Quantum Resistant Ledger (QRL)

Autore: peterwaterland@gmail.com

Traduzione: andrea.zitti@outlook.com

Novembre 2016

Estratto

Per garantire la propria longevità, una criptovaluta dev'essere sicura rispetto ai futuri avanzamenti tecnologici. Vengono proposti in questo documento lo sviluppo e l'emissione di un registro (catena dei blocchi) che fa uso di firme digitali basate su hash crittografici caratterizzati dalla resistenza a possibili attacchi computazionali da parte di calcolatori classici o quantistici.

1 Introduzione

Il concetto di creare un registro contabile basato su tecnologia Peer-to-Peer, registrato su una catena dei blocchi e reso sicuro tramite un meccanismo di Proof-of-Work trova per la prima volta riscontro nel 2008[11]. Bitcoin rimane, ad oggi, la criptovaluta più ampiamente utilizzata. Centinaia di simili registri crittografici sono stati successivamente creati ma, salvo alcune eccezioni, tutti si basano sullo stesso sistema crittografico di chiavi pubbliche a curva ellittica (ECDSA) per generare le firme digitali che permettono di verificare le transazioni in maniera sicura. I metodi crittografici più comunemente utilizzati al giorno d'oggi - ad esempio "ECDSA", "DSA" e "RSA" - sono in linea teorica vulnerabili all'attacco computazionale quantistico. In quest'ottica ha valore esplorare la progettazione e lo sviluppo di un registro crittografico "quanto-resistente" per far fronte ad un possibile, improvviso e non lineare progresso nello sviluppo dei calcolatori quantistici.

2 Sicurezza delle transazioni Bitcoin

Attualmente l'unico modo per spendere (output di transazione non ancora spesi) da un indirizzo bitcoin è quello di creare una transazione contenente una firma a curva ellittica ("secp256k1") proveniente dalla chiave privata ($x \in N | x < 2^{256}$) che controlla quello specifico indirizzo bitcoin. Se una chiave privata - che sia stata generata in maniera veramente casuale - è mantenuta segreta o viene persa, si può ragionevolmente considerare inamovibili i fondi da essa controllati.

La probabilità di collisione di una specifica chiave privata di bitcoin è 1 su 2^{256}. È possibile stimare la probabilità di collisione tra una qualunque chiave privata di un indirizzo bitcoin usando il Paradosso del Compleanno. Il numero di indirizzi bitcoin che devono essere generati per ottenere una probabilità di collisione dello 0.1% è 5.4×10^{23}[14].

3 Vettori di attacco computazionale quantistici

RSA, DSA e ECDSA restano sicure basando rispettivamente la loro difficoltà computazionale sulla fattorizzazione di grandi numeri interi, sul problema del logaritmo discreto e sul problema del logaritmo discreto con curva ellittica. L'algoritmo quantistico di Shor (1994) risolve la fattorizzazione di larghi interi e di logaritmi discreti in tempo polinomiale. Dunque, un computer quantistico sarebbe teoricamente in grado di ricostruire una chiave privata a partire dalla corrispettiva chiave pubblica ECDSA. Si ritiene che la crittografia ECDSA sia più vulnerabile all'attacco quantistico rispetto alla RSA per via dell'uso di chiavi più corte, con computer quantistici da 1300 e 1600 qubit (2^{11}) in grado di forzare cifrature ECDSA a 228 bit.

Lo sviluppo pubblico di computer quantistici non ha ad oggi sorpassato i 2^5 qubit o la fattorizzazione di piccoli numeri (15 o 21). Tuttavia, nell' Agosto 2015 la NSA ha deprecato la crittografia a chiave ellittica, apparentemente basandosi su preoccupazioni riguardo al calcolo quantistico. Non è chiaro quanto il calcolo quantistico sia stato sviluppato fino ad ora, né se eventuali passi avanti in questo campo verranno pubblicizzati per permettere una messa in sicurezza post-quantica dei protocolli crittografici comunemente usati su internet. Date le sue origini "anti-establishment", bitcoin potrebbe trovarsi tra i primi bersagli di un avversario in possesso del computer quantistico.

Se venisse reso pubblico un significativo avanzamento del calcolo quantistico, gli sviluppatori di nodi potrebbero implementare in bitcoin schemi di firma crittografica "quanto-resistenti" ed incoraggiare tutti gli utenti a muovere i propri fondi dagli indirizzi basati su ECDSA a quelli nuovi a prova di calcolo quantistico. Per mitigare la proporzione di indirizzi afflitti sarebbe ragionevole disabilitare il riutilizzo di chiavi pubbliche a livello di protocollo. Un aggiornamento pianificato in tal modo risulterebbe inoltre nella possibile movimentazione del milione di monete appartenenti a Satoshi Nakamoto - con la conseguente volatilità di prezzo che ne deriverebbe.

Uno scenario meno favorevole consisterebbe in un silenzioso e non lineare sviluppo del calcolo quantistico seguito da un graduale attacco agli indirizzi bitcoin dei quali sia stata esposta la chiave pubblica. Tali furti avrebbero un effetto devastante sui prezzi di scambio per via della grande e improvvisa pressione di vendita e della completa perdita di fiducia nel sistema man mano che verrebbe scoperta la portata dei furti. Il ruolo di bitcoin come riserva di valore ("oro digitale") sarebbe danneggiato con estreme conseguenze a livello mondiale. In questo contesto gli autori credono sia ragionevole sperimentare sistemi di firma crittografica resistenti al calcolo quantistico applicandoli al registro contabile di una criptovaluta e potenzialmente creando un valore "di riserva" in caso di eventi imprevisti di tale portata.

4 Firme quanto-resistenti

Esistono diversi e importanti sistemi crittografici ritenuti quanto-resistenti: la crittografia "hash-based", quella "code-based", quella basata su reticoli, quella sulle equazioni quadratiche multivariate e quella a chiave segreta. Tutti gli schemi menzionati sono considerati resistenti al calcolo quantistico fintanto che siano scelte chiavi di sufficiente lunghezza.

Esistono schemi di firma crittografica "hash-based" che con minimi requisiti di sicurezza sono "sicuri a posteriori" (dall'inglese "forward secure") e fanno esclusivamente affidamento alla resistenza alla collisione di una certa funzione di "hashing" crittografico. Cambiare la funzione di "hashing" prescelta produrrà un nuovo schema di firma digitale "hash-based". Le firme digitali "hash-based" sono un argomento ampiamente esplorato e rappresentano il principale candidato nell'impiego di

firme crittografiche post-quantistica in futuro. In quanto tali sono la tipologia di firma scelta per QRL.

5 Firme digitali hash-based

Le firme hash-based quanto-resistenti fanno affidamento sulla sicurezza di una funzione non invertibile che, dato un messaggio m, produce un output h (detto "hash-digest") di dimensione fissa n es. SHA-256, SHA-512. L'uso di una funzione di hash crittografico dovrebbe rendere infattibile l'individuazione di m a partire da h ("resistenza della controimmagine") tramite ricerca esaustiva ("brute force"), o il brute force di h a partire da $hdi2$, dove $hdi2 = Hash(h)$, mentre sarebbe molto difficile trovare due messaggi ($mdi1! = mdi2$) che producano lo stesso output h (resistenza alla collisione).

L'algoritmo quantistico di Grover può essere utilizzato per trovare collisioni tra hash o per effettuare un attacco alla controimmagine e trovare m, richiedendo $O(2^{n/2})$ operazioni. Dunque per mantenere una sicurezza a 128 bit - assumendo che la funzione di hash crittografico usata sia perfetta - sarebbe necessario impiegare un hash digest di lunghezza n di almeno 256 bit.

Le firme digitali hash-based richiedono una chiave pubblica pk per la verifica e una chiave privata sk per la firma del messaggio. Diversi sistemi di firma hash-based monouso (OTS) verranno discussi per quanto concerne la loro idoneità all'inclusione come componente in un registro contabile su blockchain.

5.1 Lamport-Diffie OTS

Nel 1979 Lamport descrisse un sistema di firme hash-based monouso per un messaggio di lunghezza m (solitamente risultante da una funzione di hashing resistente alla collisione). La generazione delle chiavi crittografiche crea m paia di chiavi segrete casuali, $sk_j^m \in \{0,1\}^n$, dove $j \in \{0,1\}$. es. la chiave privata è $sk = ((sk_0^1, sk_1^1), .., .., (sk_0^m, sk_1^m))$. Sia f una funzione non invertibile $\{0,1\}^n \rightarrow \{0,1\}^n$, con m paia di chiavi pubbliche generate $pk_j^m = f(sk_j^m)$, es. la chiave pubblica è $pk = ((pk_0^1, pk_1^1), .., .., (pk_0^m, pk_1^m))$. Firmare comporta l'ispezione bit a bit dell' hash del messaggio per selezionare sk_j (es. se $bit = 0, sk_k = sk_0, bit = 1, sk_j = sk_1$), creare la firma: $s = (sk_j^1, .., sk_j^m)$ che rivela metà della chiave privata. Per verificare una firma, l'ispezione bit a bit ($j \in \{0,1\}$) dell'hash del messaggio verifica che $(pk_j = f(sk_j))^m$.

Assumendo che dopo l'algoritmo di Grover sia desiderata una sicurezza a 128 bit, nella quale la lunghezza del messaggio è un hash di output fisso da SHA256, $m = 256$ e $n = 256$, risultando in $pk = sk =$ 16kb, e una firma di 8kb per ciascuna firma OTS utilizzata. Una firma Lamport dorebbe essere usata una sola volta e sebbene essa si possa generare molto velocemente, le notevoli dimensioni di chiavi, firme e conseguentemente anche delle transazioni, rendono il sistema di firme Lamport-Diffie impratico per un registro pubblico su blockchain.

5.2 Winternitz OTS

Per il digest di un messaggio, M, di lunghezza in bit m, con chiavi pubblica e privata di lunghezza n bit, una funzione non invertibile, $f : \{0,1\}^n \rightarrow \{0,1\}^n$ ed un parametro di Winternitz $w \in N | w > 1$,

l'idea generale della firma monouso di Winternitz è quella di iterare una funzione di hash su di una lista di chiavi private scelte casualmente, $sk \in \{0,1\}^n, sk = (sk_1, .., sk_{m/w})$, creando catene di hash di lunghezza $w-1$, terminando con chiavi pubbliche $(pk \in N|\{0,1\}^n), pk_x = f^{2^{w1}}(sk_x), pk = (pk_1, .., pk_{m/w})$. Diversamente dall'ispezione bit a bit del digest del messaggio nelle firme di Lamport, il messaggio è invece passato w bit per volta per estrarre un numero $i \in N, i < 2^w - 1$, dal quale la firma è generata. Al crescere di w si garantiscono chiavi e firme più corte al costo di un maggior impiego computazionale.[10]. La verifica consiste nel generare $pk_x = f^{2^{w-1}-i}(s_x)$ da M, s e confermare che le chiavi pubbliche coincidano.

Usando lo SHA-2 (SHA-256) come funzione non invertibile di hash crittografico, $f : m = 256$ e $n = 256$, con $w = 8$ risulta in una $pk = sk = s$ dimensione di $\frac{(m/w)n}{8}$ byte = 1kb. Per generare pk sono richieste f^i iterazioni di hash, dove $i = \frac{m}{w}2^{w-1} = 8160$ per ogni coppia di chiavi OTS generata. A $w = 16$ le chiavi e le firme si dimezzano di dimensioni, ma $i = 1048560$ diventa impratico.

5.3 Winternitz OTS+ (W-OTS+)

Buchman introdusse una variante dell'originale Winternitz OTS che applica ripetutamente la funzione iterante non invertibile ad un numero casuale, x, ma questa volta parametrizzata temporalmente da una chiave, k, che è generata dalla precedente iterazione di $f_k(X)$. Ciò è molto difficile da falsificare tramite attacchi a messaggio scelto adattivo quando si usa una funzione pseudo randomica (PRF: "Pseudo Random Function") e una prova di sicurezza può essere computata per i parametri dati[3]. Ciò rimuove l'esigenza di una famiglia di funzioni di hashing resistenti alla collisione effettuando un'esecuzione casuale della funzione piuttosto che una semplice iterazione. Huelsing ha introdotto un'ulteriore variante W-OTS+, che permette la crezione di firme di minori dimensioni a fronte di una pari sicurezza in termini di *bit* tramite l'aggiunta di una bitmask XOR all'interno della funzione di concatenazione iterativa[6]. Un'altra differenza tra W-OTS (variante del 2011) / W-OTS+ e W-OTS è che il messaggio è elaborato $log_2(w)$ bit per volta piuttosto che w, diminuendo le iterazioni della funzione di hashing ma aumentando le dimensioni di chiavi e firme.

Verrà ora brevemente descritto il metodo W-OTS+. Con un parametro di sicurezza, nN, corrispondente alla lunghezza in bit del messaggio (m), delle chiavi e delle firme, determinato dalla funzione di hashing crittografico scelta e dal parametro di Winternitz, $w \in N|w > 1$ (solitamente $\{4, 16\}$), l è calcolato. l è il numero di elementi (stringhe di n *bit*) in una chiave o firma WOTS+, dove $l = l_1 + l_2$:

$$l_1 = \lceil \frac{m}{log_2(w)} \rceil, l_2 = \lfloor \frac{log_2(l_1(w-1))}{log_2(w)} \rfloor + 1$$

Viene usata una funzione di hashing con chiave, $f_k : \{0,1\}^n 0 > \{0,1\}^n | k \in \{0,1\}^n$.

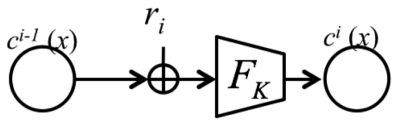
In pseudocodice:

$$f_k(M) = Hash(Pad(K)||Pad(M))$$

Dove $Pad(x) = (x||10^(b|x|1))$ per $|x| < b$.

La funzione di concatenazione, $c_k^i(x, r)$: su input di $x \in \{0,1\}^n$, contatore di iterazione i, chiave $k \in K$, e elementi di randomizzazione, $r = (r_1, ..., r_j) \in \{0,1\}^{n*j}$, con $j >= i$ è così definita:

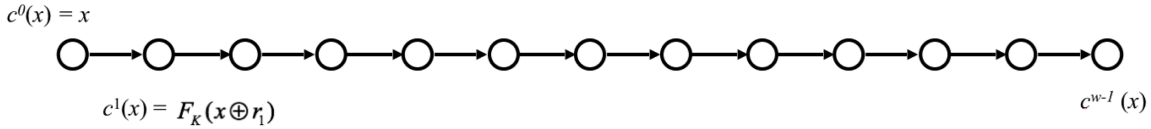
Fig. 1: Funzione di concatenazione W-OTS+



Dove:

$$c^i(x,r) = \begin{cases} x & \text{se } i = 0; \\ f_k(c_k^{i-1}(x,r) \oplus ri) & \text{se } i > 0; \end{cases}$$

Fig. 2: Esempio di generazione di una catena di hash



Che è uno xor bit per bit della precedente iterazione c_k e l'elemento di randomizzazione seguito da f_k sul risultato, che è poi dato come argomento della successiva iterazione di c_k.

5.3.1 Chiave di firma

Per creare la chiave privata, sk, vengono scelte uniformemente a caso (con PRF) $l + w - 1$ stringhe di n *bit*, delle quali la prima l costituisce la chiave privata, $sk = (sk_1, ..., sk_l)$ e le restanti $w - 1$ stringhe di n *bit* diventano $r = (r_1, ..., r_{w-1})$. Una chiave di funzione, k è scelta uniformemente a caso.

5.3.2 Chiave di verifica

La chiave pubblica è:

$$pk = (pk_0, pk_1, ..., pk_l) = ((r,k), c_k^{w-1}(sk_1, r), c_k^{w-1}(sk_2, r), .., c_k^{w-1}(sk_l, r))$$

Si noti che pk_0 contiene r e k.

5.3.3 Firma

Per effettuare una firma: il messaggio, M, di lunghezza m, è elaborato in modo tale che $M - (M_1, ..., M_l), M_i \in \{0, w-1\}$ (creando una rappresentazione di M in base w).

In seguito la "checksum", C, di lunghezza l_2 è concatenata e apposta:

$$C = \sum_{i=1}^{l_1}(w - 1 - M_i)$$

In modo tale che: $M + C = b = (b_0, ..., b_l)$

La firma è:

$$s = (s_1, ..., s_l) = (c_k^{b_i}(sk_1, r), ..., c_k^{b_l}(sk_l, r))$$

5.3.4 Verifica

Come verifica, viene ricostruita da M una firma $b = (b_1, .., b_l)$.

Se $pk = (c_k^{w-1-b_1}, .., c_k^{w-1-b_l}(s_l))$ allora la firma è valida.

W-OTS+ garantisce un livello di sicurezza di almeno $n - w - 1 - 2log(lw)$ bit[3]. Usando SHA-256 ($n = m = 256$), una tipica firma dove $w = 16$ è ln bit o 2.1kb.

6 Schemi di firma Merkle tree

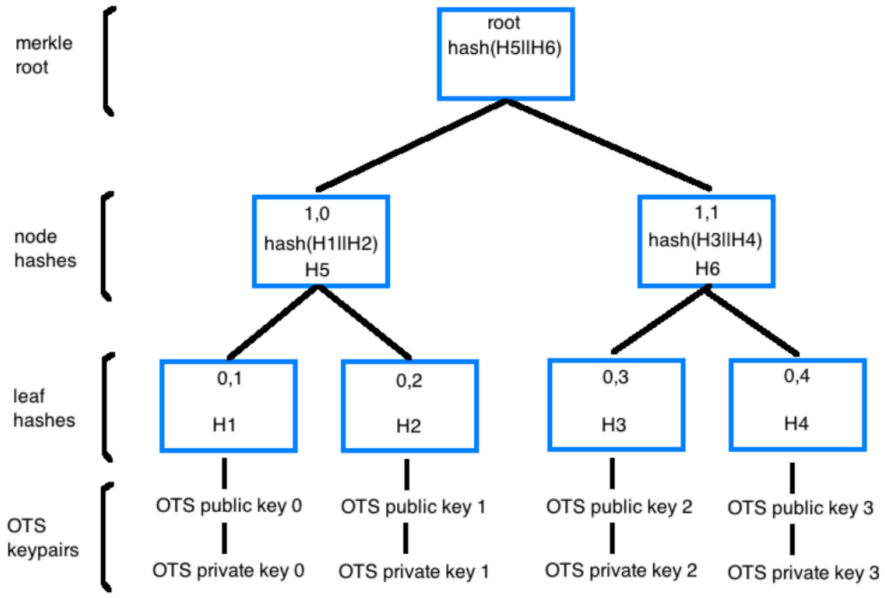
Sebbene le firme monouso forniscano un livello di sicurezza crittografica sufficiente per la firma e la verifica di transazioni, hanno come principale svantaggio il fatto di poter essere usate in modo sicuro solo una volta. Se un registro contabile si basa su una qualche trasformazione della chiave pubblica di un singolo paio di chiavi OTS, allora questo porterebbe ad avere un registro su blockchain estremamente restrittivo: tutti i fondi che si trovano su un indirizzo mittente avrebbero bisogno di essere mossi a ogni singola transazione (effettuata da quell'indirizzo) per non lasciarli esposti al rischio di furto. Una soluzione è quella di estendere lo schema di firma, incorporandogli più di una firma OTS valida per ciascun indirizzo del registro, permettendo tante firme quante paia di chiavi OTS sono state precedentemente generate. Un modo logico per ottenere questo risultato è quello di usare un albero di hash binari conosciuto come "merkle tree".

6.1 Albero di hash

L'idea generale sulla quale si basa il merkle tree è quella di un albero invertito composto di nodi genitori computati calcolando l'hash della concatenazione dei nodi figli tra loro fratelli salendo livello dopo livello verso la radice. L'esistenza di un qualunque nodo o foglia può essere crittograficamente dimostrato computando la radice.

Un albero merkle è formato da n foglie base e altezza a merkle root, $h(n = 2^h)$ - si parte a contare dagli hash foglia (livello 0), salendo di livello ad ogni successivo strato di nodi. Ciascun nodo foglia è creato in questo ipotetico caso di utilizzo del registro computando l'hash di una chiave pubblica OTS casualmente pre-generata. Dall'albero sottostante si può vedere che il nodo al di sopra di ciascun paio di hash foglia è a sua volta formato calcolando l'hash della concatencazione degli hash figli.

Fig. 3: Esempio di uno schema di firma merkle tree.



Ciò prosegue salendo tra gli strati dell'albero fino a confluire nell'hash radice dell'albero, noto come "merkle root" - radice di merkle.

Dall'albero esemplificativo mostrato nel diagramma, prendendo la merkle root come chiave pubblica, si possono usare quattro paia di chiavi OTS per generare quattro firme mononouso valide che sono crittograficamente sicure. La radice merkle dell'albero di hash binari può essere trasformato in un indirizzo del mastro (possibilmente tramite hashing iterativo con una checksum postposta). Una firma intera, S, di un messaggio, M, per un dato paio di chiavi OTS include: la firma, s, il numero di chiave ots, n, ed il percorso di autenticazione merkle. es. per il paio di chiavi OTS 0 (dunque $n = 0$):

$$S = s, n, OTS\_public\_key\_0, H1, H2, H5, H6, root$$

Dato che la chiave pubblica OTS e l'hash foglia possono essere dedotti da s, ed i nodi genitori possono essere computati dai loro figli, l'esempio sopra può essere semplificato in:

$$S = s, n, H2, H6, root$$

Dove S è validato verificando la chiave pubblica OTS da s e M, controllando poi che gli hash provenienti dal percorso di autentcazione merkle ricreino una corrispondente merkle root (chiave pubblica).

6.2 Stato

L'uso dello schema di firma merkle ("Merkle Signature Scheme", abbr. "MSS") sopracitato affida la sua sicurezza sul non riutilizzo delle chiavi OTS. Di conseguenza, dipende dallo stato delle firme o transazioni firmate che vengono registrate. Nell'uso del mondo reale, ciò sarebbe potenzialmente un problema, ma un registro pubblico su blockchain è lo strumento di archiviazione ideale per uno

schema di firme crittografiche a stati. Un più recente schema di firma crittografica hash-based chiamato SPHINCS che offre firme pratiche apolidi ("practical stateless signatures") con sicurezza a 2^{128} bit è stato annunciato nel 2015.

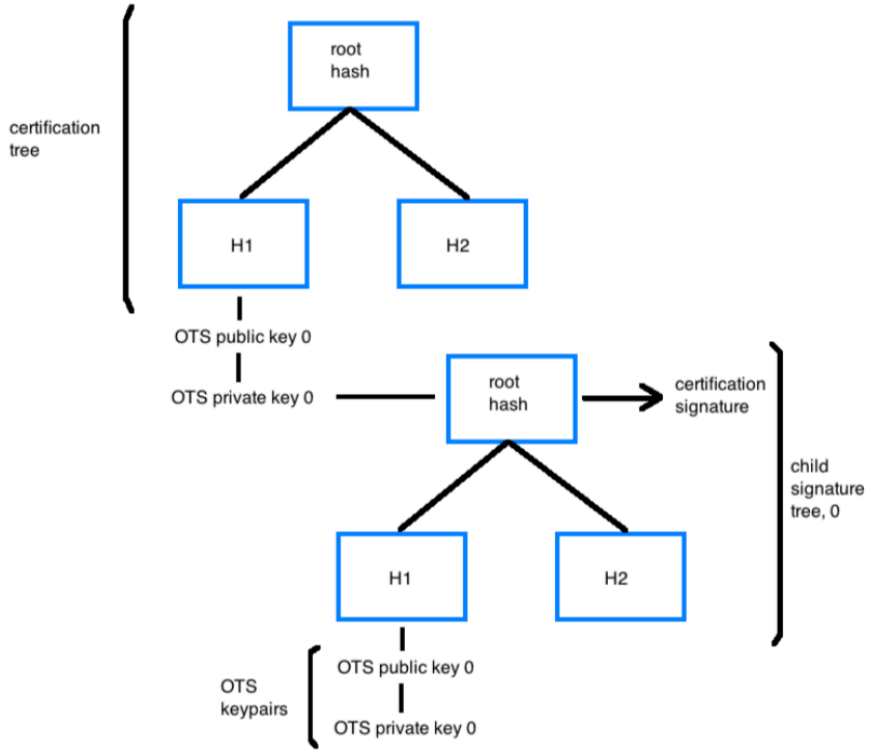
6.3 Iper-alberi

Un problema con l'MSS è che il numero di firme possibili è limitato e tutte le paia di chiavi devono essere generate in anticipo prima di procedere al calcolo dell'albero merkle. Il tempo di generazione di chiavi e firme cresce esponenzialmente con l'altezza, h, dell'albero: ciò significa che alberi più larghi di 256 paia di chiavi diventano temporalmente e computazionalmente onerosi da generare.

Una strategia per procrastinare il calcolo durante la generazione di chiavi e albero, e per estendere il numero di paia OTS disponibili è quella di usare un albero che è a sua volta composto da altri merkle trees, chiamato iper-albero.

L'idea generale è quella di firmare la radice di merkle di un albero figlio con la chiave OTS proveniente dall'hash foglia di un albero merkle genitore noto come albero di certificazione.

Fig. 4: Collegamento tra alberi merkle



Nella sua forma più semplice (altezza, $h = 2$) un albero di certificazione è pre-computato con 2^1 paia di chiavi OTS e quando è richiesta la prima firma viene computato un nuovo albero di firme merkle (albero di firme 0) e firmato usando una delle paia di chiavi OTS dell'albero di certificazione. L'albero di firme è composto di n hash foglia con corrispondenti paia di chiavi OTS, le quali servono

a firmare i messaggi come richiesto. Quando ciascun paio di chiavi OTS dell'albero di firme sarà stato usato, allora verrà firmato il successivo albero di firme (albero di firme 1) usando il secondo paio di chiavi OTS dell'albero di certificazione e sarà possibile generare il prossimo gruppo di firme.

Una firma, S, di una tale costruzione a iper-albero, diventa leggermente più complicata ed includerebbe: 1. dall'albero di firme: $s, n, merklepath, root$ 2. da ciascun albero di firme: s(da merkle root di albero figlio), $n, merklepath, root$

È teoricamente possibile annidare strati di alberi fino all'albero di certificazione per estendere il MSS indefinitamente. La dimensione delle firme cresce linearmente per ogni albero aggiuntivo che viene firmato, dunque la capacità di firma dell'iper-albero cresce esponenzialmente.

6.3.1 Esempi di iper-albero

Per dimostrare quanto facilmente si possa estendere il MSS con una costruzione ad iper-albero, si consideri un iniziale albero di certificazione di altezza, $h_1 = 5$, con 2^5 hash foglia e associate paia di chiavi OTS. La radice merkle di questo albero è trasformata per generare un indirizzo del registro contabile. Un altro albero merkle, un albero di firme di identiche dimensioni ($h_2 = 5$, 2^5 foglie e paia di chiavi OTS) è istanziato. Sono possibili 32 firme prima che sia necessario creare un altro albero di firme. Il numero totale di firme possibili è $2^{h_1+h_2}$ che in questo caso è $2^{10} = 1024$.

Su di un Macbook pro 2.7Ghz i5, con 8gb di ram, la creazione delle paia di chiavi OTS e di un albero di certificazione merkle di diverse dimensioni ha condotto questi risultati (codice python non ottimizzato, Winternitz OTS): $2^4 = 0.5s, 2^5 = 1.2s, 2^6 = 3.5s, 2^8 = 15.5s$. Un iper-albero consistente dell'iniziale generazione di due alberi da 2^4 richiede circa $1s$ a confronto dei $15.5s$ per un albero MSS da 2^8 di pari capacità di firma.

Aumentando la profondità (o l'altezza) di un iper-albero si prosegue in questa direzione. Un iper-albero composto concatenando quattro alberi da 2^4 ed un albero di firme di dimensione 2^4 ha una capacità di $2^20 = 1.048.576$ firme. Al costo di una maggior dimensione delle firme, il tempo di creazione è di soli $2.5s$.

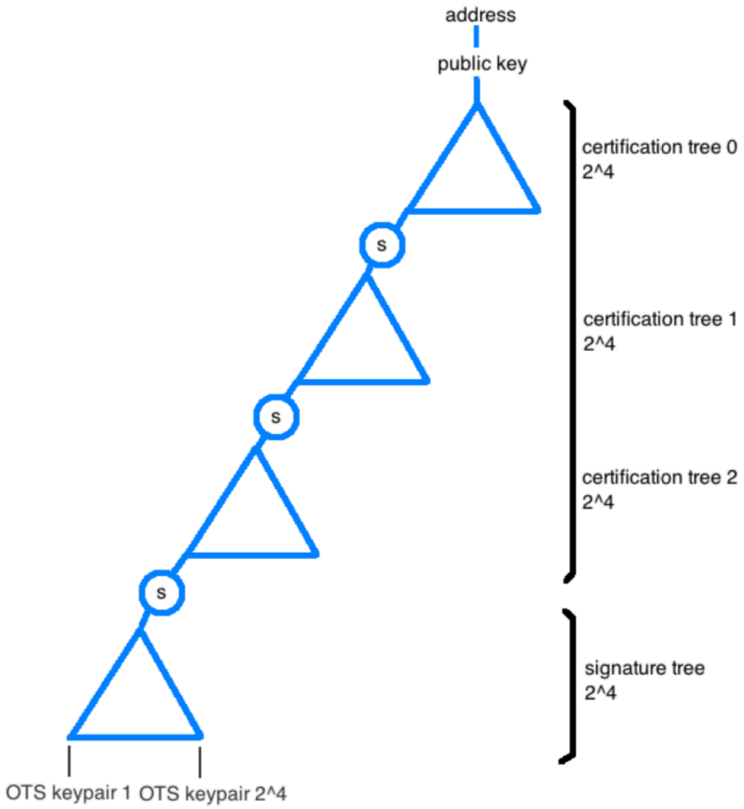
Non è necessario che un iper-albero sia simmetrico e dunque se inizialmente composto da due alberi potrà in seguito essere esteso firmando ulteriori strati di alberi. Le firme provenienti da un indirizzo del mastro contabile sarebbero dunque inizialmente piccole e aumenterebbero di dimensioni al raggiungimento di maggiori profondità dell' iper-albero.

L'uso di un iper-albero merkle per la creazione e la firma di transazioni da un indirizzo del registro contabile non richiederà verosimilmente mai piú di 2^{12} transazioni. Dunque, la capacità di firmare con buona facilità computazionale e in sicurezza fino a 2^{20} ad una profondità dell'iper-albero di $h = 5$ è più che sufficiente.

7 XMSS

Lo schema di firma merkle esteso ("Extended Merkle Signature Scheme", XMSS) fu menzionato per la prima volta da Buchmann et al. nel 2011 e fu pubblicato come bozza IETF l'anno successivo[4][7]. È dimostrabilmente sicuro in avanti ed esistenzialmente non falsificabile per quanto riguarda gli attacchi a messaggio selezionati, e ha minimi requisiti di sicurezza: una PRF ed una seconda funzione

Fig. 5: Costruzione di un iper-albero



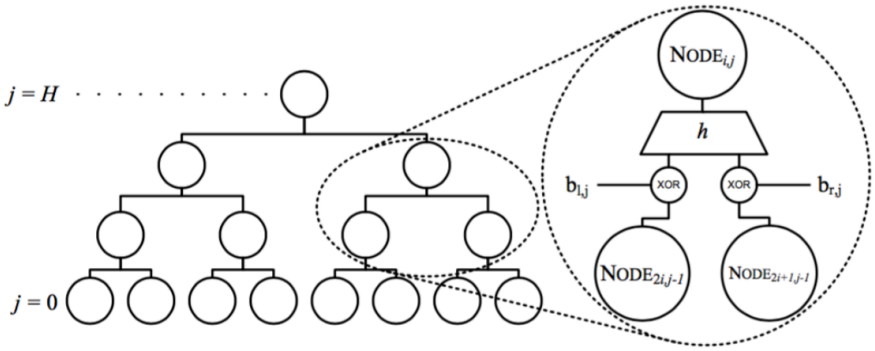
di hashing resistente alla controimmagine. Lo chema permette l'estensione di firme monouso attraverso un albero merkle che adotta come principale differenza l'uso della maschera di bit XOR dei nodi figli prima della concatenazione degli hash nel nodo genitore. L'uso della maschera di bit XOR permette di rimpiazzare la famiglia di funzioni di hashing resistenti alla collisione.

Le foglie dell'albero inoltre non sono hash di paia di chiavi OTS ma la radice degli L-alberi figli che contengono le chiavi pubbliche OTS con, l pezzi a formare le foglie di base. Per le firme monouso è utilizzato il sistema Winternitz OTS+ (nonostante sia stata descritta prima la variante del 2011).

La lunghezza in bit della chiave pubblica dell' XMSS è $(2(H + \lceil logl \rceil) + 1)n$, una firma XMSS ha lunghezza $(l + H)n$, e la lunghezza della chiave segreta di firma è $< 2n$.

Buchmann riporta performance, con un Intel(R) $i5$ 2.5 Ghz, un'altezza, $h = 20$ dell'albero XMSS, dove $w = 16$ e funzione di hashing crittografico SHA-256 ($m = 256$), di fino a circa un milione di firme. Con gli stessi parametri e lo stesso hardware, la firma ha impiegato 7ms, la verifica 0.52ms e la generazione delle chiavi 466 secondi. Il livello di sicurezza raggiunto con tali parametri è 196 bit per una dimensione di chiave pubblica di 1.7kb, chiave privata di 280 bit e firma di 2.8kb. XMSS rappresenta uno schema attraente, con il principale inconveniente di avere tempi di generazione delle chiavi estremamente lunghi.

Fig. 6: Costruzione di un albero XMSS



8 Performance dell'albero XMSS

Utilizzando una libreria python non ottimizzata costruita per un nodo test QRL, la formazione di un albero XMSS di 4096 foglie ($h = 12$), con tutte le chiavi e le maschere di bit generate da una PRF hash-based, ha impiegato 32s sullo stesso hardware sopra descritto (Macbook pro 2.7 Ghz i5, 8gb ram). Ciò include la generazione via PRF di oltre 8000 maschere di bit e oltre 300.000 fragmenti di sk. Un più efficiente algoritmo di attraversamento dell'albero merkle e la necessità di computare soltanto $w - 1$ hash per funzione a catena di chiave segreta ("$w1$ hashes per secret key chain function") in WOTS+ piuttosto che 2^{w-1} con WOTS contribuiscono all'estremo aumento di performance rispetto al convenzionale MSS.

In questa costruzione è stata raggiunta una dimensione di firma di circa 5.75kb (codifica di stringa esadecimale 11.75kb) includendo: paia di chiavi OTS n, firma, percorso di autenticazione XMSS, chiave pubblica OTS e la chiave pubblica dell'albero XMSS (incluso il seme della chiave pubblica della PRF e la radice dell'albero XMSS).

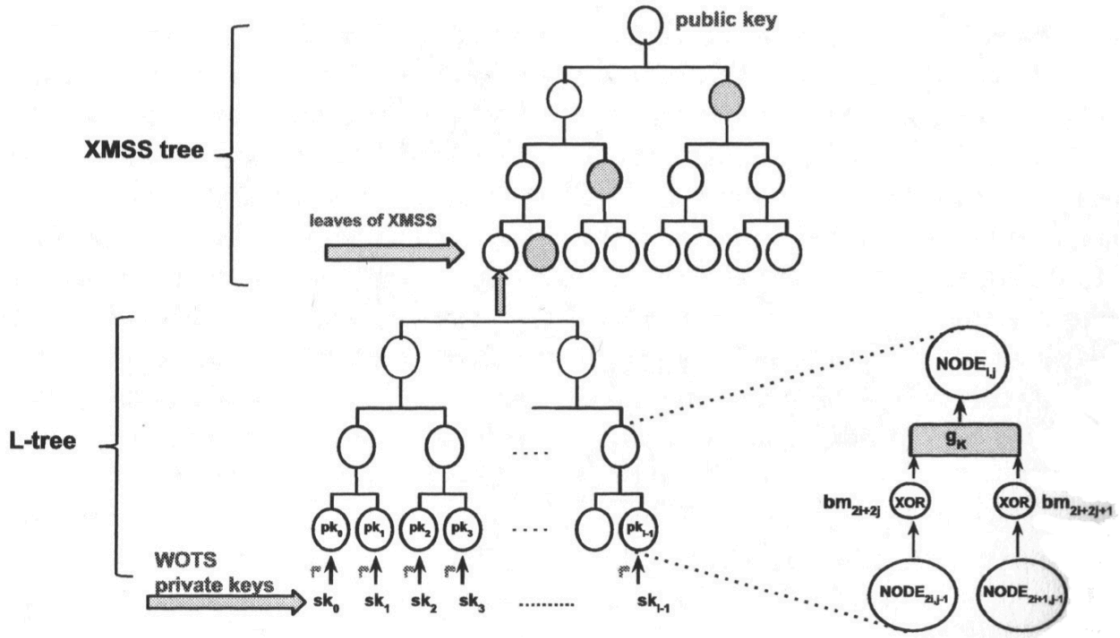
Per alberi di diversa capacità di firma, generati usando una PRF ed un seme casuale, sono stati ottenuti questi risultati: ($h = 9$) 512 4.2s, ($h = 10$) 1024 8.2s, ($h = 11$) 2048 16.02s.

9 Schema di firma proposto

9.1 Requisiti di sicurezza

Nella progettazione di QRL è importante che lo schema di firma sia utilizzato sia sicuro contro attacchi computazionali classici e quantistici sia ad oggi sia per decadi future. L'uso di XMSS e funzione di hashing SHA-256, con $w = 16$, offre una sicurezza a 196 bit che si prevede resterà al sicuro da attacchi computazionali a forza bruta fino all'anno 2164[9].

Fig. 7: Costruzione XMSS [8]



9.2 Firme QRL

È proposto uno schema di firme a iper-albero asimmetrico a stati composto di alberi XMSS concatenati. Questo garantisce il duplice vantaggio di impiegare uno schema di firme validato e di permettere la generazione di indirizzi contabili aventi l'abilità di firmare transazioni evitando il considerevole tempo di pre-computazione riscontrato nella costruzione di grandi XMSS. W-OTS+ è il sistema di firma hash-based monouso scelto per lo schema per ragioni sia di sicurezza che di performance.

9.3 Costruzione dell'iper-albero

9.3.1 Dimensione di chiavi e firme

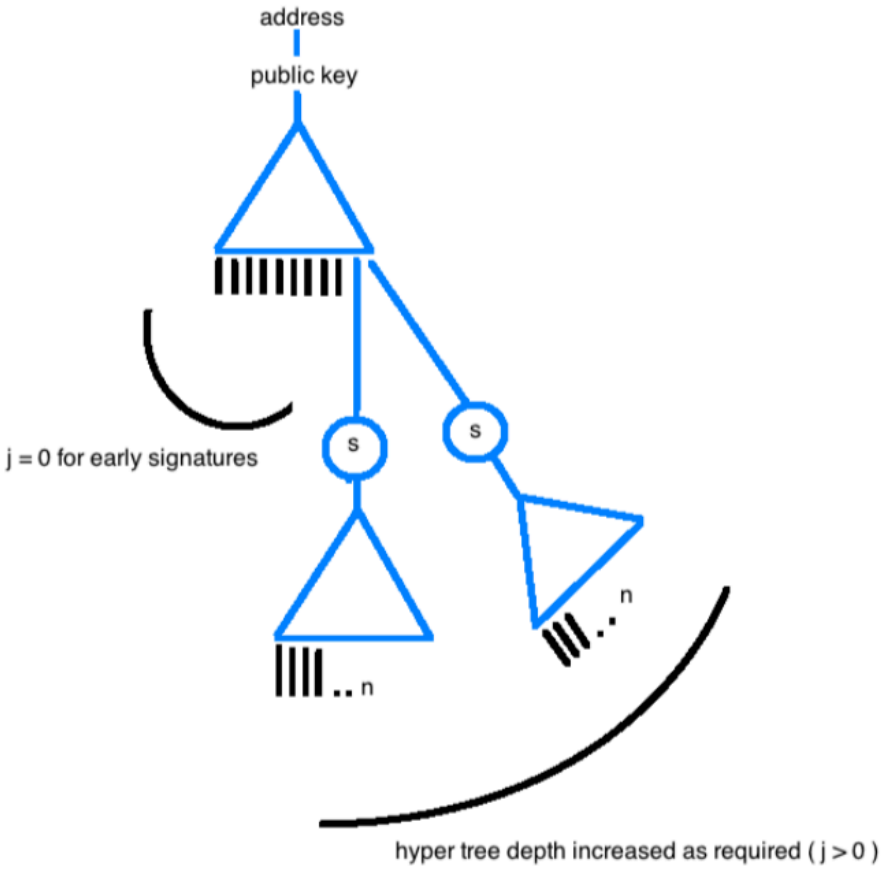
Al crescere del numero di alberi all'interno dell'iper-albero, le dimensioni di chiavi e firme crescono linearmente - ma la capacità di firma cresce esponenzialmente. Le dimensioni di diverse chiavi pubbliche e firme derivanti dall'albero XMSS (basandosi sulla descrizione del 2011), dove $w = 16, m = 256, h$ è l'altezza dell'albero e SHA-256 è scelto come algoritmo di hash crittografico, sono:

- $h = 2$, 2^2 firme: chiave pubblica 0.59kb, firma 2.12kb (0.4s);
- $h = 5$, 2^5 firme: chiave pubblica 0.78kb, firma 2.21kb (0.6s);
- $h = 12$, 2^{12} firme: chiave pubblica 1.23kb, firma 2.43kb (32s);
- $h = 20$, 2^{20} firme: chiave pubblica 1.7kb, firma 2.69kb (466s[3]).

9.3.2 Asimmetria

La creazione di un albero asimmetrico permette la preparazione delle prime firme con la costruzione di un singolo albero XMSS che verrà eventualmente esteso alla richiesta dalle successive firme, al costo di una meno ampia capacità di firma. Le ragioni di ciò stanno nel fatto che questa scelta non ha verosimilente conseguenze negative nell'applicazione di un registro contabile su blockchain, ed il wallet può comunque dare all'utente l'opzione di favorire la capacità di firma a discapito di maggiori dimensioni di chiavi e firme prodotte. Una profondità massima di $j = 2$ dovrebbe essere sufficiente in ogni circostanza.

Fig. 8: Iper-albero asimmetrico



9.4 Specifiche dell iper-albero di QRL

I seguenti parametri di default sono da adottare per la costruzione di un iper-albero standard:

- $j = 0(j \in 0 <= x <= 2), h = 12(h \in 1 <= x <= 14)$, limite massimo di firme possibili: 2^36, minima dimensione di firma: 2.21kb, massima dimensione di firma: 7.65kb.

es. Un singolo albero XMSS, $h = 12$ con 4096 possibili firme, che potrebbe se richiesto essere esteso

con ulteriori alberi fino a $h = 14$. Per la maggior parte degli utenti è improbabile che sia mai richiesta la generazione di alberi aggiuntivi.

9.4.1 Esempio di firma QRL

Assumento che la più complicata costruzione a iper-albero sia quella con $j = 2$ e $h = 14$, una firma, s per un messaggio, m, dove n è la posizione del paio di chiavi OTS per ogni albero XMSS, richiederebbe:

- Albero di firma, $j = 2$: firma OTS di m, n, prova di autenticazione merkle, radice merkle dell'albero di firma;
- Albero di certificazione, $j = 1$: firma OTS della merkle root dall'albero di firma ($j = 2$), n, prova di autenticazione merkle, radice merkle;
- Albero XMSS originale, $j = 0$: firma OTS di della merkle root ($j = 1$), n, prova di autenticazione merkle, radice merkle.

La verifica consiste nel generare la chiave pubblica OTS da m e la firma, confermando poi la prova di autenticazione merkle fornita si genera la merkle root dell'albero di firma. Questo diventa il messaggio per la prossima firma OTS e da questo è generata la successiva chiave pubblica OTS, la prova di autenticazione merkle fornita è usata per ricreare la radice merkle dell'albero di certificazione, che diventa il messaggio per la firma OTS del prossimo albero di certificazione, e così via. Una firma è valida soltanto se la merkle root dell'albero più alto - ovvero l'albero XMSS originale ($j = 0$) - viene generato correttamente.

Si noti che le chiavi pubbliche OTS non sono richieste per la verifica della firma dell'albero XMSS. Infatti la merkle root per ciascun albero può esser dedotta e quindi omessa tramite verifica della firma dell'iper-albero qualora sia noto l'indirizzo mittente (in quanto è un derivato della computazione della merkle root per il più alto albero di certificazione XMSS ($j = 0$) all'interno della firma QRL - si veda successivamente la sez. "Conti").

9.5 PRF

PRF dal seed. HMAC_DRBG.

9.6 Portafogli deterministico

Usando un singolo SEED è possibile generare un albero XMSS molto largo, che dovrebbe per la maggior parte degli utenti essere sufficiente per lungo tempo. Una fonte sicura di entropia è usata per generare questo SEED che passando attraverso una funzione PRF genera un set di chiavi pseudo-casuali che generano l'albero. Uno svantaggio nell'utilizzare lo stesso albero XMSS è che l'utente è confinato ad un singolo indirizzo (nonostante l'esposizione della chiave pubblica non sia un problema con un MSS).

Un indirizzo bitcoin o ethereum è derivato dalla chiave pubblica associata e come tale una singola chiave pubblica o privata può generare un singolo indirizzo. Invece, un indirizzo XMSS è derivato

dalla chiave pubblica, PK, che contiene la merkle root ed il SEED pubblico. Se il SEED rimane costante ma il numero di paia di chiavi OTS per computare l'albero varia, allora la radice merkle cambierà a ciascuna variazione. Dunque per ogni singola addizione o sottrazione di singole paia di chiavi OTS cambierà l'indirizzo derivato.

Questa caratteristica potrebbe essere utilizzata dal software portafogli/nodo per generare numerose variazioni dell'albero XMSS (estendendo/contraendolo in base all'esigenza usando il SEED iniziale) permettendo tanti indirizzi unici quanto fosse necessario generare. Registrare questa informazione in maniera sicura, compatta e con stato è un compito banale.

10 Parametri di progettazione della crittovaluta

La parte restante del foglio bianco indicherà i parametri di progettazione proposti per il registro QRL. L'obiettivo del registro è quello di essere una catena dei blocchi pubblica altamente sicura rispetto ai vettori di attacco computazionale classici e quantistici. Questa è una bozza iniziale e pertanto ogni aspetto potrebbe essere soggetto a cambiamento.

10.1 Proof-of-Stake

QRL sarà un registro aperto su una blockchain pubblica protetta da un algoritmo di Proof-of-Stake. Un'epoca dura 10.000 blocchi. I validatori di stake sono determinati dalle transazioni di stake nell'epoca precedente. L'idea generale è che ciascun validatore di stake firma una transazione contenente l'hash finale di una catena iterativa di 10.000 hash di lunghezza (una maschera di bit XOR potrebbe essere applicata durante ciascuna iterazione al fine di ridurre i requisiti di sicurezza della funzione di hash). Con la conferma su blockchain della transazione di stake ciascun nodo della rete può ora collegare l'identità crittografica dell'indirizzo di stake alla catena di hash per la prossima epoca.

10.1.1 Progettazione e casualità

Per ogni blocco, ciascuno dei nodi di convalida che sta partecipando (al Proof-of-Stake) durante l'epoca attuale rivela il prossimo hash precedente consecutivo nella catena per dimostrare crittograficamente la sua partecipazione e votare per essere il selettore del blocco vincente.

Viene usato HMAC_DRBG per generare una sequenza numerica pseudocasuale di 32 byte di output dai dati seed provenienti dalla blockchain (inizialmente il blocco di genesi, poi entropia aggiunta prendendo gli hash degli header di blocco recenti concatenati per ogni epoca successiva).

Pertanto, per ogni blocco, è scelto come selettore del blocco il nodo validatore di stake che ha l'hash numericamente più vicino all'output generato dalla PRF per quel blocco. Ciò risulta difficile da anticipare in quanto la PRF è sconosciuta ai partecipanti durante la generazione della catena di hash. Inoltre, una catena di hash iterativa (con chiave) è essenzialmente una sequenza di numeri casuali. Infine, anche se i validatori di stake colludessero in qualche modo, non potrebbero conoscere l'altro contenuto della catena di hash del validatore di stake poiché non sarebbe ancora stato rivelato.

Per prevenire attacchi sotto forma di ritardo nei blocchi, la mancata produzione di un blocco valido dopo che sia stato presentato un hash di rivelazione valido è punito con la perdita dell'intero premio

del blocco da quell'indirizzo, che viene quindi bandito temporaneamente dalla partecipazione.

Per mitigare l'attaccabilità rispetto a blocchi vuoti provenienti da nodi Sybil o indirizzi validatori di stake con saldo troppo basso, si applica una soglia flessibile alla lista di validatori di stake. Il premio del blocco è pagato ponderatamente al saldo dell'indirizzo in stake. Con il messaggio hash di rivelazione ogni nodo rivela anche un hash della radice del merkle tree di un elenco ordinato di txhash presenti nella sua transaction pool insieme al numero di transazioni che resta in attesa di un blocco. Ciascun nodo seleziona una percentuale dall'alto e dal basso della lista per vedere quante transazioni sono previste per il blocco successivo. Se il blocco è vuoto o ne ha meno del numero previsto, il numero di validatori di stake al quale è permesso partecipare (esclusi i validatori di stake da poveri a ricchi) aumenta a ogni blocco. Se i nodi selettori dei blocchi si stanno comportando onestamente allora é vero il contrario e cresce il numero di validatori di stake ai quali è permesso partecipare. I fondi non possono essere spostati per tutta l'epoca in cui sono tenuti in stake - questo previene tentativi di truccare la selezione dei blocchi creando molteplici indirizzi di validatori Sybil.

10.2 Commissioni

Le maggiori dimensioni delle transazioni rispetto altri registri su blockchain rende necessario il pagamento di una tariffa a ciascuna transazione. L'autore è dell'opinione che mercati di tariffe artificiali (si veda bitcoin) non siano necessari e siano in contrasto con l'idea di un registro contabile aperto su blockchain pubblica. Ciascuna transazione, fintanto che paghi una tariffa minima, dovrebbe essere valida quanto ogni altra. La tariffa minima che i miners hanno intenzione di accettare dovrebbe essere variabile in base a quanto deciso dal mercato. es. nodi/miners determinano competendo tra di loro il costo minimo di una transazione. Verrà imposto un limite minimo assoluto a livello di protocollo. I miner ordineranno le transazioni da includere nel blocco a loro discrezione.

10.3 Blocchi

10.3.1 Tempo del blocco

Bitcoin ha un tempo di blocco di circa 10 minuti, ma con la varianza questo può in alcune occasioni portare a tempi di attesa piuttosto lunghi prima che venga minato il blocco successivo. I modelli di registro piú moderni, come ethereum, sono migliorati sotto questo punto di vista e godono di tempi di blocco molto piú brevi (15 secondi) senza perdita di sicurezza o ulteriore centralizzazione dei miner causata da alti tassi di blocchi orfani o in stallo. Ethereum usa una versione modificata del protocollo "Greedy Heaviest Observed Subtree" che permette ai blocchi orfani o in stallo di essere comunque inclusi nella blockchain e ricompensati[13][5].

Dal momento che QRL prevede di utilizzare un algoritmo proof-of-stake sin dall'inizio, ci aspettiamo di poter utilizzare in sicurezza un tempo di blocco di durata compresa tra i 15 e i 30 secondi.

10.3.2 Premio del blocco

Ciascun nuovo blocco creato includerà una prima transazione "coinbase" contenente un indirizzo di mining al quale far pervenire il premio uguale alla somma del premio coinbase e la somma combinata delle tariffe di transazione pagate all'interno del blocco. Il premio di blocco è ponderato in base al saldo dell'indirizzo del validatore di stake scelto come selettore del blocco.

Il premio del blocco è ricalcolato dal nodo miner a ogni blocco e segue lo schema di emissione della moneta.

10.3.3 Dimensione del blocco

Per evitare controversie, verrebbe usata una soluzione adattiva modellata sulla proposta di Bitpay di aumentare la dimensione del blocco in base ad un multiplo, x, della dimensione mediana, y, degli ultimi z blocchi[12]. L'uso di una mediana previene l'abuso da parte dei miners attraverso l'inclusione di blocchi vuoti o troppo grandi per alterare la dimensione media del blocco. x e z sarebbero quindi regole di consenso difficili da rispettare per la rete. La massima dimensione del blocco, b sarebbe così semplicemente calcolata con:

$$b = xy$$

10.4 Unità e denominazioni della valuta

QRL userà un gettone monetario, il *quantum* (plurale *quanta*), come unità monetaria di base. Ciascun *quantum* è divisibile fino a un più piccolo elemento come segue:

- 1 : Shor
- 10^3 : Nakamoto
- 10^6 : Buterin
- 10^{10} : Merkle
- 10^{13} : Lamport
- 10^{16} : Quantum

Dunque, ogni transazione che coinvolga una frazione di *quantum* corrisponde a un numero intero di unità *Shor*. Le spese di transazione sono pagate e quantificate in unità *Shor*.

10.5 Conti

I saldi degli utenti sono registrati negli accounts. Ciascun account è semplicemente un indirizzo unico riutilizzabile del registro identificato da una stringa che inizia per "Q".

Un indirizzo è generato eseguendo lo SHA-256 della merkle root del più alto albero di certificazione XMSS disponibile. A questa è poi apposta una checksum di quattro byte (formati dai primi 4 byte del doppio hash SHA-256 della merkle root) ed è preposta la lettera "Q". es. in pseudocodice simile al python:

$$Q + sha256(merkleroot) + sha256(sha256(merkleroot))[:4]$$

Un tipico indirizzo di un account:

$$Qcea29b1402248d53469e352de662923986f3a94cf0f51522bedd08fb5e64948af479$$

Ciascun account ha un saldo denominato in *quanta* divisibile fino alla singola unità di *Shor*.

Gli indirizzi sono aggiornati con ogni transazione usando una nuova coppia di chiavi OTS e QRL regisra tutte le chiavi pubbliche mai utilizzate (questa potrebbe essere rimossa in quanto può venire rigenerata sul momento a partire dalla firma e dal messaggio della transazione, ma sarebbe operativamente intensiva) da ciascun account. Un contatore di transazioni chiamato nonce viene incrementato ad ogni transazione inviata da un account. Ciò consente ai portafogli che non memorizzano l'intera blockchain di tenere traccia della loro posizione nello schema di firma ad iper-albero merkle a stato.

10.6 Emissione di moneta

10.6.1 Considerazioni storiche

Bitcoin fu la prima criptovaluta decentralizzata e inizialmente era un esperimento privo di valore monetario, dunque fu appropriato distribuire la valuta interamente attraverso il mining. Più recentemente Zcash ha scelto lo stesso processo ma devolvendo una % del premio di ciascun blocco minato durante un primo periodo di emissione al progetto open source - risultando in un'incredibile volatilità di prezzo. Altri registri, come ethereum, hanno invece venduto una ampia % della riserva finale come parte di una "Initial Coin Offering" ("ICO"). Questo comporta il vantaggio che gli "early adopters" possano ancora potenzialmente guadagnare supportando il progetto, ma aggiuntivamente il progetto stesso può generare fondi per continuare lo sviluppo, avviare e crescere il progetto fin dalla sua infanzia. L'approccio ICO permette ad un mercato di svilupparsi facilmente in quanto una maggior quantità di monete è disponibile da comprare e vendere sin dal blocco di genesi.

Auroracoin (2014) adottò un approccio differente, offrendo una ugual quota della ICO a chiunque vivesse in Islanda, mentre gli svliluppatori tennero per loro il 50% dell'intera riserva di moneta.

Altre criptovalute hanno o semplicemente clonato bitcoin per intero oppure iniziato da zero con una nuova catena e diversa codebase.

10.6.2 Trasferimento di saldi tra catene

È possibile inizializzare QRL basandosi su una immagine istantanea dell'attuale registro bitcoin inserito nel blocco di genesi di QRL. L'idea generale sarebbe quella di permettere agli utenti di creare una transazione di importazione monouso contenente un messaggio unico ed una firma (es. l'indirizzo a un portafogli QRL generato casualmente firmato con una chiave privata bitcoin appartenente ad un indirizzo con saldo positivo alla data dell'istantanea). Questa caratteristica potrebbe rimanere disponibile fino ad una certa altezza di blocco e successivamente la parte restante della riserva di moneta verrebbe minata come normale. L'iniziale accumulo di informazioni nel blocco di genesi verrebbe "potato" alla stessa altezza di blocco. Uno svantaggio di questo metodo è che, per quanto equo, penalizzi i possessori di altre crittovalute oltre a bitcoin ed è tecnicamente potenzialmente complicato da eseguire per i nuovi utenti. Potrebbe esporre permanentemente le chiavi pubbliche

degli indirizzi bitcoin utilizzati nel processo e sarebbe importante spostare successivamente i fondi in un nuovo indirizzo bitcoin generato casualmente per mitigarne i rischi.

(? Si potrebbe anche permettere lo stesso processo ai possessori di ethereum.)

10.6.3 Emissione proposta - bozza

L'emissione iniziale di QRL sarà la seguente:

- ICO di 1 milione di *quanta* (4.7% della riserva finale di moneta) prima del lancio.
- Una istantanea dello stato di tutti gli indirizzi bitcoin con saldo superiore a 0.01 btc utilizzato per formare il blocco di genesi iniziale di QRL. Chiunque voglia trasferire direttamente monete ad un cambio 1:1 dal registro di bitcoin a quello di QRL sarà in grado di farlo fino ad altezza di blocco 518400 (3-6 mesi) attraverso il nodo wallet.
- Un ulteriore 1 milione di *quanta* sarà destinato alla fondazione nel blocco di genesi;
- La riserva restante sarà minata (21.000.000 - (2.000.000 + saldi btc importati entro l'altezza di blocco 518400)).

10.7 Piano di emissione delle monete

Caratteristiche determinanti di bitcoin sono la sua scarsità ed il limite massimo imposto alla sua l'emissione monetaria. QRL seguirà bitcoin in tal proposito, imponendo un limite fisso superiore all'emissione di moneta di 21×10^6 *quanta*. È però preferito un continuo decadimento esponenziale nel premio del blocco fino al raggiungimento del limite massimo di circolante. Questo eliminerà la volatilità associata al fenomeno di dimezzamento del premio di blocco presente in bitcoin.

La massima riserva di moneta, $x = 21 \times 10^6$, meno le monete create durante il blocco di genesi, y, ridurrà esponenzialmente a partire da Z_0. La curva di decadimento è calcolata in modo tale da distribuire i premi del mining per approssimativamente 200 anni (fino al 2217AD, 420480000 blocchi a 15 blocchi al secondo) finchè rimarra un singolo *quanta* rimasto non minato (il mining potrà comunque continuare oltre a quel momento).

Si può calcolare la riserva monetaria residua al blocco t, Z_t con:

$$Z_t = Z_0 e^{-\lambda t}$$

Il coefficiente, λ, è calcolato da: $\lambda = \frac{ln Z_0}{t}$. Dove t è il numero totale di blocchi nello schema di emissione fino al *quanta* finale. Fino al blocco 518400, $\lambda = 3.98590885111 \times 10^{-08}$. Il premio del blocco, b, è calcolato per ciascun blocco con:

$$b = Z_{t-1} - Z_t$$

Tra il blocco di genesi ed il blocco numero 518400 i saldi bitcoin potranno essere trasferiti nel registro contabile tramite funzioni di importazione. Al blocco 518401 lo schema di emissione verrà aggiornato confermando i saldi appena importati, riducendo Z_t e aggiornando il premio del blocco.

Bibliografia

[1] http://oxt.me/charts

[2] D. Bernstein *Sphincs: practical stateless hash-based signatures.* 2015

[3] J. Buchmann *On the security of the winternitz one-time signature scheme.*

[4] J. Buchmann *Xmss – a practical forward secure signature scheme based on minimal security assumptions.* 2011

[5] V. Buterin *Ethereum whitepaper.* 2013

[6] A. Hulsing *W-ots+ - shorter signatures for hash-based signature schemes.* 2013

[7] A. Hulsing *Xmss: Extended hash-based signatures.* 2015

[8] A Karina *An efficient software implementation of the hash-based signature scheme mss and its variants.* 2015

[9] A. Lenstra *Selecting cryptographic key sizes.* 2001

[10] R. Merkle *A certified digital signature. CRYPTO, 435,* 1989

[11] S. Nakamoto *Bitcoin: A peer-to-peer electronic cash system.* 2008

[12] S. Pair *A simple, adaptive blocksize limit.* 2016

[13] Yonatan Sompolinsky *Accelerating bitcoin's transaction processing fast money grows on trees, not chains.* 2014

[14] A. Toshi *The birthday paradox.* 2013