Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica per il Management

METODOLOGIE PER LA REALIZZAZIONE DI UNA SECURITY TOKEN OFFERING

Relatore:
Chiar.mo Prof.
STEFANO FERRETTI

Presentata da:
GIACOMO MINELLO

II Sessione
Anno Accademico 2018/2019

Introduzione

Le *Security Token Offerings*, abbreviate in *STOs*, sono un fenomeno recente che si è diffuso a partire dalla seconda metà del 2017 mantenendo inizialmente la connotazione di *Initial Coin Offerings (ICOs)*, per poi prestare maggiore attenzione alla regolamentazione e differenziarsi in *token sales* in cui il token è uno strumento finanziario regolamentato.

Come si avrà modo di osservare, questo cambio di paradigma è ciò che contraddistingue le STOs. Nel 2017 le ICOs hanno raggiunto un picco di popolarità per poi la maggior parte fallire in meno di un anno, facendo capire agli investitori che le ICOs sono state una bolla speculativa. Nonostante ciò, la validità del modello di raccolta di capitale tramite la vendita di token basati su tecnologia Blockchain non è stata messa in discussione. Proprio per questo sono nate le STOs, delle token sales in cui il token è uno strumento finanziario, che offre tutela agli investitori.

Lo scopo di questo lavoro di tesi è stato approfondire la comprensione di questo fenomeno in particolare analizzandone le motivazioni, le caratteristiche e le metodologie con le quali queste STOs vengono realizzate.

La tesi è strutturata come segue:

- Nel primo capitolo si ripercorre la storia delle monete digitali per capire le complessità del processo di diffusione di queste. Vengono analizzate nel dettaglio le fasi della creazione e diffusione sia di Bitcoin che di Ethereum. Una volta comprese le motivazioni che hanno portato alla nascita di queste monete ne verrà considerata un'applicazione particolare, la raccolta di capitale. Di conseguenza, vengono confron-

tate le caratteristiche di diverse tipologie di token, in modo da poterli identificare e distinguere tra loro.

- Nel secondo capitolo vengono presentati i protocolli principali con i quali le STOs vengono realizzate. Tra questi vengono prese in considerazione tre categorie principali. La prima categoria comprende lo standard ERC-20 e altri standard proprietari che estendono ERC-20. Nella seconda categoria vengono presentati gli standard dedicati a beni non fungibili, ovvero lo standard ERC-721 e il suo successore, lo standard ERC-1155. Infine, l'ultima categoria è lo standard ERC-1400 che comprende altri standard per i security tokens.

- Nel terzo capitolo si discute dei principali driver di innovazione nel campo delle STOs. In primo luogo vengono presentate le IEOs, dei particolari tipi di token sales che avvengono su exchange dedicati. Successivamente, si discute del ruolo di Libra, la criptovaluta proposta da Facebook, quale apripista per aprire un dialogo con i legislatori e stabilire un framework normativo chiaro.

- Nel quarto capitolo vengono presentati i dati ottenuti dopo una difficile ricerca. La difficoltà riscontrata è dovuta alla natura stessa delle STOs: le STOs si rivolgono esclusivamente ad investitori autorizzati, non ai consumatori. I dati presentati vengono poi analizzati per poter elaborare alcune considerazioni riguardo alle STOs.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Differenze tra Initial Coin Offering e Security Token Offering

1.1 Storia delle monete elettroniche e delle criptomonete

Sebbene le valute digitali abbiano raggiunto un livello di rilievo solo negli ultimi anni, è importante ricordare che la monete elettroniche hanno una storia che risale a decenni fa. Il whitepaper *"Bitcoin: A Peer-to-Peer Electronic Cash System"*[1], pubblicato nel 2008 da un autore anonimo o da un gruppo di persone sotto lo pseudonimo di Satoshi Nakamoto, per quanto risulti innovativo e a posteriori anche rivoluzionario, si basa su tecnologie preesistenti e ben affermate tra le quali ad esempio le reti P2P, un fenomeno che ha radici nelle prime fasi della storia di Internet. Non sono solo le reti distribuite ad essere una fondazione per Bitcoin; nel whitepaper, Nakamoto si avvale di altri lavori precedenti che risultano fondamentali al funzionamento della criptovaluta. Tra questi possiamo citare il lavoro di tesi pubblicato da Ralph Merkle nei primi anni '70 riguardo alla comunicazione sicura attraverso canali insicuri e gli sviluppi apportati da Diffie e Hellman[2, 3]. Inoltre, è proprio a Merkle che dobbiamo l'invenzione dei Merkle Tree usati in Bitcoin poiché forniscono un metodo per firmare digitalmente messaggi contenuti

in grandi strutture dati[4]. I Merkle Tree verranno successivamente ripresi da Bayer, Haber e Stornetta per la realizzazione efficiente di una catena di blocchi firmata crittograficamente[5].

Altri lavori più recenti risultano essere punti chiave per lo sviluppo della rete Bitcoin ed in generale dei sistemi di pagamento elettronici. Nel 1983 in *"Blind Signatures for Untraceable Payments"* e nei sui lavori successivi[6, 7], Chaum introduce l'idea di un sistema di pagamento elettronico anonimo basato sulla tecnologia di Blind Signature, sistema che verrà poi implementato dallo stesso autore con ECash™ nel 1994. Come osserva Schoenmakers[8], nonostante le istituzioni finanziare già utilizzassero dietro le quinte sistemi elettronici per processare transazioni, l'apertura al pubblico dei sistemi di pagamento elettronici introduce la necessità di un sistema di pagamento elettronico che possa essere utilizzato in una rete ad accesso pubblico, particolarmente in riferimento alla rete Internet. Nel 1998 DigiCash fallisce, principalmente a causa della forte competizione posta in atto dal sistema di pagamento tramite carta di credito. In particolare, secondo le parole di Chaum, la compagnia ha sofferto di un chicken-and-egg problem: fu difficile convincere abbastanza venditori ad adottare il sistema per poter acquisire clienti e vice versa. (Lo stesso destino spetta ad altre forme di pagamento elettronico dello stesso periodo tra le quali possono essere citate CyberCoin e Virtual PIN).

Nonostante si sia rivelata infruttuosa, l'esperienza di Chaum rappresenta il primo esempio di un sistema di pagamento elettronico anonimo e risulta pionieristico sia nei problemi che vengono presentati come nelle soluzioni proposte, come avviene nel caso del problema di double spending.

Altri sistemi di moneta digitale si sono susseguiti negli anni, tra i più rilevanti possiamo citare e-gold, Liberty Reserve e PayPal: i primi due non più attivi, entrambi per controversie legali, mentre l'ultimo ancora attivo.

Un altro tassello fondamentale nello sviluppo delle monete elettroniche è l'introduzione nel 1997 di Hashcash[9] da parte di Adam Back che indipendentemente descrive un sistema simile a quanto proposto da Cynthia Dwork,

Moni Naor e Eli Ponyatovski nel 1992 in *"Pricing via Processing or Combatting Junk Mail"*[10]. Entrambe le proposte prevedono l'utilizzo di quella che nel 1999 viene formalizzata da Markus Jakobsson e Ari Juels come *"proof of work" (PoW)* per arginare il fenomeno dello spam[11]. Appena un anno prima, nel 1998, Wei Dai introduce B-money[12], un sistema basato su Hashcash che comprendeva varie funzionalità oggi comuni alla maggior parte delle criptomonete. B-money tuttavia non fu mai lanciato e rimase solo una proposta ma il lavoro di Dai non fu vano; dieci anni dopo, all'inizio dello sviluppo di Bitcoin, Dai fu il primo ad essere contattato da Nakamoto. A lui seguirono altri sviluppatori tra i quali Hal Finney, che introdusse l'idea di *"reusable PoW"*[13]. Nel 1998 Nick Szabo creò bitgold[14], un meccanismo per una digital currency decentralizzata e sicura che implementava già alcune idee alla base degli smart contracts. Le teorie di Szabo riguardo gli accordi auto-attuanti furono formulate all'incirca durante lo stesso periodo in cui Ian Grigg introdusse l'idea dei Ricardian Contract, un metodo per registrare un documento come un contratto legale e collegarlo in modo sicuro ad altri sistemi[15]. Bitgold, come B-money, non fu mai implementato.

1.1.1 Creazione e diffusione di Bitcoin

La pubblicazione del whitepaper di Bitcoin avviene finalmente nel 2008, poco dopo il clou della crisi finanziaria e del collasso di Lehman Brothers. L'obbiettivo a cui Bitcoin mira è fornire una valuta digitale P2P che non necessiti della presenza di banche ed altri intermediari. La prima implementazione di Bitcoin fu opera di Nakamoto ma successivamente la direzione dei lavori passò gradualmente ad un gruppo di primi utenti e sviluppatori capitanati da Gavin Andersen.

E' interessante notare come nel suo whitepaper[1] Nakamoto non si riferisca a *"blockchain"* ma solo a *"chain of blocks"*. Il termine *"blockchain"* si diffonde successivamente durante lo sviluppo di protocolli alternativi a Bitcoin.

Dalla sua creazione nel 2008 fino ad oggi il prezzo di acquisto per un bitcoin è passato da 0.003$, il valore della prima quotazione su un exchange dedicato, fino al valore massimo di circa 19000$ come è possibile osservare nella figura 1.1. A questo valore è stato stimato che il patrimonio di Nakamoto abbia raggiunto un valore di 19 miliardi di dollari, dato che possiede circa un milione di bitcoins. Nonostante ciò, dalla sua uscita di scena nel 2011 Nakamoto non ha operato transazioni con i bitcoins di cui sappiamo essere in possesso.

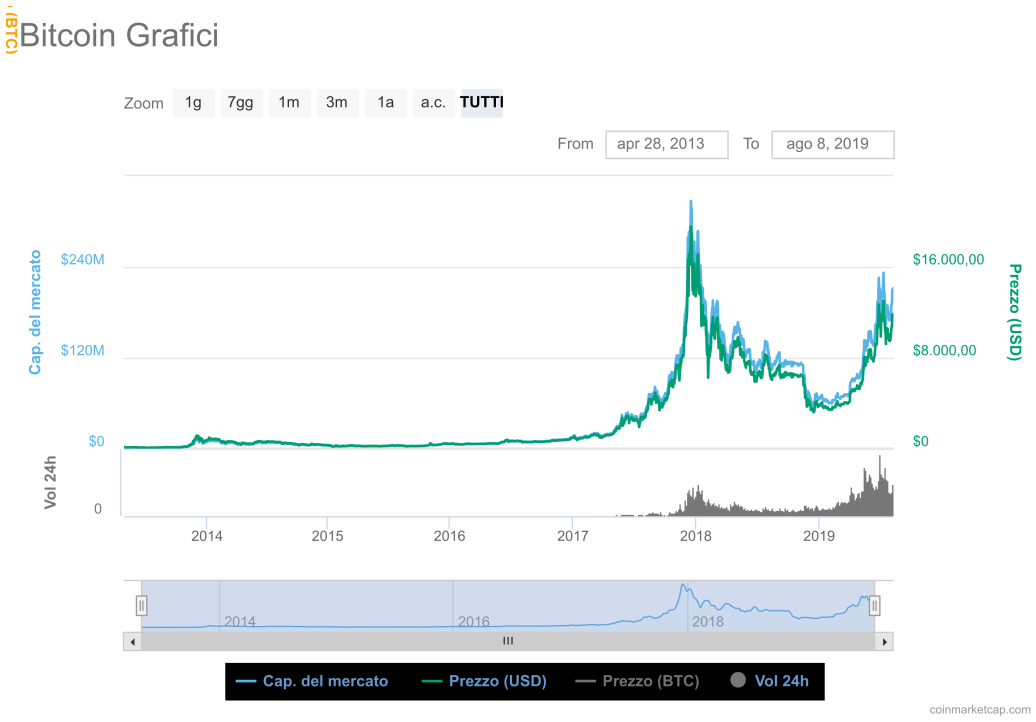

Figura 1.1: Grafico di prezzo di Bitcoin

Il codice open source di Bitcoin ha permesso ad altri sviluppatori di creare protocolli alternativi[16] ed è grazie a questo che si assiste alla nascita di quelle che verranno in seguito definite *"Altcoin"*, un'abbreviazione di *"alternative coin"*. Allo stesso tempo si inizia a diffondere anche l'utilizzo di Bitcoin in alcune organizzazioni, tra le prime possiamo citare *Wikileaks* e *Electronic Frontier Foundation*. Nel 2011 viene pubblicato il sito tematico

Bitcoin Magazine che tra i suo fondatori annovera anche Vitalik Buterin. Nel 2012 la Bitcoin Foundation inizia i propri sforzi di standardizzazione e promozione di Bitcoin. Nello stesso anno BitPay, un azienda che offre servizi di pagamento tramite bitcoins riporta che più di 1000 commercianti avevano adottato il suo sistema di pagamento. Nel 2013, Coinbase, un'altra azienda per il processo dei pagamenti tramite Bitcoin, annuncia di aver venduto bitcoins per un valore pari a 1 milione di dollari in un mese, ad un prezzo di circa 22$ per unità.

A questo punto, l'utilizzo di Bitcoin entra nei radar degli enti regolatori a causa di una alta volatilità dovuta a diversi fattori concorrenti. L'*American Financial Crimes Enforcement Network (FinCEN)* stabilisce delle linee guida per monete virtuali decentralizzate, con particolare riferimento a Bitcoin, introducendo l'obbligo per i miners di essere registrati come *Money Service Businesses (MSB)* qualora vendessero i bitcoins generati[17]. A partire dal 2013 si può osservare una divergenza nelle regolamentazioni adottate da diversi stati. Tra gli esempi più rilevanti possiamo citare il caso tailandese: il *Foreign Exchange Administration and Policy Department* sancisce che Bitcoin è illegale data l'assenza di un framework legale al quale possa essere sottoposto[18]. Il 2013 risulta essere un anno saliente per Bitcoin anche per un altro aspetto: la Cina diventa il più grande punto di scambio di bitcoins. Tuttavia la *People's Bank of China (PBC o PBOC)* proibisce l'uso di Bitcoin agli istituti finanziari cinesi, causando in dicembre un crollo dei prezzi[19].

Nel 2014 emergono prodotti derivati basati su Bitcoin, approvati dall'*U.S. Commodity Futures Trading Commission* come prodotti di scambio over-the-counter che hanno come sottostante il prezzo del bitcoin. Inoltre, aumentano i casi di furto elettronico di bitcoins, di particolare rilievo risultano essere i casi di Mt Gox e Bitstamp. Da qui in poi l'espansione globale di Bitcoin continua inesorabilmente, nonostante le incertezze della mutevole configurazione giuridica nei diversi stati, grazie anche ad una tendenza di regolamentazione più favorevole.

1.1.2 Creazione e diffusione di Ethereum

Ethereum viene descritto in un whitepaper[20] da Vitalik Buterin, verso la fine del 2013. Vitalik riporta che durante l'ottobre del 2013, lavorando in Israele a contatto con il team di Mastercoin, in *"Utimate Scripting"* avanza delle proposte per generalizzare il protocollo utilizzato da Mastercoin. Queste proposte risultano abbastanza distanti dalle funzionalità di Ethereum perché sono una prima versione di un protocollo finalizzato per ora alla creazione di un contratto tra due parti nelle quali il valore monetario viene redistribuito secondo una formula specificata nel contratto stesso. Sebbene impressionato, il team di Mastercoin decise di non implementare le modifiche proposte. In Dicembre, Buterin propone una nuova versione, la prima che adotta il nome Ethereum e che presenta una generalizzazione degli smart contracts. Invece di utilizzare un linguaggio di scripting per descrivere i termini di una relazione tra due parti, in questa versione gli smart contracts sono account veri e propri, con la possibilità di possedere, spedire e ricevere assets[21]. Da questo momento in poi Ethereum attira l'interesse di altri sviluppatori, primi tra i quali Gavin Wood, Jeffrey Wilcke, Andrew Miller, Mihai Alisie, Anthony Di Iorio e Charles Hoskinson.

Ethereum viene formalmente annunciato il 25 Gennaio 2014 a Miami, durante la *North America Bitcoin Conference* e nell'estate del 2014 dopo i primi lavori degli sviluppatori il protocollo si stabilizza, quindi Wood pubblica una specifica semi-formale di Ethereum e dell'*Ethereum Virtual Machine*[22]. Intanto altri sviluppatori si aggiungono al progetto ed anche alcuni finanziatori decidono di prendervene parte. Parallelamente a queste prime fasi di sviluppo vengono formate diverse entità legali per poter coordinare il progetto, in particolare *Ethereum Switzerland GmbH (EthSuisse)* e *Ethereum Foundation (Stiftung Ethereum)*. Lo sviluppo fu finanziato tramite crowdfounding, vendendo il token di Ethereum, denominato ether, in cambio di bitcoins. Grazie alla crowdsale, 11,9 milioni di ethers furono venduti, per un valore di circa 18,4 milioni di dollari.

Sin dal suo lancio iniziale, Ethereum ha ricevuto numerosi aggiornamenti

pianificati del proprio protocollo, con cambiamenti importanti per quanto riguarda le funzionalità dello stesso.

A testimonianza del successo del progetto Ethereum, nel marzo 2017, varie startups, gruppi di ricerca e aziende appartenenti alla Fortune 500 hanno annunciato la creazione della *Enterprise Ethereum Alliance (EEA)* con trenta membri fondatori.

In maggio, l'organizzazione no-profit conta 116 membri, includendo ConsenSys, CME Group, Cornell University's research group, Toyota Research Institute, Samsung SDS, Microsoft, Intel, J. P. Morgan, Cooley LLP, Merck KGaA, DTCC, Deloitte, Accenture, Banco Santander, BNY Mellon, ING, and National Bank of Canada. In Luglio il numero di membri raggiunge i 150, tra i quali MasterCard, Cisco Systems, Sberbank and Scotiabank. Altri membri di rilevo sono Advanced Micro Devices (AMD), FedEx Corporate Services, Hyperledger, Securitize, TOKENY, UniCredit e VMware.

1.2 Le criptomonete come strumento di finanziamento

Come già accennato, inizialmente Ethereum ricevette finanziamenti tramite una crowdsale sulla rete Bitcoin. Questo ha permesso agli sviluppatori di intercettare l'interesse degli utenti disposti a supportare lo sviluppo della piattaforma. Tuttavia, la prima token sale avvenne nel luglio 2013 grazie a J.R. Willett con lo scopo di raccogliere fondi per il progetto Omni tramite la vendita di Mastercoin[23]. Per motivare i potenziali investitori a contribuire al progetto, Willett fece in modo di rendere disponibile molte delle funzioni solo ai possessori di mastercoins. Fu quindi Willet ad ideare il modello delle ICO ed a lui dobbiamo anche l'introduzione di quelli che successivamente verranno definiti come utility token. La token sale permise a Willet di raccogliere finanziamenti per un valore di 600 mila dollari, al tempo circa 4740 bitcoins. Il protocollo introdotto con Mastercoin, oggi chiamato *"Omni Layer"*, ha avuto molto successo tanto da essere usato come protocollo su cui

si basa il token Tether (USDT), ad oggi una delle criptovalute più scambiate dopo Bitcoin[24].

Il secondo token venduto tramite ICO è NextCoin (NXT), che tra settembre ed ottobre del 2013 permette al suo creatore, rimasto anonimo, di raccogliere 21 bitcoins, per un valore compreso tra i 3000 e 17000 dollari. A NextCoin seguono Counterparty (XCP), MaidSafeCoin (MAID), Swarm e successivamente Ethereum.[25, 26, 27, 28]

Poiché Ethereum è pensata come una piattaforma per la creazione ed esecuzione di smart contracts e offre anche la possibilità di sviluppare token sulla piattaforma stessa, risulta ovvio il perché Ethereum sia diventata la piattaforma più usata per ospitare altre token sales. La prima Initial Coin Offering (ICO) basata sulla piattaforma Ethereum fu lanciata il 17 Agosto 2015, per la vendita di token Augur[29]. La vendita durò fino al 5 Settembre dello stesso anno. In seguito alla vendita vennero raccolti più di 5 milioni di dollari per lo sviluppo del progetto Augur. Il progetto è stato lanciato ufficialmente nel luglio del 2018.

Il fenomeno delle ICOs è via via diventato popolare con un picco nel 2017. Dall'inizio del 2017 fino ad ottobre 2017 infatti, vengono raccolti fondi per un valore di 2,3 miliardi di dollari, un risultato dieci volte maggiore rispetto al 2016. Durante tutto il 2017 vengono raccolti fondi per un valore di circa 6 miliardi di dollari, il 37% dei quali dovuti a solamente 20 ICO.

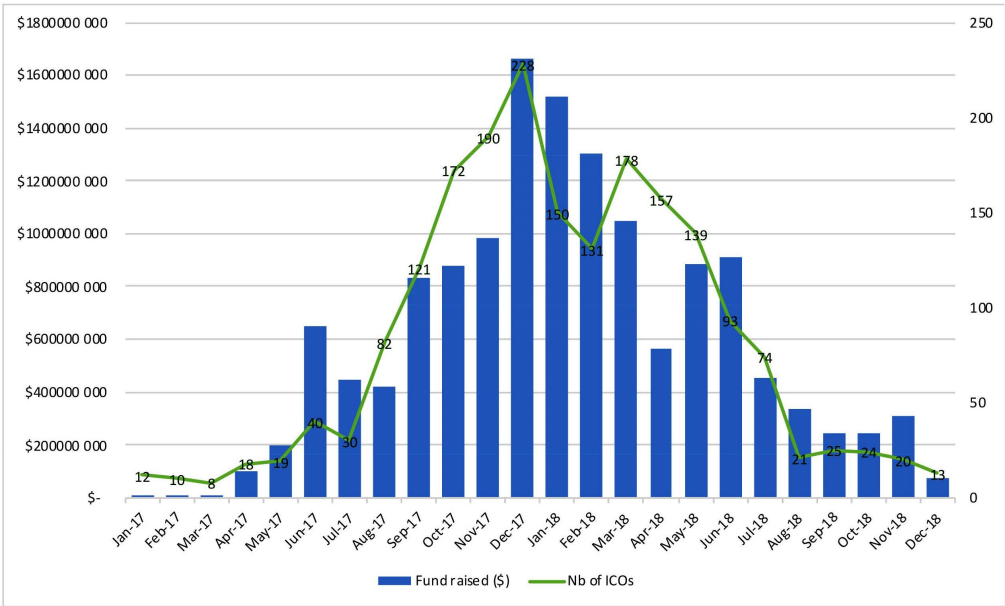

Figura 1.2: ICO data, source: Florie Mazzorana-Kremer[30]

Viene stimato che al Febbraio 2018 il 46% dei progetti finanziati tramite una ICO nel 2017 sono falliti. Questo crollo alimenta un'accesa discussione sulla legittimità delle ICOs e sulla possibilità che queste risultino sempre, o quasi, in bolle speculative.

1.3 Classificazione dei token nelle token sales

In base alla sua funzione, un token venduto in una token sale può essere considerato un utility token, un security token o un currency token. Questa distinzione è comunemente riconosciuta ed è adottata anche dalla *Swiss Financial Market Supervisory Authority (FINMA)*. In particolare si riporta quanto segue:

"Asset tokens represent assets such as participations in real physical underlyings, companies, or earnings streams, or an entitlement to dividends or interest payments. In terms of their economic function, the tokens are ana-

logous to equities, bonds or derivatives.While, Utility tokens are tokens which are intended to provide digital access to an application or service."

Un currency token, come ad esempio Bitcoin, rappresenta una valuta digitale ed è quindi un mezzo di scambio.

Un utility token è usato all'interno di una rete per poter accedere ad una funzionalità o per il funzionamento di questa.

I security tokens, invece, si riferiscono ad una rappresentazione digitale tramite blockchain di strumenti finanziari (securities). La definizione di security tuttavia varia a seconda della legislazione applicabile. Negli Stati Uniti, mercato di riferimento sia per le ICOs che per le STOs, le securities sono definite nell'*U.S. Code of Laws 15 §77b(a)(1)* come:

"The term "security" means any note, stock, treasury stock, security future, bond, debenture, evidence of indebtedness, certificate of interest or participation in any profit sharing agreement, collateral-trust certificate, preorganization certificate or subscription, transferable share, investment contract, voting-trust certificate, certificate of deposit for a security, fractional undivided interest in oil, gas, or other mineral rights, any put, call, straddle, option, or privilege on any security, certificate of deposit, or group or index of securities (including any interest therein or based on the value thereof), or any put, call, straddle, option, or privilege entered into on a national securities exchange relating to foreign currency, or, in general, any interest or instrument commonly known as a "security", or any certificate of interest or participation in, temporary or interim certificate for, receipt for, guarantee of, or warrant or right to subscribe to or purchase, any of the foregoing."

In particolare, per quanto concerne le STOs queste devono aderire ad una delle seguenti normative: *Regulation D*, *Regulation A+* o *Regulation S*[31, 32, 33].

In Italia l'equivalente delle securities sono gli strumenti finanziari definiti nel *D.lgs 24 febbraio 1998 n.58*, noto come testo unico della finanza, abbreviato in *TUF* o *legge Draghi*[34]. Ad oggi in Italia vi sono ulteriori linee guida riguardanti la configurazione da adottare per il pagamento di imposte

e in materia di antiriciclaggio. Inoltre, con l'introduzione nel nostro ordinamento delle tecnologie DLT e Smart Contract, come previsto dalla *legge 12/2019* di conversione del *D.L. n.135/2018*, l'emissione di security tokens da parte di soggetti in possesso dei requisiti necessari è teoricamente possibile, nonostante nessuna STO sia stata realizzata fino ad oggi in Italia.

Alcune legislazioni si sono espresse con linee guida specifiche applicabili ai security token, come ad esempio la *Financial Conduct Authority (FCA)* in UK che nel Gennaio 2019 ha indicato la differenza tra security e utility token come segue:

"Security tokens are tokens with specific characteristics that mean they meet the definition of a Specified Investment like a share or a debt instrument as set out in the Regulated Activities Order, and are within the perimeter. While, utility tokens grant holders access to a current or prospective product or service but do not grant holders rights that are the same as those granted by Specified Investments."

Da questo punto di vista le Security Token Offerings sono l'evoluzione naturale delle ICOs. Questa evoluzione è dovuta alla necessità per le aziende di operare nel settore delle criptomonete definendo con chiarezza le normative da rispettare. Lo scopo delle STOs è esclusivamente quello di creare token equiparabili a securities. In questo modo, configurandosi come securities, viene aggirato il vuoto normativo o l'incompletezza legislativa ancora presente in vari stati. Ciò è vantaggioso per un azienda. Trattare un token come una securities, poiché ci si configura in un ambito normativo già consolidato e di cui molti professionisti hanno già esperienza, permette di agevolare l'adozione delle nuova tecnologia. In definitiva, se le ICOs hanno avuto modo di proliferare grazie al vuoto normativo sulle criptomonete, le STOs puntano a diffondersi grazie al fatto di consistere in una vendita di strumenti finanziari rappresentati in forma digitale. La distinzione principale tra una ICO e una STO risulta appunto essere il ruolo del token e della regolamentazione. Una STO può offrire solamente security tokens e deve essere regolamentata. Una ICO, invece, non può offrire un security token poiché non è garantita regola-

mentazione. Non a caso, durante il 2018 la SEC ha sanzionato varie aziende per aver offerto securities tramite ICO non regolamentate. L'approccio alle STOs si basa sull'idea che una security non è definita dal mezzo con la quale essere viene rappresentata. Si pone quindi un'equivalenza tra un documento cartaceo e uno digitale grazia alla definizione legislativa di security che è nella maggior parte dei casi agnostica rispetto alla tecnologia. Alla singola azienda è quindi richiesto di produrre la documentazione legalmente richiesta e di assolvere agli obblighi normativi. Allo stesso tempo è compito dell'azienda assicurarsi di essere autorizzata all'emissione di securities. Ovviamente il maggiore sforzo richiesto all'issuer della securities si traduce in un maggiore costo rispetto ad un ICO. Tuttavia, il maggiore costo permette all'azienda di ridurre il rischio di contenziosi legali e di evitare la possibilità di sanzioni.

In un articolo pubblicato recentemente, Ante e Fielder[35] sostengo l'idea che le STOs siano più adatte delle ICOs nel soddisfare i bisogni di un'azienda e degli investitori. Oltre alla chiarezza normativa vengono citati come argomenti la possibilità di operare in un mercato regolamentato, la riduzione dei costi rispetto ad una security tradizionale ed infine la trasparenza e sicurezza fornite dell'utilizzo della tecnologia blockchain in modo legale.[30] Altri vantaggi che si sostiene vengano offerti dalle STOs sono la possibilità di implementare pagamenti automatici per realizzare un meccanismo di pagamento dei dividendi e la frazionabilità della security[36]. In seguito in questo elaborato si andrà a verificare e obbiettare alcuni di questi pretese.

Per riassumere, nella tabella 1.1 possiamo confrontare le caratteristiche delle ICOs e delle STOs.

	Initial Coin Offering (ICO)	Security Token Offering (STO)
Periodo di successo	2017	2018
Investitori	Copertura globale, barriere all'entrata minime	Solo autorizzati, barriere all'entrata
Capitale richiesto	Capitale minimo, processo automatizzato	Capitale medio-basso, più efficienti di una security tradizionale
Regolamentazione	Non regolamentate	Regolamentate in base alla giurisdizione applicabile
Token	Qualsiasi tipo di token	Security (equity, asset, ecc)
Garanzie	A discrezione dell'azienda	Diritti legali, sorveglianza dei regolatori e auditing aziendale

Tabella 1.1: Confronto tra ICOs e STOs

Capitolo 2

Confronto tra le implementazioni

Nel corso degli ultimi anni la community di sviluppatori ha portato avanti diverse iniziative per imporre uno standard condiviso per le STOs. Inizialmente lo standard de facto è stato *ERC-20*[37], già ampiamente adottato dalla maggior parte delle ICOs. Tra gli standard utilizzati per le STOs, ERC-20 è lo standard più maturo e con maggiore supporto essendo stato creato verso la fine del 2015. Altri tentativi di standardizzazione nascono dalla necessità di garantire all'issuer della STO una maggiore governance e per facilitare il rispetto delle regolamentazioni sulle securities. Non a caso possiamo osservare come molti tentativi di standardizzazione siano stati operati da aziende che offrono piattaforme per l'emissione di tokens. Per la maggior parte di queste aziende l'approccio è stato quello di estendere lo standard ERC-20, in modo da ottenere retrocompatibilità ed essere più appetibili alla community di sviluppatori con conoscenza dello standard sopracitato. Tra i casi che verranno analizzati troviamo:

- *ST-20*, sviluppato da Polymath
- *R-Token*, sviluppato da Harbor
- *DS Protocol*, sviluppato da Securitize
- *T-REX*, sviluppato da Tokeny

- *S3*, sviluppato da OpenFinace

Altri standard si sono sviluppati a partire da necessità diverse, come nel caso degli standard *ERC-721* ed *ERC-1155* che implementano tokens non fungibili. La notorietà di questi protocolli è in parte dovuta all'introduzione del concetto di scarsità digitale tramite l'utilizzo di tokens non omogenei, un approccio che ben si presta all'ambito videoludico. Una proposta più recente e che ha ottenuto maggiore supporto sia dalla community sia da aziende, alcune delle quali già in possesso di uno standard proprietario, è lo standard *ERC-1400*, rinominato *Security Token Standard*. ERC-1400 è una collezione di diversi standard sviluppati per essere interoperabili e facilmente estendibili all'emergere della necessità di nuove funzionalità da introdurre. Il passaggio ad uno standard condiviso e supportato da più stakeholder rappresenta uno step fondamentale par la diffusione di questo fenomeno.

2.1 Implementazione con token ERC-20

ERC-20 definisce un'interfaccia standard che rappresenta un token. Lo standard fornice funzionalità basilari per il trasferimento dei tokens. Lo scopo principale dello standard è quello di permettere interoperabilità tra le applicazioni che supportano lo standard, come ad esempio wallets ed exchanges. Lo standard ERC-20 prevede le seguenti funzioni:

```
totalSupply() public view returns (uint256 totalSupply)
```

```
balanceOf(address _owner) public view returns (uint256
    balance)
```

```
transfer(address _to, uint256 _value) public returns (bool
    success)
```

```
transferFrom(address _from, address _to, uint256 _value)
    public returns (bool success)
```

```
approve(address _spender, uint256 _value) public returns (
    bool success)
```

```
allowance(address _owner, address _spender) public view
    returns (uint256 remaining)
```

Inoltre, vengono definiti i seguenti eventi:

```
Transfer(address indexed _from, address indexed _to, uint256
    _value)
```

```
Approval(address indexed _owner, address indexed _spender,
    uint256 _value)
```

Due delle implementazioni più diffuse di questo standard sono state realizzate da OpenZeppelin e ConsenSys. Le due implementazioni sono entrambe pienamente compatibili con lo standard sebbene presentino delle lievi differenze come si può osservare in appendice A.

Una appunto interessante si può trarre osservando i cambiamenti avvenuti durante lo sviluppo iniziale dello standard. Se in principio le prime bozze dello standard prevedevano un focus sulla creazione di *"Coins"*, gli sviluppatori in modo lungimirante hanno deciso di generalizzare il protocollo per renderlo applicabile ad ogni bene fungibile trasferibile, motivo per il quale è stata scelta la denominazione più generica di *"token"*.

2.1.1 ERC-20 proprietary extension

ST-20

Lo standard ST-20, introdotto da Polymath, nasce con l'obbiettivo di aggiungere all'interfaccia ERC20 la possibilità di inserire restrizioni allo scambio di tokens in modo da poter essere conforme alle normative sulle securities. In appendice B è possibile osservare l'interfaccia dello smart contract che definisce un security token che rispetta lo standard ST-20. Trattandosi di un'estensione, lo smart contract è compatibile con lo standard ERC-20 e ne implementa tutte le funzioni e gli eventi previsti, aggiungendo a questi altre funzionalità e caratteristiche che agevolano l'adattabilità e la governance della STO. Per esempio, nel caso dello scambio senza restrizioni tra due

parti, caso che non sempre è desiderabile nelle STOs, le transazioni possono essere limitate da una funzione *"verifyTransfer"*. Ciò permette inoltre l'implementazione di una whitelist, una blacklist, un limite minimo/massimo di trasferimento, etc. In modo più dettagliato, l'interfaccia creata da Polymath si avvale principalmente di tre tipi di componenti: un primo tipo detto *"Transfer Manager"*, un tipo denominato *"Permission module"* ed infine un modulo che si occupa delle funzionalità della token sale, ovvero *"STO module"*. L'utilizzo di numerose componenti per implementare diverse funzionalità permette di avere un architettura modulare che comprende di volta in volta solamente le funzionalità richieste, permettendo così di rendere lo smart contract più efficiente.

R-Token

R-token, implementato da Harbor, offre delle caratteristiche che lo rendono particolarmente innovativo, tanto da essere concettualmente compatibile con un token ERC-1400. Un altro aspetto particolare del protocollo è l'integrazione di una whitelist per gli operatori presenti nell'ecosistema proposto da Harbor. R-Token, che sta per *"regualted token"* si differenzia dagli altri standard compatibili con ERC-20 per l'adozione di alcuni servizi di document management e di custody. Generalmente una STO Harbor si compone di tre smart contracts:

- Un *"Regulator Service"* che si occupa di interagire tramite un oracolo con la regolamentazione definita off-chain.
- Un *"Service Registry"* che contiene un riferimento al *"Regulator Service"*. Questo riferimento può essere modificato per permettere di aggiornare la business logic della STO.
- Uno smart contract *"R-Token"* che contiene le caratteristiche del token, le sue funzioni e un riferimento immutabile al *"Service Registry"*.

DS Protocol

L'estensione di ERC-20, proposta da Securitize, ha due componenti principali. Il primo è lo smart contract *"DSTokenInterface"* che nel complesso non risulta apportare cambiamenti rilevanti rispetto ai protocolli già presentati. Il secondo componente risulta essere più originale ed è lo smart contract *"DSServiceConsumerInterface"* definito come segue:

```
function getDSService(uint _serviceId) public view returns (
    address);

function setDSService(uint _serviceId, address _address)
    public /*onlyMaster*/ returns (bool);
```

Queste funzioni permettono di associare in modo dinamico le componenti e di verificarne la presenza (al fine di mantenere compatibilità tra diverse versioni del protocollo). I servizi sono identificati da un id nel seguente modo:

```
uint public constant TRUST_SERVICE = 1;
uint public constant DS_TOKEN = 2;
uint public constant REGISTRY_SERVICE = 4;
uint public constant COMPLIANCE_SERVICE = 8;
uint public constant COMMS_SERVICE = 16;
uint public constant WALLET_MANAGER = 32;
uint public constant LOCK_MANAGER = 64;
uint public constant ISSUANCE_INFORMATION_MANAGER = 128;
```

T-REX

T-REX, per esteso *"Token for Regulated EXchanges"*, presenta un approccio diverso rispetto ad altri protocolli. L'identità, infatti, ricopre un ruolo fondamentale nel protocollo, tanto da prevedere dei meccanismi di gestione dell'identità dell'investitore on-chain secondo i protocolli *ERC-725* ed *ERC-735*. Il meccanismo di controllo dell'identità sostituisce la logica di whitelisting e blacklisting rendendo il protocollo più flessibile. Possiamo osservare il funzionamento del protocollo nel seguente diagramma in figura 2.1.

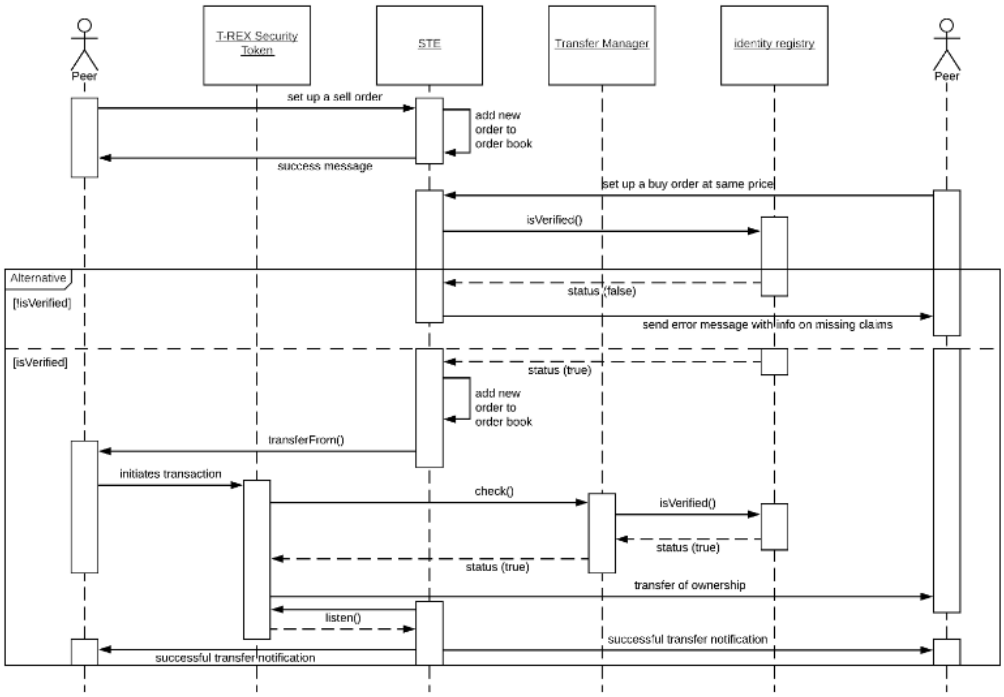

Figura 2.1: Sequence diagram of a Distributed Security Token Exchange transaction with T-REX tokens

S3

S3, per esteso *"Smart Securities Standard"*, è un altra estensione di ERC-20 che si distingue per una semplice caratteristica di compliance poiché il protocollo S3 include degli smart contracts dedicati ad implementare le restrizioni e le regole imposte dalla *Regulation D*, la *Regulation S*, la *Regulation A+* e *Regulation CF*.

```
///RegD506c.sol
/// @title Functions that tokens will need to call to
   configure themselves
contract RegD506c is TransferRestrictor {
  function startHoldingPeriod() public;
  function registerAmlKycChecker(address _checker, address
     _token) public;
```

```
  function registerAccreditationChecker(address _checker,
      address _token) public;
}
```

Questo approccio semplifica notevolmente la struttura del protocollo ma presenta numerosi svantaggi, soprattutto in termini di manutenibilità. Nell'ultima versione disponibile del protocollo l'approccio è mutato notevolmente, tanto che la restrizione degli scambi è delegata ad un altro smart contract non specificato.

2.2 Implementazione con token ERC-721 e ERC-1155

Sebbene non siano pensati in modo specifico per le securities, gli standard ERC-721 ed ERC-1155 sono rilevanti per l'argomento analizzato in quanto permettono di tokenizzare assets fisici. Tramite l'introduzione del concetto di non fungibilità nello standard ERC-721 è possibile definire uno smart contract in cui un token è unico. Questo concetto si può applicare con facilità ad oggetti rari, unici oppure ogni tipo di oggetto collezionabile. La sua implementazione più famosa è il progetto *CryptoKitties*, un gioco che permette di comprare, vendere e scambiare carte virtuali. Nel 2017 il gioco divenne talmente popolare da congestionare la rete Ethereum. Ogni carta rappresenta un *CryptoKitty*, ovvero un token non-fungibile. Ad oggi sono stati scambiati CryptoKitties per un valore superiore a 27 milioni di dollari. Il valore di un singolo token può raggiungere cifre molto alte e sono nati exchanges dedicati a questo protocollo, come possiamo vedere in figura 2.2: un singolo token, associato a questo CryptoKitty, è stato venduto per 253 ethers, ovvero 114mila dollari.

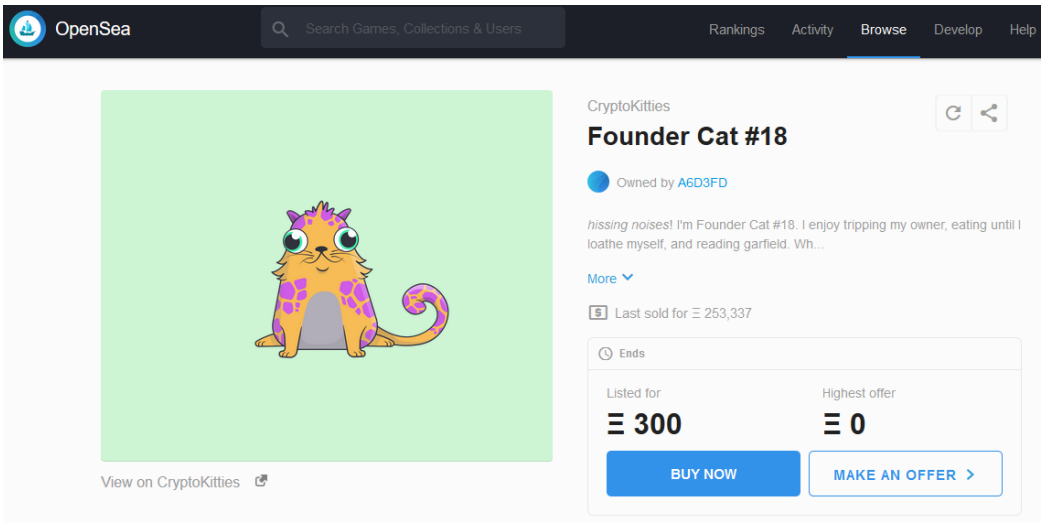

Figura 2.2: Second most expensive CryptoKitty, traded on a specialized exchange

ERC-1155 rappresenta un evoluzione dei tokens ERC-20 ed ERC-721, poiché permette di gestire sia tokens fungibili che non fungibili in un unico contratto, permettendo transazioni con più tokens di tipo diverso. Questa funzionalità è stata realizzata con l'obbiettivo di permettere l'adozione di tokens in videogiochi più complessi rispetto a CryptoKitties, come nel caso di giochi che permetto la vendita, l'acquisto e lo scambio di una currency del gioco e di oggetti virtuali.

2.3 Implementazione con token ERC-1400

Lo standard ERC-1400 è nato in seguito allo sforzo di numerosi operatori del settore per cercare di ottenere uno standard condiviso. Grazie all'ampio supporto ricevuto da queste aziende, alcune delle quali già promotrici di standard illustrati in precedenza, ERC-1400 si sta affermando come uno standard *de facto* per la creazione di STOs. Inizialmente ERC-1400 è stato pensato come uno standard per realizzare security tokens ma già dalle prime fasi di sviluppo è emersa la necessità di definire un protocollo modulare e facilmen-

te espandibile. Proprio per questo, ERC-1400 è una collezione di standard interoperabili. Lo sviluppo di ERC-1400 non segue una roadmap definita in quanto lo scopo degli autori è quello di aggiungere il supporto di nuove funzionalità qualora gli utilizzatori ne esprimano il bisogno. Nonostante la mancanza di una roadmap, gli autori hanno informalmente assicurato nuove release approssimativamente ogni sei mesi, questo per facilitare l'adozione da parte delle aziende interessate. Grazie alla discussione tra più gruppi e al contributo di più aziende, lo standard copre la maggior parte degli scenari che possono presentarsi in una STO. Ad esempio, nella figura 2.3 possiamo vedere come siano stati dedicati diversi codici di errore a varie situazioni, un notevole miglioramento nell'aspetto di governance rispetto allo standard ERC-20.

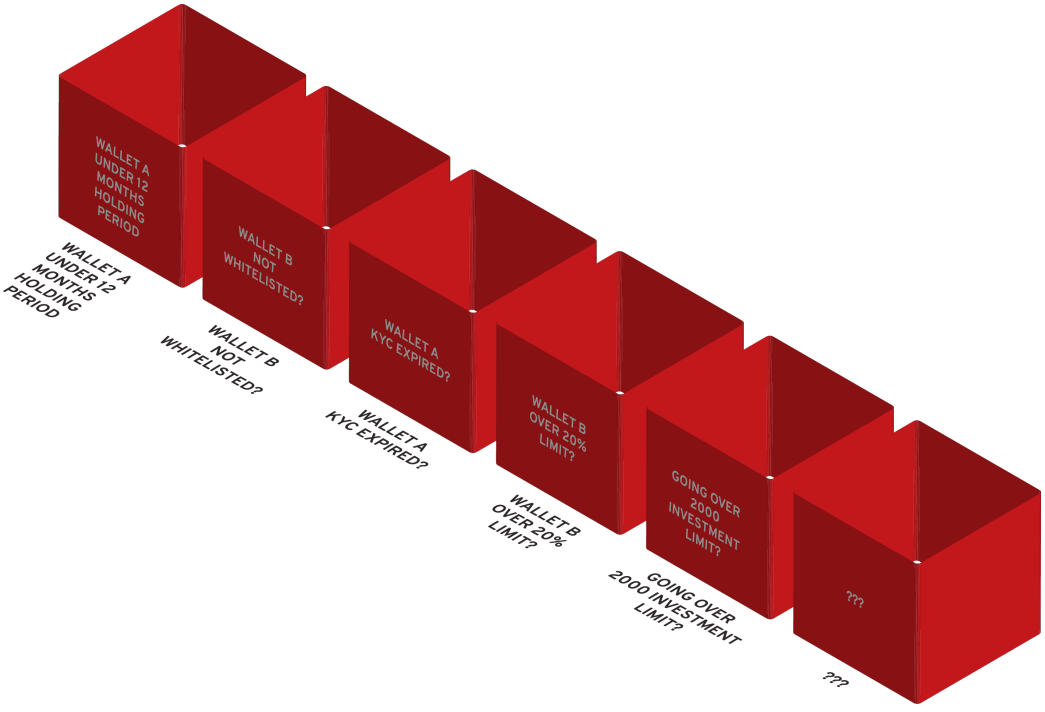

Figura 2.3: ERC-1400 Error cases examples

La collezione di standard definiti da ERC-1400 ad oggi fornisce funzionalità di gestione della documentazione, gestione degli errori, controllo delle transazioni, input di dati off-chain, gestione dell'emissione dei tokens e diffe-

renziazione dei tokens. Analizzare nel dettaglio queste features risulta agevole grazie alla modularità dello standard. Al momento, ERC-1400 comprende i seguenti standard:

- ERC-1410
- ERC-1594
- ERC-1643
- ERC-1644
- ERC-2258 *(in fase di adozione)*

2.3.1 ERC-1410: Partially Fungible Token Standard

Il primo standard previsto da ERC-1400 è lo standard ERC-1410, che riguarda l'introduzione del concetto di fungibilità parziale. Oltre ad essere il primo standard introdotto è anche il più corposo poiché aggiunge un meccanismo di partizioni per ottenere non fungibilità parziale. All'interno di ogni partizione ogni tokens è fungibile ma tokens di partizioni diverse sono non fungibili tra loro.

```
/// @title ERC-1410 Partially Fungible Token Standard
/// @dev See https://github.com/SecurityTokenStandard/EIP-
   Spec

interface IERC1410 {

    // Token Information
    function balanceOf(address _tokenHolder) external view
       returns (uint256);
    function balanceOfByPartition(bytes32 _partition, address
        _tokenHolder) external view returns (uint256);
    function partitionsOf(address _tokenHolder) external view
        returns (bytes32[]);
    function totalSupply() external view returns (uint256);
```

```
    // Token Transfers
    function transferByPartition(bytes32 _partition, address
        _to, uint256 _value, bytes _data) external returns (
        bytes32);
    function operatorTransferByPartition(bytes32 _partition,
        address _from, address _to, uint256 _value, bytes
        _data, bytes _operatorData) external returns (bytes32)
        ;
    function canTransferByPartition(address _from, address
        _to, bytes32 _partition, uint256 _value, bytes _data)
        external view returns (byte, bytes32, bytes32);

    // Operator Information
    function isOperator(address _operator, address
        _tokenHolder) external view returns (bool);
    function isOperatorForPartition(bytes32 _partition,
        address _operator, address _tokenHolder) external view
         returns (bool);

    // Operator Management
    function authorizeOperator(address _operator) external;
    function revokeOperator(address _operator) external;
    function authorizeOperatorByPartition(bytes32 _partition,
         address _operator) external;
    function revokeOperatorByPartition(bytes32 _partition,
        address _operator) external;

    // Issuance / Redemption
    function issueByPartition(bytes32 _partition, address
        _tokenHolder, uint256 _value, bytes _data) external;
    function redeemByPartition(bytes32 _partition, uint256
        _value, bytes _data) external;
    function operatorRedeemByPartition(bytes32 _partition,
        address _tokenHolder, uint256 _value, bytes
        _operatorData) external;

    // Transfer Events
    event TransferByPartition(
```

```
        bytes32 indexed _fromPartition,
        address _operator,
        address indexed _from,
        address indexed _to,
        uint256 _value,
        bytes _data,
        bytes _operatorData
    );

    // Operator Events
    event AuthorizedOperator(address indexed operator,
        address indexed tokenHolder);
    event RevokedOperator(address indexed operator, address
        indexed tokenHolder);
    event AuthorizedOperatorByPartition(bytes32 indexed
        partition, address indexed operator, address indexed
        tokenHolder);
    event RevokedOperatorByPartition(bytes32 indexed
        partition, address indexed operator, address indexed
        tokenHolder);

    // Issuance / Redemption Events
    event IssuedByPartition(bytes32 indexed partition,
        address indexed operator, address indexed to, uint256
        amount, bytes data, bytes operatorData);
    event RedeemedByPartition(bytes32 indexed partition,
        address indexed operator, address indexed from,
        uint256 amount, bytes operatorData);

}
```

Ogni partizione è identificata da una chiave e dei metadati arbitrari. Ogni token contiene le informazioni sulla partizione a cui appartiene. Questo meccanismo di partizioni risulta utile per identificare e distinguere tokens che hanno restrizioni o benefici diversi. Infatti, questa funzionalità permette di distinguere i tokens in base alla fase della token sale in cui sono stati generati per permettere il *vesting* dei tokens oppure per determinare differenti

capacità di voto legate al possesso del token.

2.3.2 ERC-1594: Core Security Token Standard

Questo standard permette l'esecuzione di operazioni che includono l'iterazione tra dati on-chain e off-chain. Fornire dati off-chain permette ad esempio di eseguire una operazioni autorizzate off-chain con una firma, senza dover ricorrere ad una whitelist on-chain. Questo protocollo si occupa anche di verificare la possibilità di eseguire un operazione e restituire un appropriato *status code* di risposta, evitando di dover provare ad eseguire l'operazione per poi vederla fallire. Infine, in questo standard vengono incorporate le funzionalità di gestione della token sale, in particolare con le funzioni relative all'emissione e alla riscossione.

```
/// @title IERC1594 Security Token Standard
/// @dev See https://github.com/SecurityTokenStandard/EIP-
   Spec

interface IERC1594 is IERC20 {

    // Transfers
    function transferWithData(address _to, uint256 _value,
       bytes _data) external;
    function transferFromWithData(address _from, address _to,
        uint256 _value, bytes _data) external;

    // Token Issuance
    function isIssuable() external view returns (bool);
    function issue(address _tokenHolder, uint256 _value,
       bytes _data) external;

    // Token Redemption
    function redeem(uint256 _value, bytes _data) external;
    function redeemFrom(address _tokenHolder, uint256 _value,
        bytes _data) external;

    // Transfer Validity
```

```
    function canTransfer(address _to, uint256 _value, bytes
        _data) external view returns (bool, byte, bytes32);
    function canTransferFrom(address _from, address _to,
        uint256 _value, bytes _data) external view returns (
        bool, byte, bytes32);

    // Issuance / Redemption Events
    event Issued(address indexed _operator, address indexed
        _to, uint256 _value, bytes _data);
    event Redeemed(address indexed _operator, address indexed
         _from, uint256 _value, bytes _data);

}
```

2.3.3 ERC-1643: Document Management Standard

Questo standard permette di collegare dei documenti ad uno smart contract ed offre la possibilità di aggiornarli. I documenti possono essere per esempio prospetti informativi sulla security ed in generale qualsiasi documento legale necessario.

```
/// @title IERC1643 Document Management (part of the ERC1400
    Security Token Standards)
/// @dev See https://github.com/SecurityTokenStandard/EIP-
    Spec

interface IERC1643 {

    // Document Management
    function getDocument(bytes32 _name) external view returns
         (string, bytes32, uint256);
    function setDocument(bytes32 _name, string _uri, bytes32
        _documentHash) external;
    function removeDocument(bytes32 _name) external;
    function getAllDocuments() external view returns (bytes32
        []);
```

```
    // Document Events
    event DocumentRemoved(bytes32 indexed _name, string _uri,
        bytes32 _documentHash);
    event DocumentUpdated(bytes32 indexed _name, string _uri,
        bytes32 _documentHash);

}
```

Per quanto riguarda la gestione dei documenti privati, però, si può fare una lieve critica a questo standard poiché rendere possibili solo due alternative per gestire questo caso d'uso: rendere pubblico il documento privato oppure imporre una password per accedere al documento (dovendo quindi fornire una password all'interessato tramite un canale di comunicazione esterno). Al momento, non essendo presenti alternative valide, il metodo preferibile è il secondo.

2.3.4 ERC-1644: Controller Token Operation Standard

Questo standard permette di eseguire operazioni di trasferimento forzato dei tokens. Poiché soggetti a regolamentazioni diverse in base alla giurisdizione, in alcuni casi i security tokens devono poter essere trasferiti forzatamente. Questa situazione si presenta nel caso in cui ci sia necessità di annullare una transazione fraudolenta o per rispondere ad ordini del tribunale. Vista la sensibilità di questa operazione solo un indirizzo autorizzato ha i permessi per eseguirla:

```
/// @title IERC1644 Controller Token Operation (part of the
    ERC1400 Security Token Standards)
/// @dev See https://github.com/SecurityTokenStandard/EIP-
    Spec

interface IERC1644 is IERC20 {

    // Controller Operation
    function isControllable() external view returns (bool);
```

```
    function controllerTransfer(address _from, address _to,
        uint256 _value, bytes _data, bytes _operatorData)
        external;
    function controllerRedeem(address _tokenHolder, uint256
        _value, bytes _data, bytes _operatorData) external;

    // Controller Events
    event ControllerTransfer(
        address _controller,
        address indexed _from,
        address indexed _to,
        uint256 _value,
        bytes _data,
        bytes _operatorData
    );

    event ControllerRedemption(
        address _controller,
        address indexed _tokenHolder,
        uint256 _value,
        bytes _data,
        bytes _operatorData
    );

}
```

Dato il contrasto tra questa funzionalità (necessaria per motivi di compliance) e i principi di decentralizzazione della blockchain, questa operazione deve svolgersi in modo trasparente. L'unico modo per garantire questa trasparenza è un protocollo dedicato che rende chiara la distinzione tra un trasferimento ed un trasferimento forzato.

2.3.5 ERC-2258: Custodial Ownership Standard

Dato che un security token rappresenta un modo per registrare la proprietà di un asset sottostante, lo standard ERC-2258 è stato creato per soddisfare alcuni casi d'uso in cui il concetto di proprietà non è più sufficiente. Questo

è il caso della custodia di un token da parte di un entità terza per conto del proprietario del token. In questo caso la proprietà del token non varia. Un esempio che rende più chiaro il meccanismo di funzionamento di questo standard è la necessità di utilizzare un security token come collaterale. Il proprietario deve poter continuare a beneficiare dei dividendi o del diritto di voto permessi dal token ma, allo stesso tempo, il custode deve poter trasferire la proprietà del token ad un altro beneficiario in caso di liquidazione della posizione. A seguire, possiamo osservare come le funzioni esposte dall'interfaccia dello standard abbiano lo scopo di definire le operazioni eseguibili da un custode:

```
/// @title IERCx Custodial Ownership Standard (part of the
    ERC1400 Security Token Standards Library)
/// @dev See https://github.com/SecurityTokenStandard/EIP-
    Spec

interface IERCx {

    // Increase the custody limit of a custodian either
        directly or via signed authorisation
    function increaseCustodyAllowance(address _custodian,
        uint256 _amount) external;
    function increaseCustodyAllowanceOf(address _tokenHolder,
         address _custodian, uint256 _amount, uint256 _nonce,
        bytes _sig) external;

    // Query individual custody limit and total custody limit
         across all custodians
    function custodyAllowance(address _tokenHolder, address
        _custodian) external view returns (uint256);
    function totalCustodyAllowance(address _tokenHolder)
        external view returns (uint256);

    // Allows a custodian to exercise their right to transfer
         custodied tokens
    function transferByCustodian(address _tokenHolder,
        address _receiver, uint256 _amount) external;
```

```
    // Custody Events
    event CustodyTransfer(address _custodian, address _from,
        address _to, uint256 _amount);
    event CustodyAllowanceChanged(address _tokenHolder,
        address _custodian, uint256 _oldAllowance, uint256
        _newAllowance);

}
```

Lo standard permette al proprietario di nominare un custode e di affidargli un numero definito di tokens. Un altro motivo rilevante per l'introduzione del concetto di custodia è favorire la riduzione del rischio di frodi e di hacking grazie alla specializzazione di servizi dedicati. Ciò permette di evitare al proprietario del token l'onere di questi compiti.

Capitolo 3

Sviluppi futuri delle STOs

3.1 Initial Exchange Offerings (IEOs)

Le *Initial Exchange Offerings*, abbreviate in *IEOs*, furono inizialmente introdotte nel 2017 ma non si sono affermate prima dell'inizio di quest'anno. Le IEOs consistono in STOs condotte attraverso un exchange. L'exchange quindi si occupa di emettere il token e di listarlo, permettendo il trading secondario, spesso con diritti di esclusività. All'exchange spettano inoltre gli obblighi di gestione del token.

Potenzialmente la partecipazione degli investitori non è limitata ma di fatto viene circoscritta alla base di utenti che utilizza l'exchange. Questo avviene poiché gli accordi tra un azienda ed un exchange per l'emissione di un token di norma contengono clausole di esclusività. La relazione dell'issuer con l'exchange presenta anche dei vantaggi dal punto di vista dell'investitore. Oltre alle garanzie legali tipiche degli strumenti finanziari, le IEOs forniscono maggiore fiducia in quanto è nell'interesse dell'exchange garantire la qualità della IEO e del progetto per cui vengono realizzate. Ciò sprona l'exchange ad eseguire approfondimenti ulteriori sulla qualità dell'investimento. Un altro fattore a favore delle IEOs è la possibilità di utilizzare un infrastruttura preesistente con standard di sicurezza maggiori, un servizio di *customer care*, meccanismi di *KYC/AML* già sviluppati e infine la possibilità di avere

una liquidità al momento del lancio della token sale. Queste caratteristiche permettono all'azienda che voglia emettere il token di delegare quasi completamente il processo di emissione dei tokens e di fare leva sulla piattaforma offerta dall'exchange per i contatti con gli investitori. Allo stesso tempo, l'investitore ha il vantaggio di non dover ripetere le procedure di KYC/AML per ogni STO in cui vuole investire e di poter scambiare il token con liquidità elevata appena cessata la token sale.

Ad oggi c'è molto interesse per le IEOs, soprattutto da parte degli exchanges. Tra le piattaforme per IEOs più famose possiamo citare:

- Binance Launchpad
- Huobi Prime
- OKEx Jumpstart
- Coineal Launchpad
- Bittrex
- Kucoin Spotlight
- Shortex
- Probit Launchpad
- Coinbene MoonBase
- Bitfinex Token Sales
- Coinbase (non annunciato ufficialmente)

L'interesse mostrato sia dagli exchanges che dalle aziende interessate a creare un security token risulta chiaro ma per capire le dimensioni di questo fenomeno ne verranno presentati alcuni dati quantitativi nel prossimo capitolo.

Ad oggi non esistono standard specifici per le IEOs: è a discrezione dell'exchange valutare quale degli standard già presenti si adatti meglio alla sua

infrastruttura. Solitamente un exchange ha già la necessità di supportare gli standard utilizzati nelle STOs ed è quindi ragionevole pensare che continui ad essere adottato uno degli standard già disponibili.

3.2 Libra

Libra è una criptovaluta e un sistema di pagamento proposto da Facebook nel 2019. Al momento la data di lancio del progetto è prevista per il 2020 ma sono già stati espressi commenti riguardo a possibili ritardi. A livello tecnico libra è una criptovaluta a cui corrisponde un sottostante composto da un portafoglio di valute al fine di limitarne la volatilità. La gestione di Libra sarà assegnata ad un consorzio, *Libra Association*, che per ora conta 27 membri. Ogni membro verserà 10 milioni di dollari per poter garantire il valore della criptovaluta. Libra è il primo caso in cui un'azienda privata di tali dimensioni mira ad emettere un token utilizzabile in un contesto internazionale così amplio. In questo senso Libra sta svolgendo il ruolo di apripista laddove altre criptovaluta hanno fallito a causa di scarsa rilevanza o a causa di uno scarso interesse del legislatore. Citando le parole di Mark Zuckerberg come riportate da *The Verge*:

".. And we have this bigger, or at least more exotic, project around Libra, which is to try to stand up a new kind of digital money that can work globally, [and] that will be stable ... But it's a big idea, and it's a new type of system, especially to be implemented by big companies. We're not the only ones doing this. We've led that the thinking and development on it so far, but the idea is to do this as an independent association, which is what we announced with about 27 other companies. By the time it launches, we expect we'll have 100 or more companies as part of it. But part of what we're trying to do overall on these big projects now that touch very socially important aspects of society is have a more consultative approach. So not just show up and say, "Alright, here we're launching this. here's a product, your app got updated, now you can start buying Libras and sending them around." We want to

make sure. We get that there are real issues. Finance is a very heavily regulated space. There's a lot of important issues that need to be dealt with in preventing money laundering, preventing financing of terrorists and people who the different governments say you can't do business with. There are a lot of requirements on knowing who your customers are. We already focus a lot on real identity, across especially Facebook, so there's even more that we need to do in order to have this kind of a product. And we're committed to doing that well, and part of doing that well is not just building the internal tools and showing up and saying, "Hey, we think we've solved this," but addressing and meeting with all the regulators up front, hearing their concerns, hearing what they think we should be doing, making sure other folks in the consortium are handling this appropriately. .."

Risulta quindi chiaro come il focus dei creatori di Libra non sia l'aspetto tecnologico del progetto, bensì quello di avviare un dialogo con i legislatori. Questo aspetto rappresenta il più importante contributo innovativo apportato da Libra. Il progetto ha già ricevuto critiche ed una forte opposizione dalle banche centrali, da membri del governo statunitense e da parte del governo francese. Il ministro della finanza francese Bruno Le Maire ha dichiarato durante il *G7* che la Francia non permetterà lo sviluppo di Libra poiché rappresenta una minaccia alla sovranità monetaria delle nazioni europee. Pareri altrettanto preoccupanti sono stati espressi dagli altri ministri riuniti al *G7*. Risulta palese che l'introduzione di Libra andrà a fare chiarezza sullo stato delle regolamentazioni anche a livello internazionale. Questo passaggio è probabilmente il fattore fondamentale per determinare le possibilità di espansione nei prossimi anni del mercato degli strumenti finanziari basati su tecnologia blockchain.

Capitolo 4

Analisi delle Security Token Offering

Ad oggi sono stati pubblicati solamente due studi empirici riguardo le STOs. Il primo studio, pubblicato da Ante e Fielder, si avvale di un *data-set* composto da 151 security tokens, o meglio tokens che svolgono il ruolo di securities, in quanto non tutte soddisfano necessariamente le normative riguardanti le securities. Ad esempio, tra i tokens analizzati nel suddetto studio è considerato il token *DAO* che è stato riconosciuto nel 2017 come security dalla *SEC*. Tuttavia il token *DAO* non è mai stato autorizzato per l'emissione. Allo stesso modo molti altri tokens considerati non sono mai stati registrati come securities. Di conseguenza, includere queste ICOs in un'analisi empirica delle STOs risulta inappropriato poiché sia gli issuer che gli investitori non erano al corrente di operare con delle securities.

Il secondo studio, condotto da Florie Mazzorana-Kremer, analizza un ristretto sottoinsieme di STOs selezionate secondo criteri specifici con il fine di analizzare il *turnover* dei tokens. Questo studio presenta delle conclusioni particolarmente rilevanti sebbene l'autrice stessa esprime le grandi limitazioni di un analisi empirica di un fenomeno così recente. La maggior parte delle STOs che rispettano i canoni definiti nel presente elaborato infatti sono limitate al trading per investitori accreditati o sono limitate a giurisdizioni

specifiche. Ciò presenta un problema per la raccolta dei dati storici, anch'essi spesso disponibili solo a investitori accreditati. In primo luogo, ai fini di quest'analisi, verranno analizzati fonti secondarie che permettono di fornire un'interessante panoramica del fenomeno delle STOs. Avendo l'obiettivo di fornire un contributo originale, in seguito si analizzeranno non dati di trading bensì dati riguardo le caratteristiche di un sottoinsieme di STOs.

4.1 Analisi dei dati

4.1.1 5th ICO/STO Report

Una fonte secondaria che risulta molto utile per comprendere meglio il fenomeno delle STOs è il report pubblicato da PWC in associazione con CryptoValley. Questo report, giunto alla quinta edizione mira a delineare con chiarezza l'evoluzione del fenomeno delle ICOs, delle STOs e delle IEOs. Sebbene le fonti di dati primarie utilizzate dallo studio non siano state rese pubbliche, è ragionevole assumere la bontà dei risultati in quanto congrui con le osservazioni presentate in questo elaborato. La definizione adottata nel report per identificare se una token sale sia o meno una STO è:

"Security Token Offering ("STO") is a sale of tokens with features comparable to normal securities, i.e., fully regulated and approved within at least one jurisdiction."

Ciò risulta essere in linea con la definizione adottata in questo elaborato ed è quindi possibile procedere a presentare i risultati di questo report.

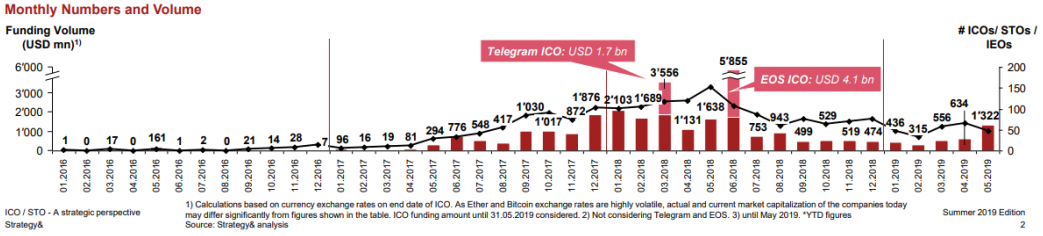

Figura 4.1: Token sales data

Per prima cosa osserviamo che i dati presentati in figura 4.1 sono in linea con i dati osservati in figura 1.2. Successivamente, in ambito di STOs, osserviamo la figura 4.2.

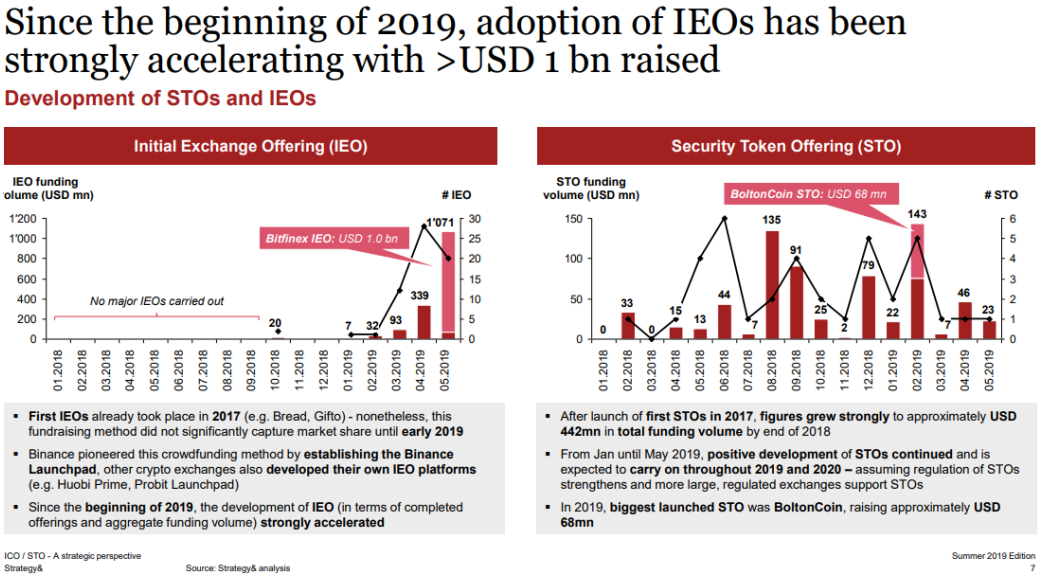

Figura 4.2: IEO e STO

Le informazioni esposte permettono di associare dei dati quantitativi al fenomeno di cui si è discusso fino ad ora. Nel report si stima che il volume delle STOs dalla fine del 2017 alla fine del 2018 sia pari a 442 milioni di dollari.

4.1.2 Dati delle STOs Polymath

La seguente serie di dati riguarda i tokens emessi tramite la piattaforma Polymath. La fonte dei dati non è ufficiale e non è pubblicata o mantenuta da Polymath, bensì opera di un membro della community. Nonostante ciò è possibile verificare l'elenco dei tokens emessi e le loro caratteristiche grazie alla struttura della piattaforma Polymath e alla natura pubblica della blockchain Ethereum. Nei dati sono compresi nove tokens di test emessi dalla stessa azienda per verificare il corretto funzionamento della piattaforma. Inoltre,

è possibile notare come alcuni tokens siano stati aggiornati dalla versione 2.0.0 basata sullo standard ST-20 alla versione 3.0.0 basata sullo standard ERC-1400.

Tabella 4.1: Polymath STO data

Number	Symbol	Version	Granularity	Total Supply	Investor count
1	WE.R.LIVE	2.0.0	1	773,617	23
2	ANN	2.0.0	1.00E+18	0	0
3	7PASS	2.0.0	1	0	0
4	EXAT	2.0.0	1.00E+18	0	0
5	BHG	2.0.0	1	0	0
6	BHG	2.0.0	1	1,000,000,000	1
7	RMT	2.0.0	1	0	0
8	BUFF	2.0.0	1	0	0
9	RMT	2.0.0	1	0	0
10	CRYP	2.0.0	1.00E+18	0	0
11	ITALY	2.0.0	1	0	0
12	SLASH	2.0.0	1.00E+18	0	0
13	TRANXIT	2.0.0	1	0	0
14	FACT	2.0.0	1.00E+18	0	0
15	MSC	2.0.0	1	0	0
16	MSC	2.0.0	1	100,000	1
17	LAND	2.0.0	1	0	0
18	GSE	2.0.0	1	0	0
19	GSE	2.0.0	1	0	0
20	BLABS	2.0.0	1.00E+18	0	0
21	SONATA	2.0.0	1.00E+18	0	0
22	BKYA	2.0.0	1.00E+18	0	0
23	CNODE	2.0.0	1.00E+18	0	0
24	EGX	2.0.0	1.00E+18	0	0
25	NTVL	2.0.0	1.00E+18	45,000,000	1

Tabella 4.1 continuata dalla pagina precedente

Number	Symbol	Version	Granularity	Total Supply	Investor count
26	EXRP	2.0.0	1	0	0
27	DECS	2.0.0	1.00E+18	99,000,000	1
28	JET	2.0.0	1.00E+18	0	0
29	GLX	2.0.0	1.00E+18	0	0
30	BRIXCORP	2.0.0	1.00E+18	0	0
31	KEEPTOKEN	2.0.0	1	0	0
32	SAAKO	2.0.0	1	0	0
33	X	2.0.0	1.00E+18	0	0
34	CLT	2.0.0	1	0	0
35	LOVE	2.0.0	1	144,122,222	5
36	PYA	2.0.0	1.00E+18	260,000,000	1
37	RGTST	2.0.0	1.00E+18	0	0
38	BLANK	2.0.0	1.00E+18	0	0
39	PERRY-A	2.0.0	1	0	0
40	DBX-A	2.0.0	1	0	0
41	RISEPIC-A	2.0.0	1.00E+18	0	0
42	VEGAN	2.0.0	1	0	0
43	KIZGLOBAL	2.0.0	1	0	0
44	SEPAT	2.0.0	1	0	0
45	MJKJ	2.0.0	1	0	0
46	TCX	2.0.0	1	0	0
47	MINDS	2.0.0	1.00E+18	3,000,000	2
48	OX	2.0.0	1	1,500	2
49	ANGUS	2.0.0	1	120	1
50	KOBE	2.0.0	1.00E+18	0	0
51	JD-1113-00	2.0.0	1	768,900	4
52	CAMMAZOL	2.0.0	1	9,992,590	10
53	BRANGUS	2.0.0	1	0	0
54	RIN	2.0.0	1	0	0

Tabella 4.1 continuata dalla pagina precedente

Number	Symbol	Version	Granularity	Total Supply	Investor count
55	NEST	2.0.0	1	0	0
56	RMT	2.0.0	1	40,000	4
57	XHOR	2.0.0	1.00E+18	0	0
58	CTEST	2.0.0	1	258,300	5
59	BEEFALO	2.0.0	1	66,855	1
60	BUY	2.0.0	1	50,000,000	31
61	KIZGT	2.0.0	1	150,000,000	2
62	JTSN	2.0.0	1.00E+18	0	0
63	NOTE	2.0.0	1.00E+18	20,000,000	2
64	AW	2.0.0	1.00E+18	40,000,000,000	49
65	SWRD	2.0.0	1	100,000,000	1
66	B1X	2.0.0	1.00E+18	0	0
67	NUCOIN	2.0.0	1	0	0
68	QRY	2.0.0	1	0	0
69	MBTX	2.0.0	1	3,401,125	133
70	MHST	2.0.0	1.00E+18	0	0
71	OVIS	2.0.0	1	0	0
72	NEW	2.0.0	1	0	0
73	KGT	2.0.0	1	150,000,000	285
74	TRBN	2.0.0	1	0	0
75	XS	2.0.0	1.00E+18	100,000,000	2
76	XA	2.0.0	1.00E+18	50,000,000	2
77	WNDR	2.0.0	1	0	0
78	IDGT	2.0.0	1	10,000,000	2
79	BNYX	2.0.0	1	5,000,100	3
80	TIANA	2.0.0	1	0	0
81	DCAP	2.0.0	1	0	0
82	INR	2.0.0	1.00E+18	500,070,009	440
83	VPNSTK	2.0.0	1.00E+18	0	0

Tabella 4.1 continuata dalla pagina precedente

Number	Symbol	Version	Granularity	Total Supply	Investor count
84	ROBOT	2.0.0	1.00E+18	0	0
85	MINERAL	2.0.0	1	105,710,000	44
86	PAL	2.0.0	1.00E+18	0	0
87	CHRIS	2.0.0	1	0	0
88	RSF	2.0.0	1	0	0
89	ASLT	2.0.0	1.00E+18	1,000	2
90	JOYA1	2.0.0	1	1,000,000	6
91	EMI	2.0.0	1	0	0
92	BOB-S	2.0.0	1	0	0
93	AFK	2.0.0	1.00E+18	0	0
94	PI	2.0.0	1.00E+18	500,000,000	2
95	XX	2.0.0	1	1,001,050	2
96	FLETCH	2.0.0	1.00E+18	0	0
97	MOBITOKEN	2.0.0	1	0	0
98	NU	2.0.0	1.00E+18	0	0
99	FARM	2.0.0	1	0	0
100	ABODS	2.0.0	1	0	0
101	ABODS-A	2.0.0	1.00E+18	0	0
102	ABODS-D	2.0.0	1.00E+18	0	0
103	CF	2.0.0	1.00E+18	0	0
104	X4C	2.0.0	1.00E+18	6,500,000	5
105	FCK	2.0.0	1	0	0
106	RBS	2.0.0	1	0	0
107	ZRS	2.0.0	1	850,000,000	11
108	HMV-V	2.0.0	1	0	0
109	SNTX	2.0.0	1.00E+18	0	0
110	DSTS	2.0.0	1.00E+18	0	0
111	DSX	2.0.0	1.00E+18	0	0
112	HMV-P	2.0.0	1	0	0

Tabella 4.1 continuata dalla pagina precedente

Number	Symbol	Version	Granularity	Total Supply	Investor count
113	PHK	2.0.0	1	100,000	1
114	FLOT	2.0.0	1	0	0
115	CXGLD	2.0.0	1	0	0
116	CXSLV	2.0.0	1	0	0
117	OSQO	2.0.0	1	109,900,017	27
118	BIU	2.0.0	1.00E+18	0	0
119	PRIVKEY	2.0.0	1.00E+18	0	0
120	CET	2.0.0	1.00E+18	0	0
121	ZONTLE	2.0.0	1.00E+18	0	0
122	SANA	2.0.0	1	64,003,255	22
123	GMI	2.0.0	1.00E+18	8,000,000	1
124	USDL	2.0.0	1	0	0
125	BBE	2.0.0	1	0	0
126	RBTET	2.0.0	1	0	0
127	SIKZ	2.0.0	1	0	0
128	ALUX	2.0.0	1.00E+18	15,000,000	3
129	QUAKE	2.0.0	1	50,500	5
130	RPRC	2.0.0	1.00E+18	0	0
131	MAPLE	2.0.0	1.00E+18	15,000,000	3
132	SEX	2.0.0	1	1,000,000,000	1
133	GRNRBR	2.0.0	1.00E+18	0	0
134	CTODIV1	2.0.0	1	1,011,615	5
135	BRBI	2.0.0	1	0	0
136	SCM	2.0.0	1	0	0
137	EURO	2.0.0	1	0	0
138	EXST	2.0.0	1.00E+18	0	0
139	EVG	2.0.0	1.00E+18	0	0
140	PSIMP	2.0.0	1	0	0
141	HEALTH	2.0.0	1	0	0

Tabella 4.1 continuata dalla pagina precedente

Number	Symbol	Version	Granularity	Total Supply	Investor count
142	BUSHNELL-A	2.0.0	1.00E+18	0	0
143	CROW	2.0.0	1.00E+18	0	0
144	BLT	2.0.0	1.00E+18	0	0
145	TIMER	2.0.0	1	0	0
146	HFH	2.0.0	1	500,000,000	3
147	ZIYENCOIN	2.0.0	1.00E+18	64,503,625	24
148	RUDRA	2.0.0	1.00E+18	0	0
149	V3DOTZERO	3.0.0	1	1,032	7
150	MTLSTR	3.0.0	1	100,000,000	1
151	COSERA	3.0.0	1	100,000,000	1
152	BNKR	3.0.0	1	100,000,000	1
153	RNTM	3.0.0	1	0	1
154	PCTFM	3.0.0	1	0	0
155	CON	3.0.0	1	512,600	4
156	DSI	3.0.0	1.00E+18	0	0
157	99EAST	3.0.0	1	100,000,000	1
158	BZBK	3.0.0	1	0	1
159	BLABS	3.0.0	1	100,000,000	1
160	BKYA	3.0.0	1	100,000,000	1
161	CNODE	3.0.0	1	100,000,000	1
162	SONATA	3.0.0	1	100,000,000	1
163	BRIXCORP	3.0.0	1	100,000,000	1
164	VPNSTK	3.0.0	1	100,000,000	1
165	ROBOT	3.0.0	1	100,000,000	1
166	SNTX	3.0.0	1.00E+00	100,000,000	1
167	DSTS	3.0.0	1	0	0
168	DSX	3.0.0	1	100,000,000	1

Tabella 4.1 continuata dalla pagina precedente

Number	Symbol	Version	Granularity	Total Supply	Investor count
169	AFK	3.0.0	1	100,000,000	1
170	GRNRBR	3.0.0	1	100,000,000	1
171	CXGLD	3.0.0	1	100,000,000	1
172	CXSLV	3.0.0	1	100,000,000	1
173	EGX	3.0.0	1	100,000,000	93
174	USDL	3.0.0	1	0	0

Di seguito viene presentata la metodologia di analisi e i risultati della stessa.

```
STOData <- read.csv(file="token.csv", header=TRUE, sep=",");
length(count.fields("token.csv")) - 1

## [1] 174
```

```
boxplot(STOData$Investor, main = "Investors",horizontal = TRUE)
```

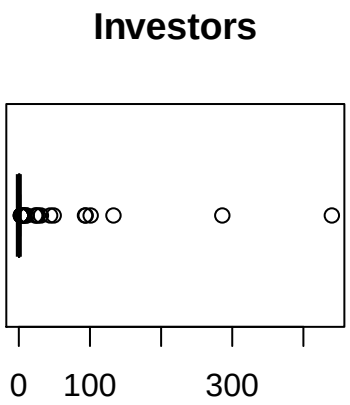

Figura 4.3: Investors

La figura 4.3 è un semplice boxplot dei dati sul numero degli investitori. Possiamo osservare graficamente che il numero degli investitori è tendenzialmente ridotto. In particolare:

```
sum(STOData$Investor==0)

## [1] 107

sum(STOData$Investor==1)

## [1] 29

summary(STOData$Investor>1)

##    Mode   FALSE    TRUE
## logical     136      38
```

Questi risultati indicano come in 107 casi nessun investitore possieda il token (caso tipico di una STO non finalizzata) e come in 29 casi solo un investitore possieda il token (caso tipico di una security token offering con minting senza successo). Per quanto riguarda le STOs che sono state lanciate con successo osserviamo che:

```
summary(subset(STOData, Investor > 1)$Investor)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00    2.25    5.00   38.13   26.50  440.00
```

```
plot(subset(STOData, Investor > 1)$Investor, main = "Investors")
```

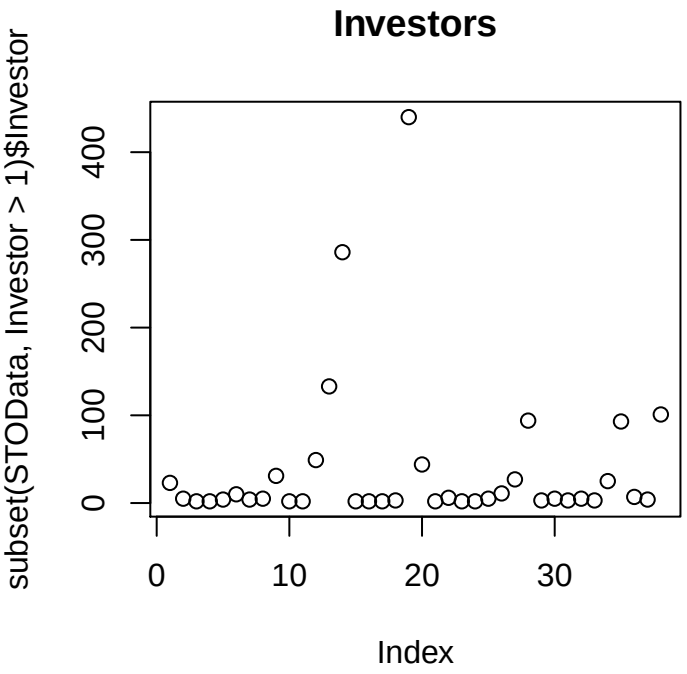

Figura 4.4: Investors in successful STO

In figura 4.4 si osserva come nonostante l'esclusione dei dati riguardanti le STOs non riuscite il numero di investitori sia comunque ridotto. Questo dato tuttavia non ci da indicazioni sull'effettivo successo o insuccesso di una STO in quanto non abbiamo informazioni effettive sulla quantità di tokens acquistati e sul prezzo di vendita.

Un altro elemento di analisi da considerare è la granularità del token che ne permette il frazionamento.

```
sum(STOData$Investor>1)

## [1] 38

sum(subset(STOData, Investor > 1)$Granularity=="1")

## [1] 25
```

Tra le STOs di successo, definite in base ad un numero di investitori maggiore di uno, in 25 casi su 38 il token non è frazionabile. Questo indica che in questo campione l'assenza della frazionabilità non è un fattore limitante per quanto riguarda la possibilità di investimento. Tuttavia, essendo una funzionalità relativamente banale da implementare, risulta comunque essere una caratteristica che può fornire un valore aggiunto anche se marginale.

Un ultimo dato da considerare è il numero massimo di tokens. Possiamo osservare che:

```
options(scipen = 999)
summary(as.numeric(gsub(",", "",
subset(STOData, Investor > 1)$Total)))

##        Min.     1st Qu.      Median        Mean     3rd Qu.         Max.
##        1000     1000262    15000000  1146316645   104282500 40000000000
```

Questo ci da un'indicazione utile per spiegare lo scarso interesse per la frazionabilità. Infatti, con un numero di investitori limitato e in media una grande quantità di tokens, la frazionabilità può risultare superflua.

Conclusioni

Le STOs rappresentano una soluzione tecnologica interessante che ha ampie prospettive di espansione. In conclusione, l'analisi di questo fenomeno ha chiarito alcuni aspetti sul perché le STOs siano nate e perché si siano diffuse; grazie all'approfondimento delle tecniche di implementazione inoltre si è potuto delineare un panorama complessivo delle funzionalità che le STOs offrono e i differenti casi d'uso in cui risulta possibile utilizzarle.

Nonostante la scarsità dei dati disponibili, dovuti sia alla modernità delle STOs che al valore economico degli stessi dati, è stato dimostrato come un'analisi empirica, seppur limitata, può chiarire alcune tendenze osservabili in questo mercato. Tuttavia sarebbe azzardato considerare l'analisi effettuata come soddisfacente: è auspicabile che in lavori futuri sia possibile raccogliere una quantità di dati maggiore.

Grazie alle funzionalità di cui dispongono, le STO e le IEOs, che altro non sono che un particolare sottoinsieme delle STOs, hanno la potenzialità di espandersi in un mercato molto amplio. In particolare, la spinta legislativa dovuta a fattori esterni e la facilità di esecuzione offerta dalle IEOs fanno presumere che questa modalità di raccolta di capitale continuerà a diffondersi, anche se non è possibile sostenere l'affermazione con dati empirici.

Appendice A

ERC-20 Interface Implementation

```
1  pragma solidity ^0.4.24;
2
3  /**
4   * @title ERC20 interface
5   * @dev see https://github.com/ethereum/EIPs/issues/20
6   */
7  interface IERC20 {
8    function totalSupply() external view returns (uint256);
9
10   function balanceOf(address who) external view returns (
       uint256);
11
12   function allowance(address owner, address spender)
13     external view returns (uint256);
14
15   function transfer(address to, uint256 value) external
       returns (bool);
16
17   function approve(address spender, uint256 value)
18     external returns (bool);
19
20   function transferFrom(address from, address to, uint256
       value)
21     external returns (bool);
```

```
22
23   event Transfer(
24     address indexed from,
25     address indexed to,
26     uint256 value
27   );
28
29   event Approval(
30     address indexed owner,
31     address indexed spender,
32     uint256 value
33   );
34 }
```

Listing A.1: OpenZeppelin ERC-20 Inferface Implementation

```
1  // Abstract contract for the full ERC 20 Token standard
2  // https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.
       md
3  pragma solidity ^0.4.21;
4
5
6  contract EIP20Interface {
7      /* This is a slight change to the ERC20 base standard.
8      function totalSupply() constant returns (uint256 supply);
9      is replaced with:
10     uint256 public totalSupply;
11     This automatically creates a getter function for the
           totalSupply.
12     This is moved to the base contract since public getter
           functions are not
13     currently recognised as an implementation of the matching
            abstract
14     function by the compiler.
15     */
16     /// total amount of tokens
17     uint256 public totalSupply;
18
```

```
19      /// @param _owner The address from which the balance will
            be retrieved
20      /// @return The balance
21      function balanceOf(address _owner) public view returns (
            uint256 balance);
22
23      /// @notice send `_value` token to `_to` from `msg.sender
            `
24      /// @param _to The address of the recipient
25      /// @param _value The amount of token to be transferred
26      /// @return Whether the transfer was successful or not
27      function transfer(address _to, uint256 _value) public
            returns (bool success);
28
29      /// @notice send `_value` token to `_to` from `_from` on
            the condition it is approved by `_from`
30      /// @param _from The address of the sender
31      /// @param _to The address of the recipient
32      /// @param _value The amount of token to be transferred
33      /// @return Whether the transfer was successful or not
34      function transferFrom(address _from, address _to, uint256
            _value) public returns (bool success);
35
36      /// @notice `msg.sender` approves `_spender` to spend `
            _value` tokens
37      /// @param _spender The address of the account able to
            transfer the tokens
38      /// @param _value The amount of tokens to be approved for
            transfer
39      /// @return Whether the approval was successful or not
40      function approve(address _spender, uint256 _value) public
            returns (bool success);
41
42      /// @param _owner The address of the account owning
            tokens
43      /// @param _spender The address of the account able to
            transfer the tokens
44      /// @return Amount of remaining tokens allowed to spent
```

```
45      function allowance(address _owner, address _spender)
            public view returns (uint256 remaining);
46
47      // solhint-disable-next-line no-simple-event-func-name
48      event Transfer(address indexed _from, address indexed _to
            , uint256 _value);
49      event Approval(address indexed _owner, address indexed
            _spender, uint256 _value);
50  }
```

Listing A.2: ConsenSys ERC-20 Inferface Implementation

Appendice B

ST-20 Interface

```
1  pragma solidity ^0.4.24;
2
3  /**
4   * @title Interface for all security tokens
5   */
6  interface ISecurityToken {
7
8      // Standard ERC20 interface
9      function decimals() external view returns (uint8);
10     function totalSupply() external view returns (uint256);
11     function balanceOf(address _owner) external view returns
           (uint256);
12     function allowance(address _owner, address _spender)
           external view returns (uint256);
13     function transfer(address _to, uint256 _value) external
           returns (bool);
14     function transferFrom(address _from, address _to, uint256
           _value) external returns (bool);
15     function approve(address _spender, uint256 _value)
           external returns (bool);
16     function decreaseApproval(address _spender, uint
           _subtractedValue) external returns (bool);
17     function increaseApproval(address _spender, uint
           _addedValue) external returns (bool);
```

```
18      event Transfer(address indexed from, address indexed to,
            uint256 value);
19      event Approval(address indexed owner, address indexed
            spender, uint256 value);
20
21      //transfer, transferFrom must respect the result of
            verifyTransfer
22      function verifyTransfer(address _from, address _to,
            uint256 _value) external returns (bool success);
23
24      /**
25       * @notice Mints new tokens and assigns them to the
             target _investor.
26       * Can only be called by the STO attached to the token (
             Or by the ST owner if there's no STO attached yet)
27       * @param _investor Address the tokens will be minted to
28       * @param _value is the amount of tokens that will be
             minted to the investor
29       */
30      function mint(address _investor, uint256 _value) external
             returns (bool success);
31
32      /**
33       * @notice Mints new tokens and assigns them to the
             target _investor.
34       * Can only be called by the STO attached to the token (
             Or by the ST owner if there's no STO attached yet)
35       * @param _investor Address the tokens will be minted to
36       * @param _value is The amount of tokens that will be
             minted to the investor
37       * @param _data Data to indicate validation
38       */
39      function mintWithData(address _investor, uint256 _value,
            bytes _data) external returns (bool success);
40
41      /**
42       * @notice Used to burn the securityToken on behalf of
             someone else
```

```
43      * @param _from Address for whom to burn tokens
44      * @param _value No. of tokens to be burned
45      * @param _data Data to indicate validation
46      */
47     function burnFromWithData(address _from, uint256 _value,
           bytes _data) external;
48
49     /**
50      * @notice Used to burn the securityToken
51      * @param _value No. of tokens to be burned
52      * @param _data Data to indicate validation
53      */
54     function burnWithData(uint256 _value, bytes _data)
           external;
55
56     event Minted(address indexed _to, uint256 _value);
57     event Burnt(address indexed _burner, uint256 _value);
58
59     // Permissions this to a Permission module, which has a
           key of 1
60     // If no Permission return false - note that IModule
           withPerm will allow ST owner all permissions anyway
61     // this allows individual modules to override this logic
           if needed (to not allow ST owner all permissions)
62     function checkPermission(address _delegate, address
           _module, bytes32 _perm) external view returns (bool);
63
64     /**
65      * @notice Returns module list for a module type
66      * @param _module Address of the module
67      * @return bytes32 Name
68      * @return address Module address
69      * @return address Module factory address
70      * @return bool Module archived
71      * @return uint8 Module type
72      * @return uint256 Module index
73      * @return uint256 Name index
74
```

```
75      */
76     function getModule(address _module) external view returns
           (bytes32, address, address, bool, uint8, uint256,
           uint256);
77
78     /**
79      * @notice Returns module list for a module name
80      * @param _name Name of the module
81      * @return address[] List of modules with this name
82      */
83     function getModulesByName(bytes32 _name) external view
           returns (address[]);
84
85     /**
86      * @notice Returns module list for a module type
87      * @param _type Type of the module
88      * @return address[] List of modules with this type
89      */
90     function getModulesByType(uint8 _type) external view
           returns (address[]);
91
92     /**
93      * @notice Queries totalSupply at a specified checkpoint
94      * @param _checkpointId Checkpoint ID to query as of
95      */
96     function totalSupplyAt(uint256 _checkpointId) external
           view returns (uint256);
97
98     /**
99      * @notice Queries balance at a specified checkpoint
100     * @param _investor Investor to query balance for
101     * @param _checkpointId Checkpoint ID to query as of
102     */
103    function balanceOfAt(address _investor, uint256
           _checkpointId) external view returns (uint256);
104
105    /**
```

```
106         * @notice Creates a checkpoint that can be used to query
                historical balances / totalSuppy
107         */
108        function createCheckpoint() external returns (uint256);
109
110        /**
111         * @notice Gets length of investors array
112         * NB - this length may differ from investorCount if the
                list has not been pruned of zero-balance investors
113         * @return Length
114         */
115        function getInvestors() external view returns (address[])
               ;
116
117        /**
118         * @notice returns an array of investors at a given
                checkpoint
119         * NB - this length may differ from investorCount as it
                contains all investors that ever held tokens
120         * @param _checkpointId Checkpoint id at which investor
                list is to be populated
121         * @return list of investors
122         */
123        function getInvestorsAt(uint256 _checkpointId) external
               view returns(address[]);
124
125        /**
126         * @notice generates subset of investors
127         * NB - can be used in batches if investor list is large
128         * @param _start Position of investor to start iteration
                from
129         * @param _end Position of investor to stop iteration at
130         * @return list of investors
131         */
132        function iterateInvestors(uint256 _start, uint256 _end)
               external view returns(address[]);
133
134        /**
```

```
135         * @notice Gets current checkpoint ID
136         * @return Id
137         */
138        function currentCheckpointId() external view returns (
               uint256);
139
140        /**
141        * @notice Gets an investor at a particular index
142        * @param _index Index to return address from
143        * @return Investor address
144        */
145        function investors(uint256 _index) external view returns
               (address);
146
147       /**
148        * @notice Allows the owner to withdraw unspent POLY
               stored by them on the ST or any ERC20 token.
149        * @dev Owner can transfer POLY to the ST which will be
               used to pay for modules that require a POLY fee.
150        * @param _tokenContract Address of the ERC20Basic
               compliance token
151        * @param _value Amount of POLY to withdraw
152        */
153        function withdrawERC20(address _tokenContract, uint256
               _value) external;
154
155        /**
156        * @notice Allows owner to approve more POLY to one of the
                modules
157        * @param _module Module address
158        * @param _budget New budget
159        */
160        function changeModuleBudget(address _module, uint256
               _budget) external;
161
162        /**
163         * @notice Changes the tokenDetails
164         * @param _newTokenDetails New token details
```

```
165      */
166     function updateTokenDetails(string _newTokenDetails)
            external;
167
168     /**
169     * @notice Allows the owner to change token granularity
170     * @param _granularity Granularity level of the token
171     */
172     function changeGranularity(uint256 _granularity) external
            ;
173
174     /**
175     * @notice Removes addresses with zero balances from the
            investors list
176     * @param _start Index in investors list at which to start
             removing zero balances
177     * @param _iters Max number of iterations of the for loop
178     * NB - pruning this list will mean you may not be able to
             iterate over investors on-chain as of a historical
            checkpoint
179     */
180     function pruneInvestors(uint256 _start, uint256 _iters)
            external;
181
182     /**
183      * @notice Freezes all the transfers
184      */
185     function freezeTransfers() external;
186
187     /**
188      * @notice Un-freezes all the transfers
189      */
190     function unfreezeTransfers() external;
191
192     /**
193      * @notice Ends token minting period permanently
194      */
195     function freezeMinting() external;
```

```
196
197     /**
198      * @notice Mints new tokens and assigns them to the
            target investors.
199      * Can only be called by the STO attached to the token or
             by the Issuer (Security Token contract owner)
200      * @param _investors A list of addresses to whom the
            minted tokens will be delivered
201      * @param _values A list of the amount of tokens to mint
            to corresponding addresses from _investor[] list
202      * @return Success
203      */
204     function mintMulti(address[] _investors, uint256[]
           _values) external returns (bool success);
205
206     /**
207      * @notice Function used to attach a module to the
            security token
208      * @dev  E.G.: On deployment (through the STR) ST gets a
            TransferManager module attached to it
209      * @dev to control restrictions on transfers.
210      * @dev You are allowed to add a new moduleType if:
211      * @dev - there is no existing module of that type yet
            added
212      * @dev - the last member of the module list is
            replacable
213      * @param _moduleFactory is the address of the module
            factory to be added
214      * @param _data is data packed into bytes used to further
             configure the module (See STO usage)
215      * @param _maxCost max amount of POLY willing to pay to
            module. (WIP)
216      */
217     function addModule(
218         address _moduleFactory,
219         bytes _data,
220         uint256 _maxCost,
221         uint256 _budget
```

```
222     ) external;
223
224     /**
225     * @notice Archives a module attached to the SecurityToken
226     * @param _module address of module to archive
227     */
228     function archiveModule(address _module) external;
229
230     /**
231     * @notice Unarchives a module attached to the
            SecurityToken
232     * @param _module address of module to unarchive
233     */
234     function unarchiveModule(address _module) external;
235
236     /**
237     * @notice Removes a module attached to the SecurityToken
238     * @param _module address of module to archive
239     */
240     function removeModule(address _module) external;
241
242     /**
243      * @notice Used by the issuer to set the controller
            addresses
244      * @param _controller address of the controller
245      */
246     function setController(address _controller) external;
247
248     /**
249      * @notice Used by a controller to execute a forced
            transfer
250      * @param _from address from which to take tokens
251      * @param _to address where to send tokens
252      * @param _value amount of tokens to transfer
253      * @param _data data to indicate validation
254      * @param _log data attached to the transfer by
            controller to emit in event
255      */
```

```
256     function forceTransfer(address _from, address _to,
            uint256 _value, bytes _data, bytes _log) external;
257
258     /**
259      * @notice Used by a controller to execute a foced burn
260      * @param _from address from which to take tokens
261      * @param _value amount of tokens to transfer
262      * @param _data data to indicate validation
263      * @param _log data attached to the transfer by
             controller to emit in event
264      */
265     function forceBurn(address _from, uint256 _value, bytes
            _data, bytes _log) external;
266
267     /**
268      * @notice Used by the issuer to permanently disable
             controller functionality
269      * @dev enabled via feature switch "
             disableControllerAllowed"
270      */
271      function disableController() external;
272
273      /**
274      * @notice Used to get the version of the securityToken
275      */
276      function getVersion() external view returns(uint8[]);
277
278      /**
279      * @notice Gets the investor count
280      */
281      function getInvestorCount() external view returns(
             uint256);
282
283      /**
284       * @notice Overloaded version of the transfer function
285       * @param _to receiver of transfer
286       * @param _value value of transfer
287       * @param _data data to indicate validation
```

```
288         * @return bool success
289         */
290        function transferWithData(address _to, uint256 _value,
               bytes _data) external returns (bool success);
291
292        /**
293         * @notice Overloaded version of the transferFrom
                function
294         * @param _from sender of transfer
295         * @param _to receiver of transfer
296         * @param _value value of transfer
297         * @param _data data to indicate validation
298         * @return bool success
299         */
300        function transferFromWithData(address _from, address _to
               , uint256 _value, bytes _data) external returns(bool)
               ;
301
302        /**
303         * @notice Provides the granularity of the token
304         * @return uint256
305         */
306        function granularity() external view returns(uint256);
307    }
```

Listing B.1: Polymath ST-20 Interface

Bibliografia

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," p. 9.

[2] Merkle, R. C. (1978). Secure communications over insecure channels. Communications of the ACM, 21(4), 294-299.

[3] Diffie, W., and Hellman, M. New directions in cryptography. IEEE Trans. on Inform. IT-22, 6 (Nov. 1976), 644-654.

[4] Merkle, R. C. (1987, August). A digital signature based on a conventional encryption function. In Conference on the theory and application of cryptographic techniques (pp. 369-378). Springer, Berlin, Heidelberg.

[5] Bayer, D., Haber, S., & Stornetta, W. S. (1993). Improving the efficiency and reliability of digital time-stamping. In Sequences Ii (pp. 329-334). Springer, New York, NY.

[6] Chaum, D. (1983). Blind signatures for untraceable payments. In Advances in cryptology (pp. 199-203). Springer, Boston, MA.

[7] Chaum, D., Fiat, A., & Naor, M. (1988, August). Untraceable electronic cash. In Conference on the Theory and Application of Cryptography (pp. 319-327). Springer, New York, NY.

[8] Schoenmakers, B. (1998). Security Aspects of the EcashTM Payment System. In State of the Art in Applied Cryptography: Course on Computer Security and Industrial Cryptography Leuven, Belgium, June 3–6, 1997

Revised Lectures (pp. 338–352). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-49248-8_16

[9] Back, A. (2002). Hashcash-a denial of service counter-measure.

[10] Dwork, C., & Naor, M. (1992, August). Pricing via processing or combatting junk mail. In Annual International Cryptology Conference (pp. 139-147). Springer, Berlin, Heidelberg.

[11] Jakobsson, M., & Juels, A. (1999). Proofs of work and bread pudding protocols. In Secure Information Networks (pp. 258-272). Springer, Boston, MA.

[12] Dai, W. (1998) B-Money. http://www.weidai.com/bmoney.txt

[13] Finney, H. (2004). Reusable proofs of work (rpow).

[14] Szabo, N. (2008). Bit gold.

[15] Grigg, I. (2017). On the intersection of Ricardian and Smart Contracts.

[16] Chohan, U. W. (2017). A history of bitcoin.

[17] Network, F. C. E. (2013). Application of FinCEN's regulations to persons administering, exchanging, or using virtual currencies. United States Department of the Treasury.

[18] De Filippi, P. (2014). Bitcoin: a regulatory nightmare to a libertarian dream. Internet Policy Review, 3(2).

[19] Ponsford, M. P. (2015). A comparative analysis of Bitcoin and other decentralised virtual currencies: legal regulation in the people's republic of China, Canada, and the United States. HKJ Legal Stud., 9, 29.

[20] Buterin, V. (2013). Ethereum white paper. GitHub repository, 22-23.

[21] Buterin V. A Prehistory of the Ethereum Protocol, https://vitalik.ca/general/2017/09/14/prehistory.html

[22] Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper, 151(2014), 1-32.

[23] Willett, J. R., Hidskes, M., Johnston, D., Gross, R., & Schneider, M. (2016). Omni Protocol Specification (formerly Mastercoin). white paper), accessed January, 28.

[24] CoinMarketCap, (2018). Cryptocurrency market capitalizations. Retrieved on September, 25, 2019.

[25] Jelurida, (2013), NextCoin, https://www.jelurida.com/nxt

[26] Counterparty, (2014), Counterparty. https://counterparty.io

[27] MaidSafe (2014), MaidsafeCoin. https://maidsafe.net/

[28] SwarmCity (2014), Swarm. https://swarm.city/

[29] Peterson, J., & Krug, J. (2015). Augur: a decentralized, open-source platform for prediction markets. arXiv preprint arXiv:1501.01042.

[30] Mazzorana-Kremer, F. (2019). Blockchain-Based Equity and STOs: Towards a Liquid Market for SME Financing?. Theoretical Economics Letters, 9(5), 1534-1552.

[31] Loss, L. (1983). Fundamentals of securities regulation. Aspen Publishers Online.

[32] Clayton, J. (2017). Statement on cryptocurrencies and initial coin offerings. world.

[33] Hughes, S. D. (2017). Cryptocurrency Regulations and Enforcement in the US. W. St. UL Rev., 45, 1.

[34] Lgs, D. (24). febbraio 1998, n. 58. Testo unico delle disposizioni in materia di intermediazione finanziaria, ai sensi degli articoli 8 e 21 della legge 6 febbraio 1996, (52), 9.

[35] Ante, L., & Fiedler, I. (2019). Cheap Signals in Security Token Offerings.

[36] Hayes, A. (2015). What factors give cryptocurrencies their value: An empirical analysis. Available at SSRN 2579445.

[37] Vogelsteller, F., & Buterin, V. (2015). Erc-20 token standard. Ethereum Foundation (Stiftung Ethereum), Zug, Switzerland.