Università degli Studi di Pisa

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

Studio del Dust Attack: un attacco all'anonimato di Bitcoin

Candidato:

Jacopo Raffi

Relatori/Relatrici:

Prof.ssa Laura Emilia Maria Ricci

Prof. Damiano Di Francesco Maesa

Anno Accademico 2021-2022

Indice

Ringraziamenti

Ringrazio la mia famiglia, in particolare i miei genitori e mio fratello, per avermi sostenuto nei momenti più bui della mia vita

Ringrazio Sara non solo per avermi sostenuto nei miei momenti difficili ma anche per avermi corretto lo stile delle relazioni scritte durante questi tre anni di università

Vorrei inoltre ringraziare i miei amici universitari Simone, Giacomo e Dario per la loro compagnia, per il loro aiuto, per tutte le risate che abbiamo fatto insieme e in generale per avermi fatto trascorrere dei bellissimi momenti durante questi tre anni di università.

Infine ci tengo a ringraziare i miei due relatori, la prof.ssa Ricci e il prof. Di Francesco, il Dott. Loporchio e la prof.ssa Bernasconi per avermi permesso di svolgere la tesi con loro e per tutta l'immensa pazienza e disponibilità che hanno avuto nei miei confronti.

Capitolo 1

Introduzione

Se ripercorriamo la storia dell'uomo, possiamo osservare come i mezzi per lo scambio di beni tra le persone siano profondamente mutati nel corso del tempo, in relazione alla trasformazione culturale e tecnologica che ha intrapreso l'umanità: dal baratto siamo passati alle monete d'oro, fino ad arrivare all'immaterialità degli assegni o delle carte di credito. Il concetto di denaro stesso, nel suo significato più generale di mezzo per consentire lo scambio di valore, non è sfuggito alle trasformazioni provocate dalla diffusione di Internet, portando alla nascita di una tipologia di valuta completamente nuova denominata **criptovaluta**.

Negli ultimi anni sono stati promossi diversi tipi di criptovalute, ognuna con i propri protocolli e le proprie specifiche tecniche, però la prima criptovaluta sviluppata e che ha acquisito maggior valore nel tempo è senza dubbio **Bitcoin** [1].

Bitcoin nasce con l'intento di creare una moneta completamente libera da controlli di tipo governativo o bancario, che consenta agli utenti di effettuare transazioni attraverso una rete decentralizzata secondo il modello Peer-to-Peer. Quindi la validazione della correttezza delle transazioni non è gestita da una banca o da un ente di terze parti, ma è gestita da tutti i nodi della rete. Il fatto che ogni utente possa generare le proprie transazioni introduce una serie di problemi che non sono presenti nei sistemi di scambio tradizionali. Il problema più noto viene definito "Double Spending". Prevenire il double spending significa impedire che un utente possa spendere più volte lo

stesso importo in transazioni differenti; questo problema non esiste nei sistemi tradizionali perché lo scambio di valore avviene tramite ente centralizzato che ha il controllo dei fondi degli utenti.

Il libro contabile elettronico contenente tutte le transazioni è implementato tramite una struttura dati completamente pubblica e immutabile: la blockchain, all'interno della quale sono registrati tutti i movimenti di Bitcoin a partire dalla prima transazione, fino alla data corrente. Nella blockchain quindi è possibile osservare tutte le transazioni eseguite dai vari address di Bitcoin, questo porta a un serio problema legato alla privacy degli utenti.

Un altro concetto importante per comprendere il funzionamento di Bitcoin è quello di anonimato: nel caso di Bitcoin è più corretto utilizzare l'espressione **pseudo-anonimo**. Ogni utente che partecipa alla rete Bitcoin infatti viene identificato non dal proprio nome o cognome, bensì da un address, che nulla lascia trasparire sulla reale identità dell'utente associato a quell'address. Un utente inoltre può utilizzare address differenti ogni volta che esegue una transazione, rendendo molto complesso dedurre l'identità dell'utente associato ad un insieme di address.

E' importante notare come Bitcoin permetta agli utenti di conoscere le transazioni validate precedentemente, quindi l'intero storico delle transazioni è reso pubblico a chiunque.

Nel corso degli anni sono stati sviluppati diversi attacchi in grado di violare l'anonimato di Bitcoin [2], la tipologia di attacchi più studiata è quella basata sull'analisi della blockchain, e quindi dell'intero storico delle transazioni. L'obiettivo di questi attacchi è la violazione dell'anonimato di un utente ottenuta aggregando diversi address in cluster, ciascuno associato ad un singolo utente; questo avviene tramite l'analisi della blockchain e l'utilizzo di particolari regole euristiche.

Lo scopo di questa tesi è l'analisi di un attacco di de-anonimizzazione definito **Dust Attack**. Il Dust Attack basa la propria strategia sull'invio del dust, una piccola quantità di criptovaluta il cui valore è sotto i limiti minimi di scambio. Per questo motivo è necessario, per poter spendere un importo dust, aggregare tale importo ad altri importi di maggior valore. L'euristica su cui si basa il Dust Attack afferma che tutti gli address di input di una transazione appartengono allo stesso utente, ed è basato sul fatto che per

spendere i valori in input, occorre conoscere le chiavi private associate ad ogni address di input. È quindi probabile quindi che tutti questi address appartengano allo stesso utente. L'obiettivo dell'attaccante quindi è inviare il dust affinché venga speso insieme ad altri input in modo da poter collegare tutti gli address e formare un unico cluster da associare ad un singolo utente. In questo modo non solo l'attaccante può tracciare l'attività di un utente ma se scopre l'identità del proprietario di uno di questi address automaticamente scopre che tutti gli address del cluster appartengono al medesimo utente. Dopo aver scoperto l'identità del proprietario è possibile eseguire elaborati attacchi di phishing atti a rubare le chiavi private degli address così da poter rubare i fondi di quell'utente.

Il dataset su cui si basano le analisi è stato ottenuto mediante un filtraggio della blockchain di Bitcoin, le transazioni sono salvate in un apposito file testuale la cui dimensione si aggira intorno ai 40 GB. E' stato necessario però filtrare il dataset proprio per considerare solo le transazioni di interesse, ovvero tutto le transazioni che generano o spendono importi dust. Successivamente sono state ignorate tutte le transazioni generate da Satoshi Dice, un noto servizio di gambling nato nell'Aprile 2012. Una volta ottenuto il dataset filtrato, la cui dimensione si aggira intorno ai 335 MB, gli input e gli output delle transazioni rimanenti sono stati salvati in appositi file csv. Prima di effettuare le analisi sul dust è stato necessario ignorare tutti gli output dust generati con lo script OP_RETURN, poichè questo script impedisce al destinatario di spendere l'importo ricevuto; quindi il dust associato con lo script OP_RETURN non può essere usato per la de-anonimizzazione. Successivamente sono state prodotte diverse statistiche sulla generazione e sul consumo del dust, tramite i risultati ottenuti è stato possibile trovare due pattern con degli schemi ben precisi e che potrebbero rappresentare schemi di Dust Attack.

Abbiamo valutato l'evoluzione del dust nel tempo evidenziando gli anni in cui si è avuta maggiore produzione di dust. Quindi abbiamo analizzato come vengano spesi gli output dust, distinguendo casi in cui vengono spesi insieme ad altri input con address diversi realizzando quindi un possibile attacco di successo, casi in cui vengono spesi insieme ad altri input ma con il medesimo address, casi in cui viene speso mediante servizi di "dust-collecting" e infine

casi in cui non viene speso. Infine abbiamo valutato i cluster che si sono formati dalle transazioni che hanno speso il dust con altri address, per poi analizzare le transazioni che hanno inviato il dust solo ad address che non compaiono per la prima volta nella blockchain di Bitcoin. Infine abbiamo considerato queste transazioni e abbiamo individuato dei pattern interessanti che potrebbero rappresentare un caso di Dust Attack.

Nel capitolo 2 verrà descritta la tecnologia alla base di Bitcoin. L'attenzione sarà posta sulla blockchain, sul funzionamento delle transazioni e sul relativo protocollo di pagamento, verrà introdotto in breve l'UTXO set e infine verrà descritto il problema relativo all'anonimato di Bitcoin mostrando alcune euristiche utilizzate per la de-anonimizzazione degli address.

Nel capitolo 3 verrà spiegato più nel dettaglio il significato di dust e i suoi possibili utilizzi. Verrà descritto in dettaglio il Dust Attack, spiegando il suo funzionamento, le sue conseguenze e le possibili contromisure.

Nel capitolo 4 invece verranno mostrati il formato dei dati e alcuni algoritmi realizzati per l'analisi dei dati.

Il capitolo 5 mostrerà le statistiche realizzate. Le analisi vertono principalmente sul dust, quanti output dust sono stati generati, quanto dust è stato speso e in particolare se abbia avuto conseguenze sulla de-anonimizzazione dei wallet. Il capitolo descriverà anche alcuni pattern che rappresentano un possibile comportamento di un attaccante. Infine il capitolo 6 presenterà le conclusioni e introdurrà alcuni possibili sviluppi futuri.

Capitolo 2

Background

In questo capitolo verrà descritta la tecnologia alla base del protocollo Bitcoin. Particolare attenzione verrà data alla blockchain, ossia la struttura dati che costituisce il libro contabile dove vengono registrate le transazioni, al funzionamento delle transazioni e agli indirizzi. Infine verrà esposta la problematica relativa all'anonimato degli utenti di Bitcoin.

2.1 Funzioni hash

Il meccanismo utilizzato da Bitcoin per garantire l'integrità e l'immutabilità della blockchain sono le funzioni hash.

Una funzione hash f: X $\longrightarrow$ Y è una funzione matematica avente come dominio X e codominio Y, insiemi finiti tali che $|X| >> |Y|$. Tale funzione prende in input elementi di X di lunghezza qualsiasi e produce in output, a prescindere dalla lunghezza dell'input, stringhe binarie di dimensione fissa, i cosiddetti fingerprint, chiamati anche immagini hash o semplicemente hash.

Una proprietà fondamentale delle funzioni hash è relativa al tempo necessario al loro calcolo: devono essere calcolate efficientemente, ossia a fronte di un input di m bit, la complessità computazionale per produrne il fingerprint deve essere $O(m)$, lineare o comunque polinomiale nei bit su cui è rappresentato l'input. Vista la grande differenza di cardinalità tra i due insiemi X e Y, inevitabilmente alcuni input diversi della funzione hash avranno la stessa immagine; questo fenomeno è detto collisione: x_1 e $x_2 \in$ X, con $x_1 \neq x2$,

collidono se la loro immagine hash è la stessa f (x_1) = f (x_2). In crittografia si usano alcune famiglie di funzioni hash molto particolari, dette funzioni hash one-way o funzioni hash crittografiche, le quali devono rispettare altre importanti proprietà oltre a quelle descritte sopra:

1. **Proprietà di one-way**: dato y ∈ Y, output della funzione f, deve essere computazionalmente difficile invertire la funzione, ossia trovare un x ∈ X tale che f (x) = y. Il termine one-way significa proprio questo: una funzione hash "facile" da calcolare, ovvero di complessità polinomiale rispetto al numero di bit dell'input, ma "difficile" da invertire ovvero di complessità esponenziale, il che rende l'inversione praticamente inattuabile.

2. **Proprietà di claw-free**: Per la funzione f, deve essere computazionalmente difficile determinare due elementi x_1 e x_2 ∈ X, $x_1 \neq x_2$, tali che f (x_1) = f (x_2). Ciò significa che per una funzione hash crittografica non deve essere possibile trovare praticamente due elementi che collidono.

2.2 Bitcoin

Bitcoin, l'unità monetaria elettronica a cui facciamo riferimento in questa tesi, è stata sviluppata da Satoshi Nakamoto, un misterioso autore la cui identità resta a tutt'oggi ignota, tanto da indurre molti a pensare che si tratti di uno pseudonimo, o che dietro a tale nome si celi in realtà non una singola persona, ma addirittura un gruppo di ricercatori o di informatici. L'articolo in cui viene presentato l'intero protocollo Bitcoin viene pubblicato nel 2008, sotto il nome di "Bitcoin: A Peer-to-Peer Electronic Cash System" [3]; l'articolo contiene la descrizione dettagliata del protocollo alla base del funzionamento di Bitcoin.

La peculiarità di tale sistema è l'uso di una rete Peer-to-Peer utilizzata per effettuare, diffondere e validare le transazioni. L'intero storico delle transazioni viene mantenuto in un libro contabile distribuito e di pubblica consultazione. La grande e difficile sfida che Bitcoin dunque si pone è quella di coniugare l'anonimato degli utenti con un'alta affidabilità relativamente alle transazioni e alla loro validità e integrità.

A fronte della sfida tra trasparenza e affidabilità, è fondamentale definire un'implementazione del libro contabile che impedisca alterazioni di transazioni già registrate e validate: ricordiamo che in questo contesto paritario e distribuito, nessun controllo viene effettuato da parte di entità centrali, come per esempio le banche.

La soluzione ideata da Nakamoto per garantire l'integrità dello storico delle transazioni è stata quella di implementare il libro contabile tramite una particolare struttura dati: la blockchain. Come mostrato in figura 2.1, questa struttura si compone di una serie di blocchi collegati tra di loro come in una catena: ogni blocco racchiude un insieme di transazioni effettuate in un certo periodo temporale.

Il blocco corrente, non ancora inserito, contiene le ultime transazioni la cui legittimazione deve essere ancora approvata, mentre i blocchi precedenti, già agganciati alla catena, si riferiscono a transazioni già validate, e che possono essere considerate immutabili. Il meccanismo che garantisce la totale immutabilità della struttura, pena la sua completa invalidazione, è la crittografia.

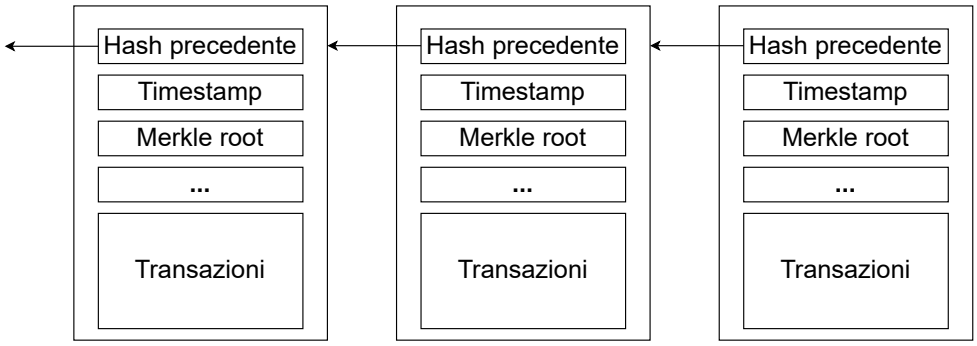


Figura 2.1: Schema della Blockchain

2.3 Blockchain

Un generico blocco B_i all'interno della blockchain contiene la sequenza di transazioni relative ad un certo periodo temporale (supponiamo che esse siano n: T_1 , T_2 , ... ,T_n) e un valore hash h_{i-1} che identifica il blocco precedente nella catena, ed è l'output di una funzione hash crittografica.

è inoltre presente un campo detto nonce, che è il risultato dell'operazione di mining, ovvero del procedimento che porta all'aggiunta di un nuovo blocco alla blockchain. Sono presenti anche altri dati all'interno del blocco, ma al fine di descrivere il meccanismo crittografico che salvaguarda l'integrità della struttura riteniamo sufficiente questo livello di dettaglio.

La Figura 2.2 mostra la struttura di un generico blocco B_i all'interno della blockchain.

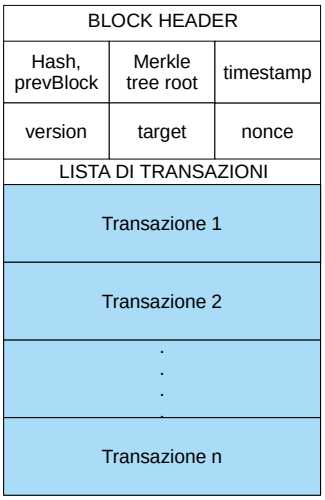


Figura 2.2: Schema di un generico blocco

Come accennato sopra, ogni blocco ha un suo identificativo univoco, una sorta di impronta digitale personale, che è l'output di una opportuna funzione hash.

Un insieme di funzioni hash crittografiche ampiamente usate è quello delle SHA (Secure Hash Algorithm). Una di queste è la SHA-256, funzione hash crittografica che da input di dimensioni variabili produce un fingerprint della lunghezza fissa di 256 bit, ed è la funzione utilizzata per calcolare gli hash dei blocchi all'interno della blockchain di Bitcoin.

L'immagine hash del blocco B_i è calcolata applicando la SHA-256 all'input formato dalla concatenazione delle transazioni contenute in B_i con l'header del blocco, che astraiamo rappresentando i due campi più importanti ovvero il nonce e l'hash del blocco precedente, come riassunto dalla figura 2.3:

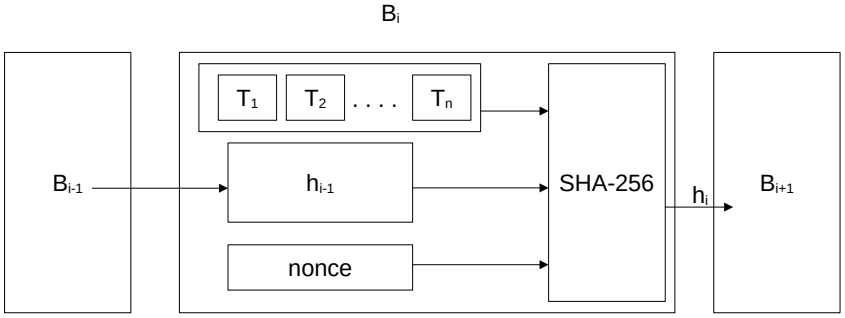


Figura 2.3: Calcolo hash blocco B_i

Mentre le transazioni sono note nel blocco, così come è noto l'hash proveniente dal blocco precedente, il valore del nonce è incognito. L'attività di mining consiste nel risolvere un puzzle crittografico: trovare il valore del nonce tale che l'immagine hash prodotta dalla SHA-256 inizi esattamente con t zeri, dove t è un valore prefissato dal sistema, e variabile nel tempo.

La ricerca del nonce per la corretta aggiunta di un blocco è denominata **Proof of work**. L'unico modo per di trovare il nonce che produca un fingerprint che inizi con t zeri, è quello di applicare ripetutamente la funzione SHA-256 a nonce via via diversi, finchè non si giunge ad un hash che soddisfa la proprietà desiderata. La risoluzione del problema è esponenziale in t, infatti la ricerca del nonce richiede tempo $O(2^t)$, ciò rende il problema di difficile risoluzione, in quanto è necessario far eseguire una serie di calcoli computazionalmente pesanti. Il valore di t viene aggiornato periodicamente dal sistema, in modo che la validazione di un blocco richieda sempre in media circa 10/15 minuti di tempo. Negli anni sono stati progettati calcolatori ottimizzati a livello hardware con lo scopo di calcolare il più rapidamente possibile la funzione SHA-256.

Questo sistema garantisce che i blocchi già inseriti non possano essere modificati retroattivamente: cambiando il contenuto di un blocco, ne cambierebbe anche il valore hash, e ciò implica il dover ricalcolare tutti i nonce dei blocchi successivi ad esso, perché altrimenti le immagini hash non corrisponderebbero più tra blocchi consecutivi; dunque ciò che è stato scritto sulla blockchain è da considerarsi immutabile, a meno di rendere inconsistente

tutta la struttura anche alterandone una sola transazione.

I miner, ossia i nodi della rete che dispongono di nodi di elaborazione abbastanza potenti da risolvere le Proof of Work, sono di fatto gli unici che hanno il diritto di aggiungere transazioni alla blockchain, e nel farlo consumano un gran quantitativo di energia e di risorse di calcolo; per questo, il primo miner che riesce ad agganciare correttamente un blocco alla blockchain, riceve un premio in bitcoin, nella forma di una transazione priva di input e con output l'address del miner: la **coinbase**, registrata anch'essa nella blockchain come una normale transazione. Il premio, di 25 bitcoin nel 2014, viene dimezzato approssimativamente ogni quattro anni per essere definitivamente azzerato nel 2140 quando il numero complessivo di bitcoin esistenti dovrebbe raggiungere 21 milioni.

2.4 Transazioni e Address in Bitcoin

In questo paragrafo verranno approfonditi i concetti stessi di transazione e di address, descrivendo con un grado di dettaglio funzionale ai nostri scopi il protocollo di pagamento e, anche in questo caso, la crittografia che ne è alla base, sia per la generazione degli address che per il protocollo di transazione.

I software wallet di bitcoin, per esempio Samurai Wallet, sono caricati sul PC o sullo smartphone di ogni utente A e generano anzitutto una coppia di chiavi privata-pubblica $kprv_A$, $kpub_A$ per un cifrario asimmetrico su curve ellittiche [4].

La chiave privata $kprv_A$ è ovviamente nota solo ad A e, come vedremo in seguito, è utilizzata da A per firmare le transazioni che genera e diffonde sulla rete. La chiave pubblica $kpub_A$ è utilizzata per controllare la firma di A ed è anche impiegata come suo identificatore: a tale scopo viene trasformata attraverso applicazioni ripetute della funzione hash SHA-256 (immagine a 256 bit), per essere poi compressa via RIPEMD-160 in un'immagine di 160 bit in testa alla quale è aggiunta una speciale sequenza che indica che la stringa complessiva è di fatto un indirizzo bitcoin.

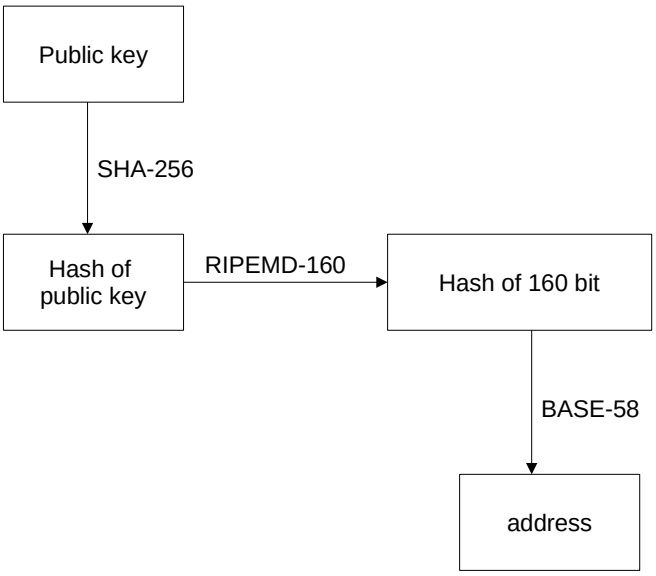


Figura 2.4: Generazione address in Bitcoin

Gli address in Bitcoin si presentano quindi come stringhe alfanumeriche di questo tipo: *1BAFWQhH9pNkz3mZDQ1tWrtKkSHVCkc3fV*.

Come già osservato questo indirizzo non corrisponde a una "locazione" dell'utente ma è un modo per identificarlo per poter inviargli una transazione. L'insieme delle coppie chiave pubblica-privata è contenuto all'interno di un apposito wallet, ad ognuna di queste coppie è associata un indirizzo però nessuno è a conoscenza del fatto che questi indirizzi appartengano allo stesso wallet.

Questo è importante perché nonostante un wallet possa avere più address nessuno è a conoscenza che questi address appartengano al medesimo wallet; è possibile tracciare l'attività dei singoli address ma non l'attività di un intero wallet, garantendo un maggiore anonimato. Oltre a creare le chiavi e gli indirizzi il software cliente permette di costruire le transazioni.

Una transazione in Bitcoin può essere riassunta come "il mittente A vuole inviare X bitcoin al ricevente B", completata dalla firma digitale di A. Si noti che A e B sono indicati attraverso i loro indirizzi Bitcoin e la transazione viene inviata per broadcast a tutti gli utenti. Diversamente da ogni altra forma di

transazione economica, il ricevente B non ha la garanzia che la transazione sia valida finchè, come vedremo, essa non è convalidata dalla rete.

In effetti B sarebbe in grado di verificare sia la firma di A che i fondi che A ha a disposizione, poichè la chiave pubblica di A è nota e tutte le transazioni eseguite sulla rete sono pubbliche, ma non ha la possibilità di verificare che A non abbia utilizzato gli stessi fondi in istanti immediatamente precedenti per pagamenti diversi. Formalmente una transazione, omessi i dettagli tecnici, è innescata secondo il seguente schema:

Protocollo Bitcoin:

1. L'utente A genera un messaggio m = (adr_A, X, adr_B) , dove adr_A , adr_B sono gli indirizzi Bitcoin di A e B, e X è la somma da trasferire da A a B.

2. L'utente A calcola l'hash $h = SHA256(m)$ del messaggio e genera la firma $f = D(h, kprv_A)$ per m.

3. La coppia $< m, f >$ viene diffusa in broadcast da A sulla rete.

4. L'utente B attende che la rete convalidi la transazione prima di accettarla.

La chiave privata è l'unico strumento valido per dimostrare la proprietà in bitcoin da parte di un utente A. La perdita della chiave comporta la perdita della proprietà a causa dell'impossibilità di firmare transazioni, e il furto della chiave da parte di un truffatore che la usi per firmare al posto di A comporta anch'esso la perdita di proprietà.

Un address acquisisce valore in base a quanti bitcoin detiene; quando si usa un address per pagare bitcoin (come input di una transazione), il suo valore viene azzerato, non è possibile spendere solo parte dei bitcoin che costituiscono il valore dell'address. Se un address A paga X bitcoin ad un address B, dopo la transazione il valore di A sarà azzerato, mentre il valore di B aumenterà di X bitcoin. In caso di eventuale resto, nel wallet del pagante si creerà un **change address**, che acquisirà un valore pari al resto ricevuto.

Il change address può essere un address diverso da A, oppure A stesso, a discrezione dell'utente: per rafforzare l'anonimato è estremamente consigliabile da parte del pagante, al fine di ricevere i resti, utilizzare sempre address diversi da quelli usati per effettuare il pagamento.

Quindi, come mostrato in 2.5, gli input di una transazione sono output precedenti di un'altra transazione. Gli output di una transazione possono essere in due stati:

1. **speso**: output che compare come input di una transazione successiva;
2. **non-speso**: output che non sono input di alcuna transazione.

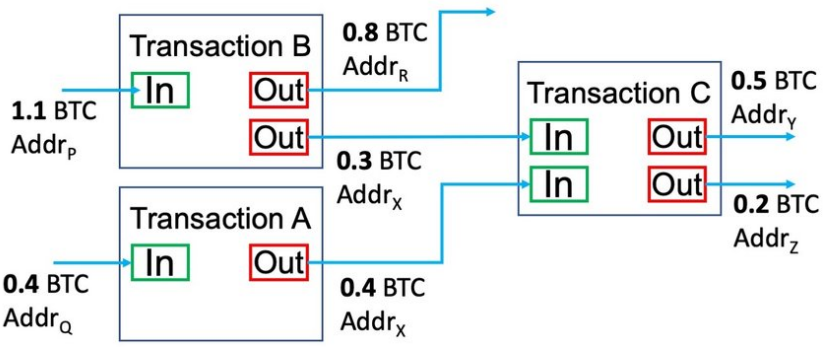


Figura 2.5: Transazioni in Bitcoin

Tutti gli output non-spesi vengono salvati in un unico insieme denominato UTXO(Unspent Transaction Output) set [5].

L'UTXO set è il sottoinsieme degli output delle transazioni Bitcoin che non sono stati spesi e che possono essere utilizzati come input di altre transazioni. Ogni volta che viene creata una nuova transazione, vengono eliminati output dall'UTXO set poichè passano dallo stato non-speso allo stato speso, inoltre vengono inseriti nuovi UTXO poichè ogni transazione genera almeno un output. Fondamentalmente, le transazioni consumano UTXO (nei loro input) e ne generano di nuovi (nei loro output). Poichè l'UTXO set contiene tutti gli output non spesi, memorizza tutte le informazioni richieste per convalidare una nuova transazione senza dover ispezionare l'intera blockchain.

Come suggerisce già il nome, gli UTXO sono effettivamente output di Bitcoin e, come tali, sono costituiti da due parti: l'importo trasferito all'output e lo script che specifica le condizioni da soddisfare per spendere l'output. Uno script è essenzialmente un elenco di istruzioni registrate con ogni transazione che descrivono come la persona successiva che desidera spendere i Bitcoin trasferiti possa accedervi, nel capitolo successivo verrà descritto il sistema di script in Bitcoin.

Il vantaggio principale di utilizzare l'UTXO set è la sua dimensione molto più ridotta rispetto all'intero database delle transazioni (la blockchain), l'UTXO set può essere quindi conservato nella RAM, il che velocizza il controllo di validità delle transazioni. L'algoritmo 1 schematizza la politica di accettazione locale di una transazione. La transazione, accettata localmente eseguendo questo algoritmo può non essere globalmente validata; le transazioni vengono aggiunte alla blockchain quando vengono confermate a livello globale.

Algorithm 1 Gestione transazione Bitcoin

```
tx ← ricevi transazione
for input i in tx do
    if (i not in local UTXO or firma non valida) then
        scarta tx e fermati
    end if
end for
if sum(inputs) < sum(outputs) then
    scarta tx e fermati
end if
for input i in tx do
    rimuovi i dall'UTXO set locale
end for
inoltra tx
```

In Bitcoin quindi non esiste un saldo memorizzato, il bilancio di un address infatti è calcolato dai Software Wallet. Il wallet calcola il saldo di un address scansionando la blockchain e sommando gli importi di tutti gli UTXO relativi a quel singolo address. Infatti se un address riceve diversi importi, questi non vengono aggregati ma vengono distribuiti all'interno dell'UTXO set. Il bilancio complessivo di un wallet invece non è altro che la

somma dei bilanci dei singoli address appartenenti a quel wallet. Non è possibile conoscere il saldo complessivo di un utente proprio perché non sanno quali address appartengano ad un determinato wallet, esistono tuttavia delle tecniche e delle euristiche che permettono non solo di conoscere gli address di un wallet ma consentono anche di conoscere le informazioni personali del proprietario.

2.5 Anonimato in Bitcoin

Garantire l'anonimato in Bitcoin vuol dire l'impossibilità di collegare address differenti, ovvero capire che appartengano ad uno stesso utente, e di scoprire l'identità dei proprietari degli address o di un wallet.

Nel protocollo Bitcoin, però, è solo garantito lo pseudo-anonimato, infatti ogni utente utlizza degli pseudonimi, inviando e spendendo bitcoin tramite un numero arbitrario di address che non contengono informazioni sulla sua identità. Per questo Bitcoin non viene definito anonimo ma pseudo-anonimo, ed esistono diversi attacchi mirati a violare l'anonimato di Bitcoin [2, 6].

Uno di questi attacchi si basa sull'analisi delle transazioni salvate sulla blockchain che, come ricordiamo, è pubblicamente leggibile ed immutabile. Lo scopo di questo attacco è di aggregare address differenti in cluster, ciascuno rappresentante un singolo utente, utilizzando le sole informazioni memorizzate nella blockchain. Una volta ottenuti questi cluster, se è poi possibile utilizzare informazioni esterne per identificare anche un solo address del cluster è quindi possibile identificare il proprietario di tutti gli address contenuti in quel cluster.

Per formare questi cluster vengono utilizzate delle euristiche basate sul comportamento degli utenti di Bitcoin, queste regole quindi dipendono dal tempo e non sono assolute.

Le due euristiche più utilizzate sono:

- **inputs multipli**
- **change address**

La prima euristica si può definire nel seguente modo:

"*In una transazione multi-input tutti gli address degli input spesi appartengono allo stesso utente*"

Infatti ogni input di una transazione deve essere firmato, questo significa che per firmare la transazione sono necessarie le chiavi private di ciascun input. Questo implica che il firmatario conosca tutte le chiavi private, e quindi sia il proprietario di tutti gli address di input. Questa prima semplice euristica è stata utilizzata con successo in passato ma non è più valida per tutte le transazioni. Una famosa contromisura proposta è stata CoinJoin [7] nel 2013, l'idea principale è che un insieme di utenti cooperi nel creare collettivamente una singola transazione utilizzando i loro address. In questo caso si potrebbe generare quindi un falso positivo della regola euristica appena enunciata, infatti gli address di input in questo caso non appartengono ad un unico utente.

La seconda euristica si può enunciare come segue:

"*il change address appartiene allo stesso proprietario degli address di input*"

Il problema principale di questa regola è capire quale degli address di output sia il change address, escludendo il caso di "self change", ovvero quando il change address è uno degli address di input. Una regola spesso usata per risolvere tale problema è la seguente, *un address a viene definito change address se e solo se:*

- la transazione non è coinbase;
- è presente più di un output;
- negli address di output non è presente alcun address di input(niente "self change");
- è la prima volta che a appare nella blockchain e non ci sono altri address di output che soddisfano questa condizione.

L'attacco che verrà analizzato in questa tesi, denominato **Dust Attack**, è una forma di attacco all'anonimato e basa la propria strategia sulla prima euristica. In particolare l'attaccante agisce in modo che le vittime generino transazioni multi-input così da poter formare un cluster di address appartenenti ad un unico utente.

Capitolo 3

Il Dust Attack

Tra i molteplici attacchi all'anonimato di Bitcoin, un attacco interessante è il Dust Attack, chiamato anche "Forced reuse address" [8]. Il Dust Attack è una nota tecnica di de-anonimizzazione il cui obiettivo è capire che certi address appartengano al medesimo wallet.

In questa tesi:

- Definiremo il concetto di dust;
- Presenteremo le modalità più comuni con cui viene speso il dust, in particolare analizzeremo se e quanto possa essere efficace un Dust Attack mostrando gli effetti del dust sulla de-anonimizzazione degli address e dei relativi wallet;
- Descriveremo alcuni pattern interessanti che sono stati individuati e che potrebbero essere messi in relazione con il Dust Attack.

Poichè il Dust Attack sfrutta l'utilizzo degli importi dust, è importante quindi spiegare cosa sia il dust e in generale quali possano essere i suoi possibili impieghi.

3.1 Definizione del dust

Il dust è una piccola quantità di criptovaluta presente in un UTXO sotto i limiti minimi di scambio, che chiariremo in seguito. Per definire il valore del dust è necessario fare alcune considerazioni.

Nelle transazioni di Bitcoin, generalmente, l'importo totale di input non viene speso completamente in output ma parte di esso viene inviato ai miners come fee. Formalmente la fee di una transazione, raccolta dai miners come ricompensa per includere la transazione in un blocco, è definita come:

$$fee = \sum_{i=0}^{\#inputs} importo(input_i) - \sum_{j=0}^{\#outputs} importo(output_j)$$

La fee quindi non è altro che la differenza tra l'importo totale di input e l'importo totale di output, e assume valori ≥ 0. In generale la fee viene stabilita autonomamente dal creatore della transazione per incentivare i miners a validare la transazione stessa. Più la fee scelta è proporzionalmente alta, e più i miner guadagneranno nel validare la relativa transazione, rendendo la transazione prioritaria e quindi validata più in fretta.

Tuttavia, per convenzione, esiste una fee minima, denominata *minimum relay fee*, che una transazione si suppone pagare affinché un qualsiasi nodo della rete peer-to-peer inoltri quella transazione agli altri nodi della rete. Questo valore non è prefissato ma varia piuttosto nel tempo in base allo stato della rete e all'andamento del suo valore.

Nella comunità Bitcoin, spesso nuove convenzioni e comportamenti suggeriti sono pubblicati attraverso il client ufficiale, Bitcoin Core [9], poichè rimane ancora oggi il client più diffuso nella rete. Tale client ha introdotto una convenzione, definita "dust limit", che va rispettata per mantenere lo stato di transazione standard, cioè quelle transazioni che i nodi sono suggeriti accettare ed inoltrare ad altri nodi.

Lo scopo è quello di considerare non-standard le transazioni che includono degli output dust proprio perché costerebbe di più per il destinatario spendere l'importo dust rispetto al valore dell'output creato. Infatti un importo dust non può essere speso da solo in una transazione standard, poichè minore della *minimum relay fee*, perciò deve essere necessariamente aggregato ad altri input.

In generale un valore dust consuma in fees un valore maggiore di quello scambiato. Questo di fatto lo rende razionalmente non spendibile, poichè il destinatario dovrebbe spendere più di quanto riceverebbe per ottenerlo. L'output però rimane tecnicamente spendibile e quindi deve essere mantenuto nell'UTXO, appesantendo inutilmente la struttura dati con valori inutili.

Il limite attuale è di 546 satoshi [10]. Quindi tutti gli importi minori di 546 satoshi sono definiti **dust**.

3.2 Possibili utilizzi del dust

Gli usi del dust possono essere molteplici. Per esempio può essere utilizzato per effettuare degli stress test della rete Bitcoin, proprio perché permette di generare tante transazioni con molti output ad un basso costo complessivo. In questo caso il costo è determinato principalmente dalla fee della transazione e non tanto dagli output. Se consideriamo che il valore di 1 satoshi, unità minima di Bitcoin, vale meno di due centesimi di euro al momento (Novembre 2022), è possibile generare una transazione con 1000 output dust, 1 satoshi ciascuno, con meno di venti centesimi (fee esclusa).

In generale, dato l'irrilevante valore economico del dust, viene spesso usato per ottenere effetti laterali, diversi dal semplice scambio di valore. Nel seguito mostriamo tre esempi interessanti di tale uso del dust: la notifica dell'esito di scommesse nel servizio di gambling Satoshi Dice, la scrittura di dati arbitrari tramite lo script OP_RETURN ed il Dust Attack.

3.2.1 Satoshi Dice

Satoshi Dice è un noto "gioco di scommesse basato su blockchain" nato nell'Aprile 2012 [11]. A differenza dei tradizionali software di gioco online, le scommesse con Satoshi Dice possono essere inviate senza accedere ad un sito Web nè eseguire alcun software client. Per giocare, viene effettuata una transazione Bitcoin a uno degli address resi pubblici da Satoshi Dice, caratterizzato da probabilità di vincita e quindi di pagamenti diversi.

Per essere meglio riconoscibili, tutti gli address resi pubblici da Satoshi Dice sono *vanity address*. Un vanity address è un normale address Bitcoin con le stesse funzionalità di qualsiasi altro address ma inizia con una stringa alfanumerica personalizzata contenente una parola o un messaggio comprensibili alle persone. Un esempio di vanity address è *1SochiWwF-FySPjQoi2biVftXn8NRPCSQC*, che contiene la parola Sochi, città in cui si

sono svolte le olimpiadi invernali nel 2014. Tutti gli address di Satoshi Dice presentano infatti il prefisso **1dice**.

Ogni address ha associate probabilità diverse di vincere la scommessa, generalmente minore è la probabilità maggiore è il compenso che si ottiene da una vincita. Per esempio l'address *1dice1e6pdhLzzWQq7yMidf6j8eAg7pkY* offre una probabilità dello 0,0015% di vincere 64 000 volte la puntata originale.

Per determinare se una scommessa è vincente o perdente, Satoshi Dice genera un numero pseudo casuale, il quale viene assegnato al giocatore. Per ottenere tale numero, il servizio utilizza una combinazione dell'hash della transazione della scommessa ed esegue un hash SHA2 a 256 bit prendendo come input l'hash della transazione e un altro parametro sconosciuto al giocatore (un valore segreto giornaliero, pubblicato il giorno dopo per permetterne la verifica). I primi quattro byte dell'output diventano il numero che determina se il giocatore abbia vinto la scommessa.

Il servizio, dopo aver determinato le scommesse vincenti e quelle perdenti, invia una transazione in risposta con il pagamento ai vincitori e restituisce un singolo satoshi ai giocatori perdenti per comunicare loro la perdita. Questo significa che Satoshi Dice utilizza il dust, 1 satoshi, per comunicare la perdita di una scommessa. Nel periodo tra l'Aprile 2012 e Agosto 2017 Satoshi Dice ha generato più di un milione di transazioni in cui invia questi importi dust, a dimostrazione della popolarità raggiunta dal servizio.

3.2.2 Dust in OP_RETURN

Le transazioni in Bitcoin a basso livello sono più complesse [12] di come descritte nella precedente sezione 2.4. Non contengono, infatti, solo valori come address e importo ma includono anche del semplice codice. In particolare ogni transazione comprende anche degli script che descrivono quali siano le condizioni che devono essere verificate affinché il destinatario dei bitcoin possa spenderli.

Bitcoin utilizza un sistema di script Forth-like, basato su stack e non Turing-completo; questa scelta è intenzionale in quanto impedisce possibili cicli infiniti. Gli script in Bitcoin sono [13]:

- Senza stato: non esiste uno stato prima dell'esecuzione di uno script nè lo stato viene salvato dopo l'esecuzione.
- Deterministici: uno script viene eseguito alla stessa maniera in ogni nodo.
- Semplici e compatti: le istruzioni, gli OP_CODE, sono codificate in un singolo byte. In totale ci sono 75 istruzioni funzionanti e 15 disabilitate.

Gli OP_CODE in Bitcoin comprendono diverse categorie:

- aritmetica di base: OP_ADD, OP_SUB;
- controllo di flusso: OP_IF, OP_ELSE;
- logica bit a bit: OP_EQUAL;
- gestione stack: OP_DROP, OP_SWAP;
- hashing: OP_SHA1, OP_SHA256;
- verifica firma o multifirma: OP_CHECKMULTISIG.

Le transazioni Bitcoin non forniscono un campo dove si possono salvare dati arbitrari [14]. Tuttavia, gli utenti hanno ideato diversi modi creativi per codificare dati arbitrari nelle transazioni, in particolare memorizzando valori arbitrari negli output delle transazioni. Questi metodi però modificavano il protocollo rendendo non spendibili gli output così generati. Il problema è che questi output artefatti non sono facilmente distinguibili dagli altri quindi i nodi della rete devono salvarli comunque nei loro UTXO. Poichè questo set, per motivi di efficienza, è solitamente memorizzato nella RAM [5] questa pratica influisce negativamente sul consumo di memoria dei nodi [15].

Per risolvere tale problema a partire dal 2014 è stato reso standard il codice OP_RETURN [16]. Questo OP_CODE era presente fin dalla nascita di Bitcoin e permette di segnare un output della transazione come non valido, quindi successivamente non potrà essere speso. Questo però era considerato uno script non-standard proprio perché la scrittura di dati arbitrari sulla blockchain non è stata considerata fin dall'inizio una buona pratica.

Poichè gli output con questo OP_CODE sono contrassegnati come non spendibili, possono essere rimossi dall'UTXO set. In questo modo OP_RETURN risolve il problema del consumo di memoria spiegato precedentemente. Il limite iniziale per la memorizzazione dei dati con questo codice era pianificato essere di 80 byte, ma inizialmente ne furono concessi 40 (versione 0.9.0). La versione 0.11.0 [17] ha esteso il limite di dati a 80 byte e la versione 0.12.0 [18] fino a un massimo di 83 byte. Inoltre una transazione standard non può contenere più di un output contenenti OP_RETURN, la figura **??** mostra un esempio di un output generato con questo script.

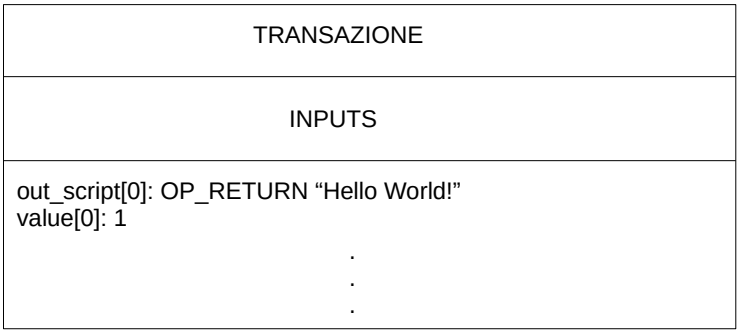


Figura 3.1: Esempio di output con OP_RETURN

Poichè per scrivere dati arbitrari sulla blockchain è comunque necessario generare una transazione con almeno un output risulta quindi molto più conveniente generare output con importi dust per ridurre al minimo i costi della transazione.

Come riportato in [19] esistono diversi protocolli che sfruttano OP_RETURN. Di solito, un protocollo è identificato dai primi byte di metadati allegati all'OP RETURN, ma il numero esatto di byte può variare da protocollo a protocollo.

Questi protocolli possono essere divisi in diverse categorie come ad esempio:

- **Risorse**: protocolli che sfruttano l'immutabilità della blockchain per certificare la proprietà, lo scambio e, infine, il valore dei beni del mondo reale. I metadati nelle transazioni vengono utilizzati per specificare ad

es. il valore del bene, l'importo del bene trasferito, il nuovo proprietario, ecc;

- **Atti notarili**: protocolli per la certificazione della proprietà di un documento. Un utente può pubblicare l'hash di un documento in una transazione, e in questo modo può provarne l'esistenza e l'integrità;
- **Arte digitale**: protocolli per la dichiarazione dei diritti di accesso e di copia di arti digitali, come ad es. foto o musica.

3.3 Il Dust Attack

Come analizzato nella precedente sezione 2.5 esistono delle euristiche che possono essere sfruttate per effettuare determinati attacchi in grado di violare l'anonimato di Bitcoin. L'attacco che verrà approfondito in questa tesi viene denominato **Dust Attack**.

L'euristica utilizzata da questo tipo di attacco è la regola degli input multipli:

"*In una transazione multi-input tutti gli address spesi negli input appartengono allo stesso utente*"

L'obiettivo del Dust Attack è la de-anonimizzazione del proprietario di un wallet, in particolare l'attaccante vuole scoprire quali address appartengano allo stesso wallet, così da ottenere un'importante informazione che può essere utilizzata per effettuare altri attacchi più elaborati e pericolosi.

Il Dust Attack deriva il proprio nome dall'impiego del dust, e si basa sulla regola che il dust, nelle transazioni standard, non può essere speso singolarmente ma deve sempre essere unito ad altri importi. L'attaccante quindi, come mostrato in figura 3.2, invia centinaia, o migliaia, di importi dust ad address diversi confidando che vengano spesi successivamente in una nuova transazione insieme ad altri address. Se ciò avviene l'attaccante riesce a collegare questi address e capire che appartengono, con una certa probabilità, allo stesso utente. Gli address a cui viene inviato il dust sono tutti address già comparsi in precedenza sulla blockchain. Infatti, le transazioni che inviano del dust ad address nuovi molto probabilmente non rappresentano un

Dust Attack, proprio perché risulterebbe alquanto inusuale de-anonimizzare un address mai visto fino a quel momento. In generale un nuovo indirizzo è creato come change address dal creatore della transazione o come nuovo indirizzo di pagamento dal destinatario del pagamento. Nel caso di Dust Attack, molto probabilmente il proprietario di un address nuovo è lo stesso utente che genera quella determinata transazione poichè è l'unico a conoscenza di quel determinato address, quindi, nel nostro caso, l'attaccante stesso.

Il Dust Attack ad un primo sguardo potrebbe sembrare particolarmente costoso, proprio perché invia tanti importi ad address differenti, ma se analizziamo il valore in euro di un singolo satoshi, si può osservare che il costo è relativo; soprattutto se l'obiettivo finale è tentare un furto dei bitcoin delle vittime. Infatti abbiamo che:

$$1 \text{ satoshi} = 0.00016 \text{ €} \text{ (in data } 16/11/2022)$$

Questo significa che 1 singolo satoshi vale meno di 1 centesimo, quindi se un attaccante genera una transazione con 1000 output dust e manda a ciascun address 1 satoshi, in totale spende 16 centesimi, escludendo la fee.

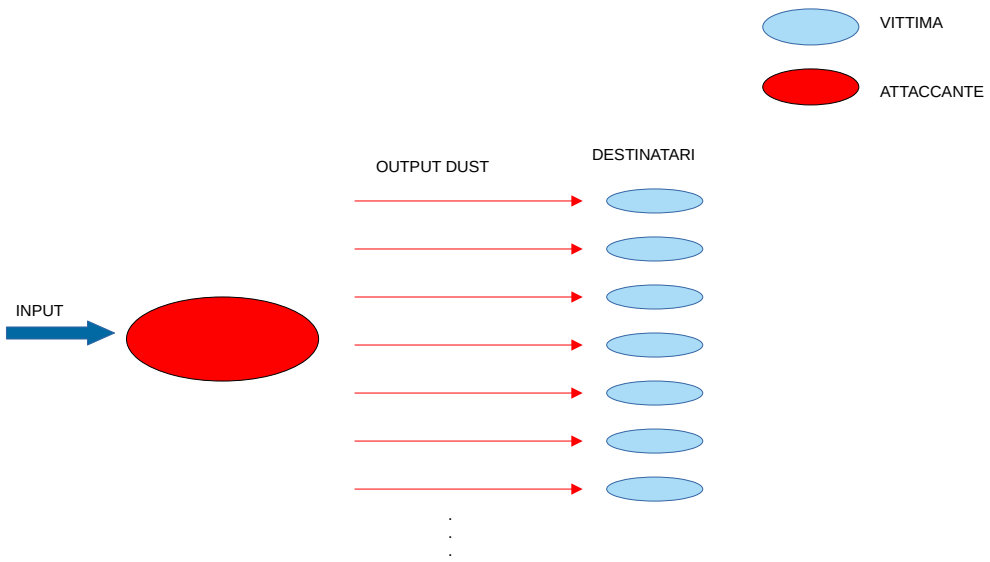


Figura 3.2: Schema Dust Attack

Possono esserci due possibili esiti del Dust Attack:

1. attacco di successo;

2. attacco fallito.

Nel primo caso, mostrato in figura 3.3, la vittima genera una transazione in cui spende l'importo dust ricevuto insieme ad almeno un altro dei suoi address.

Questi address quindi sono collegati tra loro dalla euristica sui multi-input, ed è possibile capire che appartengano ad uno stesso utente. Se l'attaccante successivamente scopre le informazioni personali, come nome o email, del proprietario di uno di questi address automaticamente scopre che tutti gli altri address hanno lo stesso proprietario. Inoltre è possibile tracciare l'attività di un utente e non solo di un singolo address.

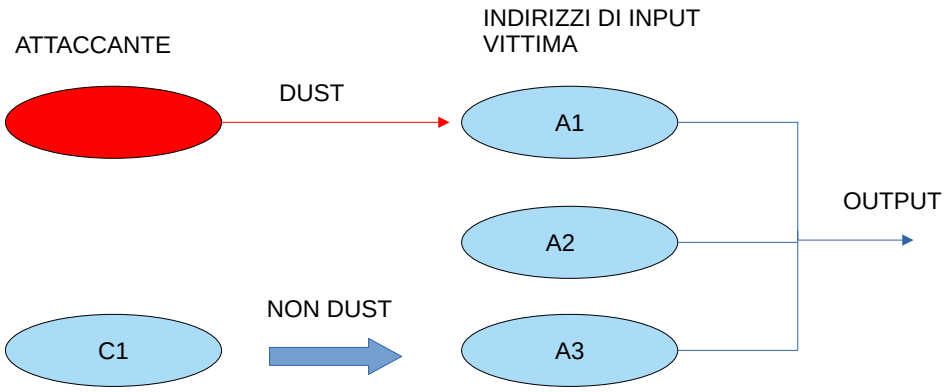
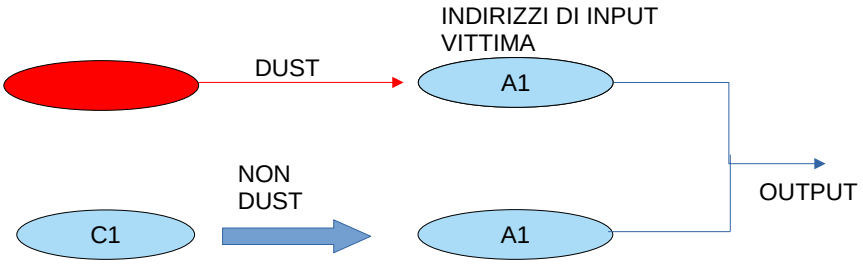


Figura 3.3: Schema di Dust Attack di Successo: l'address del dust viene speso con altri address

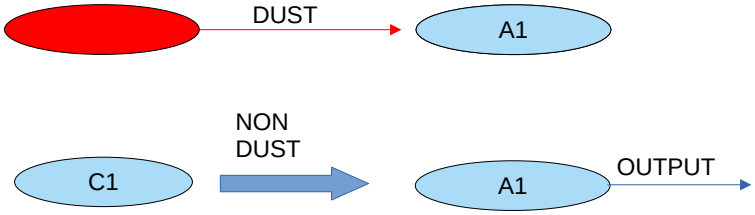
Esistono invece due possibili motivi per cui un Dust Attack può fallire. In figura 3.4 vengono schematizzati questi due possibili esiti.

Nel primo schema la vittima spende il dust ricevuto, ma utilizza una transazione i cui input si riferiscono ad UXTO relativi all'address in cui è stato depositato il dust. In questo caso l'attaccante non ricava alcuna informazione perché non riesce nell'intento di provocare l'unione di address diversi di un utente nella medesima transazione. Nel secondo schema invece la vittima non spende il dust, troncando sul nascere l'attacco.

Il Dust Attack, in generale, risulta più efficace soprattutto se il dust è diretto verso gli address che hanno un bilancio complessivo pari a zero proprio perché obbliga la vittima a spendere la cifra ottenuta con altri suoi address diversi.



(a)



(b)

Figura 3.4: Schema di Dust Attack Fallito. La vittima spende il dust con un solo address(a), la vittima non spende il dust(b)

è importante notare che lo scopo del Dust Attack non è quello di rubare fondi di altri utenti nè quello di scoprire informazioni personali, per esempio nome e cognome, della vittima. Permette, invece, di rompere lo pseudo-anonimato degli address, ovvero l'appartenenza di address differenti ad uno

stesso utente. Questa informazione può essere usata in seguito per effettuare attacchi elaborati e più pericolosi.

In generale il Dust Attack non necessariamente è legato a phishing o estorsioni ma potrebbe essere usato dalle autorità per tenere traccia degli utenti ed eventualmente rilevare attività illegali.

Infatti una volta ottenuto un cluster di address è possibile tracciare l'attività di un singolo utente e non più di un singolo address. Le autorità potrebbero notare che certi utenti interagiscono con address legati a mercati neri, e quindi indagare ulteriormente per scoprire l'identità di queste persone.

Uno dei punti fondamentali è legare informazioni personali, come e-mail, nome ed altro, ad un address Bitcoin. In molti casi sono gli utenti stessi che pubblicano sui forum i loro address, in altre situazioni invece è possibile sfruttare gli *exchange* come Coinbase.

Gli exchange sono servizi che permettono lo scambio tra criptovalute e valute tradizionali basandosi sul valore di mercato della criptovaluta. In exchange reputati affidabili, come Coinbase, è necessario creare un account fornendo informazioni personali come nome, cognome, email ed altro e, una volta registrati, viene creato un wallet associato a quel particolare account. Anche se, poichè gli exchange sono vittime appetibili di attacchi hacker, molti utenti trasferiscono i loro bitcoin su address appartenenti a software wallet. Una volta che un utente effettua un deposito dal wallet, vittima di Dust Attack, ad un account exchange l'attaccante può collegare gli address al proprietario. Ed una volta ottenuta l'identità del proprietario, l'attaccante può eseguire elaborati attacchi di phishing oppure può estorcere denaro minacciandolo di rivelare a tutti l'informazione ottenuta.

Per fortuna, il Dust Attack non è particolarmente difficile da contrastare, nel paragrafo successivo verranno mostrati due metodi per difendersi da questo tipo di attacco.

3.3.1 Contromisure al Dust Attack

Due metodi che un utente che ha ricevuto un importo dust può utilizzare per contrastare il Dust Attack sono:

1. non spendere l'importo dust ricevuto;

2. utlizzare servizi di "dust collecting".

La prima soluzione, semplice ed efficace, permette di troncare l'attacco sul nascere. Infatti se il dust non viene speso l'attaccante non potrà mai collegare address diversi dello stesso utente. Diversi software wallet implementano questo metodo; per esempio Samurai Wallet [1], nel 2018, consigliò ai suoi utenti, possibili vittime di Dust Attack, di contrassegnare come "do not spend" ogni output dust ricevuto.

La seconda soluzione riguarda i servizi di "dust collecting", come, ad esempio, Dust-B-Gone [20]. Dust-B-Gone, ideato da Peter Todd già nel 2012, era un servizio che permetteva di disfarsi del dust, non lasciando il dust ricevuto come UTXO, ma trasformandolo in fee per i miner. Il programma generava un'unica transazione alla quale potevano cooperare più utenti. Ogni utente inseriva tra gli input della transazione l'importo dust ricevuto così da potersene liberare. Allo stesso tempo questo servizio proteggeva gli utenti da una possibile de-anonimizzazione, poichè address diversi di una transazione potevano appartenere ad utenti diversi. Un attaccante quindi non avrebbe mai potuto collegare questi address, o comunque ne avrebbe ricavato una falsa informazione. Nonostante la solida idea di base, notiamo che questo servizio è stato chiuso nel 2016 per vari motivi, tra cui il suo scarso utilizzo [2] .

[1] fonte:`https://twitter.com/samouraiwallet/status/1055345822076936192?lang=en`

[2] fonte: `https://twitter.com/SamouraiWallet/status/1293659938422652935`

Capitolo 4

Implementazione

In questo capitolo verranno mostrati il formato dei dati e alcuni degli algoritmi realizzati per le analisi. Il codice è stato scritto in python, versione 3.9.12, distribuzione Anaconda.

La libreria per la realizzazione dei grafici è matplotlib versione 3.5.1 e la la libreria per le statistiche è Pandas versione 1.5.0

4.1 Strumenti utilizzati

Gli input e gli output delle transazioni sono stati salvati in appositi file csv con una struttura a tabella, quindi è stato necessario utilizzare uno strumento che permettesse di manipolare tabelle di grandi dimensioni; la scelta è ricaduta sulla libreria **Pandas**.

Pandas [21] è una libreria, scritta per il linguaggio python, realizzata per la manipolazione e l'analisi dei dati. In particolare il software permette di salvare i dati in Dataframe, una struttura dati bidimensionale, simile a una tabella contenente righe e colonne. Una volta ottenuto un Dataframe è possibile eseguire operazioni in stile SQL, come per esempio GROUPBY e JOIN.

Il codice 4.1 mostra un semplice esempio di uso della libreria Pandas:

```
1 import pandas as pd
2
3 #viene caricato un file .csv.xz in un dataframe
4 df = pd.read_csv("animali_zoo.csv.xz", compression='xz')
```

```
5  # Il dataframe ha la seguente forma:
6  #   Animale          Nome    Peso(Kg)
7  #   Panda            Icaro     1100
8  #   Orso_Bruno       Dedalo    500
9  #   Panda            Apollo    1055
10 #   Panda            Zeus      1165
11 #   Orso_Polare      Cupido    450
12 #   Orso_Bruno       Ermes     600
13
14 df.groupby("Animale").count()
15 #Risultato:
16 #  Panda: 3
17 #  Orso_Bruno: 2
18 #  Orso_Polare: 1
19
20 df[df.Animale == 'Panda']
21 #Risultato:
22 # Animale   Nome     Peso(Kg)
23 # Panda     Icaro    1100
24 # Panda     Zeus     1165
25 # Panda     Apollo   1055
26
27 #la ~(tilde) corrisponde ad una negazione
28 df[~df.Animale == 'Pandas']
29 #Risultato:
30 #   Animale          Nome     Peso(Kg)
31 #   Orso_Bruno       Dedalo    500
32 #   Orso_Bruno       Ermes     600
33 #   Orso_Polare      Cupido    450
```

Codice 4.1: Semplice uso di Pandas

4.2 Rappresentazione dei dati

Il dataset su cui sono state condotte le analisi è stato ottenuto tramite un filtraggio della blockchain originale, la dimensione del file testuale contenente il dataset filtrato si aggira intorno ai 40 GB.

Ogni riga del file rappresenta una singola transazione che presenta il seguente formato: **infos:inputs:outputs**. Il campo **infos** contiene i seguenti

dati:

- **timestamp**: valore intero, rappresenta il tempo in cui è inserita la transazione;
- **blockId**: valore intero, l'ID del blocco dove è salvata la transazione;
- **TxId**: valore intero, l'ID della transazione, garantito essere univoco in quanto è un contatore, che vale 1 per la prima transazione e viene incrementato a seguire per le altre;
- **fee**: valore intero, indica il valore della fee pagata in quella transazione;
- **approxSize**: valore intero, indica il peso, in byte, approssimato della transazione.

Il campo **inputs** contiente gli input della transazione separati da '**;**'; possono esserci zero o più input.

Ogni input presenta la seguente struttura:

- **addrId**: valore intero, rappresenta l'address che sta pagando. Il valore numerico è stato assegnato in modo incrementale. Per esempio l'addrId 1 è il primo address che compare in assoluto sulla blockchan, mentre l'addrId 2 è il primo che compare subito dopo l'addrId 1;
- **amount**: importo, in satoshi, speso dal pagante. Un satoshi equivale a 10^{-8} bitcoin;
- **prevTxId**: l'ID della transazione da cui l'address pagante ha ricevuto l'importo che sta spendendo;
- **offset**: intero relativo alla posizione dell'address che sta pagando all'interno degli output della transazione specificata nel punto precedente.

Il campo **outputs** contiene gli output della transazione separati da '**;**'; in questo caso deve essere presente almeno un output.

La forma dei singoli output è la seguente:

- **addrId**: intero con funzione analoga a quello degli input ma relativo a chi riceve l'importo;

- **amount**: importo, in satoshi, che il ricevente incassa;
- **script**: valore intero, indica il tipo di script utilizzato per quell'importo; sono presenti 5 possibili valori:

  UNKNOWN=0; P2PK=1; P2PKH=2; P2SH=3; RETURN=4;

  EMPTY=5;

La figura 4.1 riassume il formato con cui sono memorizzate le transazioni nel file testuale:

infos:inputs:outputs
infos := timestamp,blockId,TxId,fee,approxSize

inputs := $input_1; input_2; ...input_i; ...; input_n$
$input_i$:= addrId,amount,prevTxId,offset

outputs := $output_1; output_2; ...output_i; ...; output_n$
$output_i$:= addrId,amount,script

Figura 4.1: Schema formato transazioni

In seguito un esempio fittizio di una transazione:

1285666089,2,12,20,258:5,100,11,0:8,30,2;14,50,2

La transazione di esempio con TxId 12, timestamp 1285666089(martedì 28 Settembre 2010) presenta un solo input e due output: un solo addrId pagante, 5, sta inviando bitcoin a due address distinti: 8 e 14.

L'address pagante ha ricevuto i bitcoin che sta spendendo (100 satoshi) nella transazione identificata dall' ID 11; in essa era il primo output(offset zero).

In questo caso, l'addrId 8 ha ricevuto 30 satoshi, mentre l'addrId 14 ne ha ricevuti 50, entrambi gli output presentano script 2(script P2PKH). La fee complessiva, differenza tra l'importo totale di input e l'importo totale di output, è pari a 20, la differenza tra 100 e 80(50 + 30).

Gli input e gli output di ogni transazione, che presenta almeno un input o un output dust, sono stati successivamente salvati in due appositi file csv.

timestamp	blockId	TxId	addrId	amount	prevTxId	offset

Struttura 4.1: Input

timestamp	blockId	TxId	addrId	amount	script

Struttura 4.2: Output

I file presentano la seguente forma:

Grazie a questa struttura è stato possibile classificare velocemente gli output in due gruppi distinti, tramite un'operazione stile SQL JOIN. Nel primo gruppo sono salvati gli output spesi, nel secondo quelli non spesi.

Le due categorie presentano la medesima struttura a tabella con i seguenti campi:

- **timestamp**: tempo in cui è stata convalidata la transazione con identificativo TxId;
- **blockId**: il blocco contenente la transazione con identificativo TxId;
- **TxId**: l'identficativo della transazione in cui è stato generato l'output;
- **script**: script con cui è stato generato l'output;
- **addrId**: l'address che ha ricevuto l'importo;
- **amount**: l'importo in satoshi;
- **spentTxId**: l'identificativo della transazione in cui è stato speso l'output, assume valore -1 se non è stato speso;
- **spentBlock**: l'identificativo del blocco in cui è salvata la transazione con identificativo spentTxId, assume valore -1 se l'importo non è stato speso;
- **spentTimestamp**: l'istante in cui è stata validata la transazione con identificativo spentTxId, anche in questo caso assume valore -1 se non è stato speso l'output.

I primi tre campi sono relativi alla transazione in cui viene generato l'output mentre gli ultimi tre sono relativi alla transazione dove questo output appare come input, ovvero se e quando viene speso.

Un esempio di importo non speso è il seguente:

1285666089	82560	121385	2	118890	99	-1	-1	-1

L'address 118890 ha ricevuto 99 satoshi nella transazione identificata dall' ID 121385 con timestamp 1285666089. Gli ultimi tre campi contengono il valore -1 per indicare che l'ammontare ricevuto non è stato speso.

L'esempio che segue mostra un caso di importo speso :

1285666864	82561	121404	118890	99	2	124069	83232	1286048669

In questo caso l'address 118890 ha ricevuto 99 satoshi nella transazione con ID 121404 e timestamp 1285666864, successivamente li ha spesi nella transazione con ID 124069 e timestamp 1286048669.

4.3 Algoritmi

Prima di poter analizzare i dati è stato necessario un filtraggio del dataset iniziale, proprio per ottenere solo quelle transazioni di interesse. In particolare sono state considerate solo le transazioni contenenti almeno un input dust o almeno un output dust.

L'algoritmo realizzato per il filtraggio di queste transazioni è molto semplice ed è mostrato in 4.2. Si tratta di leggere riga per riga il file testuale, prelevare i campi inputs e outputs tramite una tokenizzazione della stringa, guidata dal separatore ':', e controllare che sia presente un importo dust tramite la funzione **isDust()**. Questa funzione semplicemente scorre gli input(e gli output) e controlla che esista un importo nell'intervallo [1, 545]. Se presente la funzione restituisce $True$ e la transazione viene scritta in un apposito file.

Il dataset iniziale(nel codice "source") ha un peso di 40 GB, il dataset filtrato(nel codice "destination") ha un peso di 497 MB. La dimensione del

dataset di output indica come gli importi dust siano una minoranza rispetto al totale degli importi.

```
for line in source:
    tx = line.split(':')
    inputs = tx[1]
    outputs = tx[2]
    if(isDust(inputs) or isDust(outputs)):
        destination.write(line)
```

Codice 4.2: Filtraggio transazioni dust

Una volta ottenutte tutti e soli i dati di interesse è stato necessario eliminare le transazioni generate da Satoshi Dice. Il motivo di questa scelta è stato perché l'obiettivo era di esaminare il comportamento del dust proveniente da address non noti, in modo da restringere il campo di ricerca dei creatori di dust, così da poter trovare possibili pattern di Dust Attack.

Il codice 4.3 è molto simile a quello mostrato per il filtraggio delle transazioni dust. Viene letto il file riga per riga, viene prelevato il campo inputs e si controlla che non sia presente un identificativo associato ad uno degli address resi pubblici da Satoshi Dice.

è stato necessario quindi convertire gli address pubblici di Satoshi Dice nei corrispettivi identificativi. Ciò è stato abbastanza semplice perché avevamo a disposizione un file che memorizzava la corrispondenza **address-addrId**. Una volta ottenuti, questi identificativi sono stati salvati in una lista(SD_ids). La funzione **is_SD()**, applicata solo al campo inputs, restituisce *True* se è presente un identificativo appartenente alla lista "SD_ids". La transazione viene scritta in un ulteriore file quando **is_SD()** restituisce *False*, ovvero quando la transazione non è stata generata da Satoshi Dice.

```
for line in source:
    tx = line.split(':')
    inputs = tx[1]
    if((not is_SD(inputs, SD_ids))):
        destination.write(line)
```

Codice 4.3: Filtraggio transazioni generate da Satoshi Dice

Dopo aver applicato questi filtraggi, gli input e gli output delle varie transazioni sono stati salvati in appositi file csv, con la struttura spiegata preceden-

temente 4.1 4.2. Una volta eliminati gli output dust con script OP_RETURN, è stato possibile ottenere quali output dust siano stati spesi e quali invece no. Questa operazione è stata realizzata grazie ad una funzione della libreria Pandas, **merge()**, che permette di realizzare operazioni di JOIN, in stile SQL.

Dopo aver ottenuto gli importi dust spesi è stato possibile classificare le transazioni in tre categorie: Successo, Fallimento e Speciali. Le prime sono le transazioni dove il dust viene speso insieme ad un altro address, ma la fee è diversa dall'importo totale di input, la seconda categoria invece comprende quelle transazioni dove è presente un unico address di input e infine la categoria "Speciali" contiene le transazioni dove l'importo totale di input viene trasformato in fee, in quest'ultima categoria rientrano anche le transazioni di "dust-collectng".

Gli identificativi di quest'ultima categoria sono stati salvati in un file testuale così da poter essere riutilizzati in seguito per altre analisi. Per riconoscere una transazione speciale semplicemente è stato controllato il valore della fee e la somma degli importi di input, se risultano uguali allora il suo identificativo viene salvato in questo file, altrimenti no.

Una volta salvate la transazioni speciali, nel codice 4.4 viene calcolato quante transazioni delle prime due categorie siano presenti negli anni. Una volta eliminati dal dataframe "inputs" tutti gli input relativi alle transazioni speciali vengono calcolate, per ogni anno, quante transazioni siano presenti nelle categorie "Successo" e "Fallimento". Il dataframe "inputs" contiene tutti e soli gli input relativi alle transazioni in cui viene speso il dust.

Viene creato, dentro il ciclo for, la variabile "inp" che contiene tutti gli input di quel determinato anno; il timestamp è stato opportunatamente convertito in anno. Una volta presi gli input di quel determinato anno è stata eseguita una **groupby** sugli identificativi delle transazioni per poi eseguire la funzione **agg()** che, in questo caso, equivale ad una SELECT COUNT() DISTINCT di SQL.

Una volta eseguita questa operazione è stato semplice capire quali transazioni appartenessero a quale categoria. Infatti se il numero di addrId è ≥ 2 vuol dire che sono presenti almeno due address diversi e quindi è una transazione di successo, mentre se il valore è $= 1$ vuol dire che rientra nel-

la categoria “Fallimento”. La variabile “categories” è un dizionario dove la chiave è l’anno e il valore un array di tre interi: la prima posizione indica quante transazioni sono della prima categoria, la seconda posizione quante della seconda e la terza quante della terza categoria.

```
tx_special = collect_tx("TxId_special.txt")
inputs = inputs[~inputs.TxId.isin(tx_special)]

for year in range(2010, 2018):
    inp = inputs[inputs.timestamp == year]
    inp = inp.groupby("TxId").agg({'addrId':'nunique'})

    categories[year][0] += len(inp[addrId >= 2])
    categories[year][1] += len(inp[addrId == 1])
```

Codice 4.4: Classificazione transazioni

L’ultima analisi effettuata riguarda la presenza di address nuovi all’interno delle transazioni che generano almeno un dust speso con altri address, ovvero speso in una transazione della categoria “Successo”. La definizione “address nuovo” indica un address che non era mai comparso nella blockchain fino a quel momento. Come spiegato precedentemente il proprietario di un address nuovo probabilmente è lo stesso utente che genera quella determinata transazione poichè è l’unico a conoscenza di quel determinato address. Quindi de-anonimizzare un proprio address risulta alquanto improbabile

Per capire se un address sia nuovo è stata sfruttata la proprietà con i quali vengono assegnati gli identificativi agli address. Il valore numerico è stato assegnato in modo incrementale. Per esempio l’addrId 1 è il primo address che compare in assoluto sulla blockchan, mentre l’addrId 2 è il primo che compare subito dopo l’addrId 1, quindi per sapere quali address sono nuovi nella transazione X basta sapere l’id dell’ultimo address usato (cioè quello con id massimo) fino alla transazione precedente (X-1).

Per esempio se l’addrId massimo prima della transazione X è 5 tutti gli addrId, presenti nella transazione X, che hanno un valore > 5 sono nuovi. Nell’algoritmo 4.5 si scorrono, tramite ciclo for, tutte le transazioni che generano almeno un output dust di successo. Si filtrano, dal dataframe ”outputs_dust” tutto il dust generato in quella transazione. Prima vengono contati quanti address, senza ripetizioni, totali ci sono, successivamente si contano

quanti addrId sono nuovi, sfruttando la proprietà degli identificativi appena descritta. Se la transazione non presenta address nuovi, ovvero la variabile "new_addr" = 0, il suo TxId viene aggiunto ad una lista; questa lista verrà poi salvata in un opportuno file testuale. Per le transazioni che presentano address nuovi semplicemente la percentuale media di address non-nuovi.

```
1  for tx in tx_dust:
2      maxId = tx_MaxId[tx_maxId.TxId == tx]['prevMaxAddrId']
3      out = outputs_dust[outputs_dust.TxId == tx]
4      tot_addr = len(out['addrId'].to_list())
5      new_addr = len(out[out.addrId > maxId]['addrId'].to_list
           ())
6      old_addr = tot_addr - new_addr
7
8      if(not new_addr):
9          tx_notNew_addr += 1
10         tx_Id_notNew.append(tx)
11     else:
12         tx_newAddr += 1
13         perc_sum += (old_addr/tot_addr)*100
14
15 perc_mean = perc_sum / tot_tx
```

Codice 4.5: Analisi address nuovi

Una volta ottenute le transazioni che presentano address non-nuovi, ovvero address che erano già stati utilizzati in passato, è stato possibile trovare alcuni pattern interessanti che sono stati impiegati per generare grandi quantità di output dust.

Capitolo 5

Risultati Sperimentali

I dati analizzati comprendono tutte le transazioni Bitcoin dal **3 Gennaio 2009** al **10 Agosto 2017**.

Come spiegato precedentemente il primo obiettivo è quello di presentare statistiche generali sull'uso del dust, mostrare gli effetti del dust sulla de-anonimizzazione di address e, infine, descrivere pattern che potrebbero rappresentare dei Dust Attack.

5.1 Filtraggio transazioni

Il primo compito è stato il filtraggio di tutte le transazioni dust, ovvero tutte quelle transazioni che comprendono tra gli input e/o tra gli output un importo minore di 546 satoshi, valore di riferimento del dust.

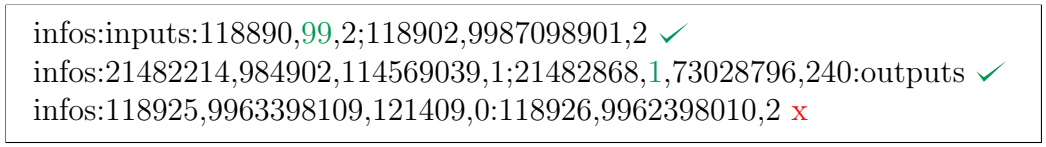


Figura 5.1: Esempi di transazioni accettate o rifiutate

La figura 5.1 mostra due esempi di transazioni dust e uno di transazione non-dust, le prime due transazioni vengono considerate nelle analisi successive perché contengono importi dust, mentre la terza viene scartata.

La prima infatti contiene, tra gli output, un importo di 99 satoshi, la seconda invece contiene un importo di 1 satoshi tra gli input. L'ultima transazione invece è stata scartata proprio perché tutti gli importi sono maggiori o uguali a 546 satoshi. Le transazioni non-dust di questo tipo sono state ignorate poichè le analisi vertono sulla generazione e sul consumo del dust e sulla de-anonimizzazione causata da questi importi.

Le transazioni totali sono 245 410 083 mentre le transazioni che generano o consumano dust sono 2 114 335. Questo significa che il dust è usato solo nello 0.8% delle transazioni totali. Inoltre, come confermato da [22], 1 705 560 creano dust mentre solo 429 544 lo consumano. Siccome la somma delle transazioni che consumano dust e delle transazioni che generano il dust è superiore al totale delle transazioni possiamo dedurre che ci siano transazioni in cui il dust è presente sia negli input che negli output.

Il passaggio successivo è stato il filtraggio delle transazioni generate da Satoshi Dice poichè non riteniamo siano parte di Dust Attack. Satoshi Dice, come spiegato in precedenza, è un noto servizio di gambling nato nell'Aprile 2012. Questo servizio utilizza il dust per comunicare ai giocatori l'esito perdente della loro scommessa. Data la notorietà del servizio, risulta molto poco probabile che l'intento sia quello di un Dust Attack, inoltre le analisi vogliono mostrare il comportamento degli utenti nei confronti del dust proveniente da address sconosciuti. Non solo gli address di Satoshi Dice sono noti proprio perché il servizio stesso li ha resi pubblici, ma l'indirizzo di ricezione del dust dello scommettitore è lo stesso con cui la scommessa viene piazzata, e quindi non scelto discrezionalmente dal servizio.

Le transazioni generate da Satoshi Dice sono 1 465 295, ovvero il 69% delle transazioni dust. Questo risultato, oltre a dimostrare la popolarità del servizio, permette di concentrare le future analisi sul rimanente 31%(649 040) delle transazioni.

5.2 Analisi del dust

Una volta ottenute tutte e sole le transazioni di interesse sono stati realizzati gli istogrammi mostrati in figura 5.2. L'istogramma, mostrato nella parte sinistra della figura 5.2, mostra la distribuzione del numero di input dust,

quello a destra invece la distribuzione del numero di output dust. In entrambi i grafici non sono state contate le transazioni con 0 importi dust. La prima colonna quindi rappresenta, in entrambi gli istogrammi, l'intervallo [1, 50].

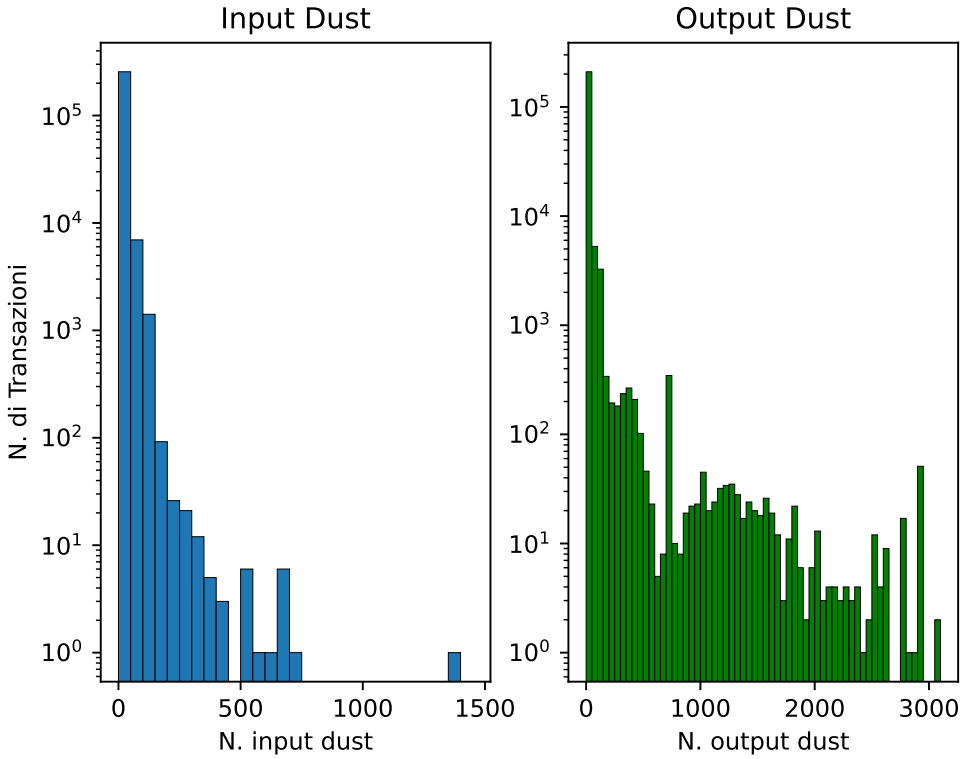


Figura 5.2: Distribuzione numero input dust(sinistra) e output dust(destra). Intervalli di ampiezza pari a 50 con primo intervallo [1, 50]. Si noti la scala logaritmica sull'asse delle ordinate

Entrambi gli istogrammi mostrano come siano molto frequenti le transazioni presenti nel primo intervallo di ampiezza [1, 50]. Nel primo grafico notiamo anche che le transazioni con un elevato numero di input dust risultino poco frequenti. Al contrario del primo grafico, il secondo mostra come siano presenti tante transazioni che generano un elevato numero di output dust. Dall'analisi di questi due grafici inoltre possiamo già intuire che diversi output dust generati non vengano successivamente spesi.

Manualmente ispezionando alcune delle numerose transazioni con un numero basso di input o output, abbiamo intuito di poter introdurre un livello di filtraggio ulteriore. In tutte le analisi successive viene infatti considerato

solo il dust generato con script diverso da OP_RETURN. Questa scelta è motivata dal fatto che gli output con questo script non possono essere spesi, questa tipologia di dust quindi non può avere effetti sulla de-anonimizzazione. E quindi, chi lo sta generando sicuramente non sta attuando un Dust Attack. In totale sono presenti 18 357 output dust con questo script, di cui il 97.5% ha come importo 1 satoshi.

Il grafico 5.3 mostra quindi la distribuzione nel tempo degli output dust rimanenti (cioè quelli potenzialmente spendibili).

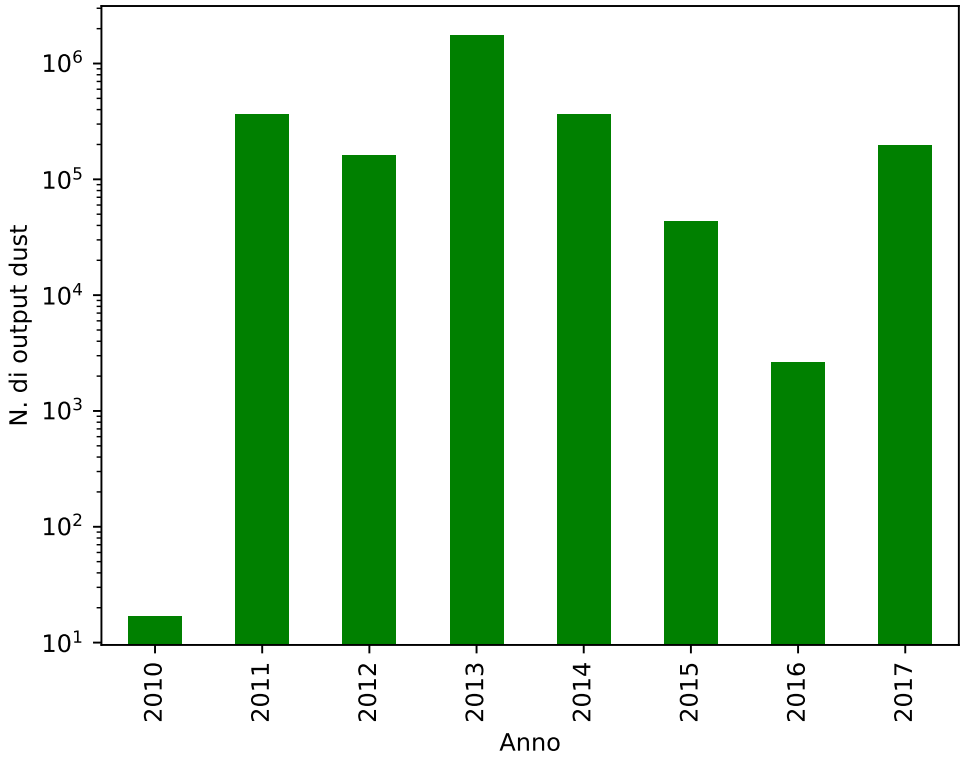


Figura 5.3: Distribuzione delle date di creazione di output dust (filtrati) nel tempo

Dal grafico è possibile notare che a partire dal 2010 sono comparsi i primi output dust, anche se in quantità molto ridotta. Possiamo osservare un rapido aumento tra il 2010 e il 2011. Solo nel mese di luglio dell'anno 2011 infatti sono stati generati più di 27 000 output dust.

Il picco della generazione di dust è presente nel 2013, dove sono stati trovati pattern interessanti di transazioni legate "a catena". Nel paragrafo successivo verranno approfonditi questi schemi. Dopo il picco del 2013 osserviamo una costante diminuzione negli anni fino al 2017 dove si assiste ad un nuovo incremento. Bisogna però specificare che circa il 95% del dust generato nel 2017 proviene da due address: $1Enjoy1C4bYBr3tN4sMKxvvJDqG8NkdR4Z$ e $1SochiWwFFySPjQoi2biVftXn8NRPCSQC$.

Questi due noti address sono comparsi per la prima volta nel 2014, in concomitanza con le olimpiadi di Sochi in Russia, generando transazioni con circa 750 output di 1 satoshi ciascuno. È importante notare come, nonostante il caos causato in vari forum Bitcoin, la tempesta di spam del 2014 "Enjoy Sochi" abbia lasciato un'impronta relativamente piccola sulla blockchain. Solo 65 transazioni(48 750 output dust) sono state confermate in quell'anno.

L'aspetto singolare di questo fenomeno è che abbiamo visto dei suoi echi tornare negli anni successivi. Nel 2015 infatti sono state confermate 23 transazioni(17 250 output dust) mentre nel 2017 sono presenti 255 transazioni(189 495 output dust) generate da questi vanity address. Sebbene sia possibile che siano state eseguite dalla stessa entità, che spendeva importi da quegli address anche nel 2018, è anche possibile che queste transazioni fossero state generate nel 2014 e confermate solo nel 2015 e nel 2017. In questi tre anni il fenomeno "Enjoy Sochi" ha generato un totale di 343 transazioni per un totale di output dust intorno ai 255 000.

In generale sono stati generati 2 893 877 output dust filtrati (ossia non imputabili a Satoshi Dice e con script diverso da OP_RETURN), e circa il 48.5% è stato speso. Quindi la quantità di dust non speso (51.5%) è simile a quello speso.

La tabella 5.1 mostra il rapporto in percentuale negli anni del numero di output dust generati e il numero di output dust spesi e non-spesi, non necessariamente il dust è stato speso nello stesso anno in cui è stato generato ma può essere stato speso anche in anni successivi. Il 2009 non è stato considerato perché non è stato generato alcun output dust in quell'anno. La superiorità del dust non-speso rispetto al dust speso è presente soprattutto negli anni 2011 e 2017, anche se parte del dust del 2017 potrebbe essere

stato speso successivamente al 10 Agosto 2017, data dell'ultima transazione del dataset. Il 2010, nonostante il dust non-speso sia l'83%, non è molto rilevante poichè sono stati generati solo 14 output dust in totale.

categoria/anno	2010	2011	2012	2013	2014	2015	2016	2017
speso	17%	0,2%	54%	44%	76%	94,6%	95,6%	10%
non-speso	83%	99,8%	46%	56%	24%	5,4%	4,4%	90%

Tabella 5.1: Dust Speso e Non-Speso negli anni

5.3 Classificazione del dust

Come descritto in sezione 4.3 gli output dust sono stati suddivisi in quattro categorie:

1. **Successo**: dust speso in transazioni che presentano almeno due address diversi di input, in queste transazioni la fee è diversa dall'importo totale di input;

2. **Fallimento**: dust speso in transazioni con un solo address di input, anche se ripetuto più volte;

3. **Speciale**: dust speso in possibili transazioni di "dust collecting", transazioni la cui fee è uguale all'importo totale di input;

4. **Non speso**.

I termini "Successo" e "Fallimento" si riferiscono alla possibilità di collegare gli address di input una volta che il dust è stato speso. Nonostante il "dust-collecting" rappresenti un caso di de-anonimizzazione fallita, poichè gli address di input potrebbero appartenere ad utenti diversi, questa situazione viene separata dalla categoria "Fallimento" per mostrare se e quanto questa tecnica sia stata utilizzata.

Il grafico 5.4 mostra la distribuzione delle categorie nel corso degli anni.

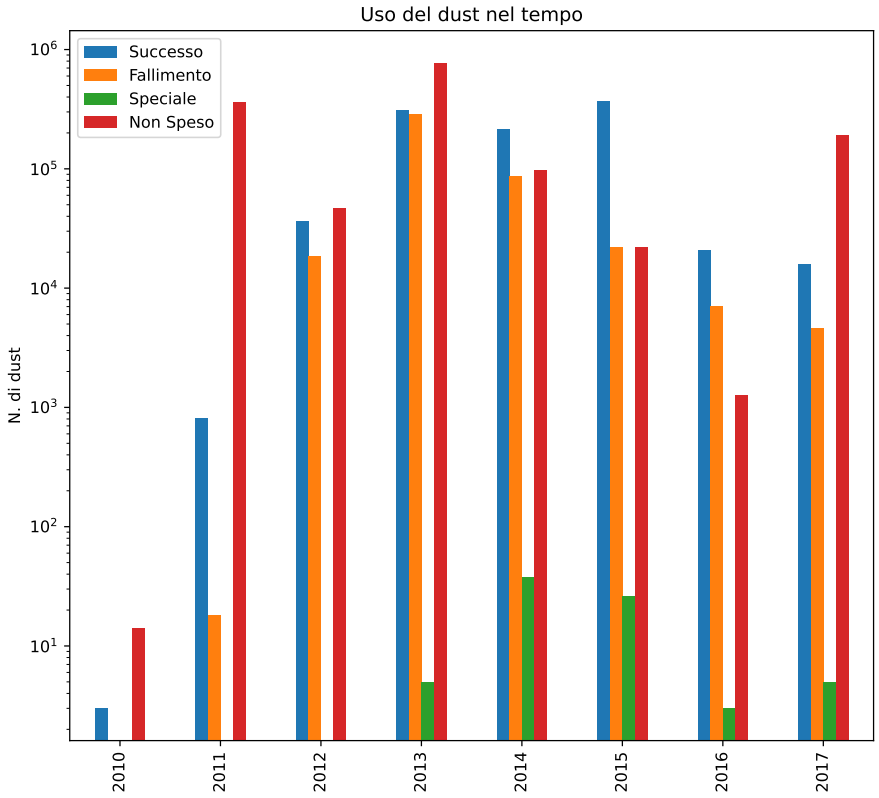


Figura 5.4: Uso finale (anche oltre l'anno di creazione considerato) degli output dust generati in un determinato anno

Non solo possiamo constatare graficamente quanto già detto in precedenza sulla differenza tra dust speso e non speso, ma possiamo anche fare due importanti riflessioni.

La prima riflessione riguarda l'uso del servizio di "dust collecting". Infatti risulta evidente quanto poco esso sia stato utilizzato. È importante notare che il dust della categoria "Speciale" non necessariamente è legato a Dust-B-Gone, questa categoria infatti è presente anche nel 2017 nonostante il servizio sia stato ufficialmente chiuso nel 2016. Infatti rientrano in questa categoria anche gli utenti che decidono di propria iniziativa di spendere tutto il dust in fee. In generale non è possibile distinguere una transazione di "dust-collecting" da una transazione generata da un singolo utente, tuttavia il numero di dust della categoria "Speciale" permette di dare una sovrastima su quanto sia stato utilizzato questo servizio. Infatti il numero di dust spesi

in transazioni di "dust-collecting" può essere al massimo uguale al numero totale di dust della categoria "Speciale".

La seconda riflessione invece riguarda la tendenza a spendere il dust con almeno un altro address, in particolare notiamo un netto distacco nel 2015 dove la categoria "Successo" è nell'ordine di 10^4 mentre "Fallimento" dell'ordine di 10^3. In generale questa informazione è molto importante perché dimostra come il dust possa essere efficace per la de-anonimizzazione di un wallet, ovvero capire che più address appartengano ad un medesimo utente.

Una volta constatata la presenza di transazioni con input dust e con almeno due address diversi tra gli input, la fase seguente è l'analisi delle transazioni.

5.4 Classificazione delle Transazioni

Considereremo tre categorie di transazioni:

- **2+ address**: transazioni che presentano almeno due address di input differenti, la fee però è diversa dall'importo totale di input;
- **1 address**: transazioni con un singolo address di input;
- **speciale**: transazioni dove l'importo totale di input è uguale alla fee.

In totale il dust è stato speso in 263 963 transazioni, la tabella 5.2 mostra la percentuale delle transazioni nelle tre categorie, la tabella 5.3 mostra la distribuzione negli anni.

2+ address	63,2%
1 address	36,7%
speciale	0,1%

Tabella 5.2: Transazioni delle tre categorie

categoria/anno	2010	2011	2012	2013	2014	2015	2016	2017
2+ address(%)	100	93	55,6	68,4	55,9	64,5	62,7	75
1 address(%)	0	7	44,4	31,5	44,09	35,4	37,2	24,7
speciali(%)	0	0	0	0,1	0,01	0,1	0,1	0,3

Tabella 5.3: Transazioni delle tre categorie nel tempo

Dalle tabelle 5.2 5.3 confermiamo quanto detto prima, la prevalenza a spendere il dust in transazioni con almeno due address diversi e lo scarso utilizzo di servizi come Dust-B-Gone. Siccome sono presenti solo 17 transazioni della terza categoria

5.4.1 Categoria 1 Address

La categoria "1 address" è stata ulteriormente suddivisa in due classi:

- **NOD**(Not Only Dust): transazioni che presentanto almeno un input con importo maggiore di 545
- **OD**(Only Dust): transazioni con soli input dust.

Dalla tabella 5.4 possiamo dedurre che il fallimento nella de-anonimizzazione sia dovuto principalmente a quegli address che presentano un saldo positivo, ovvero un bilancio complessivo maggiore di zero. In questo caso gli address hanno potuto spendere il dust con altri input provenienti tutti dal medesimo address, impedendo ad un possibile attaccante di creare un cluster da associare ad un unico utente.

NOD	92 712	95,5%
OD	4328	4,5 %

Tabella 5.4: Classificazione OD e NOD

Particolarmente interessante invece è il secondo gruppo, ed abbiamo approfondito manualmente in particolare due address appartenenti ad esso:

- *1JwSSubhmg6iPtRjtyqhUYYH7bZg3Lfy1T*
- *1PEDJAibfNetJzM289oXsW1qLAgjYDjLgN*

Il fatto interessante del primo address è che la sua chiave privata è stata compromessa [1], consentendo a chiunque di riscattare bitcoin non appena vengono inviati a questo address. Questo address è tipicamente destinatario di transazioni di 2936 output dust, con tutti gli importi destinati ad esso. Speigazione più probabile è che tale address venga utilizzato come metodo per scrivere dati arbitrari sulla blockchain (nascosti all'interno di pagamenti irrisori) minimizzando i costi.

Caso anomalo invece riguarda il secondo address, rinominato *1PED* per semplicità. Questo address, come riportato in [22], sappiamo essere un noto gambler di Satoshi Dice, ma la sua singolarità deriva da alcune transazioni che ha generato. In queste transazioni infatti è presente un solo input di 1 satoshi e un solo output, ovviamente sempre di 1 satoshi. Questi singoli satoshi però non provengono da Satoshi Dice ma da altri address che seguono un pattern ben preciso, schematizzato nella figura 5.5

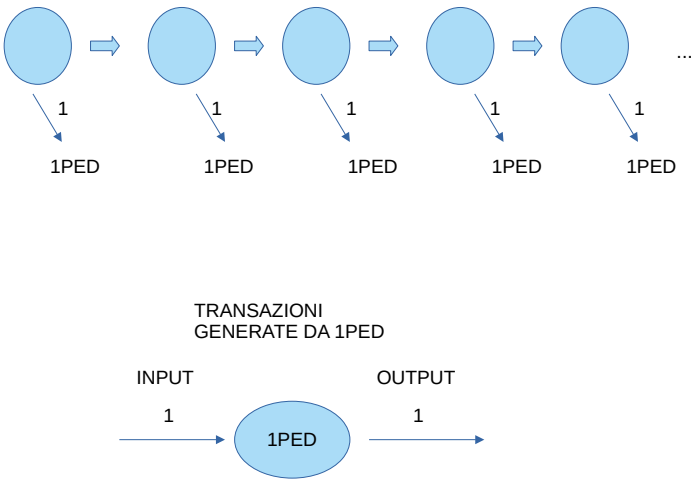


Figura 5.5: Schema transazioni legate a *1PED*

Ogni address, appartenente al pattern, invia due output, 1 satoshi con destinatario *1PED* e un secondo importo che viene inviato ad un altro address

[1] fonte:`https://privatekeys.pw/address/bitcoin/1JwSSubhmg6iPtRjtyqhUYYH7bZg3Lfy1T`

che seguirà il medesimo schema. In tutte queste transazioni la fee è sempre di 50 000 satoshi. *1PED* ha generato poi 1835 transazioni con un solo satoshi di input ed un solo satoshi di output, tutte in data 10 Marzo 2013 alle ore 16:20. È interessante sottolineare come *1PED* sia riuscito a spendere un singolo satoshi senza la necessità di aggregare il dust con altri input. Questo significa che, nonostante queste transazioni fossero non-standard, un miner ha comunque deciso di validarle, però sebbene questo comportamento sia anomalo risulta alquanto improbabile che si tratti di Dust Attack. Questa considerazione deriva dal fatto che tutti gli address di output delle transazioni generate da *1PED* sono address mai comparsi prima nella blockchain.

5.4.2 Categoria 2+ Address

Una volta esaminate le transazioni "1 address" il prossimo passaggio è l'analisi delle transazioni della categoria "2+ address". Anche in questo caso vengono mostrati i gruppi OD e NOD. La tabella 5.5 mostra come quasi sempre gli importi dust vengano aggregati ad altri importi non-dust.

NOD	166 778	99,9%
OD	128	0,1 %

Tabella 5.5: Classificazione OD e NOD

è importante però capire quanto successo abbia avuto la deanonimizzazione causata dal dust, analizzando soprattutto gli address di input. In particolare è importante capire quanti address diversi ci siano all'interno di una singola transazione, in modo da capire il livello di de-anonimizzazione che il dust permette. Generalmente più address riesco ad unire in un singolo cluster, maggiore è il livello di de-anonimizzazione ottenuta, infatti più address associo ad un singolo utente maggiore è la tranciabilità dell'attività di quell'utente.

Il grafico 5.6 mostra la distribuzione del numero di address diversi. Osserviamo subito come un notevole numero di transazioni tende ad avere un numero di address nel primo intervallo, ovvero nell'intervallo [1, 50]. È possibile notare come sia presente un elevato numero di transazioni con centinaia

di address di input. In tali transazioni è stato quindi possibile collegare centinaia di address diversi appartenenti ad un unico utente. Ad esempio la transazione con il maggior numero di address differenti contiene ben 1750 address diversi, nel qual caso è stato possibile collegare più di mille address ad un unico utente. In generale la media è di 13 address diversi per una singola transazione.

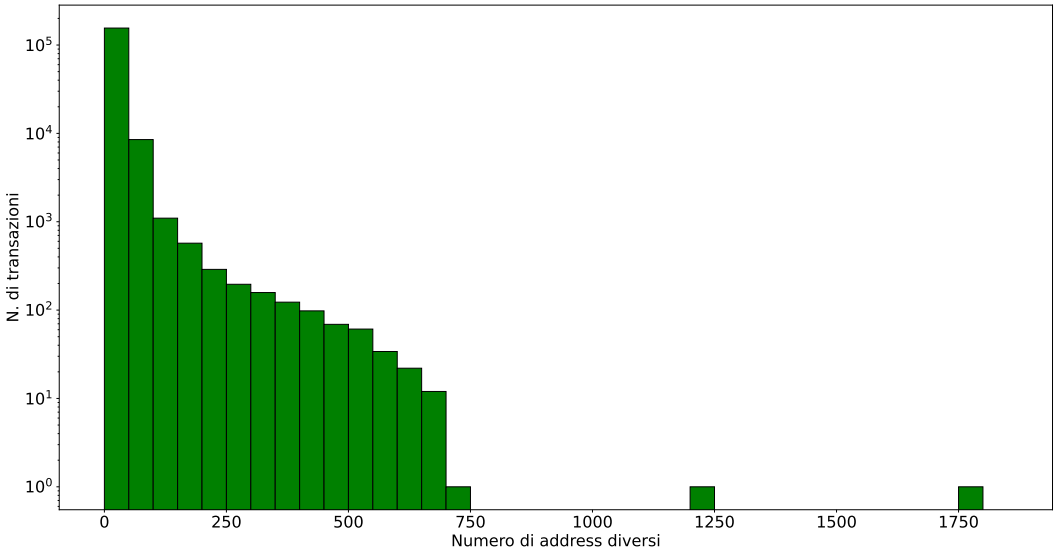


Figura 5.6: Distribuzione numero address differenti. Colonne di ampiezza 50, la prima colonna rappresenta l'intervallo [1, 50] (anche se non possono esistere valori con un singolo address essendo nella categoria 2+ "address")

Questi dati sembrerebbero confermare l'efficacia del Dust Attack, tuttavia è importante precisare che stessi address possano apparire in più transazioni, quindi diverse transazioni potrebbero riferire i medesimi address di input, portando ad una de-anonimizzazione ripetuta.

Nonostante siano stati generati 2 893 877 output dust gli address a cui sono stati inviati sono 1 059 836 e solo il 29%(312 114) lo ha speso. Di questo 29% abbiamo che 259 252 hanno speso il dust in transazioni della categoria "2+ address" anche se 4 270 address hanno generato anche transazioni nella categoria "1 address". Una particolarità interessante delle transazioni che hanno generato dust etichettato con "Successo" è che in molti casi erano

presenti tra gli output dust address nuovi, ovvero address che non erano mai comparsi sulla blockchain.

Ci sono 98 198 transazioni che generano almeno un dust della categoria "Successo", ma solo circa il 59%(58 146) non presenta address nuovi tra gli output. Come spiegato in sezione 3.3 è molto improbabile che un attaccante invii del dust ad un address nuovo, perché risulterebbe strano de-anonimizzare un address mai visto. Nella fase finale della procedura di analisi, quindi, sono state considerate solo transazioni senza indirizzi nuovi, così da trovare dei pattern interessanti di possibili Dust Attack di successo. Nel paragrafo successivo verranno approfonditi due di tali possibili schemi di Dust Attack.

5.5 Analisi di Pattern Interessanti

In generale è complicato affermare con certezza se un address stia compiendo un Dust Attack, però abbiamo individuato due schemi, con certe proprietà, che rappresentano candidati sospetti di Dust Attack. La proprietà più importante riguarda l'assenza di address nuovi tra gli output dust, poichè è alquanto improbabile tentare di de-anonimizzare un address mai visto fino a quel punto.

Il primo pattern, sintetizzato in figura 5.7, risulta molto simile al caso 1PED. Esistono diversi address legati a catena che generano transazioni con esattamente due output. Il secondo output, inviato sempre al solito address (probabilmente controllato dall'attaccante), viene speso dall'attaccante per generare transazioni contenenti solo output dust. Inoltre l'attaccante, nell'immagine l'ovale rosso, non compare mai nella catena dei finanziatori, rappresentata dai cerchi azzurri.

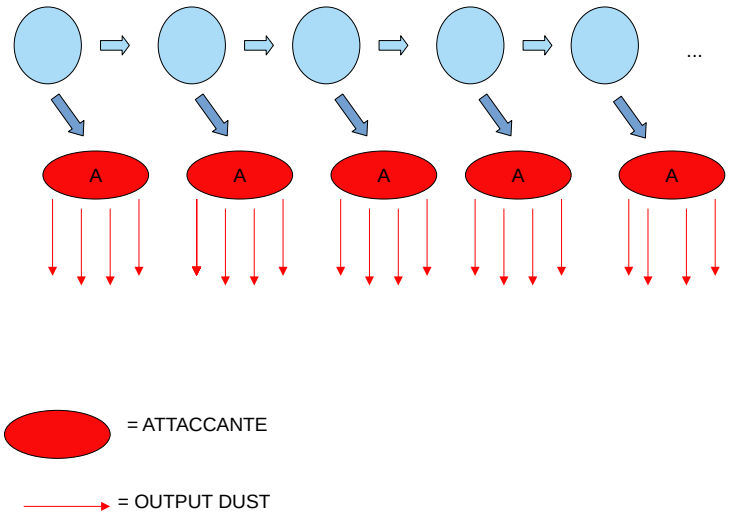


Figura 5.7: Pattern "Un finaziatore - un attaccante"

Altre proprietà importanti di questo pattern sono: fare riferimento ad un unico finanziatore, generare sempre lo stesso numero di output dust, avere sempre il medesimo importo di input ed, infine, eseguire tutte le transazioni in un breve intervallo di tempo. Un esempio di tale schema è realizzato dall'address *1DiRy9Giiq1GCkAD7VMSrXoKVe2dimnovm*, finanziato da *1Nj3AsYfhHC4zVv1HHH4FzsYWeZSeVC8vj*. Inoltre sono presenti altri address che seguono lo stesso schema, pagati sempre dallo stesso finanziatore di $1DiRy$. Questo significa che il pattern appena enunciato può essere eseguito in parallelo utilizzando diversi address attaccanti.

Potrebbero esserci anche più finanziatori che inviano denaro a più attaccanti. Per esempio l'address *16JLbXYe5xmxGNX8hiqooyTUJnhitNNqTh* ha ricevuto fondi da *1NPfnbqZAMUnGuNuYwVZdCN4qVzUq4ejG4*, l'aspetto interessante però è che il numero di output dust, l'importo dust inviato, l'importo input speso sono esattamente uguali a quelli usati da *1Diry*. Questo fatto fa ipotizzare che probabilmente questi address appartengano allo stesso utente o ad utenti con lo stesso obbiettivo. Ovviamente possono esserci eccezioni a questi due esempi trovati, per esempio avere più finanziatori o più attaccanti in una stessa catena. Inoltre gli importi e il numero di output dust

potrebbero avere una natura più casuale. In generale si potrebbero ipotizzare diverse variazioni di questo pattern, anche se non ne abbiamo verificato la presenza.

Il secondo pattern riguarda le transazioni "a catena" del 2013 citate nel paragrafo precedente. Questo schema è stato inizialmente scoperto tramite l'address *1JYvvL67LrSGCG77cy4rmpUXCFfSub4JkG* (rinominato *1JYvv*). Ogni address della catena invia centinaia o migliaia di output dust e il resto dell'importo viene inviato al prossimo address che seguirà lo stesso schema. La singolarità di questo modello è che gli address della catena, ad eccezione del primo, vengono utilizzati solo per eseguire quell'unica transazione. Non verranno mai più riutilizzati.

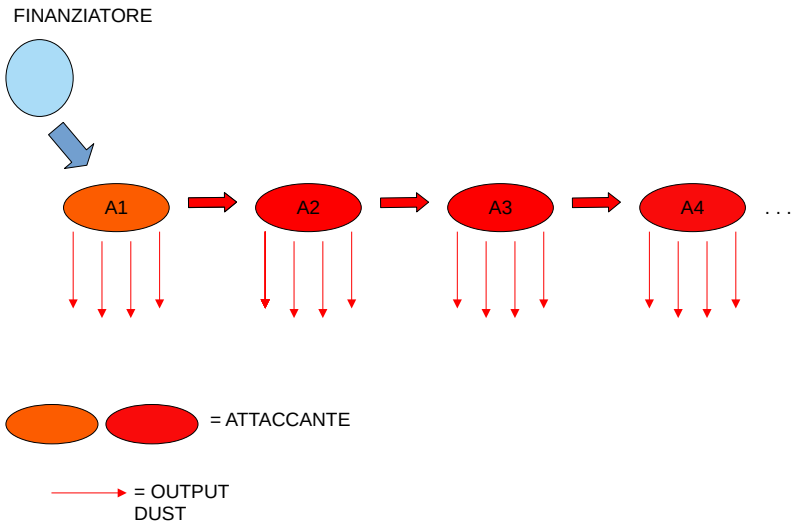


Figura 5.8: Pattern "Un finanziatore - più attaccanti"

Consideriamo ad esempio l'address A3 della figura 5.8. A3 riceve un importo da A2 che utilizza in parte per inviare centinaia o migliaia di output dust, ed il resto, esclusa la fee, viene inviato ad A4 che seguirà il medesimo schema. Gli address A2, A3 e A4 vengono utilizzati solo per eseguire queste transazioni, successivamente non verranno mai più riutilizzati, al contrario di A1 che viene usato per cominciare una serie di catene secondo questo schema.

Potrebbe essere interessante in futuro capire quanti address abbiano seguito schemi di questi tipo e se ci siano altri fini oltre a quello di un possibile Dust Attack.

Capitolo 6

Conclusioni e sviluppi futuri

In questa tesi abbiamo definito il Dust Attack, mostrando gli obiettivi dell'attacco, le conseguenze e le possibili contromisure. È stato analizzato il dust, escludendo gli importi dust generati da Satoshi Dice, in particolare è stato mostrato come il dust abbia permesso la creazione di diversi cluster di address in varie transazioni, dimostrando come il Dust Attack possa risultare efficace nella de-anonimizzazione dei wallet in Bitcoin. Infine sono stati descritti due pattern che hanno generato un elevato numero di output dust inviato ad address non-nuovi e che potrebbero rappresentare degli schemi di Dust Attack.

Alcuni sviluppi futuri di questo lavoro riguardano proprio questi due pattern introdotti nel paragrafo 5.5. Potrebbero essere svolte ulteriori analisi per identificare quanti pattern di questo tipo siano stati generati negli anni e se vengano utilizzati solo per eseguire un Dust Attack. Potrebbe essere interessante anche capire se gli address a cui viene inviato il dust siano scelti in modo casuale o se siano scelti address con determinate caratteristiche.

Un'altra idea potrebbe riguardare il confronto tra il Dust Attack ed altri attacchi di de-anonimizzazione che non si basano sull'invio di dust. In particolare potrebbe essere interessante analizzare i cluster ottenuti mediante aggregazione di importi dust e i cluster ottenuti con altre tipologie di analisi della blockchain, così da capire se il Dust Attack abbia permesso l'individuazione di cluster non osservabili utilizzando le altre tecniche di analisi.

Bibliografia

[1] Andreas M Antonopoulos. *Mastering Bitcoin: unlocking digital cryptocurrencies.* " O'Reilly Media, Inc.", 2014.

[2] Sarah Meiklejohn et al. «A fistful of Bitcoins: characterizing payments among men with no names». In: *Commun. ACM* 59.4 (2016), pp. 86–93. DOI: 10.1145/2896384. URL: https://doi.org/10.1145/2896384.

[3] Nakamoto Satoshi. «Bitcoin: A Peer-to-Peer Electronic Cash System». In: white paper (2008). http://www.bitcoin.org/bitcoin.pdf.

[4] Lawrence C. Washington. *Elliptic Curves Number Theory and Cryptography.* II Edition. Chapman Hall, 2008.

[5] *Peter Todd delayed txo commitments.* https://petertodd.org/2016/delayed-txo-commitments.

[6] Giulia Fanti e Pramod Viswanath. «Deanonymization in the Bitcoin P2P Network». In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA.* A cura di Isabelle Guyon et al. 2017, pp. 1364–1373. URL: https://proceedings.neurips.cc/paper/2017/hash/6c3cf77d52820cd0fe646d38bc2145ca-Abstract.html.

[7] Maxwell. *CoinJoin.* https://bitcointalk.org/index.php?topic=279249.0. 2013.

[8] Bitcoin Wiki. *Privacy – Forced address reuse.* https://en.bitcoin.it/wiki/Privacy#Forced_address_reuse.

[9] *Bitcoin Core.* https://github.com/bitcoin/bitcoin.

[10] *The Bitcoin Core developers: Bitcoin dust limit.* https://github.com/bitcoin/bitcoin/blob/c536dfbcb00fb15963bf5d507b7017c241718bf6/src/policy/policy.cpp.

[11] *Satoshi Dice.* https://en.bitcoin.it/wiki/Satoshi_Dice.

[12] Massimo Bartoletti, Bryn Bellomy e Livio Pompianu. «A Journey into Bitcoin Metadata». In: *J. Grid Comput.* 17.1 (2019), pp. 3–22. DOI: 10.1007/s10723-019-09473-3. URL: https://doi.org/10.1007/s10723-019-09473-3.

[13] *Bitcoin script.* https://en.bitcoin.it/wiki/Script.

[14] Ruben Recabarren e Bogdan Carbunar. «Toward Uncensorable, Anonymous and Private Access Over Satoshi Blockchains». In: *Proc. Priv. Enhancing Technol.* 2022.1 (2022), pp. 207–226. DOI: 10.2478/popets-2022-0011. URL: https://doi.org/10.2478/popets-2022-0011.

[15] Khaled Baqer et al. «Stressing Out: Bitcoin "Stress Testing"». In: *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers.* A cura di Jeremy Clark et al. Vol. 9604. Lecture Notes in Computer Science. Springer, 2016, pp. 3–18. DOI: 10.1007/978-3-662-53357-4_1. URL: https://doi.org/10.1007/978-3-662-53357-4%5C_1.

[16] *Bitcoin v0.9.0.* https://bitcoin.org/en/release/v0.9.0#how-to-upgrade.

[17] *Bitcoin v0.11.0.* https://bitcoin.org/en/release/v0.11.0.

[18] *Bitcoin v0.12.0.* https://bitcoin.org/en/release/v0.12.0.

[19] Massimo Bartoletti e Livio Pompianu. «An Analysis of Bitcoin OP_RETURN Metadata». In: *Financial Cryptography and Data Security - FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers.* A cura di Michael Brenner et al. Vol. 10323. Lecture Notes in Computer Science. Springer, 2017, pp. 218–230. DOI: 10.1007/978-3-319-70278-0_14. URL: https://doi.org/10.1007/978-3-319-70278-0%5C_14.

[20] Peter Todd. *Dust-B-Gone*. `https://github.com/petertodd/dust-b-gone`.

[21] *Pandas*. `https://github.com/pandas-dev/pandas`.

[22] Matteo Loporchio et al. ≪An analysis of Bitcoin dust through authenticated queries≫. In: Complex Networks (Novembre 2022).