TÜBİTAK

Açık Anahtarlı Kriptografi için Verimli Algoritmaların Geliştirilmesi

Program Kodu: 1001

Proje No: 115R289

Proje Yürütücüsü: Doç. Dr. Murat CENK

Araştırmacılar: Prof. Dr. Ferruh Özbudak

Bursiyerler:
Shoukat Ali
Murat Demircioğlu
Şeyma Fetvacı
Kübra Kaytancı
İrem Keskinkurt Paksoy
Buse Taşcı
Halil Kemal Taşkın
Hasan Bartu Yünüak

TEMMUZ 2018
ANKARA

ÖNSÖZ

TÜBİTAK tarafından desteklenen 115R289 numaralı "Açık Anahtarlı Kriptografi için Verimli Algoritmaların Geliştirilmesi" isimli proje açık anahtar kriptografide kullanılan aritmetik işlemlerin daha verimli olacak şekilde iyileştirilmesi üzerinedir. Yapılan çalışmalar üç ana başlık altında toplanmıştır. Bunlar, eliptik eğri işlemlerinde iyileştirmeler, polinom çarpmasında iyileştirmeler ve modüler üst almada iyileştirmelerdir. Proje kapsamında, Montgomery ve Edwards tipindeki eliptik eğrileri kullanan kriptosistemlerin işlemsel karmaşıklığının iyileştirilmesi, polinom çarpma işlemlerinde kullanılan çarpma sayısının düşürülmesi ve modüler üst alma işleminin hızlandırılması üzerine araştırmalar yapılmıştır. Verdiği destekten dolayı TÜBİTAK'a teşekkür ederiz.

Proje kapsamında üç dergi makalesi yayınlanmış ve üç konferans bildirisi sunularak konferans bildiriler kitapçığında yer almıştır. Bunlara ek olarak iki poster sunulmuştur. Projeden destek gören bir doktora öğrencisi ile iki yüksek lisans öğrencisi mezun olmuştur. Hali hazırda üç adet makale çalışmamız yazım aşamasındadır. Proje kapsamında elde edilen sonuçlar üzerine yapılan yayınlar şunlardır:

- H. K. Taşkın ve M. Cenk, "Speeding up Curve25519 using Toeplitz Matrix-vector Multiplication", Fifth Workshop on Cryptography and Security in Computing Systems, basım aşamasında, 2018.
- M. Cenk, "Karatsuba-Like Formulae and Their Associated Techniques", Journal of Cryptographic Engineering, basım aşamasında, 2018, https://doi.org/10.1007/s13389-017-0155-8.
- S. Ali ve M. Cenk, "Faster Residue Multiplication Modulo 521-bit Mersenne Prime and an Application to ECC", IEEE Trans. on Circuits and Systems 65(8): 2477-2490, 2018.
- M.B. İlter ve M Cenk, "Efficient Big Integer Multiplication in Cryptography", International Journal of Information Security Science 6 (4), 70-78, 2017. (Genişletilmiş versiyon)
- M.B. İlter ve M. Cenk, "Efficient Big Integer Multiplication in Cryptography", ISCTurkey 2017 Conference Proceedings, 8-13, 2017.

Yazım aşamasında olan ve yakında ibraz edilecek olan çalışmalarımız:

- M. Cenk, T. Banerjee ve A. Hasan, "Faster Multiplication in the Quadratic Extension of Prime Fields and Its Applications to SIDH".

- H. B. Yünüak, M. Cenk, V. Dimitrov ve A.Hasan, "Faster modular exponentiation".
- S. Baloğlu, M. Cenk ve C. Çalık, "On the New Upper Bounds for Computing Some First Terms of Product of Polynomials".

Proje yürütücüsünün danışmanlığında proje kapsamında desteklenen bursiyerlerin tezleri:

- "Faster Residue Multiplication Modulo 521-bit Mersenne Prime and Application to ECC", Shoukat Ali, ODTÜ Uygulamalı Matematik Enstitüsü Kriptografi Programı, Doktora, Eylül 2017.
- "Homomorphic Encryption Based on the Ring Learning with Errors (RLWE) Problem", İrem Keskinkurt, ODTÜ Uygulamalı Matematik Enstitüsü Kriptografi Programı, Yüksek Lisans, Eylül 2017.
- "Modular Exponentiation Methods in Cryptography", Hasan Bartu Yünüak, ODTÜ Uygulamalı Matematik Enstitüsü Kriptografi Programı, Yüksek Lisans, Eylül 2017.

Poster olarak sunulan çalışmalarımız:

- S. Ali ve M. Cenk, "Faster Residue Multiplication Modulo 521-bit Mersenne Prime and Application to ECC", Workshop dedicated to the 15th Anniversary of the Foundation of Institute of Applied Mathematics, 2017.
- H. B. Yünüak ve M. Cenk, "Modular Exponentiation Methods in Cryptography", Workshop dedicated to the 15th Anniversary of the Foundation of Institute of Applied Mathematics, 2017.

Doç. Dr. Murat Cenk

Ankara, Temmuz 2018

İçindekiler

Şekiller

Tablolar

ÖZET

Projenin genel amacı, kriptografide sıklıkla kullanılan modüler üst alma, polinom çarpması ve eliptik eğriler üzerindeki işlemlerin karmaşıklığını iyileştirecek geliştirmelerin yapılması ve elde edilecek yeni algoritmaların çeşitli platformlar üzerinde gerçeklenmesidir. Bu çalışmalar sonucunda modüler üst alma, eliptik eğri aritmetiği ve polinom çarpma işlemlerinde iyileştirmeler elde edilmiştir. Çalışmalar kapsamında P-521, E-521 ve Curve25519 eğrileri üzerindeki işlemler Toeplitz matris vektör çarpımları (TMVÇ) kullanılarak hızlandırılmıştır. Eliptik eğrilerin üzerinde tanımlandığı ve eleman sayıları 521 ve 255 bitlik asal sayılar olan cisimlerde çarpma işlemleri için yeni TMVÇ algoritmaları tasarlanmış ve bu algoritmaların sağladığı iyileştirmeler teorik olarak gösterilmiştir. Yapılan gerçeklemeler ile teorik çıkarımlardaki iyileştimeler pratikte de gözlemlenmiştir. Diğer taraftan polinom çarpma işleminin iyileştirilmesi için arama algoritmalarının verimi üzerine çalışmalar yapılmıştır. Polinomun terim sayısı arttıkça arama uzayı oldukça büyüdüğü için, çarpım polinomunun tüm terimlerini hesaplamak yerine, n terimli iki polinomun çarpmının ilk n teriminin hesaplanması üzerine analizler yapılmıştır. Böylece arama uzayının boyutu düşürülmüş ve Çinli Kalan Teoremi ile polinom çarpımı için algoritmalar elde edebilme olanağı sağlanmıştır. Diğer bir yaklaşım ise n terimli iki polinomum ilk ℓ teriminin hesaplanmasıdır $(n+1 \leq \ell \leq 2n-2)$. Ayrıca, bu yaklaşımda arama uzayının boyutunun düşürülmesi için ikili doğrusal formların simetriklerinin alınması ve bazı terimlerin elenmesi yöntemleri kullanılmıştır. Bu yaklaşımlar arama uzayının boyutunu belirgin şekilde azaltmıştır. Ek olarak interpolasyon metodunda hesaplanacak noktalar dikkatlice seçilerek, süper singüler izojen bazlı kuantum sonrası kriptografide kullanılan $\mathbb{F}_{p^2}$ çarpma işlemi ve büyük sayıların çarpımları hızlandırılmıştır. Proje kapsamında çalışılan diğer bir konu olan modüler üst alma işleminin hızlandırılması için, literatürdeki küp şeker algoritması incelenmiştir. Bu algoritma, en küçük toplam zinciri ve karma üst alma metotları ile birlikte kullanılmıştır. Ayrıca, sonuçların daha da hızlandırılması adına, n bitlik bir tamsayının küp alma işleminden sonra $3n$ olan boyutunu indirgemek için kullanılan Barett metodu değiştirilmiş ve böylece teorik olarak işlem karmaşıklığında iyileştirmeler yapılmıştır.

Anahtar kelimeler: Kriptografik hesaplamalar, polinom çarpımı, tamsayı çarpımı, eliptik eğri kriptografisi, modüler üst alma, RSA

ABSTRACT

The primary aim of this project is to develop algebraic techniques for improving the complexity of the operations that are widely used in cryptography such as modular exponentiation, polynomial multiplication, arithmetic on elliptic curves and to implement these algorithms on various platforms. As a result of the studies, improvements on modular exponentiation, polynomial multiplication and elliptic curve arithmetic are obtained. Within the scope of studies, the arithmetic on the curves P-521, E-521 and Curve25519 are accelarated by using Toeplitz matrix vector product (TMVP). For the multiplication in 521 and 255 bit prime fields on which the elliptic curves are defined, new TMVP algorithms are designed and the improvements that these algorithms provide are proved theoretically. The implementations show that the improvements can also be observed in practice. On the other side, to improve the polynomial multiplication, studies are focused on the efficiency of the search algorithms. As the number of the terms of the polynomials increases the size of the search space grows so instead of computing all the terms, computing first n terms of the product of two n term polynomials is analyzed. By this, the size of the search space decreases and this makes it possible to develop new polynomial multiplication algorithms using the Chinese remainder theorem. Another approach is to compute the first ℓ terms of the product of two n term polynomials. $(n+1 \leq \ell \leq 2n-1)$. Moreover, in this approach, to reduce the size of the search space, symmetric bilinear forms and elimination of some terms are used. These methods decrease the size of the search space significantly. In addition, by choosing the evaluation points carefully in the interpolation method, the multiplication over $\mathbb{F}_{p^2}$ that is used for supersingular isogeny based post quantum cryptography and large integer multiplicaitons are accelerated. To speed up modular exponentiation which is another subject studied in this project, the sugar cube algorithm is examined. Sugar cube algorithm is combined with the addition chains and hybrid exponentiation methods. Moreover, to speed up the operations more, the Barrett reduction method for reducing the $3n$ bit size of the cube of a n bit integer is modified and by this the computational complexity is improved theoretically.

Keywords: Cryptographic computations, polynomial multiplication, integer multiplicaiton, elliptic curve cryptography, modular exponentiation, RSA

1 GİRİŞ

Dijital cihazlar günlük yaşamın ayrılmaz bir parçası haline gelmiştir. Genel amaçları günlük hayatı kolaylaştırmak olan bu cihazlar aynı zamanda birçok kişisel/gizli bilginin işlenmesi ve/veya saklanması amacıyla da kullanılmaktadır. Örneğin; bir web uygulamasına kullanıcı ve parolayla giriş yapmak, çevrimiçi bankacılık işlemleri gerçekleştirmek, uzaktan bir sunucuya erişmek gibi işlemler sırasında aradaki iletişim güvenli bir şekilde sağlanmalıdır. Bu noktada gizlilik ve güvenliğin sağlanması için kriptografik çözümler devreye girmektedir. Sezar şifreleme sisteminden günümüze kadar kriptografi insanlık tarihinde yer almıştır. 20. yüzyıl başlarına kadar basit kriptografik sistemler kullanılmaktayken, özellikle II. Dünya Savaşı sırasında çok daha karmaşık sistemler geliştirilmiştir. Buna en iyi örnek Almanların geliştirdiği Enigma cihazıdır. O zamanlarda yapılan şifrelemeler küçük veriler için bile uzun zaman alabilmekteydi. Oysa ki günümüz teknoloji çağında gerçekleşebilecek gecikmeler büyük sorunlara sebep olabilmektedir. 20. yüzyılın son çeyreğinde başlayıp 21. yüzyıl başında büyük bir ivme kazanan bilgisayar teknolojileri sayesinde şifreleme işlemleri çok daha büyük boyutlu veriler üzerinde daha kısa sürelerde gerçekleştirilebilmektedir. Her ne kadar modern bilgisayarlar kısa sürede çok büyük işlemler yapılmasına imkan sağlasa da, kullanılan kriptografik sistemlerin dizaynı ve içerdiği işlemler istenmeyen gecikmelere sebep olabilmektedir. Günümüzde kriptografik bir sistem tasarlanırken yalnızca güçlü bilgisayarlar değil, standart ev bilgisayarlarında, cep telefonlarında, PDA vb. nispeten daha zayıf donanıma sahip cihazlarda da çalışacağı göz önünde bulundurulmaktadır. İşlem gücüne sahip bir cihazda programın çalışma süresini belirleyen problem boyutu, kullanılan algoritma, bellek erişim hızı, işlemci hızı, işlemci sayısı, compiler/linker optimizasyonu gibi birçok etken vardır. Bu etkenlerden donanım ile ilgili maddelere müdahale ederek bir iyileştirme sağlamak oldukça güçtür. Problem boyutunun sabit olduğu da göz önünde bulundurulursa, geriye kullanılan algoritmada iyileştirmeler yapmak kalmaktadır. Örneğin; ilkokulda öğretilen metot ile iki büyük sayıyı çarpmak yerine Karatsuba çarpmasını kullanmak büyük bir kazanım sağlamaktadır.

Bu projenin temel amacı, Karatsuba örneğinde olduğu gibi özellikle kriptografi ve bilgisayar cebirinde önemli bir yer tutan işlem ve algoritmalarda zaman karmaşıklığı adına iyileştirmeler yapmaktır. Yapılan çalışmaları üç ana başlık altında toplanmıştır. Eliptik eğri işlemlerinde iyileştirmeler, polinom çarpmasında iyileştirmeler ve modüler üst alma algoritmasında iyileştirmelerdir.

Eliptik eğri işlemleri, anahtar değişim protokollerinde oldukça fazla kullanılır. Anahtar değişim problemi kriptografi için oldukça önemli problemlerden bir tanesidir. Sonlu cisimler üzerinde tanımlı anahtar değişim protokolleri güvenlik sebepleri ile giderek daha büyük boyutlu uzayda tanımlan-

maları dolayısıyla giderek daha yavaş çalışır hale gelmektedirler. Bu noktada eliptik eğriler kriptografi için önemli bir dönüm noktası olmuştur. Eliptik eğriler anahtar değişimi açık anahtarlı kriptografi'nin vazgeçilmez yapıtaşlarındır. Proje kapsamında özellikle çok çekirdekli 32-bit ve 64-bit işlemci mimarisine sahip platformlarda eliptik eğri tabanlı anahtar değişim sistemlerinin hem aritmetik olarak hızlandırılmasını sağlayan cebirsel çalışmalar hem de elde edilecek yeni yöntemlerin ve algoritmaların etkin şekilde gerçeklenmesi üzerine çalışmalar yapılmıştır. Projedeki iş paketlerinden 4, 5, 6, 7 ve 8. paket eliptik eğri çalışmaları ile ilgilidir. Bu çalışmalar sonucunda elde edilen yöntemler ve bulgular 3. ve 4. bölümlerde verilmiştir. Bulunan sonuçlar 32-bit ve 64-bitlik mimarilerde gerçekleştirilmiştir. İş paketleri arasında olan 8-bit ve 16-bit mimarilerdeki gerçekleştirmeler, 32-bit ve 64-bitlik mimarilerde yapılanlara oldukça benzemektedir fakat küçük boyutlu bu platformlardaki gerçekleştirmeler henüz bitirilmemiştir. Projenin bu bölümünde elde edilen metotlar ve bunların genel mimariler üzerinde verimli bir şekilde gerçekleştirilmesi, bu bölümle ilgili hedeflerin büyük bir kısmı gerçekleştirilmiştir sonucunu vermektedir. Projenin bu bölümü ile ilgili yapılan yayınlar şunlardır: (Ali 2017; Ali ve Cenk 2018a; b; Cenk, Banerjee, vd. 2018; Taskin ve Cenk 2018).

Diğer ana konu polinom çarpmasında iyileştirmelerdir. Polinom çarpmasının kullanıldığı birçok alan mevcuttur. Bu alanlar arasında tamsayı aritmetiği algoritmaları, sonlu cisim aritmetiği ve bilgisayar cebir sistemleri sayılabilirler. Sıkça kullanılan bir işlem olduğu için polinom çarpmasının zaman karmaşıklığının da dahil olduğu sistemi aksatmayacak seviyelerde olması beklenmektedir. Okullarda öğretilen klasik çarpma metoduyla yapılan tam sayı ve polinom çarpma işlemleri nispeten yavaş çalışmaktadır. Bu alanda yapılan akademik çalışmalar sonucunda Karatsuba ve Toom-Cook benzeri çeşitli formüller ortaya konmuş ve hesaplamalarda bunların kullanımı yaygınlaşmıştır. Projenin amaçlarından biri de polinom çarpmasını daha da hızlandıracak formüller geliştirmek üzere çalışmaların yapılmasıdır. Bu bağlamda literatürdeki güncel yöntemler incelenmiş ve bu yöntemlerin daha verimli gerçeklemeleri üzerine çalışmalar yapılarak arama algoritmalarında arama uzayının düşürülmesi ve interpolasyon yaklaşımlarından elde edilen algoritmaların özyinelemeli yaklaşımlarda karma (hybrid) kullanılmasıyla iyileştirmelerin elde edilmesi sonucuna ulaşılmıştır. Projenin iş paketlerinden 1, 2 ve 3 polinom çarpmaları üzerine olup bu iş paketlerinde belirlenen hedeflere ulaşılmıştır. Buradaki geliştirilen metotlar ve bulgular raporun 3. ve 4. bölümünde sunulmuştur. İlgili yayınlar: (Baloglu vd. 2018; Cenk 2018; Ilter ve Cenk 2017a; b; Keskinkurt 2017)

Son olarak modüler üst alma algoritmasında iyileştirmeler üzerine çalışılmıştır. Asimetrik anahtarlı kripto sistemlerde modüler üst alma temel bir işlem olarak sıklıkla kullanılmaktadır. Günümüzde güvenlik gerekçeleri ile bu sistemlerdeki modüler üst alma işlemi büyük sayılar üz-

erinden yapılmaktadır. Büyük sayılarla yapılan modüler üst alma için gerekli çalışma süresi ise oldukça uzundur. Bu nedenle seçilen metot, algoritmanın verimli bir şekilde çalışması noktasında kritik öneme sahiptir. Modüler üst almak için zaman içerisinde birçok metot tanımlanmıştır. Bu metotlar üstlü sayıların temel özelliklerine dayanan yöntemlerdir. Başlangıçta kullanılan yöntemler, hesaplanacak kuvvetin 2 tabanında genişletilmesine dayanmaktaydı. Ancak zaman içerisinde üstü alınacak olan sayıların büyümesi, kullanılan metotlarda iyileştirme yapma zorunluluğu doğurmuştur. Yapılan iyileştirmelere bakıldığında ise genellikle aynı metodun 2'nin kuvvetleri ile kullanılmasına dayandığı görülmektedir. Modüler üst alma yöntemlerinde iyileştirmeler yapılırken diğer sayılar dikkate alınmamıştır. Çünkü diğer sayılarla kuvvet almak için ihtiyaç duyulan işlem sayısı hesaplandığında daha fazla işlemin gerekli olduğu görülmüştür. Dolayısıyla bu sayılar için herhangi bir iyileştirme sağlanamayacağı düşünülerek göz ardı edilmiştir. Örneğin, 3 ve 3'ün kuvvetleri küp hesaplamanın maliyetinden dolayı hiç kullanılmamıştır. Bu projede küp algoritması ile modüler üst alma metotlarındaki iyileştirme çalışmalarına alternatif yaklaşımlar sunmuştur. Böylece hiçbir verimlilik elde edilemeyeceği düşünülerek uygulama dışı bırakılan yöntemlerin tekrar değerlendirmeye alınabileceği gösterilmiştir. Ayrıca elde edilen sonuçların kriptografik algoritmalarda etkin bir şekilde kullanılabileceği gösterilmiştir. Projedeki 9, 10, ve 11. iş paketleri bu konuyla ilgilidir ve buradaki hedeflere ulaşılmıştır. Bu konu ile ilgili yöntem ve bulgular 3. ve 4. bölümde verilmiştir. Bu çalışmalar ile ilgili yapılan ve hazırlanan yayınlar (Yunuak ve Cenk 2018; Yunuak, Cenk, vd. 2018) dır.

2 LİTERATÜR ÖZETİ

Kriptografi'nin en önemli problemlerinden birisi anahtar değişim problemidir. Anahtar değişim problemi, güvensiz bir ortam üzerinden haberleştiği varsayılan iki tarafın (bu taraflar kişiler ya da bilgisayarlar olabilir) sadece kendileri tarafından bilinecek ve diğer kişiler tarafından elde edilmesi mümkün olamayacak ortak bir gizli değer üzerinde anlaşma problemi olarak tanımlanmaktadır. Günümüzde, açık anahtarlı kriptografi yöntemleri ile bu problemlere çözüm getirilmektedir. Kullanılan en yaygın yöntem Diffie-Hellman anahtar değişim protokolü (Diffie ve Hellman 1976) olarak bilinmektedir. Bu protokol sayesinde, birbiri ile sadece güvensiz kanal üzerinden haberleşebilen iki taraf, yine bu güvensiz kanal üzerinden sadece kendilerinin bildiği bir ortak gizli bilgi (anahtar vb.) üzerinde anlaşmaktadır. Üzerinde anlaşılan bu ortak gizli bilgi, simetrik şifrelemede gizli anahtar olarak kullanılmaktadır. Kriptoanaliz çalışmaları ve teknolojinin ilerlemesiyle artan işlemsel hesap gücü ile birlikte halihazırda kullanılan Diffie-Hellman anahtar değişimi algoritmaları olası saldırılara dayanıklı olması amacı ile giderek daha büyük boyutlu sonlu cisimler üzerinde çalıştırılmaktadır. Günümüzde en az 2048-bit'lik sonlu cisimler üzerinde bu işlemler gerçekleştirilmektedir. Dolayısıyla, boyutun artması ile birlikte bu algoritmaların da çalışma hızları yavaşlamaktadır. Günlük yaşantımızda cep telefonları, online bankacılık işlemleri, uydu yayınları, web siteleri bağlantıları vb. gibi neredeyse her yerde kriptografik algoritmaların kullanıldığı gözönüne alındığında bu işlemlerin yeterince hızlı olması önem arz etmektedir. Bu bağlamda, eliptik eğri tabanlı sistemler (Koblitz 1987; Miller 1985) kriptografi için yeni bir dönüm noktası olmuştur. Bu sayede, büyük boyutlu cebirsel halkalar üzerinde yapılan işlemlerin, güvenlik seviyesi aynı kalmak üzere, eliptik eğriler üzerinde tanımlı daha küçük boyutlu cebirsel gruplar ile yapılması sağlanmıştır. Bu hem işlemsel hızın artmasını hem de bellek kullanımının azalmasını ve depolanan verinin boyutunun daha küçük olmasını sağlamıştır. Halihazırda bir çok kriptografik protokolün ve sistemin eliptik eğri tabanlı sistemlere geçiş yaptığı ya da desteklemesi için gerekli güncellemelerinin yapıldığı değerlendirilmektedir. Eliptik eğrilerin düzgün kullanımı için NIST tarafından uluslararası standartlar (NIST 2013) belirlenmiştir. Ancak, çeşitli kriptoanaliz yöntemleri ile bu standartlarda tanımlanan eliptik eğrilerin bazı zaafiyetlerinin olduğu tespit edilmiştir (Shumow ve Ferguson 2007), bununla birlikte çeşitli akademisyenler tarafından bağımsız olarak eliptik eğri tabanlı sistemler önerilmiştir. Bunlara örnek olarak Curve25519 (Bernstein 2006) verilebilir. Günümüzde Apple firması dahil birçok büyük firmanın yanısıra çeşitli kurumlar ve yazılımlar tarafından Curve25519 algoritması kullanılmaktadır (IANIX 2017). Ayrıca, NIST, Curve25519'un standartlar kapsamına alınması için gerekli çalışmaların başlatıldığını duyurmuştur (NIST 2017). Diğer bir yandan, günümüz teknolojisi ele alındığında, taşınabilir sistemlerin (akıllı telefonlar, tabletler vb.) kullanım oranının hızla arttığı

tartışılmaz bir gerçektir. Bununla birlikte, “Nesnelerin interneti” kavramının hayatımıza girmesi ile çevremizdeki her cihazın (sensörler aracılığıyla) bilgi sağladığı, birbiri ile konuşabildiği ve internete bağlandığı bir geleceğe doğru gittiğimiz şüphesizdir. Bu sistemler üzerinde de kriptografik işlemlerin yapılması gerekliliği tartışılmazdır. Ancak, taşınabilir yapıları itibariyle bu sistemlerin işlem gücü, bellek kapasiteleri ve güç tüketimleri kritik öneme sahiptir. Dolayısıyla, bu sistemler üzerine çalışacak kriptografik işlemlerin de bu sistemlere özel olarak tasarlanması veya uyarlanması gerekmektedir.

Günümüz teknolojisi ele alındığında taşınabilir sistemlerde 64-bit mimariye sahip işlemcilerin kullanılmaya başlandığı görülmektedir. Bu noktada, yukarıda benzer şekilde bahsedildiği üzere 64-bit mimari üzerinde koşturulacak kriptografi algoritmalarının da bu mimarilere özel olarak tasarlanması ya da uyarlanması gerekmektedir. Ayrıca, çok çekirdekli işlemci yapıları ile tek bir fiziksel işlemci üzerinde paralel hesaplama yapmak mümkün hale gelmiştir. Bu bağlamda, bahsi geçen kriptografik algoritmaların iyileştirilmesi noktasında çok çekirdekli işlemciler üzerinde koşturulmaya uygun hale getirilmesi ve bu sayede başta hız olmak üzere algoritmanın çalışmasının iyileştirilmesi önem arz etmektedir. Mevcut algoritmaların tasarımlarının bu çok çekirdekli yapıya uygun olacak şekilde yeniden tasarlanması ya da halihazırdaki sistemlerin uyarlanması hala önemli bir çalışma alanıdır. Değerlendirilmesi önem arz eden bir diğer konu ise “Nesnelerin interneti” olarak tanımlanan sensör ağları sistemidir. “Nesnelerin interneti” kavramı ile birlikte, sensörler hayatımızın her alanına dahil olmaktadırlar. Örneğin, akıllı ev tasarımlarında evin her bölgesinin sıcaklığı, evdeki her priz ve lambanın durumu gibi bilgiler uygun konumlara yerleştirilen sensörler ile takip edilmekte ve merkezi bir sunucuya gönderilmektedir. Yapısı itibariyle bu sensörler pil ile çalışan, kablosuz haberleşme yapan, mikrokontrolcüler ile programlanan küçük elektronik sistemlerdir. Sensörlerin iletişiminin güvenliğinin gerekliliği şüphesiz önem arz etmektedir. Bu bağlamda da bu sensörler üzerinde kriptografik algoritmaların çalıştırılması kritik bir problemdir. Özellikle, bahsi geçen sensör iletişiminde Açık Anahtarlı Kriptografi kullanmak bu alanda oldukça yenilikçi çalışmalar gerektirmektedir. Mikrokontrolcüler üzerinde Eliptik eğri tabanlı kriptografi algoritmalarının çalıştırılabilmesi bu problemi büyük ölçüde çözmektedir. Ancak, yapıları itibariyle mikrokontrolcüler işlem gücü, bellek kullanımı ve güç tüketimi açısından çok kısıtlı oldukları için eliptik eğri tabanlı kriptografi algoritmalarının etkin şekilde gerçeklenmesi önemli bir çalışma alanı olmuştur.

Polinom çarpması özellikle kriptografik algoritmalarda sıklıkla kullanılmaktadır. Bu kadar sık kullanılan bir işlem olarak, işlem karmaşıklığında yapılacak olan herhangi bir iyileştirmenin uygulanabilirlik noktasında sağlayacağı katkının yanı sıra akademik anlamda da önemi büyüktür. Polinom çarpması ile ilgili literatürde birçok çalışmanın bulunmasına rağmen bu çalışmaların hiçbiri polinom çarpmasına dair genel bir çözüm sunmamaktadır. Bu çalışmalar genellikle terim sayısı

belirli polinomlar için spesifik çözümler sunmaktadırlar. Örneğin; iki terimli iki polinom çarpmasında klasik çarpma yöntemi uygulanırsa dört tane tam çarpım yapmak gerekecektir. Oysaki (Karatsuba 1963) bu işlemin 3 tam çarpma ile yapılabileceğini göstermiştir. (Montgomery 2005) ise n=5,6 veya 7 olmak üzere iki tane n-terimli polinomun çarpımında kullanılabilecek bir formül önermiştir. Sonrasında bu çalışmayı geliştirmek üzere (Fan ve Hasan 2007b) Çinli Kalan Teoremini kullanmışlardır. Birçok durumda ise Toom'un çalışması (Toom 1963) daha iyi sonuçlar vermektedir. (Kaminski ve Bshouty 1989) yaptıkları çalışmada ise polinom çarpmaları için bir limit ortaya koymuşlardır. (Oseledets 2011; 2008) ise yaptığı çalışmalarda Karatsuba-benzeri formüller oluşturmak için yeni metotlar önermiştir. (Cenk ve Özbudak 2008; 2009; 2011; 2010) yaptıkları çalışmalar ile farklı karakteristiğe sahip sonlu cisimlerde tam sayı ve polinom çarpmaları konusunda yeni algoritmalar oluşturmuşlardır. Yakın zamanlı bir çalışmada (Barbulescu vd. 2012), Karatsuba ve Strassen formüllerini açıklamak için "bilineer rütbe problemi"ni tanımlanmıştır. Geliştirdikleri metot sayesinde literatürde bulunan çeşitli formüllerin optimal olduğunu göstermişlerdir. Projede yapılacak çalışmanın odağında (Cenk ve Özbudak 2011) ile (Barbulescu vd. 2012) yayınları ve interpolasyon teknikleri olmuştur.

Modüler üst alma metotları arasında küp ve tekrarlı küp alma üzerine çalışma yapılmadığı gözlemlenmiştir. Modüler olmayan küp alma üzerinde ise (Zanoni 2010) ve (Bodrato ve Zanoni 2012) harici çalışma bulunamamıştır. Bu çalışmalarda küp hesabında kare alıp çarpma yerine dengesiz Toom-3 metotunun değiştirilmiş bir hali kullanılmaktadır. Makalelerde bahsedildiğine göre, metodun C kütüphanesi GMP ile karşılaştırmalarında $12\% - 25\%$ arası iyileşmenin olduğu görülmüştür. Proje sırasında bu yönteme yardımcı olacak pek çok makale kullanılmıştır. Sayıların büyük üstlerini alırken, üssün, kullanılan yönteme göre ayrılmasını sağlamak gerekmektedir. Örnek olarak üst almak için kare alma kullanılıyorsa, üssün 2 tabanında yazılması işlemleri hızlandırmak için kullanılabilir. Böylece işlemler belirli bir yol izlenerek kare ve çarpma ile yapılabilir. Bu konuda küp almaya da uygun olması açısından izlenebilecek yolları araştırma amacıyla bazı makaleler (Adikari vd. 2011; Bernstein, Chuengsatiansup, vd. 2017; Dimitrov ve Cooklev 1995; Walter 1998) incelenmiştir. İşlemleri modüler hale getirebilmek için farklı yöntemler taranmıştır. Bu konuda da öncelikle şu makaleler göze çarpmaktadır: (Barrett 1986; Bosselaers vd. 1993; Knezevic vd. 2009). Bu yöntemlerin içinde küp almaya uygun olarak Barrett yöntemi (Barrett 1986) seçilmiştir. Barrett yöntemini geliştirme amacıyla da, algoritmanın belirli bir yerinde orta çarpma (middle-product) yapılması gerektiği için (Hanrot vd. 2004) makalesinden yararlanılmıştır.

3 GEREÇ VE YÖNTEM

3.1 Eliptik Eğriler

3.1.1 Toeplitz Matrislerinin Kullanımı

Asal köşegeni ve bu köşegene paralel olan tüm doğrultular boyunca elemanları aynı olan kare matrislere Toeplitz matris denilmektedir. Aşağıda $n \times n$ boyutunda bir matris formu gösterilmiştir.

$$\begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_{n-1} \\ a_n & a_0 & a_1 & \ddots & a_{n-2} \\ a_{n+1} & a_n & a_0 & \ddots & a_{n-3} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ a_{2(n-1)} & \cdots & a_{n+1} & a_n & a_0 \end{bmatrix}$$

Toeplitz matrisleri kullanılarak karakteristiği 2 olan cisimlerde yapılan çarpma işlemine ait karmaşıklık hesapları (Fan ve Hasan 2007a) tarafından verilmiştir. Yapılan bu çalışmada işlemler ikili genişlemeli cisimler üzerine yapılmasına rağmen, proje kapsamında yapılan çalışmalar ile bu işlemler tam sayılar üzerinde de uygulanabilir duruma getirilmiştir. T, 2×2 Toeplitz matrisi ve V, 2×1 boyutlu matris olmak üzere, tam sayılar üzerinde çalışan Toeplitz matris vektör çarpımı aşağıdaki gibi yapılabilir:

$$T \cdot V = \begin{bmatrix} T_0 & T_1 \\ T_2 & T_0 \end{bmatrix} \cdot \begin{bmatrix} V_0 \\ V_1 \end{bmatrix} = \begin{bmatrix} P_1 + P_2 \\ P_1 + P_3 \end{bmatrix}$$

$P_1 = T_0(V_0 + V_1)$, $P_2 = V_1(T_1 - T_0)$, $P_3 = V_0(T_2 - T_0)$ olarak tanımlanmıştır.

Verilen örnekteki matris çarpım işlemi, klasik çarpma yöntemi ile yapıldığında toplam işlem sayısı $4\mathbf{Ç} + 2\mathbf{T_i}$ yaparken Toeplitz formunun etkileri dikkate alındığında işlem sayısı $3\mathbf{Ç} + 2\mathbf{T} + 2\mathbf{T_i}$ tutmaktadır. Burada, **Ç**; çarpma işlemini, **T**; toplama işlemini, $\mathbf{T_i}$; iki yazmaç boyutlu toplama işlemini ifade etmektedir.

Diğer taraftan, 3x3'lük matris ile 3x1'lik vektör çarpımı için ise aşağıdaki çarpma metodu elde

edilmiştir:

$$T \cdot V = \begin{bmatrix} T_0 & T_1 & T_2 \\ T_3 & T_0 & T_1 \\ T_4 & T_3 & T_0 \end{bmatrix} \cdot \begin{bmatrix} V_0 \\ V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} P_3 + P_4 + P_6 \\ P_2 - P_4 + P_5 \\ P_1 - P_2 - P_3 \end{bmatrix}$$

$P_1 = (T_4+T_3+T_0)V_0$, $P_2 = T_3(V_0-V_1)$, $P_3 = T_0(V_0-V_2)$, $P_4 = T_1(V_1-V_2)$, $P_5 = (T_0+T_1+T_3)V_1$, $P_6 = (T_0+T_1+T_2)V_2$ olarak tanımlanmıştır.

Bu durumda, notasyonlar yukardaki ile aynı olmak üzere, toplam işlem sayısı $6\mathbf{Ç} + 8\mathbf{T} + 6\mathbf{T_i}$ olmaktadır. Diğer taraftan klasik çarpma yönteminde ise $9\mathbf{Ç} + 6\mathbf{T_i}$ işlem kullanılmaktadır.

Bu sayede toplama işleminden ödün verilip, çarpma işleminden tasarruf edilmektedir. Bu yaklaşım kullanılarak Curve25519, P-521 ve E-521 eliptik eğrilerinde kullanılan sonlu cisim çarpma işlemleri için yeni algoritmalar geliştirilmiş ve işlem sayısında tasarruflar sağlanmıştır (Ali ve Cenk 2018b; Taşkın ve Cenk 2018). Bu da aynı işlemin daha etkin şekilde gerçekleştirilmesini sağlamaktadır.

3.1.2 Eliptik Eğri Kriptografi için Güvenli Parametre Seçim Yöntemleri

Eliptik eğri tabanlı kriptografi algoritmalarının güvenliğinden bahsedebilmek için seçilecek sonlu cisim, eliptik eğri denklemi gibi farklı parametrelerin hepsi göz önüne alınmalı ve mevcut saldırılara da dayanaklı olacak şekilde bu seçimler yapılmalıdır. Bu kapsamda oluşturulmuş çeşitli standartlar mevcuttur. Bunlara örnek olarak, ANSI X9.62 (1999), IEEE P1363 (2000), SEC 2 (2000), NIST FIPS 186-2 (2000), ANSI X9.63 (2001), Brainpool (2005), NSA Suite B (2005) ve ANSSI FRP256V1 (2011) verilebilir. Ancak bu standartların hiçbirinin tek başına yeterli olduğu düşünülmemektedir. Dolayısıyla parametre seçimleri sırasında tüm bu standartları güncel ataklarla birlikte değerlendirip kapsamlı bir çalışma yapılması gerekmektedir. Bu proje kapsamında SafeCurves (Bernstein ve Lange 2018) çalışmasında tanımlanan ve aşağıdaki başlıklarda belirtilen kriterler temel alınmış ve buna göre Montgomery formunda kriptografik olarak güçlü yeni eliptik eğriler bulmak için bir metodoloji oluşturulmuştur. Aşağıda ilgili tasarım kriterleri özet olarak sunulmuştur. Oluşturulan metodoloji “Bulgular” bölümünde açıklanmıştır.

Yeni eliptik eğri seçimi için aşağıdaki başlıklarda temel kriterler gözetilmelidir.

- **Eliptik Eğri Parametre Seçimi**
  - **Sonlu cisim seçimi:** Karakteristiği 2 olan sonlu cisim genişlemelerinin güvenlik anal-

izleri karakteristiği asal olan sonlu cisimlere oranla daha karmaşık olmasından dolayı seçilecek sonlu cisimin büyük asal (p) karakteristikte olması önerilmektedir.

- **Denklem seçimi:** Eliptik eğri denklemi olarak en yaygın kullanılan kısa Weierstrass, Mongomery veya Edwars formlarının kullanılması beklenmektedir. Bu proje kapsamında öncelikli olarak Montgomery formu üzerinde çalışmalar yapılmıştır.
- **Baz nokta seçimi:** Eliptik eğri üzerinde seçilecek baz noktasının mertebesi ℓ olmak üzere, ℓ değerinin asal olması ve noktanın eliptik eğri üzerinde olduğunun doğrulanması beklenmektedir.
- **Asalların ispatı:** Hesaplamalarda kullanılan bütün asal sayıların ve birleşik sayıların asal çarpanlarının asal olup olmadığının test edilmesi beklenmektedir.

- **Eliptik Eğri Ayrık Logaritma Problemine (EEALP) Karşı Dayanıklılık Kriterleri**
  - **Rho Yöntemine Karşı Dayanıklılık:** Rho yöntemi ile EEALP, eliptik eğri üzerinde yaklaşık $0.886\sqrt{\ell}$ toplama işlemi yapılarak çözülebilmektedir. ℓ değerinin en az 2^{200} olması beklenmektedir.
  - **Transferler:** EEALP'nin doğrusal cebirsel grup üzerinde ayrık logaritma problemine çevrilmesi işlemi olarak tanımlanan transfer işlemi toplamsal ve çarpımsal olarak tanımlanmaktadır. Güvenlik için bunların desteklenmemesi beklenmektedir. $\ell = p$ olduğu durumda toplamsal çarpım desteklenmektedir. Dolayısıyla bu durumun olmaması güvenlik için gereklidir. ℓ değerinin $p^n - 1$ değerini böldüğü en küçük n değeri çarpımsal transfer değeri olarak tanımlanmıştır ve Brainpool (2005) standardına göre bu değerin en az $(\ell - 1)/100$ olması beklenmektedir.
  - **Diskriminantlar:** t eliptik eğrinin iz (Trace) değeri ve s^2, $t^2 - 4p$ değerini bölen en büyük tam kare sayı olmak üzere, $(t^2 - 4p)/s^2$ mod $4 = 1$ olduğu durumda $D = (t^2 - 4p)/s^2$, diğer durumlarda $D = 4(t^2 - 4p)/s^2$ olarak tanımlanan D değeri kompleks çarpım cisim diskriminantıdır. Bu değerin 2^{100}'den büyük olması beklenmektedir.
  - **Sağlamlık:** Eliptik eğri parametreleri belirlenirken izlenen algoritma tümüyle açık olarak yayınlanmalıdır.

- **Eliptik Eğri Kriptografi Güvenliği için Temel Kriterler**
  - **Montgomery Merdiven Yöntemi:** Eğri seçiminde temel kriter basit, hızlı, sabit-zamanlı tek-koordinatlı skaler çarpma işleminin sağlanmasıdır. Bu kapsamda Montgomery merdiveni yöntemi (Montgomery, 1987) en etkin algoritmayı sunmaktadır. Dolayısıyla, seçilecek eliptik eğrinin Montgomery merdiveni yöntemini desteklemesi beklenmektedir.

– **Twist güvenliği:** Bir eliptik eğrinin eşleniği (Twist'i) o eliptik eğrinin tanımlı olduğu sonlu cismin cebirsel kapatmasında tanımlı olan ve o eliptik eğriye izomorf olan eliptik eğridir. EEALP'ye dayanıklılık için geçerli olan tüm kriterler eliptik eğrinin eşleniği üzerinde de kontrol edilmelidir.

– **Bütünsellik:** Eğri seçiminde temel kriter basit, hızlı, sabit-zamanlı tek-koordinatlı skaler çarpma işleminin sağlanmasıdır. Bunun için tek-skaler ve çok-skaler çarpımlar için tüm formüllerin tanımlı olduğu eğri denklemleri seçilmesi beklenmektedir. Montgomery ve Edwards formundaki eliptik eğriler için bu bütünsellik (Bernstein 2006; Bernstein ve Lange 2007) çalışmalarında gösterilmiştir.

– **Ayırt edilememezlik:** Eliptik eğri üzerinde yapılan işlemlerden sonra elde edilen noktanın gösteriminin rastgele bir sayı gösteriminden farklı olmaması beklenmektedir. Ancak çoğu durumda eliptik eğri üzerindeki noktalar kolaylıkla ayırt edilebilmektedir. (Bernstein, Hamburg, vd. 2013) çalışmasında tanımlanan "Elligator 2" tipinde birebir ve örten eşlemeler için $AB(A^2 - 4B)$ değeri sıfırdan farklı olmak üzere $y^2 = x^3 + x$ hariç tüm $y^2 = x^3 + Ax^2 + Bx$ formundaki eliptik eğrilerin nokta yapısının ayırt edilemez olduğu gösterilmiştir.

3.2 Polinom Çarpması

Projede tam sayılar üzerindeki polinom çarpımlarının hızlandırılması için Karatsuba benzeri metotlar ile Toom-Cook tipi interpolasyon kullanan yaklaşımlar hibrit olarak kullanılmıştır. Büyük tamsayılar, üzerinde gerçekleştirme yapılan platformun kelime boyutuna bağlı olarak polinom şeklinde gösterilirler. Bundan dolayı elde edilen algoritmalar asal cisim üzerindeki kriptografik sistemlerin performanslarını artırmada önemli bir rol oynar. Elimizde aşağıdaki gibi verilen n terimli A ve B polinomları olsun.

$$A(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1},$$
$$B(x) = b_0 + b_1x + b_2x^2 + \cdots + b_{n-1}x^{n-1}.$$

Bu polinomları aşağıdaki gibi bölerek A_0, A_1, B_0 ve B_1 polinomlarını elde edelim.

$$\begin{aligned} A_0 &= a_0 + a_1x + \cdots + a_{\frac{n}{2}-1}x^{\frac{n}{2}-1}, \\ A_1 &= a_{\frac{n}{2}} + a_{\frac{n}{2}+1}x + \cdots + a_{n-1}x^{\frac{n}{2}-1}. \\ B_0 &= b_0 + b_1x + \cdots + b_{\frac{n}{2}-1}x^{\frac{n}{2}-1}, \\ B_1 &= b_{\frac{n}{2}} + b_{\frac{n}{2}+1}x + \cdots + b_{n-1}x^{\frac{n}{2}-1}. \end{aligned}$$

Burada $y = x^{n/2}$ olmak üzere okul kitabı algoritması (the schoolbook algorithm) aşağıdaki gibidir.

$$A(x)B(x) \quad = A_0B_0 + y[A_1B_0 + A_0B_1] + y^2A_1B_1.$$

Diğer taraftan Karatuba 2-yol ve rafine Karatsuba 2-yol (the refined Karatsuba 2-way) algoritmaları için ise öncelikle 3 tane çarpma tanımlanır.

$$\begin{aligned} P_1 &= A_0B_0, \\ P_2 &= (A_0 + A_1)(B_0 + B_1), \\ P_3 &= A_1B_1. \end{aligned}$$

Karatsuba 2-yol algoritması $A(x)B(x) = P_1 + y[P_2 - P_1 - P_3] + y^2P_3$ şeklindedir. Rafine Karatsuba 2-yol ise $A(x)B(x) = (y-1)(yP_3 - P_1) + yP_2$ ifadesiyle verilir. Optimize edilmiş Karasuba algoritmasında ise $i = 1, 2, 3$ olmak üzere yukarıda tanımlanmış P_i çarpmaları P_{iL} ve P_{iH} olarak ikiye bölünür. Daha sonra çarpım aşağıdaki gibi hesaplanır.

$$\begin{aligned} A(x)B(x) &= P_{1L} + x^{\frac{n}{2}}[P_{1H} + P_{2L} - P_{1L} - P_{3L}] + x^n[P_{2H} - P_{1H} - P_{3H} + P_{3L}] + x^{\frac{3n}{2}}P_{3H} \\ &= P_{1L} + x^{\frac{n}{2}}[(P_{1H} - P_{3L}) + P_{2L} - P_{1L}] + x^n[-(P_{1H} - P_{3L}) + P_{2H} - P_{3H}] + x^{\frac{3n}{2}}P_{3H} \end{aligned}$$

Çarpılacak polinomlar başlangıçta iki yerine üç parçaya da ayrılabilirler. Bu durumda her bir polinom için aşağıdaki gibi boyutu orijinal boyutun üçte biri olan yeni polinomar elde edilir:

$$\begin{aligned} A_0 &= a_0 + a_1x + \cdots + a_{\frac{n}{3}-1}x^{\frac{n}{3}-1}, \\ A_1 &= a_{\frac{n}{3}} + a_{\frac{n}{3}+1}x + \cdots + a_{\frac{2n}{3}-1}x^{\frac{n}{3}-1}, \\ A_2 &= a_{\frac{2n}{3}} + a_{\frac{2n}{3}+1}x + \cdots + a_{n-1}x^{\frac{n}{3}-1}, \\ B_0 &= b_0 + b_1x + \cdots + b_{\frac{n}{3}-1}x^{\frac{n}{3}-1}, \\ B_1 &= b_{\frac{n}{3}} + b_{\frac{n}{3}+1}x + \cdots + b_{\frac{2n}{3}-1}x^{\frac{n}{3}-1}, \\ B_2 &= b_{\frac{2n}{3}} + b_{\frac{2n}{3}+1}x + \cdots + b_{n-1}x^{\frac{n}{3}-1}. \end{aligned}$$

Kısaltma olarak $y = x^{\frac{n}{3}}$ olmak üzere, okul kitabı algoritması bu durumda aşağıdaki gibidir:

$$A(x)B(x) \quad = \quad A_0B_0 + y[A_1B_0 + A_0B_1] + y^2(A_0B_2 + A_1B_1 + A_2B_0) + y^3(A_1B_2 + A_2B_1) + y^4A_2B_2.$$

Karatsuba benzeri 3-yol algoritması da aşağıdaki gibidir:

$$\begin{aligned} A(x)B(x) = \; & A_0B_0 + y[(A_0 + A_1)(B_0 + B_1) - A_0B_0 - A_1B_1] + y^2[(A_0 + A_2)(B_0 + B_2) \\ & -A_0B_0 - A_2B_2 + A_1B_1] + y^3[(A_1 + A_2)(B_1 + B_2) - A_1B_1 - A_2B_2] + y^4A_2B_2. \end{aligned}$$

Optimize edilmiş Karatsuba 3-yol algoritması için $P_1 = A_0B_0$, $P_2 = (A_0+A_1)(B_0+B_1)$, $P_3 = A_1B_1$, $P_4 = (A_0 + A_2)(B_0 + B_2)$, $P_5 = A_2B_2$ ve $P_6 = (A_1 + A_2)(B_1 + B_2)$ olarak tanımlanır. Daha sonra $i = 1, 2, 3, 4, 5, 6$ olmak üzere yukarıda tanımlanmış P_i çarpmaları P_{iL} ve P_{iH} olarak ikiye bölünür. Optimize edilmiş çarpma algoritması aşağıdaki gibi hesaplanır.

$$\begin{aligned} A(x)B(x) &= P_{1L} + y[P_{1H} + P_{2L} - P_{1L} - P_{3L}] + y^2[P_{2H} - P_{1H} - P_{3H} + P_{4L} - P_{1L} - P_{5L} + P_{3L}] \\ &\quad + y^3[P_{4H} - P_{1H} - P_{5H} + P_{3H} + P_{6L} - P_{3L} - P_{5L}] \\ &\quad + y^4[P_{6H} - P_{3H} - P_{5H} + P_{5L}] + y^5P_{5H} \\ &= P_{1L} + y(P_{1H} - P_{3L}) + P_{2L} - P_{1L}] + y^2[-(P_{1H} - P_{3L}) - P_{3H} + P_{2H} \\ &\quad + P_{4L} - P_{1L} - P_{5L}] + y^3[(P_{3H} - P_{5L}) + P_{4H} - P_{1H} - P_{5H} + P_{6L} \\ &\quad - P_{3L}] + y^4[-(P_{3H} - P_{5L}) + P_{6H} - P_{5H}] + y^5P_{5H}. \end{aligned}$$

Algoritmaların karmaşıklıkları aşağıdaki tabloda verilmiştir. Burada SB, okul kitabı algoritmasını; KA2, Karatsuba 2-yol algoritmasını; RK2, rafine Karatsuba 2-yol algoritmasını; OP2, optimize Karatsuba 2-yol algortimasını göstermektedir. Son satırlardaki gösterimler ise aynı algoritmaların 3-yol olanlarıdır.

Algorithm	$M_{\oplus S}(n)$	$M_{\oplus D}(n)$	$M_{\otimes}(n)$	$M(n)$
SB		$n^2 - 2n + 1$	n^2	$2n^2 - 2n + 1$
KA2	$2n^{1.58} - 2n$	$4n^{1.58} - 6n + 2$	$n^{1.58}$	$7n^{1.58} - 8n + 2$
RK2	$2n^{1.58} - 2n$	$3.5n^{1.58} - 5n + 1.5$	$n^{1.58}$	$6.5n^{1.58} - 7n + 1.5$
OPK2	$2n^{1.58} - 2n$	$3.5n^{1.58} - 5n + 1.5$	$n^{1.58}$	$6.5n^{1.58} - 7n + 1.5$
SB		$n^2 - 2n + 1$	n^2	$2n^2 - 2n + 1$
KA3	$2n^{1.63} - 2n$	$3.8n^{1.63} - 6n + 2.2$	$n^{1.63}$	$6.8n^{1.63} - 8n + 2.2$
OPK3	$2n^{1.63} - 2n$	$\frac{53}{15}n^{1.63} - \frac{16}{3}n + \frac{9}{5}$	$n^{1.63}$	$\frac{98}{15}n^{1.63} - \frac{22}{3}n + \frac{9}{5}$

Tablo 1: Algoritmaların karmaşıklıklarının karşılaştırılması

Özellikle büyük tam sayı çarpımları yapılırken bu algoritmaları direkt olarak kullanmak yerine bunların hibrit bir şekilde kullanılması, karakteristiği iki olan cisimler üzerindeki polinomlarda olduğu gibi iyileştirmeler ve daha iyi sonuçlar getirmiştir. Bu sonuçlar bir sonraki bölümde sunulmuştur.

Asal cisimler üzerindeki polinom çarpımlarının başka bir uygulamasıda süper singüler izojen tabanlı kuantum sonrası kriptografidir. Burada boyutları 1000 biti bulabilen cisimler kullanılmaktadır. Ayrıca buradaki eliptik eğriler bu asal cisimlerin ikinci derece genişlemesi olan cisimler üzerinedir. Yukarıdaki çalışmanın benzeri bu cisimler için uygulanmış ve literatürde kullanılan metotlardan daha iyi sonuçlar getirdiği gösterilmiştir. Burada elde edilen bulgular bir sonraki bölümde

verilmiştir.

Proje kapsamında polinom çarpımları üzerine yapılan diğer bir çalışma, n-terimli polinomların çarpımının ilk ℓ-teriminin bulunması için (Barbulescu vd. 2012)'de sunulan arama tekniğinin kullanılmasıdır. Bu arama tekniği 'bilineer rütbe problemi'ne (bilinear rank problem) dayanmaktadır. Bilineer rütbe problemi, bir bilineer eşlemi (map) hesaplamak için gereken tüm çarpımlar arasından minimum k tane çarpımı bulmaktır. Diğer bir deyişle, rütbesi 1 olan $n \times n$-bilineer yapıları tutan kümeye 'üretici küme' $\mathcal{G}$ dersek, bilineer rütbe problemi $\mathcal{G}$'nin minimum k tane elemanının lineer kombinasyonlarını kullanarak, ℓ tane bilineer yapıyı tutan hedef küme $\mathcal{T} = \{t_0, t_1, ..., t_{\ell-1}\} \subseteq \text{Span}(\mathcal{G})$'yi elde etmektir ya da bu optimal k sayısı için çözümler bulmaktır.

Çarpılacak n-terimli polinomlar K cismi üzerinde olsun ve bu polinomların katsayıları, sırasıyla, $(a_0, ..., a_{n-1})$ ve $(b_0, ..., b_{n-1})$ olsun. Bu durumda, herhangi bir $n \times n$-bilineer yapı $\sum_{i,j=0}^{n} c_{i,j} a_i b_j$ şeklinde ifade edilir. Her yapının kendine özgü katsayıları olduğu için, bu yapılar $(c_{0,0}, c_{0,1}, ..., c_{n-1,n-1})$ şeklinde de ifade edilebilir. Bu yüzden, her bir $n \times n$-bilineer yapı $\{a_0 b_0, ..., a_{n-1} b_{n-1}\}$ bazına sahip n^2-boyutlu K üzerindeki vektör uzayı V'nin elemanıdır. Dolayısıyla, vektör uzayı V, üretici küme $\mathcal{G}$'yi kapsar. Üretici küme $\mathcal{G}$'nin k elemanlı alt kümesine $\mathcal{W}$, bu alt kümeden (span) elde edilen alt uzaya W diyelim. Herhangi bir k için, üretici küme $\mathcal{G}$'den hedef küme $\mathcal{T}$'yi elde etmek için, alt uzay W, hedef küme $\mathcal{T}$'yi içermelidir. Bunun için en temel yaklaşım, $\mathcal{G}$'nin k elemanlı bütün alt kümelerini sıralayıp, her birisini hedef kümenin (spani) $T = \text{Span}(\mathcal{T})$'yi kapsayıp kapsamadığını test etmektir. Fakat, bu yaklaşımın dezavantajı alt kümelerin birbirleriyle doğrusal bağımlı olabilmesidir. Bu da aynı alt uzayların (W'lerin) tekrar tekrar üretilmesine sebep olur. Bu fazlalıktan kurtulmak için (ibid.)'de $\mathcal{G}$'nin alt kümeleri $\mathcal{W}$'ler yerine bunlardan üretilen alt uzay W'lere bakılmaktadır ve bunlardan aşağıdaki üç kriteri sağlayanlar aranmaktadır:

(i) $T \subset W$: Alt uzay W, hedef uzay T'yi kapsar.

(ii) $\text{Span}(W \cap \mathcal{G}) = W$: Alt uzay W, üretici küme $\mathcal{G}$'nin elemanlarından (span) edilmiştir.

(iii) $\dim(W) = k$: Alt uzay yalnızca k tane (üreteçten) elde edilmiştir.

Herhangi bir alt uzay W'nin birinci kriteri sağlaması için T'den genişletilmiş olması gerekmektedir. Fakat, T'den genişletilen alt uzayların sayısı oldukça fazladır. Bu nedenle, Osedelets'in sezgisel (heuristic) algoritmalarında kullandığı teknik düşünülmüştür. Buna göre, T'ye üreteç eklenerek genişletilen alt uzaylara bakılmaktadır ve ek bir kriter oluşturulmuştur:

$$(ii') \; \exists \mathcal{W}' \subset \mathcal{G} \text{ such that } W = T \oplus \text{Span}(\mathcal{W}').$$

Arama algoritmasında öncelikle (i), $(ii^{'})$, (iii) kriterlere bakılmıştır. Daha sonra bulunan alt uzaylarda kriter (ii) aranmıştır. Üretici küme $\mathcal{G}$, hedef uzay T ve k ($\dim(T) \leq k \leq$ rank$(\mathcal{G})$) verildiğinde alt uzayları arama algoritmasındaki basamaklar şöyledir:

1. T'nin boyutunu ve $W \cap \mathcal{G}$'nin rütbesini bul.
2. Eğer $\dim(T) = k$ ve rank$(\mathcal{G}) = k$ ise, T aranılan alt uzaydır.
3. Eğer $\dim(T) < k$ ise, T'yi $\mathcal{G}/W$'nin elemanlarının her biriyle genişlet.
4. Her bir genişletilmiş alt uzay için, 1. basamağa dön.

Bu projede n-terimli polinomların çarpımının ilk ℓ-teriminin bulunması için yukarıdaki teknik kullanılmıştır. Amaç, bir k sayısı için hedef kümeyi üretici kümedeki elemanlarla genişleterek çözüm alt uzayları bulmaktır. Bu çözüm alt uzaylarından, k tane çarpımdan oluşan istenilen ℓ-terimli polinom çarpımını bulmak kolaydır. Bu sebeple arama algoritmasıyla bu çözüm alt uzayları aranmaktadır. Çarpılan polinomlar $A(x) = a_0 + a_1x + ... + a_{n-1}x^{n-1}$ ve $B(x) = b_0 + b_1x + ... + b_{n-1}x^{n-1}$ olsun. Bu durumda, çarpım

$$A(x) \cdot B(x) = a_0b_0 + (a_0b_1 + a_1b_0)x + ... + a_{n-1}b_{n-1}x^{2n-2}$$

ve hedef küme

$$\mathcal{T} = \{t_0 = a_0b_0, t_1 = a_0b_1 + a_1b_0, ..., t_{2n-1} = a_{n-1}b_{n-1}\}$$

şeklindedir. Bu çarpımı elde etmek için kullanılan üretici küme $\mathcal{G}$, $\alpha_i, \beta_j \in K$ ve $i, j = 0, 1, ..., n-1$ için,

$$(\alpha_0a_0 + \alpha_1a_1 + ... + \alpha_{n-1}a_{n-1}) \cdot (\beta_0b_0 + \beta_1b_1 + ... + \beta_{n-1}b_{n-1})$$

şeklindeki, 0 dışında, bütün olası terimleri kapsar. Polinom çarpımının ilk ℓ terimi için hedef küme $\overline{\mathcal{T}} = \{t_0 = a_0b_0, t_1 = a_0b_1 + a_1b_0, ..., t_{\ell-1}\}$'dir. Hedef küme $\overline{\mathcal{T}}$'yi genişletmek için üretici küme $\mathcal{G}$ ile ilgili üç strateji kullanılabilir:

1. Hedef küme $\overline{\mathcal{T}}$'deki elemanlarla üretici küme $\mathcal{G}$'deki birçok üreteç lineer bağımlıdır. Bu yüzden, hedef küme $\overline{\mathcal{T}}$'yi genişletmek için $\overline{\mathcal{T}}$ ile lineer bağımlı olmayan üreteçleri almak gerekir. Yani, üretici küme $\mathcal{G}$ daraltılarak, yalnızca polinom çarpımındaki geriye kalan $2n - 1 - \ell$ terimdeki her bir bileşen a_ib_j'nin bulunduğu üreteçler alınmalıdır.
2. Hedef küme $\overline{\mathcal{T}}$ genişletilmek istense de amaç, buradaki elemanların üretilmesidir. Bu sebeple, üretici küme $\mathcal{G}$'deki elemanlardan, $\overline{\mathcal{T}}$'deki elemanlarda bulunan her bir bileşen a_ib_j'nin

bulunduğu üreteçleri almak yeterlidir. Yani, üretici küme $\mathcal{G}$ polinom çarpımındaki geriye kalan $2n-1-\ell$ terimdeki her bir bileşen $a_i b_j$'nin bulunduğu üreteçler elenerek daraltılmalıdır.

3. Üretici küme $\mathcal{G}$, olduğu gibi alınmalıdır.

Örnek olarak $\mathbb{F}_2$ üzerinde 3-terimli polinom çarpımının ilk 3 terimin elde edilmesi verilebilir. $A(x) = a_0 + a_1 x + a_2 x^2$ ve $B(x) = b_0 + b_1 x + b_2 x^2$ olsun. Bu iki polinomun çarpımı

$$A(x) \cdot B(x) = a_0 b_0 + (a_0 b_1 + a_1 b_0)x + (a_0 b_2 + a_1 b_1 + a_2 b_0)x^2 + (a_1 b_2 + a_2 b_1)x^3 + a_2 b_2 x^4$$

olsun. Bu durumda, hedef küme

$$\overline{\mathcal{T}} = \{t_0 = a_0 b_0,\ t_1 = a_0 b_1 + a_1 b_0,\ t_2 = a_0 b_2 + a_1 b_1 + a_2 b_0\}$$

dir. Üretici küme ise,

$$\mathcal{G} = \{a_0 b_0, a_0 b_1, a_0(b_0 + b_1), a_0 b_2, ..., (a_0 + a_1 + a_2)(b_0 + b_1 + b_2)\}$$

dir. Eğer,

- 1. strateji kullanılırsa, içinde $a_1 b_2$, $a_2 b_1$ ve $a_2 b_2$ bileşenlerinin bulunduğu üreteçler alınmalıdır. Geri kalanlar elendiğinde geriye 32 tane üreteç kalır.
- 2. strateji kullanılırsa, içinde $a_1 b_2$, $a_2 b_1$ ve $a_2 b_2$ bileşenlerinin bulunduğu üreteçler elenmelidir. Bu durumda, geriye 17 tane üreteç kalır.
- 3. strateji kullanılırsa, üretici kümeyi olduğu gibi alınmalıdır. Bu durumda, üretici kümedeki üreteç sayısı $(2^3 - 1) \cdot (2^3 - 1) = 49$'dur.

Sonuç olarak, 1. ve 2. stratejideki kümeler birbirinin tamamlayıcılarıdır ve en az elemanlı üreteç 2. stratejiye aittir. Bu stratejiler, yazılan C# koduyla denenmiştir:

- 1. strateji, Cenk ve Özbudak (2011)'deki k için minimum değer olan 5'i vermemektedir ve verimsizdir. Fakat, spektrum olarak $k = 6$'dan $k = 9$'a kadar sonuç vermektedir.
- 2. strateji, Cenk ve Özbudak (2011)'deki k için minimum değer olan 5'i vermektedir ve 1. stratejiye göre daha verimlidir. Fakat, spektrum olarak yalnızca $k = 5$ ve $k = 6$ için sonuç vermektedir.

- 3. strateji de Cenk ve Özbudak (2011)'deki k için minimum değer olan 5'i vermektedir ve spektrum olarak $k = 5$'ten $k = 9$'a kadar olan tüm sonuçları vermektedir fakat, verimsizdir.

Bütün bunlar göz önüne alındığında, amaç en verimli yöntemle, minimum k için sonuç bulmak olduğundan, 2. strateji en iyi stratejidir.

Bütün çözüm alt uzayları W'ler bulunduktan sonra, n-terimli polinomların çarpımının ilk ℓ-terimini veren formülü elde etmek için, aşağıdaki basamaklar takip edilir:

1. Öncelikle, her bir çözüm alt uzayı W için $W \cap \mathcal{G}$ bulunur.
2. $W \cap \mathcal{G}$ kümesinde bulunan elemanlardan rütbesi k olan, k elemanlı bütün alt kümeler bulunur.
3. Her bir alt küme, hedef küme $\overline{\mathcal{T}}$'deki elemanları elde etmek için birer bazdır.
4. Bu baza göre hedef küme $\overline{\mathcal{T}}$'deki elemanların koordinatları bulunur ve formül elde edilir.

Örnek olarak $\mathbb{F}_2$ üzerinde 3-terimli polinom çarpımının ilk 3 terimin elde edilmesinde formülün bulunması verilebilir. Burada k=5 için, bulunan çözüm alt uzaylarından bir tanesini ele alalım. Çözüm alt uzayı $W = \text{Span}(\overline{\mathcal{W}})$ iken, $\overline{\mathcal{W}}$, hedef küme $\overline{\mathcal{T}}$'nin, üretici küme $\mathcal{G}$'den iki eleman eklenerek genişletilmiş kümesi olsun. Bu durumda, ele aldığımız küme:

$$\overline{\mathcal{W}} = \{a_0b_0,\ a_0b_1 + a_1b_0,\ a_0b_2 + a_1b_1 + a_2b_0,\ a_2b_0,\ a_1b_1\}$$

'dir. O halde, $W \cap \mathcal{G} = \text{Span}(\overline{\mathcal{W}}) \cap \mathcal{G} = \{a_2b_0, a_1b_1, a_0b_2, a_0b_0, a_0b_0 + a_0b_2,\ a_0b_0 + a_2b_0, a_0b_0 + a_0b_1 + a_1b_0 + a_1b_1\}$ olur. Bu kümenin rütbesi 5 olan bütün 5 elemanlı alt kümeleri birer formül oluşturur. Örneğin, bu alt kümelerden bir tanesi

$$\begin{aligned}\mathcal{F}_1 \quad = \quad & \{f_0 = a_2b_0,\ f_1 = a_1b_1,\ f_2 = a_0b_2,\ f_3 = a_0b_0, \\ & f_4 = a_0b_0 + a_0b_1 + a_1b_0 + a_1b_1\}\end{aligned}$$

'dir. Bu alt kümenin rütbesi 5'tir. Bu alt kümeyi baz olarak aldığımızda hedef küme $\overline{\mathcal{T}}$'deki elemanların koordinatları

$$t_0 = f_3,\ \ t_1 = f_4 - f_1 - f_3,\ \ t_2 = f_0 + f_1 + f_2$$

olur. Bu durumda, 3-terimli polinomların çarpımının ilk 3 terimi,

$$\begin{aligned}\overline{A(x)\cdot B(x)} &= f_3 + (f_4 - f_1 - f_3)x + (f_0 + f_1 + f_2)x^2 \\ &= a_0b_0 + ((a_0+a_1)(b_0+b_1) - a_1b_1 - a_0b_0)x \\ &\quad +(a_2b_0 + a_1b_1 + a_0b_2)x^2\end{aligned}$$

5 çarpımla bulunabilir.

3.3 Modüler Üs Alma

Modüler üst alma konusunda, öncelikle modüler küp alma üzerine yoğunlaşılmıştır. Karatsuba, dengeli ve dengesiz Toom-Cook çarpma yöntemleri incelenmiştir. Sonrasında, dengesiz Toom-Cook çarpmasını kullanan "sugar-cube" küp algoritması incelenmiştir. $u = a_1x + a_0$, a_i'ler n-kelime büyüklüğünde olacak şekilde ve $A = a_1^2$, $B = a_0^2$ ve $A = A_1x + A_0$, $B = B_1x + B_0$ iken, $u^3 = f(x)$ denirse,

$$\begin{aligned}g(x) = (A_1a_1)x^4 + (A_0a_1 + 3A_1a_0)x^3 + (3A_0a_0 + 3B_1a_1)x^2 + (27B_0a_1 + 9B_1a_0)x \\ + 81B_0a_0 = \sum_{i=0}^{4} c_ix^i\end{aligned}$$

c_0 81'e ve c_1 9'a bölünerek g, f'e dönüştürülebilir. Bu işlem belirli ve küçük sayılarla yapıldığı için eklediği karmaşıklığın göz önüne alınması gerekmez. Dengesiz Toom-3 algoritması kullanılırsa, algoritmanın işlem karmaşıklığı $C_{sugar}(2n) = 2S(n) + 5M(n)$ olarak bulunmaktadır.

Bu algoritma belirli boyutlarda, klasik $a^3 = a^2 \cdot a$ algoritmasından daha hızlı olduğu için modüler halde de hızlı olacağı düşünülmüştür. Klasik metot, modüler yapılmadığı zaman önce bir kare alma, sonrasında da alınan kareyle orjinal sayının çarpılmasıyla bulunmaktadır. Kare almanın Karatsuba, ikinci kısmın da dengesiz Toom-3 ile yapıldığını varsayarsak, karmaşıklığın $C_{klasik}(2n) = 3S(n) + 5M(n)$ olduğunu görüyoruz. Yani, $C_{sugar} < C_{klasik}$ olmaktadır. Bu sebeple "sugar-cube" algoritması modüler hale getirilmeye çalışılmıştır. Ancak, 2. raporda gösterildiği üzere modüler "sugar-cube" yöntemi, klasik modüler küp almaya göre yavaş kaldığı gözlemlenmiştir. Bunun sebebi modüler durumda klasik metotta iki kere indirgeme yaparak, işlem büyüklüklerinin azaltılabilmesidir. a^2 hesaplandıktan sonra, mod alınıp tekrar $2n$-kelime büyüklüğünde bir sayıya düşürülürse ortaya çıkacak karmaşıklık $C_{klasik}(2n) = 3S(n) + 3M(n)$ olacaktır. Fakat gözlemlendiği ve 2. raporda anlatıldığı şekilde, yeni modüler küp algoritmasının sonunda mod al-

mak, o yöntem için olabilecek en hızlı indirgemedir. Bu sebepten de karmaşıklığında bir değişiklik olmamaktadır.

Bu da $6n$-kelime büyüklüğündeki bir sayının $2n$-kelime büyüklüğündeki bir başka sayıyla modunun alınması demektir. Klasik yöntemde ise iki kere $4n$-kelime büyüklüğündeki bir sayının $2n$-kelime büyüklüğündeki bir sayıyla modunun alması kullanılan yöntemlerden biridir. İki algoritmanın arasındaki işlem farkları hesaplandığında, yeni modüler küp algoritması, eğer aşağıdaki formüle uyacak bir şekilde değiştirilebilirse klasik metottan daha hızlı olacağı görülmüştür.

$$2R(4n, 2n) - 1, 2M(n) > R(6n, 2n) \tag{1}$$

Bu notasyonda $R(a, b)$, a-bitlik bir sayının b-bitlik bir sayıyla modunun alınmasının, $M(n)$ ise n-bitlik iki sayının çarpılmasının asimptotik olarak işlem sayısını göstermektedir.

Amaç küp almak olduğu için, Barrett indirgemesi (Barrett reduction), Montgomery indirgemesi ve benzeri algoritmalardan daha uygun bulunmuştur. Barrett indirgemesi daha genel bir algoritma olmasına rağmen normal hali sadece $R(4n, 2n)$ oranında çalışmaktadır. Genel Barrett indirgemesi her türlü boyutta çalışmaktadır fakat üzerine araştırma yapılmadığı görülmüştür. Bu sebeple genel Barrett indirgemesini, $R(6n, 2n)$ için özelleştirip, küp algoritmasını hızlandırmaya odaklanılmıştır.

Araştırmadaki karşılaştırmalar için C yazılım dili kullanılmıştır. Küp ve indirgeme karşılaştırmaları yapılırken, kodlarda C dilinde yazılmış GMP Büyük Sayı kütüphanesi kullanılmış, ve GMP kütphanesinin içerdiği kodlara olabildiğince benzer şekilde yazılmasına dikkat edilmiştir.

4 BULGULAR

4.1 Eliptik Eğriler

Proje kapsamında yapılan çalışmalarda özellikle $\mathbb{F}_{2^{255}-19}$ cismi üzerinde tanımlı Curve25519 (Bernstein 2006) eliptik eğrisi ile $\mathbb{F}_{2^{521}-1}$ cismi üzerinde tanımlı P-521 (NIST 2013) ve E-521 (Aranha vd. 2013) eğrilerinin aritmetik işlem karmaşıklıklarının düşürülmesi üzerinde durulmuştur. Bu bağlamda, literatürdeki Edwards ve Montgomery formundaki eliptik eğriler ile ilgili mevcut çalışmalar ile farklı platformlar için geliştirilen algoritmalar ve gerçeklenmeleri incelenmiştir. $\mathbb{F}_{2^{255}-19}$ ve $\mathbb{F}_{2^{521}-1}$ sonlu cisimleri üzerinde tanımlı bu eğriler üzerine inşa edilen kriptografik algoritmaların daha etkin çalışması için geliştirmeler üzerine çalışmalar yapılmıştır. Araştırma süresince Toeplitz matris tabanlı vektör çarpım (TMVÇ) işlemi temel alınarak Curve25519, P-521 ve E-521 tabanlı algoritmaların iyileştirilmesi üzerine çalışmalar yapılmıştır. Bu amaçla literatürde ikili genişleme cisimleri üzerinde verimli sonuçlar getiren Toeplitz matris vektör çarpım tekniklerinin (Fan ve Hasan 2007a) tam sayılar üzerine uygulaması yapılmıştır. Burada matris vektör çarpımları için rekürsif yöntem kullanılmış ve her rekürsif adımda olabilecek en iyi algoritmanın ne olacağı incelenmiştir. Bu teknik $\mathbb{F}_{2^{255}-19}$ ve $\mathbb{F}_{2^{521}-1}$ cisimlerinde çarpma yapmak için kullanılmış ve elde edilen algoritmaların aritmetik karmaşıklıklarının literatürde var olan algoritmalardan daha iyi olduğu gösterilmiştir. Bu sayede toplama işleminden ödün verilip, çarpma işleminden tasarruf edilmektedir. Bu yaklaşım kullanılarak Curve25519, P-521 ve E-521 eliptik eğrilerinde kullanılan sonlu cisim çarpma işlemleri için yeni algoritmalar geliştirilmiş ve işlem sayısında tasarruflar sağlanmıştır. Bu da aynı işlemin daha etkin şekilde gerçekleştirilmesini sağlamıştır. Yapılan çalışmalar, makale olarak yazılmış ve çeşitli konferans ve dergilerde yayınlanmıştır (Ali ve Cenk 2018b; Taşkın ve Cenk 2018).

4.1.1 Toeplitz Matrisi İçin Uygun Form Bulunması

Asal karakteristiğe sahip sonlu cisim elemanları genellikle büyük tamsayı aritmetiği kullanılarak ifade edilirler. Bu elemanlar **limb** (Chou 2015) adı verilen küçük sayılara bölünerek, işlemler bu limb dizileri üzerinde yapılır. $b-$bit uzunluğunda bir asal cisim üzerinde tanımlı f elemanı $(f_0, f_1, \ldots, f_{\lceil b/r \rceil - 1})$ şeklinde limb dizisi ile ifade edilerek

$$f = \sum_{i=0}^{\lceil b/r \rceil - 1} f_i 2^{\lceil ir \rceil}$$

eşitliği olarak gösterilmesine radix-2^r gösterimi denilmektedir. Böylelikle, cisim aritmetiği bu limb değerleri üzerinde yapılabilmektedir.

Curve25519 için, f, g ve h değerleri $\mathbb{F}_{2^{255}-19}$ üzerinde tanımlı elemanlar olmak üzere $f \cdot g = h$ işlemini yapmak için elemanları aşağıdaki polinomlar gibi tanımlayarak radix-$2^{25.5}$ gösterimini kullanabiliriz.

$$\begin{aligned}
f(x) =& f_0 + 2^{26} f_1 x + 2^{51} f_2 x^2 + 2^{77} f_3 x^3 + 2^{102} f_4 x^4 + 2^{128} f_5 x^5 + \\
& 2^{153} f_6 x^6 + 2^{179} f_7 x^7 + 2^{204} f_8 x^8 + 2^{230} f_9 x^9 \\
g(x) =& g_0 + 2^{26} g_1 x + 2^{51} g_2 x^2 + 2^{77} g_3 x^3 + 2^{102} g_4 x^4 + 2^{128} g_5 x^5 + \\
& 2^{153} g_6 x^6 + 2^{179} g_7 x^7 + 2^{204} g_8 x^8 + 2^{230} g_9 x^9 \\
h(x) =& h_0 + 2^{26} h_1 x + 2^{51} h_2 x^2 + 2^{77} h_3 x^3 + 2^{102} h_4 x^4 + 2^{128} h_5 x^5 + \\
& 2^{153} h_6 x^6 + 2^{179} h_7 x^7 + 2^{204} h_8 x^8 + 2^{230} h_9 x^9
\end{aligned}$$

Bu durumda, her polinom, 1 değeri ile hesaplanması ile elde edilen sonucu ifade etmektedir. Bu çarpma işleminin matris gösterimi aşağıda verilmiştir:

$$\begin{bmatrix}
g_0 & 38g_9 & 19g_8 & 39g_7 & 19g_6 & 38g_5 & 19g_4 & 38g_3 & 19g_2 & 38g_1 \\
g_1 & g_0 & 19g_9 & 19g_8 & 19g_7 & 19g_6 & 19g_5 & 19g_4 & 19g_3 & 19g_2 \\
g_2 & 2g_1 & g_0 & 38g_9 & 19g_8 & 38g_7 & 19g_6 & 38g_5 & 19g_4 & 38g_3 \\
g_3 & g_2 & g_1 & g_0 & 19g_9 & 19g_8 & 19g_7 & 19g_6 & 19g_5 & 19g_4 \\
g_4 & 2g_3 & g_2 & 2g_1 & g_0 & 38g_9 & 19g_8 & 38g_7 & 19g_6 & 38g_5 \\
g_5 & g_4 & g_3 & g_2 & g_1 & g_0 & 19g_9 & 19g_8 & 19g_7 & 19g_6 \\
g_6 & 2g_5 & g_4 & 2g_3 & g_2 & 2g_1 & g_0 & 38g_9 & 19g_8 & 38g_7 \\
g_7 & g_6 & g_5 & g_4 & g_3 & g_2 & g_1 & g_0 & 19g_9 & 19g_8 \\
g_8 & 2g_7 & g_6 & 2g_5 & g_4 & 2g_3 & g_2 & 2g_1 & g_0 & 38g_9 \\
g_9 & g_8 & g_7 & g_6 & g_5 & g_4 & g_3 & g_2 & g_1 & g_0
\end{bmatrix} \cdot \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{bmatrix} = \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix}$$

Burada toplam işlem maliyeti $109\mathbf{Ç} + 90\mathbf{T_i}$ olmaktadır.

Curve25519 eliptik eğrisi üzerindeki sonlu cisimde çarpma işlemi göz önüne alındığında, mevcut algoritmanın yapısının Toeplitz matris vektör çarpımına uygun olmadığı görülmektedir. Bu bağlamda, `MAGMA` (Sydney 2017) cebirsel hesaplama yazılımı kullanılarak sonlu cisim elemanlarının ifade edildiği bütün polinom formlarını analiz eden kod geliştirilmiştir. Bu kod ile radix-$2^{25.5}$ (Bernstein 2006; Chou 2015) formuna alternatif olabilecek bütün formlar incelenmiştir. Bu bağlamda, 9 terimli polinomlar için toplam 130, 10 terimli polinomlar için toplam 638 farklı form Toeplitz matrisine uygunluğu hususunda analiz edilmiştir. Yapılan çalışma sonucu sadece 1 matris

formunun belirtilen yöntem için uygun olduğu tespit edilmiştir. Bulunan form, aşağıda gösterildiği şekilde Toeplitz matris çarpımı uygulanmasına uygun hale getirilmiştir.

Bu durumda, yukarıdakine benzer şekilde, f, g ve h elemanları aşağıdaki gibi tanımlanmıştır.

$$\begin{aligned} f(x) =& f_0 + 2^{26} f_1 x + 2^{52} f_2 x^2 + 2^{78} f_3 x^3 + 2^{104} f_4 x^4 + 2^{130} f_5 x^5 + \\ & 2^{156} f_6 x^6 + 2^{182} f_7 x^7 + 2^{208} f_8 x^8 + 2^{234} f_9 x^9 \\ g(x) =& g_0 + 2^{26} g_1 x + 2^{52} g_2 x^2 + 2^{78} g_3 x^3 + 2^{104} g_4 x^4 + 2^{130} g_5 x^5 + \\ & 2^{156} g_6 x^6 + 2^{182} g_7 x^7 + 2^{208} g_8 x^8 + 2^{234} g_9 x^9 \\ h(x) =& h_0 + 2^{26} h_1 x + 2^{52} h_2 x^2 + 2^{78} h_3 x^3 + 2^{104} h_4 x^4 + 2^{130} h_5 x^5 + \\ & 2^{156} h_6 x^6 + 2^{182} h_7 x^7 + 2^{208} h_8 x^8 + 2^{234} h_9 x^9 \end{aligned}$$

Benzer şekilde, elde ettiğimiz yeni polinom gösterimleri kullanıldığında cisim çarpması için aşağıdaki matris gösterimi elde edilmiştir:

$$\begin{bmatrix} g_0 & 608g_9 & 608g_8 & 608g_7 & 608g_6 & 608g_5 & 608g_4 & 608g_3 & 608g_2 & 608g_1 \\ g_1 & g_0 & 608g_9 & 608g_8 & 608g_7 & 608g_6 & 608g_5 & 608g_4 & 608g_3 & 608g_2 \\ g_2 & g_1 & g_0 & 608g_9 & 608g_8 & 608g_7 & 608g_6 & 608g_5 & 608g_4 & 608g_3 \\ g_3 & g_2 & g_1 & g_0 & 608g_9 & 608g_8 & 608g_7 & 608g_6 & 608g_5 & 608g_4 \\ g_4 & g_3 & g_2 & g_1 & g_0 & 608g_9 & 608g_8 & 608g_7 & 608g_6 & 608g_5 \\ g_5 & g_4 & g_3 & g_2 & g_1 & g_0 & 608g_9 & 608g_8 & 608g_7 & 608g_6 \\ g_6 & g_5 & g_4 & g_3 & g_2 & g_1 & g_0 & 608g_9 & 608g_8 & 608g_7 \\ g_7 & g_6 & g_5 & g_4 & g_3 & g_2 & g_1 & g_0 & 608g_9 & 608g_8 \\ g_8 & g_7 & g_6 & g_5 & g_4 & g_3 & g_2 & g_1 & g_0 & 608g_9 \\ g_9 & g_8 & g_7 & g_6 & g_5 & g_4 & g_3 & g_2 & g_1 & g_0 \end{bmatrix} \cdot \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{bmatrix} = \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix}$$

Bu gösterimde sabit katsayı 608 (10-bit uzunluğunda) olduğu için, limb değerleri yazmaçlara sığmayacak ve işlemi yapmak mümkün olmayacaktır. Ancak bu duruma rağmen nihai değer yazmaçlara sığmaktadır. Bu durumda 608 katsayısının oluşturduğu bu problemi çözmek için matris

gösterimini aşağıdaki gibi parçalı olarak yazabiliriz:

$$\left(\left[\begin{array}{ccccc|ccccc} g_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ g_1 & g_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ g_2 & g_1 & g_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ g_3 & g_2 & g_1 & g_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ g_4 & g_3 & g_2 & g_1 & g_0 & 0 & 0 & 0 & 0 & 0 \\ \hline g_5 & g_4 & g_3 & g_2 & g_1 & g_0 & 0 & 0 & 0 & 0 \\ g_6 & g_5 & g_4 & g_3 & g_2 & g_1 & g_0 & 0 & 0 & 0 \\ g_7 & g_6 & g_5 & g_4 & g_3 & g_2 & g_1 & g_0 & 0 & 0 \\ g_8 & g_7 & g_6 & g_5 & g_4 & g_3 & g_2 & g_1 & g_0 & 0 \\ g_9 & g_8 & g_7 & g_6 & g_5 & g_4 & g_3 & g_2 & g_1 & g_0 \end{array}\right] + 608 \left[\begin{array}{ccccc|ccccc} 0 & g_9 & g_8 & g_7 & g_6 & g_5 & g_4 & g_3 & g_2 & g_1 \\ 0 & 0 & g_9 & g_8 & g_7 & g_6 & g_5 & g_4 & g_3 & g_2 \\ 0 & 0 & 0 & g_9 & g_8 & g_7 & g_6 & g_5 & g_4 & g_3 \\ 0 & 0 & 0 & 0 & g_9 & g_8 & g_7 & g_6 & g_5 & g_4 \\ 0 & 0 & 0 & 0 & 0 & g_9 & g_8 & g_7 & g_6 & g_5 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & g_9 & g_8 & g_7 & g_6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & g_9 & g_8 & g_7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & g_9 & g_8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & g_9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]\right) \cdot \left[\begin{array}{c} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ \hline f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{array}\right]$$

Bu gösterim Toeplitz matrisi yöntemlerinin kullanımı için uygulanabilir olmaktadır. Burada öncelikli işlem 608 katsayısı ile yapılacak çarpma işlemi olacaktır. 608 sayısı aşağıdaki gibi yazılabilir:

$$608 = 2^5 \cdot 19 = 2^5(2^4 + 2^1 + 1) = 2^9 + 2^6 + 2^5$$

Bu gösterim sayesinde 608 katsayısı ile çarpma işlemi aşağıdaki şekilde ifade edilebilir.

$$x \cdot 608 = (x << 9) + (x << 6) + (x << 5)$$

Dolayısıyla, 608 ile çarpma işleminin işlem maliyeti 3 **kaydırma** and 2$\mathbf{T_i}$ olacaktır. Buradaki kaydırma işleminin maliyeti ihmal edilerek, 10 çarpma işlemi için toplamda $20\mathbf{T_i}$ işlem maliyeti olacaktır.

NIST'in önerdiği P-521 ile E-521 eğrilerinin 32-bit platformlarda gerçekleştirilmesi için ise cisim elemanlarının aşağıdaki gibi 18 terimli olarak gösterilmesinin yeni sonuçların elde edilebilmesi için uygun olacağı sonucuna varılmıştır. X ve Y, $\mathbb{F}_{2^{521}-1}$ cisminin elemanı olmak üzere aşağıdaki gibi gösterilirler:

$$X = x_0 + 2^{29}x_1 + 2^{58}x_2 + \cdots + 2^{464}x_{17} + 2^{493}x_{17}$$

$$Y = y_0 + 2^{29}y_1 + 2^{58}y_2 + \cdots + 2^{464}y_{17} + 2^{493}y_{17}$$

Bu gösterimde çarpma işlemi yapıldığı zaman elde edilen Toeplitz matris vektör çarpımı aşağı-

daki gibi olur:

$$\begin{bmatrix} x_0 & 2x_{17} & 2x_{16} & 2x_{15} & 2x_{14} & 2x_{13} & 2x_{12} & 2x_{11} & 2x_{10} & 2x_9 & 2x_8 & 2x_7 & 2x_6 & 2x_5 & 2x_4 & 2x_3 & 2x_2 & 2x_1 \\ x_1 & x_0 & 2x_{17} & 2x_{16} & 2x_{15} & 2x_{14} & 2x_{13} & 2x_{12} & 2x_{11} & 2x_{10} & 2x_9 & 2x_8 & 2x_7 & 2x_6 & 2x_5 & 2x_4 & 2x_3 & 2x_2 \\ x_2 & x_1 & x_0 & 2x_{17} & 2x_{16} & 2x_{15} & 2x_{14} & 2x_{13} & 2x_{12} & 2x_{11} & 2x_{10} & 2x_9 & 2x_8 & 2x_7 & 2x_6 & 2x_5 & 2x_4 & 2x_3 \\ x_3 & x_2 & x_1 & x_0 & 2x_{17} & 2x_{16} & 2x_{15} & 2x_{14} & 2x_{13} & 2x_{12} & 2x_{11} & 2x_{10} & 2x_9 & 2x_8 & 2x_7 & 2x_6 & 2x_5 & 2x_4 \\ x_4 & x_3 & x_2 & x_1 & x_0 & 2x_{17} & 2x_{16} & 2x_{15} & 2x_{14} & 2x_{13} & 2x_{12} & 2x_{11} & 2x_{10} & 2x_9 & 2x_8 & 2x_7 & 2x_6 & 2x_5 \\ x_5 & x_4 & x_3 & x_2 & x_1 & x_0 & 2x_{17} & 2x_{16} & 2x_{15} & 2x_{14} & 2x_{13} & 2x_{12} & 2x_{11} & 2x_{10} & 2x_9 & 2x_8 & 2x_7 & 2x_6 \\ x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_0 & 2x_{17} & 2x_{16} & 2x_{15} & 2x_{14} & 2x_{13} & 2x_{12} & 2x_{11} & 2x_{10} & 2x_9 & 2x_8 & 2x_7 \\ x_7 & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_0 & 2x_{17} & 2x_{16} & 2x_{15} & 2x_{14} & 2x_{13} & 2x_{12} & 2x_{11} & 2x_{10} & 2x_9 & 2x_8 \\ x_8 & x_7 & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_0 & 2x_{17} & 2x_{16} & 2x_{15} & 2x_{14} & 2x_{13} & 2x_{12} & 2x_{11} & 2x_{10} & 2x_9 \\ x_9 & x_8 & x_7 & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_0 & 2x_{17} & 2x_{16} & 2x_{15} & 2x_{14} & 2x_{13} & 2x_{12} & 2x_{11} & 2x_{10} \\ x_{10} & x_9 & x_8 & x_7 & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_0 & 2x_{17} & 2x_{16} & 2x_{15} & 2x_{14} & 2x_{13} & 2x_{12} & 2x_{11} \\ x_{11} & x_{10} & x_9 & x_8 & x_7 & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_0 & 2x_{17} & 2x_{16} & 2x_{15} & 2x_{14} & 2x_{13} & 2x_{12} \\ x_{12} & x_{11} & x_{10} & x_9 & x_8 & x_7 & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_0 & 2x_{17} & 2x_{16} & 2x_{15} & 2x_{14} & 2x_{13} \\ x_{13} & x_{12} & x_{11} & x_{10} & x_9 & x_8 & x_7 & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_0 & 2x_{17} & 2x_{16} & 2x_{15} & 2x_{14} \\ x_{14} & x_{13} & x_{12} & x_{11} & x_{10} & x_9 & x_8 & x_7 & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_0 & 2x_{17} & 2x_{16} & 2x_{15} \\ x_{15} & x_{14} & x_{13} & x_{12} & x_{11} & x_{10} & x_9 & x_8 & x_7 & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_0 & 2x_{17} & 2x_{16} \\ x_{16} & x_{15} & x_{14} & x_{13} & x_{12} & x_{11} & x_{10} & x_9 & x_8 & x_7 & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_0 & 2x_{17} \\ x_{17} & x_{16} & x_{15} & x_{14} & x_{13} & x_{12} & x_{11} & x_{10} & x_9 & x_8 & x_7 & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_0 \end{bmatrix} \cdot \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \\ y_{11} \\ y_{12} \\ y_{13} \\ y_{14} \\ y_{15} \\ y_{16} \\ y_{17} \end{bmatrix}$$

4.1.2 P-521 ve E-521 İçin Yeni Cisim Çarpma Algoritması Geliştirilmesi ve Gerçeklenmesi

$\mathbb{F}_{2^{521}-1}$ cismi üzerinde çarpma işlemi için Toeplitz matris vektör çarpımı (TMVÇ) kullanılmıştır. Rekürsif yaklaşımın birinci seviyesinde matrisler 3'e bölünerek boyutları 3 olan Toeplitz matris vektör çarpımları kullanılmıştır. Bu metot ile 6 tane, boyutları 6 olan Toeplitz matris vektör çarpımı oluşturulmuştur. İkinci seviye rekürsif hesaplar için ise yapılan çalışmalar $2x2$'lik Toeplitz matris vektör çarpımının iyi sonuçlar getirdiğini göstermiştir. Bu çarpma ise her bir 6 boyutlu çarpımlar için 3 tane, boyutu 3 olan çarpım çağırmış; toplamda 18 tane boyutu 3 olan Toeplitz matris vektör çarpımı elde edilmiştir. Son adımda ise klasik yöntem kullanarak hesap sonlandırılmıştır. Sonuç olarak elde edilen aritmetik karmaşıklığının $189\mathbf{Ç} + 85\mathbf{T} + 198\mathbf{T_i}$ olduğu hesaplanmıştır. Yapılan gerçekleştirmeler sonucunda elde ettiğimiz sonuçlar aşağıda verilmiştir. Yapılan gerçekleştirmeler C programlama dili kullanarak Intel(R) Core i5 - 6420P CPU @2.80 GHz makinesinde çalıştırılmıştır.

Geliştirdiğimiz algoritma, P-521 ve E-521 eğrilerine uygulanmıştır. P-521 eğrisi NIST tarafından önerilmiştir ve yaygın olarak kullanılmaktadır. Bu eğri için (Bos vd. 2016) referansında verilen sabit-zaman değişken-taban çarpım algoritmaları gerçekleştirilmiştir. Nokta aritmetik işlemleri için ise "Explicit Formulas Database" internet sayfasındaki formüller kullanılmıştır. Skaler çarpma

Yöntem	Aritmetik Karmaşıklık	Saat Çevrimi
Kare alma (64-bit)	45**Ç**+ 91**T**	110
Granger-Scott (64-bit)	45**Ç**+ 176**T**	165
Karma	54**Ç**+ 133**T**	136
Kare alma (32-bit)	171**Ç**+ 343**T**	442
TMVÇ (32-bit)	189**Ç**+ 481**T**	554

Tablo 2: $\mathbb{F}_{2^{521}-1}$ Cismi Üzerinde Çarpma İşlemi Maliyet Tablosu

işlemi için sabit-pencere ile birlikte sabit-zaman değişken-taban algoritması (Ali ve Cenk 2017; Brown 2011)'da verilen iyileştirme teknikleri ile birlikte kullanılmıştır. Ayrıca, sabit-zaman sabit-taban skaler çarpması için çevrim dışı hesaplamaların zamanlaması önemli olmadığından afin koordinatlar kullanılmıştır. Diğer taraftan, E-521 eğrisi SafeCurves (Bernstein ve Lange 2018) internet sitesinden alınmıştır ve bu sitede E-521'in Bernstein-Lange ve Hamburg-Aranha tarafından bağımsız olarak tavsiye edildiğinden söz edilmektedir. (Bernstein, Chuengsatiansup, vd. 2014)'te belirtildiği gibi, Edwards eğrilerinde hızlı nokta toplama formülleri genişletilmiş projektif koordinatlar ile ifade edilmiştir. Bu nedenle, proje kapsamındaki çalışmalarda hem değişken ve hem de sabit tabanlı skaler çarpma için genişletilmiş koordinatlar kullanılmıştır. Edwards eğrilerinin bir avantajı nokta aritmetiği formüllerinin daha sade olması sebebiyle NIST eğrilerinden daha hızlı olmasıdır. Ayrıca, tam toplam formüllerinden dolayı, güvenli bir şekilde gerçeklenmesi daha kolaydır ve istisnai bir durum yoktur. Dolayısı ile, P-521'deki skaler çarpma işleminden farklı olarak, E-521'deki skaler çarpma işlemi verimli, sabit zamanlı ve istisnasızdır. E-521'deki sabit zamanlı değişken tabanlı skaler çarpımda verimlilik için (Ali ve Cenk 2017; Bernstein, Chuengsatiansup, vd. 2014; Granger ve Scott 2014)'da kullanılan strateji ve nokta işlemleri kullanılmıştır. Sadece tek skalerlerin yeniden kodlanmasından farklı olarak, sonuçların kolay doğrulanmasını sağlamak için aynı çift skaleri (Ali ve Cenk 2017; Brown 2011)'nın umumi kodunda kullanılmıştır. E-521'deki sabit-zamanlı sabit-tabanlı skaler çarpma işlemi için değişken tabanlıda kullandığımız nokta işlem formüllerini kullanan (Bos vd. 2016)'daki Algoritma 7 gerçekleştirilmiştir. Verimli Çevrimiçi hesaplamayı sağlayabilmek adına önceden hesaplanmış tablodaki noktalar da dahil olmak üzere tüm noktalar için genişletilmiş projektif koordinatlar kullanılmıştır. P-521'deki sabit tabanlı skaler çarpmadaki gibi tablo parametresi $v = 3$ olarak alınmıştır. Böylece, önceden hesaplanmış tablo boyutları 4,5 ve 6 pencere genişliği için sırasıyla 6912, 13824 ve 27648-byte olmuştur. $v = 4$ değerinin kullanılması, t taban noktasının bit uzunluğu(519-bit) olmak üzere $e = [t/vw]$ değerini düşürmesine rağmen önceden hesaplamış tablo boyutunu ve hesaplama evresinde (nispeten) masraflı toplama işlemlerinin sayısını arttırmaktadır. Zamanlamalar Tablo 3'de verilmiştir. Buradaki gerçekleştirmeler NIST P-521 ve Edwards E-521 eğrileri için yapılmıştır. Gerçekleştirmelerde Ubuntu

Versiyon	Eğri	Değişken Taban			Sabit Taban		
		$w = 4$	$w = 5$	$w = 6$	$w = 4$	$w = 5$	$w = 6$
64-bit	P-521	962456	963642	1019281	347800	317652	319855
	E-521	806305	822793	875793	368074	335077	370056
32-bit	P-521	3784073	3774360	3937199	1352660	1191477	1136798
	E-521	3115820	3091681	3233818	1312680	1227394	1181080

Tablo 3: P-521 ve E-521 için Zamanlama Değerleri

16.04 LTS sisteminde gcc 5.40 derleyicisinin 3. seviye optimizasyonu kullanılarak saat çevrimleri bulunmuştur. Test Intel® Core i5 - 6420P CPU @ 2.80GHz bir bilgisayarda tek çekirdek üzerinde Turbo Boost kapalı şekilde yapılmıştır.

4.1.3 Curve25519 İçin Yeni Cisim Çarpma Algoritması Geliştirilmesi ve Gerçeklenmesi

Geliştirdiğimiz yeni matris gösterimi sayesinde Curve25519 üzerinde cisim çarpması için Toeplitz matrisi vektör çarpımı (TMVÇ) işlemleri uygulanabilir hale gelmiştir. Yukarıdaki gösterim dikkate alındığında her iki $10x10$ boyutlu matris, $5x5$ boyutlu küçük matrislere bölünerek TMVÇ işlemleri bu $5x5$ boyutlu matrisler üzerinde gerçekleştirilmiştir. Bu gösterim aşağıda ifade edilmiştir.

$$\left[\begin{array}{c|c} A_{0_{5x5}} & 0_{5x5} \\ \hline A_{1_{5x5}} & A_{0_{5x5}} \end{array}\right]_{10x10} \cdot \left[\begin{array}{c} B_{0_{5x1}} \\ \hline B_{1_{5x1}} \end{array}\right]_{10x1} + 608 \cdot \left[\begin{array}{c|c} A_{2_{5x5}} & A_{1_{5x5}} \\ \hline 0_{5x5} & A_{2_{5x5}} \end{array}\right]_{10x10} \cdot \left[\begin{array}{c} B_{0_{5x1}} \\ \hline B_{1_{5x1}} \end{array}\right]_{10x1}$$

Toplamda 8 farklı alt matris olduğu halde matrislerin alt üçgen ve üst üçgen yapısından dolayı 2 alt matris 0 matrisi yapısındadır. Dolayısıyla, matris, yukarıda gösterildiği üzere 6 farklı alt matrise ayrılmıştır. Her bir alt matris için toplam 6 farklı çarpma işlemi göz önüne alınacaktır.

Burada, $5x5$ boyutundaki matrisler tekrar alt matrislere bölünerek $4x4$, $1x4$, $1x1$ ve $4x1$ boyutunda alt matrisler elde edilmiştir. Bu şekilde 4'e 1 kolon asimetrik bölme uygulayarak, $2x2$ Toeplitz matris vektör çarpma işlemi uygulanabilir hale getirilmiştir. Aynı şekilde matrisin 3'e 2 kolon ve tersi şeklinde de bölünmesi ihtimalleri incelenmiş, tüm 6 matris için toplam $4^4 + 4^4 = 512$ farklı ihtimal göz önüne alınarak işlem sayısı en etkin olan form tespit edilmiştir. Bu çalışma sonucu, yeni bir algoritma elde edilmiştir.

6 alt matrise ait işlemler $A_0B_0, A_0B_1, A_1B_0, A_1B_1, A_2B_0$ ve A_2B_1 olarak ifade edilebilir. $Cost()$ fonksiyonu parametre olarak verilen TMVÇ çarpma işleminin maliyetini hesaplayan fonksiyon olarak tanımlanmıştır.

A_0B_0 Çarpımının Hesaplanması

$$A_0B_0 = \left[\begin{array}{c|c} K_{2_{1x4}} & K_{3_{1x1}} \\ \hline K_{0_{4x4}} & K_{1_{4x1}} \end{array}\right]_{5x5} \cdot \left[\begin{array}{c} L_{0_{4x1}} \\ \hline L_{1_{1x1}} \end{array}\right]_{5x1} = \left[\begin{array}{c} K_2L_{0_{1x1}} + K_3L_{1_{1x1}} \\ \hline K_0L_{0_{4x1}} + K_1L_{1_{4x1}} \end{array}\right]_{5x5}$$

K_0L_0 : TMVÇ yöntemi ile sonra hesaplanacaktır.

K_1L_1 : $\begin{bmatrix} 0 \\ 0 \\ 0 \\ g_0 \end{bmatrix} \cdot \begin{bmatrix} f_4 \end{bmatrix}$ $= \cancel{4\text{Ç}} = 1\text{Ç}$

K_2L_0 : $\begin{bmatrix} g_0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix}$ $= \cancel{4\text{Ç} + 3\text{T}_\text{i}} = 1\text{Ç}$

K_3L_1 : $\begin{bmatrix} 0 \end{bmatrix} \cdot \begin{bmatrix} f_4 \end{bmatrix}$ $= \cancel{1\text{Ç}} = 0$

Son toplama işlemlerinin maliyeti $5\mathbf{T_i}$ dir. Ancak K_3L_1 işleminin maliyeti olmadığından ve K_1L_1 işlemi sadece $1\mathbf{Ç}$ maliyete sahip olduğundan, toplam maliyet:

$$Cost(K_0L_0) + 2\text{Ç} + 1\text{T}_\text{i}$$

A_0B_1 Çarpımının Hesaplanması

$$A_0B_1 = \left[\begin{array}{c|c} K_{2_{1x4}} & K_{3_{1x1}} \\ \hline K_{0_{4x4}} & K_{1_{4x1}} \end{array}\right]_{5x5} \cdot \left[\begin{array}{c} N_{0_{4x1}} \\ \hline N_{1_{1x1}} \end{array}\right]_{5x1} = \left[\begin{array}{c} K_2N_{0_{1x1}} + K_3N_{1_{1x1}} \\ \hline K_0N_{0_{4x1}} + K_1N_{1_{4x1}} \end{array}\right]_{5x5}$$

K_0N_0 : TMVÇ yöntemi ile sonra hesaplanacaktır.

$$K_1N_1 : \begin{bmatrix} 0 \\ 0 \\ 0 \\ g_0 \end{bmatrix} \cdot \begin{bmatrix} f_9 \end{bmatrix} \qquad = \cancel{4\text{Ç}} = 1\text{Ç}$$

$$K_2N_0 : \begin{bmatrix} g_0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} f_5 \\ f_6 \\ f_7 \\ f_8 \end{bmatrix} \qquad = \cancel{4\text{Ç} + 3\text{T}_\text{i}} = 1\text{Ç}$$

$$K_3N_1 : \begin{bmatrix} 0 \end{bmatrix} \cdot \begin{bmatrix} f_9 \end{bmatrix} \qquad = \cancel{1\text{Ç}} = 0$$

Son toplama işlemlerinin maliyeti $5\textbf{T}_\textbf{i}$ dir. Ancak K_3N_1 işleminin maliyeti olmadığından ve K_1N_1 işlemi sadece 1**Ç** maliyete sahip olduğundan, toplam maliyet:

$$Cost(K_0N_0) + 2\text{Ç} + 1\text{T}_\text{i}$$

A_1B_0 Çarpımının Hesaplanması

$$A_1B_0 = \left[\begin{array}{c|c} P_{0_{4x4}} & P_{1_{4x1}} \\ \hline P_{2_{1x4}} & P_{3_{1x1}} \end{array}\right]_{5x5} \cdot \left[\begin{array}{c} L_{0_{4x1}} \\ \hline L_{1_{1x1}} \end{array}\right]_{5x1} = \left[\begin{array}{c} P_0L_{0_{4x1}} + P_1L_{1_{4x1}} \\ \hline P_2L_{0_{1x1}} + P_3L_{1_{1x1}} \end{array}\right]_{5x5}$$

P_0L_0 : TMVÇ yöntemi ile sonra hesaplanacaktır.

$$P_1L_1 : \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} \cdot \begin{bmatrix} f_4 \end{bmatrix} \qquad = 4\text{Ç}$$

$$P_2L_0 : \begin{bmatrix} g_9 & g_8 & g_7 & g_6 \end{bmatrix} \cdot \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix} \qquad = 4\text{Ç} + 3\text{T}_\text{i}$$

$$P_3L_1 : \begin{bmatrix} g_5 \end{bmatrix} \cdot \begin{bmatrix} f_4 \end{bmatrix} \qquad = 1\text{Ç}$$

Son toplama işlemlerinin maliyeti $5\textbf{T}_\textbf{i}$ dir. Toplam maliyet:

$$Cost(P_0L_0) + 9\text{Ç} + 8\text{T}_\text{i}$$

A_1B_1 Çarpımının Hesaplanması

$$A_1B_1 = \left[\begin{array}{c|c} P_{0_{4x4}} & P_{1_{4x1}} \\ \hline P_{2_{1x4}} & P_{3_{1x1}} \end{array}\right]_{5x5} \cdot \left[\begin{array}{c} N_{0_{4x1}} \\ \hline N_{1_{1x1}} \end{array}\right]_{5x1} = \left[\begin{array}{c} P_0N_{0_{4x1}} + P_1N_{1_{4x1}} \\ \hline P_2N_{0_{1x1}} + P_3N_{1_{1x1}} \end{array}\right]_{5x5}$$

P_0N_0 : TMVÇ yöntemi ile sonra hesaplanacaktır.

$$P_1N_1 : \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} \cdot \begin{bmatrix} f_9 \end{bmatrix} = 4\textbf{Ç}$$

$$P_2N_0 : \begin{bmatrix} g_9 & g_8 & g_7 & g_6 \end{bmatrix} \cdot \begin{bmatrix} f_5 \\ f_6 \\ f_7 \\ f_8 \end{bmatrix} = 4\textbf{Ç} + 3\textbf{T}_\textbf{i}$$

$$P_3N_1 : \begin{bmatrix} g_5 \end{bmatrix} \cdot \begin{bmatrix} f_9 \end{bmatrix} = 1\textbf{Ç}$$

Son toplama işlemlerinin maliyeti $5\textbf{T}_\textbf{i}$ dir. Toplam maliyet:

$$Cost(P_0N_0) + 9\textbf{Ç} + 8\textbf{T}_\textbf{i}$$

A_2B_0 Çarpımının Hesaplanması

$$A_2B_0 = \left[\begin{array}{c|c} R_{1_{4x1}} & R_{0_{4x4}} \\ \hline R_{3_{1x1}} & R_{2_{1x4}} \end{array}\right]_{5x5} \cdot \left[\begin{array}{c} L'_{1_{1x1}} \\ \hline L'_{0_{4x1}} \end{array}\right]_{5x1} = \left[\begin{array}{c} R_0L'_{0_{4x1}} + R_1L'_{1_{4x1}} \\ \hline R_2L'_{0_{1x1}} + R_3L'_{1_{1x1}} \end{array}\right]_{5x5}$$

$R_0L'_0$: TMVÇ yöntemi ile sonra hesaplanacaktır.

$$R_1L'_1 : \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} f_0 \end{bmatrix} = 4\cancel{\textbf{Ç}} = 0$$

$$R_2L'_0 : \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \cancel{4\textbf{Ç} + 3\textbf{T}_\textbf{i}} = 0$$

$$R_3L'_1 : \begin{bmatrix} 0 \end{bmatrix} \cdot \begin{bmatrix} f_0 \end{bmatrix} = 1\cancel{\textbf{Ç}} = 0$$

Son toplama işlemlerinin maliyeti $5\mathbf{T_i}$ dir. Toplam maliyet:

$$Cost(R_0 L_0^{'})$$

A_2B_1 Çarpımının Hesaplanması

$$A_2B_0 = \left[\begin{array}{c|c} R_{1_{4x1}} & R_{0_{4x4}} \\ \hline R_{3_{1x1}} & R_{2_{1x4}} \end{array}\right]_{5x5} \cdot \left[\begin{array}{c} \frac{N_{1_{1x1}}^{'}}{} \\ N_{0_{4x1}}^{'} \end{array}\right]_{5x1} = \left[\begin{array}{c} R_0 N_{0_{4x1}}^{'} + R_1 N_{1_{4x1}}^{'} \\ \hline R_2 N_{0_{1x1}}^{'} + R_3 N_{1_{1x1}}^{'} \end{array}\right]_{5x5}$$

$R_0 N_0^{'}$: TMVÇ yöntemi ile sonra hesaplanacaktır.

$$R_1 N_1^{'}: \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} f_5 \end{bmatrix} = 4\cancel{\mathbf{Ç}} = 0$$

$$R_2 N_0^{'}: \quad \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} f_6 \\ f_7 \\ f_8 \\ f_9 \end{bmatrix} = 4\cancel{\mathbf{Ç} + 3\mathbf{T_i}} = 0$$

$$R_3 N_1^{'}: \quad \begin{bmatrix} 0 \end{bmatrix} \cdot \begin{bmatrix} f_5 \end{bmatrix} = 1\cancel{\mathbf{Ç}} = 0$$

Son toplama işlemlerinin maliyeti $5\mathbf{T_i}$ dir. Toplam maliyet:

$$Cost(R_0 N_0^{'})$$

K_0L_0 ve K_0N_0 Çarpımlarının Hesaplanması

$$\begin{aligned} K_0 L_0 &= \left[\begin{array}{cc|cc} g_1 & g_0 & 0 & 0 \\ g_2 & g_1 & g_0 & 0 \\ \hline g_3 & g_2 & g_1 & g_0 \\ g_4 & g_3 & g_2 & g_1 \end{array}\right] \cdot \left[\begin{array}{c} f_0 \\ f_1 \\ \hline f_2 \\ f_3 \end{array}\right] \\ &= \left[\begin{array}{c|c} X_0 & X_1 \\ \hline X_2 & X_0 \end{array}\right] \cdot \left[\begin{array}{c} C_0 \\ \hline C_1 \end{array}\right] \\ &= \left[\begin{array}{c} T_1 + T_2 \\ \hline T_1 + T_3 \end{array}\right] \end{aligned}$$

$T_1 = X_0(C_0 + C_1)$, $T_2 = C_1(X_1 - X_0)$, $T_3 = C_0(X_2 - X_0)$

Toplam maliyet:

$$Cost(K_0L_0) = 3\,(M(2)) + 8\mathbf{T} + 4\mathbf{T_i}$$

Burada $M(2)$, $2x2$ TMVÇ işleminin maliyetini ifade etmektedir.

$$\begin{aligned} K_0N_0 &= \left[\begin{array}{cc|cc} g_1 & g_0 & 0 & 0 \\ g_2 & g_1 & g_0 & 0 \\ \hline g_3 & g_2 & g_1 & g_0 \\ g_4 & g_3 & g_2 & g_1 \end{array}\right] \cdot \left[\begin{array}{c} f_5 \\ f_6 \\ \hline f_7 \\ f_8 \end{array}\right] \\ &= \left[\begin{array}{c|c} X_0 & X_1 \\ \hline X_2 & X_0 \end{array}\right] \cdot \left[\begin{array}{c} D_0 \\ \hline D_1 \end{array}\right] \\ &= \left[\begin{array}{c} T_1 + T_2 \\ \hline T_1 + T_3 \end{array}\right] \end{aligned}$$

$T_1 = X_0(D_0 + D_1)$, $T_2 = D_1(X_1 - X_0)$, $T_3 = D_0(X_2 - X_0)$

$(X_1 - X_0)$ ve $(X_2 - X_0)$ zaten hesaplamış olduğu için toplam maliyet:

$$\begin{array}{rrl} Cost(K_0N_0) = 3\,(M(2)) & +8\mathbf{T} & +4\mathbf{T_i} \\ & -3\mathbf{T} & \leftarrow (X_1 - X_0) \\ + & -3\mathbf{T} & \leftarrow (X_2 - X_0) \\ \hline 3\,(M(2)) & +2\mathbf{T} & +4\mathbf{T_i} \end{array}$$

Burada $M(2)$, $2x2$ TMVÇ işleminin maliyetini ifade etmektedir.

P_0L_0 ve P_0N_0 Çarpımlarının Hesaplanması

$$P_0L_0 = \left[\begin{array}{cc|cc} g_5 & g_4 & g_3 & g_2 \\ g_6 & g_5 & g_4 & g_3 \\ \hline g_7 & g_6 & g_5 & g_4 \\ g_8 & g_7 & g_6 & g_5 \end{array}\right] \cdot \left[\begin{array}{c} f_0 \\ f_1 \\ \hline f_2 \\ f_3 \end{array}\right]$$
$$= \left[\begin{array}{c|c} Y_0 & Y_1 \\ \hline Y_2 & Y_0 \end{array}\right] \cdot \left[\begin{array}{c} C_0 \\ \hline C_1 \end{array}\right]$$
$$= \left[\begin{array}{c} T_1 + T_2 \\ \hline T_1 + T_3 \end{array}\right]$$

$T_1 = Y_0(C_0 + C_1)$, $T_2 = C_1(Y_1 - Y_0)$, $T_3 = C_0(Y_2 - Y_0)$

$(C_0 + C_1)$ zaten hesaplanmış olduğu için toplam maliyet:

$$\begin{array}{rl} Cost(P_0L_0) = & 3\,(M(2)) + 8\mathbf{T} + 4\mathbf{T_i} \\ + & -2\mathbf{T} \quad \leftarrow (C_0 + C_1) \\ \hline & 3\,(M(2)) + 6\mathbf{T} + 4\mathbf{T_i} \end{array}$$

Burada $M(2)$, $2x2$ TMVÇ işleminin maliyetini ifade etmektedir.

$$P_0N_0 = \left[\begin{array}{cc|cc} g_5 & g_4 & g_3 & g_2 \\ g_6 & g_5 & g_4 & g_3 \\ \hline g_7 & g_6 & g_5 & g_4 \\ g_8 & g_7 & g_6 & g_5 \end{array}\right] \cdot \left[\begin{array}{c} f_5 \\ f_6 \\ \hline f_7 \\ f_8 \end{array}\right]$$
$$= \left[\begin{array}{c|c} Y_0 & Y_1 \\ \hline Y_2 & Y_0 \end{array}\right] \cdot \left[\begin{array}{c} D_0 \\ \hline D_1 \end{array}\right]$$
$$= \left[\begin{array}{c} T_1 + T_2 \\ \hline T_1 + T_3 \end{array}\right]$$

$T_1 = Y_0(D_0 + D_1)$, $T_2 = D_1(Y_1 - Y_0)$, $T_3 = C_0(Y_2 - Y_0)$

$(D_0 + D_1)$, $(Y_1 - Y_0)$ ve $(Y_2 - Y_0)$ zaten hesaplanmış olduğu için toplam maliyet:

$$\begin{array}{rrl} Cost(P_0N_0) = 3\,(M(2)) & +8\mathbf{T} & +4\mathbf{T_i} \\ & -2\mathbf{T} & \leftarrow (D_0 + D_1) \\ & -3\mathbf{T} & \leftarrow (Y_1 - Y_0) \\ + & -3\mathbf{T} & \leftarrow (Y_2 - Y_0) \\ \hline 3\,(M(2)) & +4\mathbf{T_i} & \end{array}$$

Burada $M(2)$, $2x2$ TMVÇ işleminin maliyetini ifade etmektedir.

$R_0L_0^{'}$ ve $R_0N_0^{'}$ Çarpımının Hesaplanması

$$\begin{aligned} R_0L_0^{'} &= \left[\begin{array}{cc|cc} g_9 & g_8 & g_7 & g_6 \\ 0 & g_9 & g_8 & g_7 \\ \hline 0 & 0 & g_9 & g_8 \\ 0 & 0 & 0 & g_9 \end{array}\right] \cdot \left[\begin{array}{c} f_1 \\ f_2 \\ \hline f_3 \\ f_4 \end{array}\right] \\ &= \left[\begin{array}{c|c} Z_0 & Y_2 \\ \hline 0 & Z_0 \end{array}\right] \cdot \left[\begin{array}{c} C_0^{'} \\ \hline C_1^{'} \end{array}\right] \\ &= \left[\begin{array}{c} T_1 + T_2 \\ \hline T_1 + T_3 \end{array}\right] \end{aligned}$$

$T_1 = Z_0(C_0^{'} + C_1^{'})$, $T_2 = C_1^{'}(Y_2 - Z_0)$, $T_3 = C_0^{'}(0 - Z_0)$

Toplam maliyet:

$$Cost(R_0L_0^{'}) = 3\,(M(2)) + 5\mathbf{T} + 4\mathbf{T_i}$$

Burada $M(2)$, $2x2$ TMVÇ işleminin maliyetini ifade etmektedir.

$$R_0N_0^{'} = \left[\begin{array}{cc|cc} g_9 & g_8 & g_7 & g_6 \\ 0 & g_9 & g_8 & g_7 \\ \hline 0 & 0 & g_9 & g_8 \\ 0 & 0 & 0 & g_9 \end{array}\right] \cdot \left[\begin{array}{c} f_6 \\ f_7 \\ \hline f_8 \\ f_9 \end{array}\right]$$
$$= \left[\begin{array}{c|c} Z_0 & Y_2 \\ \hline 0 & Z_0 \end{array}\right] \cdot \left[\begin{array}{c} D_0^{'} \\ \hline D_1^{'} \end{array}\right]$$
$$= \left[\begin{array}{c} T_1 + T_2 \\ \hline T_1 + T_3 \end{array}\right]$$

$T_1 = Z_0(D_0^{'} + D_1^{'}), T_2 = D_1^{'}(Y_2 - Z_0), T_3 = D_0^{'}(0 - Z_0)$

$(Y_2 - Z_0)$ ve $(0 - Z_0)$ zaten hesaplanmış olduğu için toplam maliyet:

$$\begin{array}{rll} Cost(R_0N_0^{'}) = 3\,(M(2)) & +8\mathbf{T} & +4\mathbf{T_i} \\ & -3\mathbf{T} & \leftarrow (Y_2 - Z_0) \\ + & -3\mathbf{T} & \leftarrow (0 - Z_0) \\ \hline 3\,(M(2)) & +2\mathbf{T} & +4\mathbf{T_i} \end{array}$$

Burada $M(2)$, $2x2$ TMVÇ işleminin maliyetini ifade etmektedir.

A_0B_0, A_0B_1, A_1B_0, A_1B_1, A_2B_0 ve A_2B_1 işlemlerinin yukarıda hesaplanan işlem maliyetleri aşağıdaki tabloda listelenmiştir. $4x4$ TMVÇ işlemleri için $2x2$ TMVÇ işlemleri kullanılabilir. Bu

Çarpım	**Maliyet**
A_0B_0	Cost(K_0L_0) + 2**Ç**+ 1$\mathbf{T_i}$
A_0B_1	Cost(K_0N_0) + 2**Ç**+ 1$\mathbf{T_i}$
A_1B_0	Cost(P_0L_0) + 9**Ç**+ 8$\mathbf{T_i}$
A_1B_1	Cost(P_0N_0) + 9**Ç**+ 8$\mathbf{T_i}$
A_2B_0	Cost($R_0L_0^{'}$)
A_2B_1	Cost($R_0N_0^{'}$)

Tablo 4: Matris İşlemleri Maliyet Tablosu

durumda, M_{4x4}, $4x4$'lük TMVÇ ve M_{2x2}, $2x2$'lik TMVÇ işlemi olmak üzere maliyet hesabı aşağıdaki

gibi tanımlanabilir:

$$Cost(M_{4x4}) = Cost(M_{2x2}) + 8\mathbf{A} + 4\mathbf{A}_d$$

Bu hesaba göre toplam maliyet hesabı aşağıdaki gibi ifade edilebilir:

$$\begin{array}{rcccc}
Cost(K_0L_0) & = & 3\,(M_{2x2}) & +8\mathbf{T} & +4\mathbf{T_i} \\
Cost(K_0N_0) & = & 3\,(M_{2x2}) & +2\mathbf{T} & +4\mathbf{T_i} \\
Cost(P_0L_0) & = & 3\,(M_{2x2}) & +6\mathbf{T} & +4\mathbf{T_i} \\
Cost(P_0N_0) & = & 3\,(M_{2x2}) & & +4\mathbf{T_i} \\
Cost(R_0L_0^{'}) & = & 3\,(M_{2x2}) & +5\mathbf{T} & +4\mathbf{T_i} \\
Cost(R_0N_0^{'}) & = & 3\,(M_{2x2}) & +2\mathbf{T} & +4\mathbf{T_i} \\
 & + & & & \\
\hline
 & & 18\,(M_{2x2}) & +23\mathbf{T} & +24\mathbf{T_i}
\end{array}$$

$M_{2\times 2}$ işleminin maliyeti hem TMVÇ hem de klasik yöntemlerle hesaplanabilir. Bu sayede toplam maliyet için iki farklı değer elde edilmiştir. Bu maliyete 608 katsayısı ile çarpma maliyeti de eklendikten sonraki toplam maliyetler aşağıda listelenmiştir. Geliştirdiğimiz iki farklı algoritmanın

Yöntem	**Maliyet**
Yöntem #1 (TMVÇ)	76**Ç**+ 65**T**+ 98$\mathbf{T_i}$
Yöntem #2 (Klasik)	94**Ç**+ 11**T**+ 98$\mathbf{T_i}$
Bernstein (Bernstein 2006)	109**Ç**+ 90$\mathbf{T_i}$

Tablo 5: Algoritma Maliyetleri Karşılaştırma Tablosu

hesaplama maliyeti Bernstein'ın algoritmasının maliyeti ile kıyaslandığında aşağıdaki farklar elde edilmiştir.

- Yöntem #1: -33**Ç** ve $+41$**T**
- Yöntem #2: -15**Ç** ve $+13\mathbf{T_i}$

Algoritma'nın Gerçeklenmesi

Geliştirdiğimiz algoritma 32-bit kelime uzunluğuna odaklı olduğu için algoritma gerçeklemeleri 32-bit platformlar için geliştirilmiştir. Algoritmanın hızını test etmek için x86 ve 32-bit ARM platformları farklı yapılandırmalarla kullanılmıştır. x86 gerçeklemesi için Intel i7-4750HQ işlemci ve Apple LLVM gcc 4.2.1 derleyici kullanılmıştır. ARMv7 gerçeklemesi için de Raspberry Pi 2 donanımı ve Raspbian 8 üzerinde gcc 4.9.2 (Project 2017) derleyicisi kullanılmıştır.

Algoritma C programlama dilinde yazılmıştır ve yaygın olarak kullanılan ref10 (SUPERCOP 2017) algoritmasının `fe_mul` fonksiyonu ile kıyaslanmıştır. Derlemeler derleyicinin farklı optimizasyon seviyeleri ile gerçekleştirilmiş, en iyi sonuçlar 2. ve 3. seviyelerde x86 üzerinde elde edilmiş ve $\%13$'e kadar iyileşme tespit edilmiştir.

Örnek zamanlama sonuçları aşağıdaki tabloda verilmiştir. **Algoritma**

Platform	**Algoritma**	**Opt. Seviyesi**	**Zamanlama (Ort.)**
x86	Yöntem #1	*-O2*	75*ns*
x86	ref10	*-O2*	80*ns*
x86	Yöntem #1	*-O3*	70*ns*
x86	ref10	*-O3*	80*ns*
ARMv7l	Yöntem #1	*-O2*	2600*ns*
ARMv7l	ref10	*-O2*	2600*ns*

Tablo 6: Curve25519 Gerçekleme karşılaştırma tablosu

Elde edilen yeni algoritmanın sözde-kodu oluşturulmuştur. Oluşturulan sözde-kod üzerinde iyileştirme çalışmaları yapılarak işlem gücü-bellek ödünleşimi için ideal bir gerçekleme elde edilmiştir. Geliştirilen algoritmanın sözde-kodu Algoritma 1'de verilmiştir.

Algoritma 1 TMVÇ Yöntemi Kullanılarak $F_{2^{255}-19}$ Cismi Üzerinde Çarpma İşlemi

Girdi: $F \leftarrow [f_0, \ldots, f_9], G \leftarrow [g_0, \ldots, g_9] \in [0, 2^{26}-1]^9 \times [0, 2^{21}-1]$

Çıktı: $H \leftarrow [h_0, \ldots, h_9] \in [0, 2^{26}-1]^9 \times [0, 2^{21}-1]$ öyleki $F \cdot G \leftarrow H$ mod $(2^{255}-19)$

1: $M[0] \leftarrow g_0 f_0$ $M[4] \leftarrow g_0 f_4$ $M[5] \leftarrow g_0 f_5 + g_1 f_4$
2: $M[6] \leftarrow g_2 f_4$ $M[7] \leftarrow g_3 f_4$ $M[8] \leftarrow g_4 f_4$
3: $M[9] \leftarrow g_9 f_0 + g_8 f_1 + g_7 f_2 + g_6 f_3 + g_5 f_4$
4: $N[0] \leftarrow g_1 f_9$ $N[1] \leftarrow g_2 f_9$ $N[2] \leftarrow g_3 f_9$ $N[3] \leftarrow g_4 f_9$
5: $N[4] \leftarrow g_5 f_9 + g_9 f_5 + g_8 f_6 + g_7 f_7 + g_6 f_8$ $N[9] \leftarrow 0$
6:
7: $C_0[0] \leftarrow f_0$ $C_0[1] \leftarrow f_1$ $C_1[0] \leftarrow f_2$ $C_1[1] \leftarrow f_3$
8: $T_1 \leftarrow C_0[0] + C_1[0]$ $T_2 \leftarrow C_0[1] + C_1[1]$
9:
10: $T_3 \leftarrow -g_1$ $T_4 \leftarrow g_0 - g_2$ $T_5 \leftarrow -g_0$
11: $T_7 \leftarrow g_3 - g_1$ $T_8 \leftarrow g_4 - g_2$ $T_9 \leftarrow g_2 - g_0$
12:
13: $T_{11} \leftarrow T_1 + T_2$ $T_{14} \leftarrow g_1 T_{11}$
14: $T_{12} \leftarrow g_0 - g_1$ $T_{15} \leftarrow T_{12} T_2$
15: $T_{13} \leftarrow g_2 - g_1$ $T_{16} \leftarrow T_{13} T_1$
16: $V_1[0] \leftarrow T_{14} + T_{15}$ $V_1[1] \leftarrow T_{14} + T_{16}$
17:
18: $T_{19} \leftarrow C_1[0] + C_1[1]$ $T_{14} \leftarrow T_3 T_{19}$
19: $T_{12} \leftarrow T_5 - T_3$ $T_{15} \leftarrow T_{12} C_1[1]$
20: $T_{13} \leftarrow T_4 - T_3$ $T_{16} \leftarrow T_{13} C_1[0]$
21: $V_2[0] \leftarrow T_{14} + T_{15}$ $V_2[1] \leftarrow T_{14} + T_{16}$
22:
23: $T_{20} \leftarrow C_0[0] + C_0[1]$ $T_{14} \leftarrow T_7 T_{20}$
24: $T_{12} \leftarrow T_9 - T_7$ $T_{15} \leftarrow T_{12} C_0[1]$
25: $T_{13} \leftarrow T_8 - T_7$ $T_{16} \leftarrow T_{13} C_0[0]$
26: $V_3[0] \leftarrow T_{14} + T_{15}$ $V_3[1] \leftarrow T_{14} + T_{16}$
27:
28: $KL[0] \leftarrow V_1[0] + V_2[0]$ $KL[2] \leftarrow V_1[0] + V_3[0]$
29: $KL[1] \leftarrow V_1[1] + V_2[1]$ $KL[3] \leftarrow V_1[1] + V_3[1]$
30:
31: $D_0[0] \leftarrow f_5$ $D_0[1] \leftarrow f_6$ $D_1[0] \leftarrow f_7$ $D_1[1] \leftarrow f_8$
32: $T_{17} \leftarrow D_0[0] + D_1[0]$ $T_{18} \leftarrow D_0[1] + D_1[1]$
33:

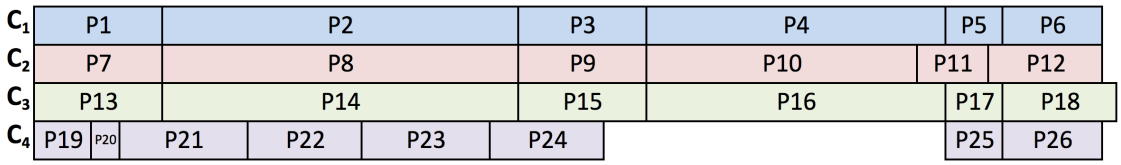
4.1.4 Çok Çekirdekli Gerçekleme İçin Gecikme Karmaşıklığı Hesabı

Curve25519 için geliştirilen cisim çarpma algoritmasının paralel olarak gerçeklenmesi için detaylı gecikme hesaplamaları yapılmıştır. Bu bağlamda, en yaygın kullanım olan 4-çekirdekli paralel hesaplama odaklı çalışmalar yapılmıştır. 4-çekirdekli hesaplama tercihi sayesinde, Raspberry Pi gibi tek kartlı bilgisayarlar üzerinde etkili hesaplama yapmak mümkün olabilecektir. Bu da, ARM mimarisi üzerinde algoritmanın daha hızlı koşmasına imkan sağlayacak bir altyapıyı oluşturmaktadır. $n \times n$ boyutundaki Toeplitz matrislerinin gecikme hesaplaması aşağıdaki eşitlik ile tanımlanmıştır:

$$\mathcal{D}(n) = 2\log_2(n)\mathcal{D}_M + \mathcal{D}_A$$

Burada, $\mathcal{D}_M$ ve $\mathcal{D}_A$ sırasıyla cisim üzerinde çarpma ve toplama işlemlerinin gecikme karmaşıklığını ifade etmektedir.

Bu çalışmada geliştirdiğimiz algoritmanın Yöntem $\#1$ gerçeklemesinin gecikme karmaşıklığı değerlendirilmiş ve 4 çekirdekli işlemcilerde gerçekleme için ideal hesaplama yolu tanımlanmıştır. Bunun için C_1, C_2, C_3 ve C_4 olmak üzere toplam 4 hesaplama yolu tanımlanmıştır. İlk 3 yol boyunca TMVÇ işlemlerinin hesabı, son yol boyunca ise diğer yan işlemlerin hesapları dahil edilmiştir. Bu bağlamda, C_1 ve C_2 yollarının gecikme karmaşıklığı $18\mathcal{D}_M + 57\mathcal{D}_A$, C_3 için $18\mathcal{D}_M + 58\mathcal{D}_A$ ve C_4 için $22\mathcal{D}_M + 29\mathcal{D}_A$ olarak hesaplanmıştır. 4-çekirdekli ideal hesaplama yolu Şekil 1'de görsel olarak verilmiştir.



Şekil 1: 4-çekirdekli ideal hesaplama yolu

Şekil 1 incelendiğinde, $P_1 - P_4$, $P_7 - P_{10}$ ve $P_{13} - P_{16}$ kısımları Toeplitz matris vektör çarpımlarının tamamlanması için geçen süreyi ifade etmektedir. Ayrıca, 4. çekirdekte de $P_{19} - P_{24}$ kısımları, hesaplama sırasındaki matris işlemlerinden bağımsız ekstra işlemlerin tamamlanması için geçen süreyi ifade etmektedir. P_5, P_{11}, P_{17} ve P_{25} kısımları da matris operasyonlarının tamamlanmasının ardından birleştirme için kullanılan toplama işlemlerinin yapılması için geçen süreyi ifade etmektedir. Son olarak P_6, P_{12}, P_{18} ve P_{26} kısımları ise 608 sabitiyle çarpma işlemini tamamlamak için geçen süreyi ifade etmektedir.

34: $T_{21} \leftarrow T_{17} + T_{18}$ $T_{14} \leftarrow g_1 T_{21}$
35: $T_{12} \leftarrow g_0 - g_1$ $T_{15} \leftarrow T_{12} T_{18}$
36: $T_{13} \leftarrow g_2 - g_1$ $T_{16} \leftarrow T_{13} T_{17}$
37: $V_1[0] \leftarrow T_{14} + T_{15}$ $V_1[1] \leftarrow T_{14} + T_{16}$
38:
39: $T_{22} \leftarrow D_1[0] + D_1[1]$ $T_{14} \leftarrow T_3 T_{22}$
40: $T_{12} \leftarrow T_5 - T_3$ $T_{15} \leftarrow T_{12} D_1[1]$
41: $T_{13} \leftarrow T_4 - T_3$ $T_{16} \leftarrow T_{13} D_1[0]$
42: $V_2[0] \leftarrow T_{14} + T_{15}$ $V_2[1] \leftarrow T_{14} + T_{16}$
43:
44: $T_{23} \leftarrow D_0[0] + D_0[1]$ $T_{14} \leftarrow T_7 T_{23}$
45: $T_{12} \leftarrow T_9 - T_7$ $T_{15} \leftarrow T_{12} D_0[1]$
46: $T_{13} \leftarrow T_8 - T_7$ $T_{16} \leftarrow T_{13} D_0[0]$
47: $V_3[0] \leftarrow T_{14} + T_{15}$ $V_3[1] \leftarrow T_{14} + T_{16}$
48:
49: $KN[0] \leftarrow V_1[0] + V_2[0]$ $KN[2] \leftarrow V_1[0] + V_3[0]$
50: $KN[1] \leftarrow V_1[1] + V_2[1]$ $KN[3] \leftarrow V_1[1] + V_3[1]$
51:
52: $T_3 \leftarrow g_3 - g_5$ $T_5 \leftarrow g_2 - g_4$ $T_4 \leftarrow g_4 - g_6$
53: $T_7 \leftarrow g_7 - g_5$ $T_9 \leftarrow g_6 - g_4$ $T_8 \leftarrow g_8 - g_6$
54:
55: $T_{14} \leftarrow g_5 T_{11}$
56: $T_{12} \leftarrow g_4 - g_5$ $T_{15} \leftarrow T_{12} T_2$
57: $T_{13} \leftarrow g_6 - g_5$ $T_{16} \leftarrow T_{13} T_1$
58: $V_1[0] \leftarrow T_{14} + T_{15}$ $V_1[1] \leftarrow T_{14} + T_{16}$
59:
60: $T_{14} \leftarrow T_3 T_{19}$
61: $T_{12} \leftarrow T_5 - T_3$ $T_{15} \leftarrow T_{12} C_1[1]$
62: $T_{13} \leftarrow T_4 - T_3$ $T_{16} \leftarrow T_{13} C_1[0]$
63: $V_2[0] \leftarrow T_{14} + T_{15}$ $V_2[1] \leftarrow T_{14} + T_{16}$
64:
65: $T_{14} \leftarrow T_7 T_{20}$
66: $T_{12} \leftarrow T_9 - T_7$ $T_{15} \leftarrow T_{12} C_0[1]$
67: $T_{13} \leftarrow T_8 - T_7$ $T_{16} \leftarrow T_{13} C_0[0]$
68: $V_3[0] \leftarrow T_{14} + T_{15}$ $V_3[1] \leftarrow T_{14} + T_{16}$
69:
70: $PL[0] \leftarrow V_1[0] + V_2[0]$ $PL[2] \leftarrow V_1[0] + V_3[0]$
71: $PL[1] \leftarrow V_1[1] + V_2[1]$ $PL[3] \leftarrow V_1[1] + V_3[1]$
72:
73: $T_{14} \leftarrow g_5 T_{21}$
74: $T_{12} \leftarrow g_4 - g_5$ $T_{15} \leftarrow T_{12} T_{18}$
75: $T_{13} \leftarrow g_6 - g_5$ $T_{16} \leftarrow T_{13} T_{17}$
76: $V_1[0] \leftarrow T_{14} + T_{15}$ $V_1[1] \leftarrow T_{14} + T_{16}$
77:

78: $T_{14} \leftarrow T_3 T_{22}$
79: $T_{12} \leftarrow T_5 - T_3$ $T_{15} \leftarrow T_{12} D_1[1]$
80: $T_{13} \leftarrow T_4 - T_3$ $T_{16} \leftarrow T_{13} D_1[0]$
81: $V_2[0] \leftarrow T_{14} + T_{15}$ $V_2[1] \leftarrow T_{14} + T_{16}$
82:
83: $T_{14} \leftarrow T_7 T_{23}$
84: $T_{12} \leftarrow T_9 - T_7$ $T_{15} \leftarrow T_{12} D_0[1]$
85: $T_{13} \leftarrow T_8 - T_7$ $T_{16} \leftarrow T_{13} D_0[0]$
86: $V_3[0] \leftarrow T_{14} + T_{15}$ $V_3[1] \leftarrow T_{14} + T_{16}$
87:
88: $PN[0] \leftarrow V_1[0] + V_2[0]$ $PN[2] \leftarrow V_1[0] + V_3[0]$
89: $PN1] \leftarrow V_1[1] + V_2[1]$ $PN[3] \leftarrow V_1[1] + V_3[1]$
90:
91: $C_0[0] \leftarrow f_1$ $C_0[1] \leftarrow f_2$ $C_1[0] \leftarrow f_3$ $C_1[1] \leftarrow f_4$
92: $T_1 \leftarrow C_0[0] + C_1[0]$ $T_2 \leftarrow C_0[1] + C_1[1]$
93: $T_3 \leftarrow g_7 - g_9$ $T_5 \leftarrow g_6 - g_8$
94:
95: $T_{11} \leftarrow T_1 + T_2$ $T_{14} \leftarrow g_9 T_{11}$
96: $T_{12} \leftarrow g_8 - g_9$ $T_{15} \leftarrow T_{12} T_2$
97: $T_{13} \leftarrow -g_9$ $T_{16} \leftarrow T_{13} T_1$
98: $V_1[0] \leftarrow T_{14} + T_{15}$ $V_1[1] \leftarrow T_{14} + T_{16}$
99:
100: $T_{19} \leftarrow C_1[0] + C_1[1]$ $T_{14} \leftarrow T_3 T_{19}$
101: $T_{12} \leftarrow T_5 - T_3$ $T_{15} \leftarrow T_{12} C_1[1]$
102: $T_{13} \leftarrow g_8 - T_3$ $T_{16} \leftarrow T_{13} C_1[0]$
103: $V_2[0] \leftarrow T_{14} + T_{15}$ $V_2[1] \leftarrow T_{14} + T_{16}$
104:
105: $T_{20} \leftarrow C_0[0] + C_0[1]$ $T_{14} \leftarrow -g_9 T_{20}$
106: $T_{12} \leftarrow g_9 - g_8$ $T_{15} \leftarrow T_{12} C_0[1]$
107: $T_{13} \leftarrow g_9$ $T_{16} \leftarrow T_{13} C_0[0]$
108: $V_3[0] \leftarrow T_{14} + T_{15}$ $V_3[1] \leftarrow T_{14} + T_{16}$
109:
110: $RL[0] \leftarrow V_1[0] + V_2[0]$ $RL[2] \leftarrow V_1[0] + V_3[0]$
111: $RL[1] \leftarrow V_1[1] + V_2[1]$ $RL[3] \leftarrow V_1[1] + V_3[1]$
112:
113: $D_0[0] \leftarrow f_6$ $D_0[1] \leftarrow f_7$ $D_1[0] \leftarrow f_8$ $D_1[1] \leftarrow f_9$
114: $T_{17} \leftarrow D_0[0] + D_1[0]$ $T_{18} \leftarrow D_0[1] + D_1[1]$
115:
116: $T_{21} \leftarrow T_{17} + T_{18}$ $T_{14} \leftarrow g_9 T_{21}$
117: $T_{12} \leftarrow g_8 - g_9$ $T_{15} \leftarrow T_{12} T_{18}$
118: $T_{13} \leftarrow g_9$ $T_{16} \leftarrow T_{13} T_{17}$
119: $V_1[0] \leftarrow T_{14} + T_{15}$ $V_1[1] \leftarrow T_{14} + T_{16}$
120:

$T_{22} \leftarrow D_1[0] + D_1[1]$ $T_{14} \leftarrow T_3 T_{22}$

$T_{12} \leftarrow T_5 - T_3$ $T_{15} \leftarrow T_{12} D_1[1]$

$T_{13} \leftarrow g_8 - T_3$ $T_{16} \leftarrow T_{13} D_1[0]$

$V_2[0] \leftarrow T_{14} + T_{15}$ $V_2[1] \leftarrow T_{14} + T_{16}$

$T_{23} \leftarrow D_0[0] + D_0[1]$ $T_{14} \leftarrow -g_9 T_{23}$

$T_{12} \leftarrow g_9 - g_7$ $T_{15} \leftarrow T_{12} D_0[1]$

$T_{13} \leftarrow g_9$ $T_{16} \leftarrow T_{13} D_0[0]$

$V_3[0] \leftarrow T_{14} + T_{15}$ $V_3[1] \leftarrow T_{14} + T_{16}$

$RN[0] \leftarrow V_1[0] + V_2[0]$ $RN[2] \leftarrow V_1[0] + V_3[0]$

$RN1] \leftarrow V_1[1] + V_2[1]$ $RN[3] \leftarrow V_1[1] + V_3[1]$

$M[1] \leftarrow KL[0]$ $M[2] \leftarrow KL[1]$ $M[3] \leftarrow KL[2]$

$M[4] \leftarrow M[4] + KL[3]$

$M[5] \leftarrow M[5] + PL[0]$

$M[6] \leftarrow M[6] + PL[1] + KN[0]$

$M[7] \leftarrow M[7] + PL[2] + KN[1]$

$M[8] \leftarrow M[8] + PL[3] + KN[2]$

$M[9] \leftarrow M[9] + KN[3]$

$N[0] \leftarrow N[0] + RL[0] + PN[0]$ $N[2] \leftarrow N[2] + RL[2] + PN[2]$

$N[1] \leftarrow N[1] + RL[1] + PN[1]$ $N[3] \leftarrow N[3] + RL[3] + PN[3]$

$N[5] \leftarrow RN[0]$ $N[6] \leftarrow RN[1]$ $N[7] \leftarrow RN[2]$ $N[8] \leftarrow RN[3]$

$C \leftarrow M[0] + (N[0] << 9) + (N[0] << 6) + (N[0] << 5)$

$h_0 \leftarrow C \bmod 2^{26}$

$C \leftarrow M[1] + (N[1] << 9) + (N[1] << 6) + (N[1] << 5)$

$h_1 \leftarrow C \bmod 2^{26}$

$C \leftarrow M[2] + (N[2] << 9) + (N[2] << 6) + (N[2] << 5)$

$h_2 \leftarrow C \bmod 2^{26}$

$C \leftarrow M[3] + (N[3] << 9) + (N[3] << 6) + (N[3] << 5)$

$h_3 \leftarrow C \bmod 2^{26}$

$C \leftarrow M[4] + (N[4] << 9) + (N[4] << 6) + (N[4] << 5)$

$h_4 \leftarrow C \bmod 2^{26}$

$C \leftarrow M[5] + (N[0] << 9) + (N[5] << 6) + (N[5] << 5)$

$h_5 \leftarrow C \bmod 2^{26}$

$C \leftarrow M[6] + (N[6] << 9) + (N[6] << 6) + (N[6] << 5)$

$h_6 \leftarrow C \bmod 2^{26}$

$C \leftarrow M[7] + (N[7] << 9) + (N[7] << 6) + (N[7] << 5)$

$h_7 \leftarrow C \bmod 2^{26}$

$C \leftarrow M[8] + (N[8] << 9) + (N[8] << 6) + (N[8] << 5)$

$h_8 \leftarrow C \bmod 2^{26}$

$C \leftarrow M[9] + (N[9] << 9) + (N[9] << 6) + (N[9] << 5)$

$h_9 \leftarrow C \bmod 2^{21}$

Sonuç H

Burada, en uzun yol C_3 olduğu için kritik yol olarak tanımlanmaktadır. Dolayısıyla, 4-çekirdekli gecikme karmaşıklığının değeri, C_3 yolunun karmaşıklık değeri $18\mathcal{D}_M + 58\mathcal{D}_A$ olarak tanımlanmaktadır. Algoritma tek çekirdek üzerinde gerçeklendiğinde ise gecikme karmaşıklığı $76\mathcal{D}_M + 201\mathcal{D}_A$ olmaktadır. Bu bağlamda, ilgili değer geliştirdiğimiz 4-çekirdekli hesaplama gecikme değeri ile karşılaştırıldığında, elde ettiğimiz yöntemin neredeyse tam paralel olarak gerçeklenebileceğini göstermiştir.

4.1.5 Yeni Güvenli Eliptik Eğri Bulma Yöntemi

Yöntem ve Gereçlerde detayları belirtildiği üzere, SafeCurves çalışması esas alınarak aşağıda adımları verilen güvenli eliptik eğri arama metodolojisi geliştirilmiştir. Bu metodoloji kullanılarak `MAGMA` üzerinde arama kodu yazılmış ve belirlenen çeşitli sonlu cisimler üzerinde eliptik eğri aramaları gerçeklenmiştir. Yapılan çalışmalar yüksek işlem gücü gerektirdiğinden proje kapsamında alınan sunucu üzerinde çalışmalar yapılmıştır. Güvenli eliptik eğri arama için geliştirilen metodoloji Montgomery formundaki eliptik eğrileri temel alacak şekilde parametre tercihleri yapılmıştır. Metodolojinin temel adımları aşağıda listelenmiştir.

1. $p \equiv 1$ mod 4 veya $p \equiv 3$ mod 4 olmak üzere büyük $(> 2^{255})$ bir asal sayı seç.
2. p asal mı test et.
3. Gerçekleme yapılacak platform kelime uzunluğunu seç. Bu metodoloji 32-bit mimariyi baz almaktadır.
4. Seçilen eliptik eğri için çarpım matrisi formunun Toepliz matris formunda olduğunu kontrol et.
5. $y^2 = x^3 + Ax^2 + x$ Montgomery formunu sağlayacak ve $A^2 - 4 \neq 0$ olacak şekilde mümkün olan en küçük A değerini seç.
6. Eliptik eğrinin mertebesinin, ℓ asal sayı olmak üzere en fazla $8 \cdot \ell$ formunda olmasını kontrol et.
7. ℓ sayısı asal mı test et.
8. Bu eliptik eğri üzerinde mertebesi ℓ olan ve x_1'in en küçük olduğu (x_1, y_1) baz noktası seç.
9. (x_1, y_1) noktasının eliptik eğri üzerinde olduğunu doğrula.
10. $\ell(x_1, y_1) = 0$ mı kontrol et.

11. $p+1-t$, $p+1+t$ ve t^2-4p değerlerinin bütün asal çarpanlarının asallığını test et.
12. Rho yöntemi karmaşıklığı değerinin 2^{100}'den büyük olduğunu kontrol et; $0.886\sqrt{\ell} > 2^{100}$.
13. $\ell \neq p$ olduğunu kontrol et. Bunun olması toplamsal transfere karşı güvenlik sağlar.
14. Çarpımsal transfer değerini hesapla ve $(\ell-1)/100$'den büyük olduğunu kontrol et. Bu çarpımsal transfere karşı güvenlik sağlar.
15. Eliptik eğrinin iz (Trace) değeri t'yi hesapla.
16. Kompleks çarpım cisim diskriminantı D'yi hesapla. Eğer $(t^2-4p)/s^2$ mod $4=1$ ise $D=(t^2-4p)/s^2$, diğer durumlarda $D=4(t^2-4p)/s^2$ olarak tanımla.
17. D'nin 2^{100}'den büyük olduğunu kontrol et.
18. Sağlamlık kriteri için tanımları yap.
19. Montgomery merdiveni desteğini kontrol et.
20. Twist ataklarına karşı güvenliğin 2^{100}'den büyük olduğunu kontrol et.
21. Twist'in Rho yöntemi karmaşıklığı değerinin 2^{100}'den büyük olduğunu kontrol et.
22. Twist eğrisinin toplamsal transfere karşı güvenli olduğunu kontrol et.
23. Twist eğrisinin çarpımsal transfere karşı güvenli olduğunu kontrol et.
24. Eliptik eğiri için tek-skaler bütünsel formüllerin varlığını kontrol et.
25. Eliptik eğiri için çok-skaler bütünsel formüllerin varlığını kontrol et.
26. Mertebesi 2 olan noktaların sayısının 1 olduğunu doğrula.
27. Mertebesi 4 olan noktaların sayısının 2 olduğunu doğrula.
28. Eliptik eğrinin ayırt edilememezlik kriterlerini sağladığını doğrula.

4.1.6 Yeni Güvenlik Eliptik Eğri Bulunması

Bölüm 4.1.5'de verilen metodoloji kullanılarak $\mathbb{F}_{2^{266}-3}$ sonlu cismi üzerinde güvenli eliptik eğri araması için `MAGMA` yazılımı ile kod geliştirilmiş ve aşağıda detayları verilen eliptik eğri elde edilmiştir.

Bu durumda, f, g ve h, $\mathbb{F}_{2^{266}-3}$'in elemanları olmak üzere, aşağıdaki gibi limb dizileri halinde tanımlanmıştır.

$$
\begin{aligned}
f =& [f_0/27, f_1/27, f_2/27, f_3/27, f_4/27, f_5/27, f_6/27, f_7/27, f_8/27, f_9/23] \\
g =& [g_0/27, g_1/27, g_2/27, g_3/27, g_4/27, g_5/27, g_6/27, g_7/27, g_8/27, g_9/23] \\
h =& [h_0/27, h_1/27, h_2/27, h_3/27, h_4/27, h_5/27, h_6/27, h_7/27, h_8/27, h_9/23]
\end{aligned}
$$

Bu elemanları, katsayıları bir önceki limb'lerin katsayılarının üzerine eklenerek ifade edilecek şekilde polinom olarak ifade edebiliriz:

$$
\begin{aligned}
f(x) =& f_0 + 2^{27} f_1 x + 2^{54} f_2 x^2 + 2^{81} f_3 x^3 + 2^{108} f_4 x^4 + 2^{135} f_5 x^5 + \\
& 2^{162} f_6 x^6 + 2^{189} f_7 x^7 + 2^{216} f_8 x^8 + 2^{243} f_9 x^9 \\
g(x) =& g_0 + 2^{27} g_1 x + 2^{54} g_2 x^2 + 2^{81} g_3 x^3 + 2^{108} g_4 x^4 + 2^{135} g_5 x^5 + \\
& 2^{162} g_6 x^6 + 2^{189} g_7 x^7 + 2^{216} g_8 x^8 + 2^{243} g_9 x^9 \\
h(x) =& h_0 + 2^{27} h_1 x + 2^{54} h_2 x^2 + 2^{81} h_3 x^3 + 2^{108} h_4 x^4 + 2^{135} h_5 x^5 + \\
& 2^{162} h_6 x^6 + 2^{189} h_7 x^7 + 2^{216} h_8 x^8 + 2^{243} h_9 x^9
\end{aligned}
$$

Elde ettiğimiz yeni polinom gösterimlerini kullandığımızda $\mathbb{F}_{2^{266}-3}$ üzerinde cisim çarpması için aşağıdaki matris gösterimini elde ederiz.

$$
\begin{bmatrix}
g_0 & 48g_9 & 48g_8 & 48g_7 & 48g_6 & 48g_5 & 48g_4 & 48g_3 & 48g_2 & 48g_1 \\
g_1 & g_0 & 48g_9 & 48g_8 & 48g_7 & 48g_6 & 48g_5 & 48g_4 & 48g_3 & 48g_2 \\
g_2 & g_1 & g_0 & 48g_9 & 48g_8 & 48g_7 & 48g_6 & 48g_5 & 48g_4 & 48g_3 \\
g_3 & g_2 & g_1 & g_0 & 48g_9 & 48g_8 & 48g_7 & 48g_6 & 48g_5 & 48g_4 \\
g_4 & g_3 & g_2 & g_1 & g_0 & 48g_9 & 48g_8 & 48g_7 & 48g_6 & 48g_5 \\
g_5 & g_4 & g_3 & g_2 & g_1 & g_0 & 48g_9 & 48g_8 & 48g_7 & 48g_6 \\
g_6 & g_5 & g_4 & g_3 & g_2 & g_1 & g_0 & 48g_9 & 48g_8 & 48g_7 \\
g_7 & g_6 & g_5 & g_4 & g_3 & g_2 & g_1 & g_0 & 48g_9 & 48g_8 \\
g_8 & g_7 & g_6 & g_5 & g_4 & g_3 & g_2 & g_1 & g_0 & 48g_9 \\
g_9 & g_8 & g_7 & g_6 & g_5 & g_4 & g_3 & g_2 & g_1 & g_0
\end{bmatrix}
\cdot
\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{bmatrix}
=
\begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix}
$$

Matris'in Toeplitz matrisi formunda olması geliştirdiğimiz TMVÇ yönteminin bu sonlu cisim üzerinde de uygulanabileceğini ifade etmektedir.

Yeni bulunan eliptik eğri denklemi:

$$y^2 = x^3 + 20710x^2 + x$$

Eliptik eğrinin mertebesi $= 2^3 \cdot \ell$, öyleki

$$\ell = 148213874223764730142170860811120522052167776443808039434531805601905579644782 97$$

ℓ sayısının asal sayı olduğu asallık testleri ile gösterilmiştir. Bu sayı, bulunan baz noktasının mertebesine de eşittir. Baz noktası:

$$(x_1, y_1) = \\ (17, 943507226411813093766456758778209612349658134889167308113829190294892578237630 54) \tag{2}$$

Benzer şekilde eliptik eğrinin Twist'i de hesaplanmış ve ilgili kontroller yapılmıştır. Twist eliptik eğrinin denklemi aşağıda verilmiştir:

$y^2 = x^3 + A'x^2 + A''x$, öyleki

$$A' = 640145143671267628184865907583931539647826971195548704940649033656580022918 91982$$
$$A'' = 389712187346585607632483748517739542770062512026597353942154623758345773416 21209 \tag{3}$$

Twist eliptik eğri'nin Mertebesi $= 2^2 \cdot \ell'$, öyleki

$$\ell' = 296427748447529460284341721622241044104406768600463609012959218916704064466906 37$$

ℓ' sayısının asal sayı olduğu asallık testleri ile gösterilmiştir.

4.2 Polinom Çarpması

Karatsuba ve benzeri algoritmaların özyinelemeli kullanımında her adımda olabilecek en iyi algoritmanın kullanılması yaklaşımı büyük tamsayıların çarpımlarında iyileştirmeler getirmiştir. Örneğin $n = 6$ için, eğer okul kitabı yöntemini kullanılırsa, 36 çarpma ve 25 ikili kelime toplamı karmaşıklığı ile sonuç bulunur. Bu karmaşıklık $M(6) = 36M + 25A_D$ gibi ifade edilir. Bunun yerine önce Karatsuba 2-yol daha sonra okul kitabı metotu uygulandığında sonucun

$$\begin{aligned} M(6) &= 3M(3) + (\tfrac{5\cdot 6}{2} - 3)A_D + (6)A_S \\ &= 3(9M + 4A_D) + 12A_D + 6A_S \\ &= 27M + 24A_D + 6A_S, \end{aligned}$$

olduğu gösterilmiştir. Ayrıca dengeli olmayan terim sayılarında (mesela terim sayısı tek olan polinomlar için 2-yol kullanmak gibi), dengesiz (unbalanced) bölme işlemi yapılarak yeni sonuçlar elde edilmiştir. Bu tur yaklaşımlar için URK2 (unbalanced refined Karasuba 2-way) gibi kıslatmalar kullanılmıştır. Tablo 7'de bir kelimelik çarpmanın, bir kelimelik toplamının ve iki kelimelik toplamının maliyetlerinin aynı olduğu platformlar için elde edilen en iyi sonuçlar listelenmiştir.

Diğer taraftan $2M = 2A_S = A_D$ olan platformlarda en iyi olabilecek durumların araştırılmasından sonra elde edilen sonuçlar Tablo 8'de verilmiştir. Ek olarak, NIST asal sayıları için de sonuçlar elde edilmiştir. Bu sonuçlar Table 9'da sunulmuştur.

n	Mult.	A_S	A_D	# Operation	Algorithm
1	1			1	SB
2	4		1	5	SB
3	9		4	13	SB
4	16		9	25	SB
5	25		16	41	SB
6	27	6	24	57	RK2, M(3)
7	40	6	35	81	M(6), last term
8	48	8	44	100	RK2, M(4)
9	65	8	59	132	M(8), last term
10	75	10	70	155	RK2, M(5)
11	78	22	73	189	URK2, M(6),M(5)
12	81	30	99	210	RK2,M(6)
13	106	30	122	258	M(12), last term
14	120	32	137	289	RK2, M(7)
15	135	36	158	329	URK2, M(8),M(7)
16	144	40	169	353	RK2, M(8)
17	177	40	200	417	M(16), last term
18	195	42	219	456	RK2, M(9)
19	214	46	244	504	URK2, M(10), M(11)
20	225	50	257	532	RK2, M(10)

Tablo 7: $M = A_S = A_D$ olan platformlarda en iyi sonuçlar

İzojen bazlı kuantum sonrası sistemler için de interpolasyon tekniği ile $\mathbb{F}_{p^2}$'de çarpma yapmak için yeni özgün bir algoritma tasarlanmıştır. Polinomla $\mathbb{F}_{p^2}$ üzerinde $a(x) = a_0 + a_1x + a_2x^2$ ve $b(x) = b_0 + b_1x + b_2x^2$ olsun. Ayrıca çarpım ise$c(x) = c_0 + c_1x + c_2x^2 + c_3x^3 + c_4x^4$ ve i, $x^2 + 1$ polinomunun kökü olsun. Çarpım polinomu $a(x)b(x) = c(x)$, $\{0, 1, i, -i, \infty\}$ noktalarında hesap-

n	Mult.	A_S	A_D	Tot. Operation	Algorithm
1	1			1	SB
2	4		1	6	SB
3	9		4	17	SB
4	16		9	34	SB
5	25		16	57	SB
6	27	6	24	81	RK2, M(3)
7	40	6	35	116	M(6), last term
8	48	8	44	144	RK2, M(4)
9	65	8	59	191	M(8), last term
10	75	10	70	225	RK2, M(5)
11	78	22	89	278	URK2, M(6),M(5)
12	81	30	99	309	RK2, M(6)
13	106	30	122	380	M(12), last term
14	120	32	137	426	RK2, M(7)
15	135	36	158	487	URK2, M(8),M(7)
16	144	40	169	522	RK2, M(8)
17	177	40	200	617	M(16), last term
18	195	42	219	655	RK2, M(9)
19	214	46	244	748	URK2, M(10), M(11)
20	225	50	257	789	RK2, M(10)

Tablo 8: $2M = 2A_S = A_D$ olan platformlarda en iyi sonuçlar

	p_{192}	p_{224}	p_{256}	p_{384}	p_{521}
8-bit	25	28	32	48	66
16-bit	12	14	16	24	33
32-bit	6	7	8	12	17
64-bit	3	4	4	6	9

Tablo 9: Nist asal sayıları ve polinom büyüklükleri

landığında, aşağıdaki sistem elde edilir.

$$\begin{bmatrix} a_0b_0 \\ (a_0+a_1+a_2)(b_0+b_1+b_2) \\ (a_0-a_2+a_1i)(b_0-b_2+b_1i) \\ (a_0-a_2-a_1i)(b_0-b_2-b_1i) \\ a_2b_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i & 1 \\ 1 & -i & -1 & i & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}.$$

Kolaylık olsun diye aaşağıdaki notasyon önerilmiştir.

$$
\begin{aligned}
p_1 &= a_0 b_0, \\
p_2 &= (a_0 + a_1 + a_2)(b_0 + b_1 + b_2), \\
p_3 &= (a_0 - a_2 + a_1 i)(b_0 - b_2 + b_1 i), \\
p_4 &= (a_0 - a_2 - a_1 i)(b_0 - b_2 - b_1 i), \\
p_5 &= a_2 b_2.
\end{aligned}
$$

Daha sonra katsayıları bulmak için önce sağdaki matrisin tersi sonra ise bulunan ters ile soldaki vektör çarpılarak katsayılar hesaplanır.

$$
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 \\
1 & i & -1 & -i & 1 \\
1 & -i & -1 & i & 1 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}^{-1}
=
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
-1 & 1/2 & (1-i)/4 & (1+i)/4 & -1 \\
1 & 0 & -1/2 & -1/2 & 1 \\
-1 & 1/2 & (1+i)/4 & (1-i)/4 & -1 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}.
$$

$$
\begin{aligned}
c_0 &= p_1, \\
c_1 &= -p_1 + \tfrac{p_2}{2} + \tfrac{(1-i)}{4} p_3 + \tfrac{(1+i)}{4} p_4 - p_5, \\
c_2 &= p_1 - \tfrac{p_3}{2} - \tfrac{p_4}{2} + p_5, \\
c_3 &= -p_1 + \tfrac{p_2}{2} + \tfrac{(1+i)}{4} p_3 + \tfrac{(1-i)}{4} p_4 - p_5, \\
c_4 &= p_5.
\end{aligned}
$$

Bu algoritmanın işlemsel karmaşıklığı Tablo 10'da verilmiştir.

Bu ve diğer bilinen algoritmalar kullanılarak Tablo 11'deki sonuçlar elde edilmiştir. Bu tablodaki kısatlmalardan SB, okul kitabı algoritmasını; RKA, rafine Karatsuba algoritmasını; Int., interpolasyon algoritmasını ve Gauss da Gauus'un 2-yol algoritmasını göstermektedir. Yapılan diğer bir çalışma arama algoritmalarının kullanılmasıyla çarpma algoritması geliştirmektir. Arama algoritmaları kullanarak polinom çarpma algoritmaları elde etmek için arama uzayı düşürülmeye çalışılmıştır. Bunun için (Barbulescu vd. 2012) makalesindeki arama uzayını düşürme tekniğini kullanarak n-terimli iki polinomun ilk ℓ teriminin aramasına bakılmıştır. Ayrıca Motgomery 'nin (Montgomery 2005) makalesinde yaptığı gibi simetrik ikili formların kullanılması ve bir ikili formun homojenizasyonundan elde edilen diğer ikili formun üretme kümesine eklenmesi arama uzayını düşürmüştür. Ek olarak n terimli iki polinomum çarpımının sonucunu hesaplamak ($M(n)$) için önce sonucun ilk ℓ teriminin hesaplanması ($\widehat{M}(n, \ell)$), sonra buradan (Cenk ve Özbudak 2011)

$Operation$	$Cost$	$Operation$	$Cost$
1. $s_1 = A_0 + A_1$	$\frac{n}{3}\widetilde{A}$	13. $P_3 = s_4 s_9$	$\widetilde{M}(\frac{n}{3})$
2. $s_2 = s_1 + A_2$	$\frac{n}{3}\widetilde{A}$	14. $P_4 = s_5 s_{10}$	$\widetilde{M}(\frac{n}{3})$
3. $s_3 = A_0 - A_2$	$\frac{n}{3}\widetilde{A}$	15. $P_5 = A_2 B_2$	$\widetilde{M}(\frac{n}{3})$
4. $s_4 = s_3 + A_1 i$	$\frac{n}{3}\widetilde{A}$	16. $s_{11} = P_1 + P_5$	$(\frac{2n}{3}-1)\widetilde{A}_D$
5. $s_5 = s_3 - A_1 i$	$\frac{n}{3}\widetilde{A}$	17. $s_{12} = P_3 + P_4$	$(\frac{2n}{3}-1)\widetilde{A}_D$
6. $s_6 = B_0 + B_1$	$\frac{n}{3}\widetilde{A_D}$	18. $s_{13} = P_3 - P_4$	$(\frac{2n}{3}-1)\widetilde{A}_D$
7. $s_7 = s_6 + B_2$	$\frac{n}{3}\widetilde{A}$	19. $s_{14} = -s_{11} + P_2/2$	$(\frac{2n}{3}-1)\widetilde{A}_D$
8. $s_8 = B_0 - B_2$	$\frac{n}{3}\widetilde{A}$	20. $s_{15} = s_{14} + s_{12}/4$	$(\frac{2n}{3}-1)\widetilde{A}_D$
9. $s_9 = s_8 + B_1 i$	$\frac{n}{3}\widetilde{A}$	21. $s_{16} = s_{15} - i s_{13}/4$	$(\frac{2n}{3}-1)\widetilde{A}_D$
10. $s_{10} = s_8 - B_1 i$	$\frac{n}{3}\widetilde{A}$	22. $s_{17} = s_{15} + i s_{13}/4$	$(\frac{2n}{3}-1)\widetilde{A}_D$
11. $P_1 = A_0 B_0$	$\widetilde{M}(\frac{n}{3})$	23. $s_{18} = s_{11} - s_{12}/2$	$(\frac{2n}{3}-1)\widetilde{A}_D$
12. $P_2 = s_2 s_7$	$\widetilde{M}(\frac{n}{3})$		

Tablo 10: 3-yol algoritmasının işlemsel karmaşıklığı

n	1	2	3	4	5	6	7	8	9	10	11	12
$\widetilde{M}(n)$	8	36	80	142	242	300	468	510	652	781	969	1069
Alg.	SB	Gauss	Gauss	Gauss	SB	RKA	SB	RKA	Int.	Gauss	Gauss	RKA

Tablo 11: $\mathbb{F}_{p^2}$ üzerinde n terimli polinomlar çarpma karmaşıklığı

makalesindeki yaklaşımla n terimli iki polinomun ilk n teriminin elde edilmesi ($\widehat{M}(n)$) ve nihayetinde bu sonucu kullanarak Çinli kalan teoremi ile iki polinomun tüm terimlerinin hesaplanması yönteminin iyileştirilmesinin geliştirilmiş algoritmalar verebileceği sonucuna varılmıştır. Bu sonuçların bir derlemesi (Cenk 2018)'de basılmıştır. Diğer taraftan literatürdeki sonuçları iyileştirme adına C programlama dili ile bahsedilen teknikler kullanılarak arama algoritması gerçekleştirilmiştir. Standart bir masaüstü bilgisayarda paralel gerçekleştirme yapılmadan yapmış olduğumuz gerçekleştirmeler sonucunda küçük n'ler için literatürdeki $\widehat{M}(n,\ell)$ sonuçları elde edilmiştir. Şu anda bu yazılımın optimizasyonları ve bunların çok daha güçlü işlemciler üzerinde ve paralel gerçekleştirmeler ile bir çok işlemci kullanarak gerçekleştirilmeleri üzerine çalışmalar devam etmektedir.

4.3 Modüler Üs Alma

Hem sabit tabanlara (modüler üs almada tekrar kullanabilmek için) hem de değişken üslere göre (alınacak üssün gösterim şekillerinin önceden hesaplanabilmesi için) kullanılabilecek algoritmalar taranmıştır. Bu algoritmalardan bazılarının (klasik yöntem, karma ikili-üçlü üs alma, tekrarlı kare

alma, tekrarlı küp alma ve NAF) C++ üzerinde gmp kütüphanesi kullanılarak gerçekleştirmeleri yapılmıştır. Bu algoritmalar hem işlem sayısı hem de işlem süresi açısından, farklı parametreler için karşılaştırılmıştır. 2. Rapordaki karşılaştırmalara göre, değişken üs ve taban için karma ikili-üçlü üs alma metodunun diğer yöntemlere kıyasla daha iyi olduğu görülmüştür. Sabit taban ve üsler için ise ikili-taban (double base) yönteminin optimize sonuçlar verdiği, fakat üssün gösterimini bulmanın yavaş olduğu ortaya çıkmıştır. Doğal sayılarda küp almada Zanoni'nin "sugar-cube" yöntemi belirli sayı aralıklarında klasik yöntemden daha iyi olmasına rağmen, modüler olarak şimdiki algoritmalar kullanıldığında daha yavaş olduğu gözlemlenmiştir. Bu sebeple bir indirgeme algoritması olan Barrett indirgemesi üzerine çalışılmaya başlanmıştır. Barrett indirgemesi'nin orijinal hali sadece $2n$-kelime büyüklüğündeki sayıların n-kelime büyüklüğündeki sayılara indirgenmesini konu alır. Genelleştirilmiş Barrett İndirgemesi (Generalized Barrett Reduction) ise herhangi 2 kelime büyüklüğündeki sayıların indirgemesi ile ilgilenir. Genelleştirilmiş Barrett İndirgemesi, üzerinde çok çalışılmamış bir konu olup, $R(3n, n)$ indirgemesi hakkında bir makale ya da uygulama, bizim bilgimiz dahilinde görülmemiştir.

$R(3n, n)$ indirgemesinde kullanılan algoritma şu şekildedir:

Algoritma 2 Barrett-3k Reduction

Girdi: b tabanında yazılmış $3k$ boyutunda x ve k boyutunda m, $\mu = \left\lfloor \frac{b^{3k}}{m} \right\rfloor$

Çıktı: $r \equiv x \pmod{m}$

1: $\hat{q} = \left\lfloor \frac{\left\lfloor \frac{x}{b^{k-1}} \right\rfloor \mu}{b^{2k+1}} \right\rfloor$

2: $r = x \pmod{b^{k+1}} - \hat{q}m \pmod{b^{k+1}}$

3: $r < 0$ **ise**

4: $\quad r = r + b^{k+1}$

5:

6: $r \geq m$ **sağlandığı sürece**

7: $\quad r = r - m$

8:

9: **Sonuç** r

Algoritmanın hız analizini yapmak için teker teker adımları incelenmiştir:

$$\begin{aligned} x &= x_0 + x_1 b + \cdots + x_{2k-1} b^{3k-1}, \\ m &= m_0 + m_1 b + \cdots + m_{k-1} b^{k-1}, \\ \mu &= \left\lfloor \frac{b^{3k}}{m} \right\rfloor = \mu_0 + \mu_1 b + \cdots + \mu_{2k} b^{2k}. \end{aligned}$$

İndirgeme işlemi art arda aynı mod ile pek çok kez tekrarlanacağı düşünülürse, μ bütün bu işlemler içinde aynı kalacaktır. Dolayısıyla μ'nün hesaplanmasının hızı analize katılmaz. İlk adımda şu işlemle karşılaşılır:

$$x' = \left\lfloor \frac{x}{b^{k-1}} \right\rfloor = x = x_{k-1} + x_k b + \cdots + x_{3k-1} b^{2k}.$$

Bu işlem sadece kaydırma işlemi olduğu için çok hızlı yapılabilir, geri kalan işlemlerin yanında hızı göz ardı edilebilir olacaktır. Sonraki adımda algoritma w'yu bulur:

$$\begin{aligned} w = x'\mu &= (x_{k-1} + x_k b + \cdots + x_{3k-1} b^{2k})(\mu_0 + \mu_1 b + \cdots + \mu_{2k} b^{2k}) \\ &= (x_{k-1}\mu_0) + (x_{k-1}\mu_1 + x_k \mu_0) b + \cdots + (x_{3k-1}\mu_{2k}) b^{4k} \\ &= w_0 + w_1 b + \cdots + w_{4k} b^{4k}. \end{aligned}$$

$x'\mu$'yü b^{2k+1} ile bölündüğü için, $\hat{q}$'nun tamamının bulunmasına gerek yoktur. Sadece büyük $2k$-kelimelik bölümüne ihtiyaç vardır.

$$\hat{q} = w_{2k+1} + w_{2k+2} b + \cdots + w_{2k} b^{2k-1}.$$

İkinci adımla devam edilirse, x (mod b^{k-1}) bulmak sadece kaydırma olduğu için işlem sayısı göz önünde bulundurulmaz. Ancak adımın ikinci parçasına bakılırsa:

$$\begin{aligned} \hat{q}_2 &= (w_{2k+1} + w_{2k+2} b + \cdots + w_{2k} b^{2k-1})(m_0 + m_1 b + \cdots + m_{k-1} b^{k-1}) \\ &= w_{2k+1} m_0 + (w_{2k+1} m_1 + w_{2k+2} m_0) b + \cdots + w_{2k} m_{k-1} b^{3k-2} \\ &= t_0 + t_1 b + \cdots + t_{3k-2} b^{3k-2}. \end{aligned}$$

Sonrasında $\hat{q}_2$'nun b^{k+1} ile modu alınırsa:

$$\begin{aligned}\hat{q}_3 = \hat{q}_2 \pmod{b^{k+1}} &= t_0 + t_1 + \cdots + t_k b^k \\ &= (w_{2k+1}m_0) + (w_{2k+1}m_1 + w_{2k+2}m_0)b \\ &\quad + \cdots + (w_{2k+2}m_{k-1} + \cdots + w_{3k+1}m_0)b^k\end{aligned} \tag{4}$$

$\hat{q}$, sadece 2. adımın 2. parçası için hesaplandığı için, sadece gerekli kısımlarının hesaplanması yeterli olacaktır. 4. denklemine bakılırsa, hesaplanması gereken katsayıların $\{w_{2k+1}, \cdots, w_{3k+1}\}$ olduğu görülür.

Bütün hesaplamalar göz önünde bulundurulursa, $R(6n, 2n)$ indirgemesinin işlem karmaşıklığının 2 tane $R(4n, 2n)$ indirgemesine göre daha az olduğu teorik olarak gözlemlenmiştir. Pratikte, yine C dili kullanılarak ve GMP kütüphanesine olabildiğince yakın yazılmaya çalışılarak, gerçekleme hızının ölçülmesine devam edilmektedir.

Eğer elde edilen hız 1'ta verilen denkleme uygun oluyorsa küp şeker algoritmasının modüler versiyonunun hızı, modüler klasik küp yöntemini geçecektir. Klasik algoritmada ise, arada indirgeme yapmayıp, $R(6n, 2n)$'i son adımda kullanmak ise bir hız vermeyecektir. Bunun sebebi de iki işlemde de son adımda aynı indirgeme yapıldığı için, öncesindeki işlemler tam olarak modüler olmayan haldekiyle aynı olmasıdır. O durumda küp şeker algoritmasının daha hızlı olduğu aralıklarda, modüler olarak da daha hızlı olması beklenmektedir.

5 SONUÇ

Bu projede eliptik eğri anahtar değişimi protokolü ile polinom çarpma ve modüler üst alma işlemlerinin performanslarını iyileştirmeye yönelik araştırma yapılmıştır. Proje kapsamında üç dergi makalesi, üç konferans bildirisi, iki poster, bir doktora tezi ve iki yüksek lisans tezi üretilmiştir. Ayrıca hali hazırda üç adet makale çalışması yazım aşamasındadır. Eliptik eğriler üzerine yapılan çalışmalar Curve25519, P-521 ve E-521 eğrileri üzerine olmuştur. Bu eğrilerin üzerinde tanımlandığı cisimlerde çarpma yapmak için Toeplitz matris vektör çarpımı tabanlı işlemlerin karmaşıklığı iyileştirilmiş, yeni algoritmalar geliştirilmiştir. Bu algoritmalar, bahsedilen eliptik eğrilerde kullanılarak 32-bit ve 64-bit mimariler üzerinde gerçekleştirmeler yapılmıştır ve literatürdeki bilinen en iyi gerçekleştirilmelere göre daha iyi sonuçlar elde edilmiştir. Ek olarak, geliştirilen bu yaklaşımın kullanılmasına olanak verip, algoritmaların yüksek hızda çalışmasını sağlayan yeni güvenli eliptik

eğriler elde edilmiştir. Çalışılan bir diğer konu ise polinom çarpmalarının hızlandırılmasıdır. Bunun için arama algoritmaları ve interpolasyon teknikleri kullanılmıştır. Arama algoritmalarında polinom çarpımının her terimini hesaplamak yerine belirli terimlerinin hesaplanması yaklaşımı arama uzayını düşürmüştür. Ayrıca, interpolasyon yaklaşımı ve özyinelemeli yöntemin karma (hybrid) yaklaşımlarla kullanılması yeni verimli çarpma algoritmaları ortaya çıkarmıştır. Proje kapsamında modüler üst alma üzerine de çalışmalar yapılmıştır. Asimetrik anahtarlı kripto sistemlerin bir kısmında kullanılan üs alma işlemi üzerine yapılan çalışmada öncelikle Zanoni'nin "sugar-cube" algoritmasını inceleme amacıyla pek çok çarpma yöntemi (Toom-Karatsuba vb.) çalışılmıştır. Sonrasında "sugar-cube" yöntemi geliştirilip modüler hale getirilmiştir. Modüler durumda indirgemenin yapılması gereken optimal yer bulunmuştur. Fakat gözlemlere göre, modüler küp alma yöntemi, klasik modüler küp alma yöntemine kıyasla yavaş kalmaktadır. Bu sebeple önce üssün farklı yazma çeşitleri incelenmiştir. Çift üsle yazma ve karma metot bunların arasında en iyi sonuçları vermesine rağmen, klasik modüler küp alma çoğu durumda daha hızlı olduğu için, indirgeme yönteminin değiştirilmesi üzerinde durulmuştur. Bu amaçla Barrett indirgemesi araştırılmış ve özellikle aralarında 3 kat fark olan sayı ve mod arasındaki indirgeme için iyileştirmeler elde edilmiştir.

Proje kapsamında elde ettiğimiz bilgi birikimi sayesinde yapılabilecek öneriler şunlardır: Asal cisimler üzerindeki eliptik eğriler için Toeplitz matris vektör çarpımları (TMVÇ) yaklaşımlarının kullanılması performası hızlandırmasından dolayı önerilmektedir. Ayrıca proje kapsamında TMVÇ'yi destekleyen yeni güvenli eliptik eğriler literatürdekilere iyi bir alternatifdir. Diğer taraftan, polinom çarpım algoritmaları tasarımında kullanılan arama algoritmalarının arama uzayını düşürmek için çarpımın belli elemanlarının bulunması ve bulunan sonuçların Çinli Kalan Teoremi ile kullanılması verimli sonuçlar verecektir. Ek olarak, interpolasyon tabanlı çarpma algoritmalarının ve Karatsuba benzeri algoritmların özyinelemeli yaklaşım içerisinde karma kullanılması yüksek performanslı çarpma algoritmaları sağlamaktadır. Son olarak, modüler üs alırken küp şeker algoritması ile değiştirilmiş Barrett algoritmasının kullanılması yüksek performans gereken uygulamalarda tavsiye edilir.

Referanslar

Adikari, J., Dimitrov, V. S. ve Imbert, L. (2011). "Hybrid binary-ternary number system for elliptic curve cryptosystems". *IEEE transactions on computers* 60.2, pp. 254–265.

Ali, S. (2017). “Faster Residue Multiplication Modulo 521-bit Mersenne Prime and Application to ECC”. PhD thesis. Orta Doğu Teknik Üniversitesi Uygulamalı Matematik Enstitüsü Kriptografi Programı.

Ali, S. ve Cenk, M. (2017). “A New Algorithm for Residue Multiplication Modulo $2^{521} - 1$”. *Proceedings of the 19th International Conference on Information Security and Cryptology Volume 10157*. New York, NY, USA: Springer-Verlag New York, Inc., pp. 181–193.

Ali, S. ve Cenk, M. (2018a). “Faster Residue Multiplication Modula 521-bit Mersenne Prime and an Application to ECC”. Poster.

Ali, S. ve Cenk, M. (2018b). “Faster Residue Multiplication Modulo 521-bit Mersenne Prime and an Application to ECC”. *IEEE Transactions on Circuits and Systems I: Regular Papers* 65.8, pp. 2477–2490.

Aranha, D. F., Barreto, P. S. L. M., Pereira, G. C. C. F. ve Ricardini, J. E. (2013). *A note on high-security general-purpose elliptic curves*. Cryptology ePrint Archive, Report 2013/647. https://eprint.iacr.org/2013/647.

Baloglu, S., Cenk, M. ve Calik, C. (2018). “On the New Upper Bounds for Computing Some First Terms of Product Polynomials”.

Barbulescu, R., Detrey, J., Estibals, N. ve Zimmermann, P. (2012). “Finding optimal formulae for bilinear maps”. *International Workshop on the Arithmetic of Finite Fields*. Springer, pp. 168–186.

Barrett, P. (1986). “Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor”. *Conference on the Theory and Application of Cryptographic Techniques*. Springer, pp. 311–323.

Bernstein, D. J. (2006). “Curve25519: New Diffie-Hellman Speed Records”. *Public Key Cryptography - PKC 2006: 9th International Conference on Theory and Practice in Public-Key Cryptography, New York, NY, USA, April 24-26, 2006. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 207–228. ISBN: 978-3-540-33852-9. DOI: 10.1007/11745853_14.

Bernstein, D. J., Chuengsatiansup, C. ve Lange, T. (2014). “Curve41417: Karatsuba Revisited”. *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*. https://doi.org/10.1007/978-3-662-44709-3_18, pp. 316–334.

Bernstein, D. J., Hamburg, M., Krasnova, A. ve Lange, T. (2013). *Elligator: Elliptic-curve points indistinguishable from uniform random strings*. Cryptology ePrint Archive, Report 2013/325. https://eprint.iacr.org/2013/325.

Bernstein, D. J. ve Lange, T. (2007). *Faster addition and doubling on elliptic curves*. Cryptology ePrint Archive, Report 2007/286. https://eprint.iacr.org/2007/286.

Bernstein, D. J. ve Lange, T. (2018). *SafeCurves: choosing safe curves for elliptic-curve cryptography*. `https://safecurves.cr.yp.to`. URL: `https://safecurves.cr.yp.to` (visited on 07/01/2018).

Bernstein, D. J., Chuengsatiansup, C. ve Lange, T. (2017). "Double-base scalar multiplication revisited." *IACR Cryptology ePrint Archive* 2017, p. 37.

Bodrato, M. ve Zanoni, A. (2012). "A new algorithm for long integer cube computation with some insight into higher powers". *International Workshop on Computer Algebra in Scientific Computing*. Springer, pp. 34–46.

Bos, J. W., Costello, C., Longa, P. ve Naehrig, M. (2016). "Selecting elliptic curves for cryptography: an efficiency and security analysis". *J. Cryptographic Engineering* 6.4, pp. 259–286.

Bosselaers, A., Govaerts, R. ve Vandewalle, J. (1993). "Comparison of three modular reduction functions". *Annual International Cryptology Conference*. Springer, pp. 175–186.

Brown, D. R. L. (2011). *Certicom Research SEC 2: Recommended Elliptic Curve Domain Parameters*. `http://www.secg.org/sec2-v2.pdf`.

Cenk, M. ve Özbudak, F. (2008). "Efficient Multiplication in $\mathbb{F}_{3^{\ell m}}, m \geq 1$ and $5 \leq \ell \leq 18$". *AFRICA CRYPT*, pp. 406–414.

Cenk, M. ve Özbudak, F. (2009). "Improved Polynomial Multiplication Formulas over F_2 Using Chinese Remainder Theorem". *IEEE Transactions on computers* 58.4, pp. 572–576.

Cenk, M. ve Özbudak, F. (2011). "Multiplication of polynomials modulo x^n". *Theoretical Computer Science* 412.29, pp. 3451–3462.

Cenk, M. ve Özbudak, F. (2010). "On multiplication in finite fields". *Journal of Complexity* 26.2, pp. 172–186.

Cenk, M. (2018). "Karatsuba-Like Formulae and Their Associated Techniques". *Journal of Cryptographic Engineering*.

Cenk, M., Banerjee, T. ve Hasan, A. (2018). "Faster Multiplication in the Quadratic Extension of Prime Fields and Its Applications to SIDH". In progress.

Chou, T. (2015). "Sandy2x: New Curve25519 Speed Records". *Selected Areas in Cryptography - SAC 2015 - 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers*, pp. 145–160.

Diffie, W. ve Hellman, M. (1976). "New directions in cryptography". *IEEE transactions on Information Theory* 22.6, pp. 644–654.

Dimitrov, V. ve Cooklev, T. (1995). "Two algorithms for modular exponentiation using nonstandard arithmetics". *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 78.1, pp. 82–87.

Fan, H. ve Hasan, M. A. (2007a). “A New Approach to Subquadratic Space Complexity Parallel Multipliers for Extended Binary Fields”. *IEEE Transactions on Computers* 56.2, pp. 224–233.

Fan, H. ve Hasan, M. A. (2007b). “Comments on" Five, Six, and Seven-Term Karatsuba-Like Formulae”. *IEEE Transactions on Computers* 56.5, pp. 716–717.

Granger, R. ve Scott, M. (2014). *Faster ECC over* $\mathbb{F}_{2^{521}-1}$. Cryptology ePrint Archive, Report 2014/852. https://eprint.iacr.org/2014/852.

Hanrot, G., Quercia, M. ve Zimmermann, P. (2004). “The middle product algorithm I”. *Applicable Algebra in Engineering, Communication and Computing* 14.6, pp. 415–438.

IANIX (2017). *Things that use Curve25519*. URL: https://ianix.com/pub/curve25519-deployment.html (visited on 11/15/2017).

Ilter, M. B. ve Cenk, M. (2017a). “Efficient Big Integer Multiplication in Cryptography”. *International Journal of Information Security Science* 6.4, pp. 70–78.

Ilter, M. B. ve Cenk, M. (2017b). “Efficient Big Integer Multiplication in Cryptography”. *ISCTurkey 2017 Conference Proceedings* 8.13.

Kaminski, M. ve Bshouty, N. H. (1989). “Multiplicative complexity of polynomial multiplication over finite fields”. *Journal of the ACM (JACM)* 36.1, pp. 150–170.

Karatsuba, A. (1963). “Multiplication of multidigit numbers on automata”. *Soviet physics doklady*. Vol. 7, pp. 595–596.

Keskinkurt, I. (2017). “Homomorphic Encryption Based on the Ring Learning with Errors (RLWE) Problem”. MA thesis. Orta Doğu Teknik Üniversitesi Uygulamalı Matematik Enstitüsü Kriptografi Programı.

Knezevic, M., Vercauteren, F. ve Verbauwhede, I. (2009). *Speeding Up Barrett and Montgomery Modular Multiplications*.

Koblitz, N. (1987). “Elliptic curve cryptosystems”. *Mathematics of computation* 48.177, pp. 203–209.

Miller, V. S. (1985). “Use of elliptic curves in cryptography”. *Conference on the theory and application of cryptographic techniques*. Springer, pp. 417–426.

Montgomery, P. L. (2005). “Five, six, and seven-term Karatsuba-like formulae”. *IEEE Transactions on Computers* 54.3, pp. 362–369.

NIST (2013). *Federal Information Processing Standards Publication (FIPS) 186-4: Digital Signature Standard (DSS)*. https://doi.org/10.6028/NIST.FIPS.186-4.

NIST (2017). *Transition Plans for Key Establishment Schemes using Public Key Cryptography*. https://csrc.nist.gov/News/2017/Transition-Plans-for-Key-Establishment-Schemes. URL: https://csrc.nist.gov/News/2017/Transition-Plans-for-Key-Establishment-Schemes (visited on 11/15/2017).

Oseledets, I. (2011). “Improved n-term Karatsuba-like formulas in GF (2)”. *IEEE Transactions on Computers* 60.8, pp. 1212–1216.

Oseledets, I. (2008). “Optimal Karatsuba-like formulae for certain bilinear forms in GF (2)”. *Linear Algebra and its Applications* 429.8-9, pp. 2052–2066.

Project, G. (2017). *GCC Options That Control Optimization*. `https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html`. URL: `https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html` (visited on 11/15/2017).

Shumow, D. ve Ferguson, N. (2007). “On the possibility of a back door in the NIST SP800-90 Dual Ec Prng”. *Proc. Crypto*. Vol. 7.

SUPERCOP (2017). *Curve25519 ref10 Implementation*. `http://bench.cr.yp.to/supercop.html`. URL: `http://bench.cr.yp.to/supercop.html` (visited on 11/15/2017).

Sydney, T. U. of (2017). *Magma Computational Algebra System*. `http://magma.maths.usyd.edu.au/magma/`. URL: `http://magma.maths.usyd.edu.au/magma/` (visited on 11/15/2017).

Taskin, H. K. ve Cenk, M. (2018). “Speeding up Curve25519 using Toeplitz Matrix-vector Multiplication”. *Fifth Workshop on Cryptography and Security in Computing Systems*.

Taşkın, H. K. ve Cenk, M. (2018). “Speeding up Curve25519 using Toeplitz Matrix-vector Multiplication”. *Workshop on Cryptography and Security in Computing Systems of the HiPEAC2018 Conference*. CS2 ’18. Manchester, UK.

Toom, A. L. (1963). “The complexity of a scheme of functional elements realizing the multiplication of integers”. *Soviet Mathematics Doklady*. Vol. 3. 4, pp. 714–716.

Walter, C. D. (1998). “Exponentiation using division chains”. *IEEE Transactions on Computers* 47.7, pp. 757–765.

Yunuak, H. B. ve Cenk, M. (2018). “Faster Modular Exponentiation”. Poster.

Yunuak, H. B., Cenk, M., Dimitrov, V. ve Hasan, A. (2018). “Faster Modular Exponentiation”.

Zanoni, A. (2010). *Another sugar cube, please! or sweetening third powers computation*. Tech. rep. Technical Report 632, Centro” Vito Volterra”, Università di Roma” Tor Vergata”(January 2010).

TÜBİTAK
PROJE ÖZET BİLGİ FORMU

Proje Yürütücüsü:	Doç. Dr. MURAT CENK
Proje No:	115R289
Proje Başlığı:	Açık Anahtarlı Kriptografi İçin Verimli Algoritmaların Geliştirilmesi Ve Gerçeklenmesi
Proje Türü:	1001 - Araştırma
Proje Süresi:	24
Araştırmacılar:	FERRUH ÖZBUDAK
Danışmanlar:	
Projenin Yürütüldüğü Kuruluş ve Adresi:	ORTA DOĞU TEKNİK Ü. UYGULAMALI MATEMATİK ENSTİTÜSÜ KRİPTOGRAFİ ABD.
Projenin Başlangıç ve Bitiş Tarihleri:	01/06/2016 - 01/06/2018
Onaylanan Bütçe:	341213.0
Harcanan Bütçe:	242099.37
Öz:	Projenin genel amacı, kriptografide sıklıkla kullanılan modüler üst alma, polinom çarpması ve eliptik eğriler üzerindeki işlemlerin karmaşıklığını iyileştirecek geliştirmelerin yapılması ve elde edilecek yeni algoritmaların çeşitli platformlar üzerinde gerçeklenmesidir. Bu çalışmalar sonucunda modüler üst alma, eliptik eğri aritmetiği ve polinom çarpma işlemlerinde iyileştirmeler elde edilmiştir. Çalışmalar kapsamında P-521, E-521 ve Curve25519 eğrileri üzerindeki işlemler Toeplitz matris vektör çarpımları (TMVÇ) kullanılarak hızlandırılmıştır. Eliptik eğrilerin üzerinde tanımlandığı ve eleman sayıları 521 ve 255 bitlik asal sayılar olan cisimlerde çarpma işlemleri için yeni TMVÇ algoritmaları tasarlanmış ve bu algoritmaların sağladığı iyileştirmeler teorik olarak gösterilmiştir. Yapılan gerçeklemeler ile teorik çıkarımlardaki iyileştimeler pratikte de gözlemlenmiştir. Diğer taraftan polinom çarpma işleminin iyileştirilmesi için arama algoritmalarının verimi üzerine çalışmalar yapılmıştır. Polinomun terim sayısı arttıkça arama uzayı oldukça büyüdüğü için, çarpım polinomunun tüm terimlerini hesaplamak yerine, n terimli iki polinomun çarpmının ilk n teriminin hesaplanması üzerine analizler yapılmıştır. Böylece arama uzayının boyutu düşürülmüş ve Çinli Kalan Teoremi ile polinom çarpımı için algoritmalar elde edebilme olanağı sağlanmıştır. Diğer bir yaklaşım ise n terimli iki polinomum ilk l teriminin hesaplanmasıdır. Ayrıca, bu yaklaşımda arama uzayının boyutunun düşürülmesi için ikili doğrusal formların simetriklerinin alınması ve bazı terimlerin elenmesi yöntemleri kullanılmıştır. Bu yaklaşımlar arama uzayının boyutunu belirgin şekilde azaltmıştır. Ek olarak interpolasyon metodunda hesaplanacak noktalar dikkatlice seçilerek, süper singüler izojen bazlı kuantum sonrası kriptografide kullanılan Fp2 çarpma işlemi ve büyük sayıların çarpımları hızlandırılmıştır. Proje kapsamında çalışılan diğer bir konu olan modüler üst alma işleminin hızlandırılması için, literatürdeki küp şeker algoritması incelenmiştir. Bu algoritma, en küçük toplam zinciri ve karma üst alma metotları ile birlikte kullanılmıştır. Ayrıca, sonuçların daha da hızlandırılması adına, n bitlik bir tamsayının küp alma işleminden sonra 3n olan boyutunu indirgemek için kullanılan Barett metodu değiştirilmiş ve böylece teorik olarak işlem karmaşıklığında iyileştirmeler yapılmıştır.
Anahtar Kelimeler:	Kriptografik hesaplamalar, polinom çarpımı, eliptik eğri kriptografisi, modüler üst alma
Fikri Ürün Bildirim Formu Sunuldu Mu?:	Hayır

Projeden Yapılan Yayınlar:	1- Karatsuba-like formulae and their associated techniques (Makale - Diğer Hakemli Makale), 2- Efficient Big Integer Multiplication in Cryptography (Makale - Diğer Hakemli Makale), 3- Faster Residue Multiplication Modulo 521-bit Mersenne Prime and an Application to ECC (Makale - İndeksli Makale), 4- Speeding up Curve25519 using Toeplitz Matrix-vector Multiplication (Bildiri - Uluslararası Bildiri - Sözlü Sunum), 5- Efficient Big Integer Multiplicationin Cryptography (Bildiri - Uluslararası Bildiri - Sözlü Sunum), 6- Yılın en iyi tezi (Ödül - Ulusal Ödül - Üniversite, Kurum veya Kuruluşların Verdiği Ödüller), 7- Homomorphic Encryption Based on the Ring Learning with Errors (RLWE) Problem (Tez (Araştırmacı Yetiştirilmesi) - Yüksek Lisans Tezi), 8- Modular Exponentiation Methods in Cryptography (Tez (Araştırmacı Yetiştirilmesi) - Yüksek Lisans Tezi), 9- Faster Residue Multiplication Modulo 521-bit Mersenne Prime and Application to ECC (Tez (Araştırmacı Yetiştirilmesi) - Doktora Tezi),