به نام خدا

دانشگاه صنعتی اصفهان

دانشکده برق و کامپیوتر

پروژه درس مبانی فناوری اطلاعات

Smart Contract

استاد

دکتر محمد حسین منشئی

اعضای گروه

یوسف زارع ده آبادی

سارا سلطانی

عرفان نجفی

سبحان صفدریان

محمد اسماعیل محمدزاده

ترم دوم تحصیلی 1401-1400

چکیده

این گزارش ابتدا به تعریف بلاک چین و معرفی انواع آن پرداخته. سپس به معرفی قرارداد های هوشمند به عنوان یکی از کاربردهای بلاک چین پرداخته و در ادامه آسیب پذیری های آن معرفی شده و ابزاری نیز در جهت تشخیص این آسیب پذیری ها مطرح شده. در ادامه نحوه اجرای این قرارداد ها در بستر ماشین مجازی بررسی می شود. همچنین در رابطه با تاثیر قرارداد های هوشمند در شکل گیری شهر هوشمند بحث شده و در ادامه در رابطه با ارتباط بلاک چین و اینترنت اشیا در دنیای امروز صحبت شده. در نهایت کاربردهای قرارداد هوشمند در سه حوزه بیمه و زنجیره تامین و مدیریت هوشمند املاک بررسی و مورد بحث قرار گرفته و یک پیاده سازی در زمینه قرارداد های هوشمند انجام شده.

کلمات کلیدی : بلاک چین، قرارداد هوشمند، اتریوم، بیت‌کوین، بیمه، زنجیره تامین، مشاور املاک.

فهرست مطالب

مقدمه

انسان همواره به دنبال استفاده از تکنولوژی برای تسهیل زندگی و در نتیجه بهبود کیفیت زندگی خود بوده است. از جمله این تلاش‌ها می‌توان به استفاده از تکنولوژی به جهت انعقاد قراردادهای مختلف اشاره کرد. امروزه قراردادها معمولا به شیوه‌های سنتی منعقد می‌شوند. به عنوان مثالی از این شیوه می‌توان به قراردادهای کاغذی اشاره کرد. در این شیوه، بندهای قرارداد روی کاغذ نوشته شده و دو طرف قرارداد پس از توافق با بندهای درج شده در قرارداد، با امضای قرارداد، خود را موظف به رعایت بندهای آن می‌کنند. امروزه تلاش می‌شود تا با استفاده از تکنولوژی بلاکچین و قراردادهای هوشمند، چنین قراردادهایی به طور دیجیتال انجام شوند.

بلاکچین زنجیره ای از بلوک‌ها بوده که در صورتی که تعداد زیادی از افراد در نگهداری آن سهیم باشند، این تضمین را فراهم می‌کند که داده‌های موجود در آن هیچگاه تغییر نخواهند کرد. همچنین داده‌های این زنجیره عموما توسط همگان قابل بازبینی هستند. بنابراین با ذخیره قراردادهای موجود در بلاکچین ضمن وجود شفافیت فرایند، تضمین می‌شود که مفاد این قراردادها تغییر نخواهند کرد.

این قراردادها عموما به صورت برنامه‌های قابل اجرا می‌باشند. به این صورت که برنامه‌ای نوشته می‌شود و روی بلاکچین قرار می‌گیرد. این برنامه‌ها توسط ماینرها در شرایطی اجرا خواهند شد. به عنوان مثال فرض کنید سازمانی قراردادی را روی شبکه بلاکچین تنظیم می‌کند که با اجرای قرارداد در یک روز خاص از ماه حقوق کارمندان به حسابشان واریز می‌شود. امروزه اتریوم متداول ترین شبکه بلاکچین است که امکان ایجاد قراردادهای هوشمند را فراهم می‌سازد. متداول ترین زبان مورد استفاده در قراردادهای هوشمند این شبکه، solidity می‌باشد.

با توجه به اینکه این قراردادها برنامه‌های قابل اجرا می‌باشند، مانند هر برنامه دیگری می‌تواند مشکلاتی در منطق خود داشته باشد. به عنوان مثال، مشابه با برنامه‌های معمول، برنامه‌های قراردادهای هوشمند نیز ممکن است به مشکل integer overflow آسیب‌پذیر باشد. به همین منظور فریمورک‌هایی برای کشف این دسته از آسیب‌پذیری‌ها برای قراردادهای هوشمند ارائه شده است که به آن خواهیم پرداخت.

امروزه تلاش می‌شود از قراردادهای هوشمند در عرصه‌هایی نظیر بیمه، خرید و فروش املاک و همچنین زنجیره تامین استفاده شود. این گزارش همچنین به شیوه استفاده از قرارداد هوشمند در این عرصه‌ها می‌پردازد.

تعریف بلاک چین

بلاک چین زنجیره ای است از بلوک ها که این بلوک ها به واسطه روش های رمزنگاری به یکدیگر متصل شده اند. در این بلوک ها دیتا ذخیره می شود. حال اگر این زنجیره ها در شبکه قرار گیرند و افراد به آن شبکه دسترسی داشته باشند یعنی بلاک چین در کل شبکه توزیع شود شبکه بلاک چین را تشکیل می دهد.

آشنایی با انواع بلاک چین ها

از نقطه نظر معماری، انواع مختلفی از بلاک چین وجود دارد که از لحاظ مجوزهای خواندن/نوشتن تفاوت دارند:

1. بلاک چین های عمومی (مانند بلاک چین بیت کوین) بلاک چین هایی هستند که می توانند برای همه قابل خواندن و به طور بالقوه قابل نوشتن باشند. بلاک چین های عمومی زمانی مفید هستند که هیچ نهاد مرکزی برای تأیید یک تراکنش در دسترس نباشد.
2. بلاک چین های خصوصی بلاک چین هایی هستند که فقط توسط اعضای سازمان می توانند نوشته شوند. مجوزهای خواندن را می توان به سازمان محدود کرد یا عمومی کرد.
3. بلاکچین کنسرسیوم ترکیبی از بلاکچین عمومی و خصوصی است که شبکه بلاک چین در آن توسط بیشتر از یک فرد یا سازمان کنترل می‌شود. یعنی توسط یک گروه، و نه یک فرد خاص اداره می‌شود. در این نوع از بلاک چین فقط

به تعدادی از کاربران مجوز تایید کردن تراکنش‌ها داده می‌شود و به صورت متفاوت حقوق و وظایف و اختیارات تقسیم می‌شود. درواقع مجموعه‌ای از گره‌های انتخاب شده متعلق به مؤسسات مختلف اعتبارسنجی را کنترل می‌کنند و زنجیره بلوکی برای به اشتراک گذاشتن اطلاعات بین مؤسسات شرکت‌کننده استفاده می‌شود.

بلاک‌چین‌های خصوصی و کنسرسیومی مزایایی مانند هزینه‌های اعتبارسنجی پایین‌تر و زمان‌های اعتبارسنجی کوتاه‌تر (به دلیل تعداد گره‌های کوچک‌تر، مسائل ریاضی را می‌توان ساده‌تر کرد)، کاهش خطر حملات (از آنجایی که گره‌هایی که تراکنش‌ها را تایید می‌کنند شناخته شده‌اند) و افزایش حریم خصوصی (بر اساس دادن مجوز خواندن فقط به گره های انتخاب شده) ارائه می‌کنند. علاوه بر این، در صورت بروز خطا یا باگ در قراردادهای هوشمند، بلاک چین‌های خصوصی و کنسرسیومی می‌توانند تراکنش‌های قبلی را به‌طور فوق‌العاده تغییر دهند یا برگردانند.

تعریف قرارداد های هوشمند

خصوصیت‌های بلاکچین موجب شده تا در عرصه‌های دیگری نیز مورد استفاده قرار گیرد و به بهبود کیفیت زندگی انسان کمک کند. از جمله این عرصه‌ها می‌توان به قراردادهای هوشمند اشاره کرد. قراردادهای هوشمند به صورت دیجیتالی قراردادهای بسته شده بین دو یا چند نفر را در بلاک چین تأیید و اجرا می کنند و زمانی که یک اکشن یا رویداد خاصی که از پیش تعریف شده اتفاق بیفتد این قرارداد ها اجرا می شوند و می توانند ارزهای دیجیتال یا سایر دارایی های دیجیتال را انتقال دهند.

از آنجایی که قراردادهای هوشمند معمولاً روی بلاک چین مستقر می شوند و توسط آن ایمن می شوند، ویژگی های منحصر به فردی دارند .کد برنامه یک قرارداد هوشمند بر روی بلاک چین ثبت و تأیید می‌شود، بنابراین قرارداد در برابر تغییر و یا حمله مقاوم می‌شود. همچنین یک قرارداد هوشمند به صورت غیر متمرکز و بدون هماهنگی شخص ثالث اجرا می‌شود (یعنی واسطه ها حذف میشوند).

آسیب پذیری های قرارداد های هوشمند

1. Transaction-Ordering Dependence (TOD) : ساختار هر بلوک شامل چندین تراکنش است و ترتیب تراکنش های اجرایی به ماینرها وابسته است که ماینرها می توانند ترتیب تراکنش ها را دستکاری کنند.
2. Timestamp Dependence : ماینرها برای هر بلاکی که استخراج می کنند یک timestamp تنظیم می کنند. Timestamp در قراردادهای هوشمند یک شرط راه اندازی برای تراکنش هایی مثل انتقال پول است. پس ماینر می تواند آن را در حد چند ثانیه دستکاری کند و این نوع قراردادها را آسیب پذیر کند.
3. Mishandled Exceptions : زمانی که یک قرارداد ، قرارداد دیگری را کال میکند اگر این قرارداد به درستی اجرا نشود خاتمه می یابد و false بر می گرداند. این مقداری که برمی گرداند به قرارداد اولی منتقل می شود. قرارداد اولی قاعدتا باید این مقدار را چک کند تا بررسی کند که آیا این قرارداد با موفقیت انجام شده یا نه. پس اگر قرارداد اولی به درستی این مقدار را چک نکند آسیب های احتمالی رخ خواهد داد.
4. Re-entrancy vulnerability : زمانی که یک قرارداد ، قرارداد دیگری را کال می کند اجرای قرارداد فعلی متوقف می شود تا فرآیند کال کردن خاتمه یابد. این مکانیزم بازگشتی به مهاجم کمک می کند که وارد قرارداد اولی شده و مکررا قرارداد دومی را کال کند که منجر به حلقه (لوپ بی نهایت) خواهد شد و می تواند بازپرداخت های متوالی و خالی کردن حساب طرف مقابل را به همراه داشته باشد. بدترین آسیب از این نوع حمله DAO است.
5. Callstack Depth : هر بار که یک قرارداد ، قرارداد دیگری را کال می کند پشته فراخوانی به اندازه یک فریم افزایش می یابد. مثلا پشته فراخوانی برای اتریوم به 1024 فریم محدود شده است که وقتی به این حد رسید، فراخوانی بیشتر ، یک استثنا ایجاد می کند. مهاجم با ایجاد یک پشته فراخوانی تقریباً کامل شروع می‌کند و سپس تابع قربانی

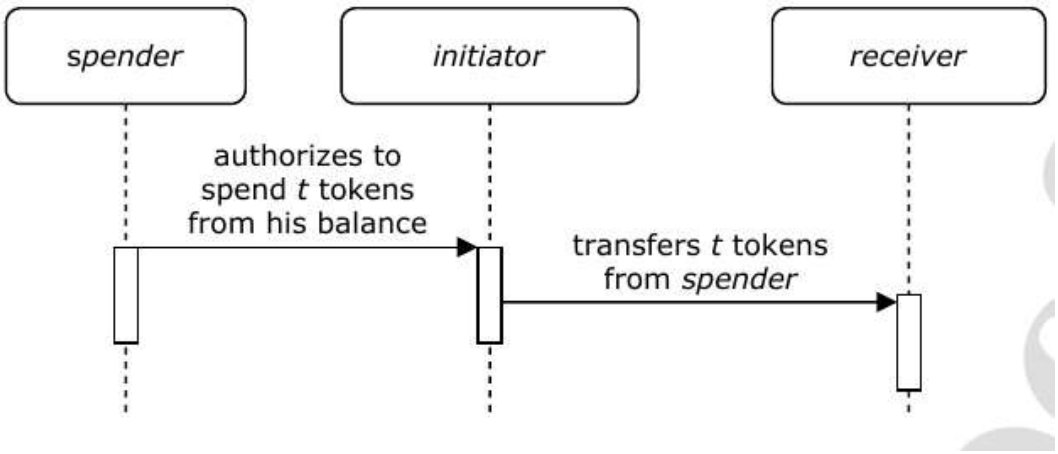
را فراخوانی کرده و یک استثنا ایجاد می‌کند. اگر این استثنا در قرارداد قربانی به درستی انجام نشود، مهاجم می تواند در حمله خود موفق شود.

ابزاری برای تشخیص آسیب پذیری ها

برخی از آسیب‌پذیری‌های قرارداد های هوشمند به منطق برنامه مرتبط است. در همین راستا ابزاری ارائه شده که سعی در تشخیص این دسته از آسیب پذیری ها دارد :

ابزار SoCRATE (مخفف Smart ContRActs TEsting)، ابزاری است که سعی در تشخیص این دسته از خطاها با استفاده از چندین ربات به عنوان نقش‌های مختلف در قرارداد، دارد. این ابزار از قسمت‌های مختلف تشکیل شده که در ادامه با این ابزار بیشتر آشنا خواهیم شد.

ابتدا مثالی از یک برنامه آسیب‌پذیر ارائه می‌شود. فرض کنید یک قرارداد هوشمند به گونه‌ای نوشته شده باشد که ابتدا فرستنده به یک شخص واسطه اجازه برداشت مقداری مشخص از حساب او می‌دهد. پس از آن، شخص مورد نظر با توجه به شرایطی می‌تواند از حساب فرستنده به حساب بقیه رمزارز واریز کند. این سناریو در تصویر زیر مشاهده می‌شود:



تصویر 1. یک نمونه تراکنش

برای چنین سناریویی برنامه زیر نوشته شده است:

```
contract SampleToken {

  function buggedTransferFrom(
      address _from,
      address _to,
      uint256 _value
  )
    public
    returns (bool)
  {
    require(_value <= allowed[_from][msg.sender]);
    require(msg.sender != _from && _from != _to);

    balances[_from] -= _value;
    balances[_to] += _value;
    allowed[_from][msg.sender] -= _value;

    emit Transfer(_from, _to, _value);
    return true;
  }

  function approve(address spender, uint _value) public returns (bool) {
    allowed[msg.sender][spender] = _value;
    Approval(msg.sender, spender, tokens);
    return true;
  }

  function deposit() public payable {
    /* not supported for brevity */
  }

  mapping(address => uint256) balances;
  mapping(address => mapping (address => uint256)) allowed;

}
```

تصویر 2. کد واریز رمز ارز از حساب فرستنده به حساب بقیه

همانطور که در این برنامه مشاهده می‌شود ، تابع buggedTransferFrom ، آدرس مقصد و مبدا و همچنین میزان واریزی از مبدا برای مقصد را به عنوان ورودی می‌گیرد. در صورتی که میزان وارد شده از میزان قابل دسترسی توسط آدرس مبدا کمتر بوده و شخص واسط برای خود رمزارز ارسال کند میزان خواسته شده از حساب فرستنده کم شده و به حساب گیرنده افزوده می‌شود.

حال فرض کنید شخص فرستنده، در حساب خود دارای 1000 واحد از رمزارز خود می‌باشد. فرستنده به یک شخص واسط اجازه برداشت 2000 واحد از حساب خود می‌دهد. در این شرایط اگر شخص واسط دست به انتقال 2000 واحد به شخص دیگری بزند، هیچ شرطی در برنامه مانع او از انجام این کار نمی‌شود. در این شرایط در خط 14 برنامه یک سرریز رخ می‌دهد. این حالت را در تصویر زیر نیز می‌توان مشاهده کرد :

balance	
spender	1,000
receiver	0

allowed		
spender	initiator	0

$spender \rightarrow approve(initiator, 2000)$

balance	
spender	1,000
receiver	0

allowed		
spender	initiator	2,000

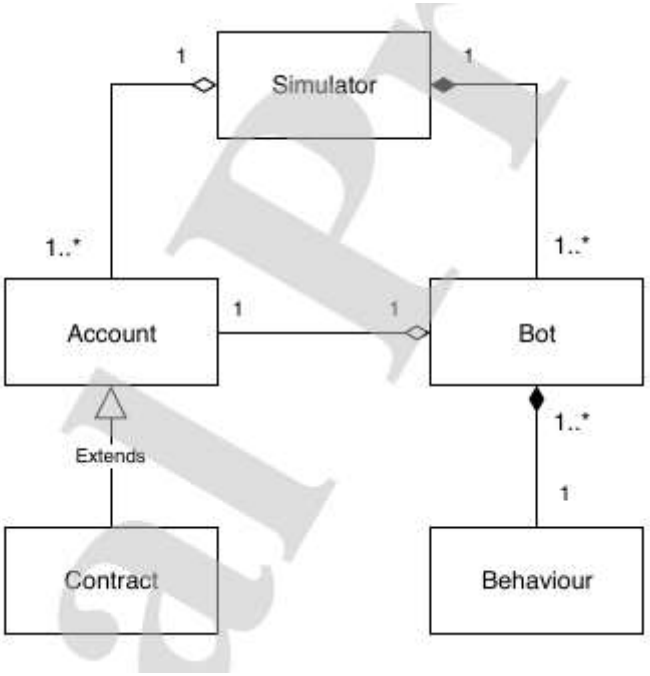
$initiator \rightarrow buggedTransferFrom(spender, receiver, 2000)$

balance	
spender	$2^{256} - 1,000$
receiver	2000

allowed		
spender	initiator	0

تصویر 3. رخداد یک سر ریز

فریمورک SoCRATE برای کشف این دسته از آسیب‌پذیری‌ها به کار می‌رود. در طراحی این فریمورک از یک ساختار شی گرایی استفاده شده است. نمودار روابط کلاس ها در تصویر زیر آمده است :



تصویر 4. روابط کلاس ها در فریمورک SoCRATE

همانطور که در تصویر مشاهده می‌شود، این فریمورک از 5 بخش اصلی تشکیل شده است :

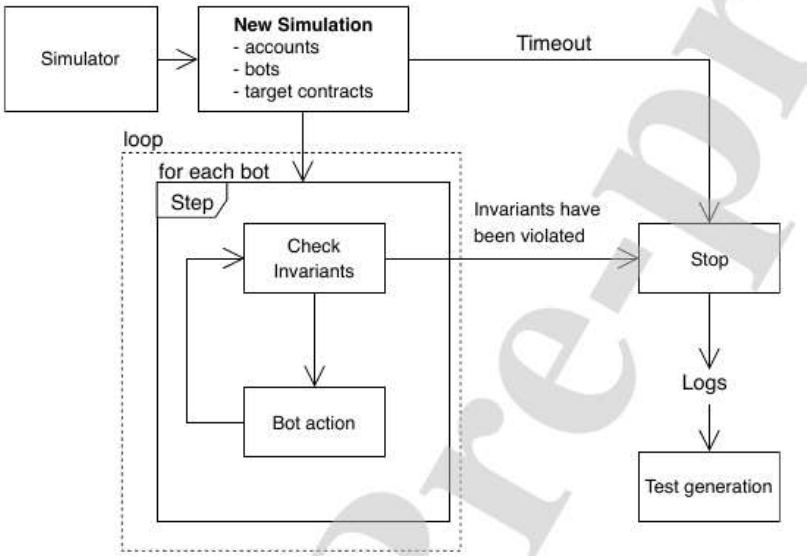
- Account : نمایانگر وضعیت یک کاربری می‌باشد.
- Contract : نوعی Account است که متغییرهای مرتبط با قرارداد هوشمند را در بر دارد.
- Bot : نمایانگر نقش‌هایی است که در قرارداد هوشمند قرار دارد.
- Behaviour : بیانگر نحوه عملکرد Bot در انتخاب مقادیر ورودی قرارداد هوشمند می‌باشد.
- Simulator : بالاترین سطح این فریمورک بوده که اجرای قرارداد هوشمند را شبیه سازی می‌کند.

به طور کلی نحوه کارکردن این برنامه بررسی حالت‌هایی است که نباید طی اجرای یک قرارداد نقض شوند. به عنوان مثال در طول اجرای برنامه نباید سرریز رخ دهد. فریمورک در حالت پیش‌فرض دارای 6 مورد از این حالت‌ها می‌باشد. که لیست این حالت‌ها در تصویر زیر آمده است :

- **I1** is a general invariant. Its condition states that there must not exist a successful transaction whose execution causes an overflow.
- **I2** is an EIP20 specific invariant. Its condition states that the sum of the tokens owned by each account (*balanceOf(a)*) must be equal to the total amount of token supply (*totalSupply*).
- **I3** is an EIP20 specific invariant, expressed in terms and *post* conditions on the function *transferFrom*. For a *transferFrom* to be successful, the amount of token to be transferred (*t.amount*) has to be less than or equal to the amount of tokens that the transaction sender is still allowed to spend on behalf of *from* (*allowance[from][msg.sender]*).
- **I4** is an EIP20 specific invariant, expressed in terms of *post* conditions on the function *transferFrom*. After each successful *transferFrom*, the amount of tokens the transaction sender can spend from the *from* address must be decreased by the amount of token transferred (*t.amount*).
- **I5** is an EIP20 specific invariant, expressed in terms of *post* conditions. Both *transfer* and *transferFrom* transactions must emit, if successful, a *Transfer* event consistent with the actual function parameters and the interface specification.
- **I6** is an EIP20 specific invariant, expressed in terms of *post* conditions on the function *approve*. Similarly to the invariant *I5*, a successful transaction *approve* must emit a consistent *Approval* event.

تصویر 5. لیست حالت ها

همچنین این فریمورک این امکان را در اختیار توسعه دهندگان قرار می‌دهد که بتوانند برخی از این قواعد را غیرفعال و یا قواعد جدیدی به این لیست اضافه کنند. در تصویر زیر شمای کلی نحوه کار فریمورک ارائه شده است :



تصویر 6. شمای کلی نحوه کار فریمورک

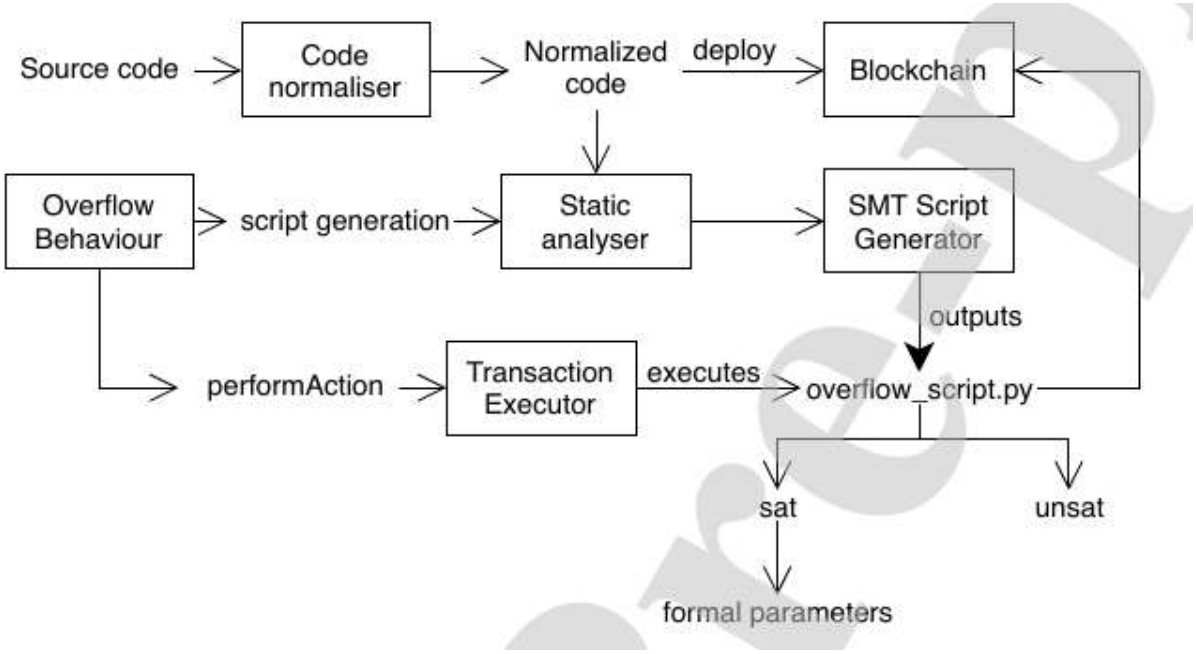
همانگونه که مشاهده می‌شود ابتدا یک شی از کلاس simulator ایجاد شده و عناصر دیگر نیز برای شبیه‌سازی قرارداد آماده می‌شود. پس از آن در یک حلقه یک فعالیتی انجام شده و بررسی می‌شود که در نتیجه انجام آن، یکی از قواعد گفته شده نقض می‌شود یا نه. اگه نقض شد، مورد مربوطه گزارش می‌شود.

نحوه انتخاب ورودی برای اجرای قراردادها با توجه به شی از کلاس behaviour تعیین می‌شود. به طور کلی این فریمورک دارای چند نوع رفتار(behaviour) می‌باشد. این رفتار باید به گونه ای انتخاب شوند که از قسمت‌های require در برنامه گذر کنند(مانند خط 11 و 12 در برنامه نمونه آورده شده). در غیر این صورت وجود چنین تست‌هایی به تشخیص آسیب‌پذیری کمکی نمی‌کند. به همین منظور از برخی روش‌های تخمینی استفاده می‌شود.

برنامه می‌تواند به صورت رندوم رفتار کند. به این صورت که مقادیر ورودی مقادیری تصادفی می‌باشند. به عنوان مثال برای میزان انتقال، یک عدد تصادفی بین 0 تا 1000 واحد بیشتر از موجودی حساب در نظر گرفته می‌شود.

برنامه می‌تواند به عنوان ورودی مقادیر مرزی را انتخاب کند. به عنوان مثال برای یک ورودی با نوع عدد صحیح می‌تواند مقادیری در رنج 0 تا 1000 و همچنین $2^{256} - 1000$ تا 2^{256} انتخاب کند. همچنین برنامه می‌تواند ورودی‌ها را متناسب با شروط برنامه با هدف سرریز انتخاب کند. در این مورد در ادامه بیشتر توضیح داده خواهد شد. همچنین برنامه می‌تواند رفتار خود را به صورت ترکیبی از رفتارهای شرح داده شده انتخاب کند.

برای انتخاب داده‌ها به نحوی که در صورت امکان، سرریز رخ دهد، ابتدا برنامه تحلیل می‌شود. به منظور ساده‌تر شدن تحلیل ابتدا برنامه نرمالایز می‌شود. همچنین این برنامه روی یک بلاکچین داخلی قرار می‌گیرد. برنامه نرمالایز شده، ابتدا تحلیل ایستا شده و با تحلیل نمادین این برنامه قیدهایی از این برنامه که به مسیرهای مختلف منتهی می‌شود استخراج می‌شود. پس از آن این قیدها به یک constraint solver به عنوان ورودی داده می‌شود. فریمورک SoCRATE از ابزار 3Z به عنوان constraint solver استفاده می‌کند. این ابزار به عنوان خروجی مقادیری متناسب با مسیرهای مختلف برنامه تولید می‌کند. این مقادیر در اختیار فریمورک قرار گرفته و سعی می‌کند تا با اجرای این ورودی ها و با بررسی انحراف از قواعد ، آسیب‌پذیری سرریز را کشف کند. این فرایند در تصویر زیر به نمایش گذاشته شده است :



تصویر 7. نحوه کشف آسیب پذیری سر ریز

نویسندگان مقاله برای ارزیابی ابزار خود 1095 مورد از قراردادهای هوشمند مورد استفاده کنونی در زنجیره بلاکچین اتریوم را در شرایط مختلف مورد تحلیل قرار داده‌اند.

در ابتدا با استفاده از 10 بات و رفتار ترکیبی توانسته‌اند تعداد 148 آسیب‌پذیری کشف کنند. نکته جالب در مورد این 148 آسیب‌پذیری آن است که در 94 مورد آسیب‌پذیری مربوط به نقض حالت 2I بوده است. یعنی این برنامه‌ها به گونه‌ای نوشته شده‌اند که پس از انجام تراکنش تعدادی از واحدهای ارز مورد نظر از بین می‌رود یا اضافه می‌شوند. همچنین در این تحلیل 32 مورد false positive وجود داشته است. به طور کلی و با مقایسه با ابزارهای پیشین نتیجه بسیار خوبی محسوب می‌شود. پیش از این فریمورک، بهترین ابزار موجود Echidna بوده که با استفاده از تکنیک fuzzing به برنامه ورودی می‌دهد. همچنین این ابزار امکان تشخیص انحراف از معیارها را ندارد.

برای بررسی تاثیر تعدد بات ها در مقایسه با تحلیل با استفاده از یک بات، آزمایش قبلی این بار تنها با استفاده از یک بات انجام شد. همچنین برای کمتر شدن زمان تحلیل بخشی تنها بخشی از برنامه‌های موجود مورد تحلیل قرار می‌گیرند. زمانی که برنامه با 10 بات کار کرده است، توانسته 95 آسیب‌پذیری به همراه 9 FP کشف کند. در حالی که با 1 بات تنها 90 آسیب‌پذیری به همراه 3 FP کشف می‌شود. به طور کلی مشاهده می‌شود که تاثیر تحلیل برنامه با استفاده از بات‌های بیشتر، به نسبت استفاده از تنها 1 بات بیشتر است.

برای بررسی میزان تاثیر در استفاده از رفتارهای مختلف نیز از برنامه‌های مورد استفاده از آزمایش قبل با 10 ربات استفاده شده است. تعداد آسیب‌پذیری‌های درست(TP) یافت شده با استفاده از روش ترکیبی، رندوم + مقادیر مرزی، سرریز،مقادیر مرزی و رندوم به ترتیب برابر 95، 89، 53، 90 و 82 می‌باشد. جزئیات دقیق‌تر آن شامل این که هر کدام از آسیب‌پذیری ها موجب نقض کدام قانون شدند در تصویر زیر آمده است :

	I1			I2			I3			I4			I5			I6			TOTAL		
	FP	TP	Uni	FP	TP	Uni	FP	TP	Uni	FP	TP	Uni	FP	TP	Uni	FP	TP	Uni	FP	TP	Uni
Complete	0	**10**	**1**	2	61	**3**	2	1	0	3	1	0	1	2	0	1	20	**2**	9	**95**	6
Bound.+Rand.	0	0	0	2	63	2	3	**2**	0	11	**2**	0	1	2	0	1	20	1	18	89	**3**
Overflow	0	9	0	**0**	44	2	**0**	0	0	**0**	0	0	**0**	0	0	**0**	0	0	**0**	53	2
Boundary	0	0	0	2	**64**	2	3	**2**	0	13	2	0	1	**2**	0	1	20	0	20	90	2
Random	0	0	0	**0**	61	2	**0**	0	0	2	0	0	**0**	1	0	1	20	0	3	82	2

تصویر 8. جزئیات دقیق از نقص قوانین توسط آسیب پذیری ها

بنابراین می‌توان نتیجه گرفت بهترین حالت، حالت کامل است که ترکیبی از روش‌های مختلف می‌باشد.

همچنین قواعد استفاده شده با استفاده از یک نوع خاص از استاندارد برنامه نویسی برای قراردادهای هوشمند استخراج شده است. همانطور که پیشتر نیز اشاره شده، می‌توان برخی از این معیارها را غیرفعال کرد و یا معیار جدیدی افزود. به همین منظور آزمایش‌هایی ترتیب داده شده تا از اثربخشی این موضوع اطمینان حاصل شود. با افزودن برخی معیار جدید مشاهده شده که ابزار توانسته متناسب با معیارها، ورودی‌هایی ایجاد کند که معیارها را نقض کند. بنابراین این ابزار در تولید مقادیر ورودی برای قراردادهای هوشمند که در شرایط خاص استفاده می‌شوند نیز موثر است.

محدودیت های قرارداد های هوشمند

1. Irreversible Bugs : به دلیل ماهیت غیرقابل برگشت بلاک چین، پس از ثبت قراردادهای هوشمند، نهایی شده و غیر قایل تغییر می شوند . درنتیجه اگر یک اشکال در قرارداد هوشمند وجود داشته باشد، هیچ راه مستقیمی برای رفع آن وجود ندارد و باید آن را به روز کرد و هنگامی که یک نسخه جدید از یک قرارداد موجود ثبت می شود داده های ذخیره شده در قرارداد قبلی به طور خودکار منتقل نمی شوند و باید به صورت دستی قرارداد جدید را با داده های گذشته مقداردهی اولیه کرد که این کار دشواری است.
2. Performance Issues : در سیستم های بلاک چین فعلی، قراردادهای هوشمند به صورت سریالی توسط ماینرها و اعتبار سنجی ها اجرا می شوند اما متاسفانه اجرای سریال توان عملیاتی سیستم را محدود می‌کند و از معماری‌های چند هسته‌ای و خوشه‌ای امروزی استفاده نمی کند.
3. Lack of Trusted Data Feeds (Oracles) : اجرای قرارداد هوشمند نیازمند داده های خارجی در مورد وضعیت ها و رویدادهای دنیای واقعی خارج از بلاک چین است. (Oracles به عنوان پل بین بلاک چین و دنیای

خارجی عمل می کند و داده ها را از دنیای واقعی گرفته و به زنجیره بلوکی سوق می دهند) . در نتیجه فقدان یک منبع داده قابل اعتماد یک مانع برای تکامل قراردادهای هوشمند در نظر گرفته می شود.

4. Lack of Standards and Regulations : یکی از مسائل و خطرات اصلی امنیت بلاک چین، فقدان استانداردها و مقررات است . درز اطلاعات محرمانه، سرقت کلیدهای رمزنگاری، و جنایات مختلف در دنیای واقعی (قتل، آتش‌سوزی، و تروریسم) نمونه هایی از رفتارهای مخرب در قراردادهای هوشمند هستند و هنگام وقوع این رفتارها در قراردادهای هوشمند به دلیل فقدان استانداردها و مقررات لازم ، نظارت بر این اعمال مخرب دشوار می شود.

نمونه هایی از کاربردهای قرارداد های هوشمند

- Finance :

1. تامین امنیت شامل فرآیندهای پیچیده‌ای است که زمان‌بر، ناکارآمد، دست و پا گیر هستند .قراردادهای هوشمند می توانند واسطه ها را در زنجیره نگهداری اوراق بهادار دور بزنند و پرداخت خودکار سود سهام، تقسیم سهام و مدیریت بدهی را تسهیل کنند و در عین حال ریسک های عملیاتی را کاهش دهند .علاوه بر این، قراردادهای هوشمند می توانند تسویه و تسویه اوراق بهادار را تسهیل کنند.
2. صنعت بیمه سالانه ده ها میلیون دلار برای رسیدگی به مطالبات خرج می کند و میلیون ها دلار به خاطر مطالبات متقلبانه ضرر می کند. قراردادهای هوشمند را می توان برای خودکارسازی و افزایش سرعت پردازش، تأیید و پرداخت و همچنین حذف تقلب و جلوگیری از مشکلات احتمالی استفاده کرد.

- Management :

1. در حال حاضر، اکثر سازمان ها توسط هیئت مدیره ای مدیریت می شوند که قدرت تصمیم گیری را در اختیار دارند .اما بهتر است مدیریت سازمانی یکنواخت و غیرمتمرکز باشد. قراردادهای هوشمند می توانند واسطه های غیرضروری را که موجب محدودیت ها و مقررات پیچیده می شوند را حذف کنند.
2. قراردادهای هوشمند می توانند کارایی و اختیارات دولت الکترونیک را بهبود بخشند و در بهبود کیفیت خدمات دولتی، توسعه سیستم اعتبار فردی، تقویت اعتبار دولت و ارتقای یکپارچگی و سیستم های پرداخت جدید برای کار و مستمری، تقویت سیستم های کمک بین المللی، رای گیری الکترونیکی تاثیرگذار باشند.

- Energy : در صنعت انرژی تمام توجهات در آینده بر روی انرژی پاک و توزیع شده است. می توان از فناوری بلاک چین و قرارداد های هوشمند برای ساخت سیستم انرژی توزیع شده و تامین انرژی استفاده کرد. درواقع بازارهای تجارت انرژی غیرمتمرکز را ایجاد کرد، کارایی مصرف انرژی را بهبود بخشید و هزینه های عملیاتی شبکه را کاهش داد.

نحوه اجرای یک قرارداد هوشمند

در ارزهای دیجیتال تراکنش ها به صورت یک طرفه از فرستنده به گیرنده صادر میشود و برای دریافت کننده ضروری نیست که رسید دریافتی را تایید کند ، ولی برای قرارداد ها این یک طرفه بودن کارآمد نیست و تایید هر دو طرف لازم است.

روش پیشنهادی برای بایگانی اسناد قراردادی : در این پروتکل ،از تراکنش به عنوان شاهدی برای موافقت پیمانکار استفاده میشود. گیرنده معامله ، با توجه به تراکنشی که دریافت کرده ، یک تراکنش جدید ایجاد می کند که نشان دهنده رضایت او از تراکنش دریافتی است. زنجیره ای از تراکنش ها توسط پیمانکاران ایجاد میشود و اخرین داده تراکنش به پیمانکار اولی که

تراکنش را اغاز کرده بود برمی گردد و سپس اولین پیمانکار تراکنش خود را تایید می کند و تراکنش جدیدی با فرد mediator دیگر ایجاد میکند.

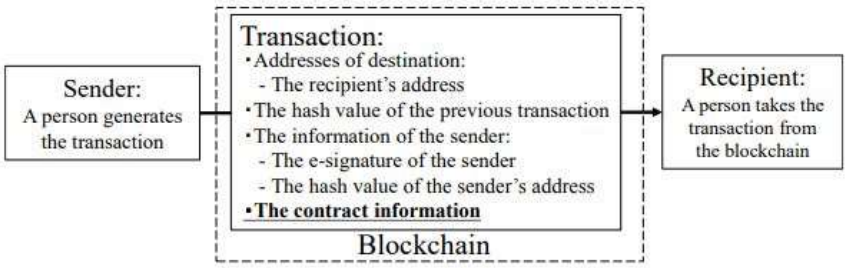


Fig. 1. The transaction of our proposed protocol

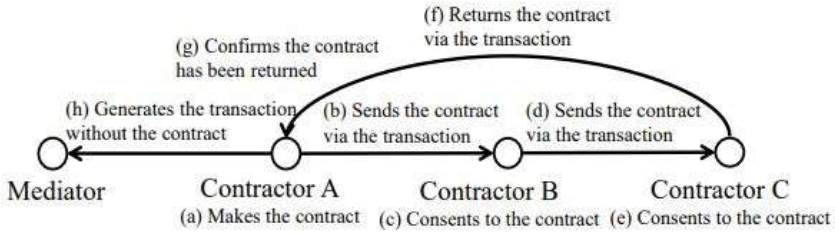


Fig. 2. Making of the transaction chain by the contractors

تصویر 9. روند انجام تراکنش

همچنین داده های قرارداد به منظور ذخیره اسناد ، باید رمزگذاری شده باشند زیرا بلاک چین عمومی است و هر کس میتواند ان را ببیند.

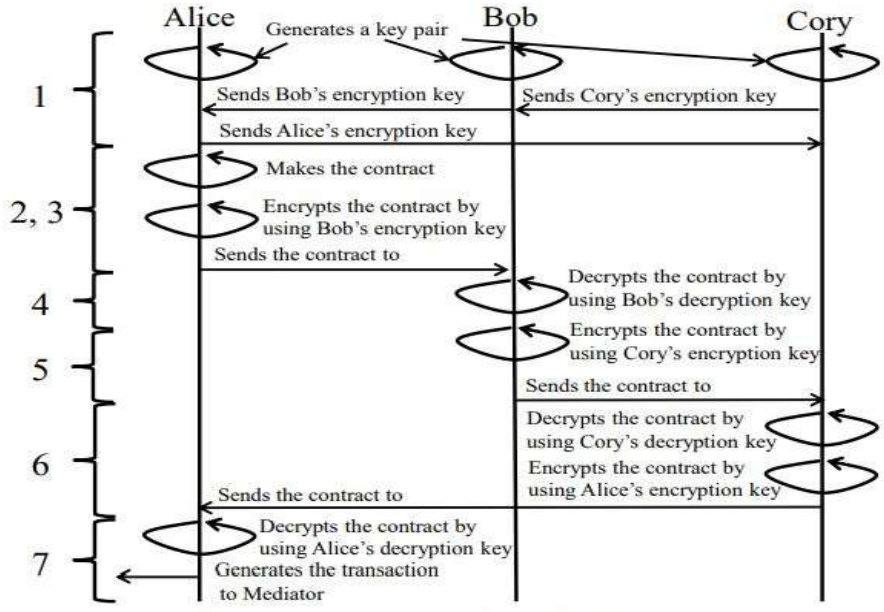


Fig. 3. The processing flow of the example
(Numbers at left are associated with numbers in the main text)

تصویر 10. مثال یک فرآیند

مثال در شکل : Alice و Bob و Cory میخواهند با همدیگر قراردادی ببندند. A قراداد را میبندد و به دنبال آن B و C آن را تایید میکنند. روند کار به این صورت است :

1. هر یک از آنها key pair ای برای رمزگذاری در حوزه خود ایجاد میکنند و یک public key برای رمزگذاری به فرستنده تراکنش ارسال میکند
2. A قرارداد را میبندد و با استفاده از کلید رمزگذاری شده که از طرف B فرستاده شده ان را رمزگذاری میکند . سپس داده های تراکنش که شامل داده های قرارداد رمزگذاری شده بود را تولید میکند و ان را به هر عضوی از شبکه پخش و broadcast میکند.
3. وقتی ماینری موفق به تولید بلاکی میشود به این معنا هست که تراکنش A از طریق آن در بلاک چین ثبت میشود.
4. از انجایی که بلاک چین با شبکه هماهنگ شده B از بلاک چین داده های تراکنش A که به B ادرس دهی شده را میگیرد و سپس چون رمزگذاری شده انها را دیکد میکند به وسیله decryption key .
5. B قرارداد را بررسی میکند و اگر رضایت از ان دارد یک تراکنش که به تراکنش A اشاره دارد تولید میکند که به C آدرس دهی شده. B فرم قرارداد که از A گرفته را با استفاده از کلید رمزگذاری C ، رمزگذاری میکند و سپس ان را پخش میکند.
6. C هم به روش مشابه قرارداد را بررسی میکند و سپس قرارداد را به واسطه تراکنش به A میفرستد. این یعنی معامله از اخرین پیمانکار به اولین پیمانکار در زنجیره باز گردانده میشود.
7. در نهایت A تراکنش C را دریافت میکند و تایید میکند که ایا قرارداد رمزگذاری شده ای که دریافت کرده صحیح است یا نه. اگر ان را تایید کند تراکنشی تولید میکند که به تراکنش c اشاره دارد که به mediator یا میانجی گر ادرس دهی شده که درخواست دنباله ای از تایید ها را داشته است.

اجرای قرارداد های هوشمند از طریق برنامه های غیر متمرکز و ماشین مجازی اتریوم

برنامه های غیر متمرکز (Dapp): برنامههای غیرمتمرکز همانطور که از اسمشان مشخص است برنامههایی هستند که به صورت غیرمتمرکز فعالیت میکنند یعنی شخص ثالث و متمرکزی آنها را کنترل نمیکند. برای اینکه بتوان برنامه غیرمتمرکزی ایجاد کرد حتما باید این برنامهها را بر روی بلاک چین اجرا کرد و بلاکچین اتریوم اولین مرجع برای ایجاد برنامههای غیرمتمرکز است. پس برنامه های غیر متمرکز، برنامههای دیجیتالی هستند که در شبکههای بلاک چین همتا به همتا (P2P) و توزیع شده اجرا میشوند و تحت اختیار واسطه و هیچ شخصی قرار ندارند ، که یک ابزار ساده و آسان برای کاربر می باشد و کاربران غیر فنی هم به راحتی میتوانند با آن سازگار شوند.

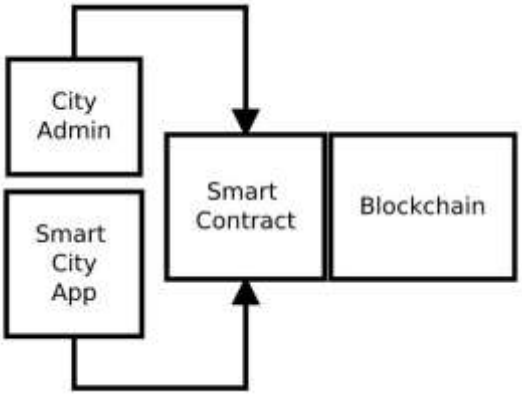
ماشین مجازی اتریوم : یک ماشین مجازی، کامپیوتری است که روی بلاکچین اجرا میشود و به چندین اسمارت کانترکت اجازه میدهد که با یکدیگر تعامل داشته باشند. به بیانی ساده، قرارداد هوشمند یک توافقنامه است که توسط کدها نوشته شده و زبان برنامهنویسی آن نیز مطابق شبکه این قرارداد است. برای این که بتوانیم Smart Contractها را روی بلاکچینهای مختلف اجرا کنیم، ماشینهای مجازی باید قراردادهایی که با یک زبان برنامهنویسی واحد نوشته شدهاند را اجرا کنند. فرض کنید که یک توسعهدهنده Dapp هستید و یک برنامه روی یک بلاک چین عمومی راهاندازی کردهاید و اکنون میخواهید Dapp خود را به دیگر شبکهها نیز گسترش دهید تا فرصتهای بیشتری برای جذب کاربران بیابید؛ اینجاست که ماشینهای مجازی وارد میشوند .ها به توسعهدهندگان کمک میکنند تا بهسرعت برنامههای غیرمتمرکز خود را روی شبکههای مختلف توسعه دهند. VM

پیوند قرارداد های هوشمند با شهر هوشمند

گوشی های هوشمند به یک کالای ضروری برای مردم تبدیل شده اند و در طول روز با مردم حمل می شوند. آنها اغلب با حسگرهای مختلفی مانند فشارسنج، میکروفون همراه با GPS تعبیه شده اند. برنامه‌های کاربردی برای حسگر گوشی‌های هوشمند (Crowdsourced) توسعه داده شده‌اند که باگ ها را شناسایی کنند و برای بهبود آن تلاش کنند. در یک مقاله بیان شده که تقریباً ¼ کاربران پس از یک بار استفاده ، برنامه را رها می کنند و کاربرانی که برنامه را بیش از یازده بار باز می کنند در محدوده 30% تا 40% قرار دارند اما برنامه‌های شهر هوشمند بر رویکرد حفظ کاربران به وسیله وفاداری شهروندان و محتوای فریبنده مانند تیترهای خبری تکیه کرده است. برنامه های شهر هوشمند با توجه به ارزش اقتصادی هر برنامه به کاربران پیشنهاد می شود که این برنامه ها با حداقل تغییر نسبت به وضعیت فعلی و حداقل هزینه اضافی عملیاتی برای شهر ، اجرا می شوند. نتایج اولیه یک مطالعه اخیر نشان می‌دهد که استفاده از برنامه‌ای که بر رفتارهای سلامتی تاثیرگذار بوده است توسط کاربران مورد استقبال زیادی قرار گرفته است.

نحوه کار قرارداد های هوشمند در شهر هوشمند

قراردادهای هوشمند برنامه‌هایی هستند که توسط گره‌های روی بلاک چین در ماشین مجازی اتریوم اجرا می‌شوند. اتر واحد پولی است که به عنوان جبران نودها برای ذخیره سازی و پردازش قراردادهای هوشمند و همچنین انتقال ارزش بین کاربران عمل می کند. بلوک های اتریوم دارای اندازه متغیر هستند و در بازه های زمانی 15 ثانیه اضافه می شوند و از 15 تراکنش در ثانیه پشتیبانی می کنند. یک برنامه کامل که شامل یک قرارداد هوشمند است، اغلب به عنوان برنامه غیرمتمرکز یا (dapp) شناخته می شود.



تصویر 11. ورودی های Smart contract

قرارداد های هوشمند شامل city Admin، city App می باشد که کار Admin ذخیره سازی برنامه و داده های مربوط به کاربران در بلاک چین و پرداخت ارز دیجیتال به کاربران نهایی برنامه می باشد.

یک رابط در پلتفرم پرداخت، برای ادغام شهر هوشمند به بخش Admin شهر اضافه شده است. اولین قدم وارد کردن مبلغ پرداخت و مبنای پرداخت است. Admin شهر در مورد پارامترهای پرداخت برای برنامه شهر هوشمند تصمیم می گیرد. ظاهراً میزان پرداخت و مبنای پرداخت براساس بازده سرمایه گذاری احتمالی در برنامه و معیارها خواهد بود. روش های پرداخت مورد انتظار و آزمایش‌شده استفاده از برنامه براساس رویداد، زمان، و مقدار مستقیم یا غیرمستقیم داده خواهد بود.

برای مثال، ROI برنامه‌ای است که رانندگان را رصد می‌کند و مکان هایی که رانندگان ترمز شدید گرفته اند را نقشه‌برداری می‌کند پس به ایستگاه های پلیس کمتری نیاز است . در این مورد یک مبنای پرداخت بر اساس زمان ، ممکن است مناسب باشد.

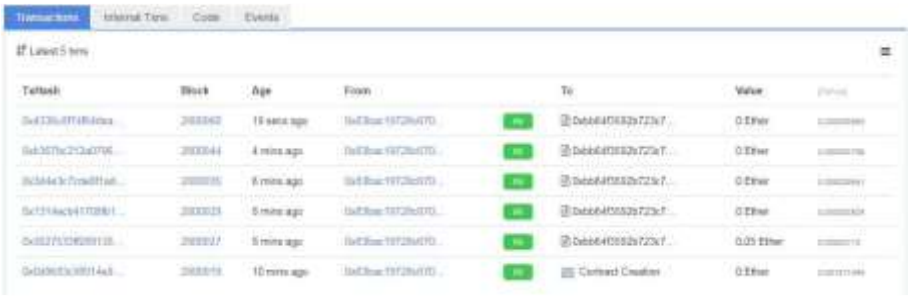
پس از اینکه مبلغ پرداخت و مبنای پرداخت در قرارداد هوشمند وارد شد، شناسه‌های برنامه ایجاد می‌شوند، به طوری که می‌توان اپلیکیشن را به طور منحصر به فرد توسط قرارداد هوشمند شناسایی کرد. شناسه‌های برنامه در قرارداد هوشمند ذخیره می‌شوند و در اختیار مدیر شهر قرار می‌گیرند تا در اختیار توسعه دهنده برنامه شهر هوشمند قرار دهد.

مدیر شهر معیارهای استفاده و شناسه های برنامه را به توسعه دهنده برنامه انتقال می دهد تا برنامه برای پرداخت به روز شود. به عنوان مثال، در جایی که مبنای پرداخت یک رویداد است، برنامه شهر هوشمند باید اطلاعات وقوع رویداد را به قرارداد هوشمند ارسال کند. در مواردی که مبنای پرداخت یک مبنای زمانی است، برنامه شهر هوشمند باید بر استفاده و ارائه اطلاعات دوره ای نظارت داشته باشد. در مواردی که مبنای پرداخت مبتنی بر داده است، برنامه شهر هوشمند باید داده ها را به قرارداد هوشمند ارسال کند.

در هنگام استفاده، برنامه به‌روزرسانی می‌شود تا آدرس کیف پول اتریوم کاربر را دریافت کند و آن آدرس را به قرارداد هوشمند ارسال کند. هنگامی که برنامه به روز شده اجرا می شود، با قرارداد هوشمند تعامل می کند و اطلاعات کاربران را ارائه می دهد، جایی که قرارداد هوشمند این اطلاعات را ارزیابی می کند و زمانی که معیارهای استفاده برآورده شد و قرارداد هوشمند تامین مالی شد، اتر را انتقال می دهد.

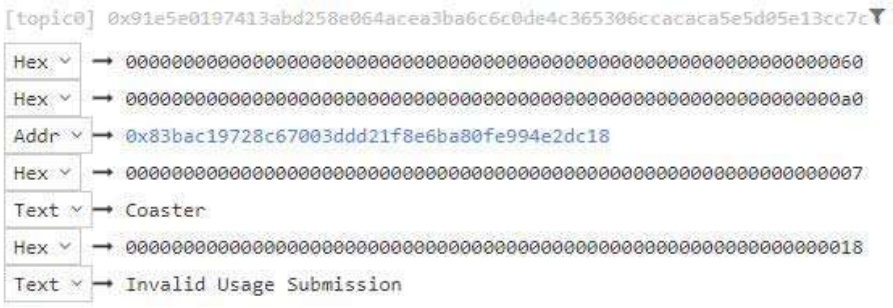
یک قرارداد هوشمند ابتدایی از عملکردهای اساسی همچون ورودی برای دریافت وجوه، افزودن کاربران واجد شرایط، تنظیم نام/توضیحات/کلید برنامه، مبنای پرداخت و رویدادهای استفاده از برنامه پشتیبانی میکند.

شکل اول زیر مجموعه ای از اقدامات در قرارداد هوشمند را نشان می دهد که مرتبط ترین آنها ایجاد قرارداد هوشمند، تأمین مالی آن، افزودن کاربران مجاز و پرداخت وجه به کاربر برای یک رویداد استفاده است.



تصویر 12. اقدامات موجود در قرارداد هوشمند

شکل دوم تلاش برای ارسال نامعتبر کاربر را نشان می دهد:



تصویر 13. تلاش برای ارسال نا معتبر کاربر

شکل سوم رکورد یک رویداد استفاده در بلاک چین شبکه آزمایشی را نشان می‌دهد که اطلاعات استفاده از جمله مبنای پرداخت، آدرس کیف پول کاربران و مبلغ انتقال‌یافته است.

تصویر 14. رکورد رویداد استفاده از بلاک چین

بر اساس 62 بایت بودن بلاک کاربر، نظارت کاربر بر روی ارسال محموله ها با اندازه مشابه مشکل تر می شود و نشان می دهد یکی از مشکلات مقیاس پذیری می تواند باشد.

پیوند بلاک چین با IoT

استفاده از فناوری Blockchain در حوزه های بسیاری از جمله IoT کاربرد دارد. IoT بیانگر ارتباط میان دستگاه ها و اینترنت از طریق نرم افزار و سنسورها به منظور برقراری ارتباط، جمع آوری و تبادل اطلاعات دستگاه ها با یکدیگر است .

عدم استفاده اغلب دستگاه های IoT از استانداردهای احراز هویت موجب شده که مهاجمان بتوانند به صورت گسترده به آن دستگاه ها نفوذ کنند، بنابراین احراز هویت و استاندارد سازی در تمامی بخشهای IoT ضروری است .

از طرف دیگر شهر هوشمند یکی از برنامه های اصلی IoT است . شهرهای هوشمند از IoT برای جمع آوری و تحلیل و تجزیه داده ها برای بهبود زیرساخت ها استفاده میکنند. یکی از چالش های مهم آن ، ارائه راه حل کنترل دسترسی قابل اطمینان به داده های اینترنت اشیا در بین کاربران مختلف هست. یک شبکه بزرگ اینترنت اشیا در داخل یک شهر هوشمند می تواند تعداد زیادی از دستگاه ها را به هم متصل کند. با این وجود اتصال دستگاه های ناهمگن بزرگ در شبکه IoT میتواند نگرانی های امنیتی به وجود آورد.

بسیاری از محققان برای چالش های موجود در شهر هوشمند مبتنی بر IoT ، بلاک چین را پیشنهاد کرده اند و یک مکانیسم کنترل دسترسی ابداع کرده اند تا بتوان داده ها را در شهر هوشمند با حفظ محرمانگی و حریم خصوصی به اشتراک گذاشت.

انواع کنترل دسترسی در IoT برای افزایش امنیت

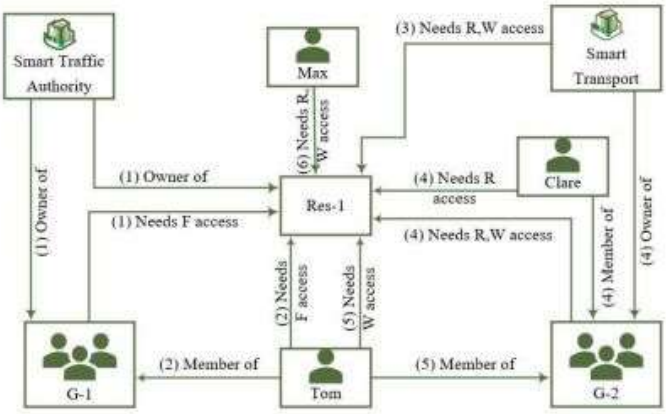
تکنولوژی اینترنت اشیا (IoT) به طور گسترده‌ای به عنوان بخش مهمی از شهرهای هوشمند شناخته شده‌است. با این حال، دستگاه‌های IoT هوشمند متصل با پلتفرم‌های ناهمگن نیز چالش‌های جدیدی را از نظر حریم خصوصی و امنیت به ارمغان می‌آورند. کنترل دسترسی (AC) یکی از نگرانی‌های امنیتی بالا است، که در به اشتراک گذاری منابع و حفاظت از اطلاعات در دستگاه‌های IoT حیاتی است. رویکردهای سنتیAC ، مانند فهرست‌های کنترل دسترسی (ACL)، کنترل دسترسی مبتنی بر نقش (RBAC)، و کنترل دسترسی مبتنی بر ویژگی (ABAC)، قادر به ارائه یک مکانیزم مقیاس پذیر، قابل مدیریت، و کارآمد برای برآورده کردن الزامات سیستم‌های IoT نیستند. ضعف دیگر در AC امروز، سرور اعتبارسنجی متمرکز است، که می‌تواند باعث تنها نقطه شکست شود. به لطف مکانیزم‌های رمزنگاری ، راه حلی امیدوارکننده برای پیاده‌سازی مدل‌های AC انعطاف‌پذیرتر فراهم می شود.

RBAC : کنترل‌دسترسی مبتنی بر نقش هست که سیاست دسترسی است که از طریق سیستم تعیین می‌شود نه صاحب. از RBAC در کاربردهای تجاری استفاده می‌شود و نیز در سیستم‌های نظامی جایی که نیازهای حفاظتی چند سطحی ممکن است وجود داشته باشد. RBAC مجموعه‌ای از اجازه‌ها را کنترل می‌کند که ممکن است شامل عملیات پیچیده باشند مثل یک تراکنش تجارت الکترونیکی، و یا می‌تواند به سادگی تنها خواندن و نوشتن باشد. در RBAC یک نقش را می‌توان به صورت مجموعه‌ای از اجازه‌ها نشان داد.

ABAC : کنترل‌دسترسی مبتنی بر مشخصه هست که دسترسی براساس حقی که object مرتبط با کاربر پس از تعیین اعتبار دریافت می‌کند اعطا نمی‌شود بلکه براساس مشخصات کاربر انجام می‌گیرد.کاربر باید ادعاهای خود را در مورد مشخصاتش برای موتور کنترل‌دسترسی ثابت کند. یک سیاست کنترل‌دسترسی مبتنی بر مشخصه تعیین می‌کند کدامیک از ادعاها باید تایید شوند تا دسترسی به یک subject اعطا شود.

CapBAC (capability-based access control) : در این مدل ، کاربر با لیستی از قابلیت ها (capabilities) و حق دسترسی ها در ارتباط هست که این حق دسترسی ها برای دسترسی به منابع (اشیا) مورد نیاز میباشد. حق دسترسی توسط توکنی که غیر قابل جعل و قابل انتقال هست تامین میشود. که این توکن شامل فهرستی از حقوق دسترسی و شرایط دسترسی مرتبط با آنها و محدودیت های امنیتی میباشد.

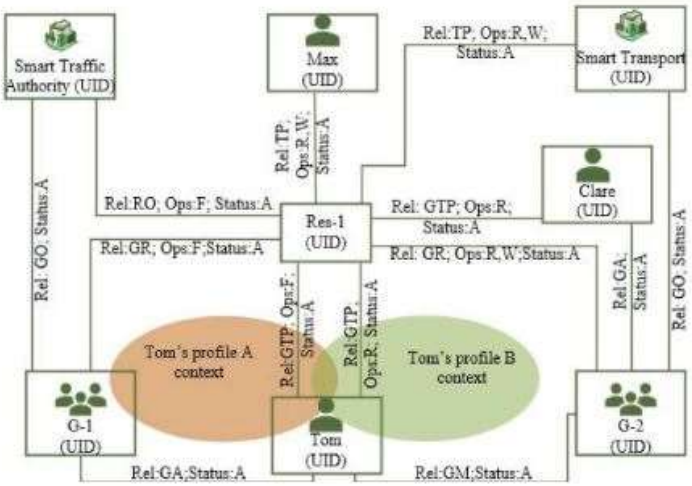
نمونه ای از سناریو کنترل دسترسی پیچیده در یک شهر هوشمند



تصویر 15. نمونه ای از سناریو کنترل دسترسی پیچیده

1. smart Traffic Authority Organization یک منبع IoT را مستقر کرده است. آنها مالک این منبع و گروه 1 (G-1) هستند.
2. فردی به نام TOM عضو G-1 هست. و نیاز به دسترسی کامل به دیتای res-1 دارد.
3. Smart Traffic Authority با ارایه دسترسی خواندن و نوشتن به Smart Transport Organization موافقت کرده است.
4. smart transport باید حق دسترسی خواندن و نوشتن خود را به گروه G-2 واگذار کند. اما علاوه بر آن میخواهد دسترسی به اعضای خود یعنی Clare که R-access دارد و Tom که w-access دارد را داشته باشد.
5. Tom یک کارگر پاره وقت معمولی برای هر دوی Smart Traffic Authority و Smart Transport organization میباشد.
6. Smart Traffic Authority توافق کرده است که دسترسی R و W را فرد سوم شخصی که در اینجا Max هست بدهد که یکسری خدمات را ارایه میدهد.

استفاده از بلاک چین در سناریو اینترنت اشیا



تصویر 16. استفاده از بلاک چین در سناریو IoT

که توجه شود معنای برخی کلمات کلیدی که در سناریو بالا به کار برده شده به صورت زیر است :

Acronym	Description
Org	An organization entity or domain. It is the registered legal business entity that runs the organization.
Ind	An individual entity.
Res	An IoT Resource.
RO	Resource Owner. It could be an Org or Ind entity.
G	A Group (division) within an Org.
GO	Group Owner. It could be either RO Org or TP Org.
P	A party. It is an entity representing an Org, Ind, G, or U.
GM	A Group Member. GM represents a relationship between an Ind and G.
GA	A Group Admin. GA also represents a relationship between an Ind and G. (GA is a GM with admin privilege).
U	A User. It is an Ind.
TP	Third Party. An external Org or external Ind.
TPGO	Third-Party Group Owner.
TPGU	Third-Party Group User.
GTP	Group Third Party. A party within an Org G.
GR	Group Resource. It is an RS made available for a G.
Rel	Relationship between a P and RS or between Ps (parties). In the proposed model, Rel can be labelled as RO, GO, GR, TP, GTP, GM, or GA.
Ops	A set of Operations (e.g., Read (R), Write (W), F (Full)).
Cs	A set of constraints (e.g., Status (Active(A)/Inactive(I))).

تصویر 17. برخی کلمات کلیدی سناریو تصویر 16

معماری کنترل دسترسی فعال شده با بلاک چین

امنیت نگرانی اصلی در مورد اینترنت اشیا است و همین مشکل است که مانع از پذیرش عام این فناوری می‌شود. ابزارهای IoT آسیب پذیری هایی دارند که باعث تسهیل حملات DDos می‌شود ؛ ابزارهای ناایمن این فناوری به راحتی مورد بهره برداری مجرمان فضای مجازی قرار می‌گیرند و از آنها برای راه‌اندازی حملات DDos استفاده می‌شود.

مشکل دیگری که در مقابل فناوری اینترنت اشیا قرار دارد، مقیاس پذیری است. همچنان که تعداد ابزارهای متصل به شبکه اینترنت اشیا افزایش پیدا می‌کند، تایید هویت و متصل کردن نودهای بلاک چین مختلف به یکدیگر در این شبکه، تبدیل به بحران می‌شود. این موضوع نیاز به سرمایه‌گذاری‌های عظیم در سرورهایی را ایجاد می‌کند که مقادیر عظیمی از مبادله اطلاعات را مدیریت می‌کنند. در این حالت، از دسترس خارج شدن سرور باعث خرابی کل شبکه می‌شود.

کاربرد بلاک چین در اینترنت اشیا فرایندهای موجود در صنایع مختلف از جمله تولید، تجارت، حمل و نقل، بخش مالی و حوزه سلامت را دگرگون خواهد کرد. به رغم این پیشرفت‌ها، همچنان امنیت بیشتر برای اکوسیستم IoT موضوع بسیار مهمی است؛ زیرا این فناوری شامل چندین دستگاه، مقدار زیادی اطلاعات، شرکای زنجیره تامین و جامعه است که حفظ امنیت آن کار آسانی نیست.

با افزایش دستگاه‌های IoT که اغلب از استانداردهای احراز هویت برای حفظ امنیت اطلاعات کاربر استفاده نمی‌کنند، این اجازه را می‌دهد که هکرها بتوانند به صورت گسترده‌تر به دستگاه‌های اینترنت اشیا نفوذ کنند؛ در این صورت زیرساخت‌های اصلی آسیب خواهند دید. برای اطمینان و امنیت بیشتر، احراز هویت و استانداردسازی در تمامی بخش‌های Internet of Things بسیار ضروری است.

چند راه وجود دارد که استفاده از بلاک چین در اینترنت اشیا می‌تواند به حل بسیاری از این چالش‌های امنیتی کمک کند :

- بلاک چین می‌تواند برای ردیابی اندازه گیری داده‌های سنسور و جلوگیری از تکرار هر گونه اطلاعات مخرب مورد استفاده قرار گیرد.
- استقرار دستگاه‌های IoT می تواند پیچیده باشد؛ اما یک دفتر کل توزیع شده ای برای شناسایی دستگاه‌های اینترنت اشیا ، تأیید هویت و انتقال داده‌های یکپارچه امن، می‌تواند گزینه مناسبی باشد.
- سنسورهای IoT به جای دریافت تاییدیه از طرف شخص ثالث، برای ایجاد اعتماد می‌توانند داده‌ها را در یک زنجیره بلوک مبادله کنند.
- یک دفتر کل توزیع شده، منابع مخرب و ناقص را در اکوسیستم از بین می‌برد و داده‌های دستگاه‌های IoT را از دستکاری محافظت می‌کند.
- قرارداد هوشمند، هویت فردی و یکپارچگی داده‌ها را فراهم می‌کند و از مشکلات و ناکارآمدی‌های تکنیکی پشتیبانی می‌کند.
- هزینه‌های نصب و راه اندازی اینترنت اشیا را می‌توان از طریق بلاکچین کاهش داد؛ زیرا هیچ واسطه‌ای وجود ندارد.
- دستگاه‌های IoT توسط بلاک چین مستقیما آدرس‌دهی می‌شوند و در اختیار شما قرار می‌گیرند و تاریخچه دستگاه‌های متصل را جهت عیب یابی مشکلات ارائه می‌دهند.

مزایای تلفیق اینترنت اشیا و بلاک چین

1. یکی از مزایای استفاده از بلاک چین در حوزه‌های مختلف این است که با تکیه بر بلاک چین، ما درواقع به سابقه‌ای از هر تراکنش دست پیدا می‌کنیم که غیرقابل تغییر است. درواقع این حساب‌های تراکنش‌ها، درصورتی‌که در دفتر کل توزیع‌شده ثبت شده باشند، توسط هیچ‌کس قابل‌تغییر نخواهند بود و همین موضوع است که باعث ایجاد قابلیت ضد هک شدن آن می‌شود.
2. علاوه بر آن، اینترنت اشیاء بایستی تا در مسیر ادغام خود با بلاک چین، روشی بسیار ساده‌تر را انتخاب کرده تا مورداستفاده همگان قرار گیرد. به‌عنوان‌مثال، یکی از نمونه‌هایی که سیستم‌های سنتی به طرز شگفت‌انگیزی در آن بد بودند، عدم استفاده از اسناد به روش‌های نامناسب بود. این در حالی است که هم‌اکنون، همه افراد به‌سادگی و به شکلی کاملاً ایمن به‌تمامی اسناد موردنیاز خود دسترسی داشته و قادر به تأیید و امضای آنها هستند.

3. همچنین کاربری فراوانی بین صنایع مختلف و افزایش سرعت فعالیت ها

تاثیر بلاک چین بر روی اینترنت اشیا

به‌طور کلی، بلاک چین پایگاه داده‌ای است که مجموعه‌ای از گزارش‌های مرتبط با داده‌هایی که به صورت مداوم در حال تغییر هستند را در خود ذخیره می‌کند. این مفهوم نشان دهنده این است که اساساً هیچ رایانه مرکزی کنترل کننده‌ای وجود ندارد که وظیفه نگهداری و نظارت بر کل زنجیره بلاکی را بر عهده داشته باشد. بلکه گره‌های مختلفی وجود دارند که بر اساس یک مدل مشارکتی، چرخ دهنده‌های تمامی بلاک چین را به حرکت در می‌آورند. درواقع به همین منوال، این زنجیره بلاکی همواره در حال رشد بوده و سوابق اطلاعاتی جدید نیز به‌سادگی به زنجیره اضافه می‌شوند. فناوری بلاک چین راه‌حلی را ارائه می‌دهد که طی آن تراکنش‌ها از طریق سیستمی بسیار ایمن، ساده، بدون قطعی، قابل‌تأیید و کاربردی، ثبت و تأیید شوند. به‌این‌ترتیب، امکان پیچیده کردن صنایع و تسهیل قوانین تجاری جدید را فراهم می‌سازد.

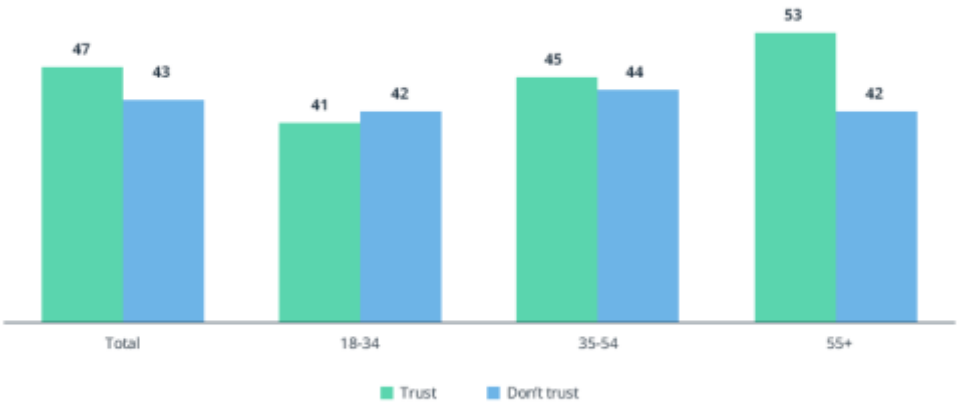
بررسی برخی کاربردهای قرارداد های هوشمند

در این قسمت به بررسی کاربردهایی از قرارداد های هوشمند در حوزه های بیمه و زنجیره تامین و مشاور املاک به صورت تخصصی تر می پردازیم.

بیمه

هدف از بکارگیری قراردادهای هوشمند در بیمه

چالش اصلی صنعت بیمه عدم اعتماد و شفافیت بین مجریان است. تنها 29 درصد از مشتریان به بیمه‌گران اعتماد دارند. این عدم اعتماد متقابل است. کلاهبرداران معمولاً به امید دریافت پرداختی ادعاهای نادرست می کنند و بیمه گذاران را مجبور می کنند که منابع اضافی را برای تأیید هر ادعا اختصاص دهند. با وجود قراردادهای هوشمند، مشکل اعتماد را می توان حداقل تا حدی از بین برد و در عین حال هزینه های اداری را کاهش داد.



تصویر 18. نمودار اعتماد مشتریان به بیمه در آمریکا

معماری قرارداد های هوشمند در بیمه

از نقطه نظر معماری، مناسب ترین انتخاب در بیمه، اتخاذ ترکیبی از بلاک چین های خصوصی و عمومی است. بلاک چین خصوصی می تواند برای ثبت سیاست ها و داده های ادعا استفاده شود، در حالی که بلاک چین عمومی می تواند برای بازپرداخت با استفاده از ارزهای دیجیتال قابل مبادله (مانند اترها یا بیت کوین ها) استفاده شود. بلاک چین خصوصی را می‌توان توسط

رایانه ها/گره‌های شرکت مورد اعتماد که با هزینه‌های استخراج پایین‌تر نسبت به بلاک‌چین‌های عمومی مشخص می‌شوند، حفظ کرد. بلاک چین عمومی توسط عموم مردم حفظ می شود. از طرف دیگر، شرکت بیمه می تواند تصمیم بگیرد که فقط از یک بلاک چین عمومی بهره برداری کند. در صورتی که شرکت نیاز به افزایش شهرت خود و جلب اعتماد مشتریان داشته باشد، این انتخاب می‌تواند موفقیت‌آمیز باشد، اما هزینه‌های تراکنش بالاتری را به همراه خواهد داشت.

در مورد نوع معماری، شرکت‌هایی که هدفشان ارائه خدمات پرداخت به ازای استفاده است(در ادامه توضیح خواهیم داد)، می‌توانند بر یک بلاک چین عمومی تکیه کنند. به این ترتیب، یک قرارداد هوشمند می تواند پول را از مشتریان (به عنوان مثال: اترها یا بیت کوین ها) جمع آوری کند، آنها را تا یک تاریخ معین نگه دارد و در صورت عدم بروز خسارت، آنها را به شرکت بیمه منتقل کند. با قرار گرفتن در یک بلاک چین عمومی، همه می توانند کد قرارداد هوشمند را بررسی کنند و اعتماد بین طرفین را افزایش دهند.

کاربردهای smart contract در بیمه

1. در این مورد، می‌توان از بلاک چین و قراردادهای هوشمند برای افزایش سرعت پردازش و همچنین کاهش هزینه‌ها (و اشتباهات) مرتبط با پردازش دستی استفاده کرد. از این منظر، یک قرارداد هوشمند می تواند قوانینی را برای امکان انتقال بازپرداخت از شرکت به بیمه گذار رمزگذاری کند.به عنوان مثال یک برنامه ساده می‌تواند شامل انتقال خودکار بازپرداخت تنها در صورتی باشد که مشتری خودرو را نزد یک مکانیک تأیید شده تعمیر کند و مکانیک برای اثبات هویت آن، تراکنش را به قرارداد هوشمند ارسال کند.

2. موارد استفاده پیچیده تر همچنین می تواند شامل اوراکل ها برای جمع آوری اطلاعات از دنیای واقعی باشد. برای مثال، در بیمه محصولات، اوراکل می‌تواند به صورت دوره‌ای داده‌های آب‌وهوا را بررسی کند و این اطلاعات را در زنجیره بلوکی قرار دهد. سپس یک قرارداد هوشمند می تواند این داده ها را بخواند و در صورت تداوم آب و هوای بد، پرداخت را آغاز کند.

اوراکل عاملی است که داده های خارجی را به دست آورده و تأیید می کند و آن را در اختیار قراردادهای هوشمند قرار می دهد. اوراکل ها ممکن است به شکل وب API ، حسگرها یا هر رابط ماشین به ماشین دیگری باشند.

3. در این مورد، تمرکز بر بیمه‌های مسافرتی و ایده‌ها برای بهره‌برداری از قراردادهای هوشمند بر روی بلاک‌چین اتریوم ایجاد می‌شود تا در صورت تأخیر در پرواز/قطار مسافران، به‌طور خودکار هزینه بازپرداخت کنند.

4. همچنین بخشی از بیمه با IoT درآمیخته است. مثل سیستم های هوشمندی که از حسگرها در منازل استفاده میشوند. با حسگرهایی که می توانند مستقیماً یک قرارداد هوشمند را از آسیب اطلاع دهند (به عنوان مثال، سنسورهای رطوبت سنج می توانند برای شناسایی اسیب های وارد بر سقف استفاده شوند). به طور مشابه، لوازم خانگی هوشمند می توانند به طور خودکار وضعیت خود را کنترل کنند و مشکلی را گزارش کنند یا در صورت نیاز مستقیماً برای کمک سریعتر با تعمیرکار تماس بگیرند.

5. مزیت دیگر این است که همه می توانند قرارداد هوشمند را بررسی کنند. یعنی مشتری که زیر یک قرارداد را امضا می‌کند،درک روشنی از شرایط قراردادی آن خواهد داشت در نتیجه، مقایسه شرایط برای او آسان‌تر می‌شود. علاوه بر این، انتخاب یک سیاست نه تنها بر اساس میزان اعتماد او به یک شرکت معین بلکه بر اساس داده های عینی است. زیرا اعتماد به طور ضمنی توسط قرارداد هوشمند تضمین می شود.

6. از طرف دیگر مکانیسم رمزنگاری زیربنای زنجیره بلوکی می تواند برای کاهش هزینه های اضافی مربوط به ورود دستی داده ها و تأیید مشتریان جدید استفاده شود.

7. با بلاک چین، مشتریان با یک آدرس منحصر به فرد (به عنوان مثال، آدرسی که به کیف پول آنها مرتبط است) شناسایی می شوند. اولین بار که آنها از یک سرویس استفاده می کنند، یک واسطه تایید شده هویت آنها را تأیید می کند و آن را به آدرس آنها پیوند می دهد. از آن زمان به بعد، هر بار که آنها زیر یک قرارداد امضا می کنند، دیگر نیازی به ارائه مدرک شناسایی ندارند. بلکه آنها فقط باید از اعتبار خود استفاده کنند. مزیت این مورد این است که منجر به کاهش زمان و هزینه جمع آوری/ارائه اطلاعات دوباره میشود.

8. از طرف دیگر بیمه‌های "پرداخت به ازای استفاده" می‌توانند در ترکیب با فرایندiot برای ثبت نام خودکار،قابل اجرا شوند. به عنوان مثال، از داده‌های GPS می‌توان برای جمع‌آوری خودکار استفاده کرد، به‌عنوان مثال، حق بیمه سفر فقط در صورتی پرداخت میشود که مشتری در خارج از کشور باشد، یا حق بیمه خودرو فقط زمانی اعمال میشود که خودرو در حال حرکت است، و غیره..

9. کاربرد مهم دیگر آن برای مثال با یک قرارداد هوشمند یک شخص می تواند خواسته خود را در زنجیره بلوکی به فرم یک قانون تنظیم شده رمزگذاری کند. در صورت فوت قرارداد هوشمند می تواند به طور خودکار پول وصیت کننده یا سایر انواع دارایی ها را به ذینفع منتقل کند. وصی می‌تواند محدودیت‌های دیگری را نیز ایجاد کند مانند امکان انتقال تنها زمانی که ذینفع به سن بلوغ برسد زمانی که وی دیپلم بگیرد و غیره.از آنجایی که شرایط قراردادهای هوشمند مبتنی بر داده های ذخیره شده در بلاک چین است، آنها باید به داده های خارجی تکیه کنند، که داده ها را از دنیای واقعی (به عنوان مثال، از سوابق مرگ) گرفته و آنها را به زنجیره بلوکی (یا برعکس) سوق می دهد .از این خدمات به عنوان «اوراکل» یاد می شود .با در نظر گرفتن مثال وصی، یک اوراکل می تواند سوابق مرگ را بررسی کند تا تشخیص دهد که آیا شخص فوت کرده است یا خیر .اگر چنین است، می‌تواند این اطلاعات را روی بلاک چین بنویسد (به عنوان مثال، با تغییر مقدار یک متغیر بولی که نشان می‌دهد فرد زنده است یا خیر) .بنابراین، قرارداد هوشمند یک عبارت شرطی (بر اساس مقدار متغیر) را راه‌اندازی می‌کند و بلوک کدی را که انتقال پول را آغاز می‌کند، اجرا می‌کند.

10. یکی دیگر از برنامه های کاربردی قرارداد هوشمند در بیمه که برای پیشگیری از تقلب ارائه شده است این است که می تواند تمام اطلاعات مرتبط با یک فرد را بخواند و به طور خودکار حق بیمه را محاسبه کند و ارزیابی ریسک را بر اساس سلامت جسمانی، رفتارهای رانندگی و غیره انجام دهد .در این سناریو، قرارداد هوشمند می‌تواند داده‌ها را جمع‌آوری کرده و در حین پردازش ادعای تقلب را شناسایی کند (به عنوان مثال، از طریق بررسی و مقایسه با داده‌های مربوط به ادعاهای قبلی یک شخص). اطلاعات هر شخص به صورت منحصر به فردی در بلاک چین ها ذخیره میشود و بخش های مختلف (شرکت های بیمه، افسران پلیس، کادر پزشکی و غیره) باید برای کیفیت بهتر نتایج همکاری کنند. البته مسئله حریم شخصی افراد هم باید رعایت شود و تنها بخش های منتخب اجازه انتساب این اطاعات به هویت فرد را داشته باشند.

این قبیل کاربردهای قرارداد های هوشمند در بیمه برای طرف های قرار داد مزیت هایی دارند: برای شرکت بیمه، که می‌تواند میزان منابعی که معمولاً برای رسیدگی به مطالبات اختصاص می‌یابد را کاهش دهد، و همچنین برای مشتریانی که حتی قبل از آگاهی از خسارت، پول دریافت می‌کنند.

مزایای استفاده از قراردادهای هوشمند در بیمه

1. Less fraud through transparency : تقلب کمتر در بیمه به دلیل شفافیت و همچنین ماهیت غیرمتمرکز و باز بلاک چین ها امکان پذیر است. بدون مالک، هر کسی می تواند هر تراکنش ثبت شده در پایگاه داده بلاک چین را ببیند. در صورت ایجاد هرگونه تغییر در قراردادهای هوشمند بیمه، همه طرفین آن را مشاهده خواهند کرد و هیچ ناهماهنگی از قلم نخواهد افتاد.

2. Task automation : اتوماسیون وظایف با بلاک چین، تمام فرآیندهای مرتبط با قراردادهای هوشمند به صورت خودکار و ایمن ارائه می شوند. حذف نیاز به واسطه و نیروی انسانی مزیت اصلی استفاده از قرارداد هوشمند در بیمه است. این خطر دستکاری توسط شرکت کنندگان شخص ثالث را کاهش می دهد. علاوه بر این، بلاک چین که برای بیمه قراردادهای هوشمند اعمال می‌شود، به شرکت‌ها اجازه می‌دهد تا رویه‌ها و فرآیندهای خود را به روشی شفاف‌تر و راحت‌تر بررسی کنند.

3. Save time on verifying claims : در روند تأیید ادعاها در زمان صرفه جویی میشود. با توجه به کاربردهایی که بالاتر گفته شد و همچنین پیوند بین بیمه و IoT.

4. Protect policy documents : بیمه گذاران می توانند اسناد بیمه نامه را در ledger های متعدد ذخیره کنند و از دست دادن آنها عملاً غیرممکن است. قراردادهای هوشمند به لطف ویژگی های فنی خود، از آسیب دیدگی داده ها جلوگیری می کنند.

5. Risk assessment : ارزیابی ریسک بلاک چین ها به شرکت های بیمه اجازه می دهند مدل های پیشرفته ارزیابی ریسک را در قراردادهای هوشمند خود بگنجانند. شناسه ها فوراً تأیید می شوند و با داده های جدید تکمیل می شوند و مراحل وقت گیر تأیید هویت سنتی را حذف می کنند. یک قرارداد هوشمند تمام اطلاعات مربوط به یک فرد را می خواند و به طور خودکار خطرات را ارزیابی می کند و در زمان و تلاش برای جمع آوری و تأیید داده ها صرفه جویی می کند.

محدودیت ها و مشکلات قراردادهای هوشمند در بیمه

قراردادهای هوشمند محدودیت هایی دارند. در حال حاضر، قراردادهای هوشمند فقط برای ابتدایی ترین انواع پرونده های بیمه می توانند ارائه شوند. به عبارت بسیار ساده، قراردادهای هوشمند می توانند تنها بر اساس یک الگوی مشروط اگر X ، آنگاه Y عمل کنند.

1. قراردادهای هوشمند تنها زمانی قابل اجرا می شوند که شرایط آنها به طور کامل به کد برنامه نویسی تبدیل شود و پوشش هر احتمالی در کد قرارداد لازم است اما متاسفانه ممکن است تبدیل به کد کارهایی که به راحتی روی کاغذ انجام میشوند دشوار باشد. به عنوان مثال، مفاهیم خاص صنعت مانند حسن نیت (good faith) یا معقول بودن (reasonableness) به طور بالقوه نمی‌توانند با قوانین ساده‌ای که قراردادهای هوشمند در حال حاضر مبتنی بر آن هستند بیان شود. برای توصیف همه موارد احتمالی و سناریو های پیچیده نیاز به مقدار بیشماری کد و منابع است.

2. علاوه بر این، از آنجایی که بیمه یک صنعت بسیار محافظه کار است، بسیاری از اعتماد به فناوری به جای یک شخص ثالث معمولی تردید دارند. با قراردادهای هوشمند، ما واقعاً واسطه را حذف نمی کنیم. ما فقط از شر عامل انسانی خلاص می شویم و کدهای کامپیوتری را جایگزین می کنیم. در حالی که کد تعبیه شده در یک قرارداد هوشمند خطر بسیار کمی برای هک شدن دارد، خود کد ممکن است ناقص باشد.

3. مهم ترین نقطه ضعف مربوط به مقیاس پذیری، مصرف انرژی و عملکرد است. در واقع، در حال حاضر، تعداد تراکنش‌هایی که می‌توان در هر ثانیه انجام داد، در مقایسه با سیستم‌های سنتی بسیار کم است (عمدتاً به دلیل قدرت محاسباتی مورد نیاز برای اعتبارسنجی بلوک‌های جدید). در این رابطه، شایان ذکر است که برخی از پلتفرم‌های بلاک چین در حال تغییر فرآیند اعتبارسنجی بلوک‌ها، کاهش پیچیدگی مشکل ریاضی برای حل کردن و محدود کردن امکان انجام ماینینگ تنها به زیرمجموعه‌ای از گره‌های قابل اعتماد هستند. جدا از زمان، فضا نیز یک مسئله است، زیرا داده ها در هر گره شبکه تکرار می شوند. علاوه بر این، مقدار انرژی مصرف شده توسط گره های شبکه و هزینه سخت افزار مورد نیاز برای اعتبارسنجی بلوک های جدید بسیار بالا است (البته باید تاکید کرد که چندین ابتکار برای محدود کردن مقدار انرژی مصرفی در حال حاضر در حال توسعه است.

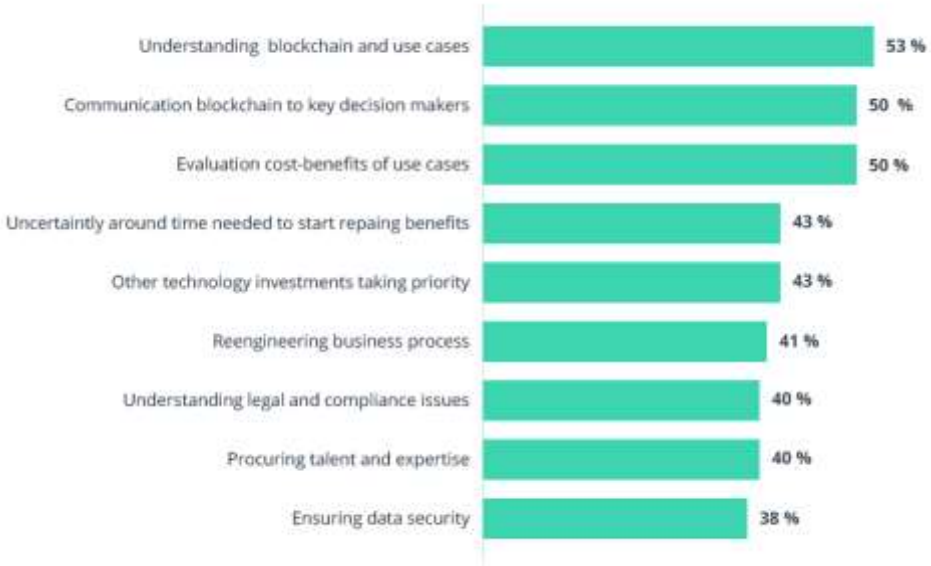
4. اشکالات احتمالی در کد قراردادهای هوشمند پیچیده هستند .از آنجایی که آنها به صورت متوالی اجرا می شوند، اگر حداقل یک قطعه حیاتی از دست رفته باشد، قرارداد اجرا نمی شود .حتی اگر حذف نیروی انسانی یکی از مزیت‌های برتر قراردادهای هوشمند در بیمه است، قراردادهای هوشمند همچنان نیازمند مشارکت انسانی در مرحله توسعه هستند.

5. کاربردهای بیان شده در فوق فقط برای تعداد محدودی از سیاست ها قابل اتخاذ است. در واقع، اکثریت خسارت های رسیدگی شده توسط شرکت های بیمه هنوز باید قبل از تسویه حساب توسط یک کارشناس خارجی ارزیابی شود.

6. استفاده از اسمارت کانترکت در بیمه دارای اشکالاتی است که شرکت باید از آنها آگاه باشد. اولین اشکال مربوط به از دست دادن / سرقت احتمالی اعتبار است. همانطور که گفته شد، از آنجایی که بلاک چین بدون واسطه کار می کند، هیچکس نمی تواند اعتبار کاربران را بازنشانی کند. یک راه حل می تواند تکیه بر سرویس های خارجی باشد که می توانند اعتبارنامه ها را ذخیره کنند و آن ها را به کاربران در صورت گمشدن بازگردانند. با این حال، استفاده از چنین خدماتی به معنای دسترسی دیگران به اطلاعات حساس است و برخی از دولت ها نمی توانند به فناوری در این مورد اعتماد کنند و از لحاظ قانونی اجازه ذخیره اطلاعات هویتی در بلاک چین ها را نمیدهند.

برای حل این مشکل ،از بلاک چین برای اجازه دادن به چندین واسطه تایید شده برای ثبت اطلاعات مربوط به یک شخص (یا پیوند دادن آنها به آدرس وی) استفاده می شود. چنین واسطه‌هایی می‌توانند شرکت‌های بیمه (مانند ثبت ادعاهای قبلی)، افسران پلیس (مثلاً برای ذخیره اعمال مجرمانه)، کادر پزشکی (مثلاً برای ثبت جراحات و درمان‌های یک فرد)، یا حتی دستگاه‌های هوشمندی که انسان استفاده میکند (که می‌توانند داده‌های مربوط به فعالیت بدنی فرد را در بلاک چین ذخیره کنند) باشند.

تصویر زیر برخی از نگرانی هایی که قراردادهای هوشمند را محدود می کند نشان میدهد :



تصویر 19. برخی از نگرانی های محدود کننده قرارداد هوشمند

زنجیره تامین

با پیشرفت سطح زندگی روزمره، افزایش سطح کیفی کالاهای مورد استفاده مردم، مورد توجه زیادی قرار گرفته است. بر همین مبنا،لازم است سیستم‌هایی فراهم شود تا مردم بتوانند از مراحل پردازش مواد و در نهایت ساخت کالاهای مورد استفاده خود آگاه شوند. این مراحل از استخراج مواد اولیه و ارسال برای کارخانه‌ها و ایجاد محصول گرفته و تا زمان فروش توسط فروشندگان ادامه می‌یابد. این موضوع خصوصا در مورد صنایع غذایی اهمیت زیادی دارد. مصرف کنندگان تمایل دارند تا از سلامت مواد غذایی خود اطمینان حاصل کنند. بنابراین با فراهم کردن بستری برای مشاهده این زنجیره، ارگان‌ها می‌توانند، اعتماد مصرف‌کنندگان را جلب کنند.

سیستم‌هایی که با هدف نظارت بر این زنجیره تامین امروزه موجود هستند،سیستم های متمرکزی هستند که توسط ارگان‌های مختلف اداره می‌شوند. این موضوع موجب عدم شفافیت در نحوه ذخیره سازی داده ها می‌شود. چرا که ممکن است ارگان‌ها بنا به منافع خود و به طور محرمانه دست به تغییر این داده‌ها بزنند. این موضوع باعث عدم اعتماد مردم به این ارگان ها می‌شود. از طرفی یکی از اهداف نظارت بر زنجیره تامین،یافتن منشا مشکلات احتمالی است. باید توجه کرد که در صورتی که این اطلاعات تغییر کند، موجب ایجاد ناهماهنگی بین داده های ارگان‌های مختلف در زنجیره تامین می‌شود که در نتیجه موجب سخت شدن ردیابی مشکلات می‌شود. به عنوان نمونه‌ای از ارگان‌های نظارتی بر زنجیره تامین، می‌توان به ارگان NLIS در استرالیا اشاره کرد. این ارگان تمام داده‌های مرتبط با کالا را از مواد اولیه تا کالای نهایی ثبت و ذخیره می‌کند.

در همین راستا سیستمی ارائه شده تا با استفاده از تکنولوژی بلاکچین و قراردادهای هوشمند به این نیازمندی‌ها پاسخ دهد. در ادامه با مفاهیم این سیستم و در نهایت با نحوه کارکرد آن آشنا می‌شویم.

این سیستم از یک شبکه بلاکچین برای ثبت تراکنش‌های مرتبط با زنجیره تامین استفاده می‌کند. با توجه به استفاده از بلاکچین، این سیستم، غیرمتمرکز در نظر گرفته می‌شود و همه افراد دسترسی به این سیستم دارند. همچنین به دلیل اینکه داده‌های این سیستم توسط همه اشخاص قابل مشاهده است، احتمال افزودن داده نادرست به چنین سیستمی کاهش می‌یابد.

از طرفی با توجه به ماهیت این سیستم، نودها به هم بی اعتماد هستند. برای حل این مشکل نیز از یک سیستم event response استفاده می‌شود.

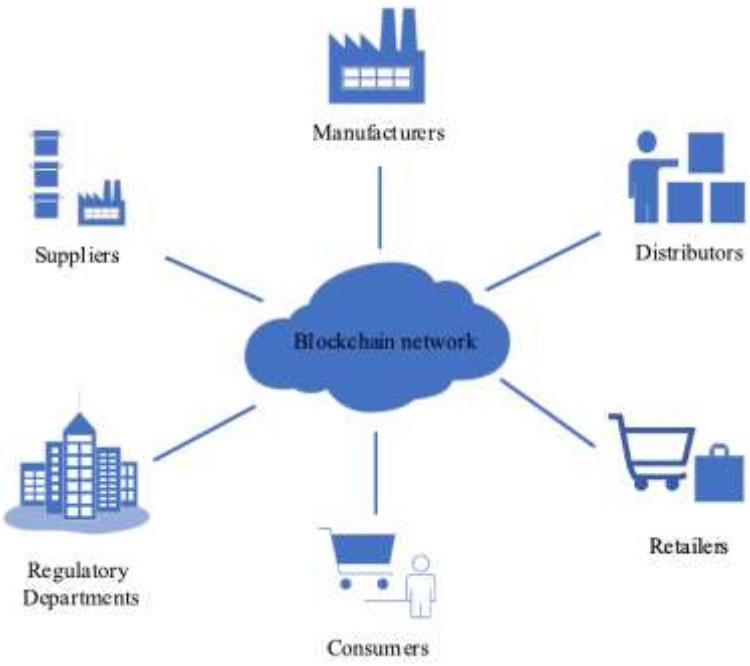
برخی از نیازمندی‌های امنیتی این سیستم به شرح زیر می‌باشد :

- داده‌ها باید برای همگان قابل دسترسی و مشاهده باشد.
- داده‌ها باید غیر قابل تغییر باشند.
- همه عملکرد سیستم باید طبق یک الگوریتم مشخص در یک محیط امن انجام شود.
- در برابر حملات MITM مقاوم باشد.

با توجه به پیچیدگی این سیستم در ادامه نقاط مختلف در زنجیره تامین به 5 دسته تقسیم‌بندی می‌شود :

- Supplier : مواد اولیه را برای کارخانجات فراهم می‌کند.
- Manufacturer : با استفاده از مواد خام، محصولی را تولید می‌کند.
- Distributor : محصولات تولید شده را بین فروشندگان توزیع می‌کند.
- Retailer : محصولات را به مصرف‌کنندگان می فروشد.

در این سیستم هر محصول جزوی از یک دسته (batch) می‌باشد. یک دسته می‌تواند شامل تعداد یک یا بیشتر از یک کالا باشد. در این سیستم فرض می‌شود تمام کالاهای مرتبط با یک دسته دارای کیفیت یکسانی می‌باشند. در واقع در فرایند انتقال از یک نقطه به نقطه دیگر، دسته‌ها در نظر گرفته می‌شوند و نه کالاها. همچنین می‌توان از یک سازمان مستقل برای نظارت بر همه این فرایندها استفاده کرد. در تصویر زیر این نقش‌ها مشاهده می‌شوند :



تصویر 20. نقش های نقاط مختلف در زنجیره تامین

انواع قرارداد های هوشمند مورد استفاده در این سیستم

در این سیستم از سه نوع قرارداد هوشمند استفاده می‌شود :

- PRC: Product Registeration Contract
- BAC: Batch Addition Contract
- TAU: Transaction Update Contract

1. در قرارداد PRC اطلاعات مرتبط با هر کالا ذخیره می‌شود. همچنین در این قرارداد آدرس قراردادی که دسته مرتبط با این کالا قرار دارد نیز ذخیره می‌شود. این قرارداد یک تابع register در اختیار می‌گذارد، که از طریق آن می‌توان کالاها را در آن ذخیره کرد. همچنین در این قرارداد از یک لیست از افراد مجاز به تغییر این قرارداد استفاده می‌شود. در صورتی که فرستنده درخواست در این لیست موجود نباشد، امکان اجرای قرارداد را ندارد. الگوریتم تابع register در زیر مشاهده می‌شود.

Algorithm 1: `register()`

Input: The massage sender's address(`msg.sender`), product code (`productCode`), product name (`productName`), raw materials (`rawMaterials`), current timestamp (`now`), BAC address (`bacAddr`), authorization list (`AL`), product count (`productCount`)

```
1  AL is the set of all authorized users' Ethereum address in
   this contract;
2  if msg.sender ∈ AL then
3  |  if productCode has not exist then
4  |  |  register productCode, productName,
   |  |  msg.sender, rawMaterials, now, and
   |  |  bacAddr to the blockchain;
5  |  |  productCount++;
6  |  else
7  |  |  Revert contract state and show an error.
8  |  end
9  else
10 |  Revert contract state and show an error.
11 end
```

تصویر 21. الگوریتم تابع Register

در تصویر زیر نمونه‌ای از داده‌های این قرارداد مشاهده می‌شود :

id	Product Code	Product Name	Product Owner	Raw Materials	Timestamp	BAC Address
1	109735813	cocoa	0xc07...f0e4	/	1562065155	0xbce0...d4c77
2	165456413	milk	0x32c...d192	/	1562065206	0x5695...9209c
3	6928480334788	coffee	0x69a...0013	109735813, 165456413	1562065276	0xa911...0ac5a
4	163113107	tea	0x38c...0d2a	/	1562065411	0xa145...0e6d4
...	...	...	...	...	...	...

تصویر 22. نمونه ای از داده های قرارداد

همانطور که در این تصویر مشاهده می‌شود، کالاهای کاکائو و شیر مواد اولیه هستند. قهوه یک کالا است که با استفاده از مواد اولیه شیر و کاکائو تهیه شده است.

2. در قرارداد BAC اطلاعات مرتبط با هر دسته از کالاها ذخیره می‌شود. همچنین آدرس قرارداد مرتبط با TUC که اطلاعات انتقال بسته‌ها را در بر دارد، ذخیره می‌کند. این قرارداد یک تابع addBatch در اختیار ما قرار می‌دهد که با استفاده از آن می‌توان داده‌های مرتبط با یک دسته را ذخیره کرد. الگوریتم این تابع در تصویر زیر مشاهده می‌شود:

Algorithm 2: `addBatch()`

Input: The massage sender's address (`msg.sender`), batch number (`batchNumber`), raw materials used (`materialBatchNumber`), current timestamp (`now`), TUC address (`tucAddr`), authorization list (`AL`), batch count (`batchCount`)

```
1  AL is the set of all authorized users' Ethereum address in
   this contract;
2  if msg.sender ∈ AL then
3      if batchNumber has not exist then
4          register batchNumber,
           materialBatchNumber, msg.sender,
           now, and tucAddr to the blockchain;
5          productCount++;
6      else
7          Revert contract state and show an error.
8      end
9  else
10     Revert contract state and show an error.
11 end
```

تصویر 23. الگوریتم تابع addBatch

در تصویر زیر نمونه‌ای از داده‌های این قرارداد مشاهده می‌شود که مربوط به کالا با کد 6928480334788 در تصویر مثال PRC می‌باشد.

id	Batch Number	Raw Materials and Their Batch Numbers	Batch Manager	Timestamp	TUC Address
1	201907021605	109735813(201906281026),165456413(201906211139)	0x69a...0013	1562622724	0xc246...1eebf
2	201907031206	109735813(201906291536),165456413(201906211139)	0x38c...0d2a	1562722756	0xe1b2...d7059
3	201907031816	109735813(201906301538),165456413(201906211953)	0x90f...c9c1	1562822789	0x8776...13dd4
4	201907041028	109735813(201906301538),165456413(201906221642)	0xffc...09f0	1562922816	0x4994...cd779
...	...	...	...	...	...

تصویر 24. نمونه ای داده های مربوط به کالا

همانطور که در این تصویر مشاهده می‌شود تمامی دسته‌های نشان داده شده از همان مواد اولیه شیر و کاکائو موجود در نمونه داده PRC استفاده می‌کنند.

3. قرارداد سوم قرارداد TUC بوده که اطلاعات مرتبط با انتقال دسته‌ها از یک نقطه به نقطه دیگر ذخیره می‌شود. این قرارداد اطلاعاتی از قبیل فرستنده و گیرنده و هش تراکنش را در بر دارد. این قرارداد تابع updateTr را در اختیار ما قرار می‌دهد که به وسیله آن یک انتقال دسته‌ای از کالاها ثبت می‌شود. در تصویر زیر الگوریتم این تابع مشاهده می‌شود.

Algorithm 3: `updateTr()`

Input: The massage sender's address (`msg.sender`), receiver's address (`receiver`), current timestamp (`now`), current tr (`currentTr`), previous tr (`previousTr`), authorization list (`AL`), tr count (`trCount`)

```
1  AL is the set of all authorized users' Ethereum address in
   this contract;
2  if msg.sender ∈ AL then
3  |  if previousTr has valid in the blockchain then
4  |  |  register currentTr, msg.sender,
   |  |  receiver, previousTr, and now to the
   |  |  blockchain;
5  |  |  productCount++;
6  |  else
7  |  |  Revert contract state and show an error.
8  |  end
9  else
10 |  Revert contract state and show an error.
11 end
```

تصویر 25. الگوریتم تابع updateTr

در تصویر زیر نمونه‌ای از رکوردهای مرتبط با این قرارداد مشاهده می‌شود که مربوط به دسته 201907021605 در نمونه BAC می‌باشد.

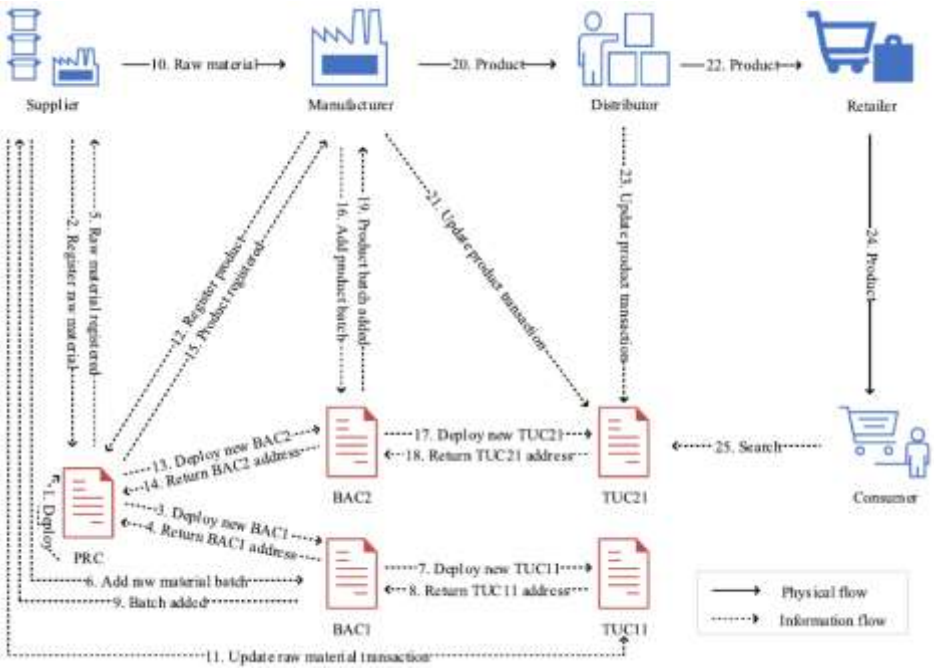
id	TrHash	Sender	Receiver	PreviousTr	Timestamp
1	0xa6036...8f5718c	0x3a0...c89b	0xdde...08a0	6928480334788(201907021605)	1563623747
2	0x0ade4...2193fe	0xdde...08a0	0x2bb...9255	0xa6036...8f5718c	1563823837
3	0x6c586...e409a5	0x2bb...9255	0xca7...c644	0x0ade4...2193fe	1563623953
4	0xdfa0b...df2467	0xca7...c644	0xaa4...da63	0x6c586...e409a5	1563624067
...	...	...	...	...	...

تصویر 26. نمونه ای از رکورد های مربوط به دسته ی 201907021605

این تصویر نشان‌می دهد دسته مورد نظر از چه نقاطی گذشته است. همانطور که در الگوریتم updateTr نیز مشاهده می‌شود.این در افزودن هر رکورد به این داده‌ها نوعی اعتبار سنجی نیز انجام می‌شود. به این صورت که برای افزودن هر تراکنش در این داده‌ها بررسی می‌شود که PriviousTr مقدار درستی داشته باشد.

نحوه کلی اجرای قرارداد های هوشمند در زنجیره تامین

در ادامه با استفاده از این توضیحات و مفاهیم، نحوه کارکرد این سیستم شرح داده می‌شود.



تصویر 27. نحوه کارکرد این سیستم

ابتدا یک قرارداد PRC به وجود می‌آید. این قرارداد شامل اطلاعات تمام مواد اولیه و مواد ساخته شده از مواد اولیه می‌باشد(قدم 1). پس از آن supplier با استفاده از تابع register، مواد اولیه خود را در قرارداد وارد می‌کند(قدم 2). این تابع نیازمند آدرس یک قرار داد BAC می‌باشد. از همین رو این تابع همیچنین یک قرارداد BAC ایجاد کرده و آدرس آن را به عنوان آدرس قرارداد BAC برای دسته ای که کالا در آن موجود است ذخیره می‌کند. در نهایت ایجاد این قرارداد به اطلاع supplier می‌رسد (قدم‌های 3 تا 5). پس از آن supplier مشخصات دسته کالا را در قرارداد BAC مربوطه، با استفاده از تابع addBatch ذخیره می‌کند. این قرارداد نیازمند آدرس یک قرارداد TUC می‌باشد. بنابراین تابع addBatch یک قرارداد TUC جدید ایجاد کرده و آدرس این قرارداد در رکورد دسته مربوطه ذخیره می‌شود (قدم‌های 6 تا 9). پیش از انتقال بسته‌ها به manufacturer تعدادی event و response رخ می‌دهد. ابتدا manufacturer یک event صادر می‌کند و مواد مورد نیاز خود را اعلام می‌کند. پس از دریافت این رخداد، supplier یک رخداد در جواب صادر می‌کند و برای ارسال مواد خواسته شده اعلام آمادگی می‌کند. پس از آن مواد خواسته شده را ارسال می‌کند. پس از دریافت این مواد توسط manufacturer یک رخداد دیگر صادر می‌شود که نشان‌دهنده دریافت مواد توسط manufacturer می‌باشد.(قدم 10) پس از دریافت این رخداد توسط supplier یک رکورد جدید در قرارداد TUC مربوطه ذخیره می‌شود که نشان‌دهنده این انتقال می‌باشد (قدم 11). پس از دریافت مواد توسط manufacturer و ایجاد کالای جدید با استفاده از مواد خام،کالای جدید را در PRC مرتبط ذخیره می‌کند. (مشابه با قسمت قبل رخ می‌دهد). Manufacturer از تابع register استفاده کرده و موجب ایجاد یک قرارداد BAC می‌شود (قدم‌های 12 تا 15). پس از آن manufacturer دسته‌های مرتبط با کالا را در قرارداد BAC با استفاده از تابع addBatch ذخیره می‌کند که به ترتیب یک قراردادر TUC ایجاد می‌کند (قدم‌های 16 تا 19). مجددا مشابه انتقال بین supplier و manufacturer، رخدادهای ذکر شده، بین manufacturer و distributor انجام می‌شود. کالاها به دست distributor رسیده و یک تراکنش جدید در TUC مربوطه، توسط manufacturer ایجاد می‌شود (قدم های 20 و 21). مجددا رخدادها این بار بین distributor و retailer انجام شده و کالا برای retailer ارسال شده و در نهایت رکوردهای قرارداد TUC مربوطه

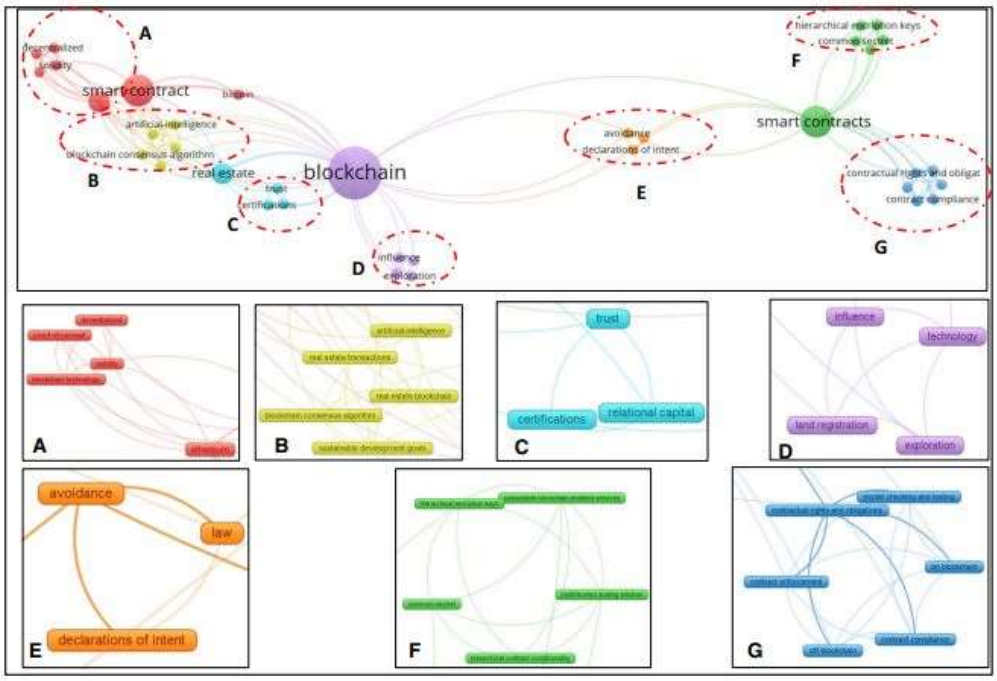
با استفاده از updateTr به روزرسانی می‌شود.(قدم‌های 22 و 23). در نهایت پس از خرید کالا توسط مصرف‌کننده می‌توان با استفاده از قرارداد TUC مرتبط با کالا تمامی مراحل ارسال را مشاهده کرد (قدم‌های 24 و 25).

مشاور املاک

مدیریت هوشمند املاک و مستغلات یک مفهوم جدید است که برای مدیریت تجارت املاک و معاملات شهر هوشمند معرفی شده است. به طور مشابه، مدیریت هوشمند املاک و مستغلات از طریق رایانه ها و فناوری های شبکه ای برای افزایش کیفیت زندگی کاربران می باشد. بنابراین، این یک حوزه وابسته به فناوری است که با سه جنبه کلیدی مشخص می شود: پایداری، نوآوری و تمرکز بر کاربر. مفهوم املاک و مستغلات هوشمند با معرفی خانه های هوشمند و دفاتر هوشمند آغاز شد و از آن زمان در حال تکامل است. بر این اساس، فناوری های مختلفی برای مدیریت هوشمند املاک و مستغلات در شهرهای هوشمند مورد بررسی قرار گرفته است که شامل استفاده از فناوری بلاک چین است. در نتیجه لازم است نگاهی متفاوت به ارتباط بلاک چین و قرارداد های هوشمند و سپس نقش قرارداد های هوشمند در زمینه مدیریت املاک داشته باشیم.

در مورد ارتباط بلاک چین و قرارداد هوشمند میتوان هفت دسته بندی و خوشه بندی ایجاد کرد.

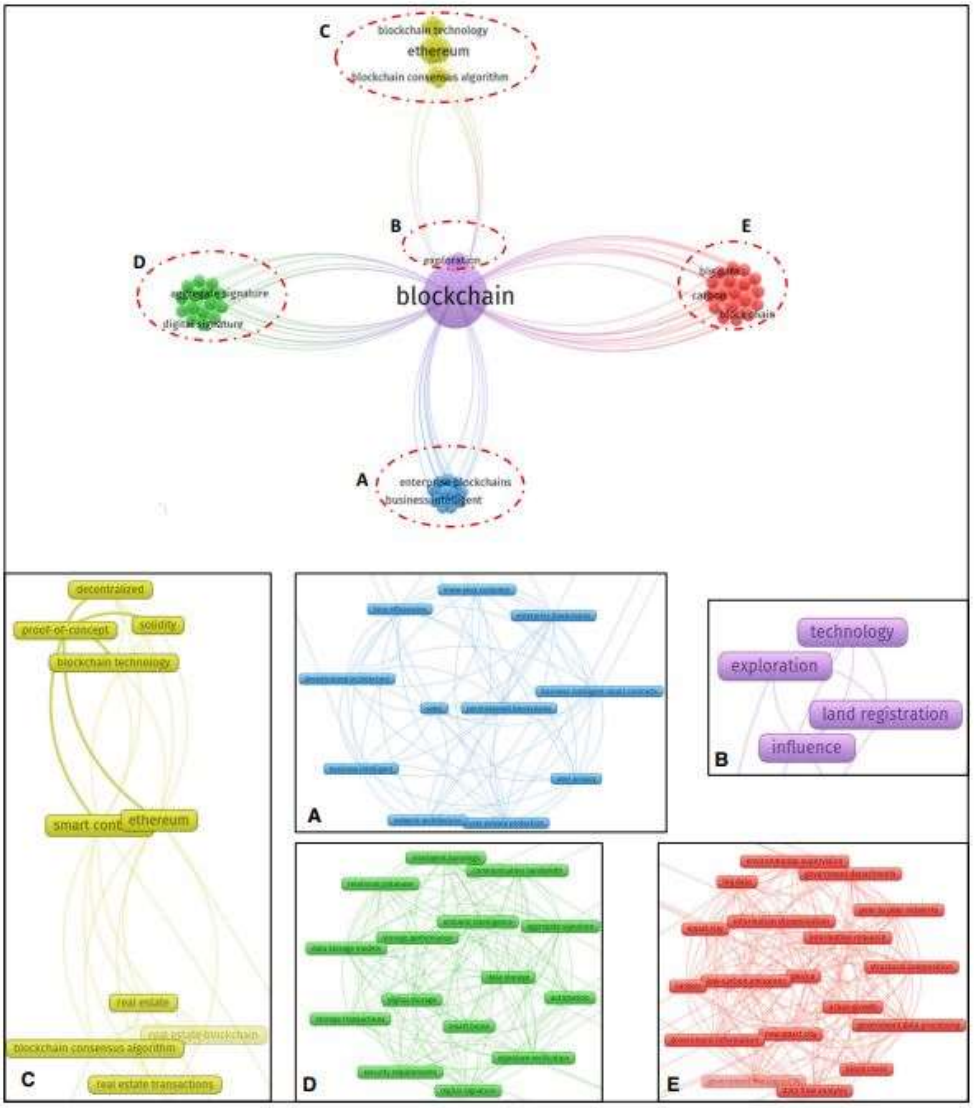
دسته بندی	کلمات کلیدی
کامپیوتری	Solidity ، اتریوم ، تمرکززدایی و بلاک چین
تراکنشی	real state transaction ، استفاده از هوش مصنوعی و اهداف توسعه پایدار
ارتباطی	اعتماد ، certifications
تکنولوژی	تاثیر تکنولوژی ، اکتشاف ، blockchain land registration
قانونی	قانون ، اجتناب
شبکه	راه حل مقیاس پذیر چند وجهی ، رمزگذاری سلسله مراتبی ، شرطی ، common secret
موافقت و انطباق دادن	مطابقت قرارداد ، اجرا و تعهد قرارداد



تصویر 28. ارتباط بین بلاک چین و قرارداد های هوشمند

همچنین در مورد بلاک چین هم میتوان 5 دسته بندی ایجاد کرد.

دسته بندی	کلمات کلیدی
سازمانی	بلاک چین سازمانی ، شناخت مشتری ، هوش تجاری داده های کاربر
تکنولوژی	تاثیر تکنولوژی ، اکتشاف ، blockchain land registration
کامپیوتری	Solidity ، اتریوم ، تمرکززدایی و بلاک چین
حافظه	ذخیره سازی داده ها ، ذخیره سازی دیجیتال ، تایید امضا ، اتوماسیون
اطلاعاتی	مدیریت اطلاعات ، زیرساخت اطلاعات انتشار اطلاعات

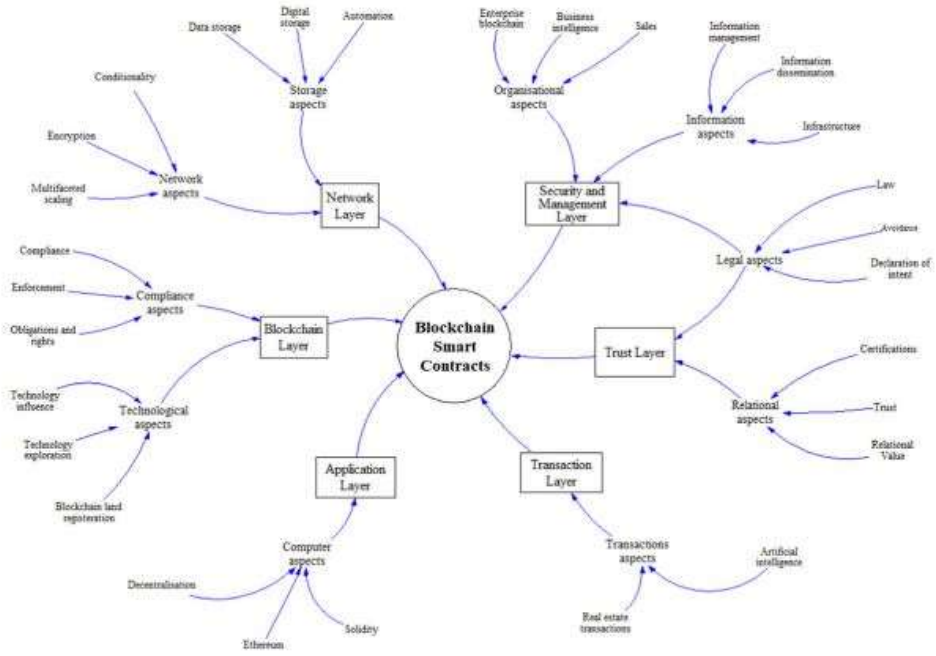


تصویر 29. دسته بندی بلاک چین

از دو نوع دسته بندی فوق نتیجه میگیریم که دو دسته بندی تکنولوژی و کامپیوتری که در بلاک چین وجود دارد ، برای ارتباط قرارداد هوشمند و بلاک چین نیز استفاده می شود. در کل ده تا دسته بندی می توانیم در این مورد داشته باشیم که عبارتند از: کامپیوتری ، تراکشنی ، ارتباطی ، تکنولوژی ، قانونی ، شبکه ، مطابقت دادن ، سازمانی ، حافظه ، اطلاعاتی.

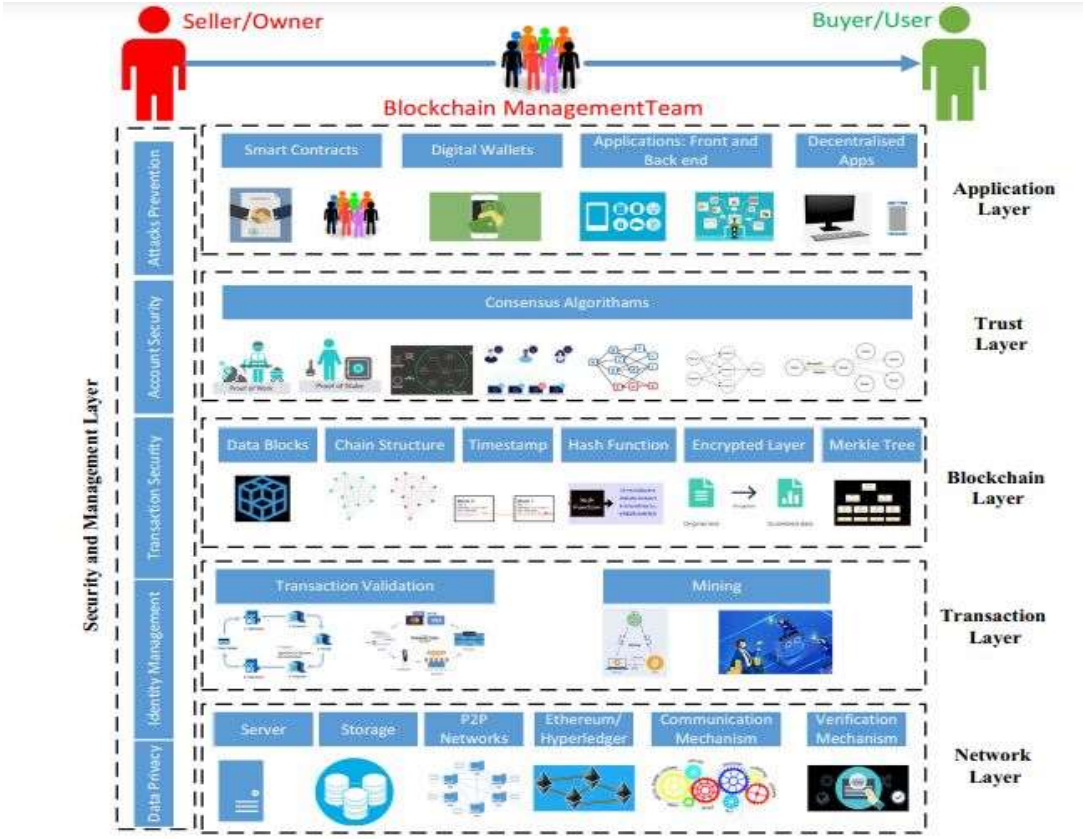
بلاک چین در فرآیند خرید/ اجاره هوشمند املاک و مستغلات

یک چارچوب 6 لایه برای آن پیشنهاد شده است. در واقع ده دسته بندی و جنبه ای که مطرح شد ، ادغام میشوند و در 6 لایه تقسیم بندی میشوند. برخی از جنبه ها، مانند جنبه های اطلاعاتی و سازمانی، در یک لایه امنیتی و مدیریتی ادغام می شوند. برخی به طور مستقیم به یک لایه مرتبط هستند، مانند جنبه های شبکه که به لایه شبکه مرتبط هستند، در حالی که برخی از جنبه ها به لایه های متعدد مرتبط هستند، مانند جنبه های قانونی که می تواند بر اعتماد و امنیت و لایه های مدیریتی تأثیر بگذارد.



تصویر 30. 6 لایه ی مشخص شده

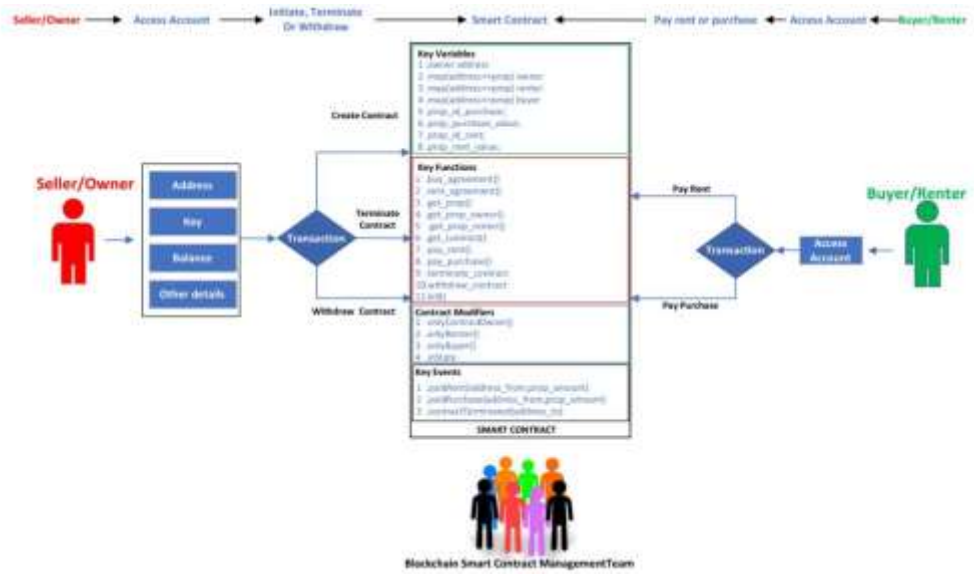
لایه	شامل چه مواردی میباشد یا اهداف آن چیست
شبکه	سرور ، دستگاه ذخیره حافظه ، شبکه P2P اتریوم یا نودهای Hyperledger ، ارتباطات و مکانیزم های تایید
تراکنش	شروع تراکنش ، اعتبار سنجی و پردازش و استخراج ارز ، (که توجه شود تراکنش میتواند توسط کاربر و یا مستقیما توسط smart contract شروع شود.)
بلاک چین	بلاک ها که حاوی اطلاعات لازم برای قرارداد هوشمند هستند ، تابع هش ، ساختار زنجیره ای ، لایه رمزگذاری شده ، کلید ها و ...
اعتماد و اطمینان	مکانیزم توسعه اجماع که در آن با استفاده از الگوریتم های اجماع، توافق بین طرفین یک قرارداد هوشمند حاصل می شود و برای احراز هویت تراکنش ها در شبکه استفاده میشود.
کاربرد	برنامه های فرانت اند و بک اند ، برنامه های غیرمتمرکز ، قراردادهای هوشمند و کیف پول دیجیتال
امنیت و مدیریت	ایمن سازی و مدیریت قراردادهای هوشمند بلاک چین



تصویر 31. فرآیند های انجام شده در هر لایه

طراحی و تعامل با قرارداد هوشمند برای مدیریت هوشمند املاک

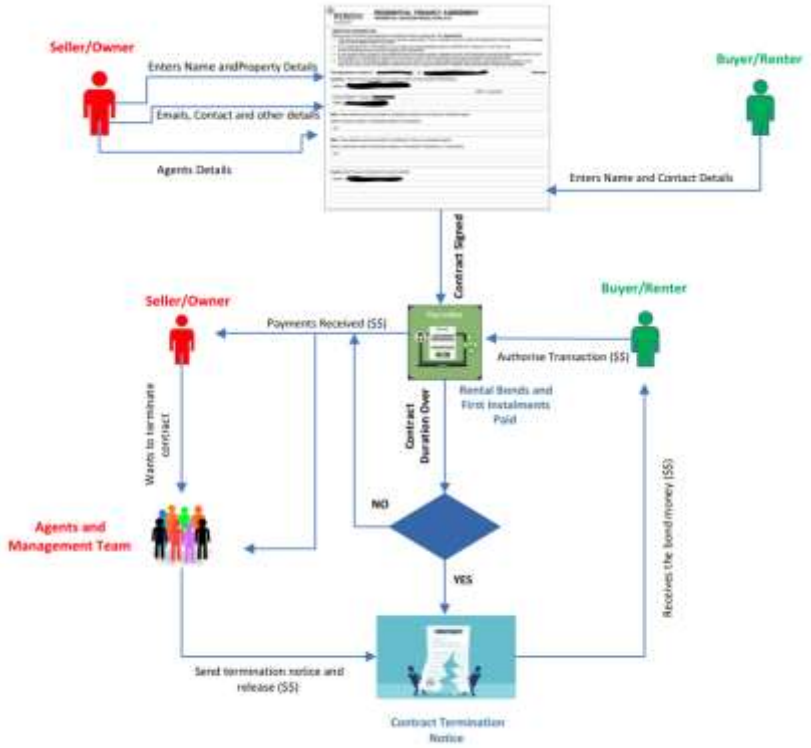
شکل زیر طراحی و تعامل سهامداران در قرارداد هوشمند برای مدیریت ملک را نشان میدهد. روند کلی به اینصورت هست که مالک آن ملک میتواند برای دیدن جزییات و بالانس و محاسبات و از این دسته کارها به حساب های سهامداران دسترسی داشته باشد. و میتوانند که قراردادی را ایجاد ، فسخ یا لغو کنند. عملکردهای کلیدی قرارداد هوشمند براین اساس فراخوانی می شوند. به طور مشابه، کاربر به حساب های خود دسترسی پیدا می کند تا تراکنش های مربوط به پرداخت های خرید یا اجاره را مجاز کند. توابع قرارداد هوشمند به جایی گفته میشود که جزئیات طرفین مبادله می شود و قرارداد هوشمند تنظیم می شود. تیم مدیریت قرارداد هوشمند بلاک چین این فرآیند را مدیریت می کند .متغیرهای کلیدی مختلف، توابع، اصلاح کننده های قرارداد و رویدادهای کلیدی در طول قرارداد هوشمند فراخوانی می شوند.



تصویر 32. طراحی قرارداد هوشمند برای مدیریت هوشمند املاک

نحوه پیاده‌سازی و شرایط خاتمه قرارداد

در ابتدا لازم به ذکر است که سیستم پیاده‌سازی شده، تنها با هدف دیجیتال کردن فرایندهای خرید و اجاره خانه به وجود آمده است. بنابراین این سیستم، جایگزین مشاورین املاک نمی‌شود. از طرفی برای انجام تراکنش‌ها در این سیستم از یک پرتال استفاده می‌شود. مراحل پیاده‌سازی یک قرارداد هوشمند و همچنین نحوه خاتمه آن با استفاده از سیستم ارائه شده، در تصویر زیر آمده است:



تصویر 33. نحوه پیاده سازی و شرایط خاتمه قرارداد

سیستم طراحی شده توسط یک صفحه وب قابل دسترسی می‌باشد. در ابتدا مالک، صفحه وب را باز کرده و مشخصات خود از قبیل آدرس ایمیل و کلیدهای رمزنگاری‌شده و همچنین مشخصات ملک خود را ثبت می‌کند. همچنین مالک، مشخصات مشاور املاک مورد نظر خود و یا هر شخص ثالث دیگری را در قرارداد ثبت می‌کند.

ازطرفی دیگر، شخص خریدار و یا اجاره کننده، نیز برای خرید/اجاره خانه، نیز صفحه وب را باز می‌کند و مشخصات قرارداد مورد نظر خود را تعیین می‌کند. برنامه وب با استفاده از داده‌های وارد شده توسط خریدار قرارداد موردنظر را پیدا می‌کند. این قرارداد توسط هر دو طرف به طور دیجیتال امضا می‌شود. بعد از کسب اجازه برداشت از حساب خریدار/اجاره‌کننده توسط سیستم، هزینه مورد نیاز (از طریق پرتال استفاده شده در سیستم) از حساب شخص برداشت می‌شود. قسمتی از این هزینه به شخص ثالث (مشاور املاک) و همچنین به صاحب سیستم پرداخت می‌شود. مابقی آن به حساب مالک واریز می‌شود. در ادامه با تایید مالک، حق مالکیت به شخص خریدار منتقل می‌شود. در مورد اجاره خانه نیز، واریز کرایه خانه به شیوه تنظیم شده در قرارداد انجام می‌شود. به این صورت که شخص اجاره کننده باید در زمان‌های تعیین شده هزینه‌ای را به عنوان کرایه خانه به صاحب خانه پرداخت کند.

قرارداد در صورتی که مربوط به خرید خانه باشد هیچ وقت خاتمه پیدا نمی‌کند. اما قراردادهایی که مربوط به کرایه خانه هستند ممکن است پس از مدت مشخص شده و یا با درخواست مالک منقضی می‌شوند. در صورتی که مالک تمایل به خاتمه قرارداد داشته باشد، پس از ثبت درخواست پایان قرارداد، این موضوع به اطلاع شخص ثالث(مشاور املاک) و همچین کرایه کننده می‌رسد. برای کرایه کننده، مهلتی مشخص شده که پس از آن موظف است ملک را تخلیه کرده و در اختیار مالک قرار دهد. همچنین در صورتی که هزینه‌ای از طرف کرایه‌کننده برای کرایه آینده پرداخت شده باشد، حساب او برخواهد گشت. در نهایت قرارداد پس از مدتی از زمان تخلیه منقضی می‌شود.

این سیستم در شهرهای هوشمند قابل استفاده بوده و در قراردادهای مشاورین املاک قابل بهره برداری می‌باشد.

مقایسه بین حوزه های بیمه ، زنجیره تامین و مشاور املاک

در این قسمت قصد داریم مقایسه ای میان کاربردهای قرارداد های هوشمند در سه حوزه بیمه ، زنجیره تامین و مشاور املاک و مستغلات داشته باشیم.

1. هدف از بکارگیری قراردادهای هوشمند در هر سه حوزه ، برقراری اعتماد متقابل میان طرفین قرارداد و همچنین افزایش کیفیت زندگی کاربران و آسان تر شدن فرآیندها بوده است.
2. خوشبختانه هر سه این سیستم ها غیر متمرکز بوده و همه افراد به این سیستم ها دسترسی دارند که باعث ایجاد شفافیت در فرآیندها شده. در نتیجه احتمال افزودن داده های نادرست و یا ادعاهای غلط را کاهش می دهد.
3. از طرفی دیگر با توجه به سوار شدن این قراردادها روی بلاک چین و خاصیت منحصر به فرد بلاک چین داده ها در همه این سیستم ها غیر قابل تغییر می باشند که این خاصیت از جهاتی می تواند مفید واقع شود.
4. لازم به ذکر است که این سیستم ها کاملا شخص ثالث را حذف نمی کنند. همان طور که قبلا اشاره شده بود سیستم مدیریت هوشمند املاک نمی تواند کاملا جایگزین مشاور املاک شود. همچنین این سیستم در بیمه نیز نمی تواند کاملا جایگزین کارشناسان بیمه شود و در اکثر موارد باید قبل از تسویه حساب خسارت ها توسط کارشناسان بیمه تایید شوند.

بکارگیری قرارداد های هوشمند در حوزه های متعدد با وجود پیشرفت های روز افزون و نقاط قوت زیاد همچنان چالش ها و نقاط ضعفی دارد که باید بهبود یابند.

پیاده سازی

با استفاده از SmartPy میخواهیم smart contract ای در حوزه لاتاری پیاده سازی کنیم. SmartPy یک کتابخانه پایتون است. از بلاک چین Tezos استفاده میکنیم که یکی از کاربردهای آن در smart contract هست. ارز دیجیتال این بلاک چین ، XTZ هست.

در کل روند پیاده سازی ما به اینصورت هست که برای قرارداد هوشمند یک تعریف کلاس میتوانیم بنویسیم که از sp.contract ارث میبرد. (توجه شود import smartpy as sp را در ابتدای کد مینویسیم). ذخیره سازی ها هم در constructor کلاس ها صورت میگیرد. نقاط ورودی یا به اصطلاح entry point ها هم یک متد از کلاس contract را تشکیل میدهند و به صورت @sp.entry_point مشخص میشوند. و میتوان نقاط ورودی های مختلف را تست کرد و نتایج را مشاهده کرد.

آشنایی با SmartPy

مانند اکثر زبانها ، expression هایی دارد به طور مثال :

self.data.x نشان دهنده contract storage field x هست که میتوانیم برای آن نام مستعاری هم در نظر بگیریم و به طور مثال در y آن را ذخیره کنیم.

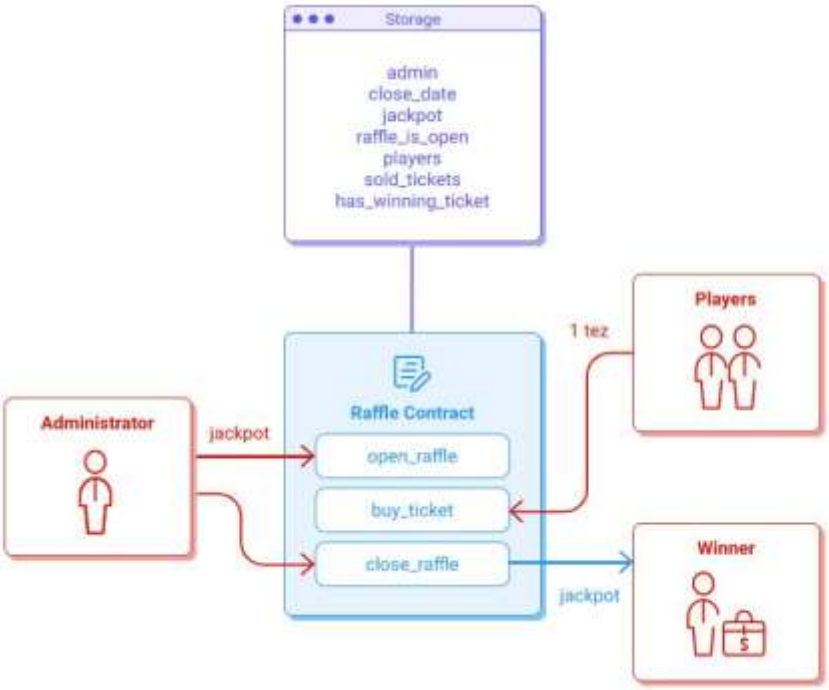
همچنین توابع آن هم با پریفیکس .sp فراخوانی میشوند مثلا :

sp.verify (self.data.x > 2) که چک میکند فیلد x بزرگتر از 2 هست یا نه و اگر برقرار نباشد اروری برگرداند. که این در زمان اجرا انجام میشود یعنی در بلاک چین پس از ترجمه به زبان Michelson(میدانیم Tezos Smart Contract به زبان Michelson نوشته شده است).

توجه شود که برای استفاده از دستوراتی مانند دستورات کنترلی ، بایستی قبل از آن عبارت .sp را بنویسیم برای مثال به جای if از sp.if استفاده کنیم که بتواند به Michelson کامپایل شود.

ساختار لاتاری (Raffle)

لاتاری به نوعی یک بازی شانسی هست که جایزه برنده را توزیع میکند. سازمان دهنده آن ، وظیفه تعیین jackpot را دارد که منظور از jackpot جوایز روی هم انباشته شده و به نوعی برنده تمام پولها است. همچنین وظیفه فروش بلیط ها را دارد. در این مثال فرض شده که تنها یک بلیط برنده خواهیم داشت.



تصویر 34. شمایی از ساختار لاتاری

Open_ruffle : فقط admin میتواند یک لاتاری جدید ایجاد کند. و مبلغ jackpot را به قرارداد ارسال میکند. تاریخ پایان مهلت خرید را ثبت میکند. شماره و هویت بلیط برنده را به صورت رمزگذاری شده نشان میدهد.

Buy_ticket : به هر کسی اجازه داده میشود که یک بلیط به قیمت 1 tez (XTZ) بخرد و در قرعه کشی شرکت کند.

Close_ruffle : تنها admin میتواند لاتاری را ببندد و jackpot را برای برنده میفرستد.

توجه شود در این مثال مقدار jackpot توسط مدیر تنظیم میشود اما میتوان مکانیزم های دیگر هم به کار گرفت مثلا مقدار jackpot با توجه به تعداد بلیط های فروخته شده تنظیم شود.

شروع پیاده سازی در ادیتور آنلاین SmartPy :

در ابتدا contract خود را ایجاد میکنیم. قرارداد SmartPy یک تعریف کلاسی است که از sp.contract ارث میبرد.

```
class Raffle(sp.Contract):
```

تصویر 35. کلاس Raffle

حافظه SmartPy در سازنده __init__ تعریف میشود که self.init در آن فراخوانی میشود و در آن مقداردهی اولیه به فیلدها داده میشود و فضای ذخیره سازی تنظیم میگردد.

Admin : تنها آدرس مجاز و احراز هویت شده ای است که میتواند لاتاری را ایجاد کند و در نهایت هم ان را ببندد.

Close_date : یک timestamp ای است که تاریخ پایان قرعه کشی (لاتاری) را معین میکند.

Jackpot : مبلغی است که برای برنده توزیع میشود.

Raffle_is_open : یک بولین هست که برای نشان دادن باز بودن یا نبودن لاتاری استفاده میشود.

Hash_wining_ticket : hash بلیط برنده است که توسط ادمین مشخص میشود.

```
def __init__(self, address):
    self.init(admin=address,
              close_date=sp.timestamp(0),
              jackpot=sp.tez(0),
              raffle_is_open=False,
              players=sp.set(),
              sold_tickets=sp.map(),
              hash_winning_ticket=sp.bytes('0x')
              )
```

تصویر 36. تابع init

در کل 3 تا entry point داریم که مربوط به باز کردن ، خرید بلیط و بستن لاتاری میباشد. به شرح هر کدام میپردازیم.

تعریف entry point برای بازکردن لاتاری :

نقاط ورودی ، یک متد از کلاس contract هست. اولین نقطه ورودی که بایستی ایجاد کنیم ، open_raffle هست که کارهای زیر در آن انجام میشود :

1. با sp.verify() یا sp.verify_equal() بررسی میکنیم که عبارت داخل آن درست هست یا نه. که ما در اینجا 4 شرط را بررسی میکنیم: 1- آدرسی که entry point را فراخوانی میکند باید همان آدرس ادمین باشد که در حافظه قرار داده شده است. sp.source و self.data.admin را با هم مقایسه میکنیم. sp.sender هم آدرسی است که نقطه ورودی فعلی را فراخوانی میکند. sp.source هم آدرسی است که تراکنش جاری را آغاز میکند که در اینجا فرض شده که با sp.sender مقدار برابری داشته باشد.
2. هیچ لاتاری ای باز نباشد که از متغیر بولین Raffle_is_open که در حافظه تعریف شده استفاده میکنیم.
3. مقدار sp.amount ارسال شده به قرارداد توسط ادمین در طول تراکنش باید حداقل از مقدار مشخص شده در آرگومان jackpot_amount بیشتر باشد.
4. تاریخ بسته شدن لاتاری طبق فرضمان مثلا باید حداقل 7 روز بعد از ایجاد شدن آن باشد.
5. پس از بررسی تمام شرایط ، ذخیره سازی را بروز میکنیم.

```
@sp.entry_point
def open_raffle(self, jackpot_amount, close_date, hash_winning_ticket):
    sp.verify_equal(sp.source, self.data.admin, message="Administrator not recognized.")
    sp.verify(~ self.data.raffle_is_open, message="A raffle is already open.")
    sp.verify(sp.amount >= jackpot_amount, message="The administrator does not own enough tz.")
    today = sp.now
    in_7_day = today.add_days(7)
    sp.verify(close_date > in_7_day, message="The raffle must remain open for at least 7 days.")
    self.data.close_date = close_date
    self.data.jackpot = jackpot_amount
    self.data.hash_winning_ticket = hash_winning_ticket
    self.data.raffle_is_open = True
```

تصویر 37. الگوریتم بروز کردن ذخیره سازی

تعریف entry point برای خرید بلیط در لاتاری

نقطه ورودی ای است که همه کسانیکه میخواهند در لاتاری شرکت کنند میتوانند آن را فراخوانی کنند و در صورت موفقیت آمیز بودن ، آدرس فرستنده به فضای ذخیره سازی اضافه میشود و آن بازیکن شانس برنده شدن را خواهد داشت. با اضافه شدن این نقطه ورودی دو فیلد جدید در فضای ذخیره سازی تعریف میکنیم :

Players : مجموعه ای است که آدرس هر بازیکن جدید که بلیط را خریده است دریافت میکند. از sp.Tset() استفاده میکند.

Sold_tickets : آدرس هر بازیکن را با یک شماره بلیط مرتبط میکند. این کار با استفاده از map صورت میگیرد که یک مقدار را به یک کلید مرتبط میکند. از sp.map() استفاده میکند.

برای اینکه این نقطه ورودی کار کند باید 3 شرط بررسی شود :

1. لاتاری باز باشد.
2. مبلغ ارسالی به قرارداد در حین تراکنش باید برابر با قیمت بلیط یعنی 1 tez باشد.
3. هر بازیکن فقط مجاز به خرید یک بلیط است.

اگر شرایط برقرار باشد ذخیره سازی را بروز میکنیم :

- آدرس بازیکن به مجموعه self.data.players اضافه میشود.
- شناسه بلیط (id) با آدرس بازیکن مرتبط است (از طریق map) در self.data.sold_tickets. شناسه بلیط برای هر بازیکن جدید افزایش می یابد. تابع abs که قدر مطلق است برای اطمینان هست از اینکه ticket_id از تایپ sp.TNat هست.

Ticket_id = abs (sp.len (self.data.players) – 1)

```
@sp.entry_point
def buy_ticket(self):
    ticket_price = sp.tez(1)
    current_player = sp.sender
    sp.verify(self.data.raffle_is_open, message="The raffle is closed.")
    sp.verify(sp.amount == ticket_price,
              message="The sender did not send the right tez amount (Ticket price = 1tz).")
    sp.verify(~ self.data.players.contains(current_player), message="Each player can participate only once.")
    self.data.players.add(current_player)
    ticket_id = abs(sp.len(self.data.players) - 1)
    self.data.sold_tickets[ticket_id] = current_player
```

تصویر 38. تابع خرید بلیط

تعریف entry point برای بستن لاتاری

فقط مدیر میتواند لاتاری را ببندد. در صورت موفقیت آمیز بودن فراخوانی ، لاتاری بسته میشود و مبلغ jackpot برای برنده ارسال میشود و فضای ذخیره سازی هم به حالت پیش فرض برمیگردد.

```
@sp.entry_point
def close_raffle(self, selected_ticket):
    sp.verify_equal(sp.source, self.data.admin, message="Administrator not recognized.")
    sp.verify(self.data.raffle_is_open, message="The raffle is closed.")
    sp.verify(sp.now >= self.data.close_date,
              message="The raffle must remain open for at least 7 days.")
    bytes_selected_ticket = sp.pack(selected_ticket)
    hash_selected_ticket = sp.sha256(bytes_selected_ticket)
    sp.verify_equal(hash_selected_ticket, self.data.hash_winning_ticket,
                    message="The hash does not match the hash of the winning ticket")
    number_of_players = sp.len(self.data.players)
    selected_ticket_id = selected_ticket % number_of_players
    winner = self.data.sold_tickets[selected_ticket_id]
    sp.send(winner, self.data.jackpot, message="winner contract not found.")
    self.data.jackpot = sp.tez(0)
    self.data.close_date = sp.timestamp(0)
    self.data.players = sp.set()
    self.data.sold_tickets = sp.map()
    self.data.raffle_is_open = False
```

تصویر 39. تابع close_reffle

حال پس از تعریف entry point ها و آماده سازی مقدمات میتوانیم سناریوهایی طراحی کنیم و عملیات بازکردن لاتاری ، خرید بلیط و بستن لاتاری را انجام دهیم. در ابتدا ادمین و کاربرانی تعریف میکنیم. و با توجه به بررسی های انجام شده ، یا ارور برمیگرداند و یا عملیات انجام میشود. و خروجی را میتوانیم به همراه اطلاعات مشاهده کنیم همچنین کد تبدیل شده به Michelson هم میتوانیم مشاهده کنیم.

```
@sp.add_test(name="Raffle")
def test():
    alice = sp.test_account("Alice")
    jack = sp.test_account("Jack")
    admin = sp.test_account("Administrator")
    r = Raffle(admin.address)
    scenario = sp.test_scenario()
    scenario.h1("Raffle")
    scenario += r
```

تصویر 40. تابع test

نمونه هایی از سناریوها

در نمونه زیر میخواهیم باز کردن لاتاری توسط ادمین را تست کنیم. سناریوها به ترتیب به صورت زیر هستند :

- در ابتدا بازکردن لاتاری توسط کاربر را نشان میدهد که در این صورت ، ارور برمیگرداند.
- درسناریوی بعدی بازکردن لاتاری توسط ادمین را نشان میدهد در شرایطی که تاریخ بسته شدن آن را کمتر از 7 روز مثلا 4 روز قرار داده که در این شرایط هم به ارور برمیخوریم.
- ادمین لاتاری ای باز میکند که به اندازه کافی tez (XTZ) به قرارداد تزریق نمیکند.
- ادمین به صورت موفقیت آمیز لاتاری را ایجاد میکند.

- ادمین نمیتواند لاتاری دیگری باز کند چون در حال حاضر لاتاری ای در جریان و باز هست.

```
scenario.h2("Test open_raffle entrypoint")
close_date = sp.timestamp_from_utc_now().add_days(8)
jackpot_amount = sp.tez(10)
number_winning_ticket = sp.nat(345)
bytes_winning_ticket = sp.pack(number_winning_ticket)
hash_winning_ticket = sp.sha256(bytes_winning_ticket)

scenario.h3("The unauthorized user Alice unsuccessfully call open_raffle")
scenario += r.open_raffle(close_date=close_date, jackpot_amount=jackpot_amount,
                          hash_winning_ticket=hash_winning_ticket) \
    .run(source=alice.address, amount=sp.tez(10), now=sp.timestamp_from_utc_now(),
         valid=False)

scenario.h3("Admin unsuccessfully call open_raffle with wrong close_date")
close_date = sp.timestamp_from_utc_now().add_days(4)
scenario += r.open_raffle(close_date=close_date, jackpot_amount=jackpot_amount,
                          hash_winning_ticket=hash_winning_ticket) \
    .run(source=admin.address, amount=sp.tez(10), now=sp.timestamp_from_utc_now(),
         valid=False)
```

تصویر 41. باز کردن لاتاری توسط ادمین

```
scenario.h3("Admin unsuccessfully call open_raffle by sending not enough tez to the contract")
close_date = sp.timestamp_from_utc_now().add_days(8)
scenario += r.open_raffle(close_date=close_date, jackpot_amount=jackpot_amount,
                          hash_winning_ticket=hash_winning_ticket) \
    .run(source=admin.address, amount=sp.tez(5), now=sp.timestamp_from_utc_now(),
         valid=False)

scenario.h3("Admin successfully call open_raffle")
scenario += r.open_raffle(close_date=close_date, jackpot_amount=jackpot_amount,
                          hash_winning_ticket=hash_winning_ticket) \
    .run(source=admin.address, amount=sp.tez(10), now=sp.timestamp_from_utc_now())
scenario.verify(r.data.close_date == close_date)
scenario.verify(r.data.jackpot == jackpot_amount)
scenario.verify(r.data.raffle_is_open)

scenario.h3("Admin unsuccessfully call open_raffle because a raffle is already open")
scenario += r.open_raffle(close_date=close_date, jackpot_amount=jackpot_amount,
                          hash_winning_ticket=hash_winning_ticket) \
    .run(source=admin.address, amount=sp.tez(10), now=sp.timestamp_from_utc_now(),
         valid=False)
```

تصویر 42. تست موفقیت آمیز بودن باز کردن لاتاری توسط ادمین

در نمونه بعدی میخواهیم خرید بلیط را تست کنیم. سناریوها به ترتیب به صورت زیر هستند :

- کاربری به نام Alice نمیتواند بلیط بخرد و به ارور میخورد زیرا مقدار tez نامناسبی فرستاده است. زیرا فرض شده که برای خرید بلیط ، 1 tez نیاز است و نه بیشتر یا کمتر.
- خرید موفقیت آمیز بلیط توسط کاربر Alice
- خرید مجدد توسط کاربری که بلیط آن لاتاری را قبلا خریده یعنی Alice. زیرا گفتیم هر کاربر در هر لاتاری فقط میتواند یک بلیط خریداری کند.
- خرید موفقیت آمیز بلیط توسط کاربر Jack

```
scenario.h2("Test buy_ticket entrypoint (at this point a raffle is open)")

scenario.h3("Alice unsuccessfully call buy_ticket by sending a wrong amount of tez")
scenario += r.buy_ticket().run(sender=alice.address, amount=sp.tez(3), valid=False)

scenario.h3("Alice successfully call buy_ticket")
scenario += r.buy_ticket().run(sender=alice.address, amount=sp.tez(1))
alice_ticket_id = sp.nat(0)
scenario.verify(r.data.players.contains(alice.address))
scenario.verify_equal(r.data.sold_tickets[alice_ticket_id], alice.address)

scenario.h3("Alice unsuccessfully call buy_ticket because she has already buy one")
scenario += r.buy_ticket().run(sender=alice.address, amount=sp.tez(1), valid=False)

scenario.h3("Jack successfully call buy_ticket")
scenario += r.buy_ticket().run(sender=jack.address, amount=sp.tez(1))
jack_ticket_id = sp.nat(1)
scenario.verify(r.data.players.contains(jack.address))
scenario.verify(r.data.players.contains(alice.address))
scenario.verify_equal(r.data.sold_tickets[alice_ticket_id], alice.address)
scenario.verify_equal(r.data.sold_tickets[jack_ticket_id], jack.address)
```

تصویر 43. سناریو خرید بلیط

در نمونه پایانی هم میخواهیم بستن لاتاری را تست کنیم. سناریوها به ترتیب به صورت زیر هستند :

- کاربر Alice لاتاری را ببندد که چون ادمین نیست به ارور برمیخورد.
- ادمین لاتاری را میبندد اما چونکه همچنان تا تاریخ پایان لاتاری و close_date زمان باقی مانده به او ارور میدهد.
- ادمین در زمان مناسب لاتاری را میخواهد ببندد اما این بار مشکل این است که hash بلیط انتخابی با بلیط فرد برنده مطابقت ندارد.
- به طور موفقیت آمیز توسط ادمین ، لاتاری بسته میشود.
- ادمین دوباره میخواهد لاتاری را ببندد اما چون لاتاری ای ، دیگر موجود نیست به ارور میخورد.

```
scenario.h2("Test close_raffle entrypoint (at this point a raffle is open and two players participated)")
selected_ticket = sp.nat(345)

scenario.h3("The unauthorized user Alice unsuccessfully call close_raffle")
scenario += r.close_raffle(selected_ticket).run(sender=alice.address, valid=False)

scenario.h3("Admin unsuccessfully call close_raffle because it was before the close_date")
scenario += r.close_raffle(selected_ticket)\
    .run(sender=admin.address, now=sp.timestamp_from_utc_now(), valid=False)

scenario.h3("Admin unsuccessfully call close_raffle because the hash of the selected ticket does not match with the winning one")
selected_ticket = sp.nat(1234)
scenario += r.close_raffle(selected_ticket)\
    .run(sender=admin.address, now=r.data.close_date, valid=False)

scenario.h3("Admin successfully call close_raffle")
selected_ticket = sp.nat(345)
scenario += r.close_raffle(selected_ticket).run(sender=admin.address, now=r.data.close_date)
scenario.verify_equal(r.data.jackpot, sp.tez(0))
scenario.verify_equal(r.data.close_date, sp.timestamp(0))
scenario.verify_equal(r.data.players, sp.set())
scenario.verify_equal(r.data.sold_tickets, sp.map())
scenario.verify(~ r.data.raffle_is_open)

scenario.h3("Alice unsuccessfully call buy_ticket because the raffle is closed")
scenario += r.buy_ticket().run(sender=alice.address, amount=sp.tez(1), valid=False)
```

تصویر 44. سناریو بستن لاتاری توسط ادمین

نتیجه گیری

قرارداد هوشمند یکی از فناوری های جدید است که با نوع سنتی و کاغذی قراردادها متفاوت است و سعی بر تسهیل کارها و حذف افراد واسطه دارد. باید توجه داشت به محض اجرای یک قرارداد هوشمند ، امکان تغییر و دستکاری شرایط حتی توسط نویسنده آن وجود ندارد و همه تراکنش ها ثبت و ذخیره شده اند. با گذر زمان اسمارت کانترکت ها نقش پررنگ تری در زندگی ما پیدا کرده اند و در حوزه های سلامت ، بیمه ، رای گیری ، عرضه اولیه سکه ، زنجیره تامین و کسب و کارها مزایا و کاربردهای فراوانی دارد. پدیده قرارداد هوشمند در ابتدای راهش قرار گرفته و به اصطلاح نو پا هست و اگر فضا برای گسترش قرارداد هوشمند مهیا شود و چالش های آن رفع شود قطعا میتواند انقلابی در زندگی ما ایجاد کند و آینده درخشانی میتواند داشته باشد و انجام بسیاری از امور را ساده و کم هزینه میکند.

مراجع

[1]- Wang, Shuai, et al. "Blockchain-enabled smart contracts: architecture, applications, and future trends." IEEE Transactions on Systems, Man, and Cybernetics: Systems 49.11 (2019): 2266-2277.

[2]- Watanabe, Hiroki, et al. "Blockchain contract: Securing a blockchain applied to smart contracts." 2016 IEEE international conference on consumer electronics (ICCE). IEEE, 2016.

[3]- Gatteschi, Valentina, et al. "Blockchain and smart contracts for insurance: Is the technology mature enough?." Future internet 10.2 (2018): 20.

[4]- E. Viglianisi, M. Ceccato and P. Tonella, "Summary of: A Federated Society of Bots for Smart Contract Testing," 2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST), 2021, pp. 282-283.

[5]- J. M. Montes, C. E. Ramirez, M. C. Gutierrez and V. M. Larios, "Smart Contracts for supply chain applicable to Smart Cities daily operations," 2019 IEEE International Smart Cities Conference (ISC2), 2019, pp. 565-570.

[6]- J. Lindsay, "Smart Contracts for Incentivizing Sensor Based Mobile Smart City Applications," 2018 IEEE International Smart Cities Conference (ISC2), 2018, pp. 1-4.

[7]- Y. Zhang, M. Yutaka, M. Sasabe and S. Kasahara, "Attribute-Based Access Control for Smart Cities: A Smart-Contract-Driven Framework," in IEEE Internet of Things Journal, vol. 8, no. 8, pp. 6372-6384, 15 April15, 2021.

[8]- Wang, Shangping, et al. "Smart contract-based product traceability system in the supply chain scenario." *IEEE Access* 7 (2019): 115122-115133.

[9]- Ullah, Fahim, and Fadi Al-Turjman. "A conceptual framework for blockchain smart contract adoption to manage real estate deals in smart cities." Neural Computing and Applications (2021): 1-22.

[10]- https://smartpy.io/

[11]- http://opentezos.com/

[12]- https://intellias.com/how-to-make-a-smart-contract-work-for-the-insurance-industry/