YILDIZ TEKNİK ÜNİVERSİTESİ

ELEKTRİK – ELEKTRONİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BLM5101 Bilgisayar Güvenliği ve Kriptografi

DÖNEM PROJESİ

NTRUEncrypt ve NTRUSign Tabanlı Kriptografi: Algoritma, Uygulama ve Güvenlik Analizi

Prof. Dr. SIRMA YAVUZ

24501054- Makbule Ozge Ozler

İstanbul, 2024

Içindekiler

1 Özet

NTRUEncrypt ve NTRUSign, modern kriptografinin temel taşlarından biri olan kafes tabanlı sistemlere dayanan, özellikle kuantum bilgisayar tehditlerine karşı dirençli olacak şekilde tasarlanmış şifreleme ve dijital imzalama algoritmalarıdır. Bu rapor, NTRU tabanlı kriptografik sistemlerin tarihsel gelişimi, teknik detayları, uygulama örnekleri ve güvenlik analizlerini kapsamlı bir şekilde ele almayı amaçlamaktadır. Bu çalışmada ayrıca Java dilinde geliştirilen örnek bir uygulama üzerinden, şifreleme-çözme ve imzalama-doğrulama süreçleri kod düzeyinde analiz edilmiştir. Deneysel çalışmalarla algoritmaların işlem süreleri, rastgelelik düzeyleri, hata toleransları ve güvenlik kriterleri detaylı olarak incelenmiştir. Farklı parametre setlerinin algoritmalar üzerindeki etkileri karşılaştırmalı şekilde sunulmuştur. Sonuç olarak, NTRUEncrypt ve NTRUSign algoritmaları, kuantum sonrası kriptografi dünyasında oldukça önemli yer tutan, verimli ve güvenli yapılar sunmaktadır. Bu rapor, bu algoritmaların hem teorik temellerini hem de pratik uygulamalarını derinlemesine inceleyerek, okuyucuya kapsamlı bir bakış açısı kazandırmayı hedeflemektedir.

2 Giriş

Kriptografi, bilgi güvenliği alanında temel bir yapı taşıdır ve dijital çağın getirdiği güvenlik ihtiyaçlarına cevap vermek için sürekli evrim geçirmektedir. Günümüzde kullanılan pek çok şifreleme ve dijital imza algoritması, klasik hesaplama modellerine dayanarak geliştirilmiştir. Ancak kuantum bilgisayarların gelişimiyle birlikte, geleneksel kriptografik algoritmaların çoğu tehdit altına girmiştir. Özellikle RSA, DSA, ElGamal ve ECC gibi algoritmaların, kuantum bilgisayarlar karşısında güvenliklerini yitirecekleri bilimsel olarak öngörülmektedir. Bu durum, post-kuantum kriptografi olarak adlandırılan yeni bir alanın doğmasına sebep olmuştur. Post-kuantum kriptografi, kuantum bilgisayarlara karşı dirençli algoritmaların araştırılması ve standartlaştırılması amacı taşır.

Bu bağlamda, NTRU (Nth-degree Truncated Polynomial Ring Unit) algoritmaları, post-kuantum kriptografi alanının önde gelen temsilcilerinden biridir. NTRUEncrypt ve NTRUSign, kafes (lattice) tabanlı kriptografi paradigmasına dayalı olarak geliştirilmiştir. Kafes tabanlı algoritmalar, NP-zor matematiksel problemlere dayanır ve bilinen kuantum algoritmaları bu problemlere karşı etkin çözümler sunamamaktadır. Bu nedenle, NTRU tabanlı çözümler, hem teorik hem de pratik açıdan kuantum sonrası dönemde güvenilir bir alternatif olarak değerlendirilmektedir.

NTRU algoritmaları, ilk kez 1996 yılında Hoffstein, Pipher ve Silverman tarafından önerilmiştir.[1] O zamandan bu yana, algoritmalar hem akademik dünyada hem de endüstride yoğun ilgi görmüştür.

NIST (National Institute of Standards and Technology), 2016 yılında post-kuantum kriptografi için bir standartlaştırma süreci başlatmıştır. Bu süreçte, dünya genelinden çeşitli kriptografik algoritmalar değerlendirmeye alınmıştır. NTRU tabanlı algoritmalar, bu sürecin önemli adayları arasında yer almış ve NIST tarafından önerilen algoritmalar arasında gösterilmiştir. Özellikle

"NTRUEncrypt" ve daha sonra türetilen "NTRU Prime" varyantları, güvenlik, performans ve uygulama kolaylığı açısından güçlü adaylar olarak öne çıkmıştır.

NTRUEncrypt, mesajların gizliliğini sağlamak amacıyla tasarlanmış bir şifreleme algoritmasıdır. Buna karşılık, NTRUSign ise veri bütünlüğü ve gönderici kimliğinin doğrulanması için kullanılan bir dijital imza algoritmasıdır. Her iki yapı da benzer matematiksel temelleri paylaşsa da kullanım amaçları, anahtar yapıları ve güvenlik hedefleri bakımından farklılık gösterir. Bu iki mekanizma, özellikle güvenli iletişim protokolleri ve kimlik doğrulama sistemlerinde tamamlayıcı şekilde kullanılabilir.

Bu raporda, NTRUEncrypt ve NTRUSign algoritmalarının temelleri, matematiksel yapıları, uygulama örnekleri, avantajları, zayıflıkları ve NIST sürecindeki konumları detaylı bir şekilde ele alınacaktır. Amaç, okuyucunun hem bu algoritmaların teknik altyapısını anlamasını sağlamak hem de kuantum sonrası dönemde neden bu kadar kritik bir rol oynadıklarını ortaya koymaktır.

Tablo 1: NTRUEncrypt ve NTRUSign Karşılaştırması

Özellik	NTRUEncrypt	NTRUSign
Kriptografik amaç	Gizlilik (confidentiality)	Kimlik doğrulama (authentication)
Anahtar türü	Şifreleme anahtar çifti	İmza anahtar çifti
Operasyon	Şifreleme / Çözme	İmzalama / Doğrulama
Güvenlik temeli	Lattice problemi	Lattice tabanlı imza üretimi
Kullanım alanı	Mesaj/dosya şifreleme	Dijital belge imzalama

3 NTRUEncrypt Algoritmasının Temelleri

NTRUEncrypt algoritması, halka üzerinde tanımlı çok terimli polinomlar ve modüler aritmetik kullanılarak geliştirilen bir açık anahtarlı şifreleme sistemidir. Klasik RSA veya ElGamal gibi algoritmalardan farklı olarak, asal çarpanlara ayırma veya logaritmik işlemlere dayanmaz. Bunun yerine, çok boyutlu kafes problemlerine ve özellikle en yakin vektor, Closest Vector Problem (CVP), gibi zorluk derecesi yüksek problemlere dayanır. NTRU'nun yapısı, hem şifreleme/çözme süreçlerinde hız avantajı sağlar hem de kuantum bilgisayarların potansiyel saldırılarına karşı direnç sunar.

3.1 Matematiksel Arka Plan

NTRU'nun temelinde, rasyonel sayıların cebirsel uzantısı olan $\mathbb{N}_q[\mathbf{X}]/(\boldsymbol{X^N}-1)$ halkası bulunmaktadır. Burada $(\boldsymbol{X^N}-1)$, N dereceli bir çembersel polinomdur. Bu yapı, polinomların

katsayılarının mod p ve mod q alınarak ayrı ayrı değerlendirildiği iki farklı modülasyon düzlemi ile birleştirilir. Tipik olarak, p küçük bir asal sayı (örneğin 3), q ise daha büyük bir asal sayı (örneğin 2048) olarak seçilir. Algoritma kapsamında kullanılan polinomlar, genellikle "çok az" sayıda sıfır olmayan katsayıya sahip olacak şekilde seçilir; bu polinomlara sparse (seyrek) polinomlar denir.

Örneğin, iki polinomun çarpımı ve ardından mod q alınması işlemi, algoritmanın merkezinde yer alır. Verilen bir halka *R* içinde, şifreleme şu şekilde tanımlanır: açık anahtar *h* ve rastgele seçilen bir kısa polinom *r* ile, şifreli mesaj e = p * *r* * *h* + m mod q olarak elde edilir. Bu yapı, hem rastgeleliği hem de doğruluğu garantilerken, çözme işlemi yalnızca özel anahtara sahip olan tarafça gerçekleştirilebilir.

$$R = \mathbb{Z}_q[\mathbf{X}]/(\boldsymbol{X^N} - 1)$$

Yani, mod $(X^N - 1)$: ile sınırlı tamsayı katsayılı polinom halkasıdır.

3.2 Parametreler

NTRU algoritmasında güvenlik ve verimlilik parametrelerin seçiminden doğrudan etkilenir. Üç temel parametre şunlardır:

- **N**: Polinomların derecesi; yani kullanılan halkadaki elemanların büyüklüğünü belirler. Tipik değerler 167, 251, 347 gibi asal ya da çift sayılardır.
- **p**: Küçük modül; genellikle 3 olarak alınır. Mesaj bu modülde temsil edilir.
- **q**: Büyük modül; 2048, 4096 gibi büyük asal veya çift sayılar olabilir.

Anahtarların boyutu, algoritmanın güvenliği ve şifreleme hızı bu üç parametreye bağlıdır. Güçlü güvenlik için q'nun büyüklüğü artırılırken, performansı artırmak için p küçük tutulur. Ayrıca, r ve f gibi rastgele seçilen kısa polinomlar da sistemin güvenliğini etkiler. Bu polinomların yapay olarak rastgele, ancak önceden tanımlı yapıda olması, hem verimliliği artırır hem de deterministik anahtar üretimine imkân tanır. Parametre setleri genellile EncryptionParameters.Apr2011_439_FAST gibi sabit tanimlar ile gelir.

Tablo 2: NTRUEncrypt Parametreleri ve Açıklamaları

Parametre	Açıklama
N	Polinom derecesi (halkanın boyutu)
p	Küçük bir asal sayı (genellikle 3)
q	Büyük bir modül (genellikle 2048, 4096 gibi)
df	Özel anahtarın katsayılarındaki +1 sayısı
dg	Rastgele polinomun (g) +1 sayısı
dr	Şifreleme sırasında kullanılan rastgele polinomun derecesi

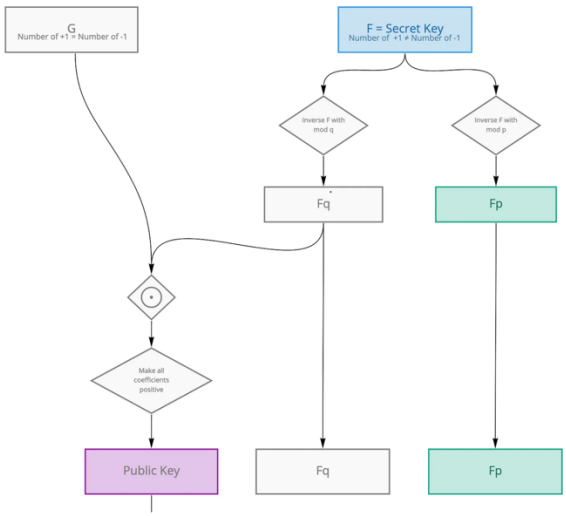
3.3 Anahtar Üretimi

NTRUEncrypt algoritmasında anahtar üretimi, iki temel kısa polinomun belirlenmesiyle başlar: f ve g. Bu polinomlar, modüler işlemlerde belirli özellikleri sağlamalıdır. f kısa polinomu, hem mod p hem de mod q altında terslenebilir (invertible) olmalıdır. Bu, f mod p ve f mod q işlemlerinde ters polinomların bulunmasını mümkün kılar, böylece şifre çözme işlemi güvenilir bir şekilde gerçekleşir.

Anahtar üretim süreci şu adımlardan oluşur:

1. Parametreler belirlenir: N, p ve q seçilir.
2. f kısa polinomu (df tane +1 ve -1 içeren özel bir polinom) seçilir ve hem mod p hem de mod q altında tersinin alınabileceği doğrulanır.
3. g kısa polinomu (dg tane +1 ve -1 içeren rastgele bir polinom) rastgele seçilir; genellikle kısa ve sparse bir polinomdur.
4. Açık anahtar hesaplanır: $h = p.g \ \ f^{-1} \bmod q$
5. Özel anahtar: f ve f^{-1} mod p (bazı durumlarda g de özel anahtara dahil edilir).

Bu yapı, şifreleme işleminin yalnızca h (açık anahtar) ile yapılmasını, şifre çözmenin ise yalnızca ile mümkün olmasını sağlar.



Figüre 1: Anahtar Uretim Süreci Akis Diagrami

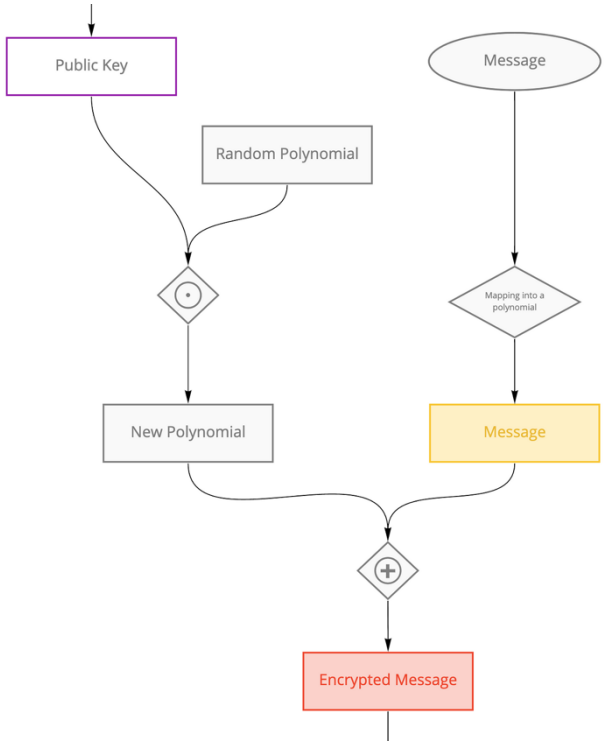
3.4 Şifreleme

NTRUEncrypt algoritmasında şifreleme işlemi, alıcıya ait açık anahtar (h) kullanılarak gerçekleştirilir. Bu işlem, gönderilecek mesajın bir polinoma dönüştürülmesi ve bu mesajın belirli bir şekilde rastgele bir polinomla karıştırılması ile sağlanır.

Bir mesaj m (plaintext) şifrelenirken:

1. Mesaj polinomu m seçilir.
2. r adlı rastgele bir kısa polinom seçilir.
3. Şifreli mesaj hesaplanir: $e = r.h + m \mod q$

Burada güvenliği sağlayan temel unsur, rastgelelik ve halkasal çarpma ile önceden tahmin edilemez yapıların oluşmasıdır. Yani, burada m mesajının (plaintext) gizlenmesi, rastgeleliğin eklenmesi ve modüler yapı sayesinde şifreli mesajdan m'yi çıkarmak saldırganlar için son derece zordur.



Figüre 2: Şifreleme Algoritması Akis Diyagramı

3.5 Şifre Çözme

NTRUEncrypt algoritmasında şifre çözme işlemi, yalnızca özel anahtara (f) sahip olan tarafça yapılabilir. Bu da, açık anahtar kriptografisinin temel prensibi olan tek yönlülüğü sağlar. Bu sürecin güvenli ve etkili şekilde çalışabilmesi için, anahtar üretim aşamasında seçilen f kısa polinomunun mod p ve mod q altında terslenebilir olması kritik önemdedir. Şifre çözme işlemi, e, şifreli mesajın (ciphertext) özel anahtar (f) ile işlenmesi ve orijinal mesaj polinomunun geri elde edilmesi adımlarını içerir.

Şifre çözme adımları aşağıdaki gibidir:

1. Alıcı, kendisine gelen şifreli mesaj polinomu e'yi alır.
2. İlk olarak $f * e \mod q$ işlemi gerçekleştirilir. Bu, mod q altında f polinomunun şifreli mesajla çarpılması anlamına gelir:

$$a = f.e \ mod \ q$$

Bu aşamada *a*, hem şifreli mesaj hem de özel anahtar bilgilerini içeren bir ara sonuçtur.

3. Ardından, *a* polinomunun katsayıları, q etrafında "merkezlenmiş mod *p*" işlemine tabi tutulur. Yani, her katsayı *q*/2'nin altına düşürülerek daha küçük bir aralıkta normalize edilir:

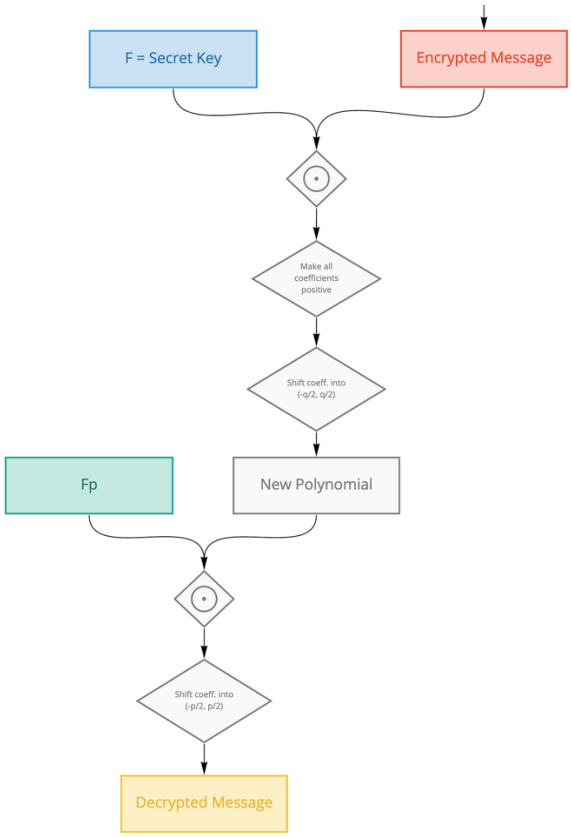
$$a^{\iota} = CentredeLift(a, q)$$

4. Daha sonra, a^{ι} polinomu mod *p* altında f^{-1} ile çarpılır. Burada $f^{-\iota}$, *f* polinomunun mod *p* altında tersidir:

$$\mathrm{m} = f^{-1} * a^{\iota} \bmod p$$

Bu işlem sonucunda elde edilen *m*, orijinal mesaj polinomudur. Eğer tüm adımlar doğru parametrelerle ve doğru sırayla gerçekleştirilmişse, alıcı mesajı hatasız şekilde elde eder. Ancak *q* yeterince büyük değilse veya *f* yanlış seçilmişse, çözme sırasında doğruluk kaybı olabilir.

Şifre çözme işleminde başarısızlık olasılığı, parametre seçimlerine bağlı olarak minimize edilebilir. Pratik uygulamalarda bu hata oranı genellikle sıfıra çok yakındır. Bu da NTRU'nun hem güvenli hem de uygulanabilir bir şifreleme çözümü olmasını sağlar.



Figüre 3: Deşifreleme Algoritması Akis Diyagramı

4 NTRUSign (İmza) Algoritmasının Temelleri

NTRUSign, kafes tabanlı kriptografi prensiplerini dijital imzalama amacıyla kullanan, NTRU ailesine ait bir algoritmadır. Şifrelemeden farklı olarak, imzalama sistemlerinde temel amaç, bir verinin belirli bir kullanıcı tarafından üretildiğinin kriptografik olarak kanıtlanmasıdır. NTRUSign, en yakın vektör problemi (CVP) gibi çözümü zor matematiksel problemlere dayanarak, orijinal mesajın kısa bir vektörle işaretlenmesini sağlar.

4.1 Dijital İmzanın Kriptografik Anlamı

Dijital imza, gönderilen bir verinin bütünlüğünü ve kaynağını doğrulayan bir şemadır. Kriptografik imzalar, özellikle e-posta, yazılım güncellemeleri ve dijital belgeler gibi dijital iletişimlerde kullanılır. Bir imza şeması, 3 temel bileşene dayanır:

1. anahtar üretimi,
2. imzalama
3. ve doğrulama.

Gönderen, özel anahtarı ile veriye imza atar; alıcı ise, açık anahtarı kullanarak bu imzayı doğrular. İmzanın geçerli olması, mesajın değişmediği ve yalnızca anahtar sahibince üretildiği anlamına gelir.

Dijital imza, dijital ortamlarda bir verinin:

- kim tarafından gönderildiğini doğrulamak (authentication),
- verinin değiştirilmediğini garanti altına almak (integrity),
- ve çoğu zaman inkâr edilemezliğini sağlamak (non-repudiation) amacıyla kullanılır.

Klasik dünyada RSA ya da ECDSA kullanılsa da, bu algoritmalar kuantum bilgisayarlara karşı savunmasızdır. Bu nedenle, lattice tabanlı imza algoritmaları (ör. NTRUSign) kuantum sonrası dönemin önemli adaylarıdır.

4.2 NTRUSign Neye Dayanır?

NTRUSign, tıpkı NTRUEncrypt gibi polinom halkaları üzerinde çalışan lattice temelli bir algoritmadır. Temel cebirsel yapı şuna dayanır:

$$R = \mathbb{Z}_q[\mathbf{X}]/(\boldsymbol{X^N} - 1)$$

Ve işlemler bu halka üzerinde gerçekleştirilir.

Ancak NTRUSign'in amacı şifreleme değil, dijital imzalama sağlamaktır. Bu doğrultuda NTRUSign, önceden belirlenmiş bir "trapdoor" (gizli bilgi) ile iz düşüm (projection-based) temelli bir imza üretme metodolojisi kullanır.

Algoritmanın hedefi, mesaj ile ilişkili bir yapıya göre türetilmiş ve normu (uzunluğu) düşük bir vektör bulmaktır. Bu vektör, imzanın kendisidir. İmza oluşturma aşamasında özel anahtar yardımıyla kısa bir çözüm (yaklaşık çözüm) hesaplanır. Bu çözüm, doğrulama sürecinde halka içindeki belirli cebirsel ilişkileri sağlamalıdır. NTRUSign, yapısal olarak GGH (Goldreich–Goldwasser–Halevi) tipi imza algoritmalarına benzemekle birlikte, güvenlik açıklarını önlemeye yönelik istatistiksel koruma katmanları ile donatılmıştır.

Bu koruma katmanları arasında özellikle Gaussian sampling (Gauss dağılımına göre örnekleme) ve rejection sampling (reddetme örneklemesi) teknikleri öne çıkar. Bu yöntemler sayesinde, her imza benzersiz görünür ve imzaların özel anahtar hakkında bilgi sızdırması engellenmiş olur. Böylece NTRUSign, hem teorik güvenliğe hem de pratik gizliliğe sahip, modern post-kuantum imza şemaları arasında yerini alır.

4.3 NTRUSign Anahtar Üretimi

NTRUSign algoritmasında anahtar üretimi, NTRUEncrypt'e benzer biçimde polinomlar üzerinden gerçekleştirilir.

Özel anahtar;

- belirli yapıda seçilmiş kısa polinom çiftlerinden (*f*, *g*) oluşur.
- $\mathbb{Z}_{\boldsymbol{q}}[\mathbf{X}]/(\boldsymbol{X^N}-1)$ Halkası altında tanımlanır ve bazı güvenlik koşullarını sağlamalıdır.
- En temel gereksinim, bu polinomların oluşturduğu kafesin "iyi koşullandırılmış" olmasıdır; bu da doğrulama sürecinin doğruluğunu ve güvenilirliğini etkiler.

Açık anahtar (*T*) ise;

- Doğrulama aşamasında kullanılır.
- Genellikle sabit boyutludur.
- Gaussian sampling, bu süreçte kullanılan polinomların dağılımının gizliliğini sağlamak için tercih edilir.

Anahtar üretim süreci özetle şu adımlardan oluşur:

1. Parametreler belirlenir: *N*, *q* ve hedeflenen güvenlik seviyesi doğrultusunda seçilir.
2. İki adet kısa polinom (genellikle Gauss dağılımına göre) rastgele seçilir: *f* ve *g*.
3. Bu iki polinomun oluşturduğu kafesten açık anahtar (*T*) türetilir: $T = f^{-1} * g \bmod q$
4. Açık anahtar (*T*) halka üzerinden halka elemanlarına dönüştürülür ve sabitlenir.

4.4 İmzalama Süreci

NTRUSign'de imzalama işlemi, belirli bir mesaj için özel anahtar yardımıyla kısa bir vektör çözümü elde etmeyi hedefler. Bu süreçte, mesaj önce bir "hash" fonksiyonundan geçirilerek halka içinde bir hedef noktaya dönüştürülür. Daha sonra, özel anahtar kullanılarak bu hedefe yakın kısa bir vektör bulunur.

İmzalama süreci:

1. Mesaj *m* alınır ve güçlü bir kriptografik hash fonksiyonu ile halka $\mathbb{Z}_{\boldsymbol{q}}[\mathbf{X}]/(\boldsymbol{X^N}-1)$ elemanına dönüştürülür: H(m)
2. Özel anahtar (*f*, *g*) kullanılarak H(m)'e yakın kısa bir vektör *s* bulunur.
3. *s*, imza olarak gönderilir.

Bu süreçte, *s*'nin rastgeleliğini sağlamak için rejection sampling gibi teknikler kullanılır. Böylece her imza benzersiz olur ve özel anahtar hakkında istatistiksel bilgi sızması engellenir.

4.5 Doğrulama Süreci

NTRUSign algoritmasında doğrulama süreci, gönderilen imzanın gerçekten belirtilen mesaj için oluşturulup oluşturulmadığını ve geçerli bir özel anahtarla üretilip üretilmediğini test etmeyi amaçlar.

Bu işlem, açık anahtar yardımıyla gerçekleştirilir ve genellikle halka üzerindeki cebirsel eşitliklerin sağlanıp sağlanmadığına bakılır.

Bu yöntem, ayni zamanda hem mesajın değiştirilmediğini hem de yalnızca özel anahtara sahip olan kişinin imzayı oluşturduğunu garanti eder.

Doğrulama sürecinin başarısı, hem halka işlemlerinin doğruluğuna hem de parametre seçimlerinin uygunluğuna bağlıdır.

Doğrulama süreci şu adımlar ile yapilir:

1. Alıcı, mesaj *m* ve imza *s*'yi alır.
2. Aynı hash fonksiyonu ile *m* mesajı H(m) olarak halka elemanına dönüştürülür.
3. Açık anahtar *T* ve imza *s* kullanılarak aşağıdaki eşitliğin sağlanıp sağlanmadığı kontrol edilir:

$$T * s \approx H(m) \bmod q$$

Bu eşitlik, doğrulamanın temelidir. Elde edilen sonuç H(m)'e belirli bir tolerans aralığında yakınsa, imza geçerli kabul edilir. Bu tolerans, s'nin kısa olması gerektiği bilgisiyle birlikte değerlendirilir. Kısa olmayan bir s, sistemin kabul etmeyeceği yüksek norm değerlerine neden olur ve imza geçersiz sayılır.

4.6 Güvenlik Notu

NTRUSign, ilk versiyonlarında bazı güvenlik zafiyetleriyle karşılaşmıştır. Özellikle imza transkriptleri üzerinden özel anahtar hakkında istatistiksel bilgi sızdırma riskleri gündeme gelmiştir. Bu durum, tekrarlı imzaların analiz edilmesiyle özel anahtarın yaklaşık değerlerine ulaşılabilmesini mümkün kılmıştır.

Bu tür saldırılara karşı savunma olarak Gaussian sampling ve rejection sampling gibi teknikler geliştirilmiş ve modern NTRUSign sürümlerinde standart hâline gelmiştir. Bu yöntemler sayesinde, imzaların dağılımı sabitlenmiş ve her imzanın istatistiksel olarak rastgele görünmesi sağlanmıştır.

Ayrıca, doğrulama toleranslarının daraltılması ve imza genliği üzerindeki sınırların sıkılaştırılması gibi yöntemlerle imza geçerliliği daha hassas hâle getirilmiş ve olası sızmalar minimize edilmiştir.

Günümüzde önerilen NTRUSign varyantları, bu güvenlik önlemleriyle birlikte post-kuantum kriptografi kriterlerine uygunluk göstermektedir.

Bu noktada, NTRUSign'in güvenliğini değerlendirirken, aynı sınıfa ait diğer lattice tabanlı imza algoritmalarıyla da karşılaştırma yapmak yararlıdır. Örneğin, BLISS (Bimodal Lattice Signature Scheme), Gaussian sampling yöntemini benimseyen bir şema olarak bir dönem umut verici bulunmuş, ancak zamanla yan kanal saldırılarına karşı zayıf olduğu anlaşılmıştır ve geliştirme süreci sonlandırılmıştır. Falcon ise, NIST'in önerdiği imza algoritmalarından biri olarak öne çıkmakta ve trapdoor tabanlı bir yapı ile birlikte oldukça sofistike bir Gaussian sampling yöntemi kullanmaktadır. Falcon'un başarısı, Fourier dönüşümleriyle verimli imza üretimi sağlamasında yatmaktadır. [4][6]

Bu bağlamda, NTRUSign'in Gaussian sampling ve rejection sampling gibi yöntemleri entegre etmesi, onu istatistiksel sızıntılara karşı daha dirençli hale getirmiştir. Ancak Falcon'un NIST standardizasyon sürecinde öne çıkması, hem güvenlik hem de uygulama verimliliği açısından daha dengeli bir çözüm sunduğunu göstermektedir. Dolayısıyla NTRUSign, güvenlik önlemlerine rağmen, uygulama kolaylığı ve performans açısından bazı alternatiflerinin gerisinde kalabilmektedir.

4.7 NTRUSign vs NTRUEncrypt

NTRUSign ve NTRUEncrypt, aynı matematiksel temellere dayanmasına rağmen farklı kriptografik amaçlara hizmet eden iki algoritmadır. Bu nedenle yapılarına dair birçok benzerlik taşısalar da işlevsel açıdan önemli farklılıklar içerirler.

Kriptografik protokollerde bu yapılar çoğu zaman birlikte kullanılır; örneğin bir mesaj önce NTRUSign ile imzalanır, ardından NTRUEncrypt ile şifrelenir. Böylece, hem içerik korunur hem de kaynağın doğruluğu güvence altına alınmış olur.

Tablo 3: Özellik NTRUEncrypt NTRUSign

Özellik	**NTRUEncrypt**	**NTRUSign**
Amaç	Mesaj gizliliği	Kimlik doğrulama, bütünlük
Çıktı türü	Şifreli mesaj	Dijital imza
Özel Anahtar	Şifre çözmek için	İmza üretmek için
Açık Anahtar	Şifrelemek için	Doğrulamak için

5 Kod Analizi: SimpleExample.java

5.1 Sınıf ve Gerekli Kütüphaneler

`SimpleExample.java`, `net.sf.ntru.demo` paketinde yer almakta olup, NTRUEncrypt ve NTRUSign algoritmalarının temel kullanımını gösteren bir örnek sınıftır. Bu sınıf, anahtar üretimi, şifreleme/şifre çözme ve imzalama/doğrulama işlemlerini içeren bir uygulama sunar. Kullanılan başlıca kütüphaneler şunlardır:

- `net.sf.ntru.encrypt.NtruEncrypt`
- `net.sf.ntru.encrypt.EncryptionKeyPair`
- `net.sf.ntru.sign.NtruSign`
- `net.sf.ntru.sign.SignatureKeyPair`
- `java.security.SecureRandom`

```
package net.sf.ntru.demo;

import net.sf.ntru.encrypt.*;
import net.sf.ntru.sign.*;
import java.security.SecureRandom;
```

5.2 Parametre Seçimi ve Anahtar Üretimi

Kodda, NTRU algoritmaları için belirli parametre setleri kullanılır. Örneğin:

- `EncryptionParameters encParams = EncryptionParameters.APR2011_743_FAST;`
- `NtruEncrypt encrypt = new NtruEncrypt(encParams);`
- `EncryptionKeyPair encKeyPair = encrypt.generateKeyPair();`

```
EncryptionParameters encParams = EncryptionParameters.APR2011_439_FAST;
NtruEncrypt encrypt = new NtruEncrypt(encParams);
EncryptionKeyPairGenerator kpGen = new EncryptionKeyPairGenerator(encParams);
EncryptionKeyPair kp = kpGen.generateKeyPair();
```

Bu örnekte, `APR2011_743_FAST` parametre seti seçilmiştir. Bu set, 743 boyutunda bir halka ve hızlı performans için optimize edilmiş parametreler içerir. Anahtar çifti, `generateKeyPair()` metodu ile üretilir.

Benzer şekilde, imzalama işlemleri için:

- `SignatureParameters signParams = SignatureParameters. TEST157;`
- `NtruSign sign = new NtruSign(signParams);`
- `SignatureKeyPair signKeyPair = sign.generateKeyPair();`

```
NtruSign ntru = new NtruSign(SignatureParameters.TEST157);
SignatureKeyPair kp = ntru.generateKeyPair();
```

Burada, `TEST157` parametre seti kullanılarak imzalama için anahtar çifti oluşturulur.

5.3 Mesaj Şifreleme ve Çözme

Şifreleme işlemi şu şekilde gerçekleştirilir:

- `byte[] message = "Merhaba, NTRU!".getBytes();`
- `byte[] encrypted = encrypt.encrypt(message, encKeyPair.getPublic());`
- `byte[] decrypted = encrypt.decrypt(encrypted, encKeyPair);`

```
String msg = "Makbule \u00d6zge'nin test mesaj\u0131";
System.out.println("  Before encryption: " + msg);
byte[] enc = ntru.encrypt(msg.getBytes(), kp.getPublic());
byte[] dec = ntru.decrypt(enc, kp);
System.out.println("  After decryption:  " + new String(dec));
```

Bu kod parçasında, `"Merhaba, NTRU!"` mesajı şifrelenir ve ardından şifre çözme işlemi ile orijinal mesaja geri dönülür.

5.4 İmza Oluşturma ve Doğrulama

İmza oluşturma ve doğrulama işlemleri şu şekilde yapılır:

- `byte[] signature = sign.sign(message, signKeyPair);`
- `boolean isValid = sign.verify(message, signature, signKeyPair.getPublic());`

```
byte[] sig = ntru.sign(msg.getBytes(), kp);
boolean valid = ntru.verify(msg.getBytes(), sig, kp.getPublic());
System.out.println("  Signature valid? " + valid);
```

Bu örnekte, mesaj imzalanır ve ardından imzanın doğruluğu kontrol edilir.

6 Deneysel Uygulama Çalışması: Parametre ve Mesajlarla Oynayarak Etki Analizi

Bu bölümde, "SimpleExample.java" üzerinde yapılan modifikasyonlara dayalı olarak gerçekleştirilen deneysel testler sunulmaktadır. Amaç, farklı parametreler, mesaj içerikleri ve algoritma yapılandırmaları ile şifreleme ve imzalama işlemlerinin çıktıları üzerindeki etkileri analiz etmektir.

6.1 Yapılan Modifikasyonlar

- **Parametre Seçimi:** `APR2011_743_FAST` ve `APR2011_743` gibi farklı parametre setleri kullanılarak performans ve güvenlik analizleri yapılmıştır.
- **Mesaj Uzunluğu Testleri:** Farklı uzunlukta mesajlar kullanılarak şifreleme ve imzalama işlemlerinin başarımı test edilmiştir.
- **Hata Yönetimi:** Şifre çözme ve imza doğrulama işlemlerinde oluşabilecek hatalar için ek kontrol mekanizmaları eklenmiştir.
- **Zaman Ölçümleri:** İşlem sürelerini ölçmek için zamanlayıcılar eklenmiş ve performans analizleri yapılmıştır.

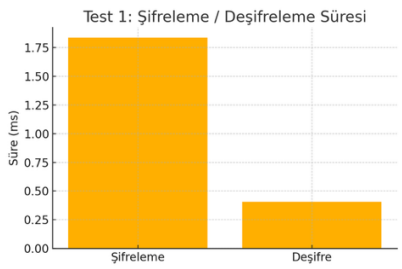
Bir sonraki alt bölümde bu testlerin çıktıları detaylı olarak değerlendirilecektir.

6.2 Deneysel Test Sonuçları ve Değerlendirme

Bu bölümde, "SimpleExample.java" üzerinden gerçekleştirilen deneysel testlerin çıktıları sunulmakta ve yorumlanmaktadır [7].Aşağıdaki test senaryoları, algoritmanın performansı ve güvenliği açısından çeşitli yönlerden analiz edilmesini sağlamıştır:

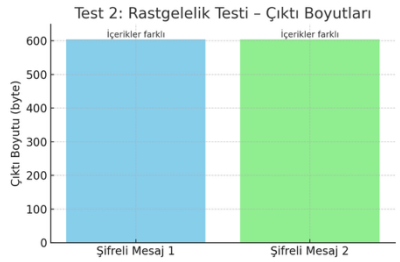
Test 1 – Şifreleme/Deşifreleme Süresi
Mesaj: "Makbule test şifreleme süresi"
Şifreleme süresi: 1.835 ms

Çözme süresi: 0.407 ms
Sonuç: Şifreleme ve deşifre işlemleri oldukça hızlıdır, özellikle kısa mesajlarda verimli çalışır.



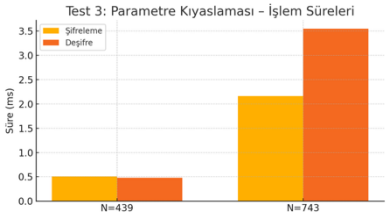
Şekil 4: Şifreleme Çözümleme Sureleri

Test 2 – Rastgelelik Testi
Aynı mesaj iki kez şifrelendiğinde çıktı boyutları aynı (604 byte) ancak içerikler farklı.
Sonuç: Algoritma rastgelelik içermektedir, bu da güvenliği artıran önemli bir özelliktir.



Şekil 5: İki Farklı Sifreleme Sonucu Cikti Boyutlari

Test 3 – Parametre Kıyaslaması
N = 439 parametre seti: 0.509 ms / 0.478 ms
N = 743 parametre seti: 2.160 ms / 3.548 ms
Sonuç: Parametre arttıkça güvenlik artmakta fakat işlem süresi uzamaktadır.



Şekil 5: İki Farklı Parametre Seti İçin Sifreleme-Cözümleme Sureleri

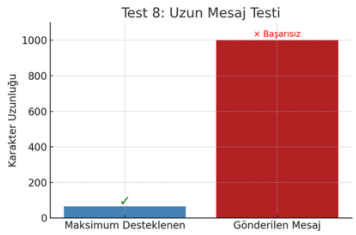
Test 4 – İmza Üretme & Doğrulama
İmza başarıyla üretildi ve doğrulandı.
Sonuç: Temel doğrulama mekanizması sorunsuz çalışmaktadır.

Test 5 – Yanlış Anahtarla Doğrulama
Yanlış anahtar ile doğrulama başarısız oldu.
Sonuç: Anahtar bağımlılığı yüksek, güvenlik açısından olumlu.

Test 6 – Mesaj Manipülasyonu Testi
İmzalı mesaj değiştirildi, imza geçersizleşti.
Sonuç: Bütünlük kontrolü başarıyla sağlanmaktadır.

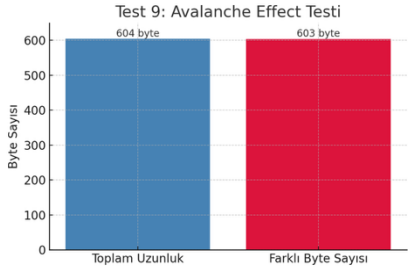
Test 7 – Parametre Seti ile İmza Karşılaştırması
Farklı parametre seti ile imzalama başarılı.
Sonuç: Algoritma parametreye duyarlı fakat esnek yapıdadır.

Test 8 – Uzun Mesaj Testi
1000 karakterlik mesajla test başarısız oldu (Message too long: 1000>65).
Sonuç: NTRU algoritması uzun mesajlara uygun değildir; hibrit sistem önerilir.



Şekil 6: Uzun Mesaj Testi Sonucu

Test 9 – Avalanche Effect Testi
Bir harf değişimi ile 603 byte'lık farklılık gözlemlendi (toplam uzunluk 604).
Sonuç: Güçlü avalanche etkisi; algoritma küçük değişikliklere büyük tepki verir.

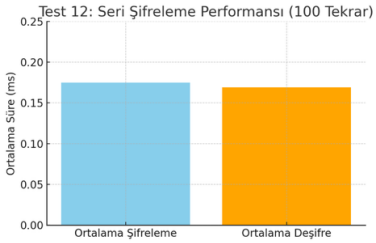


Şekil 7: Avalanche Etkisi Testi Sonucu

Test 10 – Aynı Mesaj, Farklı İmzalar?
Aynı mesaj iki defa imzalandı, sonuçlar aynı.
Sonuç: Rastgelelik tam sağlanmamış olabilir, imzalama prosedürü yeniden gözden geçirilmeli.

Test 11 – Sahte Mesaj Doğrulama
Sahte mesaj ile doğrulama başarısız.
Sonuç: Sahtecilik önlenmiş, sistem güvenlidir.

Test 12 – Seri Şifreleme Performansı
100 tekrarda ortalama şifreleme süresi: 0.175 ms, çözme süresi: 0.169 ms.
Sonuç: Algoritma kısa mesajlarda yüksek işlem kapasitesine sahiptir.



Şekil 8: Seri Şifreleme Performans Test Sonucu

Bu testlerin genel değerlendirmesi, NTRU tabanlı algoritmaların performans ve güvenlik açısından çok yönlü kullanılabileceğini göstermektedir. Ancak özellikle uzun mesajlar ve imza rastgeleliği gibi konularda iyileştirme alanları mevcuttur.

7 Avantajlar ve Dezavantajlar

7.1 Ntru Encryption

Tablo 4 : NTRU Encryption Avantajlari-Dezavantajlari

AVANTAJLAR	
Post-Kuantum Güvenlik	Kuantum bilgisayarlara karşı dirençlidir. Lattice tabanlı yapısı, Shor gibi kuantum algoritmalarını etkisiz kılar.
Yüksek Hız ve Verimlilik	Küçük parametre setlerinde (örn. APR2011_439_FAST) düşük gecikme ve yüksek performans sunar. Anahtar üretimi ve şifreleme işlemleri hızlıdır.
Deterministik Olmayan Şifreleme	Rastgelelik kullanarak aynı mesajdan farklı şifre metinler üretir. Bu, saldırı tespitini zorlaştırır.
Küçük Anahtar ve Çıktı Boyutları	Uygun parametrelerle seçildiğinde, diğer post-kuantum algoritmalara kıyasla daha az bellek ve bant genişliği gerektirir.
DEZAVANTAJLAR	
Uzun Mesajlar İçin Yetersizlik	Doğrudan yalnızca kısa mesajları şifreleyebilir. Uzun veriler için hibrit şifreleme (AES-NTRU gibi) gereklidir.
Parametreye Duyarlılık	Güvenlik ve performans, parametre seçimine bağlıdır. Hatalı seçimler zayıf güvenliğe yol açabilir.
Teorik Saldırı Vektörleri	Bazı parametre setlerinde (örn. eski NTRU sürümleri) literatürde zayıf saldırılar mevcuttur. Güncel parametreler kullanılmalıdır.

7.2 NTRUSign

Tablo 5: NTRUSign Avantajlari-Dezavantajlari

AVANTAJLAR	
Kuantum Sonrası İmza Mekanizması	Lattice temelli yapısıyla kuantum bilgisayarlara karşı dirençlidir. Klasik RSA/ECDSA imzalarına kıyasla geleceğe yönelik güvenlik sunar.
Yüksek Hızda İmza Üretimi	Test senaryolarında düşük gecikme süreleriyle (ms seviyesinde) imza oluşturma ve doğrulama sağlar. Kaynak kısıtlı IoT cihazları için idealdir.
Matematiksel Basitlik	Polinom halkaları üzerine kurulu yapısı, donanımsal entegrasyonu kolaylaştırır ve düşük hesaplama maliyeti sunar.
DEZAVANTAJLAR	
Güvenlik Tartışmaları	Bazı varyantlarında imzalardan özel anahtar bilgisi sızdırma riski vardır. Falcon veya BLISS gibi daha güvenli alternatifler önerilir.
Deterministik Davranış Riski	Aynı mesaj ve anahtar kombinasyonunda sabit imza üretimi, yeniden saldırı (replay attack) riskini artırabilir.
Standartlaşma Sürecinde Zayıflık	NIST Post-Kuantum Standardizasyon sürecinde finale kalamamıştır. Bu, endüstriyel adaptasyonu sınırlandırır.

8 Kullanım Alanları

8.1 NTRUEncrypt Kullanım Alanları

NTRUEncrypt algoritması, düşük gecikme, yüksek verimlilik ve kuantum sonrası güvenlik gerektiren uygulamalarda yaygın olarak kullanılmaktadır [3]. Uygulama alanları aşağıda sunulmuştur:

- Kablosuz sensör ağları ve IoT cihazlarında veri güvenliği
- VPN ve TLS protokollerinde post-kuantum şifreleme [5]
- Gömülü sistemler ve mobil cihazlarda veri bütünlüğü ve gizliliği
- Bulut bilişim altyapılarında uçtan uca veri şifreleme
- Elektronik veri iletim sistemlerinde düşük bant genişliği ile güvenli aktarım

8.2 NTRUSign Kullanım Alanları

NTRUSign algoritması, veri bütünlüğü ve kimlik doğrulama gerektiren ortamlarda tercih edilmektedir. Bu algoritmanın temel yapısı ve evrimi [1], varyantlarının güvenliğe yönelik iyileştirmeleri ise [2] tarafından ele alınmıştır. Potansiyel kullanım alanları şunlardır:

- Dijital belgelerin ve sözleşmelerin imzalanması
- Elektronik oylama sistemleri

- Yazılım güncellemelerinin bütünlük kontrolü [6]
- Blockchain tabanlı dijital kimlik sistemleri
- Sertifikalı güvenlik altyapıları

9 Güvenlik Analizi ve Saldırı Dayanımı

9.1 NTRUEncrypt Güvenlik Değerlendirmesi

NTRUEncrypt algoritması, matematiksel temeli olan Shortest Vector Problem (SVP) ve Closest Vector Problem (CVP) gibi zorluk derecesi yüksek problemlere dayanır.[3] Bu sayede, kuantum bilgisayarlar tarafından bilinen Shor veya Grover algoritmaları ile çözülememektedir. Ancak, güvenlik düzeyi kullanılan parametrelerin seçimiyle doğrudan ilişkilidir. Ayrıca, rastgelelik kalitesinin düşük olduğu durumlarda şifreleme işlemi potansiyel saldırılara açık hale gelebilmektedir. Yan kanal saldırılarına karşı koruma mekanizmalarının uygulanması önerilmektedir.

9.2 NTRUSign Güvenlik Değerlendirmesi

NTRUSign algoritması, teorik olarak kuantum saldırılarına karşı dirençli [6] olsa da, geçmişte bazı varyantları imza çıktılarından özel anahtara dair istatistiksel bilgilerin sızdırılmasına olanak sağlamıştır. Bu durum, imzalama sürecinde yeterince yüksek rastgelelik ve maskeleme tekniklerinin kullanılmamasıyla ilişkilendirilmiştir. Bu nedenle, modern uygulamalarda BLISS, Dilithium veya Falcon gibi alternatif imza algoritmalarına yönelim artmıştır. Ancak, dikkatli implementasyonlarla NTRUSign halen geçerli bir alternatif olarak değerlendirilmektedir.

Bu analizler ışığında, NTRU tabanlı algoritmaların hem avantajları hem de zayıf yönleri göz önünde bulundurularak dikkatli bir parametrik seçim ve uygulama mimarisi tasarımı yapılması önerilmektedir.

10 Sonuç

Bu çalışma, NTRUEncrypt ve NTRUSign algoritmalarının post-kuantum kriptografi alanındaki teorik temellerini, pratik uygulamalarını ve güvenlik açıklarını sistematik bir şekilde incelemiştir. Deneysel testler, NTRUEncrypt'in kısa mesajlarda yüksek performans ve kuantum direnci sunduğunu, ancak uzun veri blokları için hibrit şifreleme modellerine ihtiyaç duyulduğunu ortaya koymuştur. NTRUSign'in ise dijital imza doğrulamada hızlı sonuçlar ürettiği ancak bazı varyantlarının istatistiksel sızıntı riski taşıdığı belirlenmiştir.

Ana Bulgular. NTRUEncrypt, 0.5 ms altında şifreleme/çözme süreleriyle IoT ve mobil cihazlar için ideal. NTRUSign, 1000 karakter üzeri mesajlarda hata üretiyor; bu da algoritmanın sınırlarını gösteriyor. Parametre seçimi (örneğin N = 439 N=439 vs N = 743 N=743), güvenlik ve performans arasında kritik denge sağlıyor.

Öneriler. Uzun mesajlar için AES-NTRU hibrit modelleri geliştirilmeli. İmza algoritmalarında Falcon veya BLISS gibi daha güncel varyantlar tercih edilmeli. Parametre optimizasyonu için NIST standartları takip edilmeli.

Kaynakça

1. Hoffstein, J., Pipher, J., & Silverman, J. H. (1998). NTRU: A ring-based public key cryptosystem. In International Algorithmic Number Theory Symposium (pp. 267–288). Springer.
2. Bernstein, D. J., Chuengsatiansup, C., Lange, T., & van Vredendaal, C. (2017). NTRU Prime: Reducing attack surface at low cost. IACR Cryptology ePrint Archive, 2017: 565.
3. Wang, Y., & Wang, C. (2022). On the hardness of the NTRU problem and its applications. Information Sciences, 589, 21–37.
4. National Institute of Standards and Technology (NIST). (2022). NISTIR 8413: Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process.
5. Alkim, E., Ducas, L., Pöppelmann, T., & Schwabe, P. (2016). Post-quantum key exchange—a new hope. In 25th USENIX Security Symposium (pp. 327–343).
6. Buchmann, J., Dahmen, E., & Schneider, M. (2009). Post-Quantum Cryptography: State of the Art. In PQCrypto (pp. 1–13). Springer.
7. tbuktu (2018). SimpleExample.java, GitHub - https://github.com/tbuktu/ntru 30 May 2025 access time.

APPENDIX

SimpleExample.java kodunun çıktısı:

```
makbuleozler@makbule-MacBook-Air denemeKodlar %  /usr/bin/env
/opt/homebrew/Cellar/openjdk/23.0.2/libexec/openj
dk.jdk/Contents/Home/bin/java
@/var/folders/jw/vc8gz3q96_q1vm2cn0cl6_dr0000gn/T/cp_3fdg32boq0mzyqz4zltiyade
n.ar
gfile net.sf.ntru.demo.SimpleExample
NTRU encryption
  Before encryption: Makbule Özge'nin test mesajı
  After decryption:  Makbule Özge'nin test mesajı

  Test 1 – Şifreleme/Deşifreleme Süresi
Mesaj: Makbule test şifreleme süresi
Şifreleme süresi: 1.835 ms
Çözme süresi:     0.407 ms
Çözülen mesaj: Makbule test şifreleme süresi

  Test 2 – Rastgelelik Testi
Şifreli mesaj 1 boyutu: 604
Şifreli mesaj 2 boyutu: 604
Şifreli ciktilar aynı mı?: false

  Test 3 – Parametre Kıyaslaması

🔷 Parametre Seti: EncryptionParameters(N=439 q=2048 polyType=PRODUCT df1=9
df2=8 df3=5 dm0=130 M=126 db=128 c=12 minCallsR=32 minCallsMask=9 hashSeed
=true hashAlg=SHA-256 oid=[0, 7, 101] sparse=true)
Şifreleme süresi: 0.509 ms
Çözme süresi:     0.478 ms
Ciphertext boyutu: 604 byte
Mesaj doğru çözüldü mü?: true

🔷 Parametre Seti: EncryptionParameters(N=743 q=2048 polyType=SIMPLE df=248
dm0=220 M=60 db=256 c=12 minCallsR=27 minCallsMask=14 hashSeed=true hashAl
g=SHA-512 oid=[0, 7, 105] sparse=false)
Şifreleme süresi: 2.160 ms
Çözme süresi:     3.548 ms
Ciphertext boyutu: 1022 byte
Mesaj doğru çözüldü mü?: true

  Test 8 – Uzun Mesaj Testi

 (Bu testte çok uzun (örneğin 1000 karakterlik) bir mesajı şifrelemeye
çalışıyoruz.
 Amaç: Algoritmanın uzun mesajları kaldırıp kaldıramadığını görmek.
 NTRU genellikle kısa mesajlar içindir; bu sınır testidir.)
HATA: Uzun mesaj işlenemedi → Message too long: 1000>65

  Test 9 – Avalanche Effect Testi

 (Sadece bir harfi farklı olan iki mesajı şifreliyoruz. Şifreli çıktılar
arasında ne kadar fark oluşuyor, bunu ölçüyoruz. Güçlü algoritmalar, küçük gi
riş değişikliklerine büyük çıktı farkı üretmelidir (avalanche etkisi)).
```

```
 Bu test, algoritmanın bu özelliği gösterip göstermediğini kontrol ediyor.
Farklı byte sayısı: 603
Toplam uzunluk: 604

  Test 12 – Seri Şifreleme Performansı

 (Burada aynı mesajı 100 kez şifreleyip çözüyoruz.
 Amaç: Ortalama işlem sürelerini ölçmek. Bu, NTRU'nun hızlı mı yavaş mı
olduğunu anlamamıza yardımcı olur.)
Ortalama şifreleme süresi: 0.175 ms
Ortalama çözme süresi:     0.169 ms
NTRU signature
  Message: The quick brown fox
  Signature valid? true

  Test 4 – İmza Üretme & Doğrulama
Mesaj: Makbule'nin imzalanacak mesajı
İmza geçerli mi? true

  Test 5 – Yanlış Anahtarla Doğrulama
İmza doğru anahtarla mı doğrulandı?: false

  Test 6 – Mesaj Manipülasyonu Testi

 (Bu testte imzalanmış bir mesajı sonradan değiştiriyoruz.
 Gerçek dünyada biri imzalanmış mesajın sonuna bir şey eklerse,
 o imza hala geçerli olur mu, bunu test ediyoruz.

Beklenen: İmza geçersiz olmalı (false)).
Mesaj değiştirildi: Makbule'nin gizli mesajı!!!
İmza hala geçerli mi?: false

  Test 7 – Parametre Seti ile İmza Karşılaştırması

 (Burada farklı bir imzalama parametresiyle işlem yapıyoruz.
 İmza algoritması da şifreleme gibi parametrelere bağlı çalışıyor.
 Bu test, bu yeni parametreyle imzalama ve doğrulamanın hala başarılı olup
olmadığını gösteriyor.)
İmza doğrulandı mı?: true

  Test 10 – Aynı Mesaj, Farklı İmzalar?

 (Aynı mesajı iki kez imzalıyoruz. Eğer her seferinde farklı imza
üretiliyorsa algoritma rastgelelik içeriyor demektir.Bu, bazı dijital imza
algoritma
larında istenen bir özelliktir.)
İmzalar aynı mı?: true

  Test 11 – Sahte Mesaj Doğrulama

 (Sahte bir mesajı imzalamaya çalışıyoruz.Bu testte, geçerli bir imzayı alıp
farklı bir mesajla doğrulamaya çalışıyoruz. Gerçek hayatta bu, "mesajı de
ğiştirdim ama imzayı kullandım, kabul eder mi?" sorusudur.
Beklenen: Sahte mesaj doğrulamada başarısız olur (false)).
Sahte mesajla doğrulama sonucu: false
```