*Uluslararası İleri Doğa Bilimleri ve Mühendislik Araştırmaları Dergisi*
*Sayı 8, S. 303-310, 11, 2024*

***Araştırma Makalesi***

IJANSER
https://as-proceeding.com/index.php/ijanser
ISSN:2980-0811

*International Journal of Advanced Natural Sciences and Engineering Researches*
*Volume 8, pp. 303-310, 11, 2024*

***Research Article***

# Kuantum Dayanıklı Kriptografik Algoritmaların Performans Analizi

Samet TONYALI[1*]

[1]*Yazılım Mühendisliği, Mühendislik ve Doğa Bilimleri Fakültesi, Gümüşhane Üniversitesi, Türkiye*

[*]*(samet.tonyali@gumushane.edu.tr)*

*(Received: 04 December 2024, Accepted: 06 December 2024)*

(3rd International Conference on Recent Academic Studies ICRAS 2024, December 03-04, 2024)

**ATIF/REFERENCE:** Tonyalı, S. (2024). Kuantum Dayanıklı Kriptografik Algoritmaların Performans Analizi. *International Journal of Advanced Natural Sciences and Engineering Researches*, 8(11), 303-310.

***Özet*** **–** Amerikalı matematikçi Peter Shor 1994'te geliştirmiş olduğu algoritma ile şifrekıran kuantum bilgisayarların da yardımıyla ayrık logaritma ve çarpanlara ayırma problemlerinin polinom zamanda çözülebileceğini göstermiştir. Bugün İnternet, güvenliği bu iki problemin zorluğuna dayanan Diffie-Hellman anahtar kurulum algoritmasına ve RSA kriptosistemine dayanmaktadır. Her ne kadar şifrekıran kuantum bilgisayarlar henüz geliştirilememiş olsa da kötü niyetli İnternet kullanıcıları *şimdi depola, sonra deşifrele* yaklaşımını kullanarak İnternet trafiğini depolayıp bu bilgisayarlar geliştirildiğinde onlardan faydalanarak şifrelenmiş mesajların içeriğine erişebileceklerdir. NIST bunu önlemek amacıyla 2016 yılında bir standartlaştırma süreci başlatmıştır. Tüm ilgilileri şifrekıran kuantum bilgisayarlara dayanıklı açık anahtarlı şifreleme ve anahtar kurulum algoritmaları ile dijital imza algoritmaları geliştirip bu sürece katkıda bulunmaya davet etmiştir. NIST üç değerlendirme turunun ardından 2022'de standartlaştırmaya karar verdiği dört algoritmayı açıkladı: Kyber, Dilithium, SPHINCS+ ve Falcon. 2024'te bu algoritmaların ilk üçü sırasıyla ML-KEM, ML-DSA ve SLH-DSA isimleriyle ve yine sırasıyla FIPS 203, FIPS 204 ve FIPS 205 kodlarıyla standartlaştırılmıştır. Falcon'un standartlaştırılması ileri bir tarihe ertelenmekle birlikte açık anahtarlı şifreleme ve anahtar kurulumunda kullanılmak üzere sunulan bazı algoritmaların dördüncü tur değerlendirmeleri halihazırda devam etmektedir. Biz bu çalışmada standartlaştırılmış bu üç algoritmanın çalışma prensiplerini açıkladık ve farklı güvenlik seviyesi sağlayan değişkelerin performanslarını ürettiği anahtarların, şifreli metinlerin, dijital imzaların uzunluğuna ve kapsülleme/dekapsülleme, dijital imza üretme/doğrulama sürelerine göre test ettik. Test sonuçları bize ML-KEM algoritmasının şimdiden anahtar kurulumunda kullanılabileceğini, kullanım gerekliliklerine göre ML-DSA ve SLH-DSA algoritmalarından birinin tercih edilebileceğini göstermiştir.

*Anahtar Kelimeler – Kuantum Hesaplama, Kuantum Dayanıklı Kriptografi, Açık Anahtarlı Kriptografi, Anahtar Kapsülleme Mekanizması.*

## I. GİRİŞ

Peter Shor, ayrık logaritma ve asal çarpanlara ayırma problemlerini bir şifrekıran kuantum bilgisayar (CRQC – Cryptographically Relevant Quantum Computer) aracılığıyla polinom zamanda çözebilen bir algoritma geliştirmiştir [1] - [2]. Standartlaştırılmış dijital imza algoritmalarının ve anahtar kurulum şemalarının güvenliği bu problemlerin zorluğuna dayandırıldığı için günümüz İnternet altyapısı üzerinden ilerleyen her türlü iletişim büyük bir güvenlik riskiyle karşı karşıya kalmıştır. Her ne kadar günümüzde

şifre kıran kuantum bilgisayarlar mevcut olmasa da saldırganlar şifreli mesajları ele geçirip depolayarak bu bilgisayarlar geliştirildiğinde anahtar kurulum mesajlarını kırdıktan sonra kullanılan ortak anahtarı elde edip geriye dönük deşifreleme yöntemiyle orijinal mesajları ortaya çıkarabilirler. Bu yüzden en kısa zamanda daha farklı matematiksel problemlerin zorluğuna dayanan kriptografik algoritmaların ve bunlar üzerine inşa edilmiş anahtar kurulum protokollerinin ve dijital imza algoritmalarının geliştirilmesi ihtiyacı doğmuştur. Bu amaçla NIST, 2016 yılında bir standartlaştırma süreci başlatmış ve bu alanda çalışan herkesten belli özellikleri sağlamak kaydıyla geliştirmiş oldukları algoritmalarla ilgili başvuruları kabul etmeye başlamıştır [3]. Nihayetinde NIST, 2022 yılında Kyber isimli algoritmayı açık anahtarlı şifreleme ve anahtar kurulum algoritması olarak; Dilithium, Falcon ve SPHINCS+ isimli algoritmaları ise dijital imza algoritması olarak standartlaştırmaya karar verdiğini duyurdu [4]. 2024 yılında Kyber algoritmasının ML-KEM (Module-Lattice-Based Key-Encapsulation Mechanism Standard) ismiyle FIPS 203 kodlu dokümanda [5], Dilithium ve SPHINCS+ algoritmalarının ise sırasıyla ML-DSA (Module-Lattice-Based Digital Signature Standard) ismiyle FIPS 204 kodlu dokümanda [6] ve SLH-DSA (Stateless Hash-Based Digital Signature Standard) ismiyle FIPS 205 kodlu dokümanda [7] standartlaştırıldığı, Falcon'a dayanan standart taslağının yayınlanmasının 2024 yılının sonlarına doğru planlandığı duyuruldu [8].

Mevcut literatürde bu standartların performansını iyileştirmeye yönelik birtakım çalışmalar [9] - [10] ve bunların yanı sıra TLS, PGP ve blokzincir gibi uygulama alanlarında performans ölçümüne yönelik çalışmalar [11]-[16] bulunmaktadır. Standartları açıklayan ve standartlarda tanımlanmış operasyonları çalışma zamanı ve çıktı büyüklükleri açısından kapsamlı bir şekilde değerlendiren özellikle Türkçe kaynakların literatürdeki eksikliği bu çalışmayı yapmamızdaki başlıca motivasyonumuzdur.

Bu çalışmada, standartları yayınlanmış algoritmalar hakkında bilgi sunduk ve bu algoritmaların içerdiği anahtar üretme, anahtar kapsülleme/dekapsülleme, belli bir mesajı imzalama ve imzayı doğrulama gibi operasyonların gerektirdiği zamanı ve bu operasyonların çıktısı olarak elde edilen verilerin büyüklüğünü inceledik. Gerçekleştirdiğimiz testlerden elde ettiğimiz bulgular bize ML-KEM ve ML-DSA algoritmalarının üretmiş olduğu, sırasıyla şifreli metin ve dijital imzanın kabul edilebilir büyüklükte olduğunu ve içerdikleri operasyonların yine kabul edilebilir sürelerde tamamlandığını göstermiştir. Bununla birlikte SLH-DSA algoritmasının kullandığı anahtarların uzunluğu her ne kadar kabul edilebilir olsa da ürettiği imzanın uzunluğu ve imzalama ve imza doğrulama operasyonları için gerektirdiği zaman bakımından alternatifi olan ML-DSA algoritmasından çok daha yavaştır ve iletilen veriye dahil edilmesi halinde çok daha fazla bant genişliği tüketme potansiyeline sahiptir.

Makalenin geri kalan kısmı şu şekilde organize edilmiştir: Bölüm II'de standartlarla ilgili detaylar sunulmuş ve yaptığımız deneysel çalışmaların nasıl gerçekleştirildiği anlatılmıştır. Bölüm III'te deneyler sonucunda elde ettiğimiz veriler sunulmuş ve akabinde Bölüm IV'te elde edilen veriler tartışılmış ve son olarak Bölüm V'te tartışmalarımız sonucunda elde ettiğimiz çıkarımlar paylaşılmıştır.

## II. MATERYAL VE YÖNTEM

Bu bölümde ML-KEM, ML-DSA ve SLH-DSA algoritmaları sunulacak ve deneylerin yapılması sırasında takip edilen yöntem açıklanacaktır.

### *A. Standartlaştırılmış Kuantum Dayanıklı Algoritmalar*

Standartlaştırılmış kuantum dayanıklı algoritmaları bu bölümde anahtar kurulum algoritması ve dijital imza algoritmaları olmak üzere iki ayrı bölümde inceleyeceğiz.

#### *1. Standartlaştırılmış Kuantum Dayanıklı Anahtar Kurulum Algoritması*

Bu bölümde NIST tarafından standartlaştırılan tek anahtar kurulum algoritması olan ML-KEM algoritmasının detaylarını sunacağız.

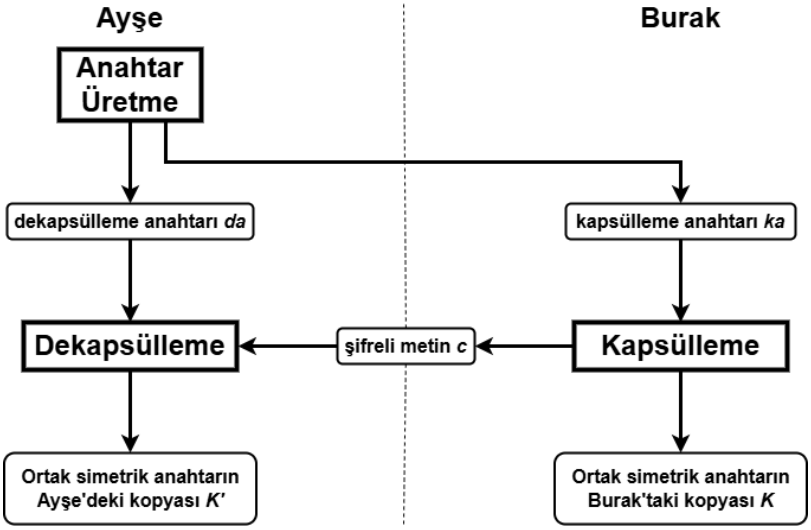
**a.** ***Modül Kafes Tabanlı Anahtar Kapsülleme Algoritması (ML-KEM – Module-Lattice-Based Key-Encapsulation Mechanism):*** ML-KEM algoritması anahtar kapsülleme mekanizmasına sahip bir anahtar kurulum algoritmasıdır. Bu algoritmanın artan güvenlik seviyelerine sahip ML-KEM-512, ML-KEM-768 ve ML-KEM-1024 isimli değişkeleri (varyantları) bulunmaktadır. ML-KEM algoritmasının kullanmış olduğu iki sabit değer bulunmaktadır: $n$ = 256 ve $q$ = 3329.

Algoritma, bu iki sabitin yanı sıra beş tam sayı değişkene sahiptir: $k$, $\eta_1$, $\eta_2$, $d_u$ ve $d_v$. Bu değişkenlerin farklı güvenlik seviyelerinde aldığı değerler FIPS 203 kodlu standartta yer alan Tablo 2'de görülebilir.

ML-KEM algoritması anahtar kapsülleme/dekapsülleme operasyonlarını gerçekleştirebilmek için beş ayrı algoritmadan yararlanır: anahtar üretme, şifreleme, deşifreleme, kapsülleme ve dekapsülleme algoritmaları. Anahtar üretme algoritması, kapsülleme ve dekapsülleme operasyonlarında kullanılmak üzere sırasıyla kapsülleme anahtarı *ka* ve dekapsülleme anahtarı *da* değerlerini üretir.

Kapsülleme algoritması güvenli iletişim kanalı kurmak isteyen iki taraftan birinin hem ortak bir simetrik anahtar üretmesini hem de üretmiş olduğu bu anahtarın kapsüllenmiş, yani sadece diğer tarafın içeriğine erişebileceği halini oluşturmayı sağlayan algoritmadır. Bu algoritma işlevini yerine getirebilmek için şifreleme algoritmasından yararlanır. Algoritmanın yürütülmesi sonucunda elde edilen açık metin halindeki ortak gizli anahtar algoritmayı yürüten tarafta saklanırken ortak gizli anahtarın kapsüllenmiş hali açık bir kanal üzerinden diğer tarafa iletilir. Dekapsülleme algoritması ise kapsüllenmiş bir şekilde gelen ortak simetrik anahtarın elde edilmesi işlevini gerçekleştirir.

ML-KEM anahtar kurulumu algoritmasının çalışma prensibi Şekil 1'de basit bir örnek üzerinden gösterilmiştir. Ayşe ve Burak güvenli bir kanal üzerinden iletişime geçmek istemektedirler. Bunun için her iki tarafın da aynı anahtara sahip olması gerekmektedir. Burak, ortadaki adam saldırılarına maruz kalmamak için Ayşe'nin kapsülleme anahtarı ***ka***'yı, gerçekten Ayşe'den ve iletişim kanalı üzerinde herhangi bir yerde değişmeden geldiğinden emin olacak şekilde edinir. Gönderici kimliği ve mesajın bütünlüğü doğrulanabildiği sürece bu anahtarın açık kanal üzerinden gönderilmesinin güvenlik açısından herhangi bir sakıncası yoktur. Sonrasında Burak ***ka*** anahtarını ve kapsülleme algoritmasını kullanarak ortak simetrik anahtarın kendisi için olan kopyası $\boldsymbol{K}$'yı ve bunun kapsüllenmiş hali olan $\boldsymbol{c}$'yi elde eder. Burak $\boldsymbol{c}$'yi açık bir kanal üzerinden Ayşe'ye gönderir. Ayşe ***da*** anahtarını ve dekapsülleme algoritmasını kullanarak ortak anahtarın kendisi için olan kopyası $\boldsymbol{K'}$ anahtarını elde eder. Tarafların paylaştığı mesajların bütünlüğü doğrulandığı sürece çok yüksek bir olasılıkla $\boldsymbol{K = K'}$ olması beklenir. Dekapsülleme algoritmasının farklı güvenlik seviyeleri için hatalı sonuç verme olasılıklarına FIPS 203 standardındaki Tablo 1'den erişebilirsiniz. Elde edilen ortak anahtar simetrik operasyonlarda doğrudan kullanılabileceği gibi SP 800-56C standardında [24] yer alan anahtar türetme algoritmalarından birinde girdi olarak da kullanılabilir.




Şekil 1. ML-KEM ile anahtar kurulumunun genel görünümü

*2. Standartlaştırılmış Kuantum Dayanıklı Dijital İmza Algoritmaları*

Bu bölümde NIST tarafından standartlaştırılan dijital imza algoritmaları olan ML-DSA ve SLH-DSA algoritmalarını sunacağız.

**a.** ***Modül Kafes Tabanlı Dijital İmza Algoritması (ML-DSA – Module-Lattice-Based Digital Signature Standard):*** ML-DSA algoritması modül kafes tabanlı bir dijital imza algoritmasıdır. Bu algoritmanın artan güvenlik seviyelerine sahip ML-DSA-44, ML-DSA-65 ve ML-KEM-87 isimli üç değişkesi vardır. Bu algoritmanın kullanmış olduğu üç sabit değer bulunmaktadır: $q = 2^{23} - 2^{13} + 1 = 8.380.417$, $\zeta = 1753$ ve $d = 13$. Algoritma, bu üç sabitin yanı sıra on tam sayı değişkene sahiptir. Bu değişkenlerin farklı güvenlik seviyelerinde aldığı değerler FIPS 204 kodlu standartta yer alan Tablo 1'de görülebilir.

**b.** ***Durumsuz Özet Tabanlı Dijital İmza Algoritması (SLH-DSA – Stateless Hash-Based Digital Signature Algorithm):*** SLH-DSA diğer özet tabanlı dijital imza şemalarını bileşen olarak kullanan bir dijital imza algoritmasıdır: (1) sadece birkaç kez imzalamayı sağlayan rastsal altkümelerin ormanı (FORS – forest of random subsets) ve (2) çok kere imzalamayı sağlayan genişletilmiş Merkle imza şeması (XMSS – eXtended Merkle Signature Scheme). XMSS, özet tabanlı Winternitz One-Time Signature Plus ($WOTS^+$) kullanılarak oluşturulur. SLH-DSA algoritması farklı güvenlik seviyelerinde çalışan ve her seviyesinde SHA2 özet fonksiyonunu veya SHAKE genişletilebilir çıktı fonksiyonunu (XOF – eXtendable Output Function) kullanabilen değişkeleri sahiptir.

Bir SLH-DSA anahtar çifti kavramsal olarak çok sayıda FORS anahtar çiftinden oluşmaktadır. FORS, her bir anahtar çiftinin az sayıda mesajı güvenli bir şekilde imzalamasını sağlar. Bir SLH-DSA imzası, mesajın rastsallaştırılmış bir özetinin hesaplanması, elde edilen mesaj özetinin bir kısmının sözde rastsal bir şekilde bir FORS anahtarı seçmesi ve mesaj özetinin kalan kısmının bu anahtarla imzalanması sonucu oluşturulur. Bu imza, FORS imzası ve FORS açık anahtarının kimliğini doğrulayan bilgiden oluşmaktadır. Bahsi geçen kimlik doğrulama bilgisi XMSS imzaları kullanılarak oluşturulur.

Bir FORS açık anahtarı için kimlik doğrulama bilgisi bir hiper ağaç imzasıdır. Hiper ağaç, XMSS ağaçlarından oluşan ağaca verilen isimdir. Bir hiper ağaç en üst katmanında (katman $d - 1$) tek bir XMSS ağacı olan $d$ katmandan oluşmaktadır. En alt katman da dahil olmak üzere (katman $0$) geri kalan tüm katmanlar her bir katmanda belli sayıda XMSS ağacı olacak şekilde oluşturulur. En alt katmandan en üst katmanın bir alt katmanına (katman $d - 2$) kadar olan tüm katmanlardaki her bir XMSS anahtarının açık anahtarı kendisinden bir üst katmandaki XMSS anahtarıyla imzalanır. En üst katmandaki XMSS anahtarının açık anahtarıyla başlayan ve bir FORS açık anahtarının kimliğini doğrulamak için gereken $d$ adet XMSS imza dizisine hiper ağaç imzası adı verilir. Bir SLH-DSA imzası, bir hiper ağaç imzası ve bir FORS imzasından oluşmaktadır.

### *B. Test Yöntemi*

Bu çalışmanın kapsamına almış olduğumuz algoritmaları gerçeklemiş ve güvenilir birkaç açık kaynak kütüphane bulunmaktadır. Open Quantum Safe (OQS) projesi [25], PQClean projesi [26] ve Stephan Müller'in yürütmüş olduğu leancrypto projesi [27] bu tür kütüphanelerin geliştirildiği projelere birkaç örnek olarak verilebilir. Geniş geliştirici topluluğu desteği ve düzenli aralıklarla güncellemeler yayınlıyor olması nedeniyle gerçekleştireceğimiz testlerde OQS projesinin liboqs kütüphanesini kullanmaya karar verdik.

Kriptografik algoritmaların performansı için tanımlı bazı metrikler vardır. Bunların başında, kullandıkları anahtar(lar)ın uzunluğu, ürettikleri şifreli metnin veya dijital imzanın uzunluğu, anahtar üretme süresi, şifreleme/kapsülleme süresi, deşifreleme/dekapsülleme süresi, imza oluşturma ve doğrulama süreleri gelmektedir.

Algoritmaların çalışma zamanı performanslarını ölçmek amacıyla 16 GB geçici belleğe ve 3.3 GHz temel frekansta çalışan Intel Core i5-4590 model dört çekirdekli işlemciye sahip, üzerinde Ubuntu 24.04.1 işletim sistemi çalışan HP ProDesk 400 G2 iş istasyonu kullanılmıştır. liboqs kütüphanesinde sunulmuş olan hız ölçüm programları kullanılarak tüm algoritma değişkelerinin her bir operasyonu 10 saniye boyunca çalıştırılmış; bu süre boyunca her bir operasyonun kaç kere tamamlanabildiği, her bir operasyon türünün ortalama ne kadar zaman aldığı ve ortalama ne kadar işlemci döngüsüne ihtiyaç duyduğu ölçülmüştür.

## III. BULGULAR

Bu bölümde ML-KEM, ML-DSA ve SLH-DSA algoritmalarının test sonuçları sunulmuştur. Her bir algoritmanın ve değişkelerinin kullandığı anahtarların uzunlukları; anahtar kapsülleme algoritmasının ve değişkelerinin üretmiş olduğu şifreli metinlerin (kapsüllenmiş ortak anahtarın) uzunluğu ve dijital imza algoritma değişkelerinin ürettiği dijital imzaların uzunluğu Tablo 1'de verilmiştir.

Tablo 1. ML-KEM değişkelerinin kullandığı anahtarların ve ürettiği şifreli metinlerin uzunluğu (bayt cinsinden)

| | Kapsülleme anahtarı | Dekapsülleme anahtarı | Şifreli metin | Paylaşılan ortak gizli anahtar |
|---|---|---|---|---|
| **ML-KEM-512** | 800 | 1632 | 768 | 32 |
| **ML-KEM-768** | 1184 | 2400 | 1088 | 32 |
| **ML-KEM-1024** | 1568 | 3168 | 1568 | 32 |

ML-DSA algoritmasının değişkelerinin kullandığı anahtarların ve ürettiği dijital imzaların uzunlukları Tablo 2'de verilmiştir.

Tablo 2. ML-DSA değişkelerinin kullandığı anahtarların ve ürettiği imzaların uzunluğu (bayt cinsinden)

| | Özel anahtar | Açık anahtar | İmza uzunluğu |
|---|---|---|---|
| **ML-DSA-44** | 2560 | 1312 | 2420 |
| **ML-DSA-65** | 4032 | 1952 | 3309 |
| **ML-DSA-87** | 4896 | 2592 | 4627 |

SLH-DSA algoritmasının değişkelerinin kullandığı anahtarların ve ürettiği dijital imzaların uzunlukları Tablo 3'te sunulmuştur.

Tablo 3. SLH-DSA değişkelerinin kullandığı anahtarların ve ürettiği imzaların uzunluğu (bayt cinsinden)

| | Özel anahtar | Açık anahtar | İmza uzunluğu |
|---|---|---|---|
| **SLH-DSA-SHA2-128f** | 64 | 32 | 17088 |
| **SLH-DSA-SHAKE-128f** | 64 | 32 | 17088 |
| **SLH-DSA-SHA2-192f** | 96 | 48 | 35664 |
| **SLH-DSA-SHAKE-192f** | 96 | 48 | 35664 |
| **SLH-DSA-SHA2-256f** | 128 | 64 | 49856 |
| **SLH-DSA-SHAKE-256f** | 128 | 64 | 49856 |

Tablo 4'te ML-KEM değişkelerinin anahtar üretimi, kapsülleme ve dekapsülleme operasyonları için çalışma zamanı performans değerleri sunulmuştur.

Tablo 4. ML-KEM değişkelerinin operasyonları sırasındaki çalışma zamanı performans değerleri

| Operasyonlar | Değişkeler | Tekrar sayısı | Ortalama zaman (µs) | Ortalama işlemci döngüsü |
|---|---|---|---|---|
| **Anahtar üretimi** | **ML-KEM-512** | 127634 | 78,349 | 257800 |
| | **ML-KEM-768** | 77564 | 128,926 | 424283 |
| | **ML-KEM-1024** | 46124 | 216,810 | 713678 |
| **Kapsülleme** | **ML-KEM-512** | 124554 | 80,287 | 264167 |
| | **ML-KEM-768** | 76410 | 130,873 | 430694 |
| | **ML-KEM-1024** | 45478 | 219,891 | 723781 |
| **Dekapsülleme** | **ML-KEM-512** | 123777 | 80,791 | 265851 |
| | **ML-KEM-768** | 76086 | 131,431 | 432627 |
| | **ML-KEM-1024** | 44935 | 222,548 | 732528 |

Dijital imza algoritmalarının performansı ölçülürken üretilen imzanın uzunluğu dışında kullanılan başlıca metrikler imzalama ve imza doğrulama süreleridir. Tablo 5'te ML-DSA değişkelerinin anahtar üretme, dijital imza üretme ve imza doğrulama için ihtiyaç duyduğu ortalama sürelerin yanı sıra yapılan testler için

ayrılan süre içerisinde bu operasyonları kaç kere tamamlayabildiği ve her bir operasyon için ihtiyaç duyulan işlemci döngüsü değerleri sunulmuştur.

Tablo 5. ML-DSA değişkelerinin operasyonları sırasındaki çalışma zamanı performans değerleri

| Operasyonlar | Değişkeler | Tekrar sayısı | Ortalama zaman (µs) | Ortalama işlemci döngüsü |
|---|---|---|---|---|
| **Anahtar üretimi** | **ML-DSA-44** | 27096 | 369,060 | 1214770 |
| | **ML-DSA-65** | 14321 | 698,304 | 2298898 |
| | **ML-DSA-87** | 8563 | 1167,860 | 3844890 |
| **İmzalama** | **ML-DSA-44** | 13890 | 719,957 | 2370085 |
| | **ML-DSA-65** | 8932 | 1119,668 | 3686166 |
| | **ML-DSA-87** | 5803 | 1723,298 | 5673544 |
| **İmza doğrulama** | **ML-DSA-44** | 31283 | 319,669 | 1052178 |
| | **ML-DSA-65** | 16329 | 612,417 | 2016208 |
| | **ML-DSA-87** | 9452 | 1058,003 | 3483234 |

SLH-DSA algoritmasının SHA2 ve SHAKE kullanan değişkeleri için ML-DSA için oluşturulana benzer iki ayrı tablo oluşturulmuştur. SLH-DSA-SHA2 değişkelerinin operasyonları sırasındaki çalışma zamanı performans değerleri Tablo 6'da gösterildiği şekildedir. SLH-DSA-SHAKE değişkelerinin performans değerleri ise Tablo 7'de sunulmuştur.

Tablo 6. SLH-DSA-SHA2 değişkelerinin operasyonları sırasındaki çalışma zamanı performans değerleri

| Operasyonlar | Değişkeler | Tekrar sayısı | Ortalama zaman (µs) | Ortalama işlemci döngüsü |
|---|---|---|---|---|
| **Anahtar üretimi** | **SLH-DSA-SHA2-128f** | 1669 | 5994,561 | 19736156 |
| | **SLH-DSA-SHA2-192f** | 1192 | 8395,412 | 27640646 |
| | **SLH-DSA-SHA2-256f** | 439 | 22825,205 | 75148960 |
| **İmzalama** | **SLH-DSA-SHA2-128f** | 72 | 140091,042 | 461232037 |
| | **SLH-DSA-SHA2-192f** | 46 | 220596,065 | 726284356 |
| | **SLH-DSA-SHA2-256f** | 22 | 461937,682 | 1520871172 |
| **İmza doğrulama** | **SLH-DSA-SHA2-128f** | 1068 | 9371,449 | 30854138 |
| | **SLH-DSA-SHA2-192f** | 812 | 12324,814 | 40577787 |
| | **SLH-DSA-SHA2-256f** | 811 | 12341,400 | 40632376 |

Tablo 7. SLH-DSA-SHAKE değişkelerinin operasyonları sırasındaki çalışma zamanı performans değerleri

| Operasyonlar | Değişkeler | Tekrar sayısı | Ortalama zaman (µs) | Ortalama işlemci döngüsü |
|---|---|---|---|---|
| **Anahtar üretimi** | **SLH-DSA-SHAKE-128f** | 667 | 14999,877 | 49385014 |
| | **SLH-DSA-SHAKE-192f** | 456 | 21939,057 | 72231435 |
| | **SLH-DSA-SHAKE-256f** | 173 | 58126,838 | 191375124 |
| **İmzalama** | **SLH-DSA-SHAKE-128f** | 29 | 351839,379 | 1158386554 |
| | **SLH-DSA-SHAKE-192f** | 18 | 568831,389 | 1872805717 |
| | **SLH-DSA-SHAKE-256f** | 9 | 1166551,333 | 3840723164 |
| **İmza doğrulama** | **SLH-DSA-SHAKE-128f** | 475 | 21096,840 | 69458587 |
| | **SLH-DSA-SHAKE-192f** | 321 | 31175,352 | 102640804 |
| | **SLH-DSA-SHAKE-256f** | 331 | 30219,704 | 99494494 |

## IV. TARTIŞMA

Tablo 1, Tablo 2 ve Tablo 3'te sunulan verilere baktığımızda sağlanan güvenlik seviyesi arttıkça kullanılan kapsülleme/dekapsülleme anahtarlarının veya özel/açık anahtarların uzunluğunun ve kapsüllenmiş ortak anahtarın veya dijital imza uzunluğunun artmakta olduğunu görüyoruz. ML-KEM ve ML-DSA değişkelerindeki bu artışın nedeni, kullanılan parametre değerlerinin güvenlik seviyesini artırmak amacıyla artırılması sonucunda polinom sayılarındaki ve polinom katsayılarındaki değer aralığı artışıdır. SLH-DSA değişkelerinde gözlemlenen benzer artış eğiliminin nedeni ise kullanılan parametre değerlerinin artması neticesinde anahtarları temsil etmek amacıyla kullanılan özet fonksiyonlarının çıktı uzunluğunun artırılması ve imzaları temsil eden hiper ağacın yüksekliğinin ve genişliğinin artmasıdır.

ML-DSA ve SLH-DSA değişkelerini ürettiği imza uzunluğu açısından karşılaştırdığımızda en düşük seviyede güvenlik sağlayan SLH-DSA değişkesinin üretmiş olduğu imzanın en yüksek seviye güvenlik sağlayan ML-DSA değişkesinin üretmiş olduğu imzadan yaklaşık dört kat daha uzun olduğunu görmekteyiz. Öte yandan SLH-DSA değişkelerinin özel/açık anahtarları ML-DSA değişkelerinin özel/açık anahtarlarından onlarca kat daha kısadır.

Tablo 4 ve Tablo 5'te, sağlanan güvenlik seviyesi arttıkça ML-KEM ve ML-DSA değişkelerinin operasyonları sırasında kullandıkları sürelerin arttığını görmekteyiz. Anahtar üretimi için bunun nedeni daha fazla sayıda rastsal sayı üretme ve daha fazla sayıda ve daha geniş aralıkta değerler içeren matrislerin çarpımının hesaplanması gerekliliğidir. Kapsülleme/dekapsülleme veya dijital imza üretme ve imza doğrulamadaki artış yine çok sayıda ve geniş değer aralığına sahip değerleri içeren matrislerin çarpılması gerekliliğiyle açıklanabilir.

SLH-DSA-SHA2 ve SLH-DSA-SHAKE değişkelerini karşılaştırdığımızda SHAKE algoritmasını kullanan değişkelerin operasyonlar için SHA2 algoritmasını kullananlara kıyasla iki kattan daha uzun süreye ihtiyaç duyduğunu görmekteyiz. Bunun nedeni SHAKE algoritmasının çıktı uzunluğunun ayarlanabilmesi için sahip olduğu iç mekanizmaların sabit uzunluklu çıktı veren SHA2 gibi algoritmalara kıyasla daha uzun süreye ihtiyaç duymasıdır.

## V. SONUÇLAR

Bu çalışmada NIST'in standartlaştırmış olduğu ML-KEM, ML-DSA ve SLH-DSA isimli üç kuantum dayanıklı kriptografik algoritmanın çalışma prensiplerini sunduk ve bu algoritmaların farklı güvenlik seviyelerindeki değişkelerini ürettiği anahtarların, şifreli metinlerin ve dijital imzaların uzunluğu; kapsülleme/dekapsülleme ve imzalama/imza doğrulama gibi operasyonlar için gerektirdiği süreler gibi metrikleri kullanarak performanslarını analiz ettik.

Yaptığımız test sonuçları gösterdi ki ML-KEM değişkeleri anahtar kurulumu konusunda sadece RSA değişkelerine değil eliptik eğri kullanan Diffie-Hellman değişkelerine bile alternatif olabilecek yapıdadır. Şu an için bu son iki algoritma değişkelerinin ML-KEM değişkelerine nazaran tek avantajlı yanı güvenliklerinin dayandığı problemler üzerinde çokça çalışılmış olmasıdır. Yalnız, *şimdi depola, sonra deşifrele* saldırılarına maruz kalmamak adına ML-KEM değişkelerinin en azından mevcut anahtar kurulum algoritmalarıyla melez bir şekilde kullanılmasını önermekteyiz.

ML-DSA ve SLH-DSA dijital imza algoritma değişkelerinden herhangi bir grup tüm metriklerde diğerine üstün gelememiştir. Bu yüzden kullanım amacına göre birinden biri tercih edilebilir. Örneğin, ürettiği anahtarın kısa olmasının tercih edildiği bir durumda veya güvenliğinin dayandığı problemde ortaya çıkabilecek açıklığın doğurabileceği zafiyetleri kaldıramayacak kritik kullanım durumlarında SLH-DSA değişkeleri kullanılırken diğer kullanım durumlarında ML-DSA değişkeleri tercih edilebilir. Çünkü SLH-DSA algoritmasının güvenliği kriptografik özet fonksiyonlarına dayanmaktadır ve bu fonksiyonların güvenliği üzerinde yıllarca çalışılmış, güvenlik gerekliliklerini karşıladıkları matematiksel olarak ispatlanmıştır. Öte yandan beklenmedik durumlarla karşılaşmamak adına mevcut dijital imzalarla birlikte melez bir yapıda kullanılmalarını önermekteyiz.

## KAYNAKLAR


[1] Shor, Peter W. "Algorithms for quantum computation: discrete logarithms and factoring." Proceedings 35th annual symposium on foundations of computer science. Ieee, 1994.

[2] Shor, Peter W. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer." SIAM review 41.2 (1999): 303-332.

[3] NIST, CFP. "Submission requirements and evaluation criteria for the post-quantum cryptography standardization process." (2016).

[4] Alagic, Gorjan, et al. "Status report on the third round of the NIST post-quantum cryptography standardization process." (2022): 07.

[5] Raimondo, G. M., Locascio, L.E. "Module-Lattice-Based Key-Encapsulation Mechanism Standard (ML-KEM)." (2024).

[6] Raimondo, G. M., Locascio, L.E. "Module-Lattice-Based Digital Signature Standard (ML-DSA)." (2024).

[7] Raimondo, G. M., Locascio, L.E. "Stateless Hash-Based Digital Signature Standard (SLH-DSA)." (2024).

[8] Boutin, C. "NIST Releases First 3 Finalized Post-Quantum Encryption Standards." NIST, https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards. Son Erişim Tarihi: 27 Kasım 2024
[9] Saarinen, Markku-Juhani O. "Accelerating SLH-DSA by Two Orders of Magnitude with a Single Hash Unit." Annual International Cryptology Conference. Cham: Springer Nature Switzerland, 2024.
[10] Medina, Francesco, et al. "Analysis and contributions to an open source Kyber library in Rust." 2024 IEEE Symposium on Computers and Communications (ISCC). IEEE, 2024.
[11] Wussler, Aron. Post-Quantum cryptography in OpenPGP. Diss. Wien, 2023.
[12] Kampanakis, Panos, and Will Childs-Klein. "The impact of data-heavy, post-quantum TLS 1.3 on the Time-To-Last-Byte of real-world connections." Cryptology ePrint Archive (2024).
[13] Gunawardena, Subodha. "Is Blockchain Ready to Handle Quantum Supremacy? A Survey of Quantum Vulnerabilities and Preparedness."
[14] Aremu, Olemilekan Rasaq, Samet Tonyali, and Abdulkadir Köse. "A Password Manager for Post-Quantum Era.", 2022
[15] Sarıbaş, Sultan, and Samet Tonyalı. "Performance Evaluation of TLS 1.3 Handshake on Resource-Constrained Devices Using NIST's Third Round Post-Quantum Key Encapsulation Mechanisms and Digital Signatures." 2022 7th International Conference on Computer Science and Engineering (UBMK). IEEE, 2022.
[16] Bozhko, Jessica, et al. "Performance evaluation of quantum-resistant TLS for consumer IoT devices." 2023 IEEE 20th Consumer Communications & Networking Conference (CCNC). IEEE, 2023.
[17] Albrecht, Martin R., Rachel Player, and Sam Scott. "On the concrete hardness of learning with errors." Journal of Mathematical Cryptology 9.3 (2015): 169-203.
[18] Boudgoust, Katharina, et al. "On the hardness of module learning with errors with short distributions." Journal of Cryptology 36.1 (2023): 1.
[19] Langlois, Adeline, and Damien Stehlé. "Hardness of decision (R) LWE for any modulus." Cryptology ePrint Archive (2012).
[20] Laguillaumie, Fabien, et al. "Lattice-based group signatures with logarithmic signature size." Advances in Cryptology-ASIACRYPT 2013: 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II 19. Springer Berlin Heidelberg, 2013.
[21] Langlois, Adeline, and Damien Stehlé. "Worst-case to average-case reductions for module lattices." Designs, Codes and Cryptography 75.3 (2015): 565-599.
[22] Lippert, Jan, Johannes Blömer, and Gitta Domik. "The Fujisaki-Okamoto Transformation." University of Paderborn, The University of the Information Society, Faculty of Electrical Engineering, Computer Science and Mathematics (2014).
[23] Hofheinz, Dennis, Kathrin Hövelmanns, and Eike Kiltz. "A modular analysis of the Fujisaki-Okamoto transformation." Theory of Cryptography Conference. Cham: Springer International Publishing, 2017.
[24] Barker, Elaine, Lily Chen, and Richard Davis. "Recommendation for key-derivation methods in key-establishment schemes." NIST Special Publication 800 (2020): 56C-rev2.
[25] Stebila, Douglas, and Michele Mosca. "Post-quantum key exchange for the internet and the open quantum safe project." International Conference on Selected Areas in Cryptography. Cham: Springer International Publishing, 2016.
[26] Kannwischer, Matthias J., et al. "Improving software quality in cryptography standardization projects." 2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). IEEE, 2022.
[27] Müller, Stephan. "Ascon-Keccak AEAD Algorithm." Cryptology ePrint Archive (2024).