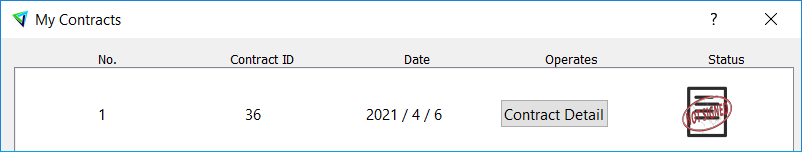
از آنجایی که کلید خصوصی از اهمیت بالایی برخوردار است، امکان ذخیره سازی کلید خصوصی در مسیر دلخواه کاربر پیاده‌سازی شده است. باید در نظر داشته باشیم که بازیابی کلید خصوصی به هیچ روشی امکان پذیر نیست و در صورت پاک شدن، امکان دسترسی به قراردادها از کاربر گرفته می‌شود. در نتیجه در این مورد کاربرد، کاربر می‌تواند کلید خود را ذخیره کند. همچنین همانطور که در بخش ورود به سیستم گفته شد، لازم است در مواقعی کاربر کلید خصوصی خود را بارگذاری کند تا کلید عمومی آن تولید شود. در این مورد کاربرد این امکان به کاربر داده می‌شود.

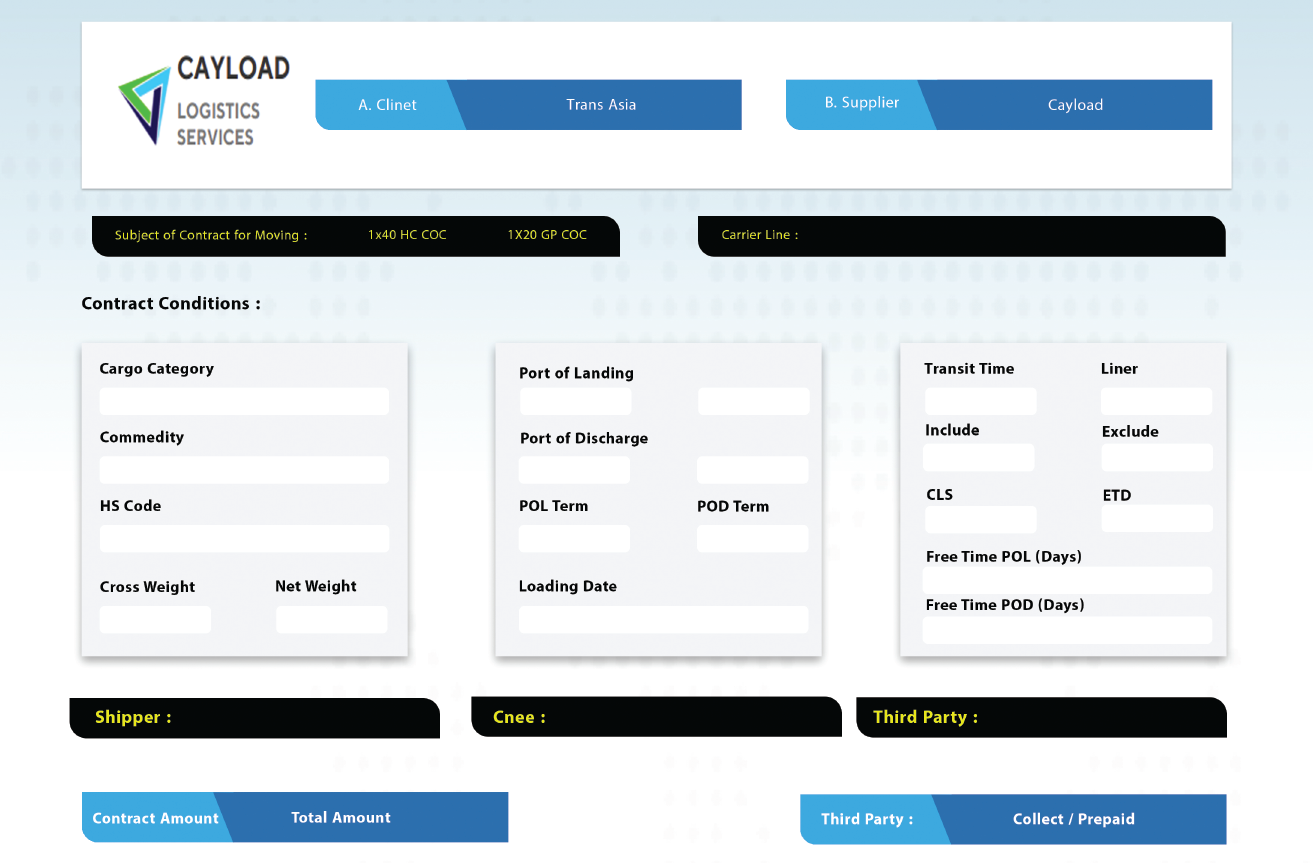
**شکل ‏3‑5: بارگذاری/ذخیره‌سازی کلید خصوصی**

1. **مورد کاربرد مشاهده قراردادها:**

مشاهده قراردهای نهایی شده توسط طرفین قرارداد برای هر یک از کاربران در نرم افزار مهیا شده است. در صفحه اصلی این نرم افزار تمام خدمات و زیرخدمات قابل مشاهده هستند. همانطور که در شکل ‏3‑1 مشاهده کردید، در حال حاضر تنها بخش‌هایی که دارای بلاکچین هستند فعال شده است. بنابراین لیست قراردادهای نهایی شده در هر خدمت و زیرخدمت در بخش خود قرار گرفته است. به عنوان مثال، در شکل ‏3‑6 قرارداد نهایی شده در بخش FCL برای کاربر قابل مشاهده است. همچنین کاربران می‌توانند قراردادهای خود را در هر زیرخدمت مشاهده کنند. همچنین این قابلیت به کاربر داده شده تا جزئیات قرارداد خود را در قالب یک فایل PDF مشاهده کند.



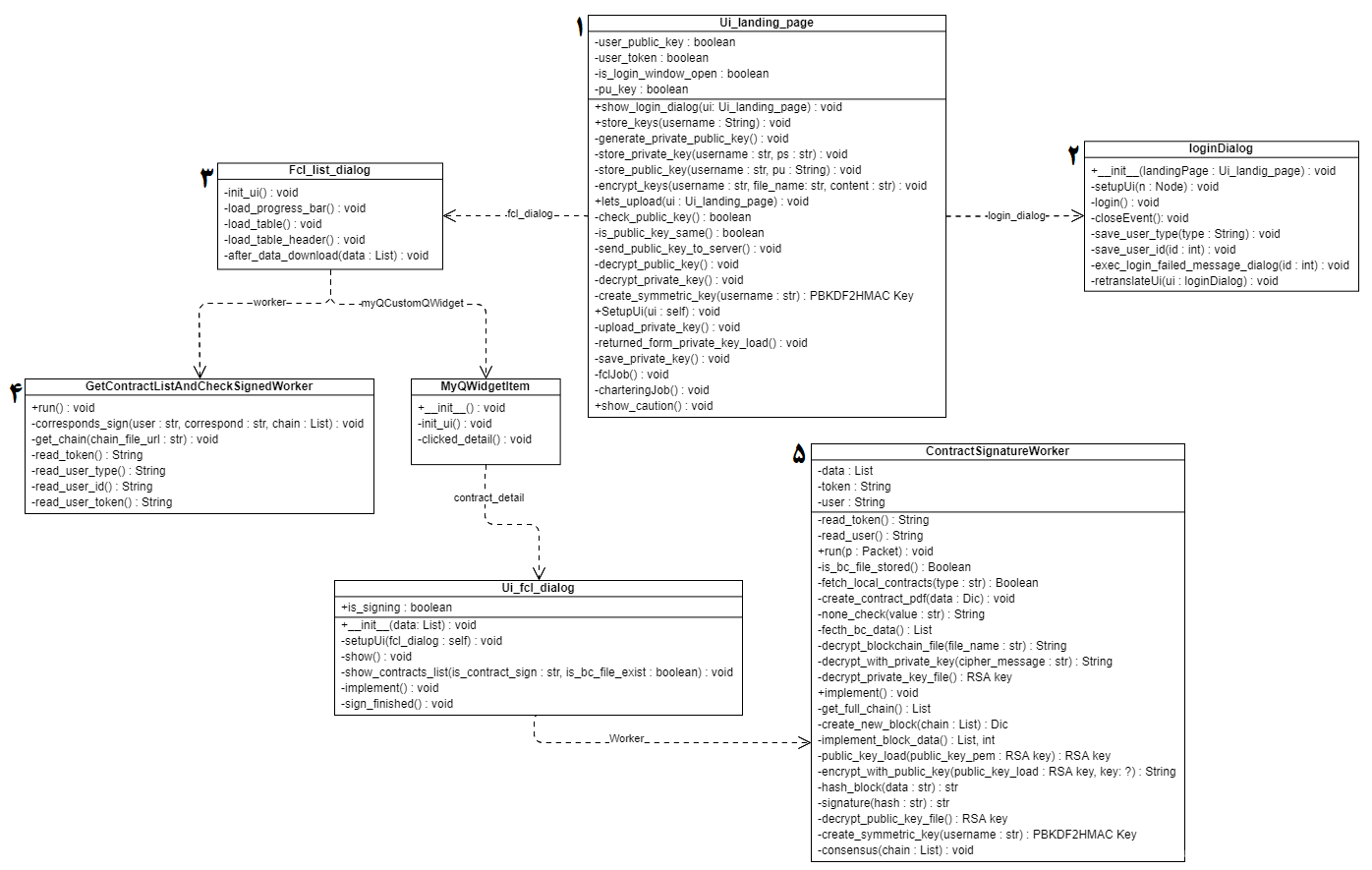
**شکل ‏3‑6: صفحه نمایش لیست قراردادها**



**شکل ‏3‑7:** **نمایی از اطلاعات قرارداد در قالب PDF**

* **نمودار کلاس زیرخدمت FCL**

جهت پیاده سازی این نرم افزار، نمودار کلاس آن رسم شده است که در تصویر زیر قابل مشاهده می‌باشد. کلاس شروع کننده کلاس Ui\_landing\_page می‌باشد که در فایل main.py پیاده سازی شده است. در نمودار زیر، تنها کلاس‌های مربوط به خدمت FCL رسم شده است. سایر خدمات کلاس‌های مشابه و مشترکی دارند و تنها باید واسط برنامه‌نویسی کاربردی[[50]](#footnote-50) هر یک مطابق خدمت خود جایگزین شود.



**شکل ‏3‑8: نمودار کلاس زیرخدمت FCL**

**جدول ‏3‑1: نگاشت توابع به سند**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| نام مورد کاربرد | قابلیت | نام کلاس | | نام متد |
| ورود به سیستم | نمایش اولیه صفحه ورود | Ui\_landing\_page | 1 | show\_login\_dialog() |
| فرایند احراز هویت | Ui\_loging\_dialog | 2 | login() |
| ذخیره اطلاعات در سیستم میزبان | Ui\_loging\_dialog | 2 | save\_user\_type()  save\_user\_id() |
| بررسی وضعیت کلیدها | Ui\_landing\_page | 1 | lets\_upload()  check\_public\_key()  is\_public\_key\_same() |
| تولید کردن کلید عمومی/خصوصی | تولید کلیدها | Ui\_landing\_page | 1 | generate\_private\_public\_key()  store\_private\_key()  store\_public\_key() |
| رمزنگاری کلیدها | Ui\_landing\_page | 1 | encrypt\_keys()  create\_symmetric\_key() |
| اضافه کردن قرارداد به زنجیره | ایجاد کردن بلوک | contractSignatureWorker | 5 | create\_new\_block() |
| امضا کردن قرارداد | contractSignatureWorker | 5 | signature() |
| جایگزین کردن زنجیره بلوک جدید در سرور | contractSignatureWorker | 5 | consensus() |
| ذخیره/بارگذاری کلید خصوصی | ذخیره کردن کلید | Ui\_landing\_page | 1 | save\_private\_key() |
| بارگذاری کردن کلید | Ui\_landing\_page | 1 | upload\_private\_key() |
| مشاهده قرارداد | ایجاد جدول قراردادها | Fcl\_list\_dialog | 3 | init\_ui()  load\_progress\_bar()  load\_table()  load\_table\_header() |
| نمایش لیست قراردادها | GetContractListAndCheckSignedWorker | 4 | run()  corresponds\_sign() |
| نمایش به صورت PDF | contractSignatureWorker | 5 | run()  create\_contract\_pdf() |

### 3-1-2- پیاده‌سازی سمت سرور

در این دو زیرخدمت پردازش‌های سمت سرور شامل ایجاد فایل اولیه بلاکچین، تایید بلوک جدید و تایید زنجیره بلوک است. همانطور که گفته شد، یکی از قابلیت‌های موجود این نرم افزار مشاهده قراردادها توسط کاربر می‌باشد. شرط نمایش قرارداد در نرم افزار ایجاد فایل اولیه بلاکچین است. به عبارت دیگر، زمانی که برای قرارداد فایل اولیه بلاکچین تولید شد، آنگاه توسط کابر قابل مشاهده خواهد شد؛ در غیر اینصورت کاربر قادر به مشاهده و امضای قراردادی که در سایت ایجاد کرده است در نرم افزار نخواهد بود. به این منظور در مرحله اول، زمانی که قراردادی توسط طرفین قرارداد به تایید نهایی می‌رسد، در سمت سرور فایل اولیه بلاکچین آن قرارداد که شامل بلوک اولیه که به آن بلوک جنسیس گفته می‌شود را تولید خواهد کرد و در مسیر سرویس جاری ذخیره خواهد شد. تایید نهایی کاربر به منزله‌ی عدم تغییر مفاد قراداد خواهد بود؛ به عبارت دیگر، زمانی که طرفین قرارداد، قرارداد مذکور را تایید نهایی می‌کنند، قادر به تغییر محتوای آن نخواهد بود. چرا که بعد از این تایید کاربر قادر به مشاهده قرارداد در نرم افزار و امضای آن خواهد بود.

در ادامه بعد از اینکه کاربر قراردادی را امضا کردن، که امضا کردن قرارداد به معنای ایجاد بلوک جدید و اضافه کردن قرارداد جدید به زنجیره است، لازم است زنجیره بلوک جدید ایجاد شده درستی و صحت آن مورد بررسی قرار گیرد. در مرحله اول باید بلوک جدید اضافه شده مورد بررسی قرار گیرد. در بررسی بلوک جدید دو شرط در آن مورد بررسی قرار میگیرد:

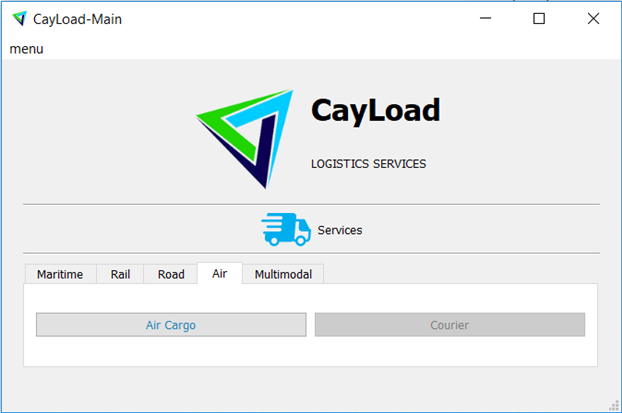
1. بررسی صحت هش تولید شده بلوک. سرور مجدد با توجه به اطلاعات بلوک، هش آن را محاسبه می‌کند و مقدار آن را با هش تولید شده توسط کاربر مقایسه می‌کند.
2. بررسی امضای کاربر در بلوک. سرور مجدد با توجه به اطلاعات بلوک، امضای کاربر را مورد صحت سنجی قرار می‌دهد. برای صحت سنجی امضای کاربر از کلید عمومی او استفاده خواهد شد. با کمک کلید عمومی کاربر، اطلاعات امضا شده صحت سنجی می‌شود.

در صورتی که هر دو شرط بالا برقرار باشد آنگاه درستی بلوک تولید شده تایید می‌شود. در مرحله دوم باید زنجیره بلوک صحت سنجی شود. به عبارت دیگر؛ باید ارتباط بین بلوک‌های تولید شده در بلاکچین مورد بررسی قرار گیرد. به این منظور، هش بلوک قبلی که در بلوک فعلی ذخیره شده است با هش خود بلوک قبلی مورد مقایسه قرار می‌گیرد. در صورت یکسان نبودن این دو مقدار می‌توان گفت بلوک قبلی دستکاری شده و در نتیجه زنجیره بلوک موجود درست نمی‌باشد. در غیر اینصورت، صحت زنجیره بلوک تایید خواهد شد.

برای اجرای فرایندهای گفته تنها سه متد در سمت سرور در شرایط لازم اجرا خواهند شد. که عبارتند create\_genesis\_block()، verify() و verify\_sign() می‌باشد.

## 3-2- خروجی دوم: استفاده از چارچوب Hyperledger Fabric بر روی زیرخدمت Air

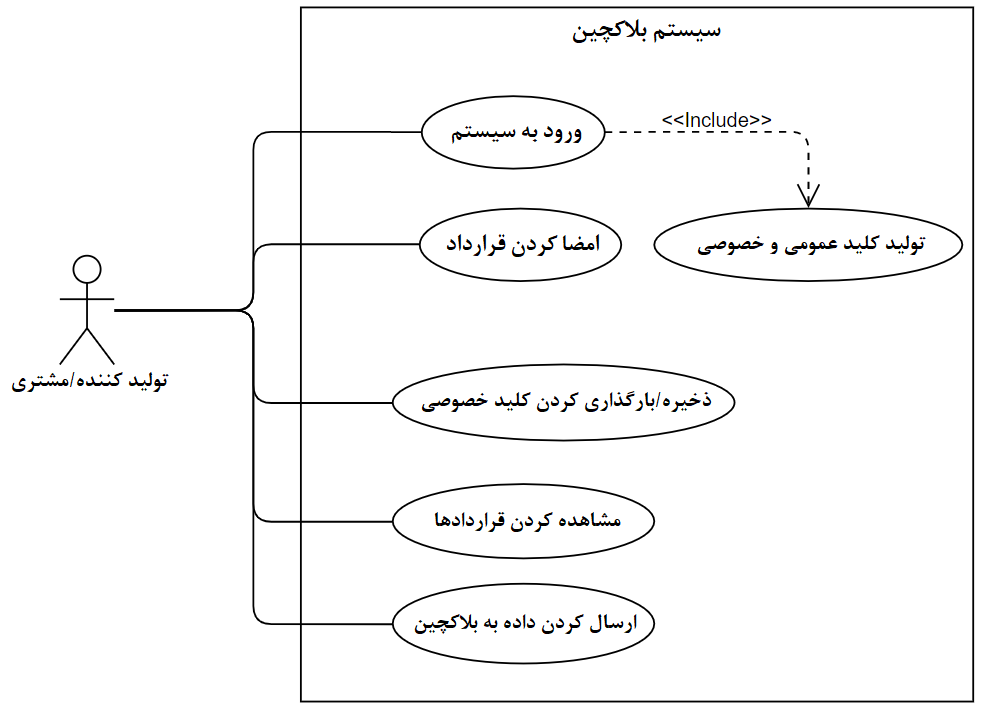
### 3-2-1- پیاده‌سازی سمت کاربر

در زیرخدمت Air، برای استفاده از بلاکچین از چارچوب آماده‌ی Fabric Hyperledger استفاده شده است که جزئیات عملکرد آن در بخش پیشینه آورده شده است. در این زیرخدمت، چارچوب ذکر شده جهت ذخیره سازی امن داده‌ها استفاده می‌شود. با توجه به اینکه این چارچوب مشکل مقیاس‌پذیری کمتری نسبت به روش زده شده در دو زیرخدمت دیگر دارد، در نتیجه تمامی قراردادها در یک زنجیره ذخیره می‌شود. در این زیرخدمت، مانند دو زیرخدمت دیگر، از یک نرم افزار یکسان استفاده شده است با این تفاوت که در پیاده‌سازی سرور با یکدیگر متفاوت هستند.

**شکل ‏3‑9: نمایی از نرم افزار – زیرخدمت Air**

در ادامه به کمک نمودار مورد کاربرد و نمودار کلاس به جزئیات هر بخش پرداخته شده است.

* **نمودار مورد کاربرد زیرخدمت Air**



**شکل ‏3‑10: نمودار مورد کاربر زیرخدمت Air**

در نمودار مورد کاربر نشان داده شده‌، تعدادی از مورد کاربردها با ماژول FCL و Chartering یکسان است که شامل موارد کاربرد ورود به سیستم، امضا کردن قرارداد، ذخیره/بارگذاری کردن کلید خصوصی، مشاهده قراردادها و تولید کلید عمومی و خصوصی است. بنابراین از توضیح مجدد آن در این سند خودداری شده است. بنابراین تنها در مورد کاربرد "ارسال کردن داده به بلاکچین" با یکدیگر متفاوت هستند که در ادامه با جزئیات بیشتری به آن پرداخته شده است.

از آنجایی که تنها تعدادی از داده‌ها مهم ضرورت ذخیره سازی در بلاکچین را دارند، در نتیجه باید داده‎‌های مورد نظر را از بین داده‌های دریافتی از سایت پیش پردازش شود. در نتیجه داده‌ها قبل از ارسال در داخل یک دیکشنری و در فرمت JSON نگهداری می‌شود.

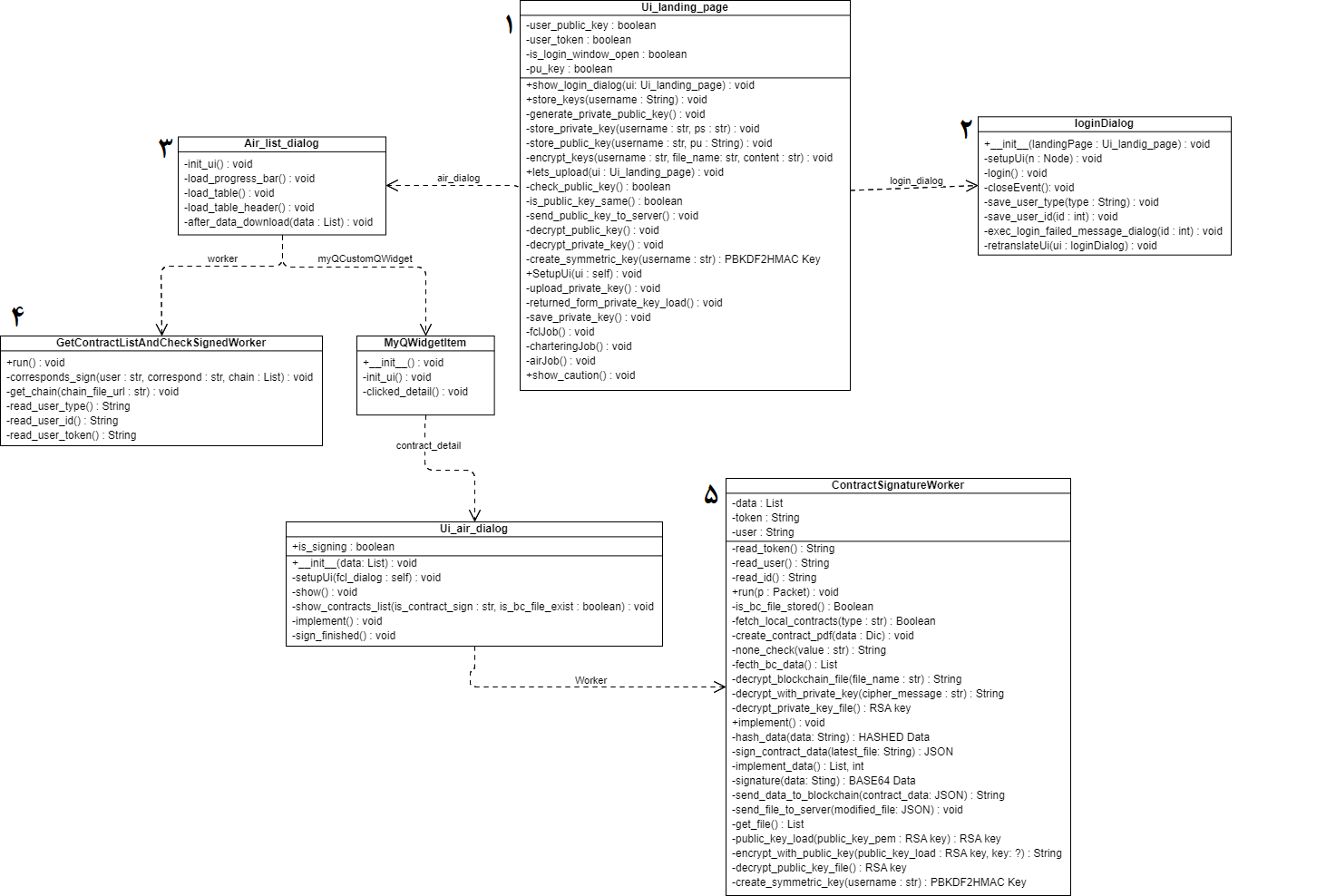
داده‌هایی که در قالب دیکشنری نگهداری می‌شود مطابق زیر است:

* key: آیدی هر قرارداد در بلاکچین (یکتا)
* data: داده قرارداد بدون امضا کاربر
* data\_signed: داده‌‌س امضا شده قرارداد توسط کاربر
* username: نام کاربری
* public\_key: کلید عمومی کاربر (کلید متناظر با کلید خصوصی که با آن قرارداد امضا شده است)

داده‌ها بعد از پردازش و آماده شدن، جهت ذخیره سازی در شبکه بلاکچین، به سمت سرور ارسال می‌شوند تا با بررسی آنها داده‌های مربوطه در شبکه ذخیره شوند. مکانیزم اجرایی چارچوب جهت ذخیره‌سازی داده‌ها در بخش پیاده سازی سرور توضیح داده شده است.

* **نمودار کلاس زیرخدمت Air**

جهت پیاده‌سازی این زیرخدمت در نرم افزار، نمودار کلاس آن رسم شده است که در تصویر زیر قابل مشاهده می‌باشد. کلاس شروع کننده کلاس Ui\_landing\_page می‌باشد که در فایل main.py پیاده‌سازی شده است. در فایل main.py سه زیرخدمت ذکر شده فعال شده است. همانطور که در نمودار مورد کاربرد مشاهده کردید، زیرخدمت Air تنها در یک مورد کاربرد متفاوت است. در نتیجه نمودار کلاس آن با دو زیرخدمت دیگر یکسان است و تنها تعدادی از توابع آن در کلاس contractSignatureWorker تغییر کرده است یا تابعی اضافه شده است. به عبارت دیگر در این کلاس، نحوه پردازش داده و ارسال آن تغییر یافته است. سایر خدمات ارائه شده در این نرم افزار از سمت کاربر با سایر ریزخدمت‌ها یکسان می‌باشد.



**شکل ‏3‑11: نمودار کلاس زیرخدمت Air**

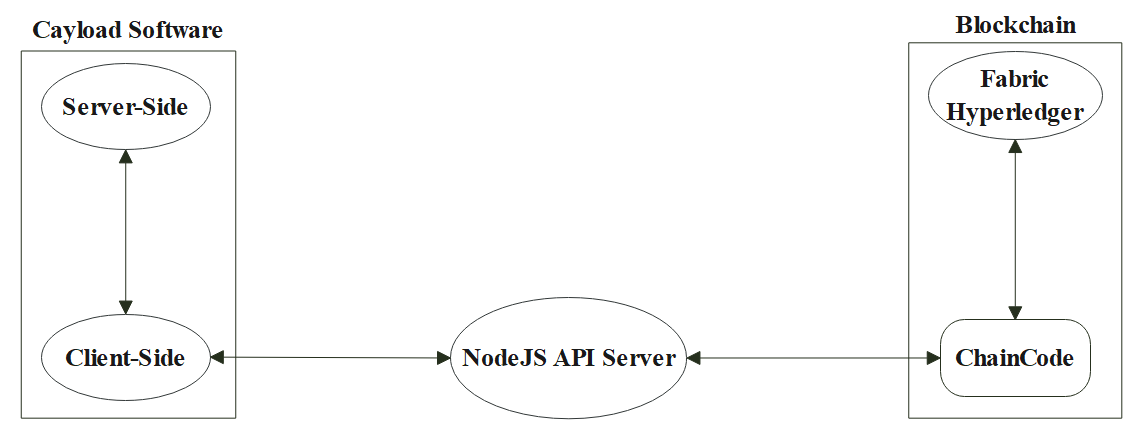
**جدول ‏3‑2: نگاشت توابع به سند**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| نام مورد کاربرد | قابلیت | نام کلاس | | نام متد |
| اضافه کردن قرارداد به زنجیره | پردازش کردن داده‌ها | contractSignatureWorker | 5 | implement\_data() |
| امضا کردن قرارداد | contractSignatureWorker | 5 | signature() |
| ارسال قرارداد به زنجیره | contractSignatureWorker | 5 | send\_data\_to\_blockchain() |
| مشاهده قرارداد | ایجاد جدول قراردادها | Air\_list\_dialog | 3 | init\_ui()  load\_progress\_bar()  load\_table()  load\_table\_header() |
| نمایش لیست قراردادها | GetContractListAndCheckSignedWorker | 4 | run()  corresponds\_sign() |
| نمایش به صورت PDF | contractSignatureWorker | 5 | run()  create\_contract\_pdf() |

### 3-2-2- پیاده‌سازی سمت سرور

در این زیرخدمت برای ذخیره‌سازی داده‌ها از چارچوب Hyperledger Fabric استفاده شده است. این بخش از سیستم، به دلیل کمبود سرور تهیه شده توسط کارفرما، چارچوب مذکور در یک سرور اجرا شده است. لازم به ذکر است با توجه به کارکرد بلاکچین مربوط به چارچوب ذکر شده، می‌توان برنامه مورد نظر را در چندین سرور اجرا کرد که این کار باعث افزایش امنیت زنجیره‌ی بلاکچین می‌شود.

پیاده‌سازی سمت سرور از دو بخش تشکیل شده است. بخش اول مربوط به چارچوب است و بخش دوم مربوط به پیاده‌سازی سرور جهت برقراری ارتباط با فریم. جزئیات هر یک از بخش‌ها را در ادامه خواهیم داشت. بطور کلی معماری نرم افزار به شرح زیر است:



**شکل ‏3‑12: شماتیک روابط بین اجزای سازنده سمت سرور**

با توجه به مکانیزم عملکرد چارچوب Fabric Hyperledger، برای دسترسی به زنجیره‌ی بلاک‌ها لازم است به کمک یک قرارداد هوشمند نحوه برقراری ارتباط با بلاکچین را مشخص کرد. به عبارت دیگر، هرگونه اقدام برای دسترسی به بلاکچین باید در قرارداد هوشمند برنامه نویسی شود. به عنوان مثال، اگر بخواهیم به یکی از بلوک‌های موجود در بلاکچین دسترسی داشته باشیم، لازم است تابعی را جهت دریافت بلوک مورد نظر در قرارداد هوشمند پیاده‌سازی کرده باشیم. در اصطلاح به این قرارداد هوشمند کد زنجیره‌ای گفته می‌شود.

در سیستم مورد نظر، برای جستوجو کردن بر روی زنجیره، اضافه کردن داده بر روی آن یا دریافت یک قرارداد خاص، سه تابع جداگانه در قرارداد هوشمند پیاده‌سازی شده است که در **Error! Reference source not found.** آمده است. لازم به ذکر است، با توجه به نحوه عملکرد چارچوب، از آنجایی که این قرارداد هوشمند باید بر روی بلاکچین نصب شود، با هر بار تغییر کد لازم است سیستم بلاکچین مجدد راه‌اندازی شود که این بدین معناست زنجیره جدیدی ایجاد می‌شود. در نتیجه باید در پیاده‌سازی آن دقت لازم را داشت چرا که بعد از نصب، امکان تغییر آن با حفظ اطلاعات موجود در بلاکچین وجود ندارد.

**جدول ‏3‑3: نگاشت توابع قرارداد هوشمند نصب شده بر روی بلاکچین**

|  |  |  |
| --- | --- | --- |
|  | **قابلیت** | **واسط برنامه‌نویسی کاربردی** |
| قرارداد‌ هوشمند )Chaincode( | مقداردهی اولیه زنجیره | InitLedger() |
| جستوجوی داده‎‌ی خاص | queryContract() |
| جستوجوی تمام داده‌ها | queryAllContract() |
| اضافه کردن داده به زنجیره | addContract() |

از سمت دیگر، با توجه به **Error! Reference source not found.**، برای برقراری ارتباط بین بلاکچین و برنامه تحت دسکتاپ لازم است برنامه‌ای به عنوان واسط وجود داشته باشد تا بتواند درخواست‌ها از سمت برنامه تحت دسکتاپ را به قرارداد هوشمند ارسال کند. از آنجایی که قراردادهوشمند تنها با چند زبان محدود برنامه نویسی امکان برقراری ارتباط را دارد، در نتیجه این برنامه به زبان NodeJS که یکی از زبان‌های قابل قبول قرارداد هوشمند است، پیاده‌سازی شده است. این برنامه در قالب واسط برنامه‌نویسی کاربردیREST زده شده است. برای برقراری ارتباط، سه واسط برنامه‌نویسی کاربردی زده شده است که در جدول زیر آورده شده است. این برنامه برای برقراری ارتباط با قرارداد هوشمند بازم است اطلاعات شبکه‌ای را که قرارداد هوشمند در آن قرار دارد را داشته باشد و به شبکه متصل شود. به همین منظور تابعی پیاده سازی شده است که به کمک یک کتابخانه، نوشته شده توسط تیم سازنده‌ی چارچوب Fabric Hyperledger، به نام fabric-network می‌توان یک دروازه اینترنت ایجاد کرد که با شبکه مربوطه متصل شد.

**جدول ‏3‑4: نگاشت توابع مربوط به برنامه واسط**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **قابلیت** | **واسط برنامه‌نویسی کاربردی/تابع** | |
| NodeJS APIs | دریافت تمام قراردادها | query() | GET |
| دریافت قرارداد خاص | queryContract() | GET |
| اضافه کردن قرارداد | addContract() | POST |
| اتصال به شبکه بلاکچین | configNetwork () | Method |

در چارچوب Fabric Hyperledger، برای راه اندازی شبکه بلاکچین از داکر استفاده شده است. در این زیرخدمت، شبکه بلاکچین از سه گره مجازی تشکیل شده که نام آن‌ها Org1، Org2 و Orderer1 است. برای راه اندازی شبکه، دو فایل bash نوشته شده است که به کمک آن فرایند مربوطه، به صورت خودکار و پشت سر هم اجرا می‌شود. در ابتدا فایل startFabric.sh را اجرا می‌کنیم. در این فایل، ابتدا اگر شبکه‌ای موجود باشد آن را غیرفعال می‌کند؛ سپس شبکه جدیدی را ایجاد می‌کند. سپس کانال جدیدی را ایجاد میکند تا به کمک آن گرههای نام برده شده به یکدیگر متصل شوند و با یکدیگر در تعامل باشند. برای ایجاد کانال جدید لازم است که فایل اصلی به نام network.sh اجرا شود؛ که همانطور که گفته شد این کارها بصورت خودکار اجرا می‌شود و لازم به وارد کردن دستورات مورد نظر به صورت دستی نیست. در فایل network.sh توابعی نوشته شده است که به شرح زیر است. لازم به ذکر است که بعضی از این توابع به یک فایل bash دیگر ارجاع داده شده است. به عبارتی دیگر، فایل network.sh فایل اجرایی اصلی به حساب می‌آید.

**جدول ‏3‑5: نگاشت توابع در راه‌اندازی شبکه بلاکچین**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **قابلیت** | **توابع** | **فایل Bash** |
| network.sh | پاک کردن ظرف‌های غیرفعال داکر | clearContainers() | - |
| پاک کردن ایمیج‌های اضافی داکر | clearUnwantedImages() | - |
| بررسی داشتن پیش‌فرض‌ها | checkPrereqs() | - |
| ایجاد گره | createOrgs() | registerEnroll.sh  ccp-generate.sh |
| ایجاد شبکه جدید | createConsortium() | - |
| فعال کردن شبکه | networkUp() | - |
| ایجاد کانال | createChannel() | createChannel.sh |
| نصب قرارداد هوشمند به کانال | deployCC() | deployCC.sh |
| غیرفعال کردن شبکه | networkDown() | - |

لازم به ذکر است که شبکه‌ی گفته شده یک شبکه مجازی می‌باشد که امکان دسترسی به آن تنها به افرادی خاص داده شده است و عموم مردم امکان دسترسی به این شبکه را ندارند. برای برقراری ارتباط با قرارداد هوشمند باید احراز هویت صورت بگیرد. در این برنامه تنها دو کاربر مجازی امکان دسترسی به شبکه بلاکچین را دارند (یکی مدیر شبکه و دیگری کاربر عادی) که به کمک یکی از این دو کاربر امکان دسترسی به بلاکچین داده می‌شود. به عبارت دیگر، برنامه واسط برای هر درخواست باید به واسط یکی از این دو کاربر درخواست خود را به قرارداد هوشمند ارسال کند.

جهت مشاهده سایت و دانلود برنامه بلاکچین می‌توانید به دو لینک زیر مراجعه کنید.

[آدرس سایت](https://sinax.info/)

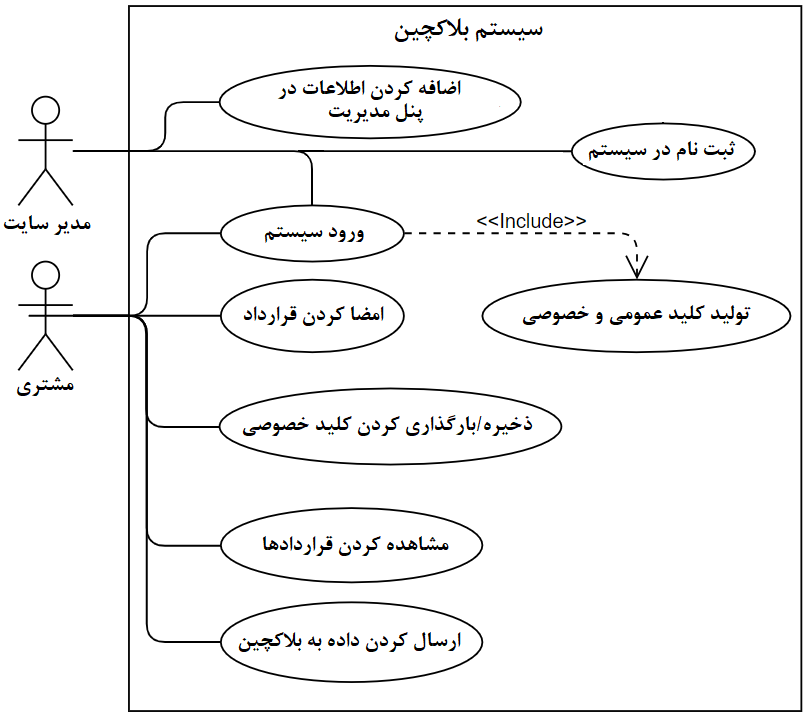
[لینک دانلود برنامه بلاکچین تحت دسکتاپ](http://dev5.cayload.com/static/cayloadblockchain.msi)

## 3-3- خروجی سوم: نرم‌افزار جامع ذخیره‌سازی امن داده‌های سایر سیستم‌های نرم‌افزاری با استفاده از چارچوب Hyperledger Fabric

باتوجه به روش‌های پیاده‌سازی شده در دو بخش قبل، از آنجایی که پروژه مورد نظر برای یک سایت خاص پیاده‌سازی شده است؛ در نتیجه قالب داده‌ها مشخص بود و بصورت مستقیم در کد برنامه وارد شده بود. از این رو این نرم افزار بصورت جامع قابل استفاده نخواهد بود. در جهت استفاده از این راهکار به صورت جامع، نرم افزار جامع ذخیره‌سازی داده طراحی شده است که دیگر سایت‌ها نیز قابلیت استفاده از آن را داشته باشند.

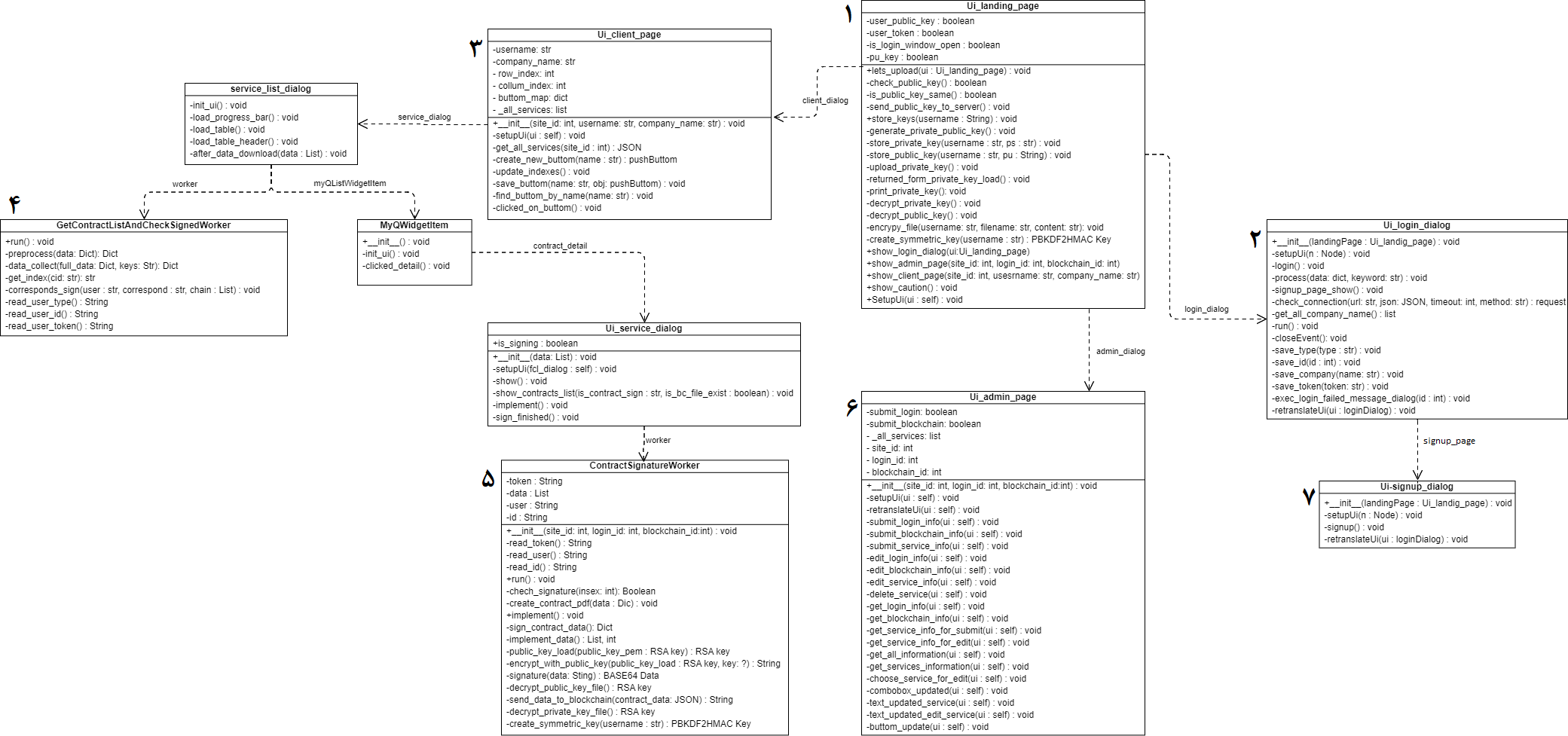
روش پیاده‌سازی در این نرم‌افزار مطابق روش پیاده شده در زیرخدمت Air است، با این تفاوت که این امکان وجود دارد که اکثر شرکت‌ها یا سایت‌ها در این نرم افزار ثبت نام کنند و از قابلیت ذخیره‌سازی داده در محیط امن بلاکچین استفاده کنند. به عبارت دیگر، در این نرم افزار جهت ذخیره‌سازی داده‌ها از چارچوب Hyperledger Fabric استفاده شده است. بنابراین تغییراتی در این بخش وجود ندارد.

جهت راه‌‌اندازی درست این نرم افزار برای هر شرکت، لازم بود در پیاده‌سازی سمت مشتری قابلیت ثبت نام شرکت/سایت و اضافه کردن اطلاعات مورد نیاز توسط مدیر شرکت/سایت اضافه شود. نمودار موردکاربر این نرم افزار در **Error! Reference source not found.** قابل مشاهده است. مطابق تصویر زیر تنها دو مورد کاربرد به آن اضافه شده است و سایر مورد کاربردها مطابق زیرخدمت Air می‌باشد.



**شکل ‏3‑13: نمودار کاربرد برنامه جامع ذخیره‌سازی داده**

* **مورد کاربر ثبت نام:** در این مورد کاربر، مدیر سایت/شرکت می‌تواند در این نرم افزار ثبت نام کند.
* **مورد کاربرد ورود به سیستم:** در این مورد کاربر، مدیر سایت/شرکت می‌تواند بعد از ثبت نام در این نرم افزار وارد پنل مدیریت شود.
* **مورد کاربرد اظافه کردن اطلاعات در پنل مدیریت:** مدیر شرکت ثبت نام شده در این نرم افزار موظف است اطلاعات لازم را تکمیل کند. به این منظور پروفایل مدیر در این نرم افزار طراحی شده است تا به کمک آن اطلاعات مورد نیاز را در هر بخش تکمیل کند. به منظور ذخیره اطلاعات مربوط به هر شرکت، پایگاه‌داده‌ای طراحی شده است تا اطلاعات مربوطه را ذخیره کند.

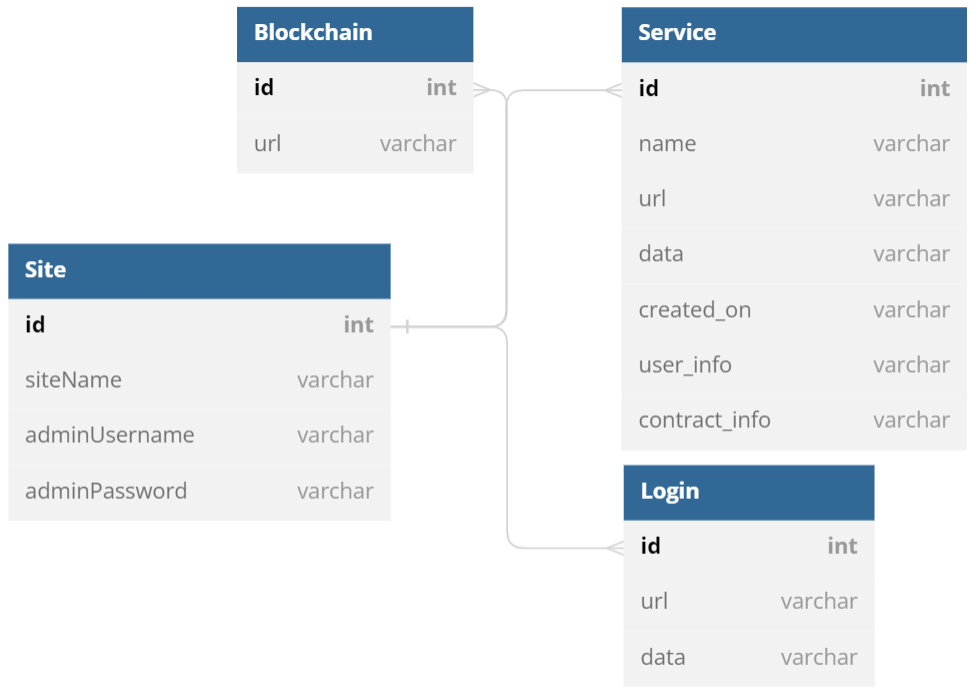
جهت پیاده سازی این نرم افزار، نمودار کلاس آن رسم شده است که در **Error! Reference source not found.** قابل مشاهده می‌باشد. کلاس شروع کننده کلاس Ui\_landing\_page می‌باشد که در فایل main.py پیاده سازی شده است.

**شکل ‏3‑14: نمودار کلاس مربوط به برنامه جامع ذخیره‌سازی داده**

**جدول ‏3‑6: نگاشت توابع مربوط به برنامه جامع ذخیره‌سازی داده**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **نام مورد کاربرد** | **قابلیت** | **نام کلاس** | | **نام متد** |
| ثبت نام | ثبت نام مدیردر نرم افزار | Ui\_signup\_page | 7 | Signup() |
| ورود به سیستم | نمایش اولیه صفحه ورود | Ui\_landing\_page | 1 | show\_login\_dialog() |
| فرایند احراز هویت | Ui\_loging\_dialog | 2 | login() |
| ذخیره اطلاعات در سیستم میزبان | Ui\_loging\_dialog | 2 | save\_user\_type()  save\_user\_id() |
| بررسی وضعیت کلیدها | Ui\_landing\_page | 1 | lets\_upload()  check\_public\_key()  is\_public\_key\_same() |
| تولید کردن کلید عمومی/خصوصی | تولید کلیدها | Ui\_landing\_page | 1 | generate\_private\_public\_key()  store\_private\_key()  store\_public\_key() |
| رمزنگاری کلیدها | Ui\_landing\_page | 1 | encrypt\_keys()  create\_symmetric\_key() |
| اضافه کردن قرارداد به زنجیره | ایجاد کردن بلوک | contractSignatureWorker | 5 | create\_new\_block() |
| امضا کردن قرارداد | contractSignatureWorker | 5 | signature() |
| جایگزین کردن زنجیره بلوک جدید در سرور | contractSignatureWorker | 5 | consensus() |
| ذخیره/بارگذاری کلید خصوصی | ذخیره کردن کلید | Ui\_landing\_page | 1 | save\_private\_key() |
| بارگذاری کردن کلید | Ui\_landing\_page | 1 | upload\_private\_key() |
| مشاهده قرارداد | ایجاد جدول قراردادها | Ui\_client\_page | 3 | init\_ui()  load\_progress\_bar()  load\_table()  load\_table\_header() |
| نمایش لیست قراردادها | GetContractListAndCheckSignedWorker | 4 | run()  corresponds\_sign() |
| نمایش به صورت PDF | contractSignatureWorker | 5 | run()  create\_contract\_pdf() |

همانطور که گفته شده، برای ذخیره‌سازی اطلاعات مربوط به هر شرکت پایگاه‌داده‌ای طراحی شده است که نمودار آن مطابق **Error! Reference source not found.**است.



**شکل ‏3‑15: نمودار پایگاه‌داده برنامه جامع ذخیره‌سازی داده**

# ارزیابی

در این بخش به ارزیابی سیستم نهایی، نرم‌افزار جامع ذخیره‌سازی ایمن داده‌های سایر سیستم‌های نرم‌افزاری پرداخته شده است. جهت ارزیابی این سیستم، زیرخدمت Air از پروژه بین المللی حمل و نقل در نظر گرفته شده است. جهت ارزیابی فرض‌هایی درنظر گرفته شده است که به شرح زیر می‌باشد:

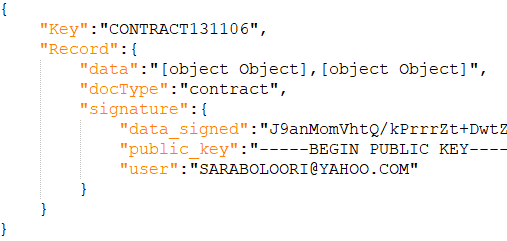
1. اطلاعاتی نظیر واسط‌های برنامه‌نویسی کاربردی برای احراز هویت کاربر، تعامل با چارچوب Hyperledger Fabric و دریافت داده‌ها وجود ندارد.
2. نوع داده‌هایی که باید ذخیره شود مشخص نیست

با در نظر گرفتن فرضیات فوق، ابتدا در سیستم به عنوان مدیر ثبت نام انجام شد. بعد از ورود به سیستم به عنوان مدیر، وارد پنل مدیریت شده و تمام اطلاعات مورد نیاز وارد شد. اطلاعات وارد شده در پایگاه‌داده طراحی شده ثبت و ذخیره شد.

سپس به عنوان یکی از کاربران وبسایت حمل و نقل بین‌المللی، وارد سیستم شده و به پنل کاربر دسترسی پیدا کردم. با توجه به اینکه اطلاعات زیرخدمت Air توسط مدیر به سیستم داده شده بود، این بخش برای کاربر قابل مشاهده بود و می‌توانست تمام قراردادهای مربوط به زیرخدمت Air را مشاهده کند. سپس با انتخاب یکی از قراردادها توسط کاربر، محتویات قرارداد در قالب PDF نشان داده می‌شود. سپس با زدن دکمه signature عملیات ذخیره‌سازی انجام می‌شود.

جهت بررسی ذخیره‌سازی صحیح، یک واسط برنامه‌نویسی کاربردی برای واکشی اطلاعات ذخیره شده در بلاکچین تهیه شده است که در جدول ‏4‑1 قابل مشاهده است.

فرایند گفته شده بر روی یک کاربر با نام کاربری [SARABOLOORI@YAHOO.COM](mailto:SARABOLOORI@YAHOO.COM) صورت گرفته است. مطابق شکل ‏4‑1، قرارداد با شناسه 131 را مشاهده می‌کنید که با کمک واسط برنامه‌نویسی کاربردی از بلاکچین واکشی شده است. این قرارداد توسط کاربر ذکر شده به عنوان یکی از طرفین قرارداد امضا و بلوک آن ایجاد و به زنجیره اضافه شده است.



**شکل ‏4‑1: نمونه خروجی واسط برنامه‌نویسی کاربردی نرم افزار جامع ذخیره‌سازی داده**

در نتیجه می‌توان گفت این نرم افزار برای سایت‌هایی که امکان ارسال داده در قالب JSON را داشته باشند، قابل استفاده است.

**جدول ‏4‑1: واسط برنامه‌نویسی کاربردی ارزیابی نرم‌افزار جامع ذخیره‌سازی امن داده**

|  |  |
| --- | --- |
| **نام زیرخدمت/برنامه** | **واسط برنامه‌نویسی کاربردی** |
| برنامه جامع ذخیره‌سازی داده | <https://b1.cayload.com:443/data/query> |

# نتیجه‌گیری و کارهای آتی

تا به امروز شاهد پیشرفت چشمگیر علم کامپیوتر، اکتشافات و اختراعات آن در زمینه‌ها و صنایع مختلف بوده‌ایم. همچنین جهان به سوی تسهیل بخشیدن و شفاف‌سازی فرایندهای کاری در صنایع مختلف از جمله صنایع غذایی و کشاورزی، فرایندهای دولتی، صنعت بیمه، صنعت بهداشت و درمان و ... بوده است.

یکی از فناوری‌های کاربردی، فناوری بلاکچین است که در سال‌های اخیر رشد بالایی داشته است و در حوزه‌های مختلف، مالی و غیرمالی، مورد استفاده و استقبال قرار گرفته است. در این پروژه جهت ذخیره‌سازی اطلاعات از این فناوری استفاده شده است.

در ابتدا برای دو زیرخدمت FCL و Chartering مربوط به پروژه حمل و نقل بین المللی، فناوری بلاکچین بدون استفاده از ابزارهای موجود بلاکچین، پیاده‌سازی شده است. پیاده‌سازی انجام شده دارای مشکلات تغییر پذیری، مقیاس پذیری و پردازش محاسباتی سمت کاربر (به عبارت دیگر، برای انجام فرایند ذخیره سازی داده، کاربر نهایی موظف به پردازش محاسباتی بدون مشوق مالی بود) بود.

به منظور حل این مشکل، از چارچوب Hyperledger Fabric در زیرخدمت Air استفاده شده است. تمام فرایندهای بلاکچین توسط این چارچوب صورت میگیرد. در نتیجه کاربر نهایی موظف به پردازش محاسبات ریاضیاتی نخواهد بود و در فرایندهای بلاکچین هیچ دخالتی نخواهد داشت. همچنین از آنجایی که ذخیره‌سازی تمام قراردادها در یک زنجیره صورت میگیرد، مشکل تغییر پذیری نیز تا حد بسیار بالایی رفع شده است. و در آخر از آنجایی که چارچوب Hyperledger Fabric مشکل مقیاس پذیری را حل کرده است، در نتیجه این مشکل در پروژه ما رفع شده است.

از آنجایی که مشکل خرابی سیستم مرکزی یکی از مشکلات موجود در سیستم‌های متمرکز است، فناوری بلاکچین این مشکل را حل کرده است. در این راستا، چارچوب Hyperledger Fabric نیز به دلیل داشتن قبلیت توزیع‌پذیری این امکان را مهیا کرده است که با برقراری شبکه در بین سرورهای مختلف از سازمان‌های مختلف، در برابر مشکل خرابی سیستم مرکزی مقاوم باشد. اما در این پروژه به دلیل کمبود سرور، شبکه ایجاد شده به صورت شبکه مجازی بر روی یک سرور می‌باشد. در نتیجه می‌‌توان گفت که این پروژه همچنان از مشکل خرابی سیستم مرکزی رنج می‌برد و با خراب شدن سرور مذکور، شبکه بلاکچین دچار مشکل می‌شود. بنابراین ایجاد کردن شبکه موجود بر روی چندین سرور در الوریت کارهای آینده خود قرار داده‌ایم که بتوانیم امنیت بالاتری را برای داده‌های ذخیره شده مهیا کنیم.

# راهنمای فنی

در این بخش جهت اجرای برنامه راهنمای فنی آن اضافه شده است که با توجه به دو نوع پیاده‌سازی انجام شده، دو راهنمای فنی خواهیم داشت. در ابتدا به بخش فرانت وبسایت که در هر سه زیرخدمت کارهای یکسانی صورت می‌گیرد، پرداخته شده است. سپس به بخش سرور می‌پردازیم. در انتها بخش کاربر را بررسی خواهیم کرد.

## راهنمای فنی – فرانت وبسایت Cayload

در ابتدا لازم است قراردادهای منعقد شده توسط طرفین قرارداد تایید نهایی شود. به این منظور لازم است به کمک متغیری در سمت سرور از تایید نهایی شدن قرارداد مطلع شویم. به همین دلیل در وبسایت مورد نظر، در انتهای هر قرارداد دکمه‌ای قرار داده شده است. با کلیک کردن دکمه مورد نظر، تابع handleFinishByRole() اجرا خواهد شد (تابع ذکر شده و دکمه‌ی مورد نظر در فایل‌های مربوط به بخش فرانت که آدرس آنها در **جدول ‏6‑1** آورده شده است، تعریف شده‌اند). به کمک این تابع، متغیر finished\_by\_provider یا finished\_by\_customer که در سمت سرور تعریف شده است به حالت True تغییر پیدا می‌کند. عملیات تابع ذکر شده در هر سه زیرخدمت یکسان است و تنها در واسط برنامه‌نویسی کاربردی ارسال داده متفاوت می‌باشند.

**جدول ‏6‑1: آدرس فایل فرانت زیرخدمت‌ها**

|  |  |
| --- | --- |
| **نام زیرخدمت** | **آدرس فایل** |
| FCL | src/scenes/dashboard/shipping/fcl/info.js |
| Chartering | src/scenes/dashboard/shipping/chartering/quatation.js |
| Air | src/scenes/dashboard/ air/info.js |

## راهنمای فنی – سمت سرور

### 6-2-1- زیرخدمت FCL و Chartering – سمت سرور

با توجه به نوع پیاده‌سازی، بخش سرور دو زیرخدمت FCL و Chartering زبان پایتون و چارچوب Django زده شده است. از آنجایی که فرایند این دو زیرخدمت در بخش سرور بصورت یکسان عمل می‌کند، تنها به زیرخدمت FCL پرداخته شده است. فایل‌های سمت سرور هر زیرخدمت در ***جدول ‏6‑2*** آورده شده است.

**جدول ‏6‑2: آدرس فایل سرور زیرخدمت‌ها**

|  |  |
| --- | --- |
| **نام زیرخدمت** | **آدرس فایل** |
| FCL | core/models/services/shipping/fcl.py |
| Chartering | core/models/services/shipping/chartering.py |

همانطور که در بخش قبل گفته شد، زمانی که هر دو متغیر finished\_by\_provider و finished\_by\_customer برابر True باشند، در این صورت فایل بلاکچین آن که شامل بلوک جنسیس است به کمک تابع blockchain\_on\_finished\_contract() تولید خواهد شد.

از طرفی دیگر، زمانی که قراردادی در نرم افزار بلاکچین توسط کاربر امضا می‌شود، بلوک مربوط به قرارداد مذکور ساخته شده و به سرور ارسال می‌شود. بلوک ارسال شده از سه جهت مورد بررسی قرار می‌گیرد که شامل صحت هش تولید شده، صحت زنجیره تولید شده و صحت امضا می‌شود. در صورتی که هر سه این موارد صحیح باشند، بلوک مورد نظر روی زنجیره قرار میگیرد. در جهت بررسی صحت موارد گفته شده توابع طبق ***جدول ‏6‑3*** فراخوانی می‌شوند.

**جدول ‏6‑3: نگاشت توابع سمت سرور**

|  |  |  |  |
| --- | --- | --- | --- |
| **جزئیات کد** | **توضیح نحوه عملکرد** | **عملیات** | **تابع** |
| پیوست 1 | در این تابع به کمک هش SHA256 و داده‌های خام، هش تولید شده و با هش ثبت شده در بلوک مقایسه می‌شود | صحت‌سنجی هش | valid\_hash() |
| پیوست 2 | در این تابع هش هر بلوک یا مقدار هش ثبت شده در بلوک قبلی مورد مقایسه قرار می‌گیرد | صحت‌سنجی زنجیره | chain\_validity() |
| پیوست 3 | در این تابع به کمک کلید عمومی کاربر و داده‌های خام، صحت سنجی امضا مورد بررسی قرار می‌گیرد | صحت‌سنجی امضا | sign\_validity() |

### 6-2-2- زیرخدمت Air – سمت سرور

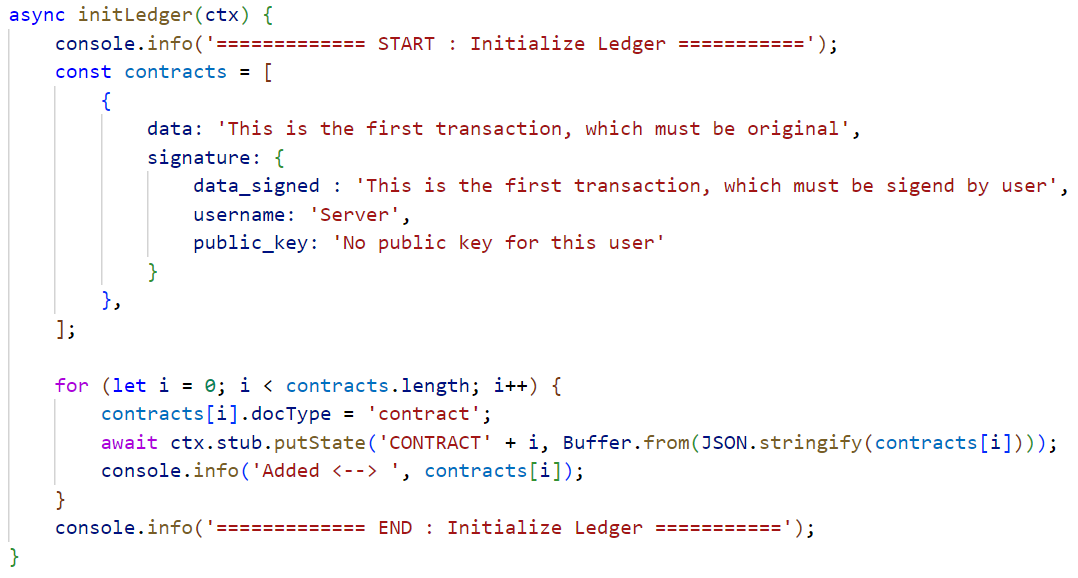
از آنجا که در زیرخدمت Air از چارچوب Hyperledger Fabric استفاده شده است، پیاده‌سازی آن با سایر زیرخدمت‌ها متفاوت شده است. در بخش سرور این زیرخدمت لازم است علاوه بر راه‌اندازی چارچوب Hyperledger Fabric، برنامه‌ای جهت برقراری ارتباط بین این چارچوب با برنامه تحت دسکتاپ به زبان NodeJS نوشته شود.

چارچوب Hyperledger Fabric این قابلیت را دارد تا عملیاتی را که لازم داریم بر روی بلاکچین انجام شود را تعریف کنیم. برای اینکار لازم است قرارداد هوشمندی را تعریف کنیم تا عملیات مورد نیاز ما را اجرا کند. در این پروژه، اضافه کردن داده جدید، جستوجوی بلوک خاص و دریافت تمام بلوک‌ها در قرارداد هوشمند تعریف شده است.

با توجه به **Error! Reference source not found.**، جهت انجام عملیات مختلف با چارچوب Hyperledger Fabric لازم است که در یک قرارداد هوشمند عملیات مجاز برای برقراری با بلاکچین مورد نظر با زبان NodeJS پیاده‌سازی شود. این قرارداد هوشمند بعد از نصب شدن دیگر قابلیت تغییر ندارد؛ بنابراین در پیاده‌سازی آن باید دقت لازم را داشته باشیم، زیرا درصورتی که لازم باشد کد آن تغییر کند، تمام اطلاعات موجود در بلاکچین از بین خواهد رفت.

از طرفی برای برقراری ارتباط با این قرارداد هوشمند، برنامه‌ای نوشته شده تا بتواند به این قرارداد هوشمند متصل شود و عملیات مورد نیاز کاربر را انجام دهد. این برنامه‌ با زبان NodeJS و با REST API زده شده است.

در ابتدا به عملیات‌های تعریف شده در قرارداد هوشمند می‌پردازیم. در زبان برنامه نویسی NodeJS کتابخانه‌ای به نام fabric-contract-list وجود دارد که به کمک آن میتوان به Hyperledger Fabric متصل شد. در این برنامه کلاسی به نام Cayload تعریف شده است. در سازنده‌ی آن اولین بلوک ساخته می‌شود که اطلاعات ذخیره شده در آن بصورت دستی وارد شده است.



**شکل ‏6‑1: سازنده کلاس Cayload در قرارداد هوشمند**

در این کلاس سه تابع تعریف شده است که طبق **جدول ‏6‑4** می‌باشد. این برنامه بعد از نصب شدن دیگر قابل تغییر نمی‌باشد، به عبارت دیگر اگر بعد از نصب شدن بلاکچین بر روی سرور، تابعی به این کلاس اضافه شود، تابع اضافه شده اعمال نمی‌شود و قابل اجرا نخواهد بود.

**جدول ‏6‑4: نگاشت توابع قرارداد هوشمند**

|  |  |  |  |
| --- | --- | --- | --- |
| **جزئیات کد** | **توضیح نحوه عملکرد** | **عملیات** | **تابع** |
| پیوست 4 | در این تابع با ارسال آیدی قرارداد، بلوک قرارداد مورد نظر که در بلاکچین ذخیره شده است واکشی می‌شود | واکشی یک بلوک | queryContract() |
| پیوست 5 | در این تابع اطلاعات یک قرارداد جدید در یک بلوک قرار گرفته و در بلاکچین ذخیره می‌شود | اضافه کردن قرارداد | addContract() |
| پیوست 6 | در این تابع تمام بلوک‌های مربوط به قراردادها واکشی می‌شود | واکشی تمام بلوک‌ها | queryAllContract() |

بعد از پیاده‌سازی قرارداد هوشمند، لازم است چارچوب Hyperledger Fabric بر روی سرور (یا چندین سرور) نصب شود. در این پروژه به دلیل محدودیت سرور، چارچوب مورد نظر بر روی یک سرور راه‌اندازی شده است. در ابتدا لازم است زنجیره بلاکچین به همراه کانال ارتباطی آن ساخته شود. سپس قرارداد هوشمند تعریف شده بر روی سرور نصب می‌شود. با اجرای فایل startFabric.sh تمام عملیات مورد نیاز برای نصب بلاکچین بصورت خودکار صورت می‌گیرد. لازم به ذکر است جهت شبیه‌سازی بلاکچین بر روی چند سرور، شبکه مجازی ایجاد شده شامل سه نود، Org1 Or2 Orderer1، است.

بعد از نصب چارچوب Hyperledger Fabric به توضیح جزئیات برنامه سرور می‌پردازیم. قبل از اجرا برنامه سرور لازم است برای دسترسی به قرارداد هوشمند یک کاربر و یک ادمین ثبت نام کنند تا از طریق این دو نوع کاربر اجازه دسترسی به بلاکچین داده شود. برای ثبت نام این دو کاربر، دو فایل enrollAdmin.js و registerUser.js اجرا می‌کنیم.

بعد از اجرای دو فایل ذکر شده، برنامه سمت سرور با اجرای فایل app.js شروع به کار می‌کند. در برنامه سمت سرور سه واسط برنامه‌نویسی کاربردی جهت برقراری ارتباط با قرارداد هوشمند پیاده سازی شده است که در

**جدول *‏6‑5*** آمده‌اند. هر سه واسط برنامه‌نویسی کاربردی در ابتدا لازم است به شبکه بلاکچین متصل شوند که به این منظور تابع configNetwork() پیاده سازی شده است.



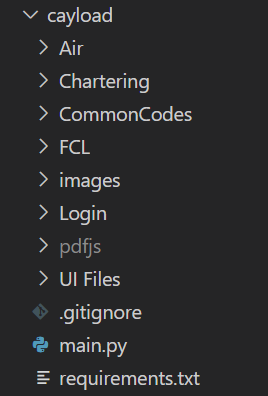
**شکل ‏6‑2: تابع configNetwork جهت برقرار ارتباط با شبکه بلاکچین**

**جدول ‏6‑5: نگاشت توابع سمت سرور زیرخدمت Air**

|  |  |  |  |
| --- | --- | --- | --- |
| **جزئیات کد** | **عملیات** | **واسط برنامه‌نویسی کاربردی** | |
|  | دریافت یک بلوک | GET | query() |
|  | واکشی تمام بلوک‌ها | GET | queryContract() |
|  | اضافه کردن قرارداد | POST | addContract() |

## راهنمای فنی – سمت مشتری

در پیاده‌سازی سمت مشتری، زیرخدمت Air با دو زیرخدمت FCL و Chartering در بعضی از بخش‌ها متفاوت هستند که در ادامه به هر یک پرداخته می‌شود. در **شکل ‏6‑3** پوشه‌بندی پیاده‌سازی سمت مشتری نمایش داده شده است. کدهای مربوط به هر زیرخدمت در پوشه‌‌ی مربوط به خود که هم‌نام با زیرخدمت است، قرار گرفته‌اند. در پوشه CommonCodes و images فایل‌هایی قراردارد که در هرسه زیرخدمت به صورت مشترک استفاده می‌شود. همچنین در پوشه Login فایل‌های مربوط به ورود به نرم افزار و پوشه‌ی UI Files شامل تمام UIهای نرم افزار می‌باشد که به کمک نرم افزار QT Designer طراحی شده‌اند. فایل اصلی و اجرایی main.py می‌باشد. در آخر، تمام کتابخانه‌های استفاده شده در فایل requirements.txt قرار گرفته است.



**شکل ‏6‑3**: **پوشه‌بندی سمت مشتری**

در ابتدا برای اجرا برنامه لازم است فرایند زیر اجرا شوند:

1. نصب پایتون (نسخه 3 و بالاتر)
2. نصب کتابخانه virtualenv به کمک ابزار pip
3. ساخت یک محیط مجازی به کمک دستور زیر جهت ایزوله بودن محیط پروژه از محیط اصلی سیستم جهت جلوگیری از تداخل کتابخانه‌های نصبی

pip install virtualenv

1. فعال‌سازی محیط مجازی به دستور زیر (در محیط ویندوز):

cd “virtualenv name”\Scripts\activate

1. نصب کتابخانه‌های مورد نیاز برنامه که در فایل requirements.txt قرار گرفته است. برای نصب این موارد می‌توان با اجرا دستور زیر در مسیری که فایل requirments.txt قرار دارد، تمام کتابخانه‌های مورد نیاز را نصب کرد:

pip install –r requirments.txt

1. اجرای فایل main.py با اجرای دستور:

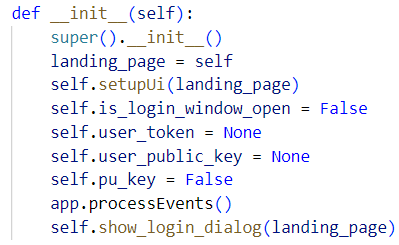
python main.py

هر فایل UI که شامل یک صفحه در نرم افزار است به کمک دستور زیر به فایل پایتون تبدیل می‌شود:

python -m PyQt5.uic.pyuic -x [FILENAME].ui -o [FILENAME].py

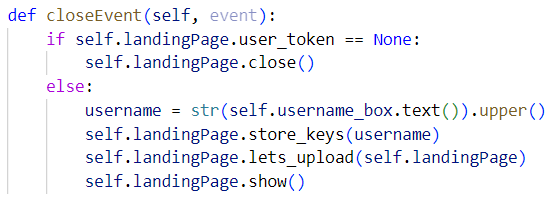
بنابراین در هر فایل، توابعی برای اجرای UI طراحی شده وجود دارد که مهم‌ترین آن setupUi(self, landing\_page) و retranslateUi(self, login\_dialog) می‌باشد. در ادامه به جزئیات بیشتر کد این بخش پرداخته شده است.

با اجرا شدن فایل main.py، سازنده‌ی مهم‌ترین کلاس با نام Ui\_lanfing\_page() اجرا خواهد شد. این کلاس صفحه‌ی اصلی نرم افزار (**شکل ‏3‑1**) را نشان می‌دهد. طبق **شکل ‏6‑4**، بعد از مقداردهی متغیرهای مورد نیاز، تابع show\_login\_dialog() فراخوانی می‌شود که به کمک آن صفحه‌ی مربوط به ورود کاربران نمایش داده می‌شود (**Error! Reference source not found.**).



**شکل ‏6‑4: سازنده کلاس Ui\_landing\_page()**

در این تابع (**پیوست 10**)، سازنده کلاس مربوط به ورود به نرم افزار فراخوانی می‌شود. بعد از ورود موفقیت آمیز کاربر به نرم افزار، صفحه مربوط به ورود بسته می‌شود که این فرایند، فراخوانی تابع closeEvent() را به همراه دارد که جزئیات آن ***شکل ‏6‑5*** آمده است.



**شکل ‏6‑5: تابع اجرایی بعد از اتمام عملیات ورود کاربر**

طبق این تابع، اگر ورود کاربر با موفقیت نباشد نرم افزار بسته می‌شود و وارد صفحه اصلی نمی‌شود. درغیراینصورت به ترتیب دو تابع store\_keys() و lets\_upload() اجرا می‌شود که فرایندهای اجرایی در **جدول ‏6‑6** آمده است. در انتها بعد از اجرا این دو تابع صفحه اصلی نرم افزار نمایش داده می‌شود.

**جدول ‏6‑6: نگاشت توابع اجرایی بعد از ورود موفقیت آمیز کاربر**

|  |  |  |
| --- | --- | --- |
| **جزئیات کد** | **نحوه‌ی عملکرد** | **تابع** |
| پیوست 11 | ساخت کلید عمومی و خصوصی و ذخیره سازی آن | store\_keys() |
| پیوست 12 | اگر کاربر برای اولین بار وارد نرم افزار شود، کلید عمومی به سرور ارسال می‌شود. درغیر اینصورت یکسان بودن کلید عمومی ذخیره شده در سرور و سیستم کاربر مورد بررسی قرار میگیرد | lets\_upload() |

همانطور که در **شکل ‏3‑1: نمایی از نرم‌افزار بلاکچین** و **Error! Reference source not found.** مشاهده می‌کنید، سه دکمه برای سه زیرخدمت FCL، Chartering و Air فعال است. با کلیک کردن هر یک از این دکمه‌ها، تابع مربوط به آن فراخوانی می‌شود (**جدول ‏6‑7**). هر یک از این توابع سازنده‌ی کلاس مربوط به زیرخدمت مورد نظر را فراخوانی می‌کند.

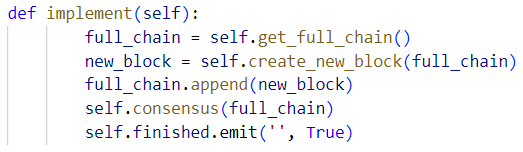
**جدول ‏6‑7: نگاشت توابع مربوط به دکمه‌های نرم افزار تحت دستکتاپ**

|  |  |
| --- | --- |
| **تابع فراخوانی شده بعد از کلیک شدن** | **نام دکمه** |
| fclJob() | FCL |
| charteringJob() | Chartering |
| airJob() | Air Cargo |

فرایند اجرایی هر سه این توابع یکسان است و تنها در واسط‌های برنامه‌نویسی کاربردی درخواستی متفاوت هستند. در این توابع، تمام قراردادهای کاربر در زیرخدمت مورد نظر از سرور دریافت می‌شود. که با توجه به اینکه تنها قراردادهایی که توسط طرفین قرارداد تایید شده‌اند قابل نمایش در نرم افزار است. در این بخش از فرایند صفحه‌ای جدید باز می‌شود که لیست قراردادهای تایید نهایی شده به کاربر نمایش داده می‌شود. طبق **Error! Reference source not found.** برای هر قرارداد یک دکمه تعبیه شده که با کلیک کردن بر روی آن جزئیات اطلاعات قرارداد در قالب PDF به کاربر نمایش داده می‌شود. جزئیات کد این بخش در **پیوست 13** آورده شده است که با توجه به یکسان بودن هر سه زیرخدمت در این بخش از کد، تنها کدهای مربوط به بهش Air آورده شده است.

همانطور که گفته شد، زمانی که کاربر بر روی دکمه جزئیات قرارداد کلیک می‌کند، صفحه‌ای جدید با محتوای جزئیات قرارداد مذکور در قالب PDF به همراه دکمه‌ای برای امضای دیجیتال کردن قرارداد باز می‌شود (**Error! Reference source not found.**) (**پیوست 14.** کاربر با زدن دکمه signature فرایند اجرایی در دو زیرخدمت FCL و Chartering با زیرخدمت Air متفاوت عمل خواهد کرد.

در دو زیرخدمت FCL و Chartering بعد از زدن دکمه signature، تابع implement اجرا می‌شود که در ابتدا برای اطلاعات قرارداد بلوکی ایجاد می‌شود که در این بین اطلاعات قرارداد با کلید خصوصی کاربر امضا می‌شود؛ سپس بلوک ساخته شده لازم است با الگوریتم اثبات کار استخراج شود. بعد از اینکه بلوک مورد نظر استخراج شد، آن را به زنجیره اضافه می‌کنیم. قبل از ارسال زنجیره جدید به سرور، لازم است زنجیره تولید شده با آخرین نسخه زنجیره زخیره شده در سرور مقایسه شود تا اگر در بین فرایند امضا شدن بلوک جدید اضافه شده بود فرایند ساخت بلوک مجدد اجرا شود.

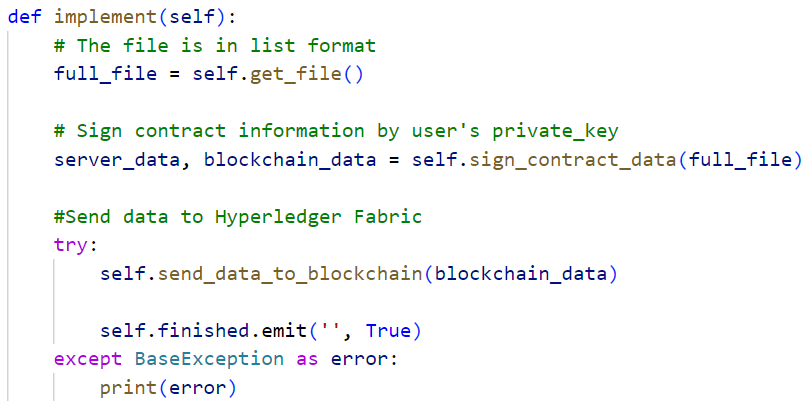


**شکل ‏6‑6: تابع implement مربوط به زیرخدمت FCL و Chartering**

**جدول ‏6‑8: نگاشت توابع فرایند امضا کردن قرارداد در زیرخدمت FCL و Chartering**

|  |  |  |
| --- | --- | --- |
| **جزئیات کد** | **نحوه‌ی عملکرد** | **توابع** |
| پیوست 15 | ساخت بلوک جدید و امضا کردن قرارداد با کلید خصوصی کاربر | create\_new\_block() |
| پیوست 16 | بررسی وضعیت زنجیره تولید شده و آخرین ورژن زنجیره ذخیره شده در سرور | consensus() |

در زیر خدمت FCL، بعد از زدن دکمه signature، تابع implement اجرا می‌شود که فرایند آن با دو زیر خدمت دیگر متفاوت است. از آنجایی که در این زیرخدمت از چارچوب Hyperledger Fabric استفاده شده تا عملیات ذخیره‌سازی قراردادها در این چارچوب صورت گیرد، دیگر فرایندهای قبلی اجرا نمی‌شود. در اینجا تنها لازم است ابتدا اطلاعات قرارداد امضا شود؛ سپس داده‌ها طبق آنچه که تعریف شده در قالب دیکشنری قرار گیرد. در انتها دیکشنری مورد نظر به بلاکچین Hyperledger Fabric ارسال می‌شود.



**شکل ‏6‑7: تابع implement مربوط به زیرخدمت Air**

**جدول ‏6‑9: نگاشت توابع فرایند امضا کردن قرارداد و ارسال به شبکه بلاکچین Hyperledger Fabric**

|  |  |  |
| --- | --- | --- |
| **جزئیات کد** | **نحوه‌ی عملکرد** | **توابع** |
| پیوست 17 | ایحاد دیکشنری و امضا کردن قرارداد با کلید خصوصی کاربر | sign\_contract\_data() |
| پیوست 18 | ارسال دیشکنری اطلاعات به بلاکچین | send\_data\_to\_blockchain() |

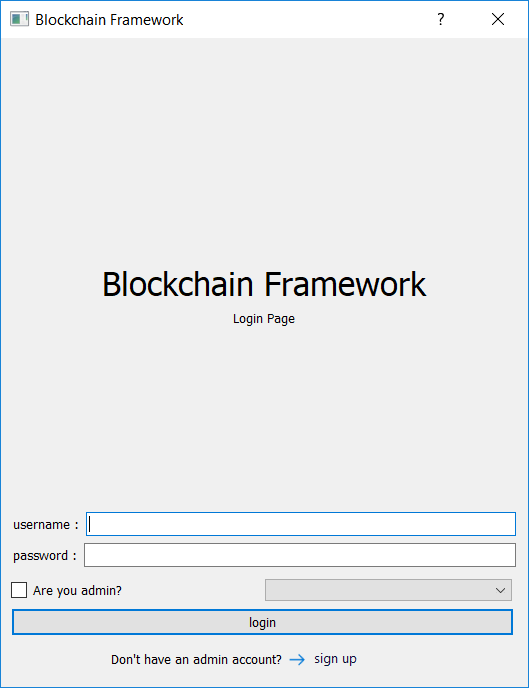
## راهنمای فنی نرم‌افزار جامع ذخیره‌سازی داده

در این بخش به جزئیات بیشتر پرداخته می‌شود. از آنجایی که بسیاری از فرایندهای این نرم‌افزار مطابق زیرخدمت Air می‌باشد. در نتیجه، جهت جلوگیری از تکرار مضاعف، راهنمای بخش‌های بسیاری نوشته نشده است. همانطور که در بخش راهکار پیشنهادی گفته شد، در این نرم‌افزار لازم است هر سایت/شرکت توسط مدیر آن ثبت شود. سپس مدیر سایت ثبت نام شده می‌تواند در این نرم‌افزار وارد شود و به پنل مدیریت دسترسی داشته باشد و اطلاعات مورد نیاز را وارد می‌کند. در ادامه به فرایند ثبت نام و ورود در بخش مشتری؛ سپس در بخش سرور به فرایند ثبت‌نام، ورود و پنل مدیریت پرداخته شده است.

* **ورود کاربر – سمت مشتری:**

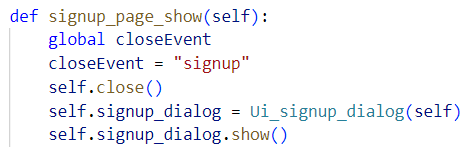
در این نرم افزار دو نوع کاربر، مدیر سایت و کاربر عادی، وجود دارد. درصورتی که کاربر مدیر سایت باشد، با ورود موفقیت آمیز به صفحه مدیریت سایت منتقل می‌شود. در صورتی که کاربر از نوع کاربر عادی باشد، با ورود موفقیت آمیز به صفحه‌ی نرم افزار منتقل می‌شود و در این صفحع قادر است سرویس‌های خود و قراردادهای مربوط به هریک را مشاهده کند.

ورود مدیر سایت: مدیر سایت جهت وارد شدن به صفحه مدیریت سایت لازم است نام کاربری و رمز عبور خود را وارد کند. همچنین لازم است از بین لیست شرکت‌های ثبت نام شده، شرکت خود را انتخاب کند. از آنجایی که کاربر مورد نظر مدیر است لازم است جعبه انتخاب را علامت بزند تا به عنوان مدیر سایت وارد نرم افزار شود. در نهایت با فشردن دکمه login عملیات مربوط به ورود کاربر توسط تابع login که در **پیوست 19** نشان داده شده است، اجرا خواهد شد.



**شکل ‏6‑8: صفحه ورود کاربران به نرم افزار**

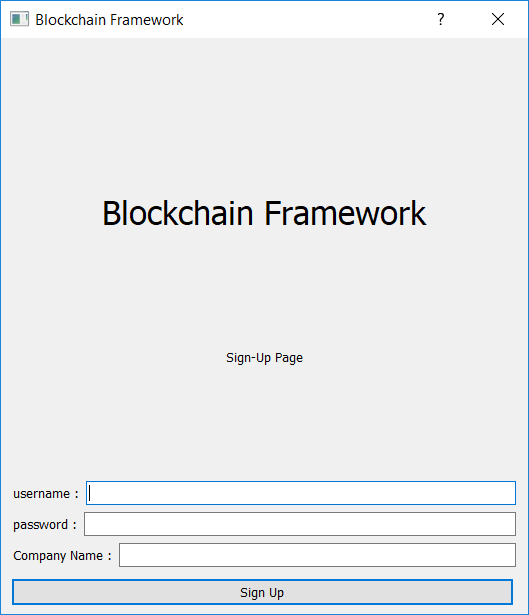
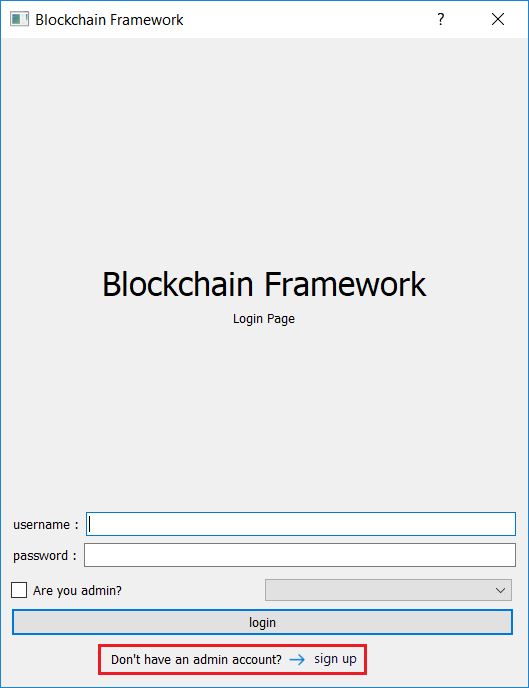
درصورتی که کاربر مورد نظر تمایل به ثبت نام داشته باشد می‌تواند با فشردن دکمه sign up به صفحه‌ی مربوط به ثبت نام کاربر منتقل شود. با فشردن این دکمه تابع زیر اجرا می‌شود که در آن از کلاس مربوط به صفحه‌ی ثبت نام یک شی ایجاد شده و آن را اجرا می‌کند.



**شکل ‏6‑9: تابع signup\_page\_show جهت نمایش صفحه ثبت نام مدیر سایت**

* **ثبت نام مدیر سایت – سمت مشتری:**

در این برنامه لازم است سایت مورد نظر جهت استفاده از آن، ثبت نام کند. بنابراین شخصی به عنوان مدیر سایت می‌تواند در این نرم افزار ثبت نام و نام شرکت خود را ثبت کند. به این منظور همانطور که در مشاهده می‌نید، در صفحه ورود کاربران، لینک sign up طراحی شده است که درصورتی که کاربر تمایل داشته باشد می‌تواند با کلیک بر آن به صفحه‌ی ثبت نام انتقال یابد.



**شکل ‏6‑10: صفحه ورود کاربر و ثبت نام**

\\

جهت ثبت نام سایت، تنها لازم است نام آن را ثبت کنند. سایر اطلاعات در صفحه مدیریت از مدیر سایت دریافت می‌شود. زمانی که کاربر دکمه “Sign Up” را فشار دهد تابع signup که در **شکل ‏6‑11** آمده است اجرا می‌شود. در این تابع اطلاعات وارد شده در قالب دیکشنری به سمت سرور ارسال می‌شود. درصورتی که ثبت نام کاربر موفقیت آمیز باشد، سرور کد 201 ارسال میکند؛ سپس پیام آن به کاربر نشان داده می‌شود و در نهایت به صفحه‌ی ورود کاربر منتقل می‌شود.

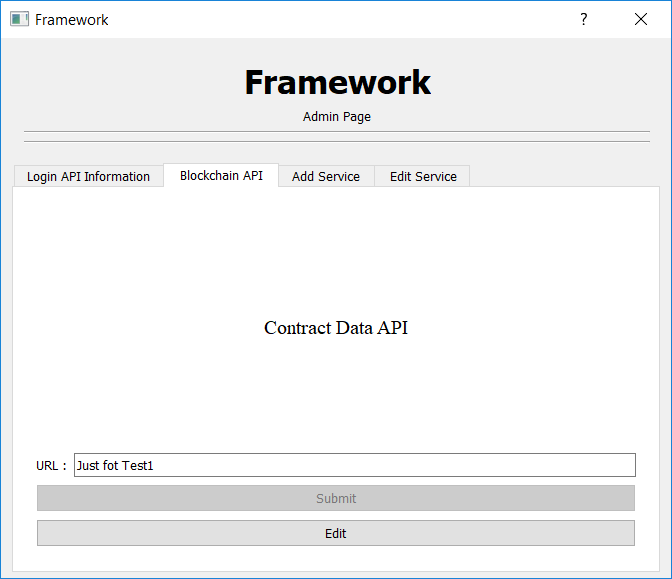
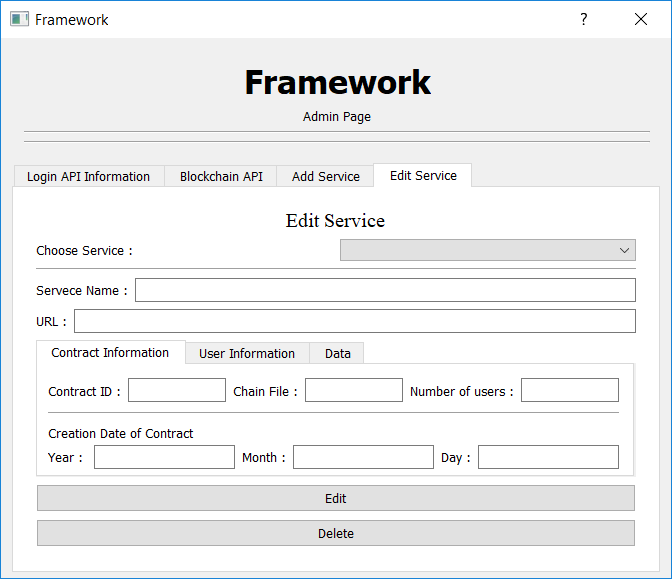
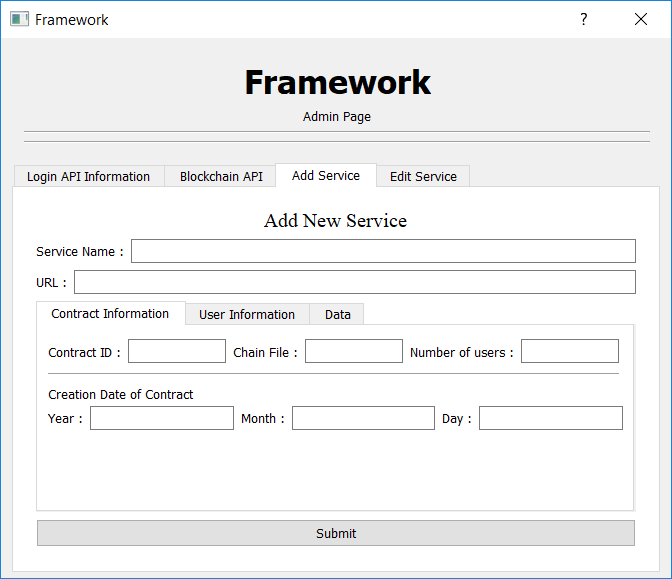
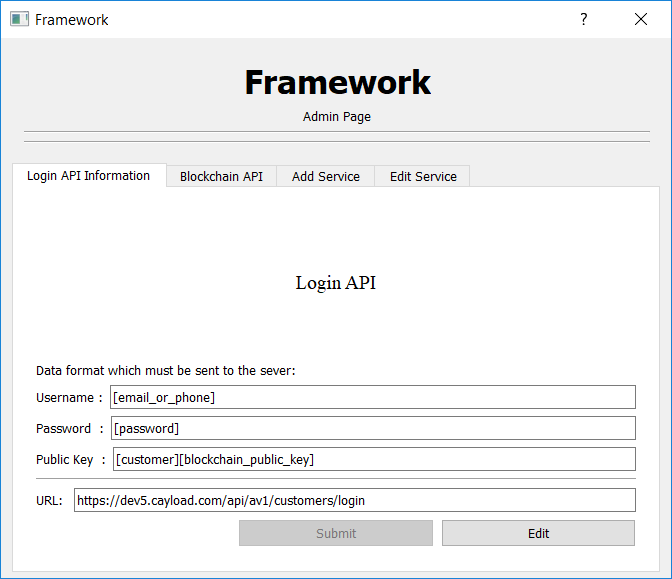


**شکل ‏6‑11: تابع signup جهت ثبت نام مدیر سایت**

لازم به ذکر است که نام شرکت ثبت شده باید یکتا باشد و یک شرکت نمی‌تواند دوبار با یک نام یکسان ثبت نام کند. درصورتی که یک سایت جهت ورود کاربران از چندین واسط برنامه‌نویسی کاربردی استفاده می‌کند، لازم است برای هر واسط برنامه‌نویسی کاربردی یک بار در نرم افزار و با نامی متفاوت ثبت نام کند.

**مورد کاربرد پنل مدیریت – سمت مشتری:**

در این مورد کاربرد، مدیر می‌تواند بعد از ورود به نرم‌افزار می‌تواند وارد پنل مدیریت شود و اطلاعات مورد نیاز را وارد کند. در شکل ‏6‑12 بخش‌های مختلف پنل مدیریت قابل مشاهده می‌باشد.



**شکل ‏6‑12: بخش‌های مختلف پنل مدیریت در برنامه جامع ذخیره‌سازی داده**

**زمانی که کاربر روی دکمه مربوط به هر سرویس میزنه:**

سازنده کلاس service\_list\_dialog فراخوانی میشود که این کلاس در فایل service\_contract\_list.py قرار دارد. در این کلاس یک شی از کلاس GetContractListAndCheckSignedWorker ساخته میشود و متد run آن اجرا میشود.

در متد run تمام قراردادهای مربوط به کاربر مورد نظر از سرور دریافت می‌شود. قراردادهای دریافتی باید در قالب تعریف شده باشد که به شرح زیر است:

{

    "services": [

        /\* ONE CONTRACT \*/

        {

            "id" : "INT",

            "blockchain\_chain\_file" : "BOOLEAN",

            "number\_of\_user" : "INT",

            "users" : {

                "user\_(ID)" : {

                    "signed" : "BOOLEAN", /\* This field is fill by application \*/

                    "id" : "USER\_ID"

                }

                /\* Based on the number of users, there are user\_() dictionaries \*/

            },

            "date\_created" : {

                "yaer": "",

                "month": "",

                "day": ""

            },

            "data" : {

                /\*\*/

            },

            "signed" : "(FULL OR SIGNED OR NOT)" /\* This field is fill by application \*/

        }

    ],

    "token" : "USER TOKEN"

}

زمانی که قراردادها دریافت شد، به ازای هر قرارداد بررسی میشود که آیا تمام کاربران قرارداد مذکور را تایید نهایی کرده‌اند یا نه. درصورتی که قرارداد توسط تمام طرفین قرارداد تایید نهایی شده باشد باید مقدار متغیر blockchain\_chain\_file برابر True باشد. اگر مقدار آن False باشد در اینصورت این قرارداد در لیست قراردادهای کاربر نمایش داده نمی‌شود.

درصورتی که قرارداد قابلیت نمایش در لیست کاربر را داشته باشد، با توجه به تعداد منعقد کنندگان قرارداد و افرادی که آن را امضای دیجیتال کرده‌اند، به قرارداد مذکور متغیری تحت عنوان signed اضافه میشود که میتواند سه مقدار FULL، USER و NOT داشته باشد. مقدار FULL برای زمانی است که تمام افراد طرفین قرارداد آن را امضای دیجیتال کرده باشند، مقدار USER زمانی است که تمام افراد امضا نکرده اند ولی کاربر وارد شده به نرم افزار قرارداد مذکور را امضا کرده است و مقدار NOT زمانی است که تمام افراد و خود کاربر امضا نکرده باشند.

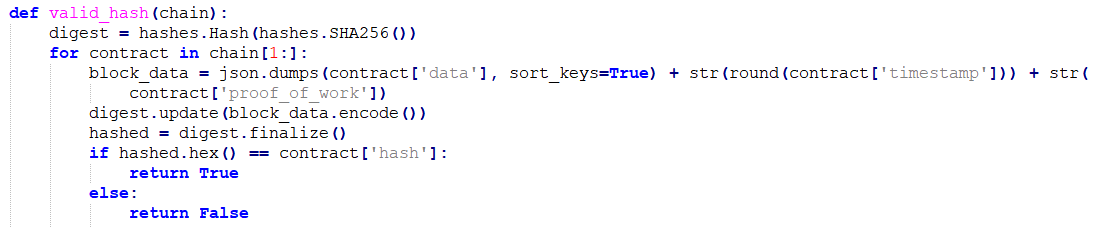
بعد از انجام این عملیات، لیست تمام قراردادهایی که قابلیت نمایش در لیست کاربر دارند به عنوان خروجی پاس داده میشود.

بعد از اجرای تابع run، تابع after\_data\_download اجرا می‌شود. در این متد با توجه به تعداد قراردادها، ردیف در لیست کاربر ایجاد می‌شود. در این تابع، به ازای هر قرارداد یک شی از کلاس MyQWidgetItem ساخته میشود که هر شی شامل اطلاعات قرارداد می‌باشد. در این کلاس برای هر قرارداد دکمه‌ای قرارداده شده است که با کلیک کردن بر روی آن اطلاعات مربوط به هر قرارداد به کلاس Ui\_service\_dialog پاس داده می‌شود؛ به عبارتی دیگر یک شی از این کلاس ساخته می‌شود و اطلاعات مربوط به قرارداد مذکور به سازنده‌ی آن پاس داده می‌شود. سپس توابعی دیگری از جمله init\_ui، load\_progress\_bar، load\_table و load\_table\_header به ترتیب اجرا می‌شوند.

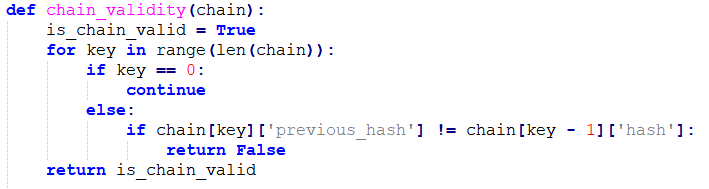
# مراجع

|  |  |
| --- | --- |
| [1] | Alan T. Sherman, Farid Javani and Haibin Zhang and Enis Golaszewski, "On the Origins and Variations of Blockchain Technologies," *IEEE Security & Privacy,* 2019. |
| [2] | Hany F. Atlam and Gary B. Wills, Chapter One - Technical aspects of blockchain and IoT, Advances in Computers, 2019. |
| [3] | "Blockchain," 2011. [Online]. Available: https://www.blockchain.com/en/charts/blocks-size?scale=1&timespan=all&showDataPoints=true.. |
| [4] | Damiano Di Francesco Maesa and Paolo Mori, "Blockchain 3.0 applications survey," *Journal of Parallel and Distributed Computing,* 2020. |
| [5] | Qalab E Abbas and Jang Sung-Bong, "A survey of blockcain and its applications," *International Conference on Artificial Intelligence in Information and Communication (ICAIIC),* 2019. |
| [6] | Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen and Huaimin Wang, "Blockchain challenges and opportunities: A survey," *International Journal of Web and Grid Services (IJWGS),* 2018. |
| [7] | Cornelius C. Agbo and Qusay H. Mahmoud, "Comparison of blockchain frameworks for healthcare applications," *Internet Technology Letters,* vol. 2, no. 5, p. e122, 2019. |
| [8] | Bhabendu Kumar Mohanta, Debasish Jena, SoumyashreeS. Panda and Srichandan Sobhanayak, "Blockchain technology: A survey on applications and security privacy Challenges," *Internet of Things,* 2019. |
| [9] | Margaret Rouse, Kaitlin Herbert and Sarah Lewis, "TechTarget," 1999. [Online]. Available: https://whatis.techtarget.com/definition/consensus-algorithm. |
| [10] | Sandeep Jain, "GeeksforGeek," [Online]. Available: https://www.geeksforgeeks.org/what-are-hash-functions-and-how-to-choose-a-good-hash-function/. |
| [11] | Mohammed Abdulhakim Al-Absi, Azamjon Abdullaev, Ahmed Abdulhakim Al-Absi, Mangal Sain and Hoon Jae Lee, "A survey of consensus algorithms in public blockchain systems for crypto-currencies," *Journal of Network and Computer Applications,* 2021. |
| [12] | "Hyperledger Fabric," The Linux Foundation, 22 January 2016 . [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/whatis.html. |
| [13] | L. Yang, "The blockchain: State-of-the-art and research challenges," *Journal of Industrial Information Integration,* 2019. |
| [14] | S. Jaiswal, "Javatpoint," 2011. [Online]. Available: https://www.javatpoint.com/blockchain-version. |
| [15] | A. Ghaffar Khan, A. H. Zahid, M. Hussain, M. Farooq, U. Riaz and T. M. Alam, "A journey of WEB and Blockchain towards the Industry 4.0: An Overview," *2019 International Conference on Innovative Computing (ICIC),* pp. 1-7, 2019. |
| [16] | A. Chang, N. El-Rayes and J. Shi, "Blockchain Technology for Supply Chain Management: A Comprehensive Review," *FinTech,* vol. 1, pp. 191-205, 2022. |
| [17] | Shihab S. Hazari and Qusay H. Mahmoud, "Comparative evaluation of consensus mechanisms," *Internet Technology Letters,* 2019. |
| [18] | Sandeep Jain, "GeeksForGeeks," [Online]. Available: https://www.geeksforgeeks.org/practical-byzantine-fault-tolerancepbft/. |
| [19] | "On the Security and Privacy of Hyperledger Fabric: Challenges and Open Issues," *IEEE World Congress on Services (SERVICES),* 2020. |
| [20] | "Medium," [Online]. Available: https://medium.com/@chamirachid/your-journey-to-consensus-part-2-raft-ab1fd88a9557. |
| [21] | Brandon Chez, "CoinMarketCap," world's most-referenced price-tracking website for cryptoassets , 2013. [Online]. Available: https://coinmarketcap.com/tokens/views/all/. |
| [22] | Fran Casino, Thomas K. Dasaklis and Constantinos Patsakis, "A systematic literature review of Blockchain-based applications," *Telematics and Informatics,* 2019. |
| [23] | "Solidity," [Online]. Available: https://docs.soliditylang.org/en/latest/. |
| [24] | "Go Ethereum," [Online]. Available: https://geth.ethereum.org/docs/getting-started. |
| [25] | "Remix - Ethereum IDE," [Online]. Available: https://remix-ide.readthedocs.io/en/latest/. |
| [26] | I. A. Qasse, M. A. Talib and A. B. Nassif, "Performance Analysis of Hyperledger Fabric Platforms," *Security and Communication Networks,* vol. 2018, p. 14 , 2018. |
| [27] | ShubhaniAggarwal and NeerajKumar, "Chapter Sixteen - Hyperledger," *Advances in Computers,* vol. 121, pp. 323-343, 2021. |
| [28] | H. Hasanova et al., "A survey on blockchain cybersecurity vulnerabilities," *International Journal of Network,* vol. 29, p. 2060, 2019. |
| [29] | Y. H. e. al., "Smart contract security: A software lifecycle perspective," *IEEE Access,* vol. 7, p. 150 184–150 202, 2019. |
| [30] | "Medium," [Online]. Available: https://medium.com/@chamirachid/your-journey-to-consensus-part-2-raft-ab1fd88a9557. |

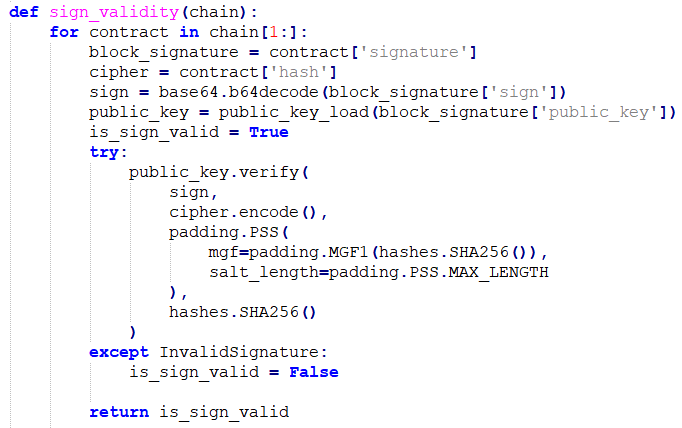
# پیوست



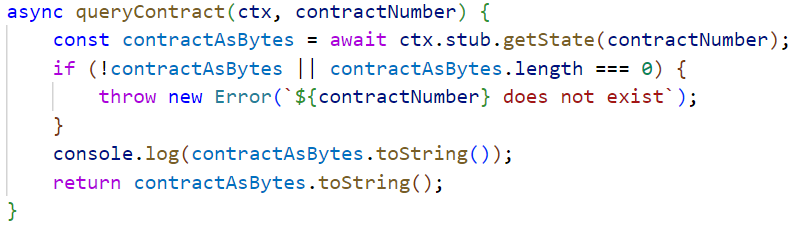
**پیوست 1. تابع valid\_hash در سمت سرور**



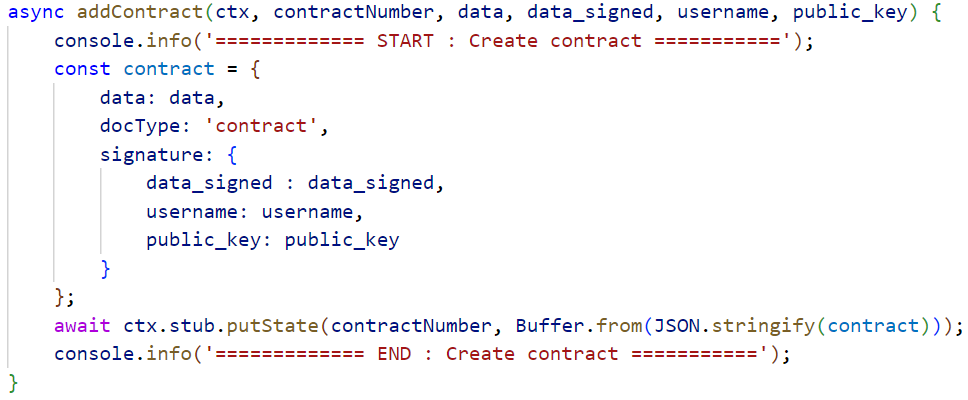
**پیوست 2. تابع chain\_validity در سمت سرور**



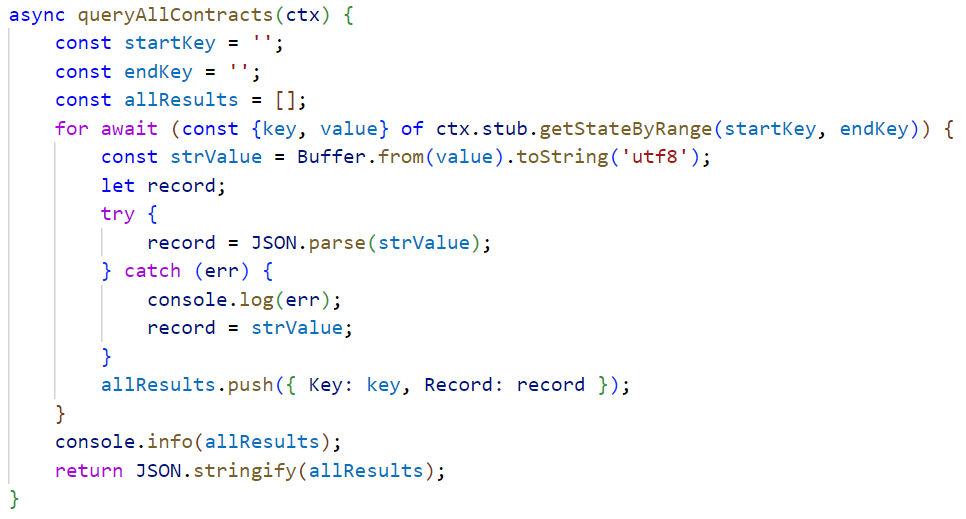
**پیوست 3. تابع sign\_validity در سمت سرور**



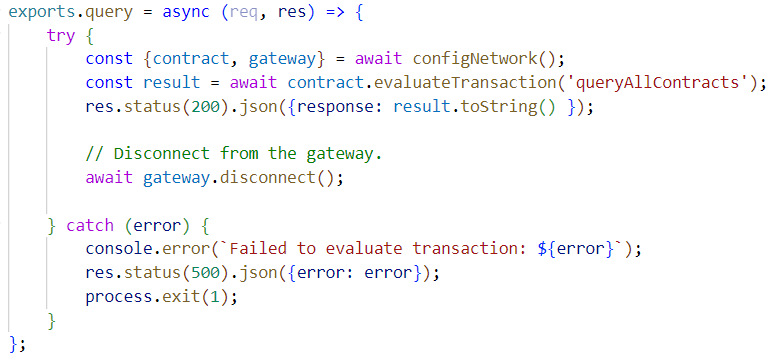
**پیوست 4. تابع queryContract در قرارداد هوشمند**



**پیوست 5. تابع addContract در قرارداد هوشمند**



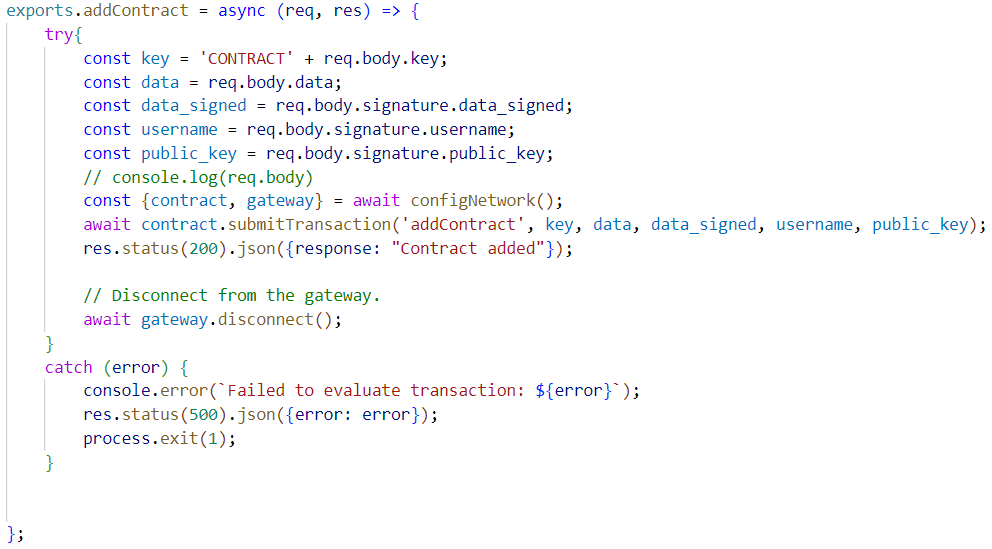
**پیوست 6. تابع queryAllContracts در قرارداد هوشمند**



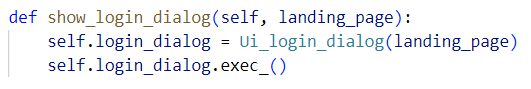
**پیوست 7. واسط برنامه‌نویسی کاربردی GET برای واکشی تمام بلوک‌ها**



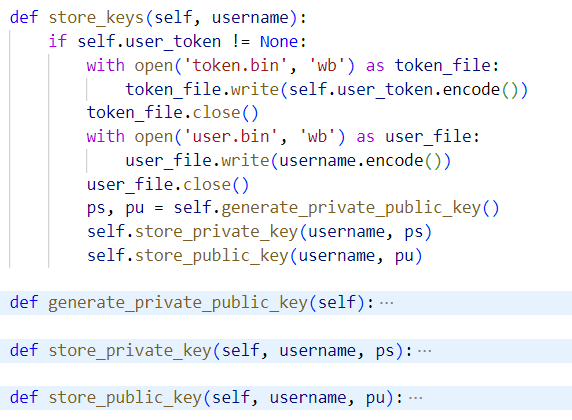
**پیوست 8. واسط برنامه‌نویسی کاربردی GET برای واکشی یک بلوک خاص**



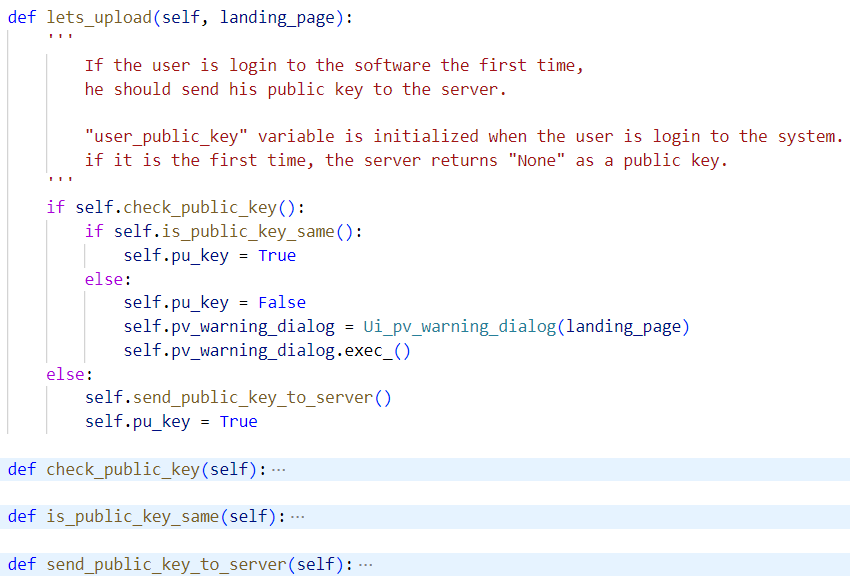
**پیوست 9. واسط برنامه‌نویسی کاربردی POST برای اضافه کردن قرارداد جدید به شبکه بلاکچین**



**پیوست 10. تابع show\_login\_dialog جهت اجرای صفحه ورود به نرم افزار تحت دسکتاپ**



**پیوست 11. تابع store\_keys و توابع مورد نیاز آن**

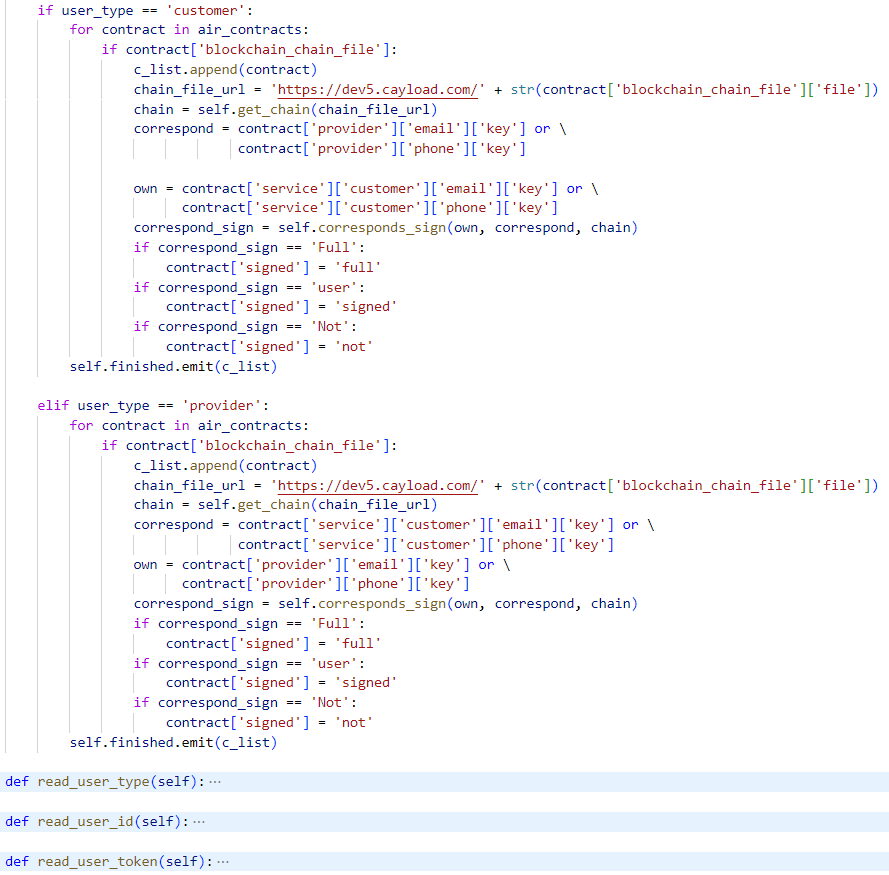


**پیوست 12. تابع lets\_upload و توابع مورد نیاز آن**

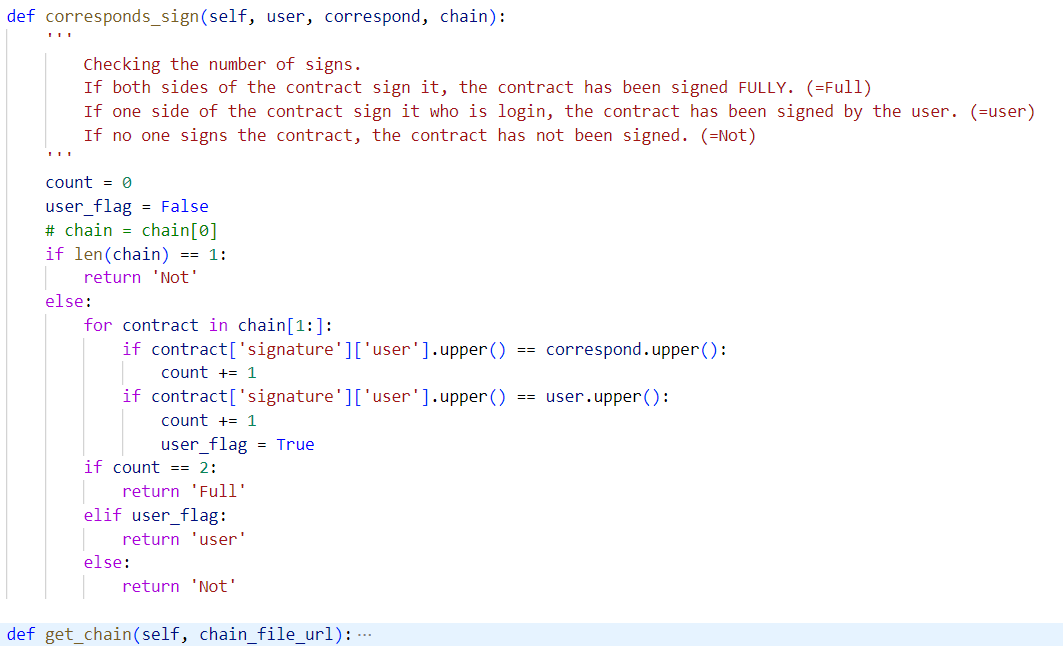
**پیوست 13.**

با کلیک کردن بر روی دکمه Air Cargo (یا سایر دکمه‌های مربوط دو زیرخدمت دیگر) تابع airJob اجرا می‌شود. که به دنبال آن تابع run اجرا می‌شود که در شکل زیر آورده شده است. در این تابع ابتدا تمام قراردادهای مربوط به Air دریافت می‌شود. با بررسی هر قرارداد، درصورت تایید نهایی قرار داد مورد نظر به لیست اضافه می‌شود تا در صفحه لیست قراردادها نمایش داده شود. همچنین در این تابع وضعیت قرارداد از جهت امضای دیجیتال شدن مورد بررسی قرار می‌گیرد.



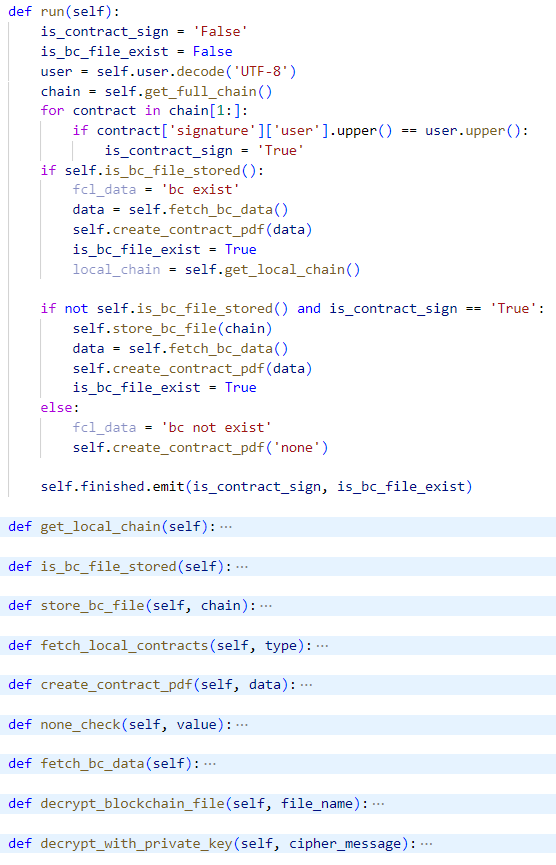


در شکل زیر، تابع corresponds\_sign وضعیت قرارداد مورد بررسی قرار می‌گیرد تا مشخص شود قرارداد مورد نظر بطور کامل توسط کاربران امضای دیجیتال شده، هیچ یک امضا نکرده‌اند یا امضا کننده قرارداد کاربر وارد شده می‌باشد.



**پیوست 14.**

کاربر با کلیک کردن بر دکمه Contract Details، تابع run که جزئیات آن در شکل زیر آمده است اجرا می‌شود. در این تابع بعد بررسی‌های لازم، PDF مربوط به قرارداد با تابع create\_contract\_pdf ساخته می‌شود.





**پیوست 15. ایجاد بلوک جدید و امضا کردن قرارداد در زیرخدمت FCL و Chartering**



**پیوست 16. بررسی وضعیت زنجیره تولید شده و زنجیره ذخیره شده در سرور**



**پیوست 17. امضا کردن قرارداد و ساخت دیکشنری اطلاعات آن در زیرخدمت Air**



**پیوست 18. ارسال داده‌های قرارداد به بلاکچین در زیرخدمت Air**



**پیوست 19. تابع login جهت ورود کاربران**

1. Non-Fungible Token (NFT) [↑](#footnote-ref-1)
2. Decentralize Application (Dapp) [↑](#footnote-ref-2)
3. Merkel Hash Tree [↑](#footnote-ref-3)
4. Satoshi Nakamoto [↑](#footnote-ref-4)
5. White-paper [↑](#footnote-ref-5)
6. Vitalik Buterin [↑](#footnote-ref-6)
7. Public Ledger [↑](#footnote-ref-7)
8. Peer to Peer [↑](#footnote-ref-8)
9. Integrity [↑](#footnote-ref-9)
10. Non-repudiation [↑](#footnote-ref-10)
11. Single Point of Failure [↑](#footnote-ref-11)
12. Miner [↑](#footnote-ref-12)
13. Block Header [↑](#footnote-ref-13)
14. Nonce [↑](#footnote-ref-14)
15. Consensus Algorithm [↑](#footnote-ref-15)
16. https://www.hyperledger.org/ [↑](#footnote-ref-16)
17. Immutability [↑](#footnote-ref-17)
18. Transparency [↑](#footnote-ref-18)
19. Efficiency [↑](#footnote-ref-19)
20. Validator [↑](#footnote-ref-20)
21. Back-End Development [↑](#footnote-ref-21)
22. Confidentiality [↑](#footnote-ref-22)
23. Legacy System Integration [↑](#footnote-ref-23)
24. Asymmetric [↑](#footnote-ref-24)
25. Symmetric [↑](#footnote-ref-25)
26. Diffie-Hellman [↑](#footnote-ref-26)
27. elliptic curve digital signature algorithm [↑](#footnote-ref-27)
28. Proof of Work (PoW) [↑](#footnote-ref-28)
29. Double-spending [↑](#footnote-ref-29)
30. Byzantine Fault Tolerance (BFT) [↑](#footnote-ref-30)
31. Sybil Attack [↑](#footnote-ref-31)
32. Halt [↑](#footnote-ref-32)
33. Crash Fault Tolerance (CFT) [↑](#footnote-ref-33)
34. Leader [↑](#footnote-ref-34)
35. Follower [↑](#footnote-ref-35)
36. Candidate [↑](#footnote-ref-36)
37. Remote Procedure Call (PRC) [↑](#footnote-ref-37)
38. Grid computing [↑](#footnote-ref-38)
39. Etheruem Virtual Machine (EVM) [↑](#footnote-ref-39)
40. Integrated Development Environment (IDE) [↑](#footnote-ref-40)
41. Permissioned [↑](#footnote-ref-41)
42. Peer [↑](#footnote-ref-42)
43. Orderer [↑](#footnote-ref-43)
44. Flexibility [↑](#footnote-ref-44)
45. Confidentiality [↑](#footnote-ref-45)
46. Modular [↑](#footnote-ref-46)
47. Configurable [↑](#footnote-ref-47)
48. Proof of work [↑](#footnote-ref-48)
49. Nonce [↑](#footnote-ref-49)
50. Application Program Interface (API) [↑](#footnote-ref-50)