Kuantuma Dayanıklı Hesap Defteri (QRL)

peterwaterland@gmail.com

Kasım 2016

Özet

Özel dijital para birimleri, uzun ömürlülüğü sağlamak adına bilişim alanındaki gelişmelere karşı güvenli olmalıdır. Kripto para birimi hesap defterinin klasik ve kuantum hesaplama ataklarına dayanıklı olan karma(hash)-tabanlı dijital imzalar kullanılarak tasarımı ve ihracı sunulmuştur.

1 Giriş

Eşler arası bir internet hesap defteri kavramı, ilk kez 2008 yılında, bir kayıt zinciri olarak kaydedildi ve iş belgesi ile güvence altına alındığı raporlandı[11]. Bitcoin, bu zamana kadar en yaygın kullanılan kripto para birimi olarak kalmıştır. Sonrasında yüzlerce benzer kripto para birimi oluşturuldu fakat, birkaç istisna dışında hepsi, işlemlerin güvenli bir şekilde onaylanması için, sayısal imza oluşturmak için aynı eliptik eğim açık anahtar kriptosuna (ECDSA) dayanmaktadır. ECDSA, DSA ve RSA gibi yaygın şekilde kullanılan imza şemaları, teorik olarak kuantum hesaplama ataklarına karşı savunmasızdırlar. Ani doğrusal olmayan kuantum hesaplamaların potansiyel ilerleyişine karşı kuantuma dayanıklı kayıt zinciri hesap defteri tasarımı ve inşasının keşfi yararlı olacaktır.

2 Bitcoin İşlem Güvenliği

Şu anda sadece o belli bitcoin adresi için özel anahtardan ($x \in N | x < 2^{256}$) geçerli bir eliptik eğim (secp256k1) imzası içeren bir işlem oluşturarak bir bitcoin adresinden harcama yapmak (harcanmamış işlem çıktıları) mümkündür. Eğer gerçekten rastgele oluşturulan özel anahtar saklandıysa veya kaybedildiyse, bu adresten herhangi bir fonun taşınması beklenmez.
Belli bir bitcoin özel anahtarının çakışması olasılığı 2^{256} 'da 1'dir. Herhangi bir bitcoin adres anahtarı çakışması olasılığı Doğum günü Paradoksu ile tahmin edilebilir. %0.1 çakışma olasılığı sonucu için oluşturulması gereken bitcoin adresi sayısı 5.4×10^{23}'tür[14].

Bununla birlikte, bir işlem imzalandığında, gönderen adresin ECDSA açık anahtarı açığa çıkar ve kayıt zincirinde depolanır. Adresler için en iyi uygulama tekrar kullanılmamasıdır, fakat Kasım 2016 itibariyle, tüm bitcoin hesap defteri bakiyelerinin %49.58'i ifşa olmuş açık anahtarlara sahip adreslerde tutulmaktadır[1].

3 Kuantum Hesaplama Atak Vektörleri

RSA, DSA ve ECDSA, büyük tamsayıların çarpanlara ayrılmasının hesaplama zorluğu, ayrık logaritma problemi ve eliptik eğim ayrık logaritma problemine bağlı olarak güvenli durumdadırlar. Shor'un kuantum algoritması (1994) büyük tamsayıların çarpanlara ayrılmasını ve çokterimli zamanda ayrık logaritmalarını çözer. Bu nedenle, bir kuantum bilgisayarı, teorik olarak, bir ECDSA açık anahtarı verilmiş olan özel anahtarı yeniden oluşturabilir. ECDSA'nın RSA'ya göre daha kısa anahtar boyutu kullanımından dolayı kuantum atağa daha savunmasız olduğu düşünülür; 1300 ve 1600 qubit (2^{11}) bir kuantum bilgisayar 228 bit ECDSA'yı çözebilir.

Herkese açık kuantum bilgisayar gelişimi 2^5 qubitin veya küçük sayıların (15 veya 21) çarpanlara ayrımından ötesine geçememiştir. Fakat, Ağustos 2015'te, NSA, görünürde kuantum hesaplama konusuna dayanan eliptik eğri şifrelemesini kabul etmedi. İleri düzey kuantum hesaplamanın halen nasıl olabileceği veya bu alanın herhangi bir atılımının, kuantum sonrasını güvenli kılmak adına kriptolu protokollerin internette yaygın olarak kullanımını sağlamak için, nasıl halkın kullanımına sunulacağı açık değildir. Düzen karşıtı kökeni ile, bitcoin, kendisini kauntum

bilgisayarlı bir rakibin erken hedefi olarak bulabilir.

Eğer kuantum hesaplamada, açık olarak, belirgin bir ilerleme yakalanabilirse, düğüm (node) geliştiricileri, kuantum dayanıklı kriptolu imza şemalarını bitcoine entegre edebilirler ve tüm kullanıcıları bakiyelerini ECDSA tabanlı adreslerden yeni kuantum korunaklı adreslere taşıyabilirler. Etkilenen adreslerin oranını azaltmak için, protokol seviyesinde açık anahtarın yeniden dolaşıma sokulmasını devre dışı bırakmak mantıklı olacaktır. Böylesine planlı bir yükseltme, aynı zamanda Satoshi Nakamoto'ya ait olan 1 milyon coin'in –ilişkin fiyat dalgalanması ile birlikte- muhtemel hareketi ile de sonuçlanacaktır.

Daha az iyimser bir senaryo, sessizce doğrusal olmayan kuantum hesaplama ilerlemesini, ifşa edilmiş açık anahtarlara sahip bitcoin adreslerine ayrıntılı bir kuantum hesaplama atağının izlemesi olacaktır. Bu tür hırsızlıklar, ağır satış baskısı ve hırsızlıkların ölçüsünün bilinmesiyle sistemde tamamen güvenin kaybedilmesi nedeniyle, bitcoin kurunda yıkıcı bir etkiye sahip olacaktır. Bitcoin'in bir değer deposu ('dijital altın') olarak rolü, dünya için aşırı sonuçlarıyla birlikte çok kötü bir şekilde zarar görecektir. Bu bağlamda, yazarlar, kripto para biriminde, kuantum dayanıklı kriptolu imzalar ile denemeler yapılmasının ve kara kuğu durumunda potansiyel olarak yedek tasarruf hesabı oluşturmanın yararlı olacağı görüşündedirler.

4 Kuantuma dayanıklı imzalar

Kuantuma dayanıklı olduğu düşünülen birkaç önemli kriptolu sistem bulunmaktadır: karma(hash) tabanlı şifreleme, kod tabanlı şifreleme, kafes tabanlı şifreleme, çok değişkenli ikinci dereceden denklemler şifrelemesi ve gizli anahtar şifreleme. Tüm bu şablonların, verilen yeterli uzunluktaki anahtar boyutlarıyla ile hem klasik hem de kuantum hesaplama ataklarına dayanaklı oldukları düşünülür.

İleri güvenli karma tabanlı dijital imza şablonları, kriptolu karma fonksiyonlarının çakışmaya dayanıklı olmalarına dayanarak, en az güvenlik gereklilikleri ile bulunmaktadır. Seçili karma fonksiyonunu değiştirmek yeni bir karma tabanlı dijital imza şablonu oluşturur. Karma tabanlı dijital imzalar çok iyi çalışılmışlardır ve gelecekte kuantum sonrası imzalar için öncelikli aday konumundardırlar. Bu nedenle QRL için kuantum sonrası imzanın seçili sınıfıdırlar.

5 Karma tabanlı dijital imzalar

Kuantum dayanıklı karma tabanlı imzalar, *m* mesajını alan, *n* sabit bir uzunluğunda (ör. SHA-256, SHA-512), karma özütü *h*'ı çıktı olarak veren tek yönlü kriptolu karma fonksiyonu üzerine kurulmuşlardır. Kriptolu bir karma fonksiyonunun kullanımıyla, hesaplama olarak, kaba kuvvet (brute force) ile *h*'den *m*'nin eldesi (ön görüntü direnci) veya h_2'den *h*'ın eldesi, ki h_2=*hash(h)* (ikinci görüntü öncesi direnç) imkansız olmasının yanı sıra, aynı *h*'yi üreten iki mesaj ($m1 \neq m2$) bulmak çok zor olmalıdır (çakışma direnci).

Bir karma çakışması bulmak veya m'yi bulmak adına $O(2^{n/2})$ işlem gerektiren bir ön görüntü atağı düzenlemek için Grover'ın kuantum algoritması kullanılabilir. Bu nedenle, 128 bit güvenliği sağlamak adına, en az 256 bit, karma uzunluğu, *n* olarak seçilmelidir – mükemmel bir kriptolu karma fonksiyonu varsayılarak.

Karma tabanlı dijital imzalar, doğrulama için bir açık anahtar, *pk*, ve mesajı imzalama için bir gizli anahtar, *sk* gerektirir. Çeşitlkarma tabanlı tek kullanımlık imzaların (OTS),bir kayıtzinciri hesap defterinin parçası olarak dahil edilmesi yönünden uygunluğu tartışılacaktır.

5.1 Lamport-Diffie OTS

1979 yılında, Lamport, *m* bit uzunluğundaki bir mesaj için karma tabanlı tek kullanımlık bir imza tanımladı (genellikle bir çakışmaya dayanıklı karma fonksiyonunun çıktısı). Anahtar çifti oluşturma, rastgele *m* çift gizli anahtar oluşturur, $sk_j^m \in \{0,1\}^n$, $j \in \{0,1\}$, ör. özel anahtar: $sk = ((sk_0^1, sk_1^1), .., .., (sk_0, sk_1))$. f'nin, $pk_j = f(sk_j)$, şeklinde oluşturulmuş *m* çift açık anahtarıyla, ör. açık anahtar: $pk = ((pk_0^1, pk_1^1), .., .., (pk_0, pk_1))$, bir tek yönlü karma fonksiyonu olduğunu varsayalım, $\{0,1\}^n \rightarrow \{0,1\}^n$. İmzalama, sk_j 'yi seçmek için (ör. eğer $bit = 0$ ise, $sk_j = sk_0$, $bit = 1$, $sk_j = sk_1$), mesaj karmasının bit bazında kontrolünü, imzanın oluşturulmasını $s = (sk_j^1, .., sk_j)$ içerir ki bu özel anahtarın yarısını ortaya çıkarır. Bir imzanın doğrulamak için, mesaj karmasının bit bazında

($j \in \{0,1\}$) denetlemesi, $(pk = f(sk))^m$ olduğunu kontrol eder.

Grover algoritmasından sonra 128 bitlik güvenlik istenildiğini varsayarsak, mesaj uzunluğu SHA256'dan sabit karma çıktısıdır, m = 256 ve n = 256, bu da pk = sk =16kb sonucunu doğurur, ve her bir OTS için 8kb'lik imza kullanılmıştır. Bir Lamport imzası sadece bir kez kullanılmalıdır, çok hızlıca oluşturulabilir, fakat uzun anahtardan, imzadan ve uzantı işlem boyutlarından muzdariptir, ki bu onu herkese açık kayıtzinciri hesap defteri için elverişsiz yapar.

5.2 Winternitz OTS

n bit uzunluğunda gizli ve açık anahtarları, bir $f:\{0,1\}^n \rightarrow \{0,1\}^n$ tek yönlü fonksiyonu, ve bir $w \epsilon N | w > 1$ Winternitz parametresi ile, m bit uzunluğunda bir mesaj özütü M için, Winternitz tek kullanımlık imzasının genel fikri, bir rastgele gizli anahtarlar listesine, $sk \in \{0,1\}^n$, tekrarlayan karma fonksiyonu uygulamak, $sk = (sk_{,..,} sk_{m/w})$, w – 1 uzunluğunda, açık anahtarlarla ($pk \in M|\{0,1\}^n$) biten karma zincirleri oluşturmaktır $pk_x = f^{2^{w-1}}(sk_x), pk = (pk)$.

Lamport imzasındaki mesaj özütünden farklı olarak, $i \in N, i<2^w$ -1, şeklinde bir sayı ortaya çıkarmak için, mesaj, her seferde w bit kadar çözümlenir, bu kullanılarak imza oluşturulur, $s_x = f^i(sk_x), s = (s_1,..,s_{m/w})$. w değişkeninin büyümesi, daha küçük anahtarlar ve imzalar ile artan hesaplama eforu arasında takas yapmanızı sağlar[10].

Doğrulama, M, s 'den $pk_x = f^{2^{w-1}}(s_x)$ oluşturulmasını ve açık anahtarların eşleşmesinin onaylanmasını içerir.

Tek yönlü kriptolu karma fonksiyonu olarak, f: m=256 ve n = 256, w=8 iken, SHA-2 (SHA-256) kullanımı; $\frac{(m/w)n}{8}$ bayt = 1kb boyutunda pk = sk = s olarak sonuçlanır.

pk oluşturmak, her bir OTS anahtar çifti oluşturma başına, $i = \frac{m}{2}2^{w-1} = 8160$ olduğu, f^i karma iterasyonu gerektirir. w = 16'da, anahtarlar ve imzalar boyut olarak yarılanırlar fakat i =1048560 kullanışsız hale gelecektir.

5.3 Winternitz OTS+ (W-OTS+)

Buchmann, tekrarlayan (iterating) tek yönlü fonksiyonunu, tekrarlar şekilde bir x rastgele sayısına uygulanmasıyla değiştirerek orijinal Winternitz OTs'nin bir türünü ortaya çıkarmıştır fakat bu kez, $f_k(x)$'in önceki iterasyondan oluşturulan k anahtarı tarafından parametrelendirilmiştir. Adaptif seçili mesaj ataklarında sözde rastgele fonksiyon (PRF) kullanıldığında, bu, oldukça üstesinden gelinemez bir hal alır ve verilen parametrelerle bir güvenirlik hesaplanabilir[3]. Bu, basit bir iterasyon yerine fonksiyon üzerinde rastgele bir gözden geçirme düzenleyerek çakışmaya dayanıklı karma fonksiyon ailesine olan ihtiyacı ortadan kaldırır. Huelsing, eşdeğer *bit* güvenliği için, tekrarlayan zincirleme fonksiyonunda XOR bitmaskını ekleyerek daha küçük imzaların oluşumunu mümkün kılıp, bir çeşit daha ortaya çıkarmıştır, W-OTS+ [6]. W-OTS(2011 türü)/ W-OTS+ ile W-OTS arasındaki bir diğer farklılık, mesajın bir kerede w yerine $log_2(w)$ bit şeklinde çözümlenmesidir ki bu, karma fonksiyonu iterasyonlarını düşürürken anahtar ve imza boyutlarını arttırır.

W-OTS+ imzasını, kısaca açıklayalım. Güvenlik parametresi $n \in N$, karşılık gelen, mesaj (m), anahtarlar ve seçili kriptolu karma fonksiyonu ve $w \in N \mid w > 1$ (genellikle {4, 16}) Winternitz parametresi tarafından belirlenen bitler şeklinde imza ile l hesaplanır. $l = l_1 + l_2$ iken, l, WOTS+ anahtar veya imzasındaki n *bit* metin elemanı sayısıdır:

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

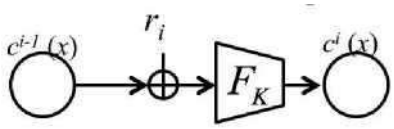
Kilitlenmiş bir karma fonksiyonu kullanılır, $f_k: \{0,1\}^n \rightarrow \{0,1\}^n | k \in \{0,1\}^n$. Yalancı kod ile;

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

$|x| < b$ *için*, $Pad(x) = (x||10^{b|x|1})$.

Zincirleyici fonksiyon, $c_k^i(x,r)$: $x \in \{0,1\}^n$ girdisinde, iterasyon sayacı i, anahtar k ve $j \geq i$ iken $r = (r_{,..,} r_j) \in \{0,1\}^{n*j}$ rastgeleleştirme (randomization) elemanları şu şekilde tanımlanır:

Şekil 1. W-OTS+ zincirleyici fonksiyon

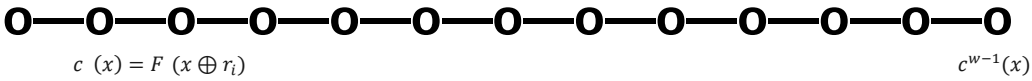


Burada:

$$c^i(x,r) = \begin{cases} x & \text{eğer i = 0;} \\ f\left(c^{i-1}(x,r) \oplus r_i\right) & \text{eğer i > 0;} \end{cases}$$

Şekil 2. Örnek karma-zinciri oluşumu

$\boldsymbol{c^0(x) = x}$



Önceki c_k iterasyonunun her bir bitinin xor edilmesi işlemini rastgeleleştirme (randomization) elemanı f_k takip eder, ve burdan çıkan sonuç sonrasında c_k 'nın sonraki iterasyonuna beslenir

5.3.1 İmza anahtarı

Gizli anahtar, *sk*'yı oluşturmak için, *l*+*w*-1, *n* bit karakter dizileri (PRF ile) rastgele homojen şekilde seçilir, bunlarla, *l*, ilk olarak gizli anahtarı oluşturur, $sk = (sk_1, .., sk_l)$ ve kalan *w*-1 bit karakter dizileri $r = (r_1, .., r_{w-1})$ haline gelir. Rastgele olarak homojen şekilde bir *k* fonksiyon anahtarı seçilir.

5.3.2 Doğrulama anahtarı

Açık anahtar:

$$pk = (pk_0, pk\ , .., pk_l) = ((r,k), c^{w-1}(sk\ , r), c^{w-1}(sk_2 r), .., c^{w-1}(sk_l, r))$$

pk_0 'ın r ve k içerdiğine dikkat ediniz.

5.3.3 İmzalama

Bir imza oluşturmak için: $M = \left(M_1, .. M_{l_1}\right), M_i \in \{0, w-1\}$ şeklinde, *m* uzunluğundaki *M* mesajı çözümlenir (M'nin w tabanında bir gösterimi oluşturuluyor).
Sonrasında, l_2 uzunluğundaki *C* sağlaması hesaplanır ve eklenir:

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

Öyle ki: M + C = b = $(b_0, .. b_l)$
İmza şu şekildedir:

$$s = (s\ , .., s_l) = (c^{b_i}(sk\ , r), .., c^{b_l}(sk_l, r))$$

5.3.4 Doğrulama

Bir imzayı doğrulamak için $b = (b_1,..,\ b_l)$, M kullanılarak tekrar oluşturulur.

Eğer $pk = (c^{w-1-b_1}(s_1),..,c^{w-1-b_l}(s_l))$ ise imza geçerlidir.

W-OTS+, en azından n-w-1-2$log(lw)$ bitlik güvenlik seviyesi sağlar[3]. SHA-256 (n=m=256) kullanılan, w=16 olan tipik bir imza, ln bit veya 2.1kb'lıktır.

6 Merkle ağacı imza şablonları

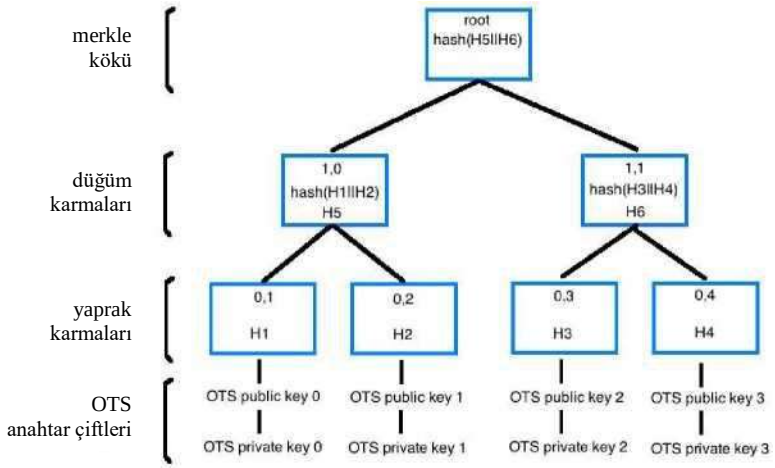
Tek kullanımlık imzalar hesap işlemlerini imzalamak ve doğrulamak için yeterli güvenliği sağlasa da, bir kere güvenli kullanılabilmeleri gibi büyük bir dezavantajları var. Eğer hesap defteri adresi, tekil OST anahtar çiftinin bazı dönüşümleri üzerine kurulu ise, bu durum, gönderen adresten tüm fonların gerçekleştirilen her bir tekil hesap işleminde taşınmasını gerektiren - veya bu fonların çalınma riski altında olmasına neden olan - oldukça kısıtlayıcı bir kayıtzinciri hesap defterine neden olacaktır.
Çözüm, her bir hesap defteri için birden fazla geçerli OTS imzası içerecek şekilde imza şemasını genişletmek ve böylece önceden oluşturulan OTS anahtar zincirleri kadar imzayı sağlamaktır. Merkle ağacı olarak bilinen bir ikili (binary) karma ağacı, bunu başarmak için mantıklı bir yoldur.

6.1 İkili karma ağacı

Merkle ağacının arkasındaki genel fikir, alt kardeş düğümlerden köke doğru ard arda karılarak (hashing) hesap edilen üst düğümlerden oluşan, ters çevrilmiş bir ağaçtır. Herhangi bir düğümün veya yaprağın varlığı kök hesaplanılarak kriptoyla kanıtlanabilir.

Bir merkle ağacı n taban yaprağından oluşur ve merkle köküne kadar boya sahiptir, h ($n = 2^h$) – yaprak karmalarından başlar (katman 0) ve her bir düğüm katmanıyla yukarı doğru sayılır. Her bir yaprak düğümü, rastgele olarak önceden oluşturulmuş bir OTS açık anahtarının karması alınarak bizim varsayımsal hesap defteri kullanımı senaryomuzda oluşturulur. Aşağıdaki ağaçta, her bir yaprak çiftinin üzerindeki düğüm, ardışık alt karmaların karması tarafından biçimlendirilmektedir

Şekil 3. Merkle ağacı imza şeması örneği



Bu, ağacın katmanlarda yukarılara doğru, ağacın merkle kökü olarak bilinen kök karmasında birleşene kadar devam eder.

Şekildeki örnek ağaca bakarsak, merkle kökünü alırken, açık anahtar olarak, dört önceden hesaplanmış OTS anahtar çifti, dört kriptoyla korunan geçerli tek kullanımlık imza oluşturmak için kullanılabilir. İkili karma ağacının merkle kökü hesap defteri adresine dönüştürülebilir (muhtemelen eklenmiş sağlama ile tekrarlayan karma kullanılarak).

Verilen bir OTS anahtar çifti için, bir *M* mesajına ait, tam bir *S* imzası şunları içerir: *s* imzası, *n* ots anahtar sayısı ve merkle yetkilendirme yolu ör. OTS anahtar çifti için 0 (bu nedenle *n* = 0):

*S=s, n, OTS açık anahtarı 0, H*1, *H*2, *H*5, *H*6, *kök*

OTS açık anahtarı verildiyse, yaprak karması *s*'den çekilebilir ve üst (parent) düğümler kendi alt öğelerinden hesaplanabilir, aslında bu, şu şekilde kısaltılabilir:

*S=s, n, H*2, *H*6, *kök*

Burada *S,* OTS açık anahtarının *s* ve *M* kullanılarak doğrulanmasıyla geçerli olur, sonrasında merkle yetkilendirme yolundan karmaları kontrol ederek, eşleşen bir merkle kökü yeniden oluşturulur (açık anahtar).

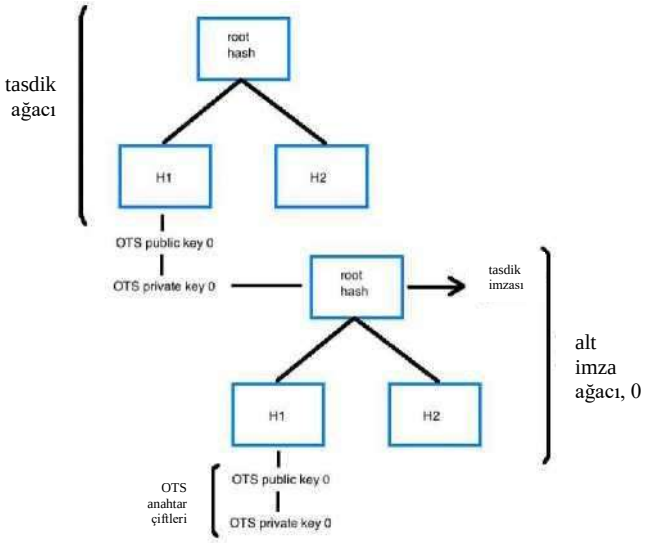
6.2 Durum

Yukarıdaki merkle imza şemasını (MSS) kullanmak, güvenli bir şekilde, OTS anahtarlarının tekrar kullanmamaya dayanır. Bu nedenle, bu, imzaların veya imzalanmış keydedilmiş hesap işlemlerinin durumuna bağlıdır. Genellikle, gerçek hayatta, bu potansiyle bir problem olacaktır ama sabit bir açık kayıtzinciri hesap defteri, geniş kapsamlı bir kriptolu imza şeması için ideal bir depolama ortamıdır. SPHINCS adında, 2^{128} bit güvenlikle uygulanması mümkün durum denetlemesiz imzalar öneren, yeni, karma tabanlı kriptolu bir imza şeması 2015'te rapor edilmiştir[2].

6.3 Hiperağaçlar

Temel MSS ile ilgili bir problem, muhtemel imza sayısının kısıtlı ve tüm OTS anahtar çiflerinin merkle ağacı hesaplanmadan önce, önceden oluşturulmuş olması gerekmesidir. Anahtar oluşturma ve imza zamanı, ağacın boyu *h* ile üssel olarak büyür, yani 256 OTS anahtar çiftinden daha büyük olan aağaçların geçici olarak ve hesaplama açısından üretimi pahalıdır.
Kendisi de merkle ağaçlarından oluşan hiperağaç denilen ağaçları kullanmak, anahtar ve ağaç oluşumu sırasında hesaplamayı ertelemek için ve ayrıca OTS anahtar çiftlerinin sayısını da arttırmak için uygulanabilir bir stratejidir. Genel fikir, alt ağacın merkle kökünü, tasdik ağacı denilen üst merkle ağacın yaprak karmasından gelen OTS anahtarı ile imzalamaktır.

Şekil 4. merkle ağaçlarını bağlama



En basit haliyle, (yükseklik, *h*=2) bir tasdik ağacı 2^1 OTS anahtar çifti ile önceden hesaplanır ve ilk imza gerektiğinde, yeni bir imza merkle ağacı (imza ağacı 0) hesaplanır ve tasdik ağacı OTS anahtar çiftlerinden birisiyle imzalanır. İmza ağacı, karşılık gelen OTS anahtar çiftleri ve *n* yaprak karmasından oluşur ve bunlar gerekli olduğunda mesajları imzalama işini görürler. İmza ağacındaki her bir OTS anahtar çifti kullanıldığında sonraki imza

ağacı (imza ağacı 1), tasdik ağacından gelen ikinci OTS anahtar çifti tarafından imzalanır ve sonraki bir grup imza kullanılabilir hale gelir.

Bu tür bir hiperağaç yapısının *S* imzası biraz daha karmaşık hale gelir ve şunları içerir:
1. imza ağacından: *s, n, merkle yolu, kök*
2. her bir tasdik ağacından: *s* (alt ağaç merkle köküne ait), *n, merkle yolu, kök*

Orijinal MSS'yi sonsuza kadar genişletmek için tasdik ağacından aşağı ağaç katmanlarına iç içe geçirmek teorik olarak mümkündür. Anahtar boyutu, imzalanan her bir ek ağaç için doğrusal olarak artar, bunun yanısıra, hiperağaç imza kapasitesi üssel olarak artar.

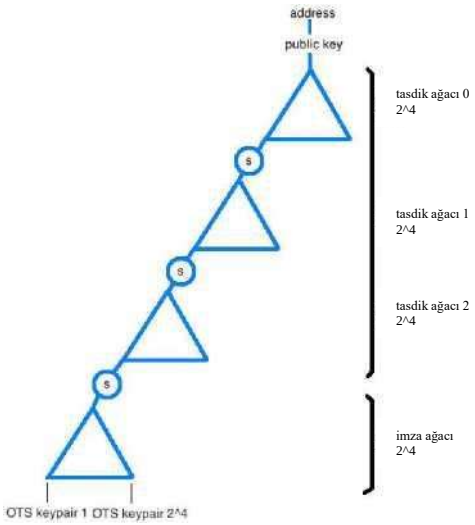
6.3.1 Hiperağaç örnekleri

MSS'nin bir hiperağaç yapısı ile nasıl kolayca genişlediğini göstermek için başlnagıç olarak $h_1 = 5$ yüksekliğinde, 2^5 yaprak karması ve ilgili OTS anahtar çiftleri ile bir tasdik ağacı düşünün. Bu ağacın merkle kökü hesap defteri adresi oluşturacak şekle dönüştürülmüştür. Bir başka merkle ağacı, aynı boyuttaki bir imza ağacı ($h_2 = 5$, 2^5 yaprak ve OTS anahtar çiftleri) somutlaştırıldı. Sonraki imza ağacının oluşturulmasından önce 32 imza kullanılabilirdir. Muhtemel imzaların toplam sayısı 2^{h1+h2} şeklindedir ki bu durumda $2^{10} = 1024$'tür.

Bir 2.6Ghz i5, 8gb ram Macbook pro'da OTS anahtar çiftleri ve çeşitli boyutlarda merkle tasdik ağacı oluşturmak şu sonuçları vermiştir (optimize edilmemiş python kodu, Winternitz OTS): 2^4 = 0.5sn, 2^5 = 1.2sn, 2^6 = 3.5sn, 2^8 = 15.5sn. 2^4 ağaç ilk oluşumunu içeren bir hiperağaç, aynı kapasitede 2^8 standart bir MSS ağacı için 15.5s ile kıyaslandığında 1s civarında zaman alır.

Bir hiperağacın derinliğinin (veya yüksekliğinin) arttırılması bu şekilde devam eder. Dört zincirlenmiş 2^4 tasdik ağacından ve 2^4 imza ağacından oluşan bir hiperağaç, imza boyutuna ek yük ile 2^{20} = 1,048,576 imza kapasitesine sahiptir fakat bunun yanında oluşturma zamanı sadece 2.5sn'dir.

Şekil 5. Hiperağaç yapısı

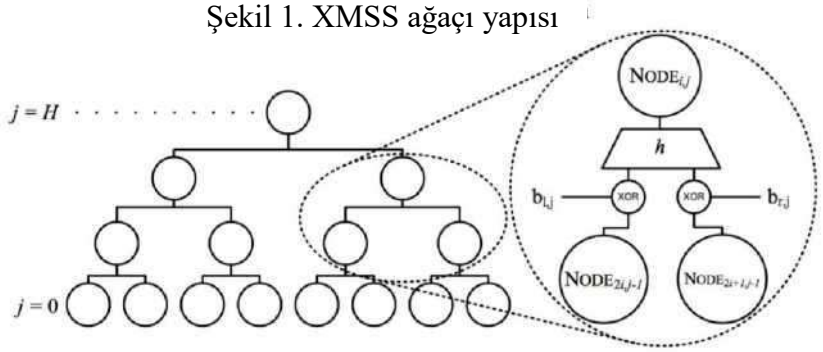


Bir hiperağacın simetrik olması gerekliliği yoktur ve böylece başlangıç olarak iki ağaçtan oluşuyorsa, sonradan daha fazla ağaç katmanı imzalanarak genişletilebilir. Hesap defteri adresinden gelen imzalar, bu nedenle küçük olarak başlarlar ve zamanla hiperağaç derinliği arttıkça büyürler. Hesap defteri adresinden gelen hesap işlemlerini oluşturmak ve imzalamak için merkle hiperağacını kullanmak, $>2^{12}$ hesap işlemi asla gerektirmeyecektir. Bu nedenle, hesaplama kolaylığı ile imzalama yeteneği, 2^{20} kez daha güvenli, $h = 5$ hiperağaç derinliğinde fazlasıyla yeterlidir.

6.4 XMSS

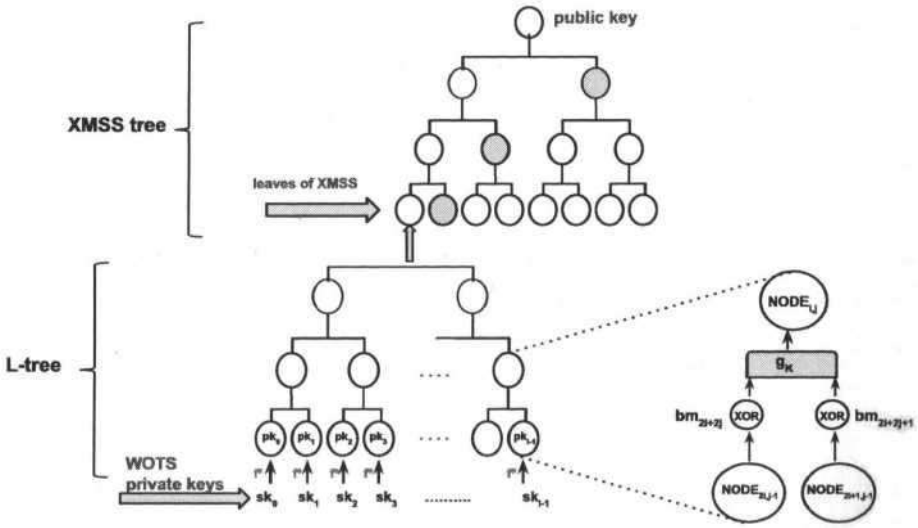
Genişletilmiş merkle imza şeması (XMSS), Buchmann ve diğerleri tarafından 2011'de ilk olarak rapor edilmiş ve sonraki yıl IETF taslağı olarak yayınlanmıştır[4][7]. Kanıtlanabilir derecede, ileri güvenlidir ve varoluşsal olarak seçili mesaj ataklarında en az güvenlik gereklilikleriyle üstesinden gelinemezdir: bir PRF ve ikinci ön görüntü dayanıklı karma fonksiyonu. Şema, tek kullanımlık imzaların bir merkle ağacı yoluyla genişletilmesini sağlar fakat karmaların üst düğümde birleşmesinden önce alt düğümlerin XOR bitmaskının kullanımı gibi büyük bir farkla bunu yapar. XOR bit maskesinin kullanımı çakışmaya dayanıklı (çakışma olmadan) karma ailesinin değiştirilmesini sağlar.

Şekil 1. XMSS ağaçı yapısı



Ağacın yaprakları da OTS anahtar çifti karması değildirler fakat OTS açık anahtarlarını *l* parça olarak tutan alt L-ağaçlarının kökü taban yapraklarını biçimlendirir. Tek kullanımlık imzalar için Winternitz OTS+ kullanılır. (2011 türü ilk olarak tanıtılmasına rağmen).

Şekil 7. XMSS yapısı [8]



XMSS açık anahtarının bit uzunluğu $(2(H+[logl])+1)n$, bir XMSS imza $(l+H)n$ uzunluğundadır ve XMSS gizli anahtarının uzunluğu $<2n$.

Buchmann, bir Intel(R) *i*5 2.5Ghz ile, h=20 yüksekliğinde, w=16 olarak kullanılan karma fonksiyonunun SHA-256 (m=256) olduğu bir XMSS ağacı için, performasının, yaklaşık bir milyon imzaya kadar olduğunu rapor etmiştir. Aynı parametreler ve donanım ile imzalama 7ms, doğrulama 0.52ms ve anahtar oluşturma 466 saniye sürmüştür. Bu parametrelerle elde edilen güvenlik seviyesi, 1.7kb boyutunda bir açık anahtar, 280 bitlik özel anahtar ve 2.8kb imza için 196 bittir. XMSS ilgi çekici bir şemadır fakat temel dezavantajı uzun anahtar oluşturma süresidir.

6.5 XMSS ağacı performansı

QRL test düğümü için inşa edilmiş optimize edilmemiş bir python kütüphanesi kullanarak, karma tabanlı PRF kullanılarak oluşturulmuş tüm anahtar ve bit maskeleri ile, 4096 yaprak XMSS ağacının (h=12) biçimlendirilmesi yukarıda açıklanmış olan donanımın aynısıyla (bir Macbook pro 2.7Ghz i5, 8gb ram) 32sn almıştır. Buna PRF yoluyla 8000'in üzerinde bitmaskesi ve 300,000'in üzerinde sk (gizli anahtar) bölümü oluşturma dahildir. Daha verimli bir aykırı merkle algoritması, WOTS ile 2^{w-1} yerine WOTS+'da gizli anahtar zinciri başına sadece w-1 karma gerekliliği, geleneksel MSS'e karşılık etkileyici bir performans gelişimine katkıda bulunuyor.

Bu yapıda 5.75kb civarı boyutunda imza gerçekleştirilmiştir (11.75kb onaltılık düzende dizin şifreleme) şunları içermektedir: OTS anahtar çifti, imza *n,* XMSS yetkilendirme yolu, OTS açık anahtar ve XMSS ağacı açık anahtarı (PRF açık anahtarı tohumu ve XMSS ağacı kökü dahil).

PRF ve rastgele tohum kullanılarak oluşturulan çeşitli imza kapasitesindeki ağaçlar için, şu performanslar elde edilmiştir: (h=9) 512 4.2sn, (h=10) 1024 8.2sn, (h=11) 2048 16.02sn.

7 Önerilen imza şeması

7.1 Güvenlik gereklilikleri

QRL tasarımında, imza şemasının kriptografik güvenliğinin hem bugün hem de gelecek yıllar için klasik ve kuantum hesaplama ataklarına karşı güvenli olması önemlidir. SHA-256'nın kullanıldığı, w =16 olan, XMSS, kaba kuvvet hesaplama ataklarına karşı tahmin edilebilir güvenlikle, 2164 yılına kadar 196 bit güvenlik sağlar[9].

7.2 QRL imzalar

Zincirlenmiş XMSS ağaçlarından oluşan genişletilebilir, durum denetimli, asimetrik hiperağaç imza şeması önerilmektedir. Bu, doğrulanmış imza şemasının uygulanması ve büyük XMSS yapılarında görülen uzun ön hesaplama gecikmelerine takılmadan hesap işlemlerini imzalama yeteneğiyle hesap defteri adreslerinin oluşturulmasının sağlanması şeklinde çifte yarar sağlar. W-OTS+, hem güvenlik hem de performans nedeniyle, şemada, karma tabanlı tek kullanımlık imza olarak seçilmiştir.

7.3 Hiperağaç yapısı

7.3.1 Anahtar ve imza boyutları

Bir hiperağaçta ağaç sayısı arttıkça, anahtar ve imza boyutları da doğru orantıda artar – ama imza kapasitesi üssel olarak artar. w =16, m =256, h ağaç yüksekliği ve SHA-256 şifreleme karma algoritması olarak seçili iken, çeşitli XMSS ağacı türevli açık anahtar ve imzalar (2011 açıklamasını baz alarak) şu şekildedir:

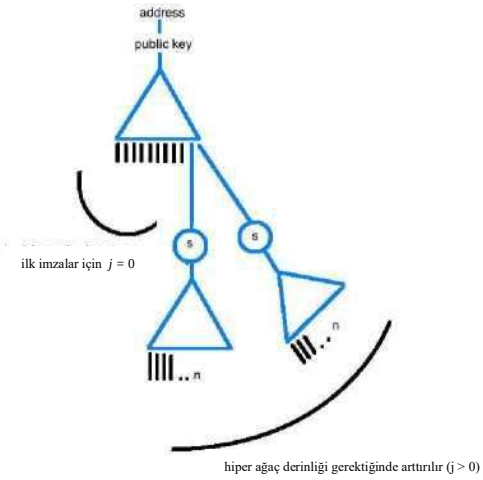
- h = 2, 2^2 imza: açık anahtar 0.59kb, imza 2.12kb (0.4sn)
- h = 5, 2^5 imza: açık anahtar 0.78kb, imza 2.21kb (0.6sn)
- h =12, 2^{12} imza: açık anahtar 1.23kb, imza 2.43kb (32sn)
- h = 20, 2^{20} imza: açık anahtar 1.7kb, imza 2.69kb (466sn[3])

2.69kb imze yerine 8.84kb imza için, 466sn'ye kıyasla 3sn'den kısa sürede 2^{20} muhtemel imza kapasitesi ile XMSS hiperağacı oluşturulması (4 ağaç, j = 3, h = 5) kabul edilebilir.

7.3.2 Asimetri

Bir asimetrik ağaç oluşturmak, ilk imzaların tek bir XMSS ağaç yapısı ile oluşmasına olanak tanır ki bu ortalama bir imza kapasitesi karşılığında sonraki imzalar için gerektiğinde genişletilebilir. Bunun mantığı, bunun bir kayıtzinciri hesap defteri uygulaması için bir sonuç doğurmayacağı ve cüzdanın kullanıcıya imza kapasitesine karşılık imza/anahtar boyutu şeklinde seçenek sunabileceğidir. Maksimum ağaç derinliği olarak j =2, her türlü durumda yeterli olacaktır.

Şekil 8. Asimetrik hiperağaç



7.4 QRL hiperağaç özellikleri

Standart bir hiperağaç yapısı için şu öntanımlı parametreler kabul edilmelidir.

•$j = 0$ $(j \in \{0 \leq x \leq 2\})$, $h = 12$ $(h \in \{1 \leq x \leq 14\})$, imzanın mümkün üst sınırı: 2^{36}, minimum imza boyutu: 2.21kb, maksimum imza boyutu: 7.65kb.

ör. Kullanılabilir 4096 imza ile h=12 olan, gerektiğinde h = 14'e kadar ek ağaçlar ile genişletilebilen tekil bir XMSS ağacı. Çoğu kullanıcı için ek ağaçların gerekmesi muhtemel değildir.

7.4.1 Örnek QRL imzası

j =2, h = 14, m işlem mesajı için imza, n her bir XMSS ağacı için OTS anahtar çifti konumu ise, en karmaşık hiperağaç kurulumu şunları gerektirecektir:

- İmza ağacı, j = 2: m OTS imzası, n, merkle yetkilendirme sağlaması, imza ağacının merkle kökü
- Tasdik ağacı, j = 1: imza ağacından merkle kökünün OTS imzası (j = 2), n, merkle yetkilendirme sağlaması, merkle kökü
- Orijinal XMSS ağacı, j = 0: merkle kökünün OTS imzası (j = 1), n, merkle yetkilendirme sağlaması, merkle kökü

Doğrulama, OTS açık anahtarının m ve imza kullanılarak oluşturulmasını, sonrasında sağlanan merkle yetkilendirme sağlamasının imza ağacı merkle kökünü oluşturmasını içerir. Bu sonraki OTS imzası için mesaj olur ve bundan sonraki OTS açık anahtarı oluşturulur, sağlanan merkle yetkilendirme sağlaması tasdik ağacı merkle kökünün tekrar oluşturulmasında kullanılır ki bu sonraki tasdik ağacı OTS imzası için mesaj olarak kullanılır ve bu böyle devam eder. İmza, sadece en yüksek ağacın merkle kökü, orijinal XMSS ağacı, (j =0) doğru olarak oluşturulduysa geçerlidir. XMSS ağaç imzasının doğrulaması için OTS açık anahtarlarının gerekli olmadığına dikkat ediniz. Aslında her bir ağaç için merkle kökü de çıkarılabilir ve bu nedenle eğer gönderen hesap defteri adresi biliniyorsa hiperağaç imza doğrulama ile hariç bırakılabilir(çünkü bu, en yüksek XMSS tasdik ağacı (j = 0) için QRL imzası içerisinde, merkle kökünün hesaplanmış türevidir – bkz. Hesaplar).

İmza şeması durum denetimli olduğu için, cüzdan uygulaması devam etmeli ve verilen bir adres için hiperağaçta oluşturulan her bir XMSS ağacı için n güncellenmelidir.

7.5 PRF

Tohumdan PRF. HMAC_DRBG.

7.6 Belirlenimci Cüzdan

Tekil bir TOHUM kullanarak, çok geniş bir XMSS ağacı oluşturmak mümkündür, ki bu uzun bir peryot için çoğu kullanıcı için yetecektir. Güvenli bir entropi kaynağı, ağacı oluşturan bir grup sözde rastgele anahtar oluşturmak için güvenli bir PRF fonksiyonundan geçen bu TOHUM'u oluşturmak için kullanılır. Aynı XMSS ağacını kullanmanın dezavantajlarından birisi de, kullanıcının tekil bir adrese sınırlandırılmış olmasıdır (açık anahtar teşhiri MSS'te bir sorun olmamasına rağmen).
Bir bitcoin veya ethereum adresi, ilgili açık anahtardan türetilir ve böylece tekil bir gizi veya açık anahtar sadece bir tek adres oluşturabilir. Fakat, bir XMSS adresi, merkle kökü ve açık TOHUM içeren, PK açık anahtarından türetilir. Eğer TOHUM sabit, fakat ağaçları hesaplamak için OTS anahtar çifti sayısı değişken ise, her bir varyasyon için merkle kökü değişecektir. Bu nedenle, her bir tekil OTS anahtar çifti eklemesi veya çıkarmasında türetilen adres değişecektir.
Bu özellik, XMSS ağacının sayısız varyasyonunu oluşturmak için cüzdan/düğüm yazılımı tarafından kullanılabilir (aynı başlangıç TOHUMunu kullanarak gerektiğinde genişleterek/daraltarak), oluşturulması gerektiği kadar çok benzersiz adrese izin verir. Bu bilgiyi, güvenli, kapsamlı ve derli toplu bir şekilde kaydetmek hesaplama olarak değersizdir.

8 Kripto para birimi tasarım parametreleri

Makalenin kalan kısmı, önerilen tasarım parametrelerini QRL hesap defteri için düzenlemek üzerinedir. Hesap defterinin odak noktası, klasik ve kuantum hesaplama atak vektörlerine karşı oldukça güvenli, herkese açık bir kayıtzinciri olmaktır. Bu ilk taslaktır ve bu nedenle her bir görüş, potansiyel olarak değişime açıktır.

8.1 Hisse- bazlı-sağlama (Proof-of-Stake, PoS)

QRL, proof-of-stake algoritması tarafından korunan, açık ortak bir kayıtzinciri hesap defteri olmalıdır. Bir dönem 10,000 blok sürer. Hisse doğrulayıcıları önceki dönemdeki hisse işlemlerinden belirlenir. Genel fikir şu şekildedir, her bir hisse doğrulayıcı, 10,000 karma uzunluğundan oluşan tekrarlayan bir zincirin son karmasını içeren bir hesap işlemi imzalar (bir XOR bitmaskesi, karma fonksiyonunun güvenlik gerekliliklerini düşürmek adına her bir tekrar sırasında uygulanabilir). Hisse işleminin zincirde onaylanması ile, artık, ağdaki her bir düğüm, sonraki dönem için karma zincirine, hisse adresinin kriptolu kimliğini bağlayabilirler.

8.1.1 Tasarım ve rastgelelik

Her bir blok için, şu anki döneme katılan her bir doğrulayıcı düğüm, katılımı kriptoyla kanıtlamak için sonraki peş peşe gelen, zincirdeki önceki karmayı ortaya çıkarır ve kazanan blok seçicisi olmak için oylar.

HMAC_DRBG, kayıtzincirinden alınan tohum verisinden 32 bit çıktılı benzetik (pseudo) rastgele bir sayı dizisi oluşturmak için kullanılır (ilk başta başlangıç (genesis) bloğu, sonrasında her bir takip eden dönem için, birleştirilmiş en son blok başlık karmalarından alınan enropi eklenir).

Böylece her bir blokta, o blok için PRF çıktısına sayı olarak en yakın karmayı ortaya çıkaran, hisse doğrulayıcı blok seçicisi olmak için seçilir. Bu, oynamak için çok zordur, çünkü karma zinciri oluşumu sırasında hisseler tarafından PRF bilinmemektedir. Dahası, bir yinelemeli (şifreli) karma zinciri, temel olarak bir rastgele sayı dizisidir. Son olarak, hisse doğrulayıcılar buna göz yumsa bile, diğer hisse doğrulayıcı karma zinciri içeriklerini bilemezler çünkü onlar daha ortaya çıkmamışlardır.

Bir blok kesintisi atağını önlemek için, ki geçerli bir açıklama karması kaydı sonrasında geçerli bir blok üretme başarısızlığı o adresten gelen tüm blok ödülünün kaybı ile ilgilidir, sonrasında bir ceza periyodu için katılımdan atılırlar.

Sybil düğümlerinden veya düşük hesap bakiyelerine sahip hisse doğrulama adreslerinden gelen bir boş blok atağını hafifletmek için esnek hisse doğrulayıcı liste eşiği kullanılmaktadır. Blok ödülü ağırlıklandırılmış bir şekilde ödenmektedir, hisse adres bakiyesini baz almaktadır. Açıklama karma mesajıyla, her bir düğüm ayrıca, hesap işlemi havuzlarında, bir blok için bekleyen bir miktar hesap işlemi ile birlikte sıralanmış bir tx karmalar (txhash) listesinin

bir merkle ağacı kökünü deşifre ederler. Her bir düğüm, sonraki blokta ne kadar işlem beklendiğini görmek için üstten ve alttan bir yüzde alır. Blok boşsa veya beklenenden azsa, bir miktar hisse doğrulayıcılarının (fakirden zengine hisse doğrulayıcılar hariç) her blokta, sözleşmelere yukarı doğru katılmalarına izin verilir. Eğer blok seçici düğümler dürüstçe davranıyorsa, sonrasında tersi doğrudur ve izin verilen, katılan hisse doğrulayıcı sayısı artar. Katıldıkları dönemde fonlar hareket etmeyebilir – bu, birçok hisse doğrulama adresinin sybil oluşumu ile blok seçimine hile karıştırmayı engeller.

8.2 Ücretler

Diğer hesap defterlerine kıyasla, daha büyük hesap işlemi boyutları, her bir hesap işleminde bir hesap işlemi ücretinin ödenmesini gerekli kılar. Yazar, yapay ücretli piyasaların (bkz. bitcoin) gereksiz olduğu ve açık bir kamu kayıtzinciri hesap defteri idealine ters düştüğü görüşündedir. Minimum ücreti öderse, her bir hesap işlemi diğerleri kadar geçerli olmalıdır. Madencilerin kabul etmeye istekli oldukları asgari ücret, pazar tarafından belirlenmeli ve hazırlanmalıdır. ör. düğümler / madenciler ücretlerin alt sınırını kendi aralarında rekabetçi bir şekilde belirler. Protokol seviyesinde mutlak minimum bir değer uygulanır. Böylece, madenciler takdir yetkisi dahilinde bir bloğa dahil etmek için hafıza havuzundan (mempool) hesap işlemleri emri vereceklerdir.

8.3 Bloklar

8.3.1 Blok süreleri

Bitcoin, bloklar arasında yaklaşık 10 dakikalık bir süreye sahiptir, ancak doğal varyasyonla, bu, zaman zaman, bir sonraki bloğun maden çıkarılması öncesinde oldukça uzun sürelere yol açabilir. Ethereum gibi daha yeni hesap defteri tasarımları, bu konuda gelişmişlerdir ve güvenlik kaybı ya da yüksek oranda geçersiz/bayat bloklardan gelen madenci merkezileştirilmesi olmaksızın, çok daha kısa bir blok süresinden (15 saniye) yararlanırlar. Ethereum, geçersiz/bayat blokların kayıtzincirinde dahil edilmelerini ve ödüllendirilmelerini sağlayan Greedy Heaviest Observed Subtree protokolünün değiştirilmiş bir sürümünü kullanır[13, 5].

QRL, baştan beridir, hisse-bazlı-sağlama (PoS) algoritması kullanmayı planladığından, 15 ila 30 saniye arasında bir blok süresini güvenli bir şekilde kullanmayı umuyoruz.

8.3.2 Blok ödülleri

Yaratılan her yeni blok, madencilik adresi içeren bir ilk ‘coinbase’ hesap işlemini içerecektir ki burada ödül, coinbase ödülü ile blok içerisindeki hesap işlemi ücretlerinin birleştirilmiş toplamının toplamına eşittir. Blok ödülü, blok seçici olarak seçilen hisse dorulama adresinin bakiyesi baz alınarak ağırlıklandırılacaktır.

Blok ödülü, madencilik düğümü tarafından her blokta yeniden hesaplanır ve kripto para (coin) emisyon çizelgesini takip eder.

8.3.3 Blok boyutu

Uyuşmazlığı önlemek için, kayıtzinciri boyutunu arttırmak için, bitpay önerisi üzerine modellenen ezber bozan uyarlanır çözümü, son *z* bloğun *y* ortanca boyutunu çoklu *x* baz alarak kullanılacaktır[12]. Ortanca kullanımı madencilerin ortalama blok boyutunu değiştirmek için, boş veya aşırı doldurulmuş bloklar içerme hilelerini engeller. Sonrasında *x* ve *z,* ağ için uyulması gereken mutabık kalınmış konsensüs kuralları olacaklardır.

Böylece, *b* maksimum bir blok boyutu, basitçe aşağıdaki gibi hesaplanabilir:

$$b = xy$$

8.4 Para birimi ve ölçüleri

QRL, temel para birimi olarak parasal belirteç, *kuantum* (çoğul *kuantumlar*) kullanacaktır. Her *kuantum,* en küçük elemente şu şekilde bölünür:

- 1 : Şor (Shor)
- 10^3 : Nakamoto
- 10^6 : Buterin
- 10^{10} : Merkle
- 10^{13} : Lamport
- 10^{16} : Kuantum

Böylece, bir miktar kuantum içeren her bir hesap işlemi, aslında *Şor* biriminde çok büyük bir tamsayıdır. Hesap işlemleri *Şor* biriminde hesaplanır ve ödenir.

8.5 Hesaplar

Kullanıcı bakiyeleri hesaplarda tutulur. Her hesap, 'Q' ile başlayan bir dizge tarafından tanımlanan benzersiz bir tekrar kullanılabilir hesap defteri adresidir.

Bir adres, en yüksek XMSS tasdik ağacının merkle kökü üzerinde bir SHA-256 uygulanmasıyla oluşturulur. Buna dört bitlik bir sağlama eklenir (merkle kökünün çifte SHA-256 karmasının ilk dört baytından düzenlenir) ve 'Q' harfi başına eklenir. ör. pythonik benzetik kodda:

Q + sha256(merklejroot) + sha256(sha256(merkle_root))[:4]

Tipik bir hesap adresi:

Qcea29b1402248d53469e352de662923986f3a94cf 0f*51522bedd08fb5e64948af 479*

Her hesap, *kuantumlar* olarak gösterilen, tekil bir *Şor* birimine bölünerek indirgenebilen bir bakiyeye sahiptir.

Adresler, her hesap işleminin taze bir OTS anahtar çifti kullandığı ve QRL'in her bir hesap için kullanılan tüm açık anahtarları depoladığı biçimde (Bu, kısaltılabilir, çünkü hesap işlemi imzası ve mesajdan çalışır haldeyken yeniden oluşturulabilir, ancak operasyonel olarak yoğun olabilir) durum denetimlidirler. Nonce adındaki hesap işlemi sayacı hesaptan her işlem gönderildiğinde arttırılır. Bu, tüm bir kayıt zincirini depolamayan cüzdanların, durum denetimli merkle hiperağacı imza şemasındaki konumlarını takip etmelerini sağlar.

8.6 Coin ihracı

8.6.1 Tarihi faktörler

Bitcoin ilk merkezi olmayan kripto para birimidir ve başlangıçta parasal değeri olmadan deneyseldi, bu nedenle, paranın tamamen madencilikle dağıtımına uygundu. Daha yakın zamanda, Zcash, emisyon erken döneminde coinbase madenciliği ödülünün % 'si ile açık kaynak projesine geçirilen aynı işlemi seçti – inanılmaz derecede fiyat dalgalanması sonucu ile.

Etherum gibi diğer hesap defterleri, bunun yerine, başlangıç coin teklifinin (ICO) bir parçası olarak, son coin desteğinin büyük bir %'sini sattı. Bu, erken benimseyenlerin hala projeyi destekleyerek potansiyel olarak kazanç sağlamaları yanında, projenin kendisinin de kalkınma ve önyüklemeyi sürdürmek ve projeyi bebeklik çağından itibaren büyütmek için fon yaratması açısından yararlara sahiptir. ICO yaklaşımı, ayrıca, daha büyük miktarda coininin yatırımcıların kullanımında, başlangıç bloğundan satın alınıp satılmasını sağlayarak piyasanın kolayca gelişmesini sağlar.

Auroracoin (2014), geliştiricilerin tüm coin tedariğinin %50'sini kendileri için aldıkları, İzlanda'daki herkese ICO'dan eşit hisse verdikleri farklı bir yaklaşımı benimsediler.

Diğer kripto para birimleri, ya basitçe bitcoini tamamen kopyaladılar, ya da farklı bir kod tabanıyla, yeni bir zincir ile başladılar.

8.6.2 Zincirlerarası bakiye transferi

QRL'yi, QRL başlangıç bloğuna eklenen şu anki bitcoin hesap defterine ait bir durum anlık görüntüsüne dayanarak tedavüle çıkarmak mümkündür. Genel fikir, kullanıcıların, benzersiz bir mesaj ve imza içeren, tek kullanımlık bir hesap defteri 'içe aktarım'ı oluşturmalarını sağlamak olacaktır (ör. anlık görüntü sırasında, bir bitcoin bakiyesine sahip bir adresten bitcoin özel anahtarı ile imzalanmış rastgele oluşturulmuş bir QRL cüzdan adresi). Bu özellik belirli bir blok yüksekliğine kadar aktif kalabilir ve sonrasında geri kalan coin tedariği normal şekilde madenden çıkarılabilir. Genesis bloğundaki ilk şişirme, aynı blok yüksekliğinde budanabilir. Bunun dezavantajı, adil olmakla birlikte, bitcoin haricindeki diğer kripto para birimlerine sahip olanları cezalandırır ve teknik olarak potansiyel olarak yeni kullanıcıların icrası zorludur. Sadece imza ve mesajdan ECDSA açık anahtarının bulunmasının mümkün olması teknik olarak bir sorun olabilir. Bu, kalıcı olarak, açık anahtarı, süreçte kullanılan bitcoin adreslerine ifşa edecektir ve sonrasında fonları yeni rastgele oluşturulmuş bir bitcoin adresine aktarmak bu zararı azaltmak adına önemlidir.

(?ethereum sahipleri için de aynı özellik sağlanabilir)

8.6.3 Önerilen ihraç - taslak

QRL başlangıç ihracı şu şekilde olacaktır:

- Faaliyete geçmeden önce 1 milyon *kuantum*luk (son kuantum tedariğinin %4.7'si) ICO.
- Başlangıç QRL bloğunu biçimlendirmek için kullanılan 0.01btc üzerindeki tüm bitcoin adres bakiyelerinin durum anlık görüntüsü. Bitcoin hesap defterinden QRL hesap defterine 1:1 oranında doğrudan coin transferi yapmak isteyenler bunu, düğüm cüzdanı yoluyla, 518400 (3-6 ay) blok yüksekliğine kadar, yapabileceklerdir.
- Kurum tarafından kullanılmak üzere bir başlangıç bloğu adresinde 1 milyon *kuantum* daha tutulacak
- Geriye kalan arz, çıkarılacak (21,000,000- (518400 blok yüksekliği ile içe aktarılan 2,000,000+ btc bakiyesi))

8.7 Coin emisyon takvimi

Bitcoinin belirleyici özelliği, altta yatan parasal simgenin yayımlanmasın yönelik kıtlık ve sabit üst sınırdır. Bu anlamda, QRL, coin arzına $21x10^6$ *kuantum* sabit bir üst sınır ile bitcoini takip edecek. Coin arzının zorlu tavanına kadar, blok ödüllerinde düzgün olarak üstel bir bozulma var. Bu, bitcoin 'yarılanma' konusuyla ilgili dalgalanmayı yok edecektir.

Toplam coin arzı, $x = 21x10^6$, eksi y, başlangıç bloğunda oluşturulan coinler, daima Z_0 'dan üssel olarak düşecektir. Bozunma eğrisi, madencilik ödüllerinin, yaklaşık 200 sene için (MS2217'ye kadar, 15sn blok süresinde 4204480000 blok), sadece bir tekil *kuantum* çıkarılmadan kalıncaya kadar dağıtımı için hesaplanır (madencilik bundan sonra da devam ederse).

t bloğunda kalan coin arzı, Z_t şu şekilde hesaplanabilir:

$$Zt = Zoe^{-\lambda t}$$

λ sabiti, şu şekilde hesaplanır: $\lambda = \frac{lnZ_0}{t}$. Burada, t,son *kuantum*a kadar emisyon takvimindeki toplam blok sayısıdır. 518400 bloğuna kadar, $\lambda = 3.98590885111x10^{-08}$. Blok ödülü, b ise şu şekilde hesaplanır:

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

Başlangıç bloğu ile 518400 bloğu arasında, hesap işlemi içe aktarma yoluyla bitcoin bakiyeleri hesap defterine aktarılabilir. 518400 bloğunda emisyon takvimi yeni içe aktarılan bakiyeleri kapatarak, Z_t 'yi düşürmeyi ve buna bağlı olarak blok ödülünü ayarlamayı yeniden hedefleyecektir.

Referanslar

[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.