RSA Algoritması Kullanılarak Hata Düzeltmeli ve Dijital İmzalı Protokol Geliştirme

Nursel AKÇAM
Gazi Üniversitesi, Mühendislik-Mimarlık Fakültesi, Elektrik-Elektronik Mühendisliği Bölümü
06570 Maltepe, ANKARA

ÖZET

Bugün evlerde kullanılan elektronik cihazlardan, endüstriyel veya askeri alanda kullanılan cihazlara kadar bir çok anlamda kullanımı kolaylaştırıcı çözümler sunulmaktadır. Bu tarzda çözümler geliştirilirken iletişimin kötü niyetli şahıslardan uzak tutularak, gizlilik içinde güvenle sürdürülebilmesi için karşılaşılabilecek problemleri hesaba katarak geliştirilen bir yeni protokole ihtiyaç vardır. Mevcut çalışmada böylesi bir ihtiyaca model olabilecek bir protokol yapısı oluşturulması hedeflenip önerilmiştir. Uzun yıllardır üzerinde çalışılarak olgunlaştırılan ve birçok farklı yöntemin sunulduğu şifreleme algoritmaları, geliştirilmesi düşünülen ve özellikle kontrol sistemlerini hedef alan uygulamalarda, kullanılması tasarlanan protokol için temel teşkil etmektedir. Bu bakımdan, özellikle son yıllarda bu alanda hakimiyet kurmuş açık anahtarlı şifreleme yöntemleri incelenmiş ve en güvenilir yöntemlerden olan RSA algoritması protokolde kullanılmaya uygun görülmüştür. Uygulama platformu olarak seçilen Microsoft Visual C++ 6.0 ile herhangi bir hazır kütüphane kullanmaksızın ihtiyaç duyulacak tüm fonksiyonlar elde edilerek yazılımı gerçekleştirilmiştir.

Anahtar Kelimeler: Protokol, Şifreleme, Kimlik Doğrulama, Hata Tespiti

Developing an Error Correctional and Digitally Signatured Protocol by Using RSA Algorithm

ABSTRACT

Today, the electronic devices which from using in the houses to industrial or military purposes have a lot of solutions that makes their usage easier in this manner. There are some needs that should get in count during such progresses because of to prevent to get the information in communication from third persons. The main aim of this study, is to present a structure that can be model provides such necesseties. The cryptographic algorithms that have been developing for years is a reference for such a protocol which is especially planned for control based system targeting applications. For this reason the asimetric public key cryptograpy algorithms which are particularly progressed in past years have been invastigated and one of the most reliable public key algorithm, RSA has been decided to use for the protocol. It has been decided to use Microsoft Visual C++ 6.0 as a development platform. All necessary fuctions have been fully implemented without using any predefined class or library.

Keywords: Protocol, Cryptograpy, Authentication, Error Detection

1. GİRİŞ

Bu çalışmada, teknolojik gelişimlerin beraberinde getirdiği veri iletişiminde güvenlik sorunlarına getirilen güncel çözümler derlenerek bir protokol altında toplanması hedeflenmiştir. Protokolün yapı itibari ile RF haberleşme ve kontrol sistemlerinde kullanılmaya uygun olması amaçlanmıştır.

Böyle bir protokolün geliştirilmesinde en önemli kısmı, şüphesiz şifreleme algoritmaları oluşturmaktadır. Şifreleme algoritmaları değerlendirilirken protokolün kontrol niteliğine uygunluğu gözetilerek gerçek zamanlı işlemleri engellemeyecek derecede hızlı olmasına dikkat edilmelidir. Şifrelemenin çarpıcı gelişimini 1976 yılında Diffie ve Hellman yapmış oldukları çalışmalarını yayınladıkları makale ile aynı yıl sunmuşlardır. Bu makalede, şifreleme için evrim niteliğinde olan "açık anahtarlamalı şifreleme" tanıtılmış, kesikli logaritmik hesaplamaların zorluğuna dayanan anahtar değişimi metodu kullanılmıştır (1). Sonraki çalışmalar da açık anahtarlı şifreleme algoritmalarını geliştirmiştir.

Benzer bir protokol oluşturma çabası 2004 yılında Nguyen, H. Aziz ve S.M. Ryan, tarafından önerilmiştir. Aziz ve Ryan araştırmalarında uzaktan kumanda edilebilen sistemler için yeni bir protokol modelini sunmuşlardır. Çalışmalarında odaklandıkları nokta iletişimin çoğunlukla komutlardan oluşması, dolayısı ile mümkün olduğunca düşük bant genişliğine ihtiyaç duyulması idi. Bu bakımdan güvenlik ihtiyacına fazlaca önem verilmemiş, sadece algoritmanın gizli tutulması önerilmiştir. Güvenlik için harcanacak bant genişliği, veri kayıplarının azaltılması için hata kontrolüne ayrılmıştır (2).

Güvenlik kaygısı gütmeyen çok daha genel bir çalışma Wolisz, Schieferdecker ve Walch tarafından sunulmuştur. Bu araştırmada iletişim protokolleri bir bütün olarak düşünülmüş, herhangi bir uygulamaya dahil

Digital Object Identifier 10.2339/2007.10.1.41-51

edilebilmeleri için ihtiyaç duyulacak gereksinimler belirlenerek birtakım test yöntemleri önerilmiştir (3).

2. AÇIK ANAHTARLI ŞİFRELEME

Açık anahtarlı şifrelemenin genel amaclarından birisi, gizli anahtarların dağıtımıdır. Açık anahtarlı şifreleme/deşifreleme algoritmaları, şifreleme için bir anahtara, deşifreleme için ise bu anahtarla ilişkisi olan başka ikinci bir anahtara ihtiyaç duyarlar. Bu yöntemle güvenlik sağlamış olur. Sadece kriptografik algoritma ve deşifreleme anahtarı verilmişken, bir takım basit hesaplamalar ile şifreleme anahtarını bulmak mümkün olmaz. Şekil 1, açık anahtarlı şifreleme yapısını göstermektedir. Bu yapıda başlıca adımlar şunlardır:

1. Ağdaki her sistem, mesaj alındığında şifreleme ve deşifreleme için kullanacak olduğu anahtar parçasını yaratır.
2. Her sistem, şifreleme anahtarını herkesçe erişilebilecek bir dosya ya da yazmaç içerisine kaydederek paylaştırır. Bu anahtarın, açık olan kısmıdır (public key). Özel anahtar saklı tutulur.
3. Eğer A, B'ye bir mesaj yollamak isterse, mesajı B'nin açık anahtarını kullanarak şifreler.

B mesajı aldığında, bu mesajı kendi özel anahtarını kullanarak deşifre eder. Diğer hiç bir alıcı mesajı deşifreleyemez, çünkü mesajı deşifre edecek olan özel anahtarı sadece B'de mevcuttur.

Şekil 1'den görüldüğü üzere her katılımcı, diğerlerinin açık anahtarlarına erişim hakkına sahiptir ve katılımcılar özel anahtarlarlarını lokal olarak yaratırlar.Bu nedenle, özel anahtarların paylaşılmasına gerek yoktur.

3. RSA ALGORİTMASI

RSA yapısı, birtakım n`ler için 0 ve (n-1) arasındaki tamsayılardan oluşan şifreli ve düz metinin içinde yer alan bir blok şifrelemedir.

Rivest, Shamir ve Adleman tarafından ortaya konulan yapı (4); düz metnin, blokların içinde şifrelenmesini esas alır. Her blok, bir 'n' sayısından daha az bir ikili değere sahiptir. Blok büyüklüğü, $\log_2(n)$'e eşit veya ondan daha küçük olmalıdır; pratikte, blok büyüklüğü 2^k bittir, n ise $2^k < n \le 2^{k+1}$ aralığındadır. Şifreleme ve deşifreleme; e ve d anahtarları oluştumak için seçilen asal sayılar olmak üzere, düz metin bloğu M ve şifreli metin bloğu C için aşağıdaki biçimdedir:

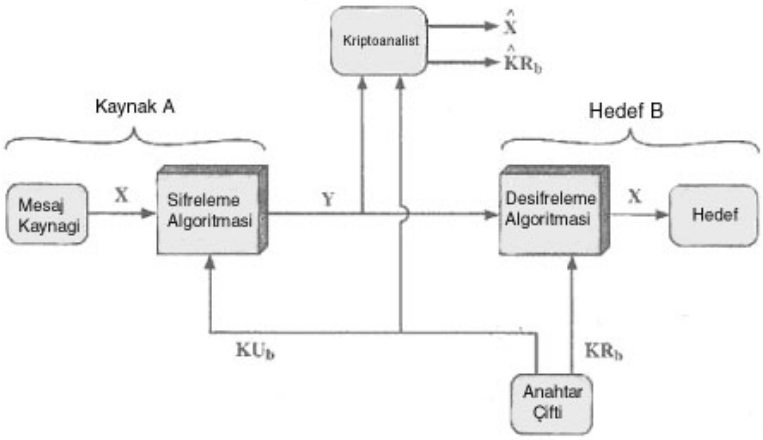
$$C = M^e \bmod n \qquad (1)$$

$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n \quad (2)$$

Gönderen ve alıcının n'nin değerini bilmesi gerekmektedir. Gönderen e'nin değerini bilir, ve sadece alıcı d'nin değerini bilmek durumundadır. Böylece; $KU = \{e, n\}$ bir genel anahtar, ve $KR = \{d, n\}$ bir özel anahtar olur ve bu bir genel anahtarlı şifreleme algoritmasıdır. Bu algoritmada aşağıdaki gereklilikler yerine getirilmelidir:

1. $M < n$ olduğu koşulda, $M^{ed} = M \bmod n$ iken; e, d, n değerlerini bulmak mümkün olmalıdır.
2. $M < n$ koşulunu sağlayan tüm M değerleri için, M^e ve C^d hesaplanması nispeten kolay olmalıdır.
3. Sadece e ve n verildiğinde, d'nin hesaplanması imkansız olmalıdır.

RSA yapının bileşenleri sırasıyla şöyledir:



Şekil 1 Açık anahtarlı şifreleme yapısı

p,q; iki asal sayı (gizli, seçilmiş)

n = pq; (genel, hesaplanmış)

e; gcd($\Phi(n)$, e) = 1; $1 < e < \Phi(n)$ olacak şekilde (genel, seçilmiş)

$$d \equiv e^{-1} \bmod \Phi(n) \text{ ; (gizli, hesaplanmış) (3)}$$

{d,n} çifti özel anahtarı, {e,n} çifti ise genel anahtarı oluşturur. Bir B kullanıcısının, A kullanıcısına, A kullanıcısının genel anahtarını kullanarak bir M mesajını göndermek istediğini varsayalım. Bu durumda B, $C = M^e (\mathrm{mod}\, n)$ formülü yardımı ile C şifreli mesajını elde edecek ve bunu A'ya gönderecektir. A kullanıcısı bu mesajı aldığında, $M = C^d (\mathrm{mod}\, n)$ formülü ile mesajı deşifre edecektir. Yukarıda belirtildiği gibi, e ve d (3) nolu denkliği sağlayacak şekilde seçilmişti. Bunun sonucunda,

$$ed \equiv 1 \bmod \Phi(n) \quad (4)$$

olur. Bu yüzden ed, $k\Phi(n)+1$`in bir formudur.

$$M^{k\Phi(n)+1} = M^{k(p-1)(q-1)+1} \equiv M(\mathrm{mod}\, n) \quad (5)$$

Dolayısıyla, $M^{ed} = \mathrm{mod}(n)$;

$$C = M^e \bmod n \quad (6)$$

$$M = C^d \bmod(n) \equiv (M^e)^d \bmod(n) \equiv M^{ed} \bmod(n) \equiv M(\mathrm{mod} n)$$

olduğu görülür.

3.1.Şifreleme ve Deşifreleme

RSA`da hem şifreleme hem de deşifreleme, tamsayıların tamsayı kuvvetlerini almayı ve mod alma işlemlerini gerektirir. Eğer ilk önce tamsayıların üsleri alınıp, daha sonra mod n ile indirgersek, ara değerler devasa büyüklükte sayılar olurlar. Bu sorunu bir miktar azaltmak için modüler aritmetiğin aşağıdaki özelliğinden yararlanılabilir:

$$\left[(a \bmod n) x (b \bmod n)\right] \bmod n = (axb) \bmod n \quad (7)$$

Bu sayede, ara değerler modül n'e göre indirgenebilir. Bu da hesaplamayı pratik hale getirir. Bir diğer husus, üssün verimidir. Çünkü, RSA' da potansiyel olarak büyük üsler ile işlem yapılır.

Genel olarak a^m değerinin (a ve m birer pozitif tam sayı) hesaplanmak istendiğini varsayalım. Eğer m tamsayısı $b_k, b_{k-1}, ..., b_0$ gibi ikilik sistemde ifade edilecek olursa:

$$m = \sum_{bi \neq 0} 2^i \quad (8)$$

olur. Böylece,

$$a^m = a^{\sum_{bi\neq 0} \gamma} = \prod_{bi \neq 0} a^{(2^i)} \quad (9)$$

$$a^m \bmod n = \left[\prod_{bi\neq 0} a^{(2^i)}\right] \bmod n = \prod_{bi\neq 0} \left[a^{(2^i)} \bmod n\right]$$

eşitlikleri yazılabilir. Bu sonuç sayesinde, a^b modn işlemini hesaplamak üzere aşağıdaki algoritma geliştirilebilir. c değeri aslında gerekli olmasa da son değeri üssün değerine eşit olacağından dolayı açıklayıcı bir niteliktedir.

```
c ← 0; d ← 1
for i ← k downto 0
    do c ← 2 x c
       d ← (d x d)mod n
       if bi=1
          then c ← c+1
               d ← (d x a)mod n
return d
```

3.2. Anahtar Üretimi

Açık anahtarlı şifreleme sisteminin uygulamasından önce, her iki katılımcı anahtar parçalarını üretmelidir. Bu üretim prosedürü aşağıdaki işlevleri içerir.

* İki asal sayı hesaplanması, p ve q
* e ya da d'nin seçilip değerinin hesaplanması.

Öncelikle, p ve q nun seçimi düşünülür çünkü, herhangi bir potansiyel saldırgan n=pq'nun değerini biliyor olacaktır, birtakım özel metodlarla p ve q nun bulunmasını engellemek için, p ve q sayıları yeterince büyük sayılar olmalıdır. Diğer yandan, büyük asal sayıları bulmak için kullanılan yöntem de verimli olmalıdır. p ve q değerlerinin tespitinden sonra bir e değeri seçilmesi ve d değerinin hesaplanması ya da alternatif olarak d değeri seçilerek e değerinin hesaplanmasının ardından, anahtar üretme işlemi tamamlanmış olur. e sayısı seçilirken, bu sayının OBEB($\Phi(n), e$)=1 eşitliğini sağlaması yani $\Phi(n)$ ile aralarında asal olması gerektiği dikkate alınmalıdır. Daha sonra da, $d = e^{-1} \bmod \Phi(n)$ eşitliği kullanılarak d değeri elde edilir.

4. PROGRAMIN YAPISI

Güvenilirliği, yaygın kullanımı ve esnek yapısı göz önüne alındığında RSA algoritmasının geliştirilecek protokolde kullanılması uygun görülmüştür.

Bir A kullanıcısı B kullanıcısına geleneksel şifreleme yöntemi kullanarak bir mesaj göndermek istediğinde, öncelikle her iki kullanıcı da sadece kendilerinin sahip olacağı bir anahtar üzerinde mutabık kalmaları gerekir. Daha sonra A kullanıcısı bu anahtarı mesajı şifrelemek için ve B kullanıcısı ise aynı anahtar ile mesajı deşifrelemek için kullanır. Böyle bir iletişimin güvenli olduğunu söyleyebilmek için iletişim hattını dinleyebilecek üçüncü bir C kullanıcısının şifrelenmiş mesajı ele geçirebilmesine rağmen onu çözümlemesinin mümkün olmaması gerekir. Aşağıdaki şartları yerine getirebilen bir metodun güvenli olduğundan bahsedilebilir:

1. Anahtar oluşturma işlemi makul sayılabilecek derecede kolay olmalıdır,

2. Anahtar yeterince uzun olmalı ve bir üçüncü kişinin tahminini önleyebilecek derecede rasgele seçilmelidir,
3. Şifrelenmiş mesaj, orijinal mesajın bir parçası bilinse dahi çözülenememelidir,
4. Anahtar tamamıyla gizli tutulmalıdır.

Şifreleme yönteminde en önemli problemi, iletişime katılacak N kullanıcı düşünüldüğünde her bir mesaj aktarımı için bir anahtara ihtiyaç duyulacağından N(N-1) anahtarın üretilmesi oluşturur. Kullanıcı sayısının artması, anahtar üretme ve anahtar dağıtım işlemlerini zorlaştırır. Açık anahtarlı şifreleme sisteminin temel olarak yukarıdaki özellikleri barındırması gerekir.

A kullanıcısının B kullanıcısına açık anahtarlı şifreleme yöntemi ile bir mesaj göndermesi, aşağıdaki adımlar izlenilerek yapılmalıdır:

1. A, geleneksel şifreleme metotları yardımı ile rasgele bir anahtar seçer. Bu anahtarı oturum anahtarı olarak kullanılır,
2. A, B' nin açık anahtarını oturum anahtarını şifreleme için kullanır,
3. A, şifrelenmiş oturum anahtarını B' ye gönderir,
4. A, oturum anahtarını kullanarak bir mesajı şifreler ve B' ye gönderir,
5. B özel anahtarını kullanarak A' nın gönderdiği şifrelenmiş oturum anahtarını deşifreleyerek oturum anahtarını elde eder,
6. B şifrelenmiş mesajı oturum anahtarını kullanarak deşifre eder.

Bu algoritma her ne kadar mesajın içeriğinin bir üçüncü kişiden korunmasını sağlasa da mesajın beklenen kişiden yani A kullanıcısından gelip gelmediğini sorgulamamaktadır. Bu duruma göre bir C kullanıcısının da A ile aynı adımları izleyerek B ile konuşması mümkündür. Bunu önlemek amacı ile 2. ve 5. adımlar aşağıdaki gibi modifiye edilmelidir:

2'. A oturum anahtarını önce kendi özel anahtarını ve daha sonra B' nin açık anahtarını kullanarak şifreler.

5'. B önce kendi özel anahtarını ve daha sonra A' nın açık anahtarını kullanarak şifrelenmiş oturum anahtarı bilgisini deşifre ederek oturum anahtarını elde eder.

Bu adımlar takip edildiğinde eğer A' nın özel anahtarı ele geçirilmemişse B mesajın A dan geldiğine emin olacaktır. Bu noktada A kullanıcısının niçin direk B' nin açık anahtarını kullanarak mesajı şifrelemek yerine bir oturum anahtarı kullandığı düşünülebilir. Bunun temel iki nedeni vardır:

1. Açık anahtarlı şifreleme sistemi geleneksel yönteme nazaran daha yavaş kalmaktadır ve tüm mesajın açık anahtar kullanılarak şifrelenmesi ve deşifrelenmesi daha çok zaman alacaktır.
2. İletişimi izleyen üçüncü bir C kullanıcısı orijinal mesajın bir kısmını biliyor olabilir ve daha önce herkese duyurulan açık anahtarın ve bu bilginin yardımı ile orijinal mesajı elde etmesi kolaylaşır.

Tüm şifreleme algoritmalarının bir zayıf noktası vardır ve tabi ki önerilen bu metodun bir istisna oluşturması beklenmemelidir. Bu protokolde gözlenebilecek ilk zayıf nokta oturuma başlarken A kullanıcısının ürettiği oturum anahtarının yeterince rasgele olmaması olabilir.

4.1.RSA Algoritmasının Açıklanması

p ve q birbirinden farklı ve büyük iki asal sayı olsun. Bir tamsayının asal olup olmadığı Fermat teoremi kullanılarak test edilebilir. Bu teorem, her pozitif a tamsayısı ile asallığı sorgulanan sayı arasında aşağıdaki ilişkinin olması gerekliliğini arar:

$$a^{p-1} \equiv 1 \pmod{p} \qquad (10)$$

p tamsayısı bu denklik testinden yeterli miktarda farklı a değeri için geçerse asal olma olasılığı yüksektir denir. p ve q asal sayılarının bu şekilde oluşturulmalarının ardından iki büyük d ve e tamsayıları bulunur. Her ikisinin de (p-1)(q-1) çarpımından küçük ve aşağıdaki eşitliğe uygun olması gerekir.

$$de \equiv 1 \pmod{(p-1)(q-1)} \qquad (11)$$

Bunu gerçekleştirmenin en kolay yolu, e'yi rasgele seçerek OBEB(e, (p-1)(q-1), d ve s değerlerini Euclid algoritması kullanarak hesaplamaktır. Böylece aşağıdaki eşitlik oluşturulabilecektir.

$$de - s(p-1)(q-1) = gcd(e, (p-1)(q-1)) \qquad (12)$$

Bu değerler yardımıyla açık anahtar; (e, pq) ve özel anahtar; (d, pq) oluşturulur. e ve d üs olarak, pq çarpımı ise modül olarak kullanılır. pq çarpımı yani modül açık anahtar ile birlikte duyurulur fakat p ve q değerleri gizli tutulur. Aslında p ve q değerleri şifreleme veya deşifreleme işlemlerinde direk olarak kullanılmamaktadır fakat bu sayıların elde edilmesi ile e ve d değerlerinin bulunması çok kolaydır.

M değeri (0, pq) aralığındaki şifrelenmek istenen mesajı simgelesin. RSA algoritması kullanılarak bu değer (0, pq) aralığındaki şifrelenmiş c değerini aşağıdaki gibi oluşturur.

$$c \equiv m^e \pmod{pq} \qquad (13)$$

Deşifreleme işlemi ise benzer şekilde d sayısı kullanılarak yapılır.

$$m \equiv c^d \pmod{pq} \qquad (14)$$

Şifreleme ve deşifreleme işleminin matematiksel anlamda da birbirinin tersi olduğu görülmektedir ve bu durum aşağıdaki gibi ispatlanabilir.

$$(m^e)^d \equiv (m^d)^e \equiv m^{de} \equiv m \pmod{pq} \qquad (15)$$

d ve e değerleri aşağıda verildiği gibi oluşturulur.

$$de = 1 + s(p-1)(q-1) \quad (16)$$

Eğer p sayısı m sayısını bölemiyorsa Fermat teoremine göre:

$$m^{p-1} \equiv 1 \pmod{p} \quad (17)$$

her iki tarafa s(q-1) ile üs alma ve m ile çarpma işlemleri uygulandığında;

$$m^{de} = m^{1 + s(p-1)(q-1)} \equiv m \pmod{p} \quad (18)$$

elde edilir. Bu eşitlik aynı zamanda m'nin p sayısında bölünebildiği durumlarda da geçerlidir. Benzer şekilde:

$$m^{de} \equiv m \pmod{q} \quad (19)$$

Dolayısı ile hem p'nin hem de q'nun (m^{de}–m)' i böldüğü anlaşılır. p ve q'nun asal olduğu bilindiğinden:

$$m^{de} \equiv m \pmod{pq} \quad (20)$$

yazılabilir.

5. RSA ALGORİTMASININ C++ İLE UYGULAMASI

Bu bölümde, RSA algoritmasının; Microsoft Visual C++ 6.0 yardımı ile Windows işletim sistemi altında herhangi bir hazır şifreleme kütüphanesi kullanmaksızın uygulaması anlatılacaktır. Daha öncede açıklandığı gibi RSA açık anahtarlı şifreleme algoritması etkisini, kullandığı anahtarların uzunluğundan almaktadır. Bu bakımdan RSA için en temel ihtiyacın büyük tamsayılarla modüler aritmetik işlemleri yapabilmek olduğu söylenebilir. Fakat günümüzde bilgisayar sistemlerinde kullanılan işlemcilerin yazmaçları (register) dahi algoritmayı karşılayacak uzunlukta tamsayıları donanımsal anlamda destekleyemezler. Kullanılan derleyicinin ve çalışılan işlemcinin 32 bitlik işlemler için tasarlandığı düşünülürse, daha büyük tamsayılar oluşturmak ve bu tamsayılarla modüler işlemler gerçekleştirmek için yazılımsal desteğe ihtiyaç olacaktır.

Bu ihtiyacı karşılamak amacıyla büyük sayıları oluşturmayı ve işlem yapabilmeyi sağlayacak bir C++ sınıfı yazılmıştır. Öncelikle "large_int" ismi verilen bu sınıfın ayrıntılarıyla anlatımı sunulacaktır.

5.1. Büyük Sayı Sınıfının Açıklanması

Bu sınıf, yapı itibariyle işaretsiz 16 bitlik tamsayılardan oluşturduğu kümenin tek bir tamsayıymış gibi işlemlere girmesini sağlayarak; lüzumlu algoritmaların işlenmesine olanak verecek şekilde yazılmıştır. Dolayısı ile sınıfın temel birimi "**unsigned short *value;"** şeklinde tanımlanan değer (value) göstericisidir. İhtiyaç duyulan büyüklükte bir sayı oluşturabilmek için 16 bit cinsinden uzunluğunun tutulacağı "**unsigned int uzunluk;"** sınıfın ikinci önemli değişkenidir denebilir.

Bu sınıfa ait bir nesne oluşturulurken kullanılacak iki farklı kurucu fonksiyondan ilki oluşturulacak sayının uzunluğunu, 16 bitlik bir tamsayının katı olarak beklerken diğeri ise yine bu sınıftan bir tam katı olarak verilmesini bekler. İşlemin algoritma yazılımı:

```
large_int::large_int(unsigned len){
uzunluk = len;
value = new unsigned short(len); }
large_int::large_int(const large_int &n) {
uzunluk = n.uzunluk;
value = new unsigned short(uzunluk);
unsigned i;
for (i = 0; i < uzunluk; i++)
value(i) = n.value(i); }
```

gibi açıklanır. Verilen uzunluk değeri, linkler tarafından uygun kurucu fonksiyona gönderilerek gerekli hafıza alanının ayrılmasında kullanılır.

Ayrıca, büyük sayılarda özellikle çarpma işlemlerinde kullanılmak üzere sayıyı verilen bit uzunluğunda sola kaydırmaya yarayan shift fonksiyonu tanımlanmıştır.

Sınıfta, büyük sayıların birbirleriyle işlemlerini sağlayacak hemen hemen tüm matematiksel operatörlerin aşırı yüklemeleri de tanımlanmıştır. Bu sayede programın anlaşılabilirliği arttırılmıştır. Programda özellikle mod alma işlemlerinde çokça kullanılan "%" operatörünün de diğer aşırı yüklenmiş operatörler gibi hem 16 bitlik değerler hem de büyük sayı cinsinden nesneler için işlem yapabilecek iki farklı aşırı yüklenmiş fonksiyonu vardır.

RSA algoritması için en çok kullanılması gereken işlemlerden biri de üs alma işlemi olacağından sınıfa, büyük sayılar için üs alma işlemini gerçekleştirecek "power" fonksiyonu da eklenmiştir. Bu fonksiyonun tüm parametreleri büyük sayı cinsinden verildiğinden yapılacak üs alma işleminin uzunluk sınırlaması bulunmamaktadır. Bu yazılım ise:

```
void large_int::Power(const large_int &base,
const large_int &exponent,
const large_int &modulus, large_int &result){
large_int r(2*base.uzunluk+1);
r = 1;
bool one = true;
unsigned i = exponent.uzunluk;
while (i-- != 0) {
unsigned bit = 1 << 15;
do {
if (!one) {
large_int n(r);
r *= n;
```

```
r %= modulus;     }
if (exponent.value(i) & bit)     {
r *= base;
r %= modulus;
one = false;     }
bit >>= 1;
} while (bit != 0);
}
result = r;}
```

şeklindedir.

Yukarıda gösterildiği gibi bu fonksiyon tamamen RSA için özelleştirilerek verilen taban, üs ve modül değerlerine göre yaptığı hesaplamaları, sonuç büyük sayı değişkenine atamaktadır.

Yapılacak işlemler için uzunluğu taban uzunluğunun iki katından bir fazla olan r büyük sayı değişkeni oluşturulmuştur. İşlemin kabarıklığının önlenmesi amacıyla modüler aritmetiğin dağılma özelliğinden yararlanılmış ve bu sayede tüm üs alma işlemini yaptıktan sonra modül alma işlemi yapmak yerine her bir üs alma işleminden sonra modül alma işlemi yapılarak sayının çok fazla büyümesi önlenmiştir. İkinci önemli nokta, üs alma işleminin tekrarlı kare alma yöntemi ile basitleştirilmesidir. Bunu yapabilmek için üssün bitleri en değerli bitten başlamak üzere test edilir. Her bir 2' nin tam üssü için tabanın karesi alınırken sonuçta artan değerler kadar taban değerinin sonuçla çarpımı yapılır. Böylece sadece artan değerler için normal üs alma işlemi yapılmış olur.

5.2. RSA Fonksiyonlarının Açıklanması

Öncelikle şifreleme için gerekli anahtarların üretilmesi daha sonra bu anahtarlar kullanılarak bir mesajın şifrelenebilmesi için kullanılan fonksiyonlar anlatılacaktır. Anahtar üretimi için ihtiyaç duyulacak ilk şey asal sayıların üretimi olacaktır. Bunun için öncelikle rasgele bir sayı seçilmekte daha sonra bu sayının komşuluklarında asal bir sayı aranmaktadır. Bulunan sayıların asal olup olmadığının test edilebilmesi için kullanılan AsalTest() fonksiyonu:

```
static bool AsalTest(const large_int &p){
large_int pminus1(p);
pminus1 -= 1;
unsigned count = 101;
while (--count != 0)  {
large_int r(p.uzunluk);
large_int x(p.uzunluk);    {
for (unsigned i = 0; i < x.uzunluk; i++)
x.value(i) = rand() << 8 | rand();    }
x %= p;
if (x != 0)    {
large_int::Power(x, pminus1, p, r);
if (r != 1)
return false;    }  }
return true;}
```

gibi açıklanabilir. Fonksiyonda Fermat teoremi uygulanmaktadır. Bu fonksiyonda bir sayının asal sayılabilmesi için asallık testine 100 kez girmesi gerekmektedir. AsalTest fonksiyonu anahtarların üretimleri sırasında AsalUret ve AsalUret fonksiyonunu kullanan AnahtarUret fonksiyonları tarafından çağırılmaktadır.

AsalUret fonksiyonu parametre olarak aldığı büyük sayı türünden değişkene önce rasgele bir değer atar. Daha sonra AsalTest fonksiyonunu çağırarak bu rasgele sayının asallığını test eder. Çift sayıların asal olması mümkün olmayacağından sadece tek sayılar için deneme yapılır. Bu bakımdan asallığı AsalTest fonksiyonu tarafından onaylanmayan sayılar iki arttırılarak tekrar teste tabi tutulur. Bu şekilde asal bir sayı bulununcaya kadar döngüye devam edilir.

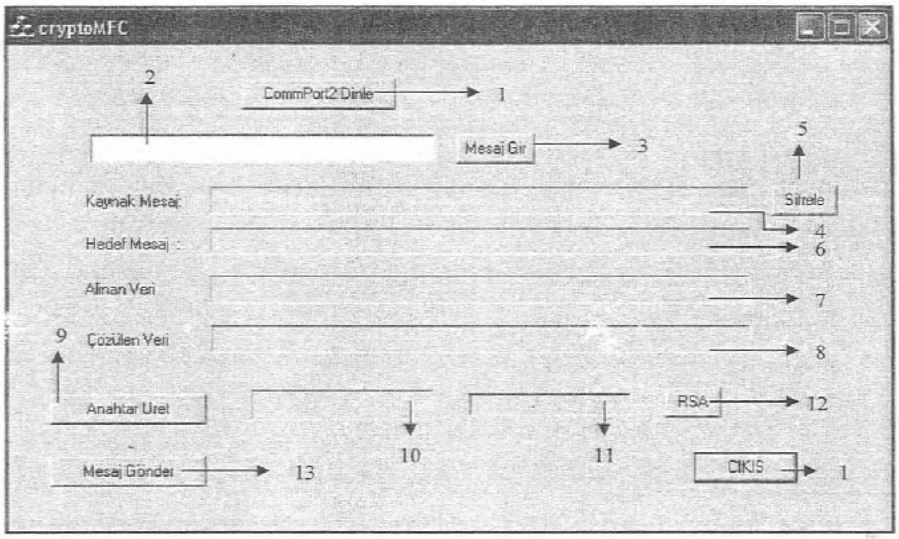
Anahtar üretimi sırasında karşılaşılan çok önemli bir başka husus aralarında asal sayıların bulunmasıdır. Hatırlanacağı gibi d ve e sabitlerinin Φ ile aralarında asal olacak şekilde seçilmeleri gerekiyordu. Bu durumun kontrolü ise Euclid algoritmasıyla sağlanır. Bu denetleme, programda EuclideanAlgorithm fonksiyonu tarafından yapılmakta (5) ve anahtar üretimi sırasında AnahtarUret fonksiyonu tarafından çağırılarak kullanılmaktadır. Böylece girdi olarak aldığı iki büyük sayının Ortak Bölenlerinin en Büyüğü' nü bularak bir diğer büyük sayı olan g'ye atamaktadır.

6. PROGRAMIN ÇALIŞMASI

Tasarlanan protokolün Visual Studio .NET C++ 7.0 ile yazılan uygulaması, seri port üzerinden yarattığı iletişim kanalı, simülasyon yapmak için kullanılmaktadır. Programın tam anlamı ile çalışabilmesi için Windows 9x, Me, XP veya 2000 tabanlı iki adet seri (COMM) kanala sahip bir bilgisayar kullanılmalıdır. 1 numaralı seri iletişim kanalı protokole uygun olarak hazırlanan veriyi göndermek için kullanılırken, 2 numaralı seri iletişim kanalı bu veriyi almak için kullanılır. Anlaşılacağı üzere program, aslında bir host ve bir çok slave cihazdan oluşabilecek protokolun simulasyonunu tek bir bilgisayar üzerinde yapabilmesine imkan sağlayacak şekilde düşünülmüştür

Program çalıştırıldığında Şekil 2'de görülen ana arabirim ile açılır. Şekil 2'de numaralandırılarak gösterilen elemanların açıklamaları aşağıda verilmiştir:

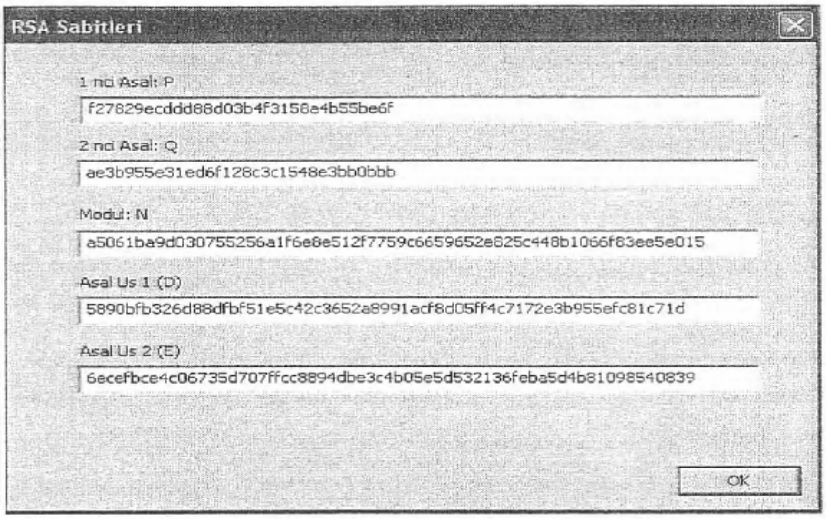
1. 2 numaralı iletişim protokolünün, program için rezerv edilmesini sağlayan tuştur. Hazırlanan verinin 1 numaralı kanal üzerinden gönderilmesinden evvel bu tuş ile COMM 2 kanalı hazır hale getirilmelidir.
2. Kullanıcının giriş yapması için yerleştirilen düzenleme kutusu. Girilen mesaj en fazla



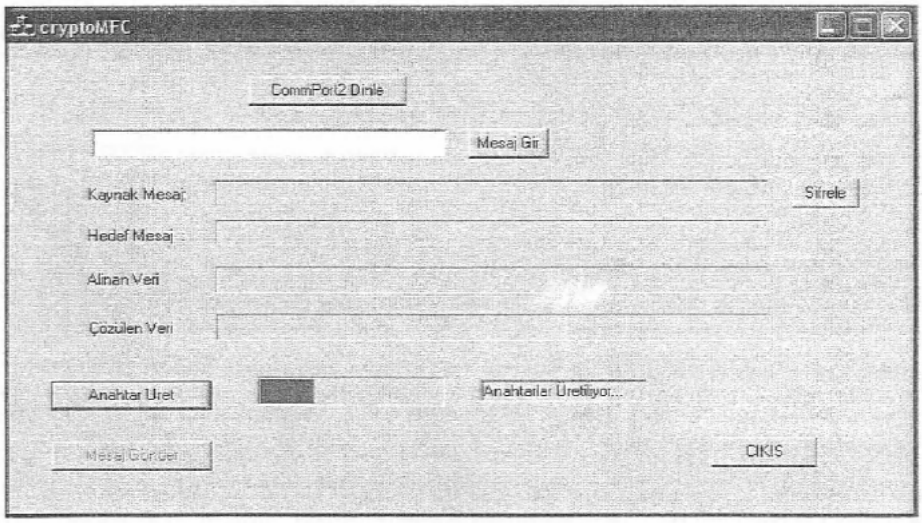
Şekil 2. Programın ana arabirim ekran görüntüsü

anahtar uzunluğu kadar olmalıdır (256 bit için 32 karakter).

3. Düzenleme kutusuna girilen mesajın protokole girilmesini sağlayan tuş.
4. Düzenleme kutusunda hazırlanarak "Mesaj Gir" tuşu ile protokole girilen mesajın kaynak mesaj olarak gösterilmesini sağlayan etiket kutusu.
5. Girilen mesajın RSA algoritmasına uygun olarak şifrelenerek gönderilmeye hazır hale getirilmesini sağlayan tuş. Bu tuş kullanılmadan önce RSA şifreleme için gereken anahtarların üretilmesi gerekir.
6. Şifrelenerek gönderilmeye hazır hale getirilen mesajın son halini gösteren etiket kutusu.
7. 2. iletişim kanalından alınan verinin gösterildiği etiket kutusu.
8. Alınan verinin deşifreleme işleminin ardından elde edilen mesajın gösterildiği etiket kutusu.
9. RSA algoritmasının uygulanabilmesi için ihtiyaç duyulan anahtar değerlerinin üretilmesini sağlayan tuş. Yeni bir iletişim oturumu gerektiren durumlar için anahtarlar yeniden üretilmelidir.
10. Anahtar üretme işleminin hangi aşamada olduğunun anlaşılmasını sağlayan durum gösterici.
11. Anahtar üretme işleminin fazlarını kullanıcıya bildiren etiket kutusu.
12. "Anahtar Üret" tuşu yardımıyla oluşturulan anahtar değerlerinin ve diğer RSA sabitlerinin yer aldığı RSA Sabitleri diyalog kutusunun açılmasını sağlayan tuş. Bu tuş şifreleme



Şekil 3. RSA sabitleri diyalog kutusu ekran görüntüsü



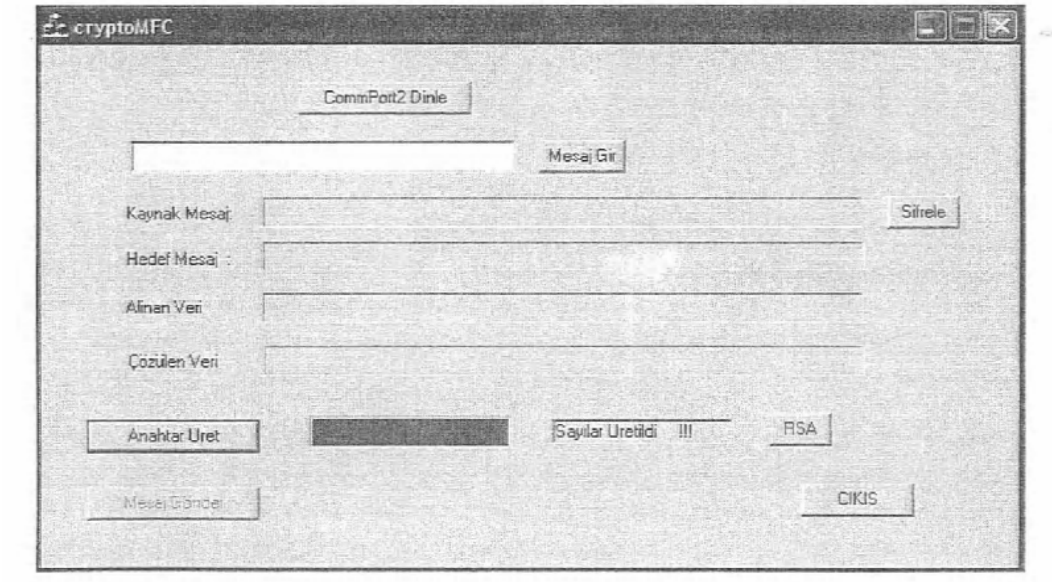
Şekil 4. Anahtarların üretilmesi sırasında programın ekran görüntüsü.

işlemlerinin gerçekleştirilebilmesi için gerekli olan anahtarlar üretilinceye kadar görünmeyecektir.

13. Şifrelenmiş yerinin 1 numaralı iletişim kanalı üzerinden gönderilmesini sağlayan tuş. Bu tuş, başarılı bir şifreleme işlemi yapılana kadar erişilebilir değildir.

mak için 9 numara ile gösterilen "Anahtar Üret" tuşu kullanılmalıdır.

Öncelikle, daha önce oluşturulan RSA sabitleri geçersiz olduğu için RSA butonu gizlenmektedir. Ardından işlemin başladığının kullanıcıya bildirilmesi amacıyla durum gösterici bir aşama ilerletilmiş ve "Anahtarlar Üretiliyor" mesajı gösterilmiştir.



Şekil 5. Anahtar üretme işleminin tamamlanmasının ardından programın ekran görüntüsü

RSA sabitlerinin gösterildiği diyalog kutusu Şekil 3'de görülmektedir.

6.1.Anahtarların Üretilmesi

Şifreleme ve Deşifreleme işlemlerinin gerçekleştirilebilmesi için öncelikle genel ve özel anahtarların üretilmesi gerekir (Şekil 4). Programda bu işlemi yap-

Daha önce RSA şifrelemede anlatıldığı gibi öncelikle p ve q asal sayılarının bulunması gereklidir. Sonrasında bu sayılar kullanılarak anahtar üretimi tamamlanır.

Tüm bu işlemlerin tamamlanması ile gerekli RSA sabitleri ve anahtarlar üretilmiş olur. Bu aşamada program Şekil 5'te gösterildiği biçimde görülür.

RSA sabitlerinin başarı ile üretilmesinin ardından RSA butonu aktif hale gelecektir. Bu buton kullanılarak Şekil 3'te gösterilen RSA Sabitleri Diyalog kutusu açılarak üretilen sabitler görüntelenebilir.

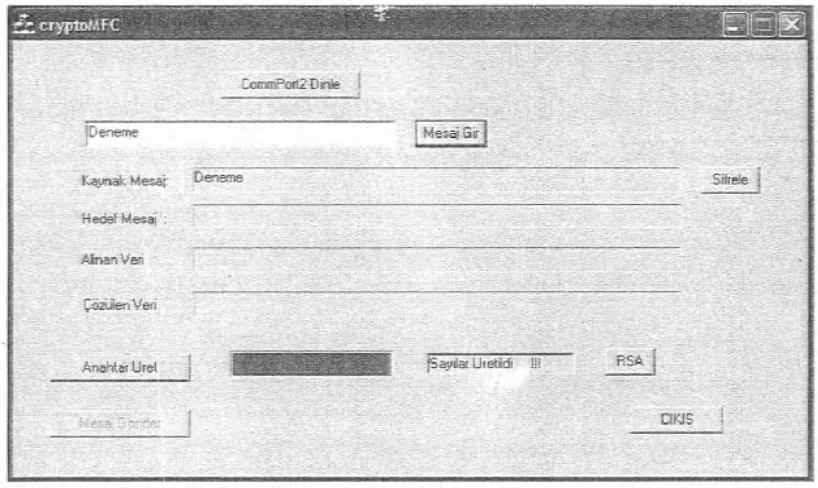
6.2. Gönderilecek Mesajin Programa Girilmesi ve Şifrelenmesi

Şifreleme işlemleri için gereken anahtarların o1uşturulmasının ardından gönderilmek istenen mesaj veya komut programa girilmelidir. Bunun için Şekil 2'de "2" ile gösterilen düzenleme kutusuna mesaj/komut yazılmalı ve "3" ile gösterilen buton kullanılarak programa girilmelidir. Bu işlemin ardından girilen mesaj, program tarafından kaynak mesaj olarak kabul edilecektir. Programın ekran göröntüsü ise Şekil 6'da gösteri1diği gibi olmalıdır.

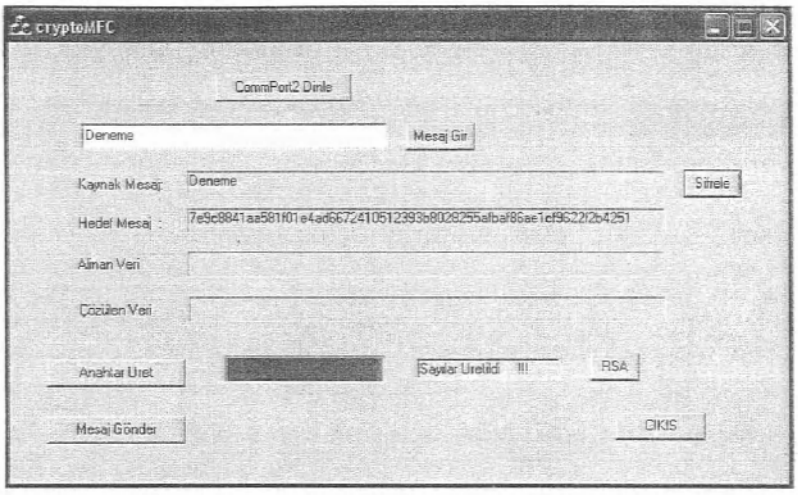
Girilen mesajın şifrelenerek gönderilmeye hazırlanması ancak bu aşamada mümkündür. Dolayısı ile Şekil 2'de 5 ile numaralanan "Şifrele" butonu kullanılarak mesaj şifrelenir. Şekil 7'de programın bu adımdaki ekran görüntüsü verilmektedir.

6.3. İletişim Kanallarının Hazırlanması

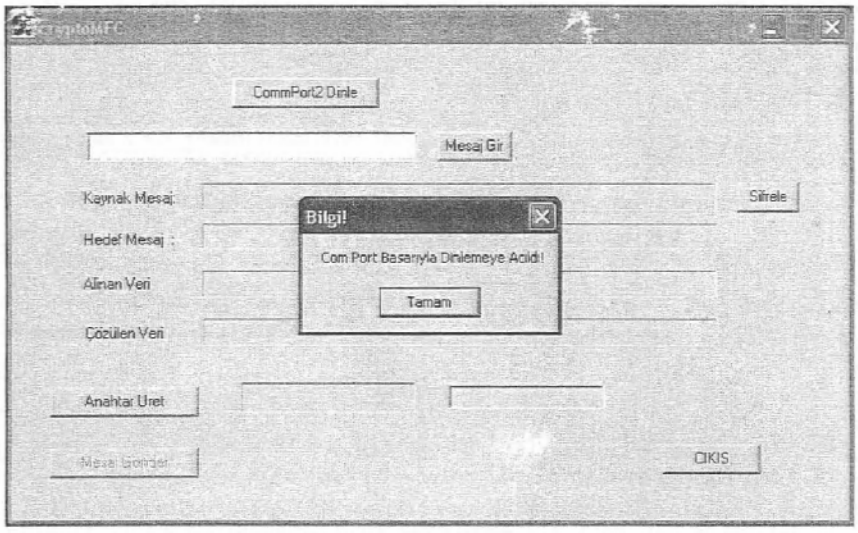
Daha önce, programın mesaji göndermek için 1 numaralı i1etişim kanalını, mesajı almak için ise 2 numaralı iletişim kanalını kullandığı belirtilmişti. Programın bir kanaldan gönderdiği şifreli mesajı, diğer kanaldan tekrar alarak deşifrelemesi tamamen simülasyon amaçlı olmasından dolayı; 1 numaralı iletişim kanalı programın yüklenmesiyle birlikte hazır hale getirilirken, 2 numaralı iletişim kanalının açılması kullanıcı tarafından yapılacak şekilde tasarlanmıştır. Dolayısıyla simülasyon işlemine başlamadan önce, Sekil 2'de 1 ile numaralanan buton kullanılarak 2 numaralı iletişim kanalı hazırlanmalıdır. Şekil 8'de bu aşamada programın ekran görüntüsü verilmektedir.



Şekil 6. Gönderilecek mesajin programa girilmesi ardında ekran görüntüsü

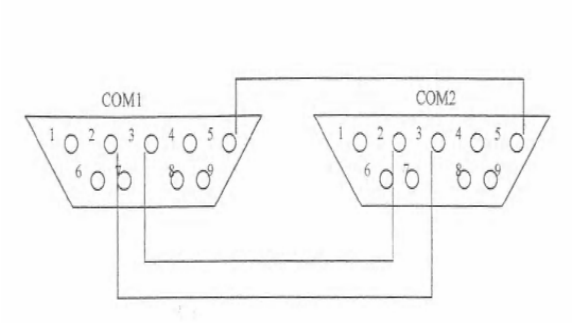


Şekil 7. Girilen mesajın şifrelenmesi sonrası programın ekran görüntüsü



Şekil 8. 2 numaralı iletişim kanalı hazırlandığında programın ekran görüntüsü

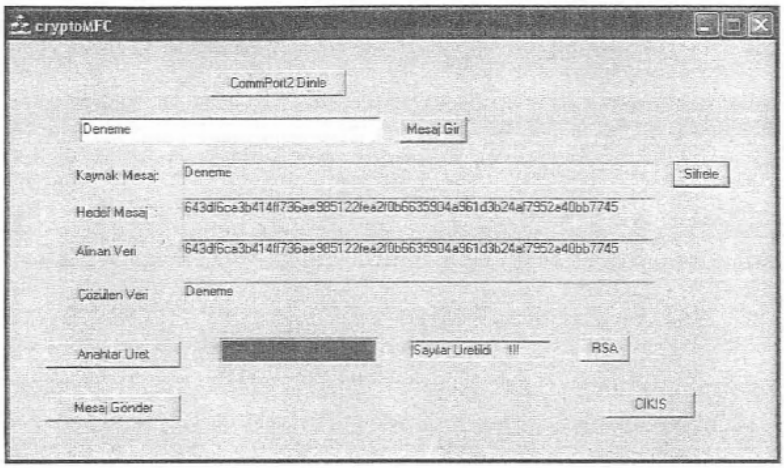
Simülasyon işleminin doğru bir şekilde yapılabilmesi için yazılımla yapılacak ayarlamaların yanında donanımsal bir bağlantıya da ihtiyaç vardır. Yapılması gereken 1 ve 2 numaralı COMM portların TX- Verici kanalının RX -Alıcı kanalı ile bağlanmasıdır (Şekil 9).



Şekil 9. Seri iletişim kanallarının simülasyon için hazırlanması

6.5. Şifreli Mesajın Gönderilmesi

Bir önceki bölümde anlatıldığı şekilde hazırlanan işletişim kanallarının üzerinden simülasyon işlemine başlamak artık mümkündür. Bu amaçla Şekil 2'de 13 ile numaralandırılan buton kullanılarak şifrelenen mesajın gönderilmesi ve Şekil 10'da belirtilen ekran görüntüsünün gözlenmesi beklenmelidir.



Şekil 10. Simülasyon sonrası şifrelenen verinin alınıp deşifrelenmesi sonrası ekran göruntüsü

7. SONUÇ

Bu çalışmada, veri iletişiminde güvenlik, iletişime giren tarafların kimliklerinin doğruluğu ve iletişim hattında kasıtlı olarak oluşturulabilecek veya gürültüden kaynaklanabilecek bozulmalar ile belirtilebilecek söz konusu hata ve eksiklerin giderildiği bir protokol modeli sunulmaya çalışılmıştır.

Sunulan bu protokolün güvenlik, kimlik doğrulama ihtiyaçları için açık anahtarlı şifreleme algoritmalarının en bilinenlerinden olan RSA algoritması ile çözülmesi öngörülmüştür.

8. KAYNAKLAR

1. Diffie, W. And Hellman, M.E.,"New Directions in Cryptography", IEEE Trans. Inform. Theory, 22(6) 644-654, 1976.
2. Nguyen, H., Aziz, S.M., Ryan, T., "A Low Bandwidth Communication Protocol for Remote Control Applications", IEEE Region 10 Conferernce, C: 37-40, 2004.
3. Wolisz, A., Schieferdecker, I., Walch, M., " An Integrated Approach to the Design of Communication Protocols", Distributed Computing Systems, 19: 411-418, 1993.
4. Rivest, R. L., Shamir, A., Adleman, L.A., " A method for Obtaining Digital Signatures and Public-Key Cryptosystem", Communications of the ACM, 21 (2): 120-126, 1978.
5. Bölücek, A.Ç., Akçam N., " Güvenilir, Hata Düzeltmeli ve Dijital İmzalı Protokol Geliştirme", Ulusal Elektronik İMZA Sempozyumu, 7-8 Aralık 2006.