Studio tecnologico Identity Trust Fabric

Autore: Simone Ballarin

Data: 12/06/18

Destinatari: Athesys

Diario delle modifiche

Data	Descrizione	Autore
12/06/18	Creazione documento. Scrittura dei capitoli Descrizione prodotto, scopo del documento e ITF, Permissionless e Permissioned blockchain, Ethereum. Stesura capitoli Valutazione applicabilità soluzione Ethereum e sotto capitoli interni.	Simone Ballarin
14/06/18	Stesura dei capitoli Programmare SmartContract, Considerazioni su comunicazione mobile e Conclusioni. Verifica.	Simone Ballarin
15/06/18	Piccola aggiunta al capitolo Considerazioni su comunicazione mobile riguardo al componente REST visto come single point of failure (SPF) della rete.	Simone Ballarin

Scopo del documento

Nell'ottica di un prossimo sviluppo di un'estensione basata su blockchain dell'attuale servizio ora in produzione MonoKee, questo documento intende valutare in termini di benefici tecnici e tecnologici l'uso della tecnologia Ethereum quale base dell'Identity Trust Fabric (ITF).

Sintesi del documento

Il documento procede descrivendo le caratteristiche del prodotto MonoKee e di come la tecnologia Ethereum si possa collocare in un tale contesto. Vengono poi trattati i principali strumenti e librerie disponibili per sviluppare in Ethereum. L'analisi conclude facendo emergere come un utilizzo di Ethereum sia possibile, ma comunque non consigliato a causa di questioni prettamente legate alla scalabilità del sistema.

Riferimenti informativi

1. Blockchain: The Dawn of Decentralized Identity (G00303143), Homan Farahmand per Gartner
2. Sito ufficiale Ethereum, www.ethereum.org
3. bornonjuly4.me/2017/01/10/blockchain-what-is-permissioned-vs-permissionless
4. www.cryptoitalia.org/bitcoin-il-51-attack
5. Documentazione ufficiale solidity, www.solidity.readthedocs.io
6. www.stateofthedapps.com

Descrizione prodotto

Il progetto ha come scopo la creazione di un'estensione del servizio MonoKee basato su blockchain. L'estensione offre un sistema di Identity Access Management (IAM) composto da quattro principali fattori:

1. Identity Wallet (IW)
2. Service Provider (SP)
3. Identity Trust Fabric (ITF)
4. Trusted Third Party (TTP)

In sintesi l'estensione dovrà operare al fine di fornire la possibilità ad un utente di registrare e gestire la propria identità autonomamente tramite l'IW, mandare i propri dati (IPP) all'ITF, il quale custodirà la sua identità e farà da garante per le asserzioni proveniente dai TTP. Inoltre il SP dovrà essere in grado con le informazioni provenienti da IW e ITF di verificare o meno l'accesso ai propri servizi.

ITF – Identity Trust Fabric

Sulla base di un primo studio di fattibilità l'unico componente coinvolto nell'uso blockchain è l'Identity Trust Fabric. La sua principale funzione è quella di poter permettere ai vari Service Provider aderenti al servizio di poter verificare le informazioni rilasciate dai vari utenti tramite l'utilizzo del loro personale Identity Wallet (IW). Il componente mantiene al suo interno l'hash della chiave pubblica degli utenti (che rappresenta la loro identità) e le asserzioni fornite dai vari IW che possono essere potenzialmente certificate da una TTP (tramite una firma con la loro chiave privata). Le asserzioni devono poter essere modificate o eliminate in ogni momento, ovviamente ogni alterazione deve essere ogni volta certificata nuovamente. Anche da parte del TTP ci dev'essere la possibilità di revocare la certificazione di un'asserzione.

Secondo lo studio Gartner in nota 1 una buona implementazione di una ITF deve possedere le seguenti caratteristiche:

1. **Fiducia**: il contenuto presente nella ITF deve essere solo quello autorizzato, non ci devono poter essere manomissioni malevoli da parte degli utilizzatori della rete. Ogni componente deve potere aver fiducia nella veridicità dei dati.
2. **Garanzia**: le regole logiche della ITF non devono poter essere manomesse. Deve essere possibile applicare le varie policy aziendali in ambito di gestione dei rischi.
3. **Tracciabilità**: ogni informazione e cambio di stato relativo alle identità e alle asserzioni deve poter essere tracciato e verificabile sia in termini cronologici sia in termini di provenienza.
4. **Sicurezza**: intesa come CIA. L'ITF deve rispettare i vincoli di confidenzialità, inalterabilità e disponibilità delle informazioni dentro lei contenute.
5. **Scalabilità**: l'ITF deve fornire un elevato grado di scalabilità soprattutto in un'ottica in cui il prodotto potrebbe essere reso disponibile ad un uso Consumer.
6. **Efficienza**: il funzionamento dell'ITF deve richiedere la minima quantità di risorse possibili.

Permissionless e Permissioned blockchain

Al fine di poter valutare la fattibilità dell'utilizzo di Ethereum quale blockchain sottostante all'ITF è necessario avere in mente le due principali categorie di blockchain: **permissionless** e **permissioned**.

Permissioned blockchain

Una permissioned blockchain pone dei vincoli sulla partecipazione alla rete. Soli i nodi autorizzati possono partecipare all'algoritmo di consenso dei blocchi. Le autorizzazioni possono essere date singolarmente, quindi i vari nodi possono avere o meno le seguenti possibilità:

- lettura dei blocchi;

- scrittura dei blocchi;
- esecuzione di codice (se prevista dalla blockchain);
- verifica dei nodi.

Permissionless blockchain

Una permissionless blockchain è una rete in cui qualsiasi nodo può partecipare al processo di verifica dei blocchi. Ogni nodo ha tutte le precedenti quattro proprietà.

Ethereum

Ethereum è una piattaforma decentralizzata pubblica ed open-source basata sulla creazione di SmartContract. Permette la creazione di applicazioni che operano su blockchain in modo che non ci sia alcuna possibilità di downtime, censura, frodi o interferenze da terze parti. Rappresenta una dei principali esempi di rete permissionless.

La piattaforma è stata rilasciata nel corso del 2014 ed è mantenuta dalla Ethereum Foundation. Questo fa di Ethereum una delle più longeve blockchain disponibili. Ciò comporta la presenza di una documentazione abbastanza nutrita rispetto ai competitor e di un buon numero di strumenti già disponibili.

L'elevata popolarità della tecnologia e alcune sue caratteristiche non presenti nei competitor, ha fatto sì che una notevole quantità di sviluppatori abbiano deciso di utilizzarla. Il sito www.stateofthedapps.com mantiene una vetrina di oltre milleseicento esempi. Sono presenti tutte le più significative applicazioni ora in produzione. Si fa notare che molte delle quali sono state le fonti dei più diffusi pattern Ethereum.

Programmare SmartContract

Solidity è il principale linguaggio di programmazione usato per scrivere SmartContract. Nonostante sia presente un'implementazione basata su Go, questa è ancora acerba e non largamente utilizzata. Per questo motivo questa implementazione non verrà trattata in questo documento.

Solidity è un linguaggio di programmazione ad oggetti ad alto livello. Il suo sviluppo è stato fortemente influenzato da linguaggi quali C++, Python e Javascript. Gli SmartContract così scritti vengono poi trasformati in bytecode e quest'ultimo viene eseguito dall'Ethereum Virtual Machina (EVM).

Il linguaggio seppur non completamente maturo offre la maggior parte delle caratteristiche tipiche di un linguaggio ad oggetti. Infatti Solidity è fortemente tipato, supporta l'ereditarietà, librerie esterne e tipi definiti dall'utente. A sottolineare la bontà del linguaggio si evidenzia come in Solidity sia presente il concetto di interfaccia, caratteristica non presente in linguaggi ben più longevi.

Queste caratteristiche rappresentano un notevole vantaggio per Ethereum rispetto ai diretti competitor, i quali spesso utilizzano linguaggi acerbi e/o a basso livello. Si ritiene che un linguaggio con le precedentemente descritte caratteristiche sia fondamentale per la buona riuscita del progetto, soprattutto in un'ottica di manutenibilità e estendibilità.

Breve nota sull'applicabilità dei pattern

Nonostante il linguaggio permetta l'applicabilità dei più diffusi pattern si vuole far notare come nel contesto di una blockchain Permissionless questi risultino spesso controproducenti.

Durante la progettazione e l'applicazione dei pattern vanno sempre ricordati i seguenti punti:

- l'esecuzione di un metodo che modifica la blockchain si paga in base al lavoro che viene effettivamente svolto;
- complessità lineari portano a costi difficilmente accettabili;
- plugin come Metamask calcolano il massimo costo possibile di una transazione, in caso il credito non sia sufficiente la transazione fallisce. Quindi un ciclo for su una lista di un elemento viene stimato presupponendo che la lista sia completamente piena;
- la velocità di esecuzione varia in base alla somma pagata per questa, anche con somme estremamente alte o su reti locali i tempi potrebbero essere considerati non giustificabili per la maggior parte degli utenti;
- ogni oggetto e campo dato si paga in base al loro spazio occupato;
- il costo della moneta e quindi delle transazioni è fortemente variabile. Approcci un giorno economici possono diventare economicamente insostenibili a distanza di giorni.

A seguito dei precedenti punti dovrebbe risultare più evidente come pattern che prevedono alta complessità temporale e spaziale siano inaccettabili su una rete Ethereum. Ad esempio i pattern Command e Decorator risultano difficilmente giustificabili.

Sono invece presenti pattern pensati appositamente per Ethereum. Particolarmente utili al contesto del progetto in esame ritengo possano utili i seguenti pattern: Owner Pattern, Vote Pattern e WhiteList Pattern, i quali sono presenti nella documentazione ufficiale Solidity.

Complessità e pratiche non convenzionali

I punti precedentemente stilati nel paragrafo sull'applicabilità dei pattern portano anche delle notevoli differenze in termini di pratiche di stile di programmazione. Tra queste riporto:

- l'uso di liste e array sono fortemente sconsigliato, vanno preferite struttura con accesso costante. Solidity fornisce il tipo mapping;
- la creazione di oggetti (in termini Solidity contratti) ha un costo notevole. Una buona pratica è quella di utilizzare ADT (Abstract Data Type) differenti come le strutture;
- cicli for che portano complessità lineare dovrebbero essere evitati, elaborazioni di questo tipo dovrebbero essere affidate a server esterni o a livello client-side;
- l'utilizzo dei puntatori (in Solidity address) nasconde completamente il tipo dell'oggetto puntato rendendo vano il controllo dei tipi. Andrebbe evitato il più possibile.

Si fa notare come in particolare l'ultimo punto degeneri completamente il concetto di programmazione ad alto livello.

Strumenti

Come già citato la relativa maturità della tecnologia ha portato alla creazione di alcuni utili strumenti.

Truffle: è una suite di development e testing. Permette di compilare, buildare ed effettuare la migrazione degli SmartContract. Inoltre ha funzioni di debugging e di scripting. La suite offre la possibilità di effettuare test degli SmartContract sia in Javascript (con l'utilizzo di Chai), sia in Solidity. Si riporta di seguito il sito del progetto: www.truffleframework.com .

Ganache: è uno strumento rapido che permette di creare e mantenere in locale una rete blockchain Ethereum personale. Può essere usata per eseguire test, eseguire comandi e per operazioni di controllo

dello stato mentre il codice esegue. Si riporta di seguito il sito del progetto: www.truffleframework.com/ganache .

Mist: è un browser sviluppato direttamente dal team Ethereum in grado di operare transazioni direttamente nella blockchain senza la necessità di possedere un intero nodo. È estremamente immaturo e non utilizzabile in produzione. Si riporta di seguito il sito del progetto: github.com/ethereum/mist.

Parity: è un client Ethereum che permette di operare sulla rete senza necessità di possedere un intero nodo. Questa soluzione a differenza di Mist dovrebbe risultare più facilmente integrabile nel prodotto senza che l'utente ne abbi consapevolezza. Si riporta di seguito il sito del progetto: www.wiki.parity.io .

Metamask: è uno plugin disponibile per i browser Chrome Firefox Opera che permette di interfacciarsi alla rete Ethereum senza la necessità di eseguire in intero nodo della rete. Il plugin include un wallet con cui l'utente può inserire il proprio account tramite la chiave privata. Una volta inserito l'account il plugin farà da tramite tra l'applicazione e la rete.

Metamask è utilizzato dalla maggioranza delle applicazioni Ethereum presenti on line, questo però rappresenterebbe un componente esterno compatibile con pochi browser desktop. Si riporta di seguito il sito del progetto: www.metamask.io .

Status: è un progetto che propone una serie di API che permettono di sviluppare un'applicazione mobile nativa operante direttamente su blockchain senza la necessità di possedere un intero nodo. Il sito del progetto propone una serie di applicazioni che utilizzano Status tuttavia nessuna di queste applicazioni risulta attualmente rilasciate in nessuno store. Status risulta in early access ed è disponibile per Android e iOS. Il sito del progetto è il seguente: www.status.im .

Valutazione applicabilità soluzione Ethereum

Al fine di poter valutare correttamente da ogni punto di vista l'applicabilità di una soluzione basata su Ethereum quale base della componente ITF, si procede ad analizzare in maniera analitica le sei caratteristiche presentate nel capitolo 'ITF – Identity Trust Fabric'.

Fiducia

Questa caratteristica è ottenuta da Ethereum da una combinazione di diversi fattori quali:

- utilizzo di incentivi economici, il pagamento per effettuare operazioni
- utilizzo di prove di interesse (Proof of Interest)

Le prove di interesse possono essere di due tipi:

- Proof of Stake, l'esibizione di un interesse;
- Proof of Work, l'uso di potenza di calcolo per risolvere un problema matematico.

Queste metodologie fanno in modo che solo chi realmente interessato possa influenzare l'algoritmo di consenso dei blocchi. Questo rende minore la possibilità di un 51% attack (nota 4). C'è comunque da ricordare che un attacco di questo tipo e praticamente impossibile.

Per queste ragioni si ritiene una rete Ethereum sia completamente soddisfacente per quanto riguarda l'aspetto fiducia, al pari di una rete di tipo permissioned.

Garanzia

Lo studio in nota 1 evidenzia come questo rappresenti un punto critico. Infatti riporta che il raggiungimento di questo obiettivo è fortemente condizionata dall'efficacia dell'algoritmo di consenso e dai nodi presenti nella rete. Lo studio prosegue facendo notare che la presenza di nodi malevoli, oltre che mettere a rischio l'algoritmo di consenso, può compromettere anche il corretto funzionamento dell'ITF. Infatti trattandosi di una blockchain pubblica ogni nodo è in grado di visionare il contenuto di ogni singolo contratto, inclusi i dati e i metodi presenti. Per quanto riguarda i dati questo potrebbe non essere un problema in quanto si può immagazzinare una versione codificata del dato. Per quanto riguarda i metodi questo non è possibile, questo potrebbe rendere in grado ad un attaccante di trovare eventuali bachi e criticità dell'ITF.

Tracciabilità

Lo studio Gartner evidenzia come in una rete permissionless la tracciabilità temporale non sia possibile, in quanto in una rete distribuita ogni nodo può avere un concetto di tempo proprio. Questo però non risulta possibile in nessun approccia risolutivo all'ITF basato su blockchain. Infatti le reti permissioned applicano timestamp a livello di blocco e non di transazione, anche ammettendo che ci sia un concetto di tempo comune tra i nodi, le transizioni rimarrebbero temporalmente non tracciabili. La cosa potrebbe permettere ad un blocco di alterare l'ordine delle transazioni.

Questo problema in una rete permissioned può essere risolto creando blocchi immutabili e ogni volta si voglia fare una modifica si dovrà creare un nuovo blocco. In questo modo ci sarà solo una transazione di creazione blocco il cui timestamp coinciderà con il timestamp del blocco.

Questo approccio in Ethereum rimane comunque impraticabile. Attualmente non sono note ulteriori tecniche per la tracciabilità temporale in Ethereum. Per questa ragione l'attribuzione di un riferimento temporale dovrà essere effettuato lato client, con i conseguenti limiti di sicurezza.

Security

La confidenzialità dei dati anche se non presente nativamente in Ethereum e facilmente ottenibile immagazzinando nei contratti solo un hash dei dati.

L'integrità dei dati invece è garantita dalla prova di lavoro che utilizza la blockchain come già ribadito nella sezione Fiducia.

La disponibilità invece è garantita dalle caratteristiche di distribuzione di ogni blockchain.

Un ulteriore punto di considerazione da fare è che chiunque ha la possibilità di vedere il contenuto di ogni SmartContract incluso il codice dei metodi. Questo come già detto può comportare la possibilità da parte di un attaccante di individuare eventuali errori logici. Ogni contratto dovrà comunicare con gli altri attraverso chiamate a metodi pubblici, in quanto non c'è in Ethereum nessun concetto di visibilità dei metodi di tipo protected o package. Questo rende possibile da parte di qualsiasi utente della rete di utilizzare questi metodi in maniera malevole. Questo tipo di problematica è facilmente superabile applicando i dovuti pattern Solidity quali WhiteList Pattern e Owner Pattern. L'applicazione dei pattern però comporterebbe un notevole aumento in termini di complessità e costo soprattutto in presenza di logiche di accesso variegate e dinamiche. Inoltre, in caso di liste di utenti autorizzati l'immaganizzazione di queste liste potrebbe risultare oneroso in termini di costi.

Scalabilità

Ethereum per poter applicare l'algoritmo del consenso fa utilizzo di una prova di lavoro. Questa deve essere fatta in occasione di ogni transazione. La prova consiste nella risoluzione di un problema crittografico la cui difficoltà è dinamica in base a diversi fattori della blockchain quali valore dell'Ether, numero di utenti, numero di transazioni, etc. Inoltre si nota come anche in lettura ci sia una lentezza che difficilmente potrebbe essere ritenuta accettabile da un utente medio. Per avere prova di questo fatto si può prendere in esame una qualsiasi Dapp presente al seguente link www.stateofthedapps.com. La questione pone anche limitazioni come già citato in termine di costo.

Considerazione su utilizzo mobile

L'estensione di MonoKee presenta un componente chiamato IW il quale comunica con l'ITF. Da un primo studio di fattibilità emerge che l'IW verrà sviluppato come applicazione mobile. Questo potrebbe rappresentare un problema in termini di Fiducia, infatti la precedente analisi sugli strumenti evidenzia come l'unica soluzione attualmente presente per operare su blockchain da mobile sia Status. Status attualmente però non sembrerebbe rappresentare una soluzione utilizzabile in quanto troppo acerba e poco utilizzata. L'unica opzione rimanente risulta quella di gestire le comunicazioni tra ITF e IW tramite API REST. Quest'ultima soluzione renderebbe l'IW totalmente incapace di verificare i dati provenienti dall'ITF rendendo così assente l'aspetto Fiducia. Tra gli altri componenti questo non rappresenta un problema in quanto applicazioni server o desktop. Oltre al problema di fiducia si potrebbe avanzare un appunto relativo all'implementazione del componente REST, infatti questo potrebbe rappresentare un componente centralizzato su un applicativo altresì completamente distribuito. Questo potrebbe rappresentare un single point of failure e compromettere la disponibilità assoluta che invece garantirebbe l'uso di Ethereum.

Per concludere questa considerazione si vuole fare notare come questo sia un problema generale di ogni tecnologia blockchain e non solo di Ethereum. Anzi Ethereum nonostante non abbia ancora una soluzione pronta rappresenta la tecnologia più vicina ad averne una. Una comunicazione basata su API REST potrebbe essere comunque usata come soluzione provvisoria in attesa di una maggiore stabilità di Status.

Conclusioni

Da quanto è emerso l'utilizzo della tecnologia Ethereum quale base dell'ITF pone una serie di vantaggi e svantaggi. Di seguito si propone una sintetica trattazione dei punti fondamentali, per maggiori dettagli si consiglia la lettura dell'intero documento.

I vantaggi sono:

- Ethereum offre un linguaggio ad alto livello e ad oggetti a differenza di altri competitor;
- Ethereum offre una notevole maturità e anche un'ampia platea di strumenti, molti dei quali estremamente maturi e largamente utilizzati.

Gli svantaggi sono:

- Ethereum è una rete pubblica, non è possibile fare nessuna restrizione di privilegi sui nodi partecipanti alla rete. Questo potrebbe rappresentare un problema di sicurezza;
- la comunicazione verso dispositivi mobili non è verificabile da quest'ultimi, in quanto dovrebbe avvenire tramite comunicazione REST;

- sono presenti forte limitazioni in termini di costo e velocità, il sistema risulterebbe lento ed estremamente costoso. Ciò comporta notevoli problemi in termini di Scalabilità del servizio.

Si ritiene che un approccio basato sul Ethereum sull'ITF sia possibile, le eventuali criticità di sicurezza e fiducia verso i dispositivi mobili sono superabili con una buona progettazione. L'unico fattore veramente critico risulta la scalabilità del sistema. Quest'ultimo fatto a mio parere è sufficiente per ritenere Ethereum non adatto all'utilizzo soprattutto in un'ottica commerciale. Quindi se pure possibile, non si consiglia l'utilizzo di Ethereum.