Università degli Studi di Perugia
Dipartimento di Matematica e Informatica

Corso di Laurea in Informatica

Tesi di Laurea

Recupero di indirizzi bitcoin dal web

Laureando:
Alessio Santoru

Relatore:
Prof. Stefano Bistarelli

Anno Accademico 2015 – 2016

Indice

Introduzione **1**

1 Bitcoin: protocollo e criptovaluta **5**

1.1 Introduzione al mondo Bitcoin 5

1.2 Struttura della rete bitcoin 8

1.3 Blockchain 9

1.4 Indirizzi Bitcoin 11

1.4.1 Evitare il riutilizzo 13

1.4.2 Creazione di un indirizzo bitcoin 14

1.4.3 Validare un indirizzo bitcoin 19

1.5 Transazioni 19

1.5.1 Sistema dei resti e delle commissioni 20

1.6 Controllo e gestione del protocollo Bitcoin 23

2 Recupero di informazioni dal web **25**

2.1 Stato delle informazioni 26

2.2 Web Crawling 27

2.3 Web Scraping 31

2.3.1 Tipologie di web scraper 32

2.4 Confronto tra i più noti crawler/scraper 35

3 Sviluppo di un sistema per catalogare indirizzi bitcoin 39

3.1 Necessità di identificare i proprietari di portafogli bitcoin . . . 39

3.2 Un framework modulare 40

3.3 Analisi di una soluzione preesistente: Bitiodine 45

3.3.1 Differenze tra i due framework 47

4 Realizzazione di uno scraper di indirizzi bitcoin 48

4.1 PHPCrawl: Web Crawler Framework 49

4.1.1 Modalità di crawling con PHPCrawl 50

4.2 Uno scraper dinamico 52

4.3 CodeIgniter 52

4.4 Sviluppo di un'interfaccia grafica: Skeleton 54

4.5 Analisi delle funzionalità 55

4.6 Esecuzione del crawler 57

4.7 Esecuzione dello scraper 58

4.7.1 Verificare gli indirizzi bitcoin 59

4.8 Caricare una lista di pagine web 62

4.9 Utilizzo degli scraper di Bitiodine 63

4.9.1 Eseguire script python da php 64

4.9.2 Recupero dell'esecuzione di uno script python 65

4.10 Controllo di tutti i task in esecuzione e recenti 66

4.11 Struttura del database 67

5 Risultati 71

5.1 Test e validità 71

5.1.1 Situazioni a carico intensivo 72
5.2 Possibili miglioramenti 72

Conclusioni **74**

A Codice **76**

A.1 Algoritmo: Decodifica base58 76
A.2 Algoritmo: Validare indirizzi bitcoin 77

Bibliografia **79**

Elenco delle figure

1.1 Il famoso logo di Bitcoin 6
1.2 Grafico che mostra il cambio di valore di 1 BTC in Euro da agosto a dicembre 2016 8
1.3 Schema della rete bitcoin 9
1.4 Schema semplificato della blockchain 11
1.5 Uno schema che illustra la creazione di un indirizzo bitcoin a partire dalle chiavi ECDSA 18
1.6 Alice possiede 10 BTC in un blocco (ottenuti da un singolo output) 21
1.7 Alice trasferisce 10 BTC a Bob 21
1.8 Alice deve trasferire solo 5 BTC a Bob, ma il blocco contiene 10 BTC 22
1.9 Alice inserisce come output il suo stesso indirizzo, ricevendo 5 BTC di resto 22

2.1 Famiglie americane e utilizzo di internet 26
2.2 L'operazione base di un crawler è effettuare richieste HTTP per ottenere il contenuto della pagina e trovare collegamenti ad altre pagine 31

3.1 Funzionamento di un mixing service 43
3.2 Schema del framework sviluppato che mostra i 6 moduli . . . 44
3.3 Schema del framework BitIodine 46

4.1 Schermata di benvenuto del software 49
4.2 Il sistema a griglia di Skeleton che mostra come è possibile affiancare due contenuti su una riga 55
4.3 Digramma della struttura del software 57
4.4 Form principale di ricerca 58
4.5 Flowchart dell'algoritmo di validazione degli indirizzi bitcoin . 61
4.6 Ricerca avanzata 62
4.7 Esecuzione dello script bitcointalk.py 65
4.8 La pagina che visualizza tutti i task in esecuzione e gli ultimi terminati 67
4.9 Notifica che indica l'esecuzione di due task 67
4.10 Schema del database relazionale 69
4.11 Form di ricerca nel database 70

Introduzione

L'utilizzo di monete elettroniche basate sulla crittografia asimmetrica ha avuto negli ultimi anni un aumento considerevole. La più famosa, il bitcoin, utilizza la crittografia sia per generare nuova moneta, sia per attribuirla all'utente, rendendola quindi una criptovaluta decentralizzata, ossia senza la presenza di un ente di controllo come una banca. Il sistema di attribuzione della moneta, essendo basato sulla crittografia, non consente di attribuire un'identità fisica a nessun utilizzatore del sistema. Per questo motivo lo scambio di denaro in bitcoin, pur venendo pubblicamente registrato, non può portare all'identificazione delle persone coinvolte. L'idea di avere un metodo di pagamento che non preveda commissioni per l'invio di denaro ovunque nel mondo e che non sia sotto il controllo di un governo ha fatto gola a molti venditori. Il concetto che questo metodo nasconda addirittura l'identità delle persone coinvolte nella transazione, rendendola quasi anonima, ha spinto inevitabilmente molti venditori ad offrire prodotti o servizi non ovunque legali. Armi, documenti falsificati e droga sono solo alcuni degli articoli che è possibile acquistare con questa criptovaluta. Il senso di sicurezza fornito dall'uso del bitcoin porta molti utilizzatori a non prestare le dovute precauzioni: se lo stesso indirizzo bitcoin viene usato per più acquisti o se l'utente lo pubblica per ricevere pagamenti all'interno di forum, ad esempio, è possibile profilare

la figura che si nasconde dietro l'indirizzo stesso analizzando le transazioni o i messaggi stessi del forum.

Lo scopo del progetto presentato è questo: cercare di recuperare tutte le informazioni disponibili online, su forum che trattano di bitcoin, sull'elenco pubblico delle transazioni, su siti di particolare interesse, al fine di tracciare e identificare persone o gruppi di persone che fanno un uso illecito di questo metodo di pagamento. Il framework sviluppato non svolge solo una ricerca nel web che noi tutti conosciamo: esiste un'insieme di servizi online ma nascosti, che fanno parte del famigerato *Dark web*. Questi servizi sono raggiungibili solo connettendosi a specifiche reti protette, tramite configurazioni particolari, orientate all'anonimato: è proprio qui che molti venditori effettuano i loro scambi illeciti.

La componente descritta in questo lavoro di tesi fornisce uno strumento per il recupero di indirizzi bitcoin e di informazioni legate agli stessi, come il nome utente di un forum o tag identificativi, presenti all'interno di siti web ottenuti tramite una ricerca di parole chiavi o in siti web che l'utilizzatore del sistema decide di analizzare. Il recupero di informazioni nel dark web non è trattato in questo testo siccome verrà introdotto nel framework successivamente.

Un esempio pratico dell'utilizzo della componente è il seguente: Un tizio, che chiameremo Aldo, mette a vendere della droga in un mercatino online per 0.5 BTC, pubblicando l'indirizzo bitcoin per il pagamento. Un utente del mercatino, Bob, procede all'acquisto utilizzando il suo indirizzo , sentendosi protetto, siccome in origine non ha ottenuto quell'indirizzo fornendo dati personali o un documento, ma soltanto generandolo dal computer. Questo

crea un falso senso di sicurezza in Bob che non penserà mai di essere scoperto. Qualche giorno dopo, durante un'indagine della Polizia, il mercatino viene controllato e l'indirizzo bitcoin del venditore di droga, Aldo, analizzato dalle Forze dell'Ordine. Utilizzando il framework sviluppato per effettuare una ricerca sulle transazioni dell'indirizzo di Aldo, la Polizia scopre che è stato recentemente ricevuto un pagamento di 0.5 BTC da un indirizzo apparentemente sconosciuto. L'indagine non si ferma qui: il modulo sviluppato viene utilizzato per identificare l'utente che ha acquistato droga: Una volta fornito l'indirizzo di Bob, il sistema trova nel database che lo stesso indirizzo è presente su un famoso forum, riguardante il mondo bitcoin, che era stato analizzato da qualche mese. L'indirizzo è collegato ad un profilo utente di un certo xBob_Smith10 che vive in provincia di Mantova ed ha 43 anni. Analizzando i messaggi nel forum, si scopre che l'utente ha recentemente divorziato. La Polizia ha ora molte informazioni su cui lavorare e ha ristretto di molto il cerchio del possibile acquirente. La ricerca di un "Bob Smith" 43enne, residente nella provincia di Mantova e con un divorzio recente, non dovrebbe lasciare alla Polizia molta scelta su chi sia realmente il compratore di droga.

I successivi capitoli sono organizzati per fornire al lettore un background minimo dei concetti affrontati durante lo sviluppo del software: il capitolo 1 tratterà del sistema bitcoin e delle sue componenti, mentre il capitolo 2 tratta dei metodi per automatizzare il salvataggio di informazioni provenienti dai siti internet. Nel capitolo 3 verrà mostrato il progetto attualmente in sviluppo con tutte le sue componenti, con un confronto ad un sistema simile. Il capitolo 4 contiene invece una descrizione del modulo sviluppato dal lau-

reando, con dettagli sull'implementazione. Infine, nel capitolo 5, si trattano i risultati del progetto di tesi con delle considerazioni sulla sua affidabilità e operatività.

Capitolo 1

Bitcoin: protocollo e criptovaluta

Con la parola bitcoin ci si riferisce generalmente ad una criptovaluta inventata nel 2009 da uno sviluppatore, o da un gruppo di sviluppatori, sotto il nome di Satoshi Nakamoto[1]. Il termine Bitcoin, scritto con la lettera B maiuscola, viene utilizzato per riferirsi al protocollo di funzionamento utilizzato dal bitcoin e quindi alla rete Bitcoin stessa. Una criptovaluta è una valuta digitale la cui implementazione, per il controllo degli scambi e del processo di creazione della moneta stessa, si basa sulla crittografia.

1.1 Introduzione al mondo Bitcoin

Bitcoin è stata la prima criptovaluta decentralizzata realizzata nel 2009, l'unità di conto di questa valuta è il *"bitcoin"*, scritto solitamente con la prima lettera minuscola o identificato dalla sigla **BTC** o **XBT** oppure rappresentato da una B sbarrata due volte come nel logo ufficiale in figura 1.1. Ha un proprio tasso di cambio che varia continuamente, come mostrato in figura 1.2.

Figura 1.1: Il famoso logo di Bitcoin.[1]

La moneta ha diverse sub-unità, la più importante è il *"satoshi"* che corrisponde a 0.00000001 BTC ma esiste anche il *"microbit"* (uBTC) che vale 0.000001 BTC o il *"Bitcent"*: 0.01 BTC. Dal 2009 ad oggi sono state create diverse criptovalute ma il bitcoin resta quella più utilizzata e con più valore. La particolarità di queste valute è che sono decentralizzate, ossia non sono regolate da nessuna banca. Per questo motivo l'intero sistema è peer-to-peer e tutte le transazioni avvengono direttamente tra i partecipanti, senza il coinvolgimento di intermediari. Tutte le transazioni sono verificate dai nodi della rete e registrate in un database distribuito pubblico chiamato *blockchain*.

Non essendoci una banca centrale che decide quando emettere nuova moneta, i bitcoin vengono generati casualmente dalla rete e distribuiti tra gli utenti che hanno attivamente fornito potenza computazionale, tramite le loro macchine, per risolvere i complessi calcoli che stanno dietro alla verifica e all'aggiunta di nuove transazioni. La probabilità di ricevere bitcoin è più alta quanto più alta è la potenza computazionale fornita dall'utente stesso. Questo procedimento è chiamato *mining*, da gold mining, ed è l'unico metodo per generare bitcoin. Gli utenti che partecipano al mining possono inoltre ricevere dei bitcoin grazie alle commissioni sulle transazioni: le commissioni sono

[1]Fonte: https://bitcoin.org/

impostate liberamente da chi effettua la transazione anche se da maggio 2013 commissioni sotto alla soglia di 0.0001 BTC non vengono ritenute standard e le transazioni potrebbero non essere approvate. Gli utenti che svolgono il mining possono scegliere quali transazioni gestire per prime e questo favorisce la validazione e l'inserimento di transazioni con un'alta commissione, rispetto ad altre con una bassa commissione. Aumentando quindi l'importo della commissione, si aumenta la probabilità di verificare rapidamente una transazione. Le transazioni sono aggiunte alla blockchain in blocchi all'incirca ogni 10 minuti e quando un nuovo blocco viene aggiunto nuovi bitcoin vengono generati e distribuiti tra gli utenti che hanno fornito più potenza.

Il numero di bitcoin generati per ogni blocco aggiunto non è casuale: inizialmente venivano generati 50 BTC ogni nuovo blocco inserito ma tale cifra viene dimezzata ogni 4 anni circa. Questo dimezzamento prevede quindi che vengano generati 21 milioni di bitcoin in un arco di 130 anni circa, con l'80% di essi generati nei primi 10 anni. Il sistema di mining porterà gli utenti ad avere un guadagno esclusivamente dalle commissioni delle transazioni, non venendo più generati nuovi bitcoin. Dal 28 novembre 2012 la ricompensa è stata di 25 BTC per blocco, nel luglio 2016 si è passati a 12.5 BTC. Oltre al mining, un utente può ricevere bitcoin se è il destinatario di una transazione, oppure può affidarsi a dei servizi che permettono di scambiare una valuta più comune, come Dollaro o Euro, con il corrispettivo in bitcoin.

Figura 1.2: Grafico che mostra il cambio di valore di 1 BTC in Euro da agosto a dicembre 2016.[3]

1.2 Struttura della rete bitcoin

La rete bitcoin, che implementa l'omonimo protocollo, è una rete peer-to-peer formata da vari nodi che formano il database distribuito della blockchain e che quindi tiene memoria di tutte le transazioni verificate. Quando si connettono, i nodi della rete scaricano i nuovi blocchi da altri nodi. Non sempre un nodo contiene una copia di tutta la blockchain, spesso i nodi scaricano solo qualche blocco. Un nodo che contiene una copia di tutta la blockchain è chiamato *full-node*. I nodi controllano che i blocchi e le transazioni siano validi per le regole del protocollo Bitcoin, in particolare controllano che:

- I blocchi abbiano generato soltanto il numero consentito di bitcoin (Attualmente 12.5 BTC)
- Le transazioni e i blocchi siano in un formato corretto.

[3]Fonte: https://bitcoincharts.com/

- Le transazioni siano state approvate (controllando che siano state correttamente firmate).
- I bitcoin non vengano spesi due volte.

Le componenti principali della rete bitcoin sono quindi i nodi che gestiscono la blockchain e le transazioni, i miners che contribuiscono a verificare le transazioni e, indirettamente, gli utenti che inviano e ricevono le transazioni (figura 1.3).

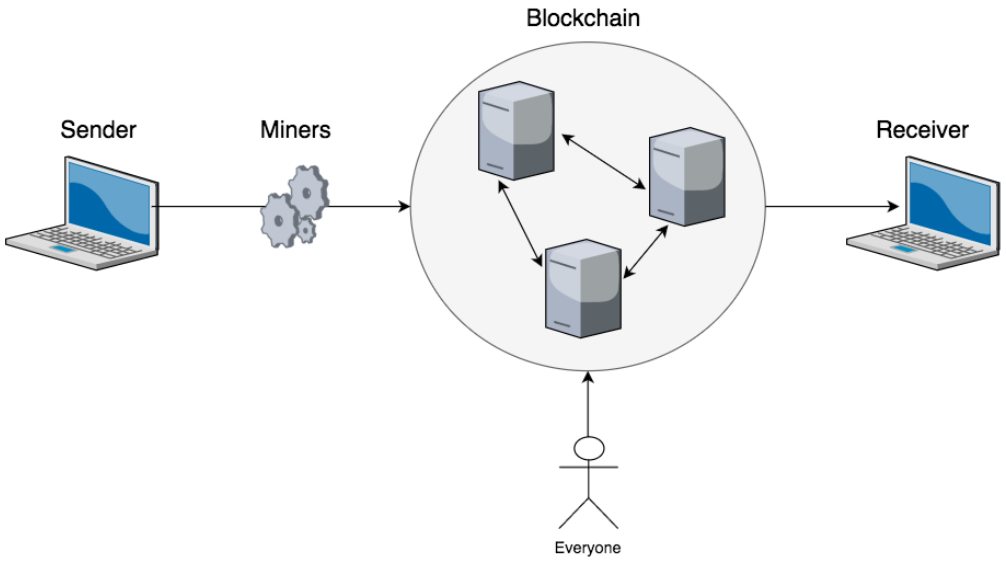

Figura 1.3: Schema della rete bitcoin.

1.3 Blockchain

La blockchain (letteralmente catena di blocchi) è un database distribuito tra tutti i nodi che implementano il protocollo bitcoin e che mantiene la lista di tutte le transazioni eseguite dagli utenti, come mostrato in figura 1.4. Una blockchain è composta da records chiamati blocchi (blocks) che contengono,

oltre ad un timestamp, anche l'hash di un collegamento al blocco precedente, consentendo difatti la creazione di una catena di blocchi dal primo blocco inserito fino all'ultimo. Il primo blocco della catena prende il nome di *Genesis* block ed è l'unico che non ha un collegamento al precedente. Questo sistema a blocchi collegati fornisce un ordine cronologico ai record inseriti e previene la modifica di un blocco già presente nella blockchain in quanto si dovrebbero modificare anche tutti i blocchi successivi. Una blockchain è quindi valida se e solo se tutti i blocchi che la compongono sono validi e se la catena inizia con il blocco Genesis. È interessante notare come per ogni blocco della catena, ci sia un solo percorso che porta al blocco Genesis ma non è valido l'opposto infatti, scendendo dal blocco Genesis, si possono trovare biforcazioni. Questo avviene nel caso in cui due blocchi vengano aggiunti con uno scarto di pochi secondi l'uno dall'altro. Quando questo si verifica, il successivo blocco generato si collega al primo dei due blocchi che riceve e questa nuova catena diventa la principale, essendo più lunga dell'altra. I blocchi nella catena più corta, chiamata catena invalida, non vengono usati e le transazioni contenute in questi blocchi vengono rimesse nella coda delle transazioni in attesa e verranno verificate e incluse in un successivo blocco. Per questo motivo è buona pratica considerare valida una transazione solo dopo che sono stati aggiunti altri blocchi alla catena, per evitare che sia annullata e rimessa in coda. Questo sistema rende molto difficile spendere gli stessi bitcoin due volte infatti, essendo tutti i movimenti presenti nella blockchain, una transazione viene confermata solo se quei bitcoin non sono già stati spesi.

Figura 1.4: Schema semplificato della blockchain.[4]

1.4 Indirizzi Bitcoin

Le transazioni registrate nella blockchain sono operazioni di scambio di bitcoin effettuate tra indirizzi bitcoin: identificatori alfanumerici univoci (case sensitive) che possono essere lunghi dai 26 ai 35 caratteri e che rappresentano ognuno una chiave crittografica pubblica diversa concatenata ai bit di controllo (*checksum*). Il proprietario di un indirizzo bitcoin possiede la chiave privata dello stesso necessaria a firmare le transazioni in uscita dall'indirizzo, questo è sufficiente per dimostrarne l'appartenenza. Gli indirizzi vengono utilizzati per rappresentare la destinazione, o l'origine, di un pagamento in bitcoin. La creazione di un nuovo indirizzo bitcoin non comporta nessuna spesa all'utente e può essere svolta offline; è quindi possibile generare un indirizzo bitcoin diverso per ogni singolo pagamento. Questa tecnica è strettamente consigliata vista la natura stessa delle transazioni e difatti molti software bitcoin o siti web, creano un nuovo indirizzo ogni qual volta l'utente apre una richiesta di pagamento. Nel caso di siti web che accettano pagamenti in bitcoin, è possibile usare appositi software che, tramite un token, sono in grado di generare un numero infinito di indirizzi bitcoin con i quali

[4]Fonte: https://blockchainfrance.net/

ricevere pagamenti da ogni cliente. Come già detto, essendo gli indirizzi bitcoin stringhe alfanumeriche case sensitive, la probabilità di commettere errori di trascrizione è alta ed è per questo consigliato utilizzare la clipboard del dispositivo in uso per copiarli e incollarli, piuttosto che trascriverli a mano. Tuttavia, grazie al codice di controllo presente, la probabilità che un indirizzo bitcoin trascritto male sia accettato come valido è 1 su 2^{32} ossia circa 1 su 4.29 miliardi, in caso di trascrizione errata l'indirizzo verrà molto probabilmente rifiutato e si dovrà provvedere a inserirlo nuovamente. Il termine *portafoglio* bitcoin viene utilizzato per definire l'insieme di chiavi private possedute da un utente, e quindi dei suoi indirizzi bitcoin. Il portafoglio può essere salvato all'interno di un file presente nel proprio computer ma deve essere opportunamente protetto in quanto l'accesso alle chiavi private fornisce automaticamente l'accesso ai bitcoin. La perdita del portafoglio, e quindi delle chiavi private, comporta la perdita dei bitcoin e per questo si consiglia di avere sempre una copia di backup aggiornata con gli ultimi indirizzi e salvata localmente. Il problema di mantenere sempre un backup aggiornato, ogni qual volta si generi un nuovo indirizzo bitcoin nel proprio portafoglio, può essere risolto grazie all'utilizzo dei più moderni e evoluti *portafogli deterministici*[2]. Questi portafogli utilizzano una sola chiave privata come token iniziale, chiamata comunemente *seed*, da cui possono generare una lista sequenziale di chiavi private e quindi di indirizzi bitcoin. Con questo sistema, ci si deve preoccupare soltanto di salvare il master seed del portafoglio una sola volta, in quanto in caso di perdita del portafoglio, basterà questo seed per rigenerare le stesse chiavi private senza nessuna perdita di bitcoin. Gli indirizzi più comuni iniziano con i numeri 1 o 3 che derivano dai due formati

utilizzati, *P2PKH (Pay To PubKey Hash)* per gli indirizzi con prefisso 1, per esempio `1BvBMSEYstWetqTFn5Au4m4GFg7xJaNVN2`, e *P2SH (Pay to script hash)* per quelli con prefisso 3, per esempio `3J98t1WpEZ73CNmQviecrnyiWrnqRhWNLy`. Da tutti i caratteri alfanumerici sono esclusi i seguenti caratteri per prevenire ambiguità visiva: 0, O, I, l ossia: zero, O maiuscola, I maiuscola e L minuscola.

1.4.1 Evitare il riutilizzo

Siccome il contenuto della blockchain è pubblicamente disponibile, tutti i movimenti legati ad un indirizzo bitcoin sono visionabili da chiunque. È fortemente sconsigliato riutilizzare lo stesso indirizzo per più transazioni in quanto non è solo una pratica scorretta di utilizzo degli indirizzi ma è anche un problema alla sicurezza e alla privacy degli utenti inclusi nella transazione, così come a futuri proprietari della moneta: nel caso in cui un utente utilizzi sempre lo stesso indirizzo, un altro utente che effettui un acquisto o una vendita dal primo potrebbe analizzare tutti i suoi movimenti e identificare la persona, esistono infatti database pubblicamente conosciuti che hanno già analizzato milioni di indirizzi bitcoin per collegare le informazioni sulle transazioni a quelle presenti pubblicamente nei siti web, come ad esempio messaggi su forum, e quindi per tracciare l'utente proprietario dei vari indirizzi. Molti dei rischi non sono collegati esclusivamente all'utilizzatore dello stesso indirizzo, ma anche alle persone coinvolte dalle sue transazioni: basti pensare ad un utente non curante dei rischi del riutilizzo che potrebbe essere identificato e attirare l'attenzione della legge se i suoi soldi sono stati spediti ad un indirizzo che li ha utilizzati direttamente per acquistare droga o armi

non registrate. Inoltre, utilizzando lo stesso indirizzo per tutte le operazioni, sono noti a tutti il saldo e la lista dei movimenti dell'utente: questo potrebbe portare ad azioni più o meno lecite rivolte all'appropriazione del denaro, come rapimenti o estorsioni, ma non solo: un datore di lavoro potrebbe prendere in antipatia o avere un comportamento differente con il dipendente se nota che lo stesso effettua donazioni in bitcoin verso terzi che non sono simpatici al datore (motivazioni politiche/religiose, as esempio).

1.4.2 Creazione di un indirizzo bitcoin

Si può creare un indirizzo bitcoin a partire da una coppia di chiavi pubblica/privata *ECDSA*. La chiave privata, disponibile solo al proprietario, viene usata per firmare le transazioni dell'indirizzi bitcoin mentre la chiave pubblica serve per verificare l'autenticità della firma. La chiave pubblica viene inoltre usata per generare un indirizzo bitcoin che contiene dei bit di controllo e che sia visivamente non ambiguo.

ECDSA (Elliptic Curve Digital Signature Algorithm) è un algoritmo crittografico derivato dal *DSA* (Digital Signature Algorithm) che utilizza la crittografia ellittica, utilizzato da Bitcoin per assicurare che i fondi siano spesi soltanto dal legittimo proprietario. La chiave privata è un intero senza segno formato da 256 bit (32 bytes). La chiave publica può essere compressa o non: le chiavi pubbliche compresse sono lunghe 33 bytes e hanno un prefisso che può essere o 0x02 o 0x03, seguito da un intero x di 256 bit (32 bytes), quelle non compresse sono lunghe 65 bytes e sono formate da un prefisso costante 0x04 e da due interi di 256 bit ciascuno chiamati x e y (2 * 32 bytes). Il prefisso delle chiavi compresse consente all'intero y di essere derivato da x.

Partendo da una coppia di chiavi pubblica/privata ECDSA possiamo creare un indirizzo bitcoin con il seguente metodo (mostrato anche dallo schema in figura 1.5): Chiave privata:

```
18E14A7B6A307F426A94F8114701E7C8E774E7F9A47E2C2035DB29A206
321725
```

Chiave pubblica:

```
0450863AD64A87AE8A2FE83C1AF1A8403CB53F53E486D8511DAD8A0488
7E5B23522CD470243453A299FA9E77237716103ABC11A1DF38855ED6F2EE1
87E9C582BA6
```

1. La chiave pubblica viene analizzata, il primo byte è 0x04 quindi non è compressa, i successivi 32 bytes sono la coordinata x e gli ultimi 32 bytes sono la coordinata y
2. Eseguire l'algoritmo di hashing *SHA-256* sulla chiave pubblica

```
600FFE422B4E00731A59557A5CCA46CC183944191006324A447BDB2D
98D4B408
```

3. Eseguire l'algoritmo di hashing *RIPEMD-160* sul risultato precedente

```
010966776006953D5567439E5E39F86A0D273BEE
```

4. Aggiungere il byte di versione all'inizio della stringa ottenuta (0x00 per versione *P2PKH*, 0x05 per versione *P2SH*)

```
00010966776006953D5567439E5E39F86A0D273BEE
```

5. Eseguire nuovamente l'algoritmo di hashing *SHA-256* sull'hash *RIPEMD-160* con byte di versione

```
445C7A8007A93D8733188288BB320A8FE2DEBD2AE1B47F0F50BC10BA
E845C094
```

6. Eseguire per la terza volta l'algoritmo di hashing *SHA-256* sul precedente hash *SHA-256*

```
D61967F63C7DD183914A4AE452C9F6AD5D462CE3D277798075B10761
5C1A8A30
```

7. Prendere i primi 4 bytes dell'hash, questo sarà il *checksum* dell'indirizzo

```
D61967F6
```

8. Aggiungere il *checksum* all'hash *RIPEMD-160* esteso del passo 4 (con byte di versione). Questo sarà l'indirizzo bitcoin binario lungo 25 bytes

```
00010966776006953D5567439E5E39F86A0D273BEED61967F6
```

9. Convertire la stringa binaria in una stringa *base58check*[5] (Eliminando quindi i caratteri visivamente ambigui) per ottenere la forma più comune di indirizzo bitcoin.

[5]https://en.bitcoin.it/wiki/Base58Check_encoding

```
16UwLL9Risc3QfPqBUvKofHmBQ7wMtjvM
```

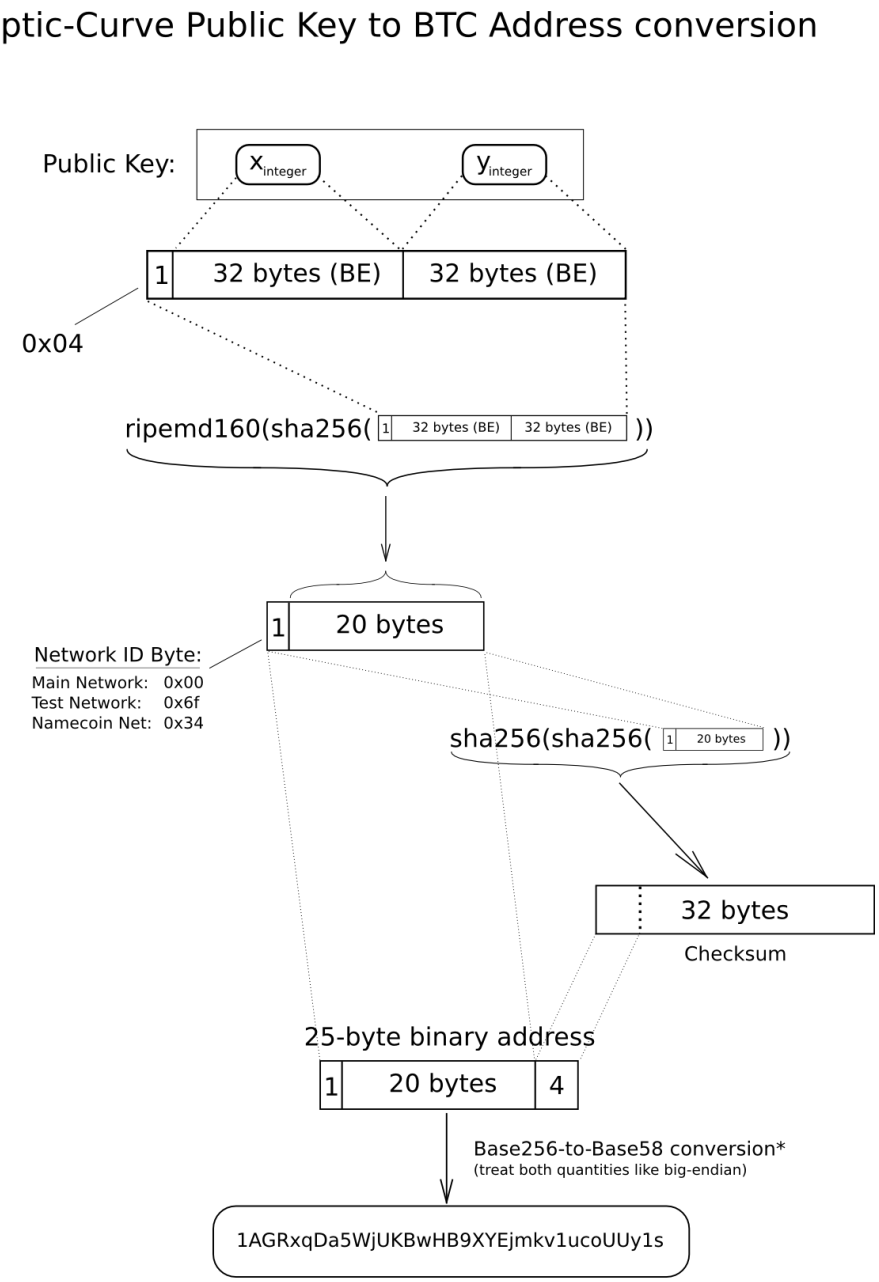

Figura 1.5: Uno schema che illustra la creazione di un indirizzo bitcoin a partire dalle chiavi ECDSA.[7]

1.4.3 Validare un indirizzo bitcoin

Riconoscere un indirizzo bitcoin o un presunto tale non è un operazione difficile, basta trovare una stringa lunga dai 26 ai 35 caratteri, che inizi per 1 o per 3 e che non contenga i caratteri ambigui già citati (0 O I l, zero, O maiuscola, I maiuscola e L minuscola). Per validarlo si deve invece verificare che i bit di controllo siano validi. Un algoritmo che valida un indirizzo bitcoin lo decodifica dalla base58 (eseguendo un operazione opposta alla *base58check*). La decodifica controlla sia la lunghezza dell'indirizzo che la presenza di caratteri non ammessi. Una volta decodificato, viene eseguito il doppio algoritmo di hashing *SHA-256* sui primi 21 bytes (1 byte per la versione e 20 per la chiave pubblica) e confrontato con gli ultimi 4 (il *checksum*). se le due stringhe sono identiche, allora il checksum è valido e lo è anche l'indirizzo. Questo sistema è sufficiente a dimostrare che un indirizzo sia valido, senza dover analizzare ulteriormente la chiave pubblica.

1.5 Transazioni

Una transazione è un trasferimento di denaro notificato a tutta la rete Bitcoin e salvato in un blocco della blockchain. Una transazione può essere formata da più input e più output. Gli input sono riferimenti a output di una transazione precedente e gli output sono indirizzi bitcoin che ricevono il denaro. Le transazioni non sono criptate e chiunque può visionarle all'interno di un blocco utilizzando un servizio apposito che le mostra in un formato human-readable. Le transazioni vengono spesso analizzare per verificare i pa-

[7]Fonte: https://en.bitcoin.it/wiki (etotheipi@gmail.com)

gamenti effettuati: solitamente una transazione di un blocco appena inserito nella blockchain viene considerata verificata, anche se per grossi importi si consiglia di aspettare che vengano aggiunti altri blocchi prima di considerare al 100% verificata una transazione. Quest'attesa è necessaria siccome un blocco da poco aggiunto potrebbe essere respinto o considerato non valido e quindi tutte le transazioni del blocco potrebbero tornate nella lista delle transazioni da approvare.

1.5.1 Sistema dei resti e delle commissioni

Si può considerare il sistema di input/output come dei blocchi di bitcoin separabili solo tramite le transazioni: Un indirizzo bitcoin può avere ricevuto N pagamenti e quindi avere N "blocchi" di output da usare come input per una transazione. Il caso base prevede che un utente abbia precedentemente ricevuto un pagamento di 10 BTC, ad esempio, e debba pagare esattamente 10 BTC (figura 1.6). La transazione conterrà come input l'id di un output precedente (di 10 BTC) e l'indirizzo del destinatario del nuovo pagamento con il numero di bitcoin da inviare (sempre 10). Tutti i bitcoin del blocco saranno trasferiti al destinatario (figura 1.7).

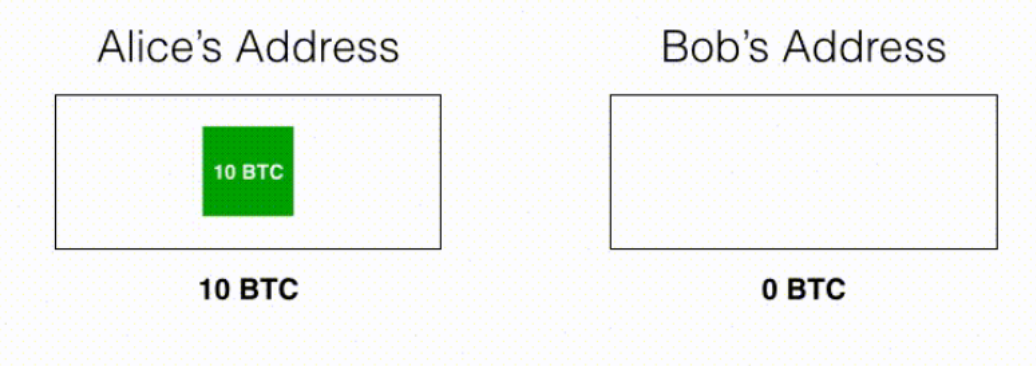

Figura 1.6: Alice possiede 10 BTC in un blocco (ottenuti da un singolo output).[9]

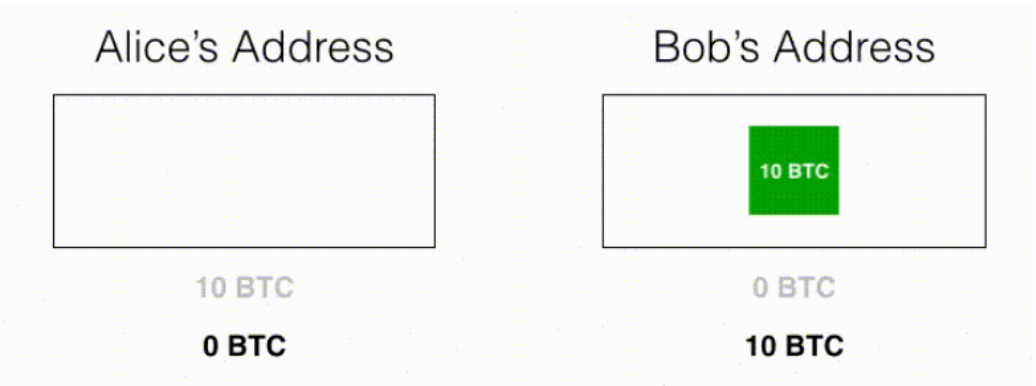

Figura 1.7: Alice trasferisce 10 BTC a Bob.[9]

La situazione è ben diversa nel caso in cui il destinatario deve ricevere solo una parte dei bitcoin di un blocco, ad esempio 5 BTC, infatti i blocchi devono essere inseriti **interi** nella transazione, non si può quindi inviare solo 5 BTC da un blocco di 10 BTC, ma vanno trasferiti i 10 BTC del blocco (figura 1.8). Per ovviare a questo problema si utilizza un indirizzo dei resti,

[9]Fonte: http://bitzuma.com/

ossia si aggiunge all'output della transazione un altro indirizzo, appartenente al proprietario dell'input, al quale vengono consegnati i bitcoin di "resto".

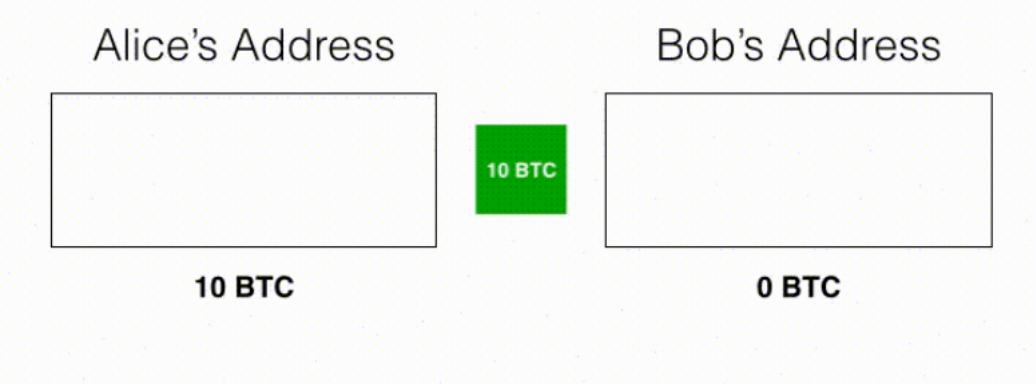

Figura 1.8: Alice deve trasferire solo 5 BTC a Bob, ma il blocco contiene 10 BTC.[9]

L'output comprende 2 indirizzi (uno del destinatario e uno di resto) al quale inviare 5 BTC ciascuno (figura 1.9).

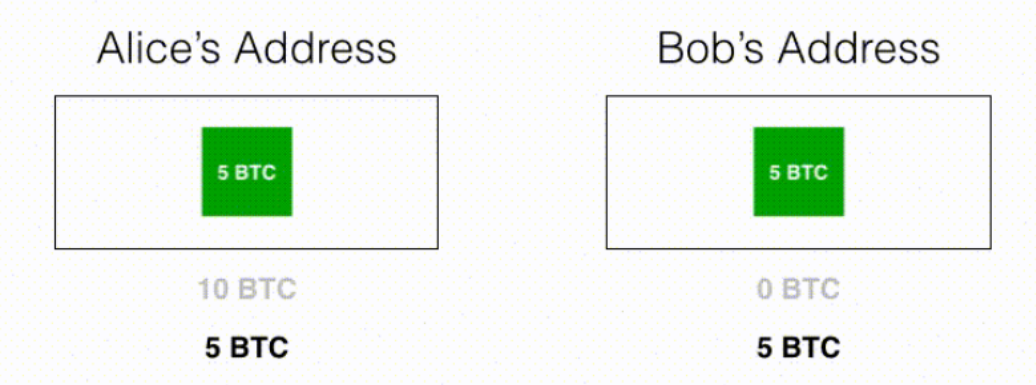

Figura 1.9: Alice inserisce come output il suo stesso indirizzo, ricevendo 5 BTC di resto.[9]

Si consiglia sempre di generare un nuovo indirizzo dei resti ad ogni pagamento, piuttosto che usare un altro indirizzo bitcoin esistente o addirittura

lo stesso da cui proviene il pagamento (come spiegato nel paragrafo 1.4.1). Questo sistema dei resti può causare perdite di denaro da parte di un utente inesperto[3], infatti l'uso comune della moneta online prevede che si paghi soltanto la cifra richiesta e che non esista un sistema di resto. Si penserebbe quindi che se si hanno 5 BTC in un indirizzo bitcoin e se ne spendono 2, allora ne restano 3 ma, come già descritto, tutti e 5 i BTC devono essere trasferiti. Il trucco è quindi immaginare ogni blocco di output, presente su un indirizzo e usato come input, come ad una banconota fisica: quando si effettua il pagamento, si fornisce una banconota che copra l'importo e si riceve un resto.

Un problema non da sottovalutare si ha nel caso in cui l'utente non fornisca un indirizzo di resto per ricevere i bitcoin non trasferiti su altri indirizzi, in questo caso i bitcoin non vengono persi totalmente ma considerati come "costo della transazione" e quindi recuperati dai *miners* che hanno contribuito a verificare la transazione. Questo procedimento che può essere complesso per utenti non esperti, viene mascherato da molti software che gestiscono i portafogli bitcoin dell'utente, rendendo di fatto un pagamento in bitcoin molto più semplice, creando un nuovo indirizzo dei resti per ogni pagamento e indirizzando i bitcoin in più in quel nuovo indirizzo.

1.6 Controllo e gestione del protocollo Bitcoin

Il sistema che gestisce l'uso dei bitcoin non appartiene a nessuno: è stato realizzato da un gruppo di sviluppatori, che ancora oggi si occupa della gestione del sito web[10], che ha reso il codice aperto e disponibile online. Chiunque

[10]http://bitcoin.org

può contribuire ma, grazie alla natura condivisa del sistema, il controllo dello stesso appartiene a tutti gli utenti: ogni "nodo" della rete decide quale software utilizzare e per rimanere compatibile con gli altri nodi, ogni utente deve usare un software conforme a determinate regole. Per questo motivo una modifica al protocollo Bitcoin viene applicata soltanto se c'è il consenso degli utenti. Gli sviluppatori hanno quindi interesse a implementare funzionalità che possano mantenere questo consenso.

La rete Bitcoin ha inoltre una comunità molto attiva sui social. Su reddit è presente un sottogruppo, */r/Bitcoin*[11], dedicato a chiunque abbia domande o considerazioni da porre. È molto attiva anche la chat IRC #bitcoin (sul server freenode)[12] ed esistono inoltre forum, il più importante è BitcoinTalk che conta quasi 1 milione di utenti. I principali argomenti di discussione nel forum BitcoinTalk trattano lo sviluppo dei software Bitcoin e dei dettagli tecnici. Un altro canale molto famoso è #bitcoin-otc disponibile sempre all'interno della chat IRC (freenode)[13]. Bitcoin-OTC è un mercato *over-the-counter*, ossia fuori dai circuiti borsistici ufficiali, che si occupa, tra le altre cose, dello scambio di bitcoin. La sicurezza degli scambi è in parte garantita da un sistema che cerca di mitigare il rischio di subire una truffa grazie ad un servizio di *web of trust* fornito dalla community di bitcoin-otc. Questo servizio fornisce un sistema che permette agli utenti interessati da uno scambio di lasciare un feedback pubblico alla controparte (da -10 a +10). Il database è pubblicamente consultabile e permette a chiunque di controllare i feedback ricevuti da un utente.

[11]Subreddit: `https://www.reddit.com/r/Bitcoin/`

[12]Webchat: `http://webchat.freenode.net/?channels=#bitcoin`

[13]Webchat: `http://webchat.freenode.net/?channels=#bitcoin-otc`

Capitolo 2

Recupero di informazioni dal web

Internet è ad oggi la più vasta collezione di informazioni e dati esistente: il suo utilizzo ha subito negli ultimi anni una crescita esponenziale portando questa tecnologia, e l'immensa mole di dati che essa offre, alla portata di tutte le famiglie. Se nel 1984 la percentuale di famiglie statunitensi che possedeva un computer era del 8.2%, nel 2011 si è arrivati al 75.6% della popolazione[4]. Tutti questi dati, resi disponibili dai numerosissimi siti web creati negli ultimi anni, hanno portato alla nascita di sistemi di raccolta, sia per fini di indicizzazione, nel caso dei motori di ricerca, sia per fini statistici, sia per questioni di marketing.

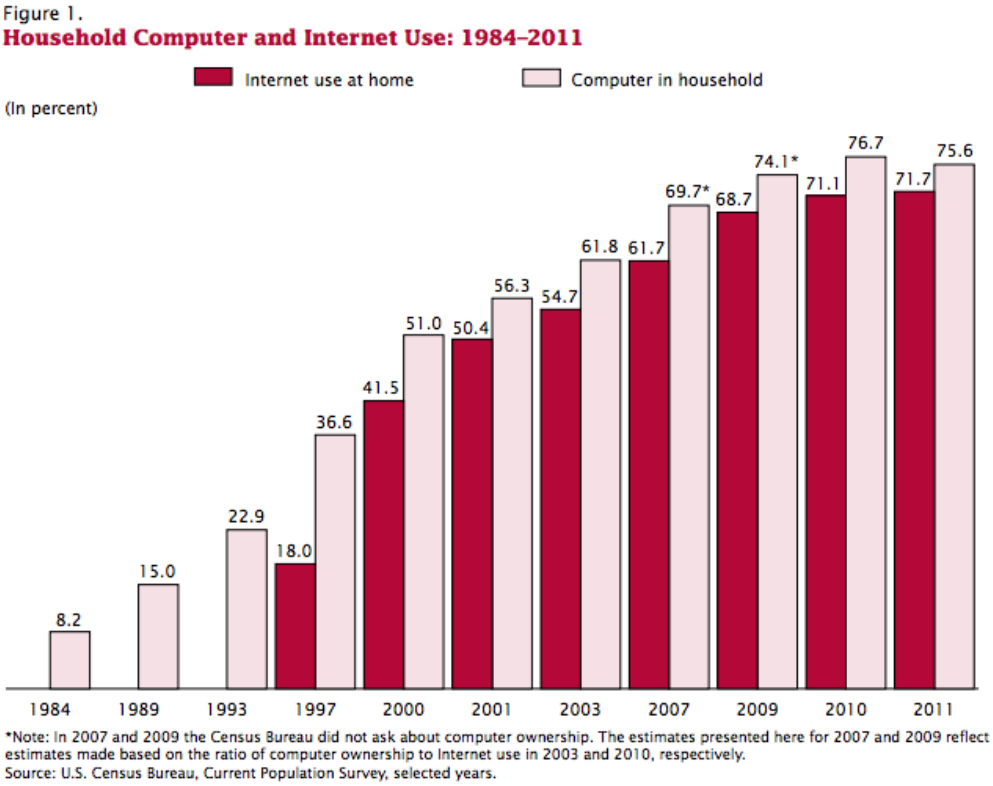

Figura 2.1: Famiglie americane e utilizzo di internet.[4]

2.1 Stato delle informazioni

Le informazioni disponibili nel web sono eterogenee: ogni sviluppatore decide come rappresentarle nel proprio sito e non viene utilizzato nessuno standard di rappresentazione dei dati. Per questo, la collezione di dati in grosse quantità è complicata da automatizzare. Il pratico copia-incolla non è un opzione quando i dati da prelevare sono dell'ordine delle centinaia, se non migliaia o addirittura milioni: se ad esempio si dovessero salvare manualmente su un foglio di calcolo delle informazioni, il processo consisterebbe nel copiare e incollare ogni singolo dato dalla sua pagina web al documento da creare, ri-

chiedendo tempo che sarebbe preferibile risparmiare utilizzando un programma che svolga tutte le operazioni. Fortunatamente, sebbene i dati inseriti nelle pagine web non siano strutturati perfettamente, è possibile utilizzare dei programmi, chiamati *scraper*, che in base al formato e alla struttura della pagina possono adattarsi ed effettuare lo scraping (letteralmente raschiatura o scrostatura) per ottenere i dati. Gli scraper sono spesso utilizzati insieme ai *crawler*: programmi che visitano tutte le pagine raggiungibili a partire da un indirizzo fornito come input, seguendone i collegamenti.

2.2 Web Crawling

Il web crawling è il processo di analisi di siti web utilizzato per indicizzarne tutti i contenuti. Un crawler, anche conosciuto come spider, è un software specializzato per prelevare tutto il contenuto di una pagina web e seguirne i vari link per analizzare siti web collegati o pagine secondarie. Molti crawler forniscono impostazioni per controllare e limitare l'esecuzione, le più importanti sono:

- Controllo della profondità: Si può impostare un parametro per comunicare al crawler quanti collegamenti seguire dalla pagina di partenza, generalmente seguire 5 link di profondità è più che sufficiente per analizzare tutto un sito web.

- Pagine da seguire simultaneamente: Qualche web crawler può eseguire richieste in parallelo a pagine web e permette di impostare un limite al numero massimo.

- Pausa tra le richieste: Per non intasare il sito web o rischiare di essere bloccati, è possibile impostare una pausa da rispettare prima di seguire un nuovo collegamento.

- Salvataggio dei log: Siccome l'esecuzione di un crawler richiede del tempo e può interrompersi, è consigliato impostare un file che tiene traccia di tutte le operazioni effettuate e degli errori riscontrati, oltre all'elenco di pagine già ottenute prima dell'interruzione.

Il caso d'uso più comune dei crawler è l'utilizzo da parte dei motori di ricerca: le pagine trovate dal crawler vengono indicizzate per fornire risultati attinenti alla query di ricerca dell'utente che utilizza il servizio.

Un programma che effettua crawling di un sito web generalmente parte da una lista di URL ed esegue una richiesta http/https per ogni URL ottenendo come risposta il contenuto della pagina dalla quale andrà a ricavare e ad aggiungere alla lista tutti i nuovi collegamenti trovati (figura 2.2).

I crawler generalmente vengono eseguiti sistematicamente sulle stesse pagine per aggiornare le informazioni e vengono avviati passando come lista il risultato del crawling precedente. Siccome i crawler seguono ogni collegamento trovato su una pagina web e possono restituire il contenuto ad uno scraper, è possibile che vengano salvate e rese disponibili dai motori di ricerca delle informazioni che dovrebbero restare private o perlomeno che non dovrebbero essere indicizzate. Per proteggersi i webmaster possono creare un file *robots.txt* nella cartella root del sito web dove specificare le regole che i crawler dovrebbero seguire, ad esempio la seguente regola

```
User-agent: *
Disallow: /admin/
```

comunica ai crawler di non andare ad analizzare la directory admin.

Oltre a delle regole di esclusione/inclusione è possibile fornire anche delle mappe del sito (*sitemap*) generalmente in formato XML che elencano gerarchicamente tutte le pagine web. Queste mappe sono ampiamente utilizzate non solo per la navigazione dell'utente ma anche per facilitare l'operazione di crawling. Il rispetto o meno delle regole contenute nel file robots.txt dipende da come è stato implementato il crawler: si potrebbe decidere di seguire le volontà del webmaster del sito e quindi ignorare certi percorsi oppure di analizzare qualsiasi collegamento trovato. Per "boicottare" crawler che ignorano il file robots.txt i webmaster possono ricorrere a delle trappole, chiamate *crawler traps*, che sono inserite nel file robots.txt come da escludere: se un crawler ignora il file robots.txt incappa nella trappola. Queste trappole possono essere volute ma anche non volute, nel caso in cui le pagine sono generate dinamicamente. Un esempio si può avere con un calendario che contiene link per il giorno successivo e precedente: il crawler troverebbe continuamente nuovi collegamenti da seguire, rimanendo difatti "intrappolato" nel calendario. Le trappole più conosciute e utilizzate sono:

- Creazione di sotto directory indefinitamente profonde con struttura simile a:

```
http://foo.com/bar/foo/bar/foo/bar/foo/bar/...
```

- Pagine dinamiche che producono un numero indefinito di collegamenti da seguire, come ad esempio i calendari.

- Documenti con un gran numero di caratteri, capaci di bloccare il processo di crawling della pagina, se il crawler non è ottimizzato.

I crawler più avanzati sono in grado di riconoscere qualcuna di queste trappole ma è difficile evitarle tutte se non viene rispettato il file robots.txt.

Uno dei più famosi web crawler è *Googlebot* che, come è facilmente intuibile dal nome, è utilizzato da Google per il suo motore di ricerca. Il crawler di Google è identificabile in quanto le richieste contengono "Googlebot" all'interno del campo user-agent e l'indirizzo host contiene "googlebot.com". Google fornisce inoltre degli strumenti ai webmaster per impostare, tra le altre cose, la frequenza di crawling e per generare e validare i file robots.txt e le sitemap.

I difetti del web crawling risiedono nel fatto che inviare richieste ad ogni pagina del sito web può saturare la banda del sito o appesantire il server stesso rallentando le normali richieste di utenti che visitano il sito.

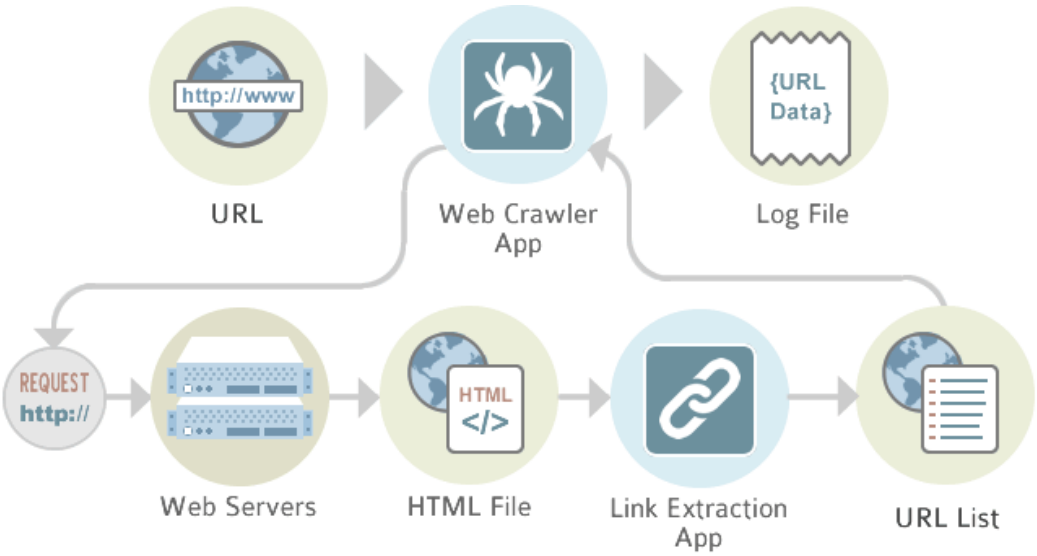

Figura 2.2: L'operazione base di un crawler è effettuare richieste HTTP per ottenere il contenuto della pagina e trovare collegamenti ad altre pagine.[2]

2.3 Web Scraping

Il web scraping, chiamato anche in diversi modi tra cui *web data extraction*, è una tecnica che consiste nel prelevare dati da siti web. Anche se l'estrazione di dati da un sito web può essere eseguita manualmente dall'utente, si parla di web scraping sopratutto quando il processo di estrazione è automatizzato grazie all'uso di un software che emula la visita di un utente ad un sito web e quando i dati ottenuti sono inseriti in un database o in dei fogli di stile, o più generalmente salvati localmente, per poi essere analizzati nel complesso. Il contenuto della pagina web viene quindi analizzato per ottenere informazioni.

[2]Fonte: https://import.io/

Esempi di utilizzo di web scraping sono il confronto dei prezzi dello stesso prodotto trovato su diversi siti web, l'analisi dei dati di diversi utenti di un sito web a fine di creare un elenco organizzato o, più generalmente, il monitoraggio di dati da siti web.

Per raccogliere grandi quantità di dati, e non solo uno specifico dato, il web scraper è spesso usato come estensione di un web crawler: i moderni web scraper possono analizzare interi siti web seguendo i collegamenti per recuperare i vari contenuti, per poi analizzarli individualmente estrapolando i dati necessari.

2.3.1 Tipologie di web scraper

Esistono diverse tecniche di scraping, in base al tipo e al formato del documento che si deve analizzare. Tra le più usate ci sono:

- Copia/Incolla Manuale: Pur essendo una tecnica banale e non troppo efficiente, il copia/incolla si rileva utile nel caso in cui debbano essere recuperati pochi dati. Sopratutto se il sito web è sviluppato in maniera tale da impedire la visualizzazione a programmi automatizzati.

- Text Pattern Matching: Un altro sistema semplice che funziona nella gran parte dei casi, consiste nell'utilizzare un'espressione regolare per il recupero dei dati da risorse non strutturate.

- Parsing di pagine HTML o XHTML: Le pagine più comuni con cui sono realizzati i siti web sono formate da un linguaggio di markup, solitamente HTML. Essendo le pagine web strutturate con i TAG HTML,

si può effettuare il parsing della pagina e prelevare il contenuto di un TAG che contiene il dato a cui si è interessati.

- Parsing di pagine JSON o XML: Per inviare dati da un'applicazione web ad un'altra, ci sono principalmente due formati di rappresentazione utilizzati: JSON e XML. Come per l'HTML, i due formati sono machine-readable ma a differenza di questo, JSON e XML contengono esclusivamente dati e per questo è molto più facile sviluppare uno scraper che li possa prelevare. Molti linguaggi di programmazione hanno un parser JSON o XML incluso, per recuperare facilmente i dati dal formato del file.

Ci sono inoltre diversi metodi per effettuare web scraping utilizzando anche un web crawler:

- Scraping di una singola pagina: Nel caso in cui i dati da estrapolare siano contenuti in una sola pagina, è superfluo effettuare il crawling del sito web in quanto basta eseguire il web scraper nella pagina bersaglio.

- Scraping di una lista divisa tra pagine correlate: Se i dati da estrarre sono presenti in più pagine identificate da un numero, come ad esempio una lista che viene divisa da dei parametri nell'url:

```
http://esempio.com/lista.php?page=1
http://esempio.com/lista.php?page=2
...
http://esempio.com/lista.php?page=N
```

allora si può creare a parte un elenco di URL che corrispondono al formato ...&page=x, dove x è un numero progressivo, e eseguire lo scraper su ogni URL.

- Scraping su pagine descrittive di un elemento: Le pagine descrittive sono pagine che non hanno parametri nell'URL da iterare ma che strutturano i dati tra le diverse pagine nello stesso modo, come ad esempio un blog dove gli articoli hanno tutti stessa struttura o come un sito di vendita online che ha una pagina strutturata per ogni prodotto, ma con URL generalmente human-readable contenente informazioni riguardanti il prodotto stesso. Siccome questa tipologia di siti web ha una pagina che contiene l'elenco di tutte le pagine strutturate, come la lista degli articoli del blog o la lista dei prodotti, si può far partire un crawler da questa pagina per ottenere la lista delle pagine strutturate, probabilmente seguendo anche la paginazione del punto precedente nel caso in cui questa lista sia divisa, per poi eseguire uno scraper sulla lista ottenuta dal crawler, ottenendo i dati.

- Scraping sull'intero sito: L'ultimo metodo consiste nell'effettuare il crawling dell'intero sito per poi effettuare scraping alla ricerca dei dati. Questo metodo richiede più tempo rispetto alle precedenti soluzioni e effettua un carico maggiore sul sito web in quanto segue ogni collegamento trovato, anche quelli che non contengono i dati cercati.

2.4 Confronto tra i più noti crawler/scraper

import.io[3] Piattaforma web-based per l'estrazione di dati da siti web, fornisce degli strumenti completamente automatizzati per effettuare scraping senza necessità di scrivere codice grazie ad un'interfaccia punta e clicca. L'utente può "istruire" la piattaforma per prelevare i dati desiderati semplicemente evidenziandoli nell'interfaccia di visualizzazione del sito. I dati sono salvati nei server dell'azienda ed è possibile esportarli in diversi formati, tra cui JSON, CSV e XLS. Si possono generare delle API per interagire con lo scraper da applicazioni di terze parti. Il piano base della piattaforma costa 249/299$ al mese per 50.000 query al mese e supporto per email. Il piano gratuito include 500 query al mese e nessun accesso alle API.

Octoparse[4] Software client-side scritto in .NET (offre un programma compilato solo per Windows) che fornisce un'interfaccia punta e clicca, non richiede di scrivere codice. Fornisce salvataggio dei dati sul computer e un servizio per connettersi in proxy per effettuare scraping anonimo. Il piano base costa 75/89$ al mese e fornisce estrazione illimitata di dati, cambio IP per evitare blocchi dai siti web, e creazione di 100 diversi task. Il piano gratuito permette lo scraping illimitato ma la creazione di soli 10 task e di 2 esecuzioni parallele del software.

Webhose.io[5] Piattaforma web-based orientata allo scraping. Il servizio ha dei crawler interni che indicizzano gran parte del web cosicché i dati sono direttamente disponibili agli utenti, il servizio fornisce inoltre la

[3]https://www.import.io/

[4]http://www.octoparse.com

ricerca su copie indicizzate nel passato fino a 30 giorni. Effettua lo scraping soltanto su certi siti, è possibile inserire un sito ma soltanto se presente nella lista, possibilità di filtrare i siti per forum, blog o news. Il piano base costa 50$ al mese e fornisce 5.000 richieste, il piano gratuito ne fornisce 1.000.

Apifier[6] Servizio cloud-based di crawling che prende una lista di siti web come parametro per avviare un crawler, è possibile impostare il pattern di URL da seguire. Ad ogni pagina ottenuta, viene eseguita una funzione in javascript e quindi è possibile utilizzare il servizio anche come scraper. ha molte impostazioni configurabili dall'interfaccia grafica e fornisce API per l'accesso da applicazioni esterne. Il servizio è gratis per gli sviluppatori, infatti con il piano gratuito si hanno 10.000 richieste al mese con rotazione dell'IP per non essere bloccati dal sito analizzato anche se è possibile eseguire una sola esecuzione parallela. Il piano base costa 19$ e fornisce, oltre a 50.000 richieste e 3 esecuzioni parallele, più IP per la rotazione dell'IP.

Scrapy[7] Uno dei più famosi crawler disponibili, Scrapy non è solo un web crawler ma è un vero e proprio framework open source scritto in Python che permette la creazione di modelli avanzati di crawling riutilizzabili anche da altri utenti. A differenza di altri servizi, essendo Scrapy un framework in Python, è necessaria la scrittura di un codice per eseguire un crawler/scraper e quindi è più complesso da utilizzare ma molto più personalizzabile, avendo accesso diretto al codice. Scrapy fornisce

[5]https://webhose.io

[6]https://www.apifier.com/

inoltre una shell che può essere usata dagli sviluppatori per testare e mettere a punto il codice dello scraper senza necessità di eseguire il crawler.

PHPCrawl[8] PHPCrawl è una libreria/framework scritta in PHP per eseguire crawling di siti web. Supporta una vasta scelta di opzioni per configurare il crawler come ad esempio filtri su URL e tipi di contenuto, gestione dei cookies, gestione del file robots.txt, diverse opzioni per limitarlo e gestione dell'esecuzione parallela. Il crawler restituisce il contenuto delle pagine trovate, insieme ai link seguiti e una serie di informazioni sull'esecuzione dello stesso. Richiede la scrittura di codice, come Scrapy.

cURL[9] Famoso software scritto in C che contiene una libreria e uno strumento utilizzabile da linea di comando. La libreria è disponibile per oltre 40 linguaggi di programmazione. Lo strumento da linea di comando è utilizzato per inviare e ricevere file utilizzando la sintassi URL, Supporta vari protocolli tra cui HTTP, HTTPS, FTP, FTPS, SCP, SFTP, TFTP, LDAP, DAP, DICT, TELNET, FILE, IMAP, POP3, SMTP e RTSP. Il comando per ottenere il contenuto di una pagina web è:

```
curl www.example.com
```

Si può anche salvare il contenuto in un file:

```
curl -o example.html www.example.com
```

[7] https://scrapy.org/
[8] http://phpcrawl.cuab.de/
[9] https://curl.haxx.se

GNU Wget - wget[10] Software scritto in C simile a cURL ma che supporta soltanto i protocolli HTTP, HTTPS e FTP. Utilizzabile da linea di comando, fornisce vari comandi per essere usato come un crawler, tra cui la possibilità di ignorare il file robots.txt, di salvare soltanto file con una determinata estensione, di salvare i cookie di sessione per riutilizzarli nelle richieste successive o di impostare un user-agent personalizzato.

[10]https://www.gnu.org/software/wget

Capitolo 3

Sviluppo di un sistema per catalogare indirizzi bitcoin

3.1 Necessità di identificare i proprietari di portafogli bitcoin

L'utilizzo dei bitcoin non è sempre lecito: la difficoltà di risalire alla persona reale nascosta dietro ad un indirizzo bitcoin ha contribuito alla crescita di mercati che offrono sostanze o materiale illegale. La normale rete internet non è però sufficientemente anonima per questo tipo di traffici: tutti gli utenti sono identificati tramite un indirizzo IP durante la navigazione e sono facilmente localizzabili. Una VPN può non essere abbastanza in quanto la struttura del web normale, e dei suoi servizi, non è orientata all'anonimato. Esistono tuttavia le cosiddette *Darknet*: reti a cui si può accedere o utilizzando software specifico o tramite particolari configurazioni. Il termine Darknet viene spesso utilizzato per riferirsi ad un particolare insieme di

Hidden Services[1] della più famosa Darknet al mondo: *Tor*[5].

Tor è un software che abilita un sistema di comunicazione sicuro e anonimo attraverso una rete non raggiungibile tramite la normale connessione internet: per accedere è infatti necessario scaricare dal sito ufficiale[2] il Tor Bundle, un pacchetto che fornisce un browser configurato per garantire l'anonimato, oltre al software necessario per la connessione alla rete Tor stessa.

Grazie all'anonimato fornito dalla rete Tor, abbinato all'uso dei bitcoin per i pagamenti, sempre più utenti hanno aperto traffici illegali nel Dark web. Sono infatti presenti molti mercatini online dove gli utenti offrono i più svariati servizi, tra cui vendita di armi non registrate, vendita di droga o sostanze illegali, vendita di documenti o passaporti falsi, ma anche vendita di banconote contraffatte o carte di credito clonate, fino ad arrivare alla vendita di omicidi su commissione[6]. È quindi utile, per scopi forensi o di analisi e identificazione, poter identificare un utente che ha acquistato merce illegale, utilizzando uno dei pochi dati che è possibile analizzare e di cui rimanga una traccia: gli indirizzi bitcoin.

3.2 Un framework modulare

La rete Bitcoin è molto vasta e contiene diverse componenti interessanti per gli scopi di analisi, come la blockchain o i nodi. Queste componenti vengono utilizzate per recuperare informazioni interessanti, da modificare successivamente per avere una visione complessiva dell'analisi. Le informa-

[1] Servizi nascosti: siti web non indicizzati dai classici motori di ricerca.

[2] https://www.torproject.org/

zioni possono provenire da più fonti, ad esempio forum o social network e possono rappresentare un punto di partenza per un'indagine complessa e approfondita.

Data la struttura particolarmente complessa del progetto generale, si è deciso di utilizzare un approccio di sviluppo modulare, che consenta alle varie componenti di essere strutturalmente separate e indipendenti, ma allo stesso tempo di lavorare e manipolare i dati "globali" o di effettuare delle modifiche di un modulo, senza compromettere tutto il software. Si è quindi ottenuto un framework modulare formato da 6 grandi componenti (figura 3.2):

Nodo Bitcoin - Il nodo bitcoin è un server connesso alla rete Bitcoin. Utilizza BitCore[3] per la connessione e per tutte le funzioni di recupero di blocchi dalla blockchain. Siccome è direttamente connesso alla blockchain, questo modulo si occupa di fornire tutte le API per il salvataggio e l'interrogazione dei blocchi, anche offline.

Database Blockchain - Il database Blockchain consiste in un database e in un'interfaccia che interroga il nodo bitcoin per il recupero di informazioni. Le informazioni recuperate dal blocco tramite il nodo, ad esempio le transazioni, vengono inserite nel database.

Clustering - Il modulo di clustering si occupa di associare e raggruppare gli indirizzi bitcoin che appartengono ad uno stesso portafoglio. Permette quindi di identificare quali indirizzi bitcoin possiede un utente. Questo tipo di clustering si basa principalmente sull'analisi delle transazioni.

[3]BitCore: https://bitcore.io/

Scraping - Una delle componenti di partenza per l'analisi forense è lo scraping, che si occupa di trovare indirizzi bitcoin utilizzati per scopi illeciti grazie all'analisi di pagine web. Lo scraping viene effettuato a partire da siti web che l'utente ha considerato "interessanti" per la presenza di determinante parole chiave, come ad esempio "droga", "armi", "documenti". Lo scraping implementa anche una piccola parte di *clustering sociale* che utilizza degli script per il recupero di informazioni da forum o gruppi di discussione.

Bitcoin Mixnet - Siccome le transazioni sono pubblicamente disponibili e consultabili, esse non sono realmente anonime. Per ovviare a questo problema sono stati sviluppati dei servizi, chiamati *mixing service* , *bitcoin mixnet* o *mixer*, che "mischiano" i bitcoin per renderli non tracciabili (figura 3.1): quando si utilizzano questi mixing service, i bitcoin vengono prima inviati ad un servizio anonimo che provvede, anche con più passaggi di scambio, a farli arrivare al destinatario.

Visualizzazione - L'ultimo modulo permette, interrogando il database per l'elenco delle transazioni, di mostrare quali indirizzi sono collegati da transazioni e quali transazioni ha effettuato un indirizzo. Questo modulo è utile per avere un riscontro visuale del percorso che svolgono i bitcoin.

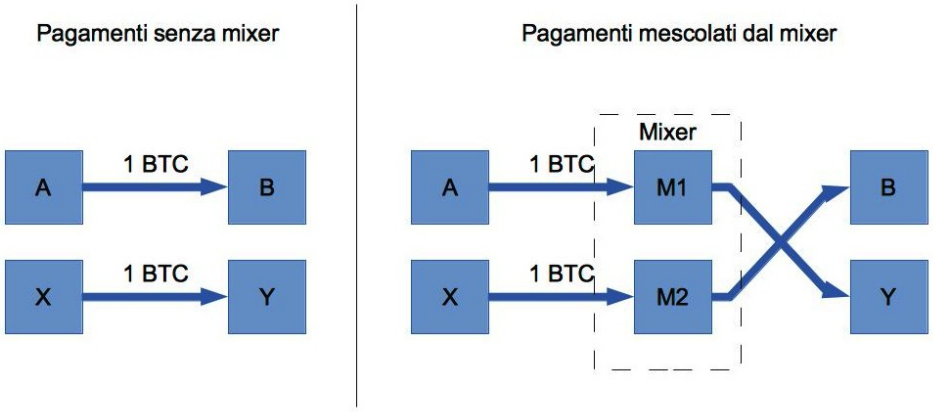

Figura 3.1: Funzionamento di un mixing service.[4]

[4]Fonte: http://www.rischiocalcolato.it/

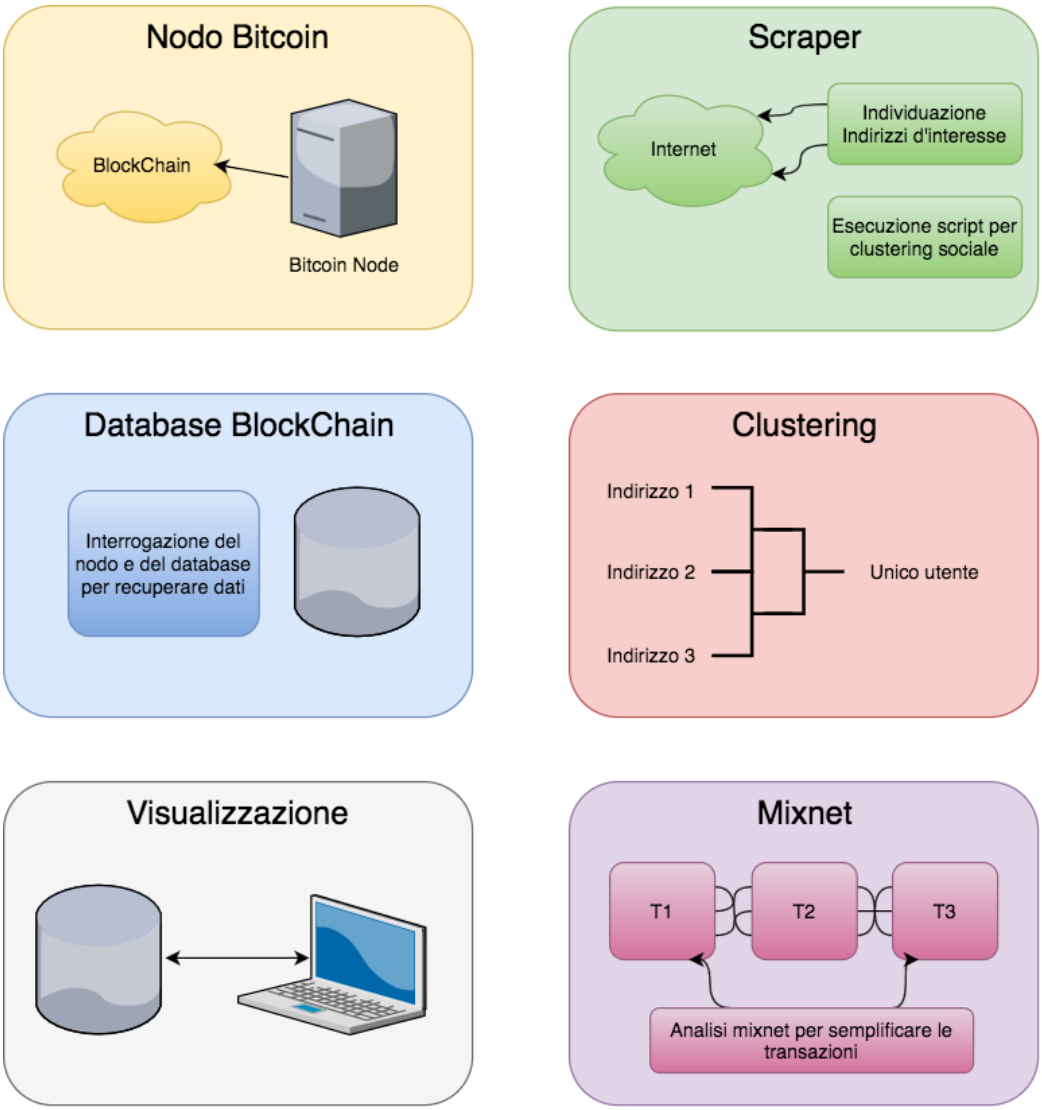

Figura 3.2: Schema del framework sviluppato che mostra i 6 moduli.

3.3 Analisi di una soluzione preesistente: Bitiodine

Un progetto simile è stato sviluppato da tre ragazzi del Politecnico di Milano, M. Spagnuolo, F. Maggi, e S. Zanero, che hanno progettato un framework modulare per l'analisi della blockchain: *BitIodine*[7][8][9].

Il framework, scritto in python, è composto da diversi moduli che possono essere utilizzati per l'analisi. I moduli, visualizzabili dettagliatamente in figura 3.3 sono i seguenti:

Block Parses - Il block parser è il modulo che ha il compito di analizzare i blocchi inseriti nella blockchain e tutte le sue transazioni per inserirle in un database.

Clusterizer - L'obiettivo del modulo è raggruppare gli indirizzi bitcoin che appartengono allo stesso utente. Due algoritmi euristici sono impiegati per lo scopo, il primo sfrutta le transazioni con input multipli, il secondo utilizza il concetto di "resto" e dell'indirizzo di ritorno utilizzato nelle transazioni.

Grapher - Utilizza sia il database della blockchain che le informazioni ottenute dal modulo Clusterizer per creare il grafico delle transazioni, dove i nodi sono gli indirizzi e gli archi sono le transazioni, e il grafico degli utenti, dove i nodi sono gli utenti e gli archi sono le transazioni aggregate effettuate tra gli utenti.

Classifier - Legge i grafi creati dal Grapher e automaticamente etichetta ogni singolo indirizzo o gruppo di indirizzi con annotazioni specifiche.

Le annotazioni vengono recuperate utilizzando degli scrapers mirati su determinati siti web, tra i quali BitcoinTalk e Bitcoin-OTC.

Exporter - L'ultimo modulo serve per esportare e filtrare i grafi in diversi formati, permettendo di interrogare il modulo per ottenere dati specifici, come as esempio le transazioni di un gruppo di indirizzi che sono stati associati allo stesso utente.

BitIodine non è più attivo: molte componenti non funzionano più in quanto i siti web che utilizzavano non sono più online. Uno scraper, ad esempio, prendeva dei dati da *MtGox*: un noto servizio di bitcoin exchange che gestiva circa il 70% di tutte le transazioni. Il servizio ha chiuso i battenti nel 2013 e per questo lo script non è più utilizzabile.

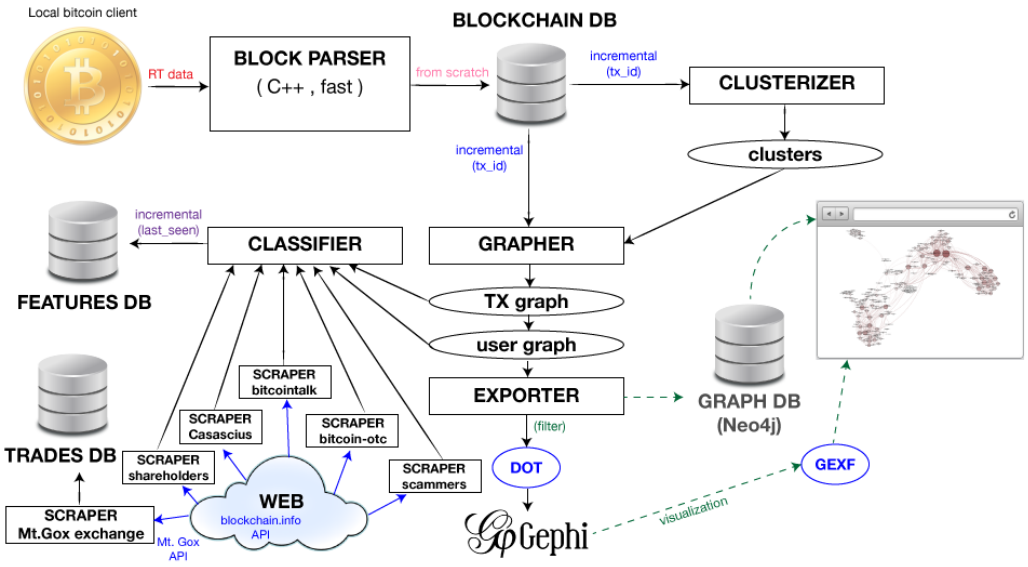

Figura 3.3: Schema del framework BitIodine.[5]

[5]BitIodine - GitHub : https://github.com/mikispag/bitiodine

3.3.1 Differenze tra i due framework

Il progetto descritto si differenzia fondamentalmente da BitIodine per una caratteristica: l'analisi del dark web. Sebbene BitIodine sia stato un framework importante per l'analisi della blockchain, le informazioni recuperate provenivano totalmente dal *clear web*, o web indicizzato. Il progetto sviluppato invece utilizza dei moduli per il recupero di informazioni anche dal dark web, riuscendo quindi a recuperare informazioni più interessanti ai fini di un'analisi forense. Quando BitIodine fu sviluppato, inoltre, non esistevano molti servizi per gli utilizzatori dei bitcoin, come ad esempio i mixing service: Il framework sviluppato consentirà l'analisi di questi nuovi servizi per invertire il processo di mixing durante il processo di clustering.

Capitolo 4

Realizzazione di uno scraper di indirizzi bitcoin

La componente che si occupa del recupero di indirizzi bitcoin deve permettere all'utente di poter analizzare un sito web e di visualizzare il risultato della ricerca, fornendo informazioni sugli indirizzi trovati e sul contenuto della pagina, per permettere di capire di chi è quell'indirizzi bitcoin o perché si trova in quella pagina. I risultati devono inoltre essere organizzati in un database per una consultazione postuma all'attività di recupero.

Per creare un sistema facilmente utilizzabile da qualsiasi utente connesso ad internet, l'applicazione è stata scritta utilizzando il linguaggio PHP: il software è caricato su un server[1] e può essere gestito tramite un sito web che mette a disposizione l'interfaccia grafica in figura 4.1. Alcuni dei software di terze parti utilizzati sono:

- PHPCrawl[2]: Libreria/Framework per il crawling di siti web

[1]Server: http://normandy.dmi.unipg.it/scraping/

[2]http://phpcrawl.cuab.de/

- CodeIgniter[3]: Framework strutturale per la gestione dell'applicazione
- Skeleton[4]: Framework CSS per la gestione della GUI

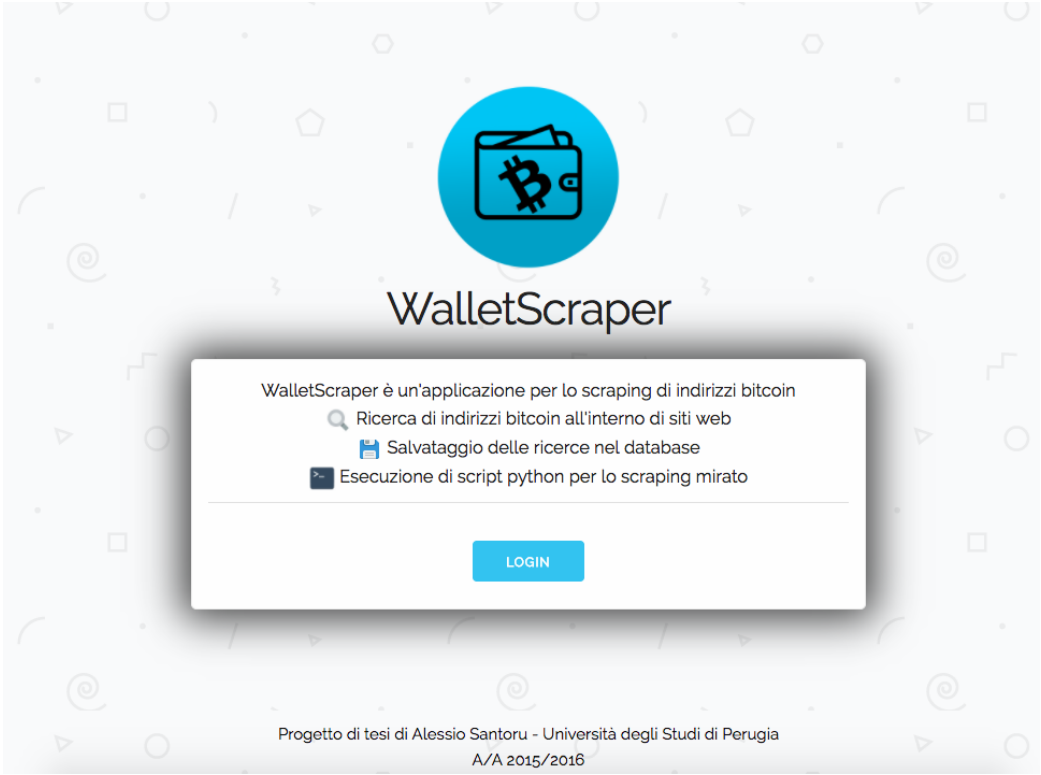

Figura 4.1: Schermata di benvenuto del software.

4.1 PHPCrawl: Web Crawler Framework

PHPCrawl è un framework che consente di realizzare ed implementare un web crawler utilizzando il linguaggio PHP. Il framework permette di effettuare crawling di siti web restituendo informazioni sui collegamenti seguiti e sul contenuto dei file trovati e di configurare il comportamento del crawler grazie

[3]https://www.codeigniter.com/
[4]http://getskeleton.com/

alla possibilità di inserire filtri sui collegamenti da seguire, opzioni per la gestione dei cookie e del file robots.txt e metodi per limitare l'esecuzione del crawler stesso in base al numero massimo di pagine da analizzare o al numero massimo di byte da scaricare. Si può in aggiunta definire l'utilizzo di un proxy per implementare soluzioni più complesse come ad esempio la rotazione dell'indirizzo IP ogni N richieste evitando così di essere bloccato dal sito analizzato per le troppe connessioni.

4.1.1 Modalità di crawling con PHPCrawl

I link che è possibile trovare all'interno di una pagina web non sono tutti uguali: I collegamenti che indirizzano la navigazione in altri siti web sono in genere contenuti all'interno dell'attributo *href* mentre le risorse esterne che vengono incluse nella pagina, come per esempio un foglio di stile, un'immagine o uno script, si trovano nell'attributo *src*. Oltre a questa distinzione, i link possono riferirsi al dominio stesso della pagina, ad un suo sottodominio, oppure ad un dominio esterno. PHPCrawl analizza il contenuto di entrambi gli attributi *href* e *src* permettendo di scegliere se analizzare soltanto all'interno del sito fornito come input o se allargare la ricerca anche a risorse esterne. In particolare il framework fornisce quattro diversi tipi di crawling:

- Ricerca in tutto il dominio: I collegamenti seguiti sono tutti quelli facenti parte dello stesso dominio del sito fornito in input.

- Ricerca limitata all'host: Dalla ricerca vengono esclusi tutti i collegamenti a domini diversi, ma anche quelli a host diversi dall'input. Se la ricerca viene effettuata su un dominio, i link relativi ai sottodomini non vengono analizzati, se la ricerca parte invece da un sotto-

dominio, vengono analizzati soltanto i collegamenti riferiti allo stesso sottodominio.

- Ricerca limitata ai discententi: Dato un indirizzo come input, vengono analizzati soltanto collegamenti che fanno parte dello stesso percorso / directory dell'indirizzo, o di suoi discendenti.

- Analisi di tutto il web: Vengono seguiti tutti i collegamenti trovati senza nessuna restrizione, con questo metodo si potrebbe idealmente effettuare uno scraping completo di tutto il web.

Siccome un sito web potrebbe avere un gran numero di pagine, portando il crawler ad avere un tempo di esecuzione alto e ad effettuare un largo numero di richieste, è possibile definire il numero massimo di collegamenti da seguire, e quindi di pagine da analizzare, o di impostare il traffico dati massimo (in bytes) per ogni esecuzione. Inoltre, data la natura del crawler di seguire ogni collegamento trovato, molti risultati saranno file di tipo immagine, fogli di stile o comunque file che non contengono informazioni utili per lo scraper. Il framework mette a disposizione due metodi per filtrare o selezionare questi risultati:

- Impostazione del tipo di contenuto per restituire soltanto file pertinenti, come ad esempio *text/html* per le pagine di testo HTML.

- Filtro sul formato dell'url applicato utilizzando un'espressione regolare che permette di seguire collegamenti con una struttura definita dall'espressione o di escludere determinati file inutili allo scraper, identificati da un'estensione comune, tra cui immagini (png, jpg, gif, ecc..), fogli di stile css o script js.

4.2 Uno scraper dinamico

L'implementazione dello scraper deve essere dinamica per permettere l'analisi di ogni sito web e quindi non dipendente dalla struttura della pagina. Molti web scraper forniscono un'interfaccia per prelevare un dato scritto all'interno di un tag HTML, basandosi di fatto sulla struttura della pagina e per questo risultano non essere adatti: solitamente gli scraper vengono modellati in base al sito web da analizzare e impiegati per catalogare dei contenuti. Cercare se un sito web contiene un indirizzo bitcoin è ben diverso dal catalogare, per esempio, tutti i prodotti di un negozio elettronico: Nella prima situazione l'indirizzo bitcoin potrebbe trovarsi ovunque nella pagina web, mentre nella seconda situazione, la struttura HTML permette allo scraper di recuperare le informazioni, come codice prodotto, prezzo e descrizione, semplicemente restituendo i contenuti dei tag HTML che strutturano quei dati. Il punto chiave per sviluppare uno scraper dinamico, che funzioni per ogni sito web e per tutte le sue pagine, è quindi basare la ricerca sulla struttura del dato stesso piuttosto che sulla struttura della pagina e dei suoi tag HTML.

4.3 CodeIgniter

CodeIgniter è un web framework open source e leggero per lo sviluppo di applicazioni dinamiche scritte in PHP che mette a disposizione dell'utente molte librerie e strumenti per semplificare la scrittura di codice per le funzioni più comuni, quali ad esempio i metodi per la gestione delle connessioni al database e delle interrogazioni oppure le funzioni di verifica e validazione dei form.

CodeIgniter è basato sul pattern architetturale di sviluppo MVC (Model-View-Controller) per la separazione della logica di visualizzazione dati, dalla sua logica di gestione. Il pattern MVC fornisce appunto tre componenti distinte, Model, View e Controller.

- Model: La componente model (Modello) fornisce la logica di accesso ai dati e alla loro memorizzazione.
- View: La componente view (Vista) gestisce l'output del dei dati e la loro rappresentazione, occupandosi dell'interazione con l'utente del programma.
- Controller: Il controller (Controllore) è la componente che gestisce il model, per la gestione dei dati, e la view per la loro rappresentazione. Fornisce anche un controllo sui dati inviati dall'utente, che generalmente passano per la componente view.

La componente principale di CodeIgniter è il controller, infatti le altre due componenti sono facoltative, nel caso in cui le operazioni da svolgere non debbano né restituire un output visivo né interagire con il database. Il controller è la prima componente che viene invocata quando si visita un url e si occupa quindi di interrogare un model per recuperare dei dati, se è richiesto, o di caricare e visualizzare le view per mostrare un output all'utente. CodeIgniter utilizza uno schema url per il caricamento dei controller, infatti, se abbiamo un sito web "http://esempio.com/", il framework ci permette di impostare un controller predefinito da caricare quando si visita la pagina principale del sito web, mentre gli altri controller sono caricati generalmente utilizzando il seguente schema:

```
http://esempio.com/controller/metodo/parametro1/parametro2/...
```

dove "controller" è il controller da caricare e "metodo" la funzione da invocare, a cui seguono i parametri della stessa. Se nessun metodo è specificato, viene invocato il metodo index() del controller. Come già specificato, se non è richiesto nessun controller, viene caricato il controller predefinito.

CodeIgniter è spesso preferito ad altri framework grazie alla sua facile configurazione iniziale e alla sua leggerezza, infatti non necessita di esser utilizzato fornendo istruzioni da riga di comando e non richiede di imparare un linguaggio ausiliario per la gestione delle sue componenti, inoltre la sua documentazione è ricca di contenuti e fornisce diversi esempi. In una conferenza svoltasi nell'agosto 2008, Rasmus Lerdorf, uno dei creatori del PHP, ha fortemente criticato la lentezza di molti framework PHP sottolineando che tra tutti preferisce CodeIgniter in quanto leggero e veloce[5].

4.4 Sviluppo di un'interfaccia grafica: Skeleton

Skeleton è un piccolo framework css per lo sviluppo di interfacce grafiche (GUI) per siti web. Il framework fornisce un insieme di regole per la gestione del contenuto mediante un sistema a griglia. La griglia (figura 4.2) è formata da 12 colonne fluide che possono essere raggruppate diversamente per ogni riga. Il contenuto è quindi distribuito e organizzato in base al numero di colonne che il contenitore ha a disposizione e questo elimina ogni problema di centramento, allineamento e floating di elementi. Skeleton fornisce inoltre regole per l'aspetto dei componenti grafici di un form e della tipografia

[5]https://www.sitepoint.com/rasmus-lerdorf-php-frameworks-think-again/

generale della pagina ma uno dei punti di forza principali del framework è il supporto completo alle media query per la visualizzazione adattiva in base alle dimensioni del dispositivo, rendendo quindi accessibile il sito web anche da dispositivi portatili.

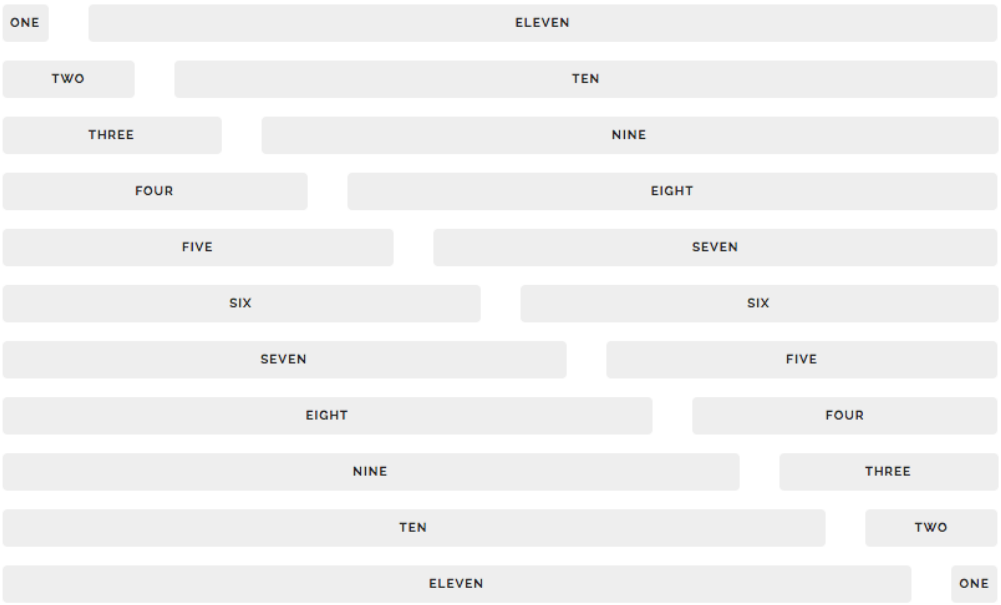

Figura 4.2: Il sistema a griglia di Skeleton che mostra come è possibile affiancare due contenuti su una riga.[7]

4.5 Analisi delle funzionalità

Lo schema dell'applicazione è visualizzabile in figura 4.3: I controllers gestiscono tutte le funzionalità disponibili, interagendo sia con i models per il controllo del database, sia con le librerie che forniscono le classi necessarie

[7]Fonte: http://getskeleton.com/

ad implementare il crawling e lo scraping, così come a verificare gli indirizzi bitcoin. In particolare sono presenti 5 controllers:

Home - Gestisce il form principale di avvio dello scraping, con le impostazioni avanzate per configurare il numero di collegamenti da seguire, la tipologia di crawling e la possibilità di caricare una lista di collegamenti di partenza da un file. È possibile inoltre applicare un filtro sulla ricerca, mediante il form avanzato, che scarta automaticamente i collegamenti non pertinenti, come ad esempio immagini o pagine non contenenti testo.

Search - Controller che gestisce l'esecuzione degli scraper, avviando e configurando sia il crawler che lo scraper e salvando i risultati nel database.

Database - I dati inseriti precedentemente all'interno del database possono essere ricercati da questo controller.

Scripts - È possibile lanciare degli scraper, modellati per particolari siti web, scritti in python.

Tasks - Visualizza tutti i task in esecuzione, sia lanciati dal controller Search (quindi crawling e scraping dall'input dell'utente) sia dal controller Scripts (quindi scraper in python) mostrandone il progresso.

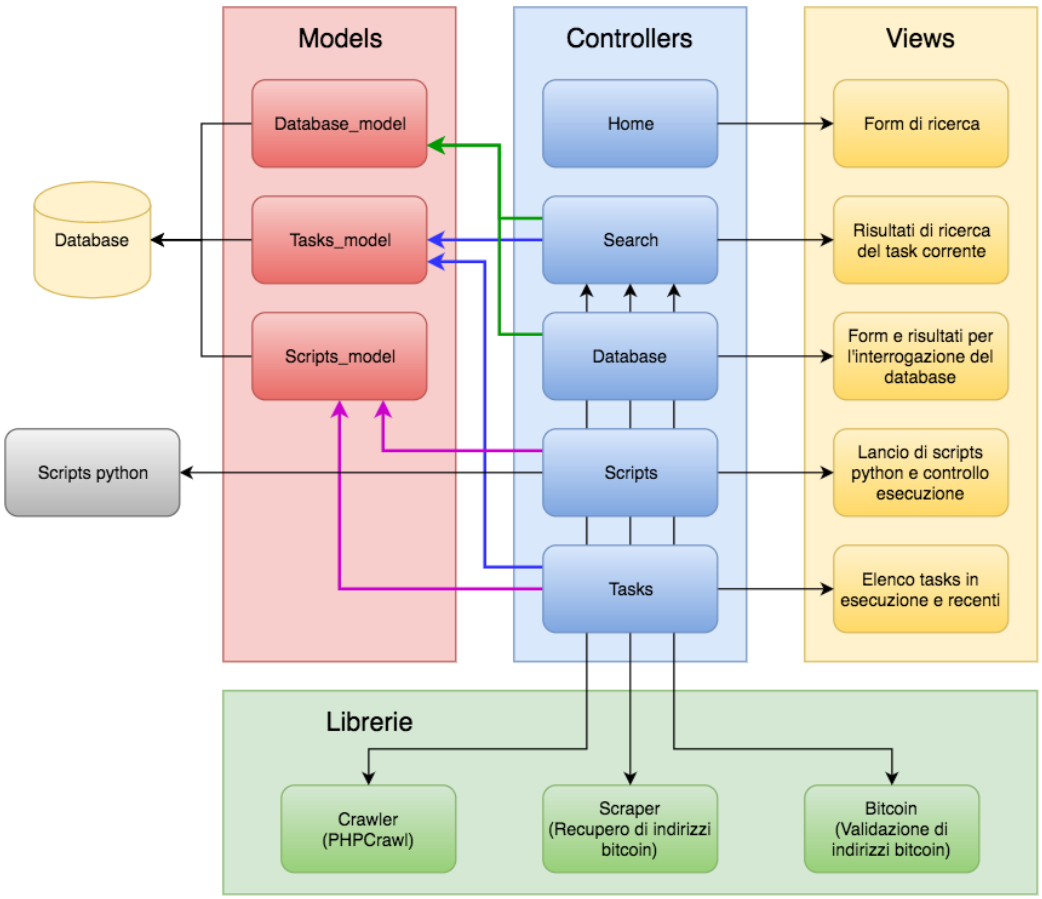

Figura 4.3: Diagramma della struttura del software.

4.6 Esecuzione del crawler

Quando l'utente avvia la ricerca di indirizzi bitcoin dal form (figura 4.4) l'input dell'utente viene utilizzato per configurare e lanciare il crawler (PHPCrawl). Ogni link, sia di partenza che trovato, viene visitato e il contenuto della pagina viene aggiunto ad un vettore. Terminata l'esecuzione del crawler, un vettore contenente tutti i collegamenti seguiti e il contenuto recuperato viene passato allo scraper, che ricercherà indirizzi bitcoin.

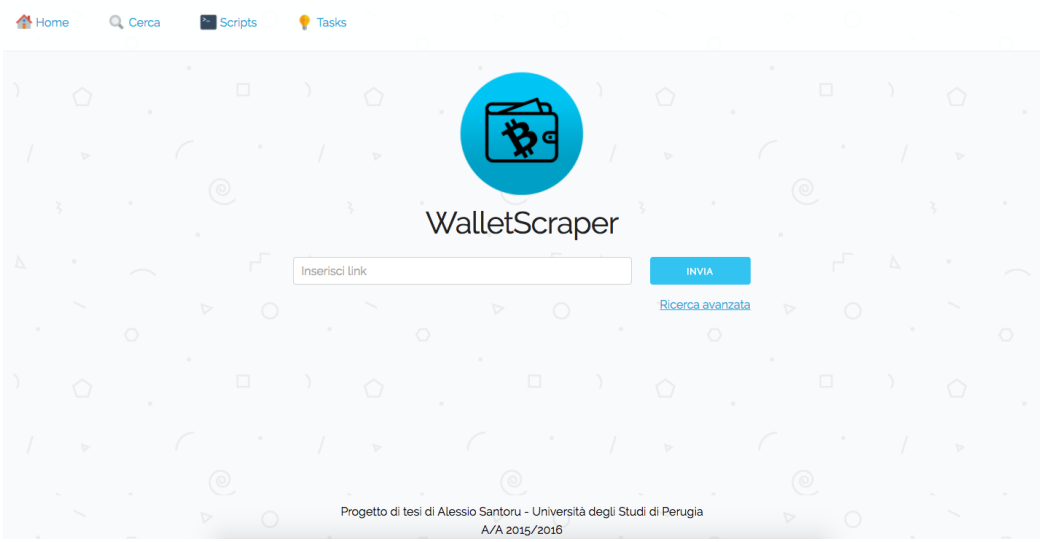

Figura 4.4: Form principale di ricerca.

4.7 Esecuzione dello scraper

La parte più importante dell'applicazione è lo scraper, ossia la componente che cerca all'interno delle pagine web la presenza di indirizzi bitcoin. Dovendo ricercare degli indirizzi bitcoin in tutta la pagina (rif. paragrafo 4.2), si può gestire il contenuto HTML del sito come se fosse una stringa: prelevando tutte le sotto stinghe che potrebbero essere indirizzi bitcoin, per poi verificare che lo siano realmente o meno grazie ad un algoritmo di validazione.

Come spiegato nel paragrafo 1.4, gli indirizzi bitcoin rispettano tutti un certo schema: sono delle stringhe di una lunghezza variabile in un intervallo definito (26-35), che non contengono caratteri ambigui (O 0 l I) e che iniziano, nelle due versioni attualmente utilizzate, con 1 o con 3 perciò si può ricercare nella stringa contenente il codice HTML della pagina web tutte le sotto stringhe che rispettino quel determinato schema, utilizzando

un'*espressione regolare*.

Un'espressione regolare è una speciale sequenza di caratteri che definisce un insieme di stringhe mediante un pattern e che può essere utilizzata per verificare l'appartenenza di una stringa generica a quel sottoinsieme definito. In particolare, l'espressione regolare

```
$regex = '/[13][a-km-zA-HJ-NP-Z1-9]{25,34}/';
```

identifica, nel linguaggio PHP, l'insieme di tutte le stringhe che rispettano lo schema tipico degli indirizzi bitcoin. L'espressione regolare è quindi utilizzata per ottenere la lista di tutti i possibili indirizzi bitcoin contenuti nella pagina, cercando tutte le occorrenze, all'interno del testo HTML, di stringhe che siano validate dal pattern definito. Questo procedimento non è sufficiente per validare un indirizzo: potrebbero esistere all'interno della pagina delle stringhe che rispettino il pattern stabilito dall'espressione regolare, ma che non sono indirizzi bitcoin.

4.7.1 Verificare gli indirizzi bitcoin

L'algoritmo di validazione degli indirizzi bitcoin[8] è utilizzato per verificare se le stringhe conformi al pattern sono effettivamente degli indirizzi bitcoin: siccome gli indirizzi vengono generati a partire da una coppia di chiavi ECDSA e non vengono attivati o registrati in un database, è possibile verificare la loro validità controllando esclusivamente che i bit di controllo (checksum) dell'indirizzo siano validi. Come descritto nel paragrafo 1.4.2 gli indirizzi bitcoin sono codificati in base58. La *codifica base58* è stata scelta dalla comunità rispetto alla classica codifica base64 per dei semplici motivi:

[8] https://rosettacode.org/wiki/Bitcoin/address_validation

- I caratteri ambigui O 0 I l sono presenti nella codifica base64, ma non in base58.
- Una stringa formata da caratteri non alfanumerici è difficile da accettare come numero di account.
- Un'email non formatterà automaticamente l'indirizzo tornando a capo, non essendoci segni di punteggiatura.
- Un doppio click sopra la stringa codificata base58 la seleziona tutta, rendendo più facile e meno soggetto ad errori il copia/incolla.

Le stringhe da validare vengono passate all'algoritmo di validazione degli indirizzi bitcoin che per prima cosa le decodifica (in base58). L'algoritmo di decodifica in base58, utilizzato come subroutine dell'algoritmo di validazione, restituisce la stringa decodificata, se è possibile decodificarla, sennò restituisce il valore booleano *false*. La decodifica controlla sia che non siano presenti i caratteri ambigui, sia che la stringa abbia una lunghezza ammessa per un indirizzo bitcoin.

La stringa decodificata è formata da 25 bytes (paragrafo 1.4.2) 1 per la versione, 20 per la chiave pubblica e 4 per il codice di controllo. Una volta restituita la stinga decodificata, l'algoritmo procede a verificare che il checksum, gli ultimi 4 bytes della stringa, sia rispettato. Siccome il checksum viene generato effettuando l'algoritmo di hashing *SHA256* due volte sulla chiave pubblica ottenuta dall'algoritmo *RIPEMD-160* e prendendo i primi 4 bytes del risultato, l'algoritmo può controllare che il codice di controllo sia valido effettuando questa operazione sui 20 bytes della chiave pubblica e confrontando i primi 4 bytes del risultato, con gli ultimi 4 della stringa decodificata

in base58. Se sono uguali l'indirizzo è valido e l'algoritmo restituisce il valore booleano *true* altrimenti *false*.

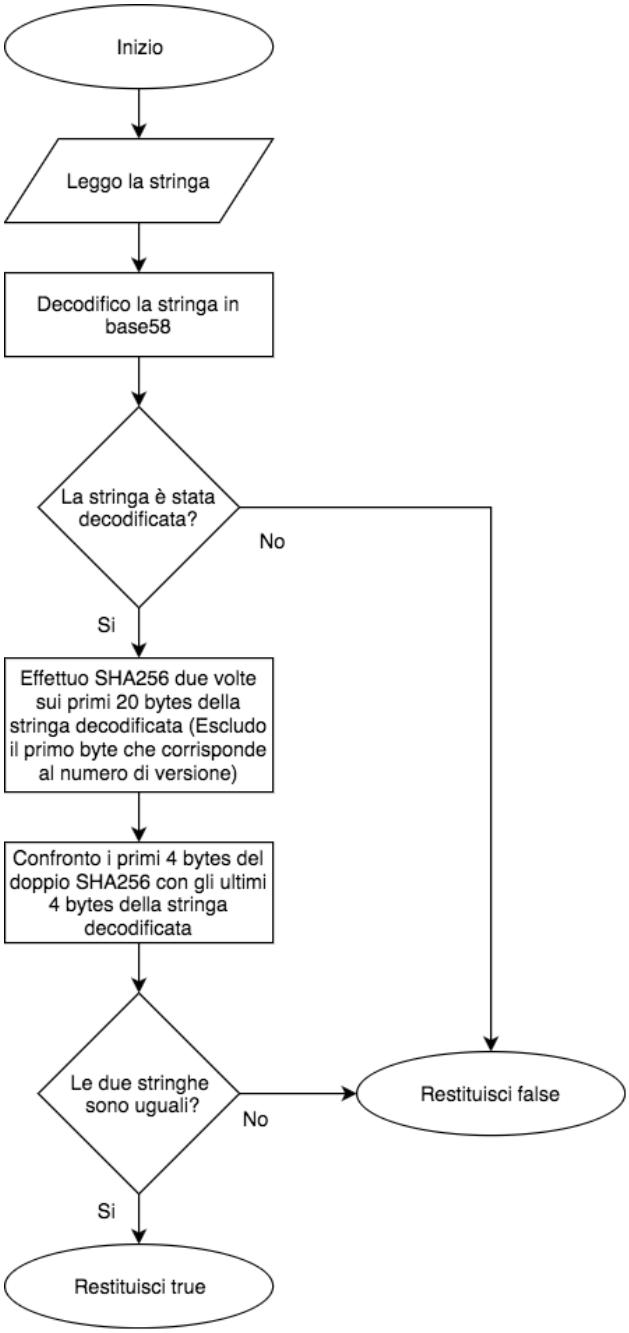

Figura 4.5: Flowchart dell'algoritmo di validazione degli indirizzi bitcoin.

4.8 Caricare una lista di pagine web

L'interfaccia grafica del programma permette l'inserimento di un indirizzo da cui partire con lo scraping (figura 4.4). Nel caso in cui l'utente abbia una lista di indirizzi, il sistema risulta scomodo, dovendo avviare manualmente la scansione per ogni collegamento. Per risolvere questo problema l'applicazione prevede la possibilità di poter caricare un file di testo contenete un elenco di collegamenti (figura 4.6): quando viene fornito in input il file, l'applicazione prova ad avviare la ricerca per ogni URL inserito su ogni riga del file, mostrando quelli trovati e quindi i task avviati.

Figura 4.6: Ricerca avanzata.

4.9 Utilizzo degli scraper di Bitiodine

Il Classifier di Bitiodine, descritto nel paragrafo 3.3, utilizza degli scraper scritti in python per il recupero di informazioni correlate ad indirizzi bitcoin. Qualche scraper non è più funzionante, siccome il sito che veniva analizzato è offline. È stato tuttavia possibile recuperare e utilizzare degli script, in particolare

- bitcointalk.py: Script che analizza il forum BitcoinTalk[9] per trovare gli indirizzi bitcoin appartenenti ai vari utenti. Lo script controlla inizialmente quanti utenti sono iscritti al forum e provvede a visitare il profilo di ogni utente alla ricerca dell'indirizzo. Se è presente, lo salva insieme al nickname dell'utente.

- casascius.py: Casascius[10] è un sito che permette di ottenere delle monete fisiche che "contengono" bitcoin: siccome il concetto di moneta fisica non esiste nel mondo Bitcoin, queste monete non sono altro che "pezzi di metallo" con il logo bitcoin stampato e un ologramma della chiave privata di un indirizzi bitcoin. La chiave privata quindi permette l'accesso ai bitcoin rappresentati dalla moneta. Lo script analizza l'elenco degli indirizzi degli utenti che hanno richiesto una moneta fisica e salva le informazioni, come la quantità di bitcoin "trasformati" in moneta.

- bitcoin-otc.py: Bitcoin-OTC[11] consiste in un mercato Over-The-Counter, ossia un mercato al di fuori del circuito borsistico, per l'acquisto diretto

[9] https://bitcointalk.org/
[10] https://www.casascius.com/
[11] https://bitcoin-otc.com/

tra utenti. Il sito web BlockChain.info fornisce una pagina[12] per l'aggiunta di tags agli indirizzi bitcoin di Bitcoin-OTC che viene analizzata dallo script, per aggiungere informazioni al database.

4.9.1 Eseguire script python da php

Gli script di Bitiodine, scritti in python, possono richiede un tempo di esecuzione dell'ordine delle ore, se non dei giorni. È importante poter controllare la loro esecuzione ed eventualmente ripristinarla in caso di arresto dello script. Per questo motivo i file python sono stati modificati per salvare all'interno del database dei dati utili al controllo: lo script *bitcointalk.py*, con un tempo di esecuzione di molte ore, salva all'interno del database il numero di utenti iscritti al forum e aggiorna un campo contatore per tenere traccia di quelli già analizzati. In caso di arresto, la nuova esecuzione riprenderà partendo dall'ultimo utente inserito nel database. Lo script *casascius.py* preleva una pagina JSON e controlla l'esecuzione confrontando il numero del risultato esaminato con il numero dei risultati totali. Questo script, scaricando il file JSON solamente all'avvio, ed avendo un tempo di esecuzione di circa 10/20 minuti, non necessita di ripartire da dove è stato arrestato ma solo di monitorare la sua esecuzione. Una volta lanciato lo script è quindi possibile seguire la sua esecuzione e eventualmente arrestarlo (figura 4.7).

[12] https://blockchain.info/it/tags?filter=4

Figura 4.7: Esecuzione dello script bitcointalk.py.

4.9.2 Recupero dell'esecuzione di uno script python

Quando l'applicazione lancia gli script in python, l'esecuzione procede in maniera non bloccante e il task viene eseguito in background: grazie alla funzione *exec()*[13] si può lanciare lo script, indirizzando *stdout* a */dev/null* e *stderr* ad un file di errore e salvando il PID del processo lanciato all'interno del database. L'esecuzione degli script può essere terminata dall'applicazione utilizzando il PID precedentemente salvato. Se questo non succede, quando lo script finirà la sua scansione e terminerà, il PID salvato nel database verrà eliminato.

Siccome l'esecuzione degli script avviene in background, al momento del lancio l'applicazione controlla che lo script non sia già in esecuzione cercando nel database se il PID sia presente o meno: nel caso in cui sia già attivo in

[13]PHP - Funzione exec(): http://php.net/manual/it/function.exec.php

background, l'applicazione restituirà informazioni sull'esecuzione dello stesso, come la percentuale di completamento e la possibilità di arrestarlo.

4.10 Controllo di tutti i task in esecuzione e recenti

Nel software sviluppato è presente una pagina per la visualizzazione di tutti i task in esecuzione e delle ultime ricerche effettuate (figura 4.8). Questa pagina controlla e visualizza gli script python in esecuzione, i task che stanno effettuando lo scraping di indirizzi bitcoin in background e gli ultimi task eseguiti e terminati. I task in esecuzione mostrano informazioni come l'orario di avvio, il progresso e il tipo di scansione. I task terminati mostrano il numero di indirizzi bitcoin trovati, la motivazione sulla terminazione del task e la data di fine esecuzione.

Si può visualizzare un elenco degli indirizzi bitcoin trovati in un dominio cliccando sul collegamento nell'elemento della lista dei task terminati. La schermata home mostra un badge accanto alla voce del menù "Tasks" se sono presenti task in esecuzione (figura 4.9).

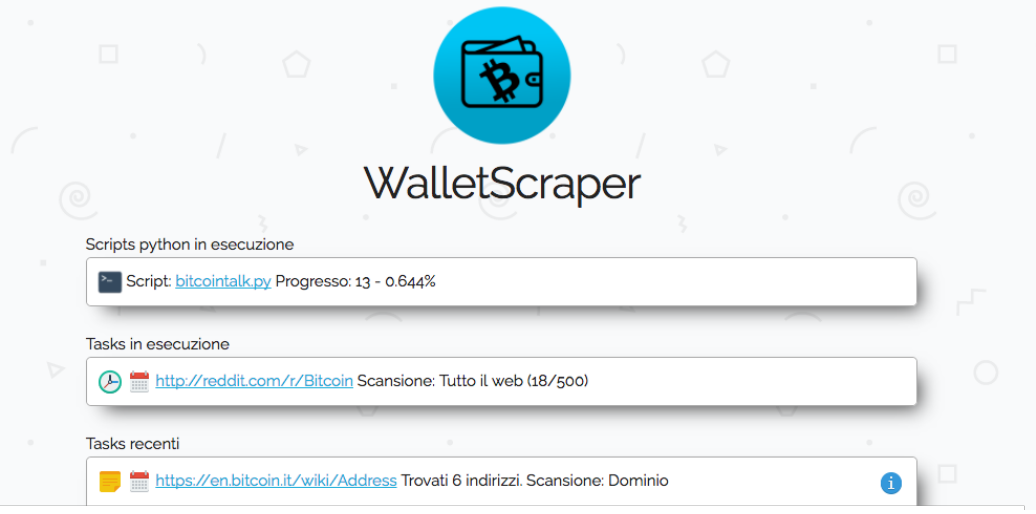

Figura 4.8: La pagina che visualizza tutti i task in esecuzione e gli ultimi terminati.

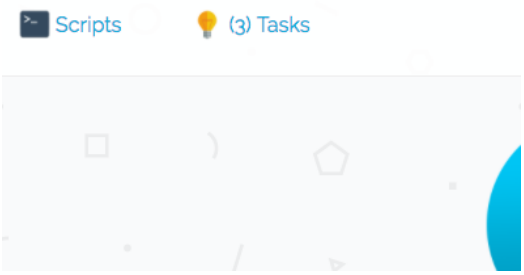

Figura 4.9: Notifica che indica l'esecuzione di tre task.

4.11 Struttura del database

Il database relazione (MySQL) mostrato in figura 4.10 è utilizzato per salvare tutti i risultati delle ricerche effettuate o degli script python eseguiti. Fornisce inoltre una tabella dove sono registrati tutti i task di scraping lanciati, con il

loro progresso, per monitorarne l'esecuzione. Nel dettaglio, le tabelle presenti sono le seguenti:

Websites - L'elenco dei siti web dove è stato trovato almeno 1 indirizzo bitcoin.

Pages - Le pagine, in relazione col sito web, dove sono stati trovati gli indirizzi e il loro contenuto HTML.

Addresses - L'elenco degli indirizzi bitcoin trovati dall'applicazione.

Presences - Tabella che elenca quali indirizzi bitcoin sono presenti in quale pagina web, con la data di ultima scansione.

Informations - Tutte le informazioni ottenute per un indirizzi bitcoin sono salvate nella tabella *Informations*, tra cui quelle ottenute dagli script python.

RunningCheck - Tabella di controllo che monitora l'esecuzione degli script python.

Tasks - Tabella di controllo che monitora l'esecuzione dei task di scraping.

La ricerca dei dati all'interno del database viene gestita tramite un form, in figura 4.11.

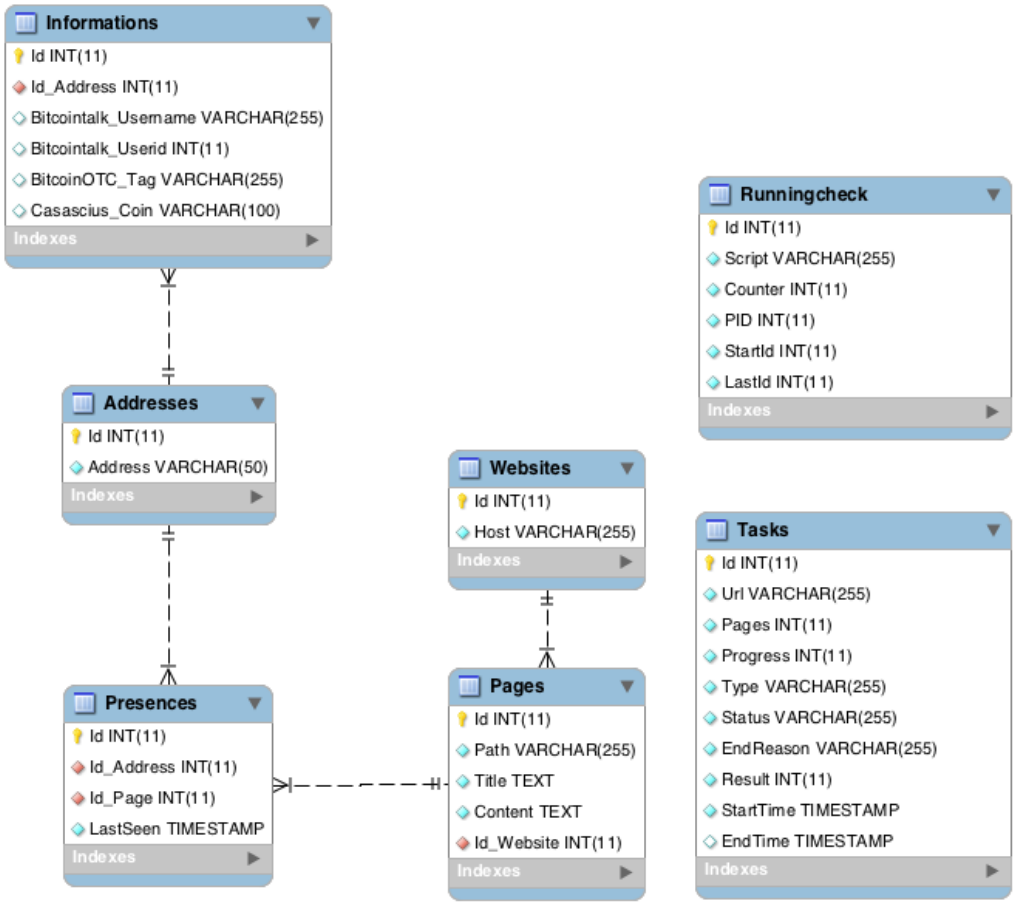

Figura 4.10: Schema del database relazionale.

Figura 4.11: Form di ricerca nel database.

Capitolo 5

Risultati

Il progetto è stato sviluppato utilizzando una metodologia agile: ci si è focalizzati maggiormente sull'obiettivo finale piuttosto che su una strutturazione dello stesso. La progettazione è stata condizionata da numerose aggiunte: da un tool per lo scraping di indirizzi bitcoin, si è arrivati ad un'applicazione che effettua non solo scraping, ma anche parte di *Social Clustering*, utilizzando degli script in python e associando delle etichette ai vari indirizzi.

5.1 Test e validità

Durante tutto lo sviluppo del progetto, sono stati raccolti file di log per monitorare l'esecuzione delle varie ricerche e controllare eventuali errori.

I test sono stati eseguiti utilizzando siti web dove la presenza di indirizzi bitcoin era conosciuta ed era noto il numero di indirizzi. Il programma ha recuperato tutti gli indirizzi aggiungendoli correttamente al database. L'algoritmo di validazione risulta corretto, essendo stato preso da una risorsa

"ufficiosa" e venendo utilizzato anche da altri servizi web che verificano gli indirizzi bitcoin.

5.1.1 Situazioni a carico intensivo

Il programma può eseguire parallelamente diversi tasks. Per prevenire tasks infiniti o particolarmente lunghi sono stati definiti dei limiti costanti al crawler:

Il numero di pagine che è possibile analizzare da un singolo task sono al momento 500.

Il massimo numero di bytes che è possibile trasferire da un singolo task è di circa 50 Mb.

Un altro limite all'esecuzione dei task è imposto dal PHP tramite il valore di *max_execution_time* inserito nel file di configurazione *php.ini* e modificabile, se autorizzato, tramite la funzione *set_time_limit()*[1].

L'ultimo limite riscontrato dipende implicitamente dal server: siccome l'applicazione utilizza la memoria RAM sia per gestire la coda di crawling che la cache di ricerca, per i task pesanti si potrebbe raggiungere il limite di memoria RAM destinata al PHP e impostata dal file *php.ini* con il valore *memory_limit*.

5.2 Possibili miglioramenti

Uno dei punti più critici dell'applicazione è la gestione di task che effettuano scraping di molte pagine, sopratutto per i limiti di timeout e RAM imposti dal

[1] set_time_limit(): http://php.net/manual/en/function.set-time-limit.php

file di configurazione di PHP. Il limite della memoria RAM può essere trattato utilizzando una funzione messa a disposizione dal framework PHPCrawl che attiva il salvataggio della cache su un database SQLite piuttosto che sulla RAM.

Per mitigare il limite del timeout si dovrebbe agire modificando il valore di *max_ execution_ time* tramite il file *.htaccess*, senza quindi intervenire direttamente sul file di configurazione.

Un'altra miglioria da apportare consiste nel migliorare la visualizzazione dei dati inseriti nel database: siccome i dati devono essere utilizzati da altre componenti, non ci si è concentrati molto su una visualizzazione completa delle informazioni già salvate. Sarebbe tuttavia implementatile una migliore ricerca nel database e la sua rappresentazione nella pagina web.

Conclusioni

Inizialmente lo sviluppo del software ha richiesto particolari attenzioni: una prima versione fu implementata utilizzando *Apifier*[2], un servizio di scraping gratuito che fornisce API per il controllo dei task. Tuttavia, utilizzando queste API di terze parti, l'esecuzione avviene su server esterni, che non possono essere controllati interamente. Siccome gli utilizzi futuri del software prevedono che venga utilizzato per effettuare scraping di indirizzi bitcoin all'interno del *dark web*, non raggiungibile tramite la normale rete internet ma dovendo passare attraverso la rete TOR[3], si è dovuto sviluppare un sistema di crawling/scraping che fosse controllabile al 100%. Per questo motivo la prima versione che utilizzava le API di terze parti è stata fortemente modificata, implementando la libreria già descritta, PHPCrawl. L'applicazione è attualmente disponibile online ma non utilizzata, infatti il modulo sviluppato, facendo parte di un progetto più grande, funge da estensione per gli altri moduli e quindi non viene usata a sé. Durante lo sviluppo ci si è concentrati particolarmente sulla facilità d'uso, sviluppando un'interfaccia grafica quanto più intuitiva e auto-descrittiva. Un'altra parte che ha chiesto particolare concentrazione è stata la gestione degli scripts python: non è stato facile

[2]https://www.apifier.com/

[3]https://torproject.net/

lanciare degli script python da controllare utilizzando il PHP, sopratutto per la gestione asincrona dei task.

Appendice A

Codice

A.1 Algoritmo: Decodifica base58

```
function decodeBase58($input) {
    $alphabet = "123456789
       ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz
       ";

    $out = array_fill(0, 25, 0);
    for($i=0;$i<strlen($input);$i++){
        if(($p=strpos($alphabet, $input[$i]))===false){
            return false;
        }
        $c = $p;
        for ($j = 25; $j--; ) {
            $c += (int)(58 * $out[$j]);
            $out[$j] = (int)($c % 256);
```

```
                $c /= 256;
                $c = (int)$c;
            }
            if($c != 0){
                return false;
            }
        }
        $result = "";
        foreach($out as $val){
            $result .= chr($val);
        }
        return $result;
    }
```

A.2 Algoritmo: Validare indirizzi bitcoin

```
    function validateAddress($address) {
        $decoded = self::decodeBase58($address);
        if (!$decoded) {
            return false;
        }
        $d1 = hash("sha256", substr($decoded,0,21), true);
        $d2 = hash("sha256", $d1, true);

        if(substr_compare($decoded, $d2, 21, 4)){
            return false;
        }
```

```
        return true;
    }
```

Bibliografia

[1] S. Nakamoto:

Bitcoin: A Peer-to-Peer Electronic Cash System (2009),

https://bitcoin.org/bitcoin.pdf

[2] G. Gutoski e D. Stebila:

Hierarchical deterministic Bitcoin wallets that tolerate key leakage,

Proc. 19th International Conference on Financial Cryptography and Data Security (FC) vol. 8975 (2015)

https://eprint.iacr.org/2014/998.pdf

[3] Richard L. Apodaca:

Five Ways to Lose Money with Bitcoin Change Addresses (2014)

http://bitzuma.com/posts/five-ways-to-lose-money-with-bitcoin-change-addresses/

[4] Thom File:

Computer and Internet Use in the United States

United States Census Bureau (2013)

http://www.census.gov/prod/2013pubs/p20-569.pdf

[5] Dr. V. Ciancaglini, Dr. M. Balduzzi, R. McArdle, e M. Rösler:
Below the Surface: Exploring the Deep Web
Trend Micro: View research paper (2015)
https://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp_below_the_surface.pdf

[6] V. Ciancaglini, M. Balduzzi:
Cybercrime in the Deep Web
Black Hat Talk (2015)
https://www.blackhat.com/docs/eu-15/materials/eu-15-Balduzzi-Cybercrmine-In-The-Deep-Web-wp.pdf

[7] M. Spagnuolo:
BitIodine: Extracting Intelligence from the Bitcoin Network (Thesis)
https://www.politesi.polimi.it/bitstream/10589/88482/3/thesis.pdf

[8] M. Spagnuolo, F. Maggi e S. Zanero:
BitIodine: Extracting Intelligence from the Bitcoin Network,
Financial Cryptography (2014)
https://www.ifca.ai/fc14/papers/fc14_submission_11.pdf

[9] M. Spagnuolo, F. Maggi e S. Zanero:
BitIodine: Codice,
https://github.com/mikispag/bitiodine