

BAHADIR ÇOLAK

EKİN ILMAN

H.DAĞHAN ŞENGÖREN

Eliptik Eğri Kriptolojisi

1 GİRİŞ

2 ELİPTİK EĞRİ KRİPTOLOJİSİ

2.1 ELİPTİK EĞRİ 'NİN TEMELLERİ

2.2 ELİPTİK EĞRİ GRUP TEORİSİ

2.3 ELİPTİK EĞRİ AYRIK LOGARİTMA PROBLEMİ

2.4 DIFFIE HELLMAN ANAHTAR DEĞİŞİMİ

2.5 ELİPTİK EĞRİNİN ŞİFRELENMESİ

2.5.1 ENCODING

2.5.2 DECODING

2.5.3 ENCRYPTION

2.5.4 DECRYPTION

2.6 ELİPTİK EĞRİ 'NİN UYGULAMALARI

2.6.1 BITCOIN

2.6.2 SSH

2.6.3 TLS

2.6.4 BULUT MİMARİSİ

3 İMPLEMENTASYON

4 SONUÇ

1. Giriş

1976'da Whitfield Diffie ve Martin Hellman şifre ve deşifre işlemleri için farklı anahtarların kullanıldığı bir şifreleme sistemi olan Açık Anahtarlı Şifrelemeyi (Public Key Cryptography) geliştirdiler. Bundan sonra Açık Anahtarlı Şifrelemenin uygulamaları tercih edilmeye başlandı. Birçok kriptoloji uygulaması, RSA gibi, güvenliğini Çarpanlarına Ayırma Problemi (Integer Factorization Problem) ve Sonlu Alan Ayrık Logaritma Problemi gibi problemlerin uygulanma zorluğuna dayandırılmaktadır. Yıllar boyunca bu problemlerin çözülmesi için polinom zamandan daha hızlı büyüyen fakat yine de üstel bir değerden çok daha küçük olan Sub-Exponential zaman algoritmaları geliştirildi.

Neden Eliptik Eğri Kriptografisi ?

Sonuç olarak yeterince güvenli bir seviyeye ulaşılabilmek için anahtar büyüklükleri binlerce bite ulaştı. Asıl sorun, son zamanlarda 1024 bitlik anahtarlar yeterli güvenliği sağlayamamaktadırlar. Artan bu anahtar uzayları da performans düşüklüğüne neden olmaktadır. RSA'de anahtar boyutunu 2 katına çıkarmak 5-7 kata kadar performans azalmasına sebep olmaktadır. 1024 bitten daha büyük anahtar ise sınırlı kaynağa (hesaplama gücü, depolama, bant genişliği vb.) sahip telefon yada akıllı kart gibi küçük cihazlara uygulaması çok zordur. Eliptik Eğri ise RSA'ın sağladığı güvenliğin aynısını bize çok daha küçük anahtar boyutlarıyla sağlayabilmektedir. Bu küçük anahtar boyutları sınırlı kaynağa sahip olduğumuz cihazlarda RSA bazında bir güvenlik sağlamaktadır.

Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Table 1: NIST Recommended Key Sizes

Karşılaştırma	Şifreleme	İmzalama	Anahtar
RSA	☑	☑	☑
Diffie - Hellman	☒	☒	☑
DSA(Digital Signature Algorithm)	☒	☑	☒
Eliptik Eğri	☑	☑	☑

USIP PRO Cryptographic Performance

Algorithms	Speed
SHA-1	2083kBps
RSA 2048 CRT decrypt	400ms
RSA 2048 encrypt	18ms
ECDSA P-192 sign	23ms
ECDSA B-163 sign	16ms

2. ELİPTİK EĞRİ KRİPTOLOJİSİ

2.1 ELİPTİK EĞRİ 'ÜN TEMELLERİ

Eliptik Eğri aşağıda denklemi verilmiş noktaların kümesi şeklinde gösterilir.

$$y^2=x^3+ax+b$$

ve ax^3+bx^2+cx+d formunda ki kübik polinomun discriminantı

$$\Delta= b^2c^2 - 4ac^3 - 4b^3d - 27a^2d^2 + 18abcd$$

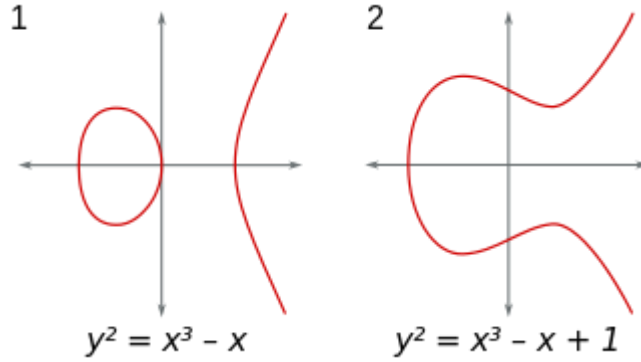
Şeklindedir. Buradan,

$$((r_1 - r_2)(r_1 - r_3)(r_2 - r_3))^2 = - (4a^3 + 27b^2)$$

Olduğundan, çakışık kökün olmaması için

$$4a^3 + 27b^2 \neq 0$$

Olması lazımdır.



Eliptik Eğri x- eksenine simetriktir ve a ve b nin değerine göre değişiklik grafikler oluştururlar. Eğer denklemin tekrarlı kökü yoksa x^3+ax+b in $y^2=x^3+ax+b$ için bir grup olduğunu söyleyebiliriz. Elipsel bir grup ile kast edilen, elipsel eğrinin üzerinde tanımlı olan noktalardır ve bu noktalar öyle bir O noktasında sonsuza gider. Bu O noktasını ∞ olarak tanımlayacak olursak,

$$\{ (x,y) \in \mathbb{R}^2 \mid y^2=x^3+ax+b, 4a^3 + 27b^2 \neq 0 \} \cup \{\infty\}$$

olarak genelleştirebiliriz.

Weierstrass eşitliği olarak adlandırılan ve Eliptik Eğri 'nin en genel formu ,

$$E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 ,$$

a_1, \dots, a_6 sabit. Weierstrass eşitliğinin karakteristiği 2 değilse , 2ye bölüp kareye tamamlarsak:

$$\left(y + \frac{a_1x}{2} + \frac{a_3}{2} \right)^2 = x^3 + \left(a_2 + \frac{a_1^2}{4} \right)x^2 + \left(a_4 + \frac{a_1a_3}{2} \right)x + \left(a_6 + \frac{a_3^2}{4} \right)$$

Ve burdan $y_1 = y + \frac{a_1x}{2} + \frac{a_3}{2}$ ve a'_2, a'_4, a'_6 sabitleri olarak yazarsak,

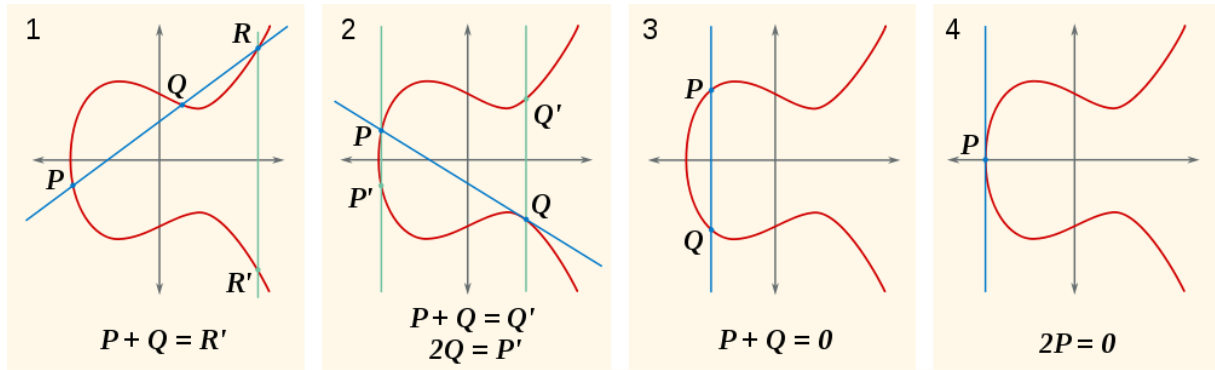
$$y_1 = x^3 + a'_2x^2 + a'_4x + a'_6$$

olur. Ayrıca eşitliğin karakteristiği 3 te değil değilse o halde $x_1 = x + a'_2/3$ için,

$$y_1 = x_1^3 + Ax + B, \quad A \text{ ve } B \text{ sabit}$$

2.2 Eliptik Eğri Grup Teori

Elipsel gruplar toplanabilir gruplardır. Toplama bu grupların en temel fonksiyonudur. Eliptik Eğri üzerinde iki noktayı (aynı iki nokta da olabilir) alıp grupsal toplama yaparak başka bir nokta elde edeceğiz.



Şekil 1.0

Elipsel Eğriler Üzerinde Nokta Gösterimi

Şekil 1.0da da gösterildiği üzere Eliptik Eğri üzerinde:

$$P = (x_1, y_1) \quad \text{ve} \quad Q = (x_2, y_2)$$

olmak üzere $y^2 = x^3 + Ax + B$ denklemini sağlayacak şekilde iki nokta alalım. P ve Q dan geçecek şekilde bir L doğrusu çizelim. Bu noktaları toplarsak,

$$P + Q = R'$$

elde ederiz. Fakat burada bahsettiğimiz toplama bildiğimiz toplamadan farklıdır. Asıl gösterimi $P +_E Q$ şeklinde olmalıdır. $P + Q$ 'nın bulunabilmesi için aşağıda ki iki şartın sağlanması gereklidir;

1. L doğrusu R noktasından da geçecek.
2. R nin x-eksenine göre yansıması R', R' = -R olacak.

Ayrıca dikkat edilmesi gereken birkaç kural;

1. P_{∞} 'un yansıması yine P_{∞} 'dur. Yani $-P_{\infty} = P_{\infty}$ olur.
2. Eğer $P = Q$ ise L doğrusu $P = Q$ 'nun tanjantıdır.
3. $P + Q = Q + P$ yani bu grup abelyandır.

Şimdi P ve Q dan geçecek şekilde bir L doğrusu çekelim ve $P \neq Q$ ve bu iki noktanın da ∞ olmadığını varsayalım. Bu doğrunun eğimi;

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

olur. Eğer $x_2 = x_1$ ise bu durumda doğru dikey olur. O yüzden şimdilik $x_2 \neq x_1$ kabul ediyoruz. L nin denklemi,

$$y = m(x - x_1) + y_1$$

elde ederiz. L doğrusu ile E eğrisinin kesiştiği yerleri belirleyebilmek için denklemde yerine koyarsak;

$$(m(x - x_1) + y_1)^2 = x^3 + Ax + B$$

şeklini alır. Kubik formda ki bu denklemi

$$x^3 + Ux^2 + Vx + W = 0$$

biçiminde düzenleyelim. Bu denklemin 3 kökü olacaktır. Normalde kübik bir denklemi çözmek kolay değildir fakat biz zaten iki kökü yani x_1 ve x_2 'yi biliyoruz. Üçüncü kökü elde edebilmek için yapmamız gereken,

$$x^3 + Ux^2 + Vx + W = (x - r)(x - s)(x - t) = x^3 - (r + s + t)x^2 + \dots$$

Burdan

$$r + s + t = -U$$

elde ederiz. Eğer iki kökü yani r ve s yi biliyorsak, dolayısıyla üçüncü kök

$$t = -U - r - s$$

yapar. Bizim denklemimiz de bu

$$x = m^2 - x_1 - x_2$$

ve

$$y = m(x - x_1) + y_1$$

Şimdi elimizde ki üçüncü nokta, $R'(x_3, y_3)$,

$$x_3 = m^2 - x_1 - x_2$$

ve

$$y_3 = m(x_1 - x_3) - y_1$$

Şimdi 4 durum söz konusu oluyor,

1. $x_1 \neq x_2$

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

iken

$$x_3 = m^2 - x_1 - x_2$$

ve

$$y_3 = m(x_1 - x_3) - y_1$$

2. $x_1 = x_2$ ama $y_1 \neq y_2$

$$P + Q = \infty$$

3. $P = Q$ and $y_1 \neq 0$

$$m = \frac{3x_1^2 + A}{2y_1}$$

iken

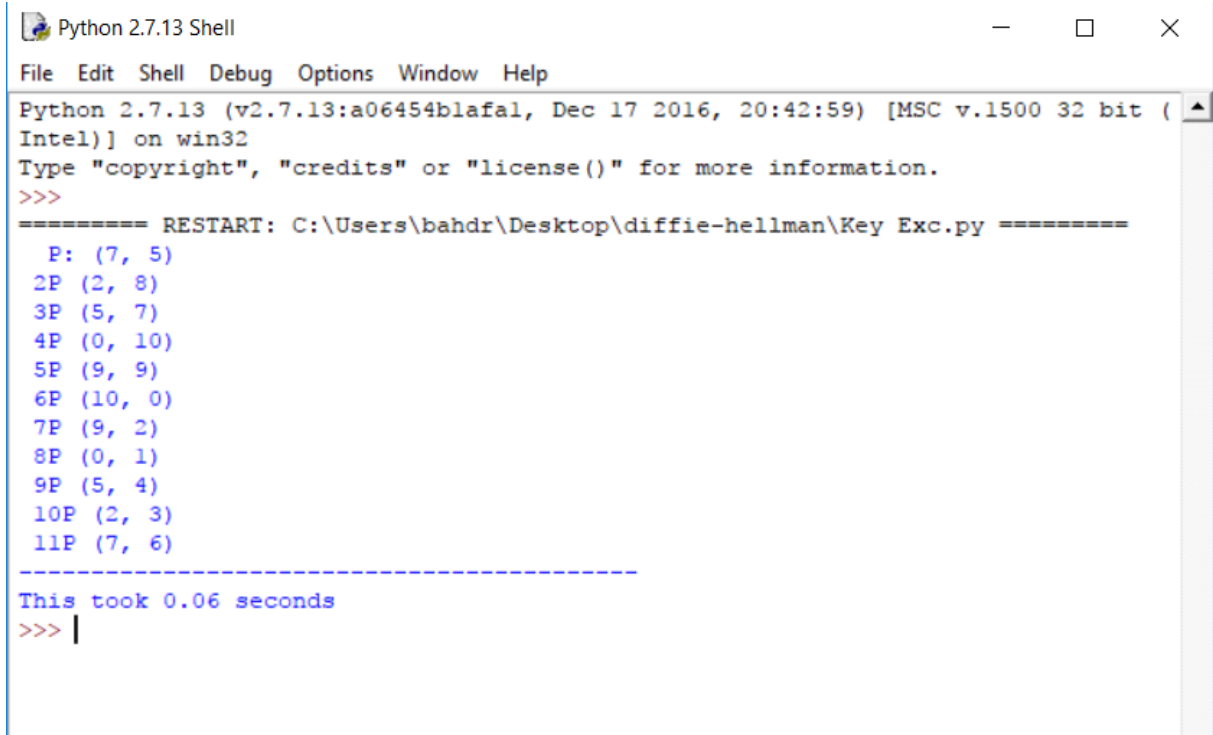
$$x_3 = m^2 - 2x_1$$

ve

$$y_3 = m(x_1 - x_3) - y_1$$

4. $P = Q$ and $y_1 = 0$

$$P + Q = \infty$$



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454blafal, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\bahdr\Desktop\diffie-hellman\Key Exc.py =====
P: (7, 5)
2P (2, 8)
3P (5, 7)
4P (0, 10)
5P (9, 9)
6P (10, 0)
7P (9, 2)
8P (0, 1)
9P (5, 4)
10P (2, 3)
11P (7, 6)
-----
This took 0.06 seconds
>>> |
```

Eliptik Eğri Ayırık Logaritma Problemi

$$E: y^2 = x^3 + Ax + B \quad , A, B \in F_p$$

şeklinde tanımlanmış olsun. S ve T , $E(F_p)$ de tanımlı noktalar olsun ve $1 < m < p$ olacak şekilde

$$T = mS$$

Bu özelliğe sahip (en küçük) m tam sayısına S 'ye göre T 'nin Discrete Logaritması (veya Dizin) olarak adlandırılır ve

$$m = \log_S(T)$$

olarak gösterilir. Buna göre bir sayının kendisi ile defalarca kere toplanması dairesel bir grup oluşturur.

$$y^2 = x^3 + 9x + 17 \mod 23$$

şeklinde tanımlı eliptik bir eğrimiz olsun. $Q = (4,5)$ noktası için $P = (16,5)$ noktasına göre ayrık logaritma nedir? Bu sorunun çözümlerinden birisi Q değerine ulaşınca kadar P noktasının kendisi ile toplanmasıdır. Bu işlem yapılırsa:

$$P = (16,5) \quad 2P = (20,20) \quad 3P = (14,14) \quad 4P = (19,20) \quad 5P = (13,10) \quad 6P = (7,3) \quad 7P = (8,7) \quad 8P = (12,17) \quad 9P = (4,5)$$

dolayısıyla P tabanında Q noktasının logaritma değeri

$$\log_p(Q) = 9$$

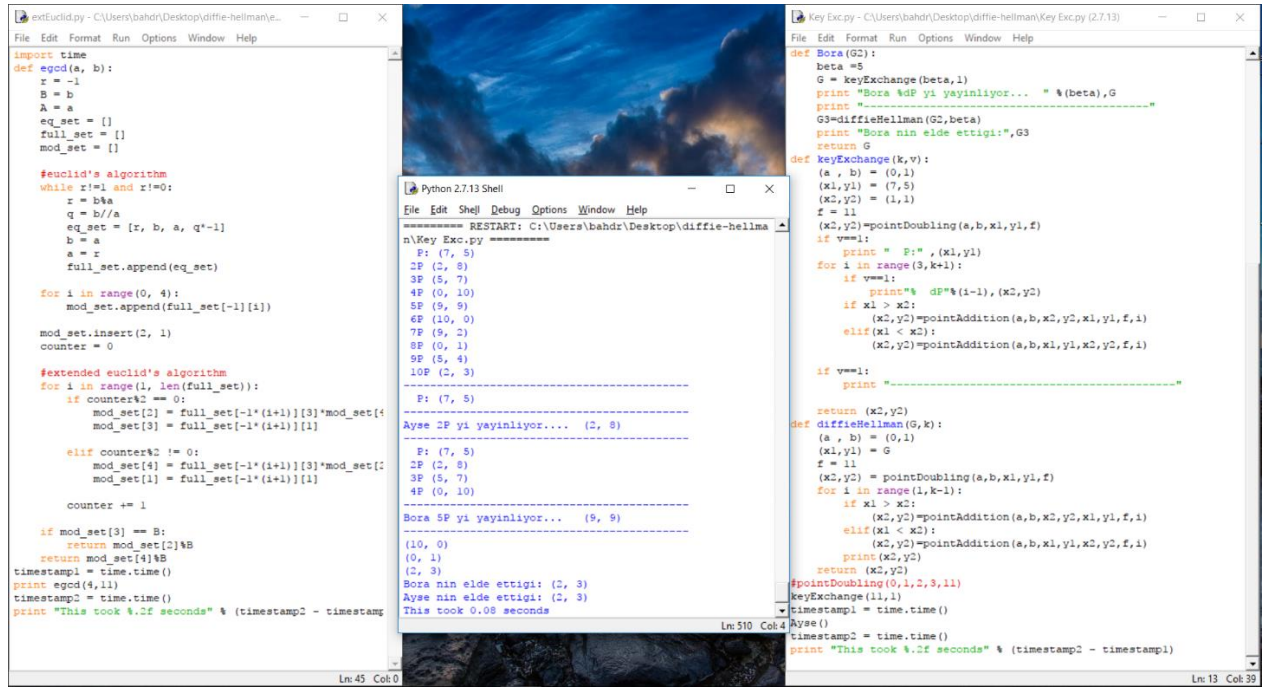
olarak bulunur.

Diffie-Hellman Anahtar Değişimi

Bora ve Ayşe güvenli bir ortak kanaldan konuşmak istiyorlar. Bunun için Eliptik Eğri Diffie-Hellman Anahtar Değişim Protokolünü kullanmaya karar veriyorlar

- 1) Ayşe ve Bora ortak, sonlu bir F_q cismiyle sınırlı bir E Eliptik Eğrisi seçiyorlar.
- 2) Ayşe gizli $1 < \alpha < n-1$ olacak şekilde bir α seçer.
- 3) Bora gizli $1 < \beta < n-1$ olacak şekilde bir β seçer.
- 4) Ayşe $A = \alpha(G)$ ve Bora $B = \beta(G)$ 'yi hesaplar
- 5) Ardında ikisi de birbirine açık anahtar olan A ve B 'yi gönderirler.
- 6) Ayşe $P = \beta.A = \beta. \alpha(G)$ 'yi ve Bora $P = \alpha.B = \alpha. \beta(G)$ 'yi hesaplar.

Şuanda Ayşe ve Bora aynı P noktasına sahipler ve tüm haberleşmeleri için bu gizli noktayı kullanabilirler.



```
import time
def egcd(a, b):
    r = -1
    s = 0
    A = a
    eq_set = []
    full_set = []
    mod_set = []

    #euclid's algorithm
    while r!=1 and r!=0:
        r = b%a
        q = b//a
        eq_set = [r, b, a, q*-1]
        b = a
        a = r
        full_set.append(eq_set)

    for i in range(0, 4):
        mod_set.append(full_set[-1][i])

    mod_set.insert(2, 1)
    counter = 0

    #extended euclid's algorithm
    for i in range(1, len(full_set)):
        if counter%2 == 0:
            mod_set[2] = full_set[-1*(i+1)][3]*mod_set[4]
            mod_set[3] = full_set[-1*(i+1)][1]

        elif counter%2 != 0:
            mod_set[4] = full_set[-1*(i+1)][3]*mod_set[2]
            mod_set[1] = full_set[-1*(i+1)][1]

        counter += 1

    if mod_set[3] == B:
        return mod_set[2]%B
    return mod_set[4]%B

timestamp1 = time.time()
print egcd(4,11)
timestamp2 = time.time()
print "This took %.2f seconds" % (timestamp2 - timestamp1)
```

```
def Bora(G2):
    beta = 5
    G = keyExchange(beta,1)
    print "Bora %dP yi yayinliyor..." % (beta), G
    print "-----"
    G3=diffieHellman(G2,beta)
    print "Bora nin elde ettigi:",G3
    return G

def keyExchange(k,v):
    (a , b) = (0,1)
    (x1,y1) = (7,5)
    (x2,y2) = (1,1)
    f = 11
    (x2,y2)=pointDoubling(a,b,x1,y1,f)
    if v==1:
        print "P = ", (x1,y1)
    for i in range(3,k+1):
        if v==1:
            print "% dP"%(i-1), (x2,y2)
            if x1 > x2:
                (x2,y2)=pointAddition(a,b,x2,y2,x1,y1,f,i)
            elif (x1 < x2):
                (x2,y2)=pointAddition(a,b,x1,y1,x2,y2,f,i)

        if v==1:
            print "-----"

    return (x2,y2)

def diffieHellman(G,k):
    (a , b) = (0,1)
    (x1,y1) = G
    f = 11
    (x2,y2) = pointDoubling(a,b,x1,y1,f)
    for i in range(1,k-1):
        if x1 > x2:
            (x2,y2)=pointAddition(a,b,x2,y2,x1,y1,f,i)
        elif (x1 < x2):
            (x2,y2)=pointAddition(a,b,x1,y1,x2,y2,f,i)
        print (x2,y2)
    return (x2,y2)

#pointDoubling(0,1,2,3,11)
keyExchange(11,1)
timestamp1 = time.time()
Ayse()
timestamp2 = time.time()
print "This took %.2f seconds" % (timestamp2 - timestamp1)
```

```
===== RESTART: C:\Users\bahdr\Desktop\diffie-hellma
n\Key Exc.py =====
P: (7, 5)
2P (2, 8)
3P (5, 7)
4P (0, 10)
5P (9, 9)
6P (10, 0)
7P (9, 2)
8P (0, 1)
9P (5, 4)
10P (2, 3)
-----
P: (7, 5)
Ayse 2P yi yayinliyor... (2, 8)
-----
P: (7, 5)
2P (2, 8)
3P (5, 7)
4P (0, 10)
-----
Bora 5P yi yayinliyor... (9, 9)
-----
(10, 0)
(0, 1)
(2, 3)
Bora nin elde ettigi: (2, 3)
Ayse nin elde ettigi: (2, 3)
This took 0.08 seconds
```

Eliptik Eğri Şifrelenmesi

Koblitz's Yöntemi

Encoding

$$E: y^2 = x^3 + Ax + B \quad , A, B \in F_p$$

seçilir.

A	B	C	D	E	F	G	H	I	J	K	L	M
65	66	67	68	69	70	71	72	73	74	75	76	77
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
78	79	80	81	82	83	84	85	86	87	88	89	90

Şifrelenecek mesajın her bir karakterinin ASCII değeri eliptik eğri üzerinde ki (x,y) noktasının x koordinatına karşılık gelecek şekilde ayarlanır. Karşılık gelen y koordinatı ise;

$$(y^2) \bmod p = (x^3 + ax + b) \bmod p$$

Formülü ile elde edilir. $\forall x \in \{0,1, \dots, (p-1)\}$ karşılık gelen bir y değeri olmalıdır. Encoding 'in asıl sorunu budur fakat Koblitz Yöntemi bu sorunu aşmamızda bize yardımcı oluyor.

Koblitz Yöntemin de yardımcı bir k parametresi seçilir. Mesajın her bir karakteri m için

$$x = m * k + 1$$

Şeklinde bir x seçilir. x 'i formülde yerine koyduğumuz da,

$$(y^2) \bmod p = (x^3 + ax + b) \bmod p$$

y değerini elde ederiz. Eğer y 'ye karşılık gelen bir nokta bulunursa (x,y) noktası bu msj karakterinin noktası olur. Aksi takdirde,

$$x = m * k + 2 \text{ den } x = m * k + (k - 1)$$

y 'ye bir karşılık bulununcaya kadar verilen aralıktan yeni bir x değeri seçilir. Eğer ki hala bir y karşılığı bulunamamışsa yardımcı parametre olan k değeri 1 arttırılır. Mesajın her bir karakteri için bu prosedür uygulanır.

Decoding

Alınan şifre noktasının decrypt edilmesi bize mesajın eliptik eğri üzerinde ki noktalarını verecektir. Alınan noktalardan x -koordinatı üzerinde,

$$x - 1/k$$

hesaplaması yapılır. Bu bölümün sonucu bize msjın ASCII değerini verecektir.

ElGamal Açık Anahtar Şifrelemesi

Encryption

Ayşe, Bora 'ya bir mesaj göndermek istesin. Öncelikle Bora

$$E: y^2 = x^3 + Ax + B \quad , A, B \in F_p$$

açık bir eliptik eğrisi ve bu eğri üzerinde bir P noktası seçer. Daha sonra gizli bir s tam sayısı seçer ve

$$B = sP$$

Hesaplar. B ve P noktaları Bora 'nın açık anahtarlarıdır. Ayşe Bora 'ya mesaj göndermek için aşağıdaki adımları uygulaması gerekir:

- 1- Bora 'nın açık anahtarını ister.
- 2- Mesajını $M \in E(F_q)$ olacak şekilde eğri üzerinde bir noktaya çevirir.
- 3- Rastgele olarak gizli bir k tam sayısı seçer ve $M_1 = kP$ 'yi hesaplar.
- 4- $M_2 = M + kB$ olacak şekilde hesaplar.
- 5- M_1 ve M_2 'yi Bora ya gönderir.

Decryption

Mesajı alan Bora

$$M = M_2 - sM_1$$

şeklinde decrypt eder.

$$M_2 - sM_1 = (M + kB) - s(kP) = M + k(sP) - skP = M$$

ELİPTİK EĞRİ 'NİN UYGULAMALARI

Bitcoin

Bitcoin ile doğrudan bir taraftan diğerine finansal ödeme yapmaksızın çevrimiçi ödeme yapmak mümkün hale gelmiştir. Bitcoin, block chaining ile yürütülen bütün işlemlerin günlüğüdür(her şeyin kaydının veya notunun tutulması gibi).

Her blok, bir önceki bloğun SHA-256 karmasını içermekte ve blokları zincirlemektedir. Bitcoin'de, ECDSA özel anahtarı genellikle bir kullanıcının hesabı olarak kullanılır. Bitcoin'leri kullanıcı A'dan kullanıcı B'ye aktarmak, dijital bir imza ekleyerek gerçekleştirilir. (Kullanıcı A'nın özel anahtarını kullanarak) İmza A kullanıcısının ortak anahtarı ile doğrulanabilir durumdadır. Bitcoin adresi hash fonksiyonlarını kullanarak elde edilebilir. Public key iki kere hash edilir.



Secure Shell (SSH)

Elıptik Eğri SSH içerisinde de kullanılmaktadır. Örneğin, SSH-2'de, oturum anahtarlarına bir Diffie-Hellman anahtar değişimi kullanılarak ulaşılır. RFC (Request for Command) 5656, SSH'da kullanılan kısa ömürlü Elıptik Eğri Diffie-Hellman anahtar değiştirme yöntemini belirtir.

Her server'ın kendini client'a tanıtılabileceği bir host key'i bulunmaktadır.

Server, anahtar değişimi sırasında ana anahtarını client'a gönderir ve user, anahtar parmak izinin kaydedilen değerle eşleştiğini doğrular. Bu host key bir ECDSA genel anahtarı olabilir. Müşteriler client kimlik doğrulaması için ECDSA genel anahtarlarını kullanabilir.

Transport Layer Security (TLS)

İletim katmanı güvenliğinde Eliptik Eğri protokollerin çeşitli yerlerinde ortaya çıkmaktadır. RFC 4492, TLS için Eliptik Eğri şifreleme takımlarını belirtir. Bütün şifreleme takımları Eliptik Eğri Diffie-Hellman (ECDH) anahtar değişimi kullanmaktadır. ECDH anahtarları ya uzun vadeli olabilir (bu durumda, farklı anahtar değişimleri için yeniden kullanılmaktadır) ya da kısa süreli olabilir (bu durumda her anahtar değişimi için yeniden oluşturulur). TLS sertifikaları, sunucunun kendisini doğrulamak için kullandığı bir ortak anahtarı da içerir. ECDH anahtar alışverişi ile bu genel anahtar ECDSA veya RSA olabilir.

SSH'den farklı olarak, bir TLS server'ı, desteklediği şifreli takımların veya eğrilerin tamamını göndermez. Bunun yerine, client desteklenen şifreleme takımlarının ve Eliptik Eğri'lerin listesini gönderir. Server, client ile ortak olarak herhangi bir şifreleme takımını desteklemiyorsa, tek bir yanıt ile bağlantıyı kopartır. Sunucunun hangi eğrileri desteklediğini öğrenmek zorlaştığında client birden çok TLS bağlantısı kullanmalıdır.

Bulut Mimarisi

Tirthani ve Ganesan Diffie-Hellman Anahtar değişimi ve ECC kullanan basit bir güvenli bulut mimarisi tasarlamışlardır. Bu mimari 4 adımdan oluşur:

- 1) bağlantının kurulması (HTTPS ve SSL)
- 2) hesap oluşturma (bulut mimarisi her kullanıcıya unique ID ve private/public anahtar eşleri verir)
- 3) yetkilendirme (verilen ID'nin doğruluğunun kontrolü)
- 4) Data değişimi (DH)

Client, server'dan bir bilgi istediğinde bu sorgu server'ın public, client'ın private anahtarı ile şifrelenmiş biçimde saklanır. Bu bilgi server'a gönderilir ve burada server'ın private anahtarı ile deşifre edilir. Server kendi private anahtarı ve Client'ın

public anahtarı ile şifreler ve Client'a yollar. Sonuç olarak Client artık kendi private anahtarı ile deşifre edip bilgiye erişebilir.

Kuantum Bilgisayarlar ve ECC

Eliptik Eğri kriptografisi, bir kuantum bilgisayar algoritması olan Shor algoritmasına karşı kırılabilir durumdadır. Anahtar uzayı daha küçük olduğu için bu sistemi kırmak için gerekli olan tema RSA'e göre daha olası kabul edilir. Proos ve Zalka'nın testlerine göre, 2048-bit RSA'i kırmak için yaklaşık olarak 4096 qubit(kuantum biti) gerekli olurken, aynı güvenlik seviyesindeki 224-bit ECC'yi kırmak 1300 -1600 qubit'inin yeterli olduğu görülmüştür. Kuantum bilgisayarlar kaygısına karşı Supersingular Isogeny Diffie–Hellman Key Exchange

Implementasyon

Donanımsal Özellikler:

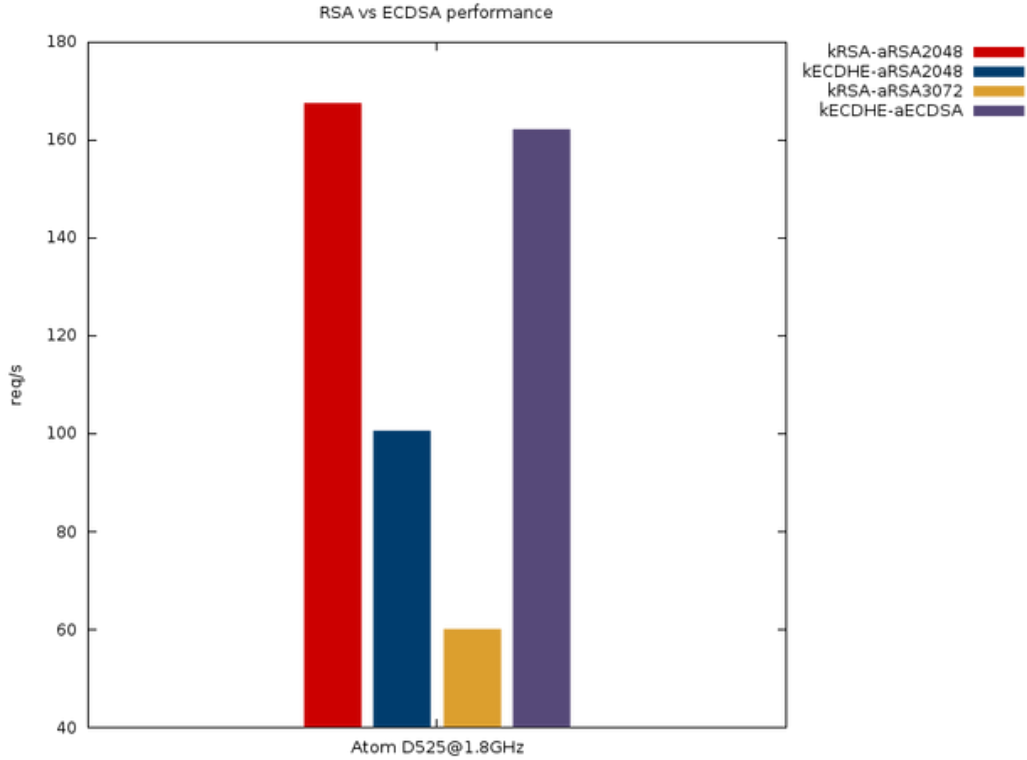
Aşağıda ECDSA için gerçekleştirilen bir mikroişlemcinin donanımsal özellikleri verilmiştir. Bu özellikler günümüz bilgisayarları ile karşılaştırıldığında çok az kaynak (bellek) kullanan bir işlemci olduğu görülmektedir.

- 1. (8kB of internal flash for program memory, 1kB of internal SRAM, 0.5kB of internal EEPROM),*
- 2. microcontroller CP210X (USB-UART bridge),*
- 3. external EEPROM memory (at least 32kB) to store multiples of generator and public objects (certificates, public keys etc.).*

Bütün bu parçalar çok ufaktır ve 20mm x 50mm'lik bir anakarta rahatlıkla sığdırılabilir.

//Bu kısım Final raporunda daha çok açıklanacak. Güzel bir örnek buldum.Kısaca;

RSA ve ECDSA hız karşılaştırması (Donanım: Atom D525 @ 1.8GHz with 4GiB RAM)



Kaynak:

1. <http://bilgisayarkavramlari.sadievrenseker.com/2008/05/06/ellipsel-egri-elliptic-curve/>
2. Elliptic curves : number theory and cryptography / Lawrence C. Washington.
3. Cryptography, information theory, and error-correction : a handbook for the 21st century / Aiden A. Bruen, Mario A. Forcinito.
4. <https://www.globalsign.com/en/blog/elliptic-curve-cryptography>
5. <https://eprint.iacr.org/2013/734.pdf>
6. http://students.mimuw.edu.pl/~ac181080/data/ecc_in_small_devices.pdf
7. https://scholar.google.com.tr/scholar?q=Elliptic+Curve+Cryptography+and+Its+Applications+to+Mobile+Devices&hl=tr&as_sdt=0&as_vis=1&oi=scholar&sa=X&ved=0ahUKewjfm6S-roTUAhWK2hoKHRGLDvQQgQMIJTAA
8. <https://www.maximintegrated.com/en/app-notes/index.mvp/id/5421>
9. <https://securitypitfalls.wordpress.com/2014/10/06/rsa-and-ecdsa-performance/>
10. <https://eprint.iacr.org/2013/734.pdf>
11. <https://tools.ietf.org/html/rfc5656>
12. <https://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/>
13. http://searchdl.org/public/book_series/elsevierst/7/24.pdf