İSTANBUL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS

J2ME TABANLI MOBİL E-TİCARETTE NOKTADAN-NOKTAYA GÜVENLİK

Ilgar Mammadov

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Danışman

Yrd.Doç.Dr. Mustafa DAĞTEKİN

İSTANBUL

Bu çalışma 16/08/2010 tarihinde aşağıdaki jüri tarafından Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği programında Yüksek Lisans Tezi olarak kabul edilmiştir.

Tez Jürisi

Danışman Adı :
(Danışman)
Y. Doç. Dr. Mustafa DAĞTEKİN

İstanbul Üniversitesi
Bilgisayar Mühendisliği

Jüri Adı:

Prof. Dr. Sabri ARIK

İstanbul Üniversitesi
Bilgisayar Mühendisliği

Jüri Adı:

Y. Doç. Dr. Oğuzhan ÖZTAŞ

İstanbul Üniversitesi
Bilgisayar Mühendisliği

Jüri Adı:

Prof. Dr. Ahmet SERTBAŞ

İstanbul Üniversitesi
Bilgisayar Mühendisliği

Jüri Adı:

Y. Doç. Dr. Fırat KAÇAR

İstanbul Üniversitesi
Elektrik-Elektronik Mühendisliği

ÖNSÖZ

Yüksek lisans öğrenimim sırasında ve tez çalışmalarım boyunca gösterdiği her türlü destek ve yardımdan dolayı çok değerli hocam Yrd.Doç.Dr. Mustafa DAĞTEKİN'e içten dileklerimle teşekkür ederim.

Bu çalışma boyunca yardımlarını esirgemeyen çalışma arkadaşlarıma teşekkürü borç bilirim.

Temmuz, 2010 **Ilgar Mammadov**

İÇİNDEKİLER

ŞEKİL LİSTESİ

KISALTMA LİSTESİ

Kısaltma Adı	İngilizce	Türkçe
RSA	The RSA Pubic Key Cryptosystem	RSA Açık anahtarlı şifreleme
SDK	Software Development Kit	Yazılım geliştirme aracı
SSL	Secure Socket Layers	Güvenli Soket Katmanları
TCP/IP	Transmission Control Protocol/Internet Protocol	İletim Kontrol Protokolü / Internet Protokolü
TLS	Transport Layer Security	Aktarım Katmanı Güvenliği
UMTS	Universal Mobile Telecommunication Service	Evrensel Mobil Telekomünikasyon Hizmetleri
WAP	Wireless Application Protocol	Kablosuz Uygulama Protokolü
WIM	Wireless Identity Module. Usually associated with SIM	Kablosuz Kimlik Modülü. Genellikle SIM ile ilişkili
WTLS	Wireless Transport Layer Security	Kablosuz Aktarım Katmanı Güvenliği
JSR	Java Specification Request	Java Özellik İsteği
KVM	Kilo Virtual Machine. The java virtual machine for mobile devices.	Kilo Sanal Makinesi. Mobil cihazlar için java sanal makinesi.
MIDlet	The J2ME™ Application.	J2ME ™ Uygulama
MIDlet SUITE	A bundle of MIDlets in the same application.	Aynı uygulamada olan MIDlet'ler paketi
MIDP	Mobile information Device Profile	Mobil Bilgi Cihaz Profili
CLDC	Connected Limited Device Configuration	Bağlantılı Sınırlı Cihaz Konfigürasyonu
J2ME	Java 2 Micro Edition	Java 2 Mikro Sürümü

ÖZET

J2ME TABANLI MOBİL E-TİCARETTE NOKTADAN-NOKTAYA GÜVENLİK

Bu tezde MIDP tabanlı J2ME ile geliştirilmiş güvenli bir uygulama geliştirmek hedefleniyor. Bu uygulama, mobil ortamda güvenli iş gerektiren çeşitli hizmetler için uygulanabilir. Amaç, güvenli ve hızlı mobil iş sağlamaktır. Kablosuz kurumsal uygulamalar için uçtan-uca uygulama katmanında güvenli uygulama Java 2 Platform Micro Edition kullanarak gerçekleştirdildi. Önerilen çözüm, saf Java bileşenlerini kullanarak J2EE sunucuları ve J2ME istemcileri arasında uçtan-uca veri güvenliğini sağlamaktır. Bu çözüm, Java MIDP cihazın sınırlı kaynakları ile mevcut olan protokol veya kablosuz ağ altyapısı için herhangi bir değişiklik yapmadan uygulanabilir. Önerilen çözüm için örnek bir uygulama geliştirildi. Bu çalışmada kullanılan yöntemin, benzer diğer çalışmalardan üstün yönü, kullandığım şifreleme algoritması çok daha hızlı ve kolay uygulanabilirliğidir.

SUMMARY

END-TO-END SECURITY IN J2ME BASED MOBILE E-COMMERCE

In this thesis, the proposed secured application is constructed by J2ME based MIDP. This application can be applied to various services that require secure business in mobile environment. It is intended to provide more secure and fast mobile business. An end-to-end application-layer security solution is implemented for wireless enterprise applications using the Java 2 Platform Micro Edition (J2ME). The proposed solution uses pure Java components to provide end-to-end data confidentiality between wireless J2ME based clients and J2EE based servers. This solution can be implemented with the available limited resources of a Java MIDP device, without any modification to the underlying protocols or wireless network infrastructure. An application is used to illustrate the implementation of the proposed solution. The advantage of the method used in this study to the other similar studies is the applicability encryption algorithm is much faster and easier.

1. GİRİŞ

Günümüzde mobil aygıtlar birçok alanda örnek olarak resim çekmede, televizyon izlemede, doküman oluşturmada, elektronik posta kullanımında vs. kullanılıyorlar. Aynı zamanda internetin hızla yayılması ve hızının artması geleneksel e-ticareti m-ticarete (mobil ticaret) uygulamada kolay imkânlar sunuyor.

Mobil-ticaretin her zaman ve her yerden erişilebilir olmasından dolayı avantajları çoktur. Bu kadar çok verinin taşındığı ortamda veri güvenliği geçmişten günümüze kadar hep önemli olmuştur. Mobil ticarette de dolayısıyla veri güvenliği çok önemlidir. Bu bakımdan mobil aygıtla servis sunucusu arasındaki tüm iletişim kanalları çok iyi korunmalıdır.

Fakat araştırmalar zamanı mobil iletişimde bazı güvenlik açıkları ortaya çıkıyor. Bu açıklardan en önemlisi ilerde ayrıntılarıyla anlatacağımız "Wap Gap" denilen fenomendir. Bu soruna WAP yeni teknolojilerde çözümler geliştirmiş ama şirketler özel verilerinin güvenliğin korumak için kendi geliştirdikleri mobil uygulamalar ile uçtan-uca güvenliği sağlıyorlar.

Geçmişten günümüze kadar bu konuda birçok uygulamalar geliştirilmiştir ve bunların birbirlerine göre avantajları ve dezavantajları vardır. Şimdi bu konuda şimdiye kadar yapılmış araştırma ve uygulamaları değerlendirelim.

Wassim İtanı and Ayman I. Kayssi yaptığı araştırmada Java 2 Platform Micro Edition (J2ME) kullanarak mobil kullanıcı ve banka sunucusu arasında uygulama katmanında uçtan uca güvenlik geliştirmek için AES algoritmasını kullanarak uygulama geliştirmişler. Nedenini, herkesin bu problem için kendi uygulamalarını geliştirmesi gerektiğini açıklamışlar. "Wap gap" problemine çözüm için WAP 2.0 da yeni standartlar geliştirilmiş ama bu standartlarda tüm mobil aygıtlar TCP protokolünü kullanmak zorundalar. Bu da bazı kısıtlamalara yol açıyor, çünkü her zaman cihazlar TCP protokolünü desteklemiyor olabilir. Önerdikleri çözüm uygulama katmanında tüm

güvenlikle ilgili problemlere daha aşağıdaki katmanları dikkate almadan destek oluyor. Önerilerini mobil bankacılık uygulaması üzerinde gerçekleştirerek açıklamışlar. Aynı zamanda oturum izleme ve istemci yetkilendirmeyi desteklemişler [1].

Bringel Filho ve arkadaşları yaptıkları araştırmada, kablosuz aygıtlar arasında veri iletişiminde verilerin güvenliğinin çok önemli olduğuna dikkat çekmişler. Burada veri yolunun güvenliği tam sağlansa da uygulama bazında veri güvenliği sağlanamamaktadır. Bu yüzden uygulama güvenliği sağlamak için daha yüksek katmanlarda (OSI katmanları) uçtan-uca güvenlik geliştirmek zorunludur. Bunu da en iyi geliştirmek ortamı J2ME'dir. Onlar bunun için bir yapı öneriyorlar ve kriptografik algoritmalar kullanarak bunu gerçekleştiriyorlar [2].

Anders Cervera kendi yaptığı tezinde mobil ödemeler için gerekli teknolojiler ve önemli başarım faktörlerini anlatmış. Son kullanıcı ve mobil aygıt arasında veri iletişimi üzerinde durmuş. Yaygın olarak kullanılan mobil ödeme sistemlerini karşılaştırarak avantajlarını anlatmış. Genel bir mobil ödeme sistemi nasıl olması gerek ve hangi prosedürlerden oluşması gerektiğini tartışmıştır[3].

Miguel Soriano ve Diego Ponce'ye göre kablosuz ortamlar için uygulama geliştirmenin bir çok zorlukları vardır. Bunlar bant genişliğinin az olması, gecikmeler, düşük CPU, bellek yetersizliği ve güç kaynağıdır. Bunları dikkate alarak kablosuz aygıtlar için uçtan-uca güvenlikli bir veri iletişimi geliştirmiş. Genel olarak herkes bu uygulamalarda RSA şifreleme yöntemi kullanmış, ama onlar yukarıdaki kısıtlamaları göz önünde bulundurarak Certicomun Elliptical Curve şifreleme yöntemini tercih etmişler [4].

Sujata Banerjee ve arkadaşları yaptıkları araştırmada, telefonlarda e-ticaret işlemlerinin artmasıyla beraber yüksek düzeyde güvenliğin sağlanılması gerektiğine dikkat çekmişler. Telefonla yapılan her işlem için güvenliğin sağlanması gerekmektedir. İşlemlerdeki güvenlik gerekliliği direk olarak işlemlerin değeri, duyarlılığı ve hacmi ile doğru orantılıdır. Bu karakteristiklerin hepsi nakit yönetimi kısmında sağlanmaktadır. Gerçekten nakit yönetiminde bu işlemlerin güvenliği çok önemlidir ve işlem güvenliği nakit yönetim kanalına bağımlı 3 zayıf yönü vardır:

1- Mobil aygıt
2- Kablosuz kanal
3- Web sunucusu ile arka uç işlem sunucu arasındaki ağ birleşimi

Mobil aygıt

Mobil aygıt genelde güvenlik zincirinin en zayıf halkasıdır. Burada olabilecek problemler virüs olarak adlandırdığımız zararlı kodların varlığından ortaya çıkmaktadır. Bu virüsler girdikleri işletim sistemlerine yayılarak çok ciddi güvenlik sorunlarına sebep olabilmektedirler.
Ayrıca bu zararlı kodlar MMS, yani, Multimedya Mesaj'dan de bulaşabilir ve mesajlaşma servislerine de zarar verebilir. Mobil aygıtın kaybedilmesi de ciddi güvenlik açıklarına sebep olabilmektedir.
Burada esas olarak anlatılmak istenen nokta kullanıcıların farkında olması gereken çok önemli güvenlik kuralları vardır ve herkes bunlara uymalıdır.

Kablosuz kanal

Kablosuz kanalının güvenliği bakış açısına göre değişmektedir. Eğer siz kampüsün her tarafından erişmek isterseniz mecburen kablosuz kanalı kullanmak gerekmektedir ve eğer kablosuz kanalı kullanıyorsanız o zaman kimlik doğrulama, şifreleme ve bunun gibi teknolojiler kullanılmak zorunludur. WEP, çok duyarlı veriler kablosuz kanal üzerinden transfer edilirken pek güvenilir sayılmaz. Geçmişten bugüne WEP çok defa kırılmıştır, onun için IEEE 802.11 standartlarına geçmek gerekmektedir. WI-FI korumalı erişim ile WEP`in çok önemli bazı problemleri çözülmüştür.

GPRS, 3G, GSM iyi seviye veri şifreleme ve dolayısıyla güvenliği sağlamaktadır. Ama finansal işlemler için daha yüksek düzeyde güvenlik gerekmekte ve eğer aygıtlar için daha fazla güvenlik isteniyorsa, sim karta araçlar eklenerek bunlar sağlanabilmektedir. Eğer genelleyecek olursak kablosuz kanal kullanırken rahat olmak için yapılandırma ayarlarının doğru yapılması, gerekli güncellemelerin yapılması, veri iletişimi sırasında SSL portunu kullanılması ve gerekli şifrelemelerin yapılması gerekmektedir.

Web sunucusu ile arka uç işlem sunucu arasındaki ağ birleşimi

Finansal işlemler sırasında kablosuz servis sağlayıcı işlemleri çok önemlidir. Bu işlemler sırasında WAP geçiti doğru ayarlanması sistemi güvenlik açıklarından korumaktadır. Ayrıca eğer sistemi daha güçlü yapılandırmak istiyorsak sunucu taraf ile bağlantı sırasında VPN (Virtual Private Network) kullanmak gerekmektedir [5].

Vipul Gupta ve Sumit Gupta'ya göre mobil ticarette güvenlik çok önemlidir. Yüksek güvenlikli veri iletişimi için uçtan uca güvenliğe ihtiyaç var. Bunun için de kendi güvenlik uygulamalarını geliştirmişler. Bu uygulamayı geliştirirken performansa önem vermişler ve önbellek yöntemlerini kullanmışlar. Bu da düşük bellekli aygıtlar için sorun olsa da, daha yüksek bellekli aygıtlar için iyi performans sağlıyor. Normal ssl uygulamalarında tepki süresi 5s, onların geliştirdiği uygulamada tepki süresini 8s kadar indirmişler [6].

Sheueling Chang ve arkadaşlarına göre şu anki kriptografik algortimalar için en iyi 2 örnek ECC ve RSA'dir. Bunlar arasında hangisinin iyi olduğuna karar vermek için onları, farklı özellikleri olan cihazlar üzerinde test etmek gerekir. Bunun için de OpenSSL speed programını kullanarak iki farklı cihazda bu 2 algoritmayı karşılaştırmışlar. Karşılaştırmalar sonucu güvenlik arttıkça algoritmanın şifreleme bit sayısı da artmaktadır. Bu da cihazda bellek, güç, bant genişliği gerektiriyor ve bu ECC ile daha avantajlı oluyor. Ama daha düşük kısıtları olan cihazlarda ise RSA kullanmak daha avantajlıdır [7].

Hay'Awi, Abdul-Kadhim'e göre şu anki kablosuz güvenlik mekanizmalarını dikkate aldığımızda, noktadan noktaya güvenliği, MIDP 1.0 tam desteklenmiyor, MIDP 2.0'da ise desteklense bile performans kaybı var. Bunun için en iyi yöntem olarak kendisi uygulama bazında güvenlik geliştirmiş. Bunu da Bouncy Castle cryptographic API'lerini MIDP'ye entegre ederek gerçekleştirmiş [8].

Sheralı Zeadally ve arkadaşları mobil kullanıcılar için kablosuz ve kablolu ağlarda uçtan uca güvenlikte araştırma yapmışlar. Önerdikleri çözüm IPSec'i baz alarak çalışıyor. Onlara göre uçtan-uca güvenlik en az iki şartı sağlamalıdır. **1.** Kablosuz-kablosuz

domainler arasında şeffaf olmalıdır, **2.** Kullanıcıya yardım amacıyla kullanıcı mobilitesine yardım etmelidir. Onlara göre geçmişte yapılmış bu tarz araştırmalar kullanıcı mobilitesini yavaşlatmıştır. Kablolu ve kablosuz ağlar için en iyi çözümün IPSec olduğuna karar vermişler ama IPSec'de yavaşlama vardır ve bu yavaşlamanın önlemek için yeni yöntemler sunmuşlar. Geçmiş araştırmalar kullanıcı mobilitesini desteklemek için IPSec ve Mobile-IP'ni beraber kullanmışlar. IKE protokol versiyon 2 üzerine ilaveler yapmışlar. Amaçları kullanıcı mobilitesini artırarak performansa ters etki yapan yükü azaltmak olmuştur. IKE 2 aşamadan oluşmuştur. Aşama 1 ve aşama 2. Aşama 1'de iki uç kendilerini tanıyorlar ve bu tanımaya göre önceden paylaşılmış anahtarı kullanabilirler. Bu anahtara göre bir tane oturum anahtarı üretilir ve oturum devamında bu anahtar kullanılır. Aşama 2'de ise aşama 1'den gelen değerler kontrol edilir. IPSec SA(Security Association)'ları periyodik olarak kontrol ediyor. Opsiyonel olarak Diffie-Hellman exchange yapılır. Yaptıkları öneride kullanıcı yer değiştirdiğinde kullanıcı IP'si değişeceği için aşama 2 tekrarlanacak. Aşama 2'nin tekrarlanmaması için aşama 1'den değerleri aktarırken tüm diğer IP listelerini de beraber aktararak bunu önlemeyi sunmuşlar [9].

Mourad Debbabı ve arkadaşlarının yaptığı araştırmaya göre, gün geçtikçe dünyada java uygulamalarını destekleyen mobil aygıtların sayısı artmaktadır. Java uygulamalarının en büyük avantajlarından biri de aynı uygulamanın başka telefonlar üzerinde de çalışmasıdır yani javanın aygıt bağımsız olarak çalışa bilmesidir. Aynı zamanda J2ME CLDC (Java 2 Micro-Edition – Connected Limited Device Configuration)'de ağ işlemleri de yapılabilir, yani veri gönderip almak mümkün hale gelmiştir. Ama bu zaman güvenlik problemlerini de göz önünde bulundurmak gerekmektedir. Mourad Debbabı ve diğerlerinin araştırmalarında güvenlik açığı analizleri yapmışlar [10]. Bunlardan, ağ güvenlik açığı:

Uzaktaki sitelere bağlanmak için MIDP-SSLv3.0 protocol'unu kullanır. SSL tokalaşma yaparken rastegele sayılar üretmek zorundadır. Bu rastgele sayıları da MIDP'nin PRand.generateData metodu ile sağlanabilir. Bu çevirme işlemi çevrilecek değerin ilk kısmı alınır sonra onu hash algoritmasından çıkartılır (md5) ve sonuç değer sahte rastgele değeri dizisine konur ve seed güncellenir. Sonra ikinci kısım alınır ve burası da güncellenir. vs… Ama sorun şu ki aynı seed için hep aynı değer üretilir. Peki bunun

önlenmesi için ne yapmamız gerekmektedir? Seed güncelleme örneği için aşağıdaki kodu örnek vermişler.

```
Metod  updateSeed
Begin
      T = şimdikiSaniyeBazındaZaman;
      byte abyte0[] =  byte[8];
      for i from 0 to 8 by 1
      begin
            abyte0[i] = T & 255 ;
            T =T  / 2^8;
      End
      md.update(seed, 0, seedUzunluğu);
md.doFinal(abyte0, 0, abyte0Uzunluğu, seed, 0);
End
```

Yukarda gösterilen küçük kod parçasında rastgele bir sayı üretilmiştir. Rastgele üretilen sayımız T değeridir. T'ye kodun çalıştığı sistemdeki o anki zamanın saniye bazındaki değeri atanmıştır. Sonra T'nin değişen değerlerini tutabilmek için bir byte dizisi oluşturulmuştur.Oluşturduğumuz bu byte dizisinin i'ci elemanına her döngüde T'nin 11111111 ikili sayısı yani 255 ile AND'lenmiş değeri atanır.Sonra ise T değeri 2^8'e bölünerek yeni T değeri elde edilir. For döngüsünün sonunda ise md değerine yeni seed değeri atanarak güncellenir. Böylece kodumuz rastgele bir sayı üretmiş olur. Bu kodun ürettiği değeri bilmek için System.CurrentTimeMillis()'i bulmamız yeterli olacaktır. Bunu bulabilmek için snifferlar vardır. Örnek olarak Netscape tarayıcısının SSL gerçekleştiriminin ürettiği değeri bulabilmek için sniffer yardımıyla bir değer aralığı bulunmuş ve bu aralıktaki bir milyon olasılık denendikten sonra atak başarıyla sonlandırılmıştır.

Diğer güvenlik açığı ise "Native Methods" açığı araştırmalarında ise şunları dikkate almışlar. "Native Method" gerçekleştirimi Java'da tanımlanmayan metottur ve kullanmadan önce belleğe yüklenmesi gerekiyor. J2ME CLDC native metod gerçekleştirimine izin vermemektedir. J2SE ise izin vermektedir. Uygulamaya güçlü güvenlik desteği sağlamak için J2ME CLDC native method gerçekleştirimine izin

vermemektedir. Ama uygulamalar bu kurala pek uymamaktadır. Onlar çalışmaları sırasında J2ME CLDC'ye içinde native metod deklarasyonları olan MIDlet'ler ekleyebilmişler. Diğer güvenlik açıkları ise aygıttan Jar dosyasını getirmek ve transfer etmektir. MIDlet indirmenin tipik senaryosu şu şekildedir: ilk önce mobil uygulama sağlayıcıya bağlanılır ve MIDlet ler indirilir. Bu MIDlet üzerinde yapabileceğimiz işlemler çalıştırmak ve kaldırmak olmalıdır. Burada MIDlet'in gönderilmesine izin verilmemelidir. Araştırmacılar deneyleri sırasında bu MIDlet'leri göndermeyi başarmışlar. Son olarak araştırdıkları bir diğer nokta ise, bellek yetersizliği güvenlik açığı olmuştur. Ne zaman ki MIDlet'ler kalıcı bellekte de bellek bilgisine ihtiyaç duysa kendisi bir kalıcı bellek üzerinde bir bellek kaydı yaratabilir ve bundan sonra bütün MIDlet'ler bu bellek alanını kullanabilir hale gelirler ama bu kısımlar üzerinde bazı kısıtlamalar yapılmak zorundadır. MIDP spesifikasiyonunda belirtildiği gibi MIDlet'ler üzerinde gerekli olan bellek kısıtlamaları yapılmamıştır. Bunun anlamı şu demek: Hiç bir şey MIDlet'leri kalıcı belleğininin tüm boş yerlerinin kullanmasını önleyemez [10].

Andre N.Klingsheim ve arkadaşlarının yaptıkları araştırmaya göre, mobil uygulamalar için en güvenli ortam J2ME seçildikten sonra bu yönde bazı güvenlik problemleri çıkmıştır. Normal bazı masaüstü uygulamaların - mesela Microsoft ürünleri - kendilerini otomatik olarak güncellemektedir. Ama cep telefonlarında bu olmamaktadır, yani kullanıcı telefonu güncellemek için cep telefonu servisine başvurması gerekmekte ve doğal olarak da çoğu insan bunu yapmamaktadır. Telefonlarda güvenliği artırmak için MIDP 2.0'in güvenlik uygulamaları vardır. Bu uygulama iletişim için HTTPS kanalını SSL veya TSL protokolleri yardımıyla kullanmaktadır. MIDP 2.0'da güvenilir iletişim kurabilmek için aşağıdakilerden biri veya bir kaçı gerçekleştirilmiş olmalıdır:

1) TLS, HTTP üzerinden (RFC 2818) TLS v1.0 ile (RFC 2246)
2) SSL v3.0,
3) Kablosuz Aktarım Katmanı Güvenliği (WTLS), veya
4) WAP TLS Profil ve Tünel Spesifikasyonu

Ayrıca MIDP 2.0 güvenli bağlantılar gerçekleştirmek için sunucunun kimlik doğrulaması için geçerli sertifikaya sahip olmasını gerektiriyor. Ancak, MIDP 2.0

istemci tarafın sertifika tabanlı kimlik doğrulamasını desteklemediği için istemcinin uygulama düzeyinde kimlik doğrulamasını gerektiriyor. MIDP 2.0 uygulamalarında bellek bölgeleri için şifreleme kullanılmamaktadır. Bu da uygulamaların ataklara açık olmasına sebep olmaktadır. Ama bunu farklı API'ler yardımıyla az da olsa sağlamak mümkündür. Bir çok uygulama varolan rastgelelik kaynaklarını kullanarak PRNG (Pseudo-Random Number Generator) için başlangıç değeri üretiyor. MIDP 2.0 ise güçlü rastgele sayı üretemiyor.

MIDP 2.0 diğer kötü taraflarından biri de rastgele fonksiyonunun J2SE'deki kadar güvenilir sayılar üretmemesidir. Bundan dolayı MIDP 2.0 uygulamaları rastgele fonksiyonunu şifreleme için kullanmamaktadır bu fonksiyon daha değişik amaçlar için kullanılmaktadır [11].

Akıllı kartlarda ayrıca oynanan oyunun profilleri de saklanabilir hale gelmiştir. Mesela oyun oynarken bazı olayların saklanması, kazanılan bazı silahların saklanması. U(Sim) kart yardımıyla kullanıcılar finansal işlerini artık kolaylıkla yapabilecekler. U(Sim) cep telefonunda takılı akıllı kart üzerinde çalışan bir uygulamadır. Bunun yanı sıra elektronik cüzdan, mobil bankacılık işlemleri de buradan yapılabilecektir. Kablosuz işlerin genişlenmesi mobil aygıtların, PCler gibi kullanılması gün geçtikçe artıyor, bu da teknolojiler için güvenlik problemlerinin ortaya çıkmasına sebep oluyor ve çeşitli ataklara maruz kalıyorlar. Bu virüs baskılarına 3G ve GPRS'in çıkmasıyla daha da ataklar olmuştur. Akıllı kart operatörlere ve mobil servis sağlayıcılara taşınabilir güvenlik sağlamaktadır. Akıllı kart veri akışı güvenliğinin en iyi yoludur. Pazarlama işleri için kablosuz kanal üzerinden erişim zamanında (U)SİM çok iyi kimlik doğrulaması yaparak sistemi korumaktadır [12].

Mobil aygıtlarda veri iletişimi mobil ağlar tarafında sağlanıyor. Mobil ağlar da birden çok atağa açıktır. Bu ağlarda ataklar verinin güvenliğini tehdit etmektedir. Bir çok satıcı, bu soruna kendi çözümlerini sunmaktadırlar. Mobil ağlar genelde proxy tabanlı mimariye sahipler. WAP'ın WTLS (Kablosuz Aktarım Katmanı Güvenliği) çözümü buna örnek olabilir. Bunun iki dezavantajı var. İlki, veriyi kodlamak ve kodunu çözme zamanı performans kaybının olması. İkincisi, verinin kodlanmak ve kodunu çözme zamanı başkası tarafından okunabilmesine imkân vermedir. Proxy tabanlı mimariye

alternatif uçtan uca güvenliktir. Bu da verinin kaynakta kodlanması ve vardığı yerde kodunun çözülmesi anlamına geliyor. Buna da örnek olarak TLS ve SSL gösterilebilir. Bu iki protokol mobil ağlar için uygun değil ve aynı zamanda kaynak tüketimi çok fazladır. Yaptığı araştırmada TLS 1.0 protokol mimarisini mobil aygıtlar için uygulamış. Bunun için TLS protokolünün yaptığı tüm işlemleri kendisi yüksek seviyeli protokol tanımlayarak gerçekleştirmiş [13].

Jøsang ve Sanderud yaptıkları araştırmada mobil ağlarda güvenliğin diğer ağlara göre daha zor olduğuna dikkat çekmişler. Bu güvenliği sağlamanın en iyi yönü uçtan uca güvenliği sağlamakla mümkün olduğuna karar vermişler [14].

Otto Kolsi ve arkadaşları yaptıkları araştırmada, MIDP 1.0 ve MIDP 2.0'daki güvenlik açıklarını karşılaştırmışlar [15].

Mobil uygulamaların çalışma karakteristiklerini incelediklerinde çalışma ortamına bağlı olarak birden fazla güvenlik riski ile karşılaşmışlardır. Örnek olarak, MIDP uygulaması başka bir uygulama ile HTTP veya WAP protokolünü kullanarak haberleşebilmektedir. Bu tür protokoller, birden fazla riske açıktırlar. Güvenli ortam için gereken anahtar servisler güvenilirlik, bütünlük ve uygunluktur. Yaptıkları araştırmada MIDP 1.0'da en önemli güvenlik açıkları güvenli ağ protokollerini desteklememesi ve kriptografik algoritmalar içermemesidir. MIDP 2.0 güvenlik analizlerinde ise MIDP 2.0′ın yeni özellikleri ile bütünlüğü doğrulama ve yetkilendirme güvenlik açıkları giderilmiştir. HTTPS güvenlik protokolü ilave edilmiştir. Ama bunların hepsi X.509 PKI'i taban aldığı için yine de 100% güvenlik sağlandığı söylenememektedir. HTTPS'in ilave edilmesine bakmayarak, halen güvenlik açıkları devam edebilmektedir, çünkü HTTPS sadece iki nokta arasında güvenli link sağlıyor, içerik şifreleme gerçekleştirmiyor. PKI tabanlı doğrulamada ve HTTPS/SSL bağlantılarında kullanıcı etkileşimi de gerekiyor. Bu gibi durumlarda kullanıcının belirli bir sertifikaya güvenmesine karar vermesi gerekmektedir. Bu da MIDP 2.0'da güvenlik problemlerinden bir tanesidir. Bir de MIDP 2.0'da yeni özelliklerin eklenmesi ile yeni güvenlik açıkları oluşmuştur [15].

Michael Yuan ve Ju Long yaptıkları araştırmada, J2ME kullanarak MIDP uygulama geliştirme zamanı var olan veya geliştirilebilecek güvenlik çözümlerini tartışmışlar [16].

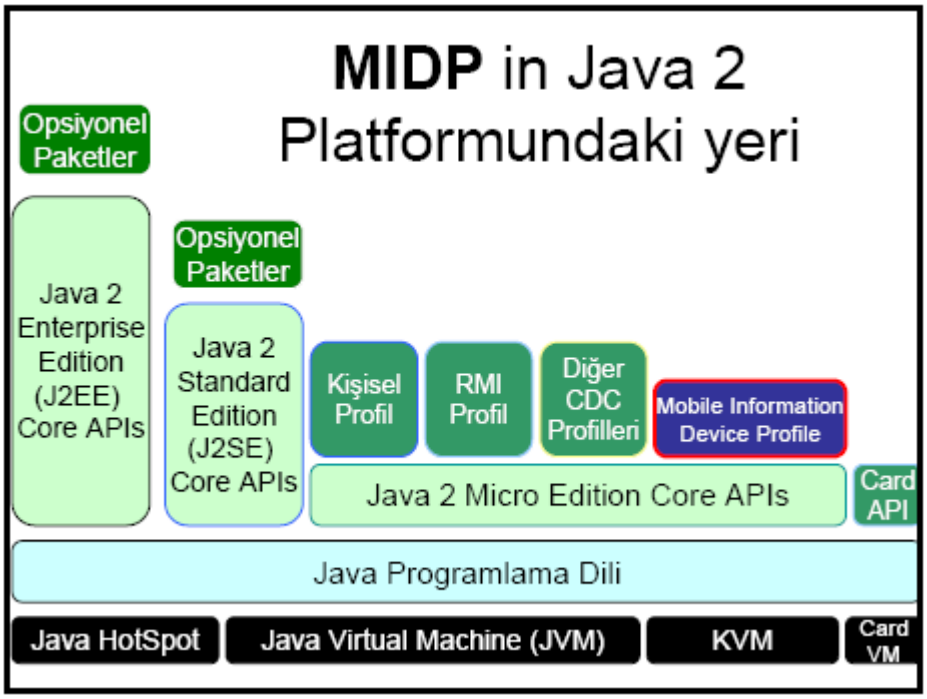
J2ME ile geliştirilen uygulamanın WAP uygulamalara göre avantajları: WAP geçit kullanmıyorlar, iletişim için gerekli bilgiyi aygıt kaynaklarını kullanarak ağ üzerindeki yükü azaltıyorlar ve mobil aygıtın işleme gücünü efektif kullanıyorlar. J2ME'nin diğer yerel platformlara göre avantajı ise class'ları class loader'da kontrol ediyor ve herhangi tehlikeli iş yapmasını engelliyor. JVM'nin izleme mekanizmasını kullanıyor. Örnek olarak, Java Çöp Toplayıcısı (Java Garbage Management) gösterilebilir. Java Çöp Toplayıcı, Java bellek yöneticisidir. JVM'ler güvenlik yöneticileri ile çalıştırılmak istenen uygulamanın güvenilir olup olmadığını uygulamanın işareti ile belirleyebilir. J2ME bytecode kontrolü yaparak uygulamanın bellekte başka yerlere ulaşmasına izin vermiyor. Yüksek hesaplama yüküne göre MIDP VM bunu gerçekleştirmiyor. Bu yüzden uygulama geliştirici bu işlemi uygulamanı harekete geçirmeden önce yapması gerek. Önceden bytecode kontrol uygulamanın daha hızlı çalışmasını sağlıyor. HTTPS'in MIDP 2.0'a uygulanmasına rağmen bazı eksiklikleri vardır. Örnek olarak, SSL birebir protokol olduğu için multicast uygulamaları desteklemiyor. SSL'in bir diğer dezavantajı ise tüm veriyi uzunluğuna bakmazsızın aynı uzunlukta anahtarla şifreliyor [16].

2. GENEL KISIMLAR

2.1 J2ME

Java 2 Platform, Micro Edition'ın (J2ME platformu) temel bileşenleri, tüketici aygıtları ve gömülü aygıtlar pazarına Java çözümleri sunan çeşitli araçların ve teknolojilerin yanı sıra Connected Device Configurations (Bağlantılı Aygıt Yapılandırmaları), Connected Limited Device Configurations (Network Bağlantılı Kısıtlı Aygıt Yapılandırmaları) ve Mobile Information Device Profiles (Mobil Bilgi Aygıtı Profilleri) yazılımlarını içerir [17].

J2ME teknolojileri, geniş bir tüketici alanının gereksinmelerini karşılamak için özel olarak geliştirilmiş olan JRE'yi içerir. J2ME teknolojileri, çok geniş bir ürün çeşitliliğiyle son derece küçük cihazlarda kullanılabilir ve akıllı kartlar, çağrı cihazları, çözücüler ve diğer küçük aygıtlar için güvenlik ve bağlanabilirlik çözümleri ile yararlı yardımcı programlar üretilmesine olanak sağlar.



Sekil 2.1: MIDP'in Java 2 Platformundaki Yeri [17]

2.1.1 J2ME'nin Java Platformları Arasındaki Yeri

Günümüzde kullanım açısından Java'nın 3 versiyonu vardır :

- Java 2 Standart Edition (J2SE)
- Java 2 Enterprise Edition (J2EE)
- Java 2 Micro Edition (J2ME)

Bu versiyonlar, değişik bilgisayar ve bilişim aygıtı barındıran ortamlar için tanımlanmıştır.
Günümüzde J2ME teknolojisi, Java teknolojisinin kökenlerine geri dönüş olarak düşünülebilir. Java programlama dili, esasında elektronik cihazlara program yazmak amacıyla geliştirildi. Ama zamanla masaüstü uygulamaları ve sunucu tabanlı uygulamalar geliştirmeye yönelik değişim geçirdi. Bu sebeple J2ME, Java'nın başlangıç fikrine geri dönüş olarak değerlendirilebilir [17].

1990'larda Green projesi ve Oak programlama diliyle yürütülen çalışmalar daha sonra Java ismini aldı. Bugünün J2ME'si de, Sun Microsystems'in küçük kapasiteli cihazlar için geliştirdiği platformların en sonuncusudur. Green projesinin Oak kısmı (1990'lar), Java Card (1996), PersonalJava (1997) ve Embedded Java (1998), Spotless System ve KVirtual Machine (1999) daha önce geliştirilmiş platformlardır [18].

J2ME, üç farklı Java2 platformundan bir tanesidir. Her bir platform farklı Java teknolojileri içerir ve farklı bir çalışma zamanı ortamı sağlar. Bu platformlar değişik alanlarda uygulama geliştiren programcıların ihtiyaçlarını karşılar. Java2 Platform Standard Edition (J2SE), temel java ortamıdır. Bu ortam, çekirdek java sınıflarını ve API' lerini web tarayıcıları üzerinde çalışabilecek uygulamalarda dahil standart istemci-sunucu uygulamalarını geliştirme ve çalıştırma olanağı sağlar. Java2 Platform Enterprise Edition (J2EE), J2SE'nin bir üst kümesini oluşturur. Genellikle ölçeklenebilir, işlem tabanlı ve veritabanı merkezli iş programların ve çok katlı, dağıtık uygulamaların geliştirilmesi için kullanılır. Java2 Platform Micro Edition (J2ME),

mobil telefonlar, PDA'lar, TV set-top box'lar gibi gömülü sistem cihazlar için uygulama geliştirmeye yönelik API'ler ve çalışma zamanı ortamı tanımlar. J2SE ve J2EE'den daha kısıtlı kaynaklara sahiptir [18].

Tipik bir iş uygulaması üç platformun da avantajlarından faydalanarak hazırlanabilir. Sunucu tarafında J2EE tabanlı uygulamalar geliştirilirken, bu uygulamalar istemci tarafında J2SE ve J2ME tabanlı uygulamalara servis verir. Platformların bu şekilde belirgin bir şekilde birbirlerinden ayrılması, geliştiricilerin farklı ortamlar için farklı uygulamalar geliştirmesine olanak sağlar. Kablosuz cihazların sınırlamaları göz önüne alındığında J2ME'nin neden diğer sürümlere oranla kısıtlı özellikler taşıması anlaşılıyor. Java'nın kablosuz cihazlar için ürettiği J2ME sürümü üretilmeden önce Wireless Access Protocol (WAP) ve i-mode adında iki programlama platformu geliştirilmiştir. Bu teknolojilerde günümüzde var olan programlama platformları tamamen yeniden düzenlenmiştir. Örneğin WAP, HTML adındaki normal bilgisayarlarda kullanılan dil yerine, Wireless Markup Language (WML) adında kendi uygulama dilini hazırlamıştır. Benzer çalışmalar i-mode teknolojisinde de yapılmıştır. Tamamen yeniden düzenlenen programlama yapısı, farklı makinelerde uyum problemine yol açmaktadır. Farklı yapıdaki programlama yapısı ile aynı programın değişik makinelerde kullanılması mümkün olmamaktadır. Örneğin, WAP destekli cihazlar normal bilgisayarlar için hazırlanan web sayfalarını ziyaret edememektedirler. Bu yüzden WAP kullanıcılarının bu sayfaları görebilmeleri için tamamen yeniden bir sayfanın hazırlanması gerekmektedir. Benzer bir şekilde, i-mode kullanıcıları sitelere basit socket bağlantıları yapamamakta ve ancak bağlantı kurucu ek protokoller ile bağlantı işlemlerini yerine getirebilmektedirler. WAP ve i-mode sistemleri hakkında bahsi geçen farklı programlama yapıları J2ME için geçerli değildir. Java dilinde yazılan bir program ile socket bağlantısında, Internet sitelerine giriş direkt olarak sağlanabilmektedir. Değişik mimarisinin sonucu olarak J2ME, J2SE ve J2EE'ye kıyasla değişik amaçlara sahiptir. J2ME mimarisinin ana hedefleri;

- Değişik yetenekteki cihazlara destek sağlar. Bu cihazlar kullanıcı arayüzü, veri depolama, ağ bağlantısı ve bant genişliği, bellek büyüklüğü, güç tüketimi, güvenlik gereksinimleri duyarlar. Çok fazla kişiselleştirilmiş hatta sadece tek bir kişi tarafından kullanılan cihazlara odaklanmıştır.

- Değişen aralıklardaki ağ kapasitelerinde ve servislerinde ağ bağlantısı sağlar. Ağ bağlantısı J2ME uzayında yer alan cihazlar için çoğunlukla hayati önem taşır ve yetenekleri, düşük bant genişliği, kablosuz ve aralıklı bağlantıdan daha sık ve daha yüksek bant genişliğindeki bağlantılar aralığında değişir. Ağ bağlantısı üzerinden uygulamaların ve verinin alınmasını sağlar. Ağ bağlantısı J2ME uygulamalarının cihazlara taşınmasında tercih edilen bir yoldur. Uygulamaların cihaz üzerinde hazırlanabilmesi veya belleğe doğrudan yüklenebilmesi ve çalıştıktan sonra bellekten atılabilmesi gerekir.

Java dilinin çapraz platform özelliklerini her cihazın eşsiz özelliklerini ve kısıtlarını alarak genişletir. Hızlıca değişen pazara esneklik ve var olan ve henüz çıkmamış uygulamalara uyum sağlar. Değişik yetenekteki, özellikteki ve işlem gücündeki cihazlarda uygulamaların ölçeklenmesini sağlar. Original Equipment Manufacturer (OEM)'dan bağımsız olarak J2ME destekli cihazlarda uygulama geliştirmeyi sağlar [19].

2.1.2 J2ME'nin Temel Kavramları

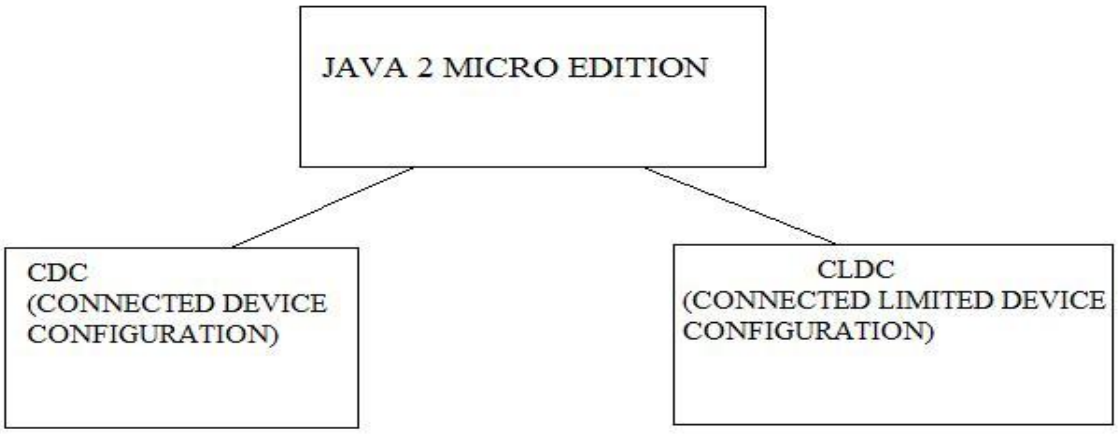
2.1.2.1 Konfigürasyonlar

J2ME konfigürasyonu bir aygıt ailesi için minimum Java platformunu tanımlar. Bu aileye ait aygıtlar benzer bellek ve işlemci yeteneklerine sahiptir. Bir konfigürasyon, sistem düzeyinde programcının o aygıtın desteklediğini bildiği bazı dil ve platform özelliklerinin tanımlamasıdır. Belli bir konfigürasyona giren aygıtlarda hangi Java kütüphanelerinin olduğunu bilir ve ona göre program yazabilir. Özetlersek, bir konfigürasyon şu maddeleri içerir: [19]

- Belli Java programlama dili özellikleri
- Belli Java sanal makinesi özellikleri
- Belli Java kütüphaneleri

Konfigürasyon, temel ve en küçük ortak paydayı tanımlayan J2ME çalışma zamanı ortamıdır. Sanal makine ve J2SE'den alınan temel sınıf kümelerini içerir. Her bir

konfigürasyon, benzer özelliklere sahip geniş cihaz aileleri için tanımlanmıştır. Tanımlanan iki konfigürasyon "Connected Limited Device Configuration" (CLDC) ve "Connected Device Configuration" (CDC)'dir.



Şekil 2.2: J2ME Konfigürasyonları

CLDC, kaynakları kısıtlı ve ağ bağlanabilirliği sınırlı olan cep telefonu, kişisel bilgisayarlar gibi alt uç (low-end) cihazları destekler. CDC ise, daha az kısıtları olan, yüksek seviyede iletişim olanağına sahip PDA ve ileri seviye gömülü cihazları destekler [19]. İsimlerinden de anlaşılabileceği üzere, konfigürasyonlarda düşük hız - yüksek hız ya da kablolu - kablosuz olmasına bakılmadan bilgisayar ağı bağlanabilirliği olan kısıtlandırılmış cihazlardan bahsedilmektedir.

2.1.2.2 CLDC (Connected Limited Device Configuration)

CLDC hesaplama gücü, pil ömrü, bellek ve iletişim ağı bant genişliği bakımından önemli kısıtlamaları olan cihazlar için en küçük seviyedeki J2ME konfigürasyonudur [19].

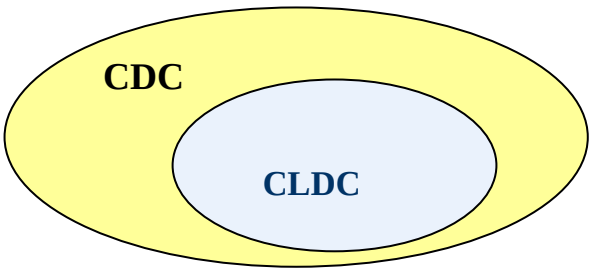
CLDC, çok fazla kaynak gerektirmez. CLDC 1.0, 16 yada 32 bit işlemcili, en az 160 KB kalıcı bellek, ve en az 32 KB uçucu bellek yani toplamda 162 KB belleğe sahip cihazları destekler [19]. Güç tüketimi de oldukça düşüktür. Ağ bağlanabilirliği düşük hızda olabilir. Bu konfigürasyonun merkezinde ise Kilobyte Virtual Machine'in

referans uygulaması olan Java sanal makinesi yer alır. Bu sanal makinede J2SE'nin bazı özellikleri kaldırılmıştır.

CLDC, J2SE geliştiricilerinin yakından bildiği Java paketlerinin alt kümesini ve buna ek olarak kısıtlandırılmış cihazlar için eklenmesi gerekli olan sınıfları tanımlar. Ek olarak CLDC, java.io ve java.net sınıflarını içerecek kadar belleğe sahip olmayan cihazlarda I/O işlemlerini destekleyebilmek amacıyla, "Generic Connection Framework" (GCF) paketi olan javax.microedition.io'u içerir. GCF, HTTP, datagram, streamler ve I/O'yu destekleyen sınıflar gibi bağlantı yaratmayı sağlayan sınıf ve arayüz hiyerarşilerini içerir.

2.1.2.3 CDC(Connected Device Configuration)

CDC, CLDC'nin desteklediği cihazlardan daha güçlü olan üst-uç cep telefonları ve kişisel bilgisayarlar (PDA), web tabanlı cihazlar, set-top TV kutuları gibi cihazlar için tasarlanmıştır. CDC'yi destekleyen cihazlar 32 bit işlemcili, tipik olarak ARM tabanlı, en az 2 MB ana bellek ve 2.5 MB okunabilir belleğe sahip, iletişim ağı kurma yeteneği olan cihazlardır [20]. Bu konfigürasyonun merkezinde ise bütün J2SE yeteneklerine sahip Java sanal makinesi yer almaktadır. Şekilde de görüldüğü üzere CDC, CLDC'nin üst sınıfıdır. CDC, CLDC' de tanımlanan bütün sınıfları ve "Generic Connection Framework" gibi J2SE'de yer almayan ve yeni eklenen sınıfları içerir.



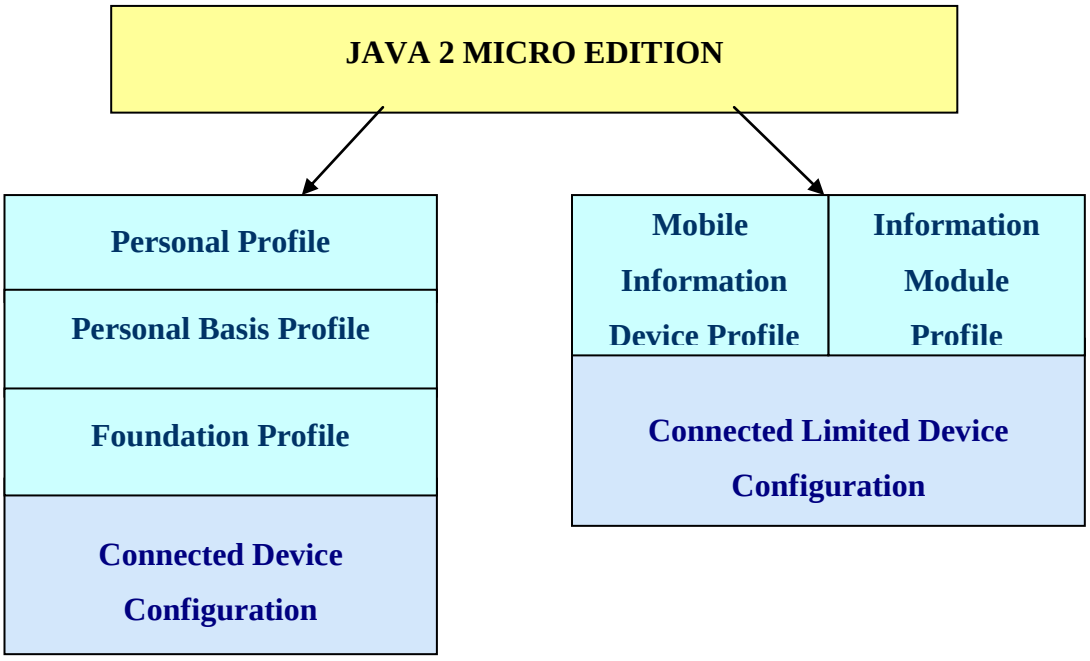
Şekil 2.3: CLDC ve CDC Arasındaki İlişki

2.1.3 Profiller

J2ME profili uygulama düzeyinde verilen servislerden oluşur. Verilen servis herhangi bir konuda olabilir. Ancak belli konulardaki profiller standarttır, o profili destekleyen bütün aygıtlarda aynı servis aynı şekilde bulunur. Profiller bir konfigürasyon üzerinde çalışırlar. Ayrıca bir profil bir başka profil üzerinde çalışabilir. Bir aygıta birden çok profili destekleyebilir ancak sadece tek bir konfigürasyona sahip olur. Örneğin SMS Mesajlaşması bir profil konusudur. Bu, mobil telefon konigürasyonu için çok yaygın bir profildir [19].

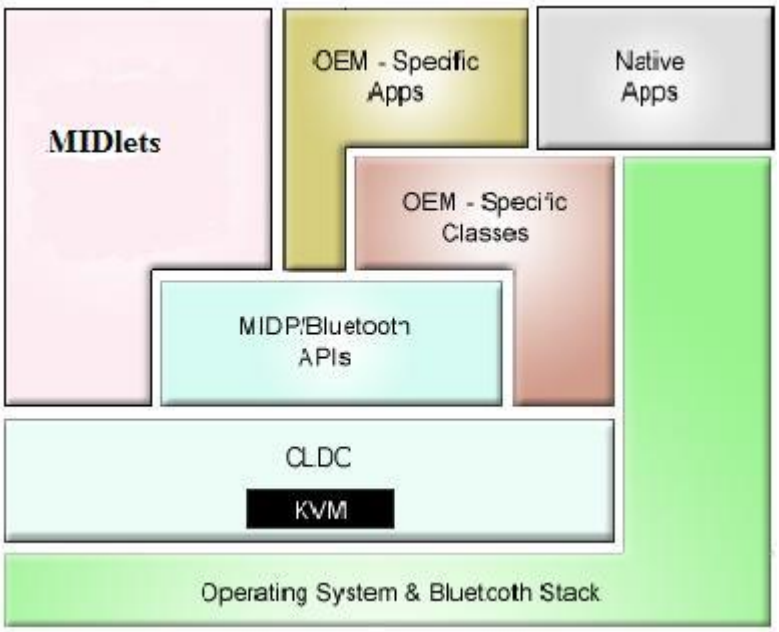
Konfigürasyonlar, uygulama yaşam döngüsünü yönetmek, kullanıcı arayüzü oluşturmak, cihaz içinde tutulan veriyi güncellemek ve yönetmek, ağ sunucusunda yer alan güvenlik bilgisine erişmek için gerekli sınıfları içermezler. Bu fonksiyonlar, profiller ya da seçimlik paketler tarafından gerçekleştirilir. Profiller, konfigürasyonlar tarafından sağlanan çekirdek sınıf kümelerine, ilgili cihaza özgü servisler için gerekli sınıfları ekler. Bu sınıflar, alt katmandaki konfigürasyonda yer almayan, ama cihazın kendine özgü olan fonksiyonları yerine getirmesinde ve kullanımında gerekli olan servisleri içerirler.

CLDC tabanlı profiller, Mobile Information Device Profile (MIDP, versiyon 1.0 ve 2.0), Information Module Profile (IMP)'dir. CDC tabanlı 3 profil vardır. Bunlar Foundation Profile (FP), Personal Basis Profile (PBP), Personal Profile (PP) dir [19]. Aşağıdaki şekilde, profiller ve alt katmanlarında yer alan konfigürasyonlar gösterilmiştir.



Şekil 2.4: Profiller ve alt katmanlarında yer alan konfigürasyonlar [19]

MIDP, ilk J2ME profilidir. PDA ve cep telefonları üzerinde milyonlarca uygulaması geliştirilen MIDP, en fazla benimsenen ve geliştirilen profildir. IMP'nin hedef cihazları ise MIDP'ye benzerdir. Ancak bu cihazlar, arayüz desteği çok az olan yada hiç olmayan cihazlardır. IMP uygulamaların yönetimi, veri depolanması, ağ bağlantısı, güvenlik gibi birçok fonksiyonu MIDP'den almıştır. Ama kullanıcı arayüzü için gerekli olan API (javax.microedition.lcdui)'ler bunun dışında tutulmuştur.



Şekil 2.5: MIDP Katmanları [19]

MIDP 1.0 JSR 37 içinde gelmiştir. MIDP 2.0 ise JSR 118 tarafından tanımlanmıştır ve profilin fonksiyonlarını arttırmak amacıyla geliştirilmiştir. İletişim, kullanıcı arayüzü ve MIDlet hayat döngüsü API'lerine ek olarak TCP soketlerini, UDP datagramlarını, güvenlik bağlantılarını destekleyen yeni API'ler eklenmiştir. Aşağıdaki şekilde MIDP ve IMP için cihaz gereksinimleri listelenmiştir:

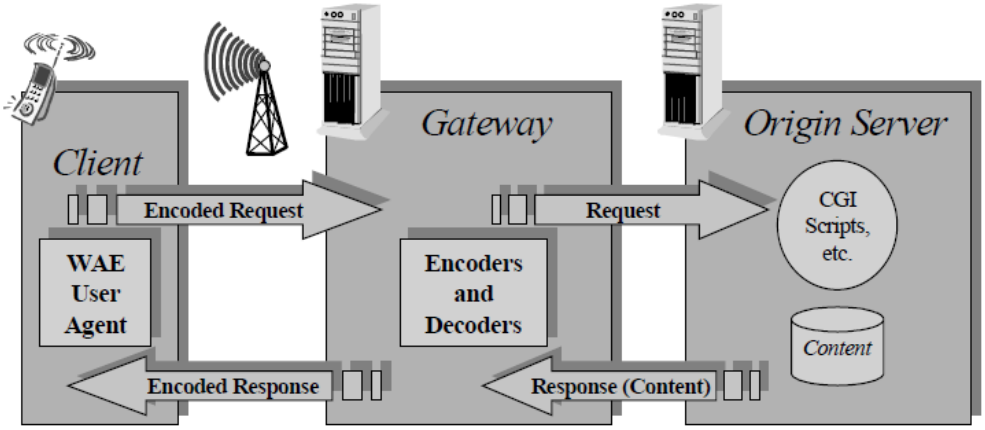
MIDP için Donanım Gereksinimleri

Parça			Kapasite
Görüntü	Ekran Boyutu		96*54 Pixel
	Görüntü Derinliği		1-bit
	Pixel Shape (aspect oranı)		Yaklaşık 1:1
Giriş			Aşağıdakilerden biri: Tek-elli klavye İki-elli klavye Dokunmatik Ekran
Hafıza	Kalıcı	MIDP Parçaları	128 Kbytes
		Uygulama yarattığı kalıcı veri	8 Kbytes
	Geçici	Java Run Time	32 Kbytes
Ağ, iletişim			İki-yönlü kablosuz, sürekli olmayan bağlantıda da çalışma özellikli

Şekil 2.6: MIDP ve IMP için cihaz gereksinimleri

2.2 WAP MODEL

WAP modeli WWW modeline program uygulama biçimi bakımından çok yakındır. Bu da WAP modeline yeni teknolojileri ilave etmeye kolaylıklar sağlamaktadır. Böylelikle de WAP modeline yapılan ilaveler ve optimizasyonlar kablosuz ortamların karakteristiklerini karşılama imkanlarını artırıyor. WAP modeli aşağıdaki gibidir:

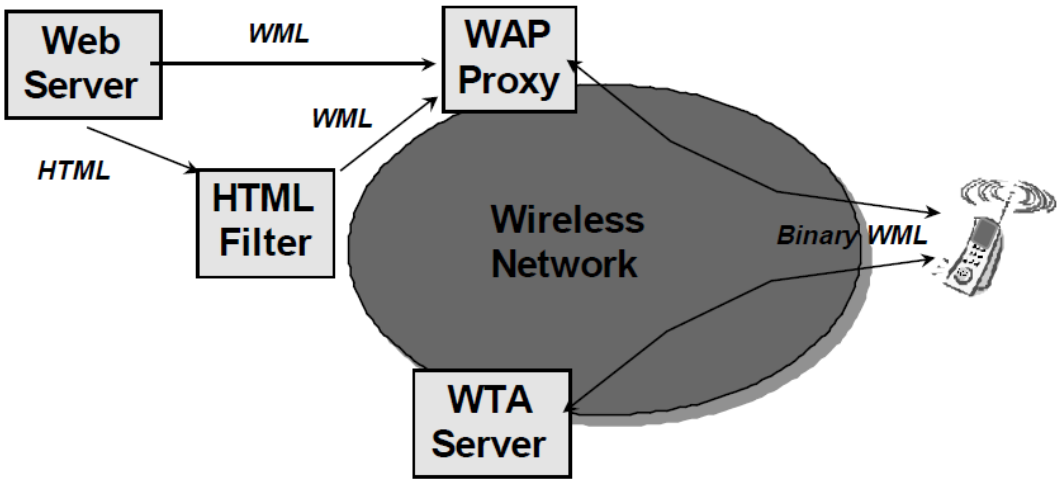


Şekil 2.7: WAP modeli [20]

WAP kablosuz alan ve WWW arasında bağlamak için Proxy teknolojisini kullanır. WAP Proxy teknolojisinin genel işlevselliği aşağıdaki gibidir [20].

Protocol Gateway: Gateway protokolü (WSP, WTP, WTLS ve WDP) WAP yığıt protokolünden istekleri WWW(HTTP ve TCP / IP) protokolü yığınına çevirir. Content Encoders and Decoders: İçerik çeviriciler WAP içeriğini kompakt kodlanmış biçime çevirerek ağ üzerindeki yükü azaltıyorlar.

Örnek bir WAP ağı:



Şekil 2.8: Örnek bir WAP ağı [20]

Bu örnekte WAP istemcisi iki sunucu ile iletişim kuruyor. WAP Proxy WAP istemcisinin isteklerini WWW isteklerine çevirerek WAP istemcisinin WEB Sunucu ile iletişimini sağlıyor.

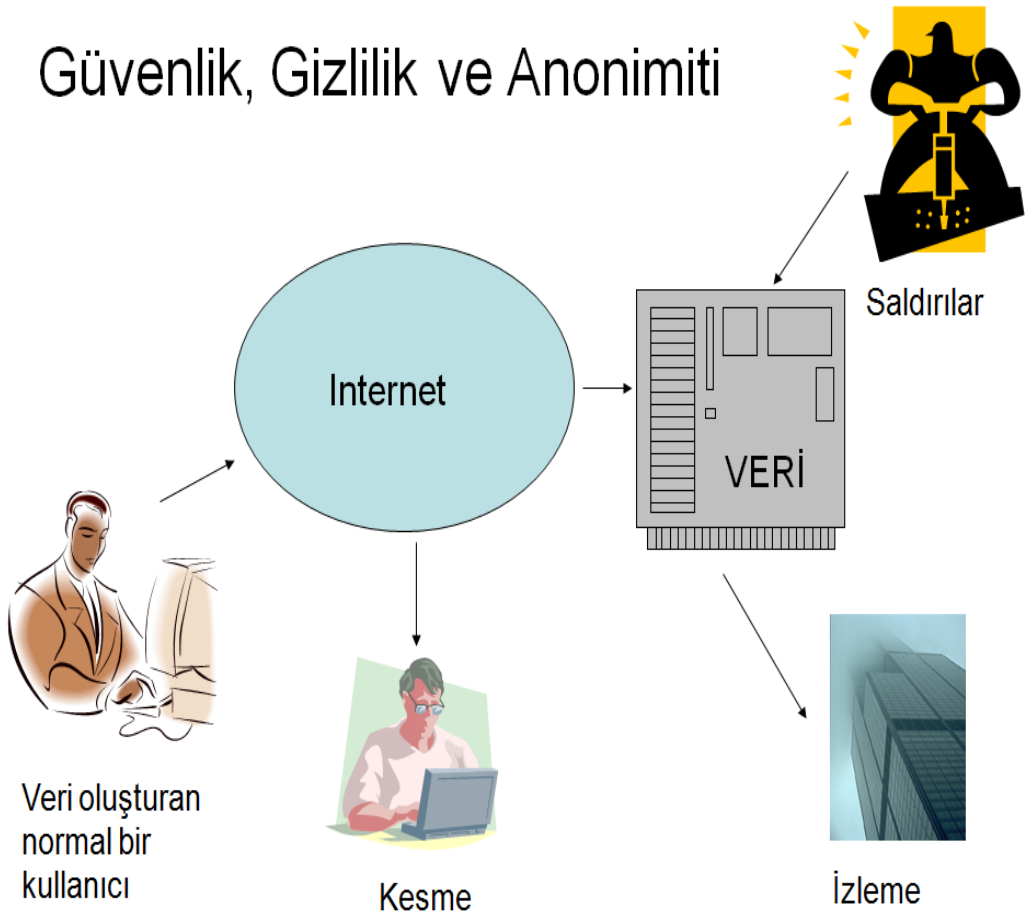
Burada dikkat edilmesi gereken nokta eğer WEB Sunucu direk olarak WAP içeriğini destekliyorsa WAP Proxy gelen içeriği direkt olarak alıyor. Eğer gelen içerik WWW uyumlu ise onda önce HTML Filter'dan geçiriliyor ve sonra WAP Proxy'e gönderiliyor [20].

İşte bu çevirme işlemleri zamanı güvenlik sorunları ortaya çıkıyor.Bu sorunu çözmek için en iyi yöntem içerik şifreleme yöntemidir.

2.3 KRİPTOGRAFİ

2.3.1 Neden Kriptografi ?

Elimizdeki verilerin herhangi bir sebepten dolayı başkaları tarafından okunmaması veya içeriğin anlaşılmamasını isteyebiliriz. Bundan dolayı insanlar tarih boyunca verileri bir şekilde gizleme veya kendileri dışında başkaların anlamayacağı şekillere sokmaya çalışmışlardır. Bu korunma yöntemlerinden en çok bilineni Kriptografi olarak bilinir.



Şekil 2.9: Örnek bir bilgi iletişimi [21]

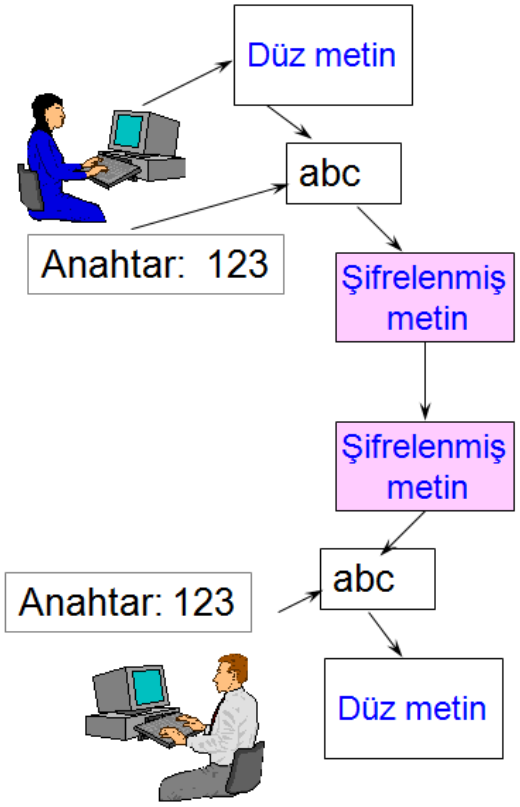
Şekil 2.9'dan da anlaşılacağı gibi eğer internetteyseniz ve bilgilerinizi bir yerlere gönderiyorsanız bilgileriniz her an bir tehlikeyle karşılaşa bilir bunu aşmak için tek çözüm kriptografidir.

Kriptografi kısaca şöyle tanımlayabiliriz: Kriptografi, gündelik hayatta insanların okuyup anlayabileceği türdeki bilgilerin şifrelenmesi için kullanılan tekniklerin

tümüdür. Kriptografi Romalılar zamanından var olup ve tarih boyunca sık -sık rastlanılmıştır. Örneğin: İkinci dünya savaşı gibi modern savaşlarda önemli rol oynamıştır. Bunun gibi son zamanlarda bilgisayar teknolojisinin gelişmesiyle dünyamız Bilgi Çağına gelmiştir ki, buda demek oluyor ki, sanal verilerin inanılmaz derecede artmasına neden olmuştur. Bu sebepten dolayı sanal dünyada kriptografi çok önem kazanmıştır. Bilgisayar endüstrisinin güvenli bilgi alış verişi ve depolama işlemlerinde kriptografi vazgeçilmez duruma gelmiştir. Kriptografi gelişmesinde sanal dünyanın inanılmaz derecede önemi vardır. Bunlara örnek olarak birçok kriptografi algoritmalar örnek verebiliriz. Elektronik ticaretin kullanılmaya başlanılmasıyla da kriptografi alanına büyük çapta ticari bir ilgi oluştu. Bazı hükümetler, özellikle ABD, kriptografinin geliştirilmesi ve uygulanmasını kontrol etmeye çalışıyor. Kriptografik teknikler sabit disk, disket ve manyetik teyp gibi medyalar üzerinde depolanan önemli bilgilerin güvenliğini sağlamak için kullanılır. Aynı zamanda Internet üzerine transfer edilen bilginin güvenliğini sağlamak içinde kullanılır [22].

2.3.2 Bilgisayar Dünyasında Kriptografi

Kriptografi şifreleme ve şifre çözme gibi iki en önemli bileşeni vardır. Kriptolama gündelik hayatta insanların okuyup anlayabileceği türdeki bilgilerin şifrelenmesi işlemine denir. Bunu gerçekleştiren ise kriptolama anahtarıdır. Dekriptolama ise şifrelenmiş bilginin eski haline getirilmesidir. Bunun da gerçekleşmesini sağlayan bir şifre çözme anahtarıdır. Eğer bir metin veya herhangi bir bilgi şifrelenecekse yazı bir kriptolama yöntemi kullanılarak şifreli yazıya çevrilir ve ardından eğer gerekirse yeniden şifre çözme işlemine tabi tutulabilir.



Şekil 2.10: Örnek bir kriptolama

Kriptolamaya ilişkin basit bir örnek verelim. Mesela gönderilmek üzere olan bir "abc" satırımız olsun. Bu bilgiyi şifrelerken şu yöntem kullanılabilir. Satırdaki her harf bir sayı denk gelir ve satır şu şekilde "123" dönüştürülüp gönderilir. Buna göre a=1, b=2, c=3 olur. Karşı taraf ise bilgiyi aldığında kriptolama anahtarını bildiğinde 1 yerine a , 2 yerine b, 3 yerine de c koyar ve orijinal veriye ulaşır. Görüldüğü gibi burada da hiçbir veri kaybı yaşanmaz.

Kriptografinin sağlaması gereken 4 ana özellik vardır:

- Gizlilik
- Kimlik tanımlama
- Bütünlük
- İnkar edememe

Bu fonksiyonları açıklayacak olursak:

- Gizlilik, veri kriptolama anahtarını bilmeyen kişiler tarafından okunamadığı için sağlanmış olur.
- Kimlik tanımlama (authentication/identification), bilginin doğru kaynaktan alındığını onaylamak için kullanılır.
- Bütünlük(integrity), verinin transferinde hiç bir değişikliğe uğramadan son kullanıcıya ulaşa bilmesini sağlamak amaçlıdır.
- İnkâr edememe, bir mesajı gönderen kişinin o mesajı size gönderdiğini inkar edememesini sağlayan alındı onayları sağlar.

2.3.3 Bilgisayarda Kullanılan Kriptografileri Denetleyenler

Amerika kriptografiyi korunmanın en önemli parçalarından biri olarak görmektedir. Bu yüzden kriptografinin ithal ve ihracatında önemli şartlar koymuştur. Önem derecesi yüksek malzemeler için Amerika'nın ihracat politikası;

- İhracat Yönetimi Kanunu
- Silah ihracat kontrolü kanunu
- Nükleer-silahsızlanma

Kanunlarında belirlenmiştir. Amerikan yönetimi kriptografiyi bir askeri malzeme olarak görür ve bu da kriptografinin önemini açık aşikâr ortaya koyar.

Amerika'da kriptografi ihracını kontrol eden kuruluşlar şunlardır:

- Ticaret Bakanlığı'na bağlı İhracat Yönetimi Bürosu (BXA). İhracat Yönetimi kanunlarını düzenlemek için oluşturulmuştur.
- Eyalet departmanında Savunma Ticareti Kontrol Ofisi (DTC). Uluslarası Silah trafiği Kanunları (ITAR) tarafından ticari işleri kontrol etmek için yetkilendirilmiştir.

Bunlardan bazıları kriptografi işinde hafif kurallar koyarken bazıları bu konuda daha serttir. Örneğin BXA ithalat ve ihracat ihtiyaçlarına hafif kurallar uygular ama DTC'nin bu konuda oldukça sert kuralları vardır [22].

Ulusal Güvenlik Kuruluşu (NSA) ekonomik, askeri ve dış işleri ile alakalı olarak kriptografiye çok fazla önem verir.

Amerika'da Kriptografi tekniklerinizi ülke dışına ihraç edebilmek için Ürün Onayı (CJ) almanız gerekmektedir. Bu onayı alabilmek için ise ürün NSA' ya (Ulusal Güvenlik Kuruluşu) bırakılmalıdır. Eğer buradan izin çıkarsa o zaman ürün ihraç edilebilir [23].

2.3.4 Donanımla Kriptografi

Bazı Kriptolama çeşitleri vardır ki bunlar kriptoloji için özel donanım kullanır. Geçtiğimiz yıllarda yazılım kriptolamadaki artış nedeniyle ordu ve bazı ticari uygulamalar donanım tekniklerini tercih etmeye başlamıştır.
Donanım tekniklerine gerek duyulmasının sebepleri:

- Hız
- Güvenlik
- Kurulum kolaylığı

Genel olarak donanım teknikleri yazılım tekniklerinden her zaman daha hızlıdır. Bilgisayarınızdaki herhangi bir yazılım kriptolama tekniğinin fiziksel güvenliği yoktur. Kurumunuzdaki yabancı kişiler fark edilmeden algoritmalarınızla oynaya bilir. Bu bakımdan donanım teknikleri daha iyi bir güvenlik sağlamaktadır. Örneğin açılması zor olan kutular bu tür donanım tekniğidir.

Kriptolama tipleri:

1- Kendi başına çalışan kriptolama modülleri
2- Haberleşme bağlantıları için kriptolama kutuları
3- Kişisel bilgisayarlara takılan kartlar

Birinci yönteme yani Kendi başına çalışan kriptolama modüllerine örnek verecek olursak banka ve benzeri kuruluşlar tarafından kullanılan şifre yönetimi ve anahtar yönetimi gibi alanlar buna örnek teşkil eder.

İkinci yöntem ise haber adanmış kriptolama kutularıdır. Bu kutular en genelde iki nokta arasında bilgi transferini güvenli hale getirmek için kullanılır.

Üçüncü yöntemde yani Kişisel bilgisayarlara takılan kartların görevi hard disk'e gönderilen her türlü bilgiyi şifrelerler. İsteğe bağlı olarak USB gibi aygıtlara gönderilen bilgiler de kriptolanmak için ayarlanabilir.

Yazılım teknikleriyle dosyalar kriptolanır ve gerektiğinde dekriptolanır. Bu tekniklerin hepsi kriptolama yaparken kriptolama anahtarları kullanırlar. Bu yüzden dosyalarınızı disk gibi kolay bulunabilecek yerlerde değil daha güvenli yerlerde saklamanız gerekir. Kriprolama sonrası oluşan şifrelenmiş yerler ve kriptolama anahtarlarını da kaldırmanız gerekir. Özellikle kriptolama anahtarlarını içeren cihazların fiziksel güvenliğine dikkat edilmelidir [22].

İki tür kriptografi vardır:

- Simetrik kriptografi
- Asimetrik kriptografi

Simetrik kriptografi, şifre oluştururken ve şifreyi çözerken aynı kriptolama anahtarını kullanır. Bu yöntemin başka bir adı da Özel Anahtar Kriptografidir. Asimetrik kriptografi veya Açık Anahtar Kriptografi sistemi ise şifreleme ve şifre çözme için çift anahtar kullanır. Bu sistemdeki açık anahtar şifreleme için kullanılırken özel anahtar şifre çözme için kullanılır.

2.3.5 Yazılımla Kriptografi

Yazılımcılar tarafından yazılan kriptografilerdir. Yazılımla yazılan kriptografik algoritmaları kolay, değiştirebilinir ve donanıma göre çok daha karmaşık olabilir. En basitinden yeni çıkarılan kriptografiler yöntemi donanıma gömülerek her kullanıcıya teker -teker ulaştırmanız imkânsızdır. Bunun yerine kriptografi algoritmalarını yazılımla yazarak, internet üzerinden göndermeniz gerekir. Yazılımla yazılan kriptografi algoritmaları karmaşık, esnek, değiştirebilir ve ucuz olurlar. Bir başka avantajı ise kendi şirketinizde rahatça yazılımcılar tarafında yazılabilir olmaları.

2.3.6 İnternet Güvenliği Kriptografi

İnternet güvenliği için atılan büyük adımlardan biri RSA in bulunmasıdır. RSA 1978' de keşfedilmiş ve kendi yaratıcılarının isminin baş harflerini almıştır: Ronald Rivest, Adi Shamir ve Leonard Adleman.

RSA algoritmasının bir özelliği de kullanılan anahtarların uzunluklarının standart olmaması, uygulamaya göre her hangi bir uzunlukta olabilmesidir. Eğer kullandığınız anahtar yeteri kadar uzunsa o zaman yaptığınız işlemin yeteri kadar güvenli olduğu düşünülür. RSA en çok test edilen algoritmalardan biridir. Bilgi şifreleme dışında RSA dijital imza sistemlerinde de kullanılabilmektedir.

Simetrik kriptografi Açık anahtar kriptografi kullanımından daha hızlı çalışır. Peki, sebebi nedir? Şöyle ki Açık anahtar kriptografi de daha önceden belirttiğimiz gibi eğer güvenli bir sistem yapmak istiyorsanız o zaman anahtarınızın uzunluğu yeteri kadar uzun olmak zorunda. Uzun anahtar kullanınca da algoritmanın daha fazla işlemci gücüne ihtiyaç duyması nedeniyle bu algoritmalar bir az daha yavaş çalışır.

Özel anahtar kriptosistemi gibi bir açık anahtar kriptosistemi de, eğer anahtar olmadan şifreli yazıyı açmak imkânsız ise güvenlidir. Ayrıca açık anahtar bilgisiyle özel anahtarı bulmak imkânsız olmalı. Bu gün kullanımda olan bütün açık anahtar kriptosistemleri, çok güçlü bilgisayarlar tarafından bile, hatırı sayılır bir süre içerisinde çözülmesi imkansız belirli matematik problemlerine dayanır. Bu tip problemlerden biri çok geniş

bir sayının asal çarpanlarını bulmaktır. Bir sayının asal çarpanları sonuç olarak sayıyı veren asal sayılardır. Örneğin 91'in asal çarpanları 7 ve 13'tir. Fakat 899 gibi bir numarayı ele aldığınızda numara büyüdükçe asal çarpanlarını bulmayı hesaplamanın da zorlaştığını görebilirsiniz. RSA açık anahtar sistemine matematiksel olarak iki geniş asal sayıyı asal çarpanlarına ayırarak saldırabilirsiniz. Bu iki asal çarpanı bulursak bunları özel anahatar numarasında kullanabiliriz. Anahtar yaratmada geniş numaralar kullanmak bu yöntemi zorlaştırıyor ve pratikte asal çarpanlarına ayırmayı imkansız hale getiriyor. Örneğin; teorik olarak 512-bitlik bir anahtarın asal çarpanlarını 1 MIPS (saniyede 1 milyon işlem) lik bir bilgisayarla bulmanız mümkün fakat 420 bin yıl süreceği hesaplanmıştır. Buna rağmen 512-bitlik anahtarlar potansiyel olarak güçsüz olarak düşünülmektedir. 1024-bitlik anahtarlar pek çok amaç için yeterli derecede güçlü olarak görülmektedir.

Eğer gelecekte geniş rakamların asal çarpanlarını bulmak için yeni ve hızlı bir yöntem bulunursa RSA sisteminde güvensiz hale gelecektir. Fakat şu an RSA bu tip saldırılardan hiçbir şekilde etkilenmemektedir. Sebebi ise, anahtar için kullanılan 'modulus' boyutu gelişen teknolojiye ayak uyduracak şekilde arttırılabilir [22].

Peki, anahtar uzunluğu seçerken nelere dikkat etmeliyiz? İlk başta güvenliğe dikkat etmeliyiz ama başka bir kriter de performanstır. Anahtar boyutu büyüdükçe bilgiyi şifrelemede ve şifre çözmede ihtiyaç duyduğu işleme gücüde artar. Bir açık anahtar sistemini kırmanın bazı yolları vardır. Bunlardan en sistematik olmayanı anahtarı çaldırmaktır. Yani anahtarı bilgisayarınızda saklıyorsanız veya bilmemesi gereken kişilere söylemişseniz o zaman şifrenizin istenmeyen kişiler tarafından kırılması olasıdır.

Bazı saldırganlar matematiksel metotta bir kural açığı bularak anahtarı kırmayı umarlar. Ayrıca RSA ve Diffie-Hellman`in çeşitli uygulamalarına karşı gerçekleştirilebilen yeni zamanlama saldırıları gibi, yeni tipte saldırılar hacker'lar tarafından geliştirilmektedir.

2.3.7 RSA Algoritması

Asimetrik kriptografi'de bir özel anahtar, onu eşleyen açık anahtarın tersidir. Bir açık anahtar algoritmasının güvenli sayıla bilmesi için açık anahtarından özel anahtarın uzun bir süre içinde bulunamaması gerekir. Bu konuda RSA'nın bir problemi yoktur çünkü büyük bir sayının asal çarpanlarını bulmak çok zordur . RSA algoritmasındaki diğer önemli bir özellik, aralarında asal çarpanlardır (relatively prime factors). Eğer iki sayı birbirlerinin çarpanları şekilde yazılamıyorlarsa bu iki rakama 'aralarında asal' denilir. Örneğin, 3 ve 7 rakamları aralarında asaldırlar çünkü 3, 7'nin bir çarpanı değildir. Fakat, 3 ve 6 aralarında asal değildirler, zira 3, 6'nin bir çarpanıdır (6=3*2).

RSA algoritması modüler matematik kullanır. Yani, sadece sıfırdan başlayan ve belirli bir sayıya kadar (o sayı dâhil değil) olan sayılar (modulus) kullanılır. Eğer modulus olarak 20 kullanılıyorsa denklik bağıntısı gereği 20 sayısı yerine 0 (mod 20) kullanırsınız, 21 yerine de 1 (mod 20) kullanırsınız. Benzer şekilde 39 sayısı 19 (mod 20) , 40 da 0 (mod 20) olarak yazılır [22].

Bu şekilde olan denklemler yardımıyla bir rakamın modüler terimlerle ifade ediliş şeklini bulabiliriz. Matematikte 1 sayısını üreten iki rakam bir diğerinin tersidir. Modüler matematikte de aynı kural geçerlidir. Örneğin, eğer modulus 20 ise, 3 ve 7 rakamları birbirinin tersidir. Bunun sebebi, 3*7=21 ve 21 rakamı 1 mod 20 olarak gösterilir.

Kullandığınız modulus yarattığınız her bir anahtar seti için farklı olabilir. Bir modulus seçmek için rastgele iki büyük asal sayı seçersiniz, p ve q. İkisini birbiriyle çarptığınızda, çıkan sonuç, n, sizin kriptolama ve dekriptolama modulus'unuzdur. Üslü ifade (exponent), e, olarak bilinen, n'den küçük değerde bir açık anahtar değeri seçersiniz. Bu sayının asal olması gerekmiyor fakat tek sayı olmalı. Açık anahtar değeri, e, (p-1)(q-1)'e ilişkili asal olmak zorundadır.

Bir düzyazı mesajı (m) şifreli yazıya (c) çevirmek için bu formülü kullanırsınız. Elde edilen şifreli yazıyı açabilmek için açık anahtarın tersini bulmalısınız. Buda özel anahtar (d). Bu formül ile de dekriptolayabilirsiniz. d'yi bulmak için modulus n'yi değil farklı

bir modulus kullanırsınız, (p-1)(q-1). Özel anahtar açık anahtarın tersi olduğundan bu denklemde bir sorun yaşanmaz.

Açık anahtar e ve özel anahtar d arasındaki ters ilişki RSA algoritmasının şifreli yazıyı orijinal mesaja başarılı bir şekilde çevirmesini sağlar. Eğer özel anahtar d'yi bilmiyorsanız açık anahtar e ile kriptolanmış şifreli yazıyı açamazsınız. Fakat açık anahtar e sizde ise ve ayrıca modulus n'nin asal çarpanları p ve q'yu da biliyorsanız bu denklemi kullanarak özel anahtarı kolayca bulabilirsiniz. Bu yüzden p ve q'yu gizli tutmak önemlidir.

Eğer sadece açık anahtar e'yi ve modulus n'i biliyorsanız özel anahtarı hesaplayabilmeniz için önce modulus un asal çarpanlarını bulmanız gerekir. Fakat yeterli derecede büyük sayılar kullanıldığında asal çarpanların bulunması nerdeyse imkânsızdır [22].

Küçük rakamlar kullanan örnek bir RSA algoritması örneği inceleyelim. Diyelimki açık anahtar değeriniz, e, olarak 3'ü seçtiniz. p=5 ve q=11 seçip birbiriyle çarparak modulus n'i hesapladınız. p ve q için geçerli sayılar seçtiğinizi test etmek içinde e nin (p-1)(q-1)'e ilişkili asal olduğunu kontrol ettiniz.

Açık anahtar e'nin tersi olan özel anahtar değeri d'yi bulmanız gerekiyor. 27 rakamı bu denklemi sağlayacaktır (3*27=81=1 mode 40), bu yüzden d=27 dir. Şimdi açık anahtar e nin neden (p-1)(q-1)'e ilişkili asal olduğunu daha iyi anlamışsınızdır.
Eğer e (p-1)(q-1)'in bir çarpanı olursa 1 mod (p-1)(q-1)'in çarpanı olamaz. Ve eğer e 1 mod (p-1)(q-1)'in çarpanı değilse modüler olarak tersi yoktur ve bu sebepten özel anahtar değildir.
Bundan sonra RSA ile bir düzyazı metni şifrelenebilir. Düzyazı mesajınız, m, 2 olsun. m'i kriptolamak için bu formülü kullanırsanız, çıkan şifreli yazı, c, 8 olacaktır. Şifreli yazıyı bu formül ile açtığınızda sonuç 2 mod 55 olarak orijinal düzyazıyı verecektir.

Mesajları açık anahtar algoritması ile kriptolamak ve açmak simetrik sistemden çok daha fazla işlem gücü harcar. En hızlı RSA çipi 512-bitlik asallar kullandığında saniyede 600kbit çıktı verebilir. Benzer DES donanım uygulamaları 1000 ile 10000 kat

daha hızlıdır. Ve DES yazılım uygulamaları RSA algoritmasından 100 kat daha hızlı kriptolama yapar.

Açık anahtar kriptolama yavaş olmasına rağmen daha güvenlidir. Örneğin, eğer simetrik kriptolama kullanarak bir mesaj gönderirseniz, her alıcıya özel anahtarınızı göndermek için güvenli bir yol bulmanız gerekir. Simetrik kriptografi kullanımında kendiniz ve her alıcı arasındaki haberleşme için kullanılacak anahtar konusunda özel olarak anlaşmanız gerekir. Fakat açık anahtar kriptografi çok yönlüdür. Açık anahtarı bilinen herhangi birine, önceden bir anlaşma yapmadan gizli mesajlar gönderebilirsiniz. Açık anahtar sistemleri çoğu zaman özel anahtarın güvenli olarak gönderilmesi için dijital zarf (digital envelope) yaratılmasında kullanılırlar [23].

3. MALZEME VE YÖNTEM

3.1 RC4 ŞİFRELEME ALGORİTMASI

3.1.1 Tarihçesi

RC4 (Ron's code # 4 or Rivest) şifreleme algoritması uygulamalardan en sık kullanılan algoritmadır. 1987 yılında Ron Rivest tarafından RSA Labaratuvarında geliştirilmiştir. Bu paylaşılan anahtar akışı şifreleme algoritmasıdır. İlk kez 1994 yılında tüm dünyayla paylaşılmıştır. Şimdiye kadar resmi olarak yayınlanmamıştır. Günümüzde RC4 şifreleme algoritmasını en yaygın olarak kullanılan teknolojiler Oracle SQL, Microsoft Windows WEP, WPA, SSL ve TLS'tir. Bu kadar yaygın olarak kullanılmasının başlıca sebepleri yüksek hız, basitliği ve kolay uygulanabilirliğidir.

RC4 hem yazılım ve hem de donanım da çok rahat uygulanabiliyor. RC4, RC2 teknolojisi üzerine geliştirilmiştir. RC4 değişken anahtar boyutlu şifreleme algoritması olduğu için veriyi çok hızlı şekilde şifreleme yeteneğine sahiptir. RC4 33 MHz'lik bir makinede 1 Mbyte/sec şifreleme hızına ulaşıyor [24].

3.1.2 Genel Özellikler

RC4 algoritması 256 byte'lık durum tablosunu doldurmak için 1-den 256-byte'a kadar değişken uzunluklu anahtar'lar kullanmaktadır. Bu durum tablosu sahte rastgele byte'ların üretilmesi için kullanılır. Bu elde edilen sahte rasgele byte'lar da düz metin'le XOR'lanarak şifrelenmiş metin elde edilir. Durum tablosundakı her eleman en az bir kere yer değiştirmiş olmalıdır.

RC4 algoritması çoğu zaman 40 bit ile kısıtlanmıştır ama bazen bu kısıtlama 128 bit de olabilir. RC4 çoğu zaman ticari uygulamalarda kullanılır bunlara örnek verecek olursak LOTUS notes ve Oracle örnek gösterilebilir.

RC4 algoritması iki aşamadan oluşur, anahtar oluşturma ve şifreleme. Anahtar oluşturma bu aşamaların ilkidir ve en zor aşama bu aşamadır. N anahtar'lık bir üretim sırasında şifreleme anahtarı kullanılır ayrıca bu işlemleri yapabilmek için iki tane dizi de kullanılır. Bu işlemler sırasında yine N sayısı kadar bir karıştırma operasyonu geçirilir. Bu karıştırma işlemi byte'ların yer değiştirilmesini ve formüllerin hesaplanması işlemlerini içerir [25].

Ve ensonda elde edilen durum tablosundan şifreleri oluşturmak işlemidir. Bu veriler de düz metin ile XOR'lanarak şifrelenmiş değerleri elde edilir. Bu XOR'lama işlemi sırasında bitler kendi aralarında XOR'lanır. Eğer her iki bit de bir birinden farklı ise sonuç 1 çıkar, tam tersi yanı iki bit de aynı ise o zaman sonuç 0 çıkmaktadır [25].

3.1.3 Algoritmanın Özellikleri

RC4 ile şifrelemeye başlamadan önce ihtiyacınız olacak ilk şey 40 bit 256 bit arasında bir anahtara ihtiyacınız olacak. Minimum değer olan 40 bit ASCII de 5 karakteri temsil etmektedir, örneğin "pwd12" 40 bitlik bir veridir. Pwd12-nin ASCII karşılığı şu şekildedir "01110000011101110110010000110001001100 10". Bundan sonraki kısım işin anahtar planlaması algoritmasıdır KSA. KSA Algoritmasının alınmış kodu aşağıdaki gibidir:

```
-------------------------------------------------------------------------------
      for i from 0 to 255
      S[i] := i
      endfor

      j := 0
      for i from 0 to 255
      j := (j + S[i] + key[i mod keylength]) mod 256
      swap(S[i],S[j])
      endfor
-------------------------------------------------------------------------------
```

KSA 0-dan 255-e kadar verisi olan 256 boyutunda bir dizi oluşturmaktadır [26].

0	1	2	...	i	i+1	...	253	254	255

S dizisinin her elemanı yine ayni dizinin j. elemanı ile yer değiştirmektedir. J. eleman ise asağıdaki formülle hesaplanmaktadır:

```
j = [(j + S(i) + key[i mod keylength]) mod 256]
```

j'ye başta 0 atanmıştır. S[i] dizinin o andaki gerçek verisidir. *key[i mod keylength]* değeri ya 1 ya da 0 ola bilir. Örneğin biz i=52 de olalım ve "pwd12" anahtarın uzunluğu da 40 olduğundan 52 mod 40 =12 olacaktır. Anahtar dizisi yani "pwd12"'nin 13. elamanı 0 olduğundan *i mod keylength=0* olacaktır. (anahtar dizinin 13. elemanına bakmamızın sebebi dizinin 0 dan başlamasıdır). Bu adımın işlemesi ile alakalı genel bir örnek verecek olursak. Diyelim ki i=0 olsun, en başta olduğumuzdan j=0 durumundadır. S[i] ise dizinin s[0] elemanına dolayısıyla 0'a eşittir. key[0 mod 40]=key[0] key dizisinin de sıfırınca elemanı "p" harfidir dolayısıyla p'nin de ASCII 112'dir [26].
J=[(0+0+112)mod 256]=112 sonucu bulunmaktadır. Böylece S dizisinin 0. ve 112. elemanı yer değiştirir. Dizinin yer değiştirdikten sonraki hali aşağıdaki gibidir.

112	1	2	...	111	0	113	114	...	255

KSA algoritması bittikten sonra dizinin en son hali aşağıdaki gibidir:

[101, 124, 172, 10, 166, 26, 46, 91, 2, 137, 39, 243, 253, 25, 3, 30, 47, 238, 196, 38, 94, 149, 15, 32, 248, 51, 158, 150, 106, 183, 67, 219, 95, 177, 138, 152, 13, 188, 118, 108, 207, 151, 41, 142, 236, 103, 55, 72, 20, 244, 216, 14, 168, 90, 4, 42, 153, 64, 250, 129, 97, 225, 87, 199, 204, 100, 16, 249, 191, 82, 43, 131, 24, 169, 69, 54, 96, 77, 255, 84, 1, 143, 242, 123, 21, 93, 61, 102, 224, 107, 109, 79, 80, 23, 229, 6, 156, 181, 105, 159, 33, 141, 18, 104, 9, 56, 233, 178, 127, 111, 135, 206, 202, 128, 31, 71, 211, 222, 45, 66, 163, 189, 167, 201, 232, 17, 251, 198, 170, 155, 115, 57, 228, 98, 190, 76, 59, 239, 37, 147, 180, 240, 197, 200, 19, 0, 213, 99, 125, 44, 195, 164, 176, 121, 220, 212, 86, 186, 34, 214, 230, 254, 40, 203, 194, 231, 162, 226, 187, 116, 208, 22, 68, 88, 192, 140, 205, 234, 119, 83, 136, 63, 12, 112, 217, 154, 184, 81, 70, 35, 174, 78, 241, 179, 210, 215, 49, 144, 130, 48, 133, 7, 209, 92, 73, 193, 28, 75, 117, 223, 50, 113, 114, 148, 173, 29, 53, 160, 8, 139, 246, 65, 252, 161, 221, 185, 27, 36, 11, 110, 237, 165, 5, 182, 145, 171, 120, 157, 134, 175, 122, 58, 235, 52, 62, 126, 85, 60, 132, 74, 245, 227, 218, 89, 247, 146]

Yukarıdaki işlemler bittikten sonra RC4 de diğer kısma geçiyoruz. Bu kısım pseudo-random generation algoritmasının(PRGA) gerçekleştirimini içermektedir. PRGA algoritmasının gerçekleştirim kodu aşağıdaki gibidir [26].

```
i := 0
j := 0

while GeneratingOutput:
i := (i + 1) mod 256
j := (j + S[i]) mod 256
swap(S[i],S[j])
output S[(S[i] + S[j]) mod 256]
endwhile
```

PRGA-ya başlarken KSA da sıralanmış olan S dizisini alıyoruz.Yani, Key Scheduling Algorithm (KSA) 'i Pseudo-random Generating Algorithm (PRGA) takip ediyor.

S dizisinin indeks i'deki değeri ile yine S dizisinin J indeks'indeki değer yer değiştiriliyor. Toplam operasyon sayısı anahtar dizisinin uzunluğuna eşittir. Bu operasyonlar sırasında S dizisinin her elemanı en az bir kere yer değiştirmiş olmalıdır.
Şimdi de PRGA algoritması için örnek gerçekleştirelim. Koda bakılırsa ilk adımdan sonra i=1 eşit olduğu anlaşılacaktır. J değeri ise (0 + S[1]) mod 256 değerine eşit olacaktır. S[1]=124 olduğundan j değeri de 124 eşit olacaktır. En sonda S dizisinin 1. elemanı ile 124. elemanı yer değiştirir. Dizinin birinci iterasyondan sonraki halı şu şekilde olacaktır:

[101, 232, 172, 10, 166, 26, 46, 91, 2, 137, 39, 243, 253, 25, 3, 30, 47, 238, 196, 38, 94, 149, 15, 32, 248, 51, 158, 150, 106, 183, 7, 219, 95, 177, 138, 152, 13, 188, 118, 108, 207, 151, 41, 142, 236, 103, 55, 72, 20, 244, 216, 14, 168, 90, 4, 42, 153, 64, 250, 129, 97, 225, 87, 199, 204, 100, 16, 249, 191, 82, 43, 131, 24, 169, 69, 54, 96, 77, 255, 84, 1, 143, 242, 123, 21, 93, 61, 102, 224, 107, 109, 79, 80, 23, 229, 6, 156, 181, 105, 159, 33, 141, 18, 104, 9, 56, 233, 178, 127, 111, 135, 206, 202, 128, 31, 71, 211, 222, 45, 66, 163, 189, 167, 201, 124, 17, 251, 198, 170, 155, 115, 57, 228, 98, 190, 76, 59, 239, 37, 147, 180, 240, 197, 200, 19, 0, 213, 99, 125, 44, 195, 164, 176, 121, 220, 212, 86, 186, 34, 214, 230, 254, 40, 203, 194, 231, 162, 226, 187, 116, 208, 22, 68, 88, 192, 140, 205, 234, 119, 83, 136, 63, 12, 112, 217, 154, 184, 81, 70, 35, 174, 78, 241, 179, 210, 215, 49, 144, 130, 48, 133, 7, 209, 92, 73, 193, 28, 75, 117, 223, 50, 113, 114, 148, 173, 29, 53, 160, 8, 139, 246, 65, 252, 161, 221, 185, 27, 36, 11, 110, 237, 165, 5, 182, 145, 171, 120, 157, 134, 175, 122, 58, 235, 52, 62, 126, 85, 60, 132, 74, 245, 227, 218, 89, 247, 146].

Bundan sonra yeni değiştirdiğimiz dizinin elamanlarını topluyoruz, yani 232+124 = 356. Elde ettiğimiz 356 değerini 256 göre mod aldığımız zaman 100 değerini elde ediyoruz. Sonra S dizinin 100. elemanını yani 33'u yazdırıyoruz. Sonra girdi dizisinin i-ci elamanına bakıyoruz "Math 310 Proves!" dizisinde bu değer "M"harfine eşit. M harfinin UNICODE değerini alıyoruz bu da 77'ye eşit. Elde ettiğimiz bu 77 değeri ile 33 değerini AND'ledikden sonra 108 değerini elde ediyoruz. Sonra Unicode karakter değeri 108 olan veriyi alıyoruz bu da "1"'e eşit."1"'in de hex gösterim şekli 6C'dir. O zaman çıktı dizimiz şu şekilde oluyor [26]:

"6CA86FE3CBC33C162595C3E78B9C97BC"

3.1.4 RC4 Algoritmasının güçlü yönleri

- Her hangi bir değerin tabloda nerde olduğunu bulabilme güçlüğü.
- Her RC4 anahtar partikülü sadece bir kere kullanılabilir.
- Başka bir önemli avantajı ise şifreleme işleminin şifre çözme işleminden 10 kat daha hızlı olmasıdır.

3.1.5 RC4 Algoritmasının zayıf yönleri

RC4 algoritmasının durum tablosu analitik ataklar sırasında çok zayıf kalmaktadır.
Zayıf ANAHTAR: bu anahtarlar şifreleme tarafından tanımlanmıştır ve bu anahtarlar belli koşullar altında çok fazla üretilebilen anahtarlardır.

3.1.6 RC4 Algoritmasinin Terminolojisi

• RC4 : Ron's code # 4 or Rivest
• Cipher: Şifre üretme ve şifre çözme sırasında kullanılan kriptografik algoritma
• Symmetric key algorithm: Şifreleme ve Şifre çözme için aynı anahtarı kullanan algoritma

• State table: 1 den 256 byte kadar verilerden oluşan tablo. Psedo-random byte'ların üretimi için kullanılan byte'ları içermektedir. Psedo–random anahtarlar da düz metin ile XOR'lanarak şifrelenmiş metin oluşturulur

3.1.7 RC4 Algoritmasının Performansı

Şekil 4.1'de bir saniyede bir milyon bit için gerekli olan MIPS'lerin sayısını her üç implementasyon için gerçekleştirimi gösterilmiştir [26].

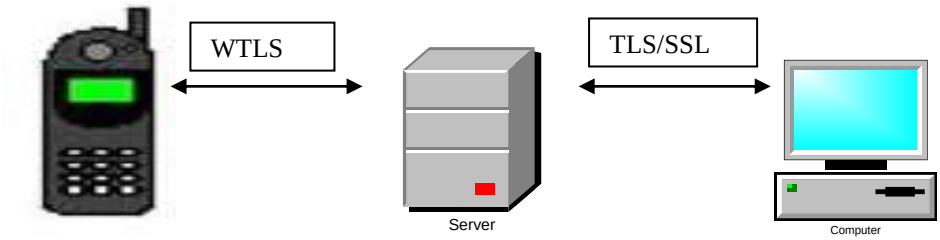
	MIPS	RAM
Optimized MIPS Assembly	2.5	hiçbiri
RC4 Operation Support UDI Primitives	1.75	0 bytes
RC4 Key Byte Generator UDI Accelerator	0.22	256 bytes

Şekil 3.1: RC4 Algoritma Performansı

4. BULGULAR

Önceki bölümlerde de bahsettiğim gibi, günümüzde internet hızının artması ve dijital aygıtlarda hızlı gelişim geleneksel e-ticareti m-ticarete uygulamak için kolay imkanlar sunuyor. M-ticaretin her yerde ve her zaman erişebilir olmasından dolayı avantajları çok yüksektir. Bununla beraber bazı güvenlik sorunları ortaya çıkıyor. Bu güvenlik açıklarından en önemlisi "Wap Gap"tır.

Bu soruna WAP yeni teknolojilerde çözüm getirmiş ama şirketlerin özel verilerinin güvenliğini korumak için kendileri mobil işlemler için uçtan-uca güvenlik uygulamaları geliştiriyorlar. Bu uygulamaları geliştirmek için en iyi ortam j2me'dir.Bu j2me'nin sunduğu imkanlardan kaynaklanıyor. Mobil aygıtlar arasındaki veri iletişimi aşağıdaki resimde gösterildiği gibidir [27].



Şekil 4.1: Mobil aygıtlarda veri iletişimi

WTLS(Wireless Transport Layer Security) : WAP'ın güvenlik protokoludur.

WAP Geçit: WAP protokolü kullanan kablosuz aygıtlar ve www arasında duruyorlar. Kablosuz aygıtlardan gelen içeriği alıp www(worl wide web)'ye uygun biçime dönüştürüyorlar ve www'den gelen içeriği alıp kablosuz aygıtlar için uygun biçime dönüştürüyorlar .

TLS/SSL: Network katmanlarından Transport Layer de uçtan uca güvenlik protokolüdür.

Bu durumda mobil ağlarda veri iletişimi yapılırken "Wap Gap" diye bir problem ortaya çıkıyor. Bu, WTLS tarafından kodlanmış verinin Wap geçitte çözümlenerek yeniden SSL'e uygun biçimde kodlandığında, ortada saf verinin çok kısa süre de olsa bulunmasıdır. Bu durumda mobil ağların çok önemli verilerinin başkaları tarafından erişilmesinin karşını almak gerekiyor [28].

"Wap gateway"e alternatif bir çözüm uçtan uca güvenliktir (end-to-end security). Uçtan uca güvenlikte verinin bir uçta kodlanması ve diğer uçta çözümlenmesidir. Uçtan uca güvenlik geliştirmek için en iyi ortam J2ME'dir. Bunun bir çok yararları var: Platform bağımsız, jvm'in güvenlik avantajları, bellek, güç ve bant genişliği yönetimi v.s.

Bölüm 1'deki bilgileri dikkate aldığımızda mobil ortamda verinin güvenliğinin ne kadar önemli olduğu anlaşılıyor. Bu yüzden, kurumsal şirketler mobil ağlar üzerinde veri alış verişi yapan uygulamalarını kendi yöntemleri ile şifreliyorlar. Bu şifrelemeyi yapmak için kullanılan 2 yöntem var: Simetrik ve asimetrik şifreleme. Bölüm 2'de bunların detaylarından daha derinden bahsedilmiştir. Şimdi bu iki mekanizmanın birbirlerine göre avantajları ve dezavantajlarını inceleyelim.

Simetrik şifrelemenin avantajları ve dezavantajları şöyledir.

Avantajları:

- Algoritmalar hızlıdır.
- Algoritmalarin donanımlar üzerinde gerçekleştirimi basittir.
- Gizlilik hizmeti verir

Dezavantajları:

- Güvenli anahtar değişim zor
- Ölçeklenebilir değil
- Birbirlerini tanımayan taraflar arasında iletişim zordur
- Bütünlük ve kimlik doğrulama hizmeti zordur

Asimetrik şifrelemenin avantajları ve dezavantajları şöyledir.

Avantajları:

- Anahtar yönetimi ölçeklenebilirdir
- Kriptoanalize karşı dayanıklıdır
- Kimlik doğrulama, bütünlük ve inkar edememeyi sağlıyor

Dezavantajı:

Simetrik şifreleme mekanizmasına göre algoritmasının çalışması çok daha yavaştır.

Şimdi yukarıdaki anlatılanları dikkate aldığımızda ve Bölüm 1'de anlatılanları incelediğimizde, önerilen çözümün diğerlerine göre avantajı daha iyi anlaşılabilir. Şimdiye kadar incelediğim uçtan-uca güvenlik konusunda yapılmış çalışmalarda hep asimetrik şifreleme yöntemleri kullanılmıştır. Bu tezde önerilen çözüm ise simetrik şifreleme yöntemini, RC4 şifreleme algoritmasını taban almıştır. Bu tezde anlatılan çözüm önerisinin, bu konuda yapılımış çalışmalara göre avantajları şunlardır.

- Şifreleme algoritmasının hızlı olması.
- Gerçekleştiriminin basit olması.

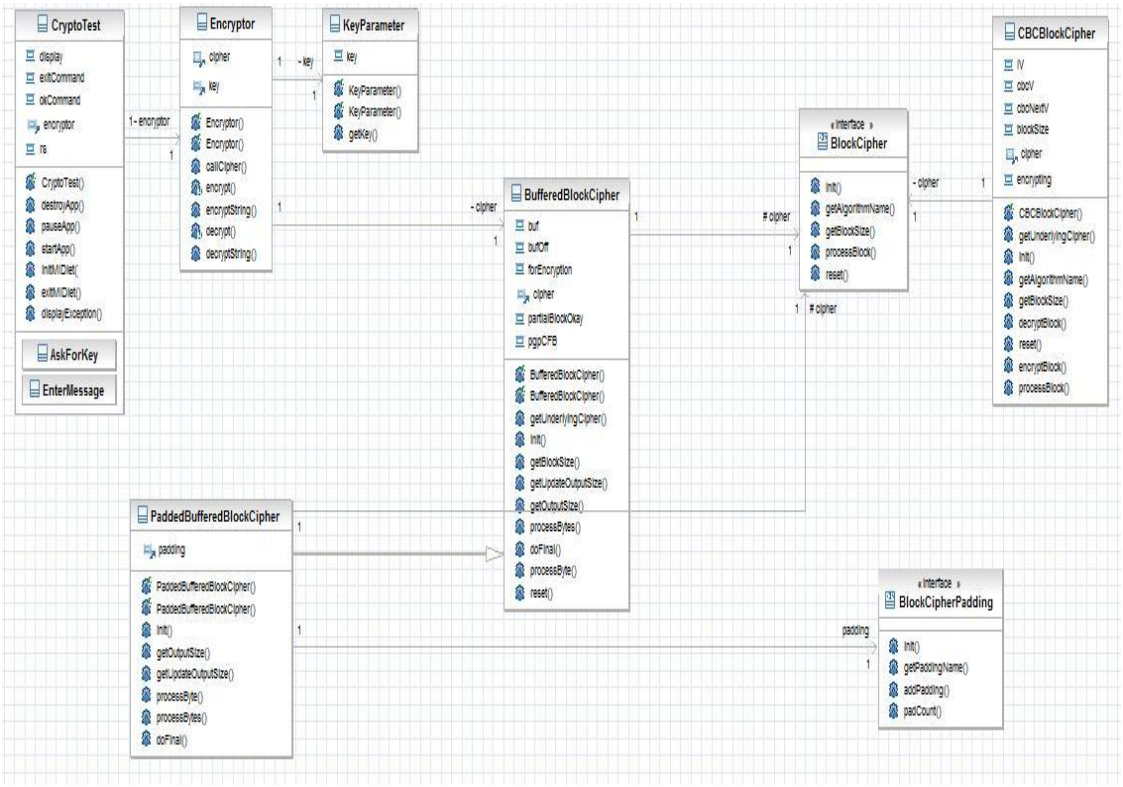
Önerilen çözüm iki aşamadan oluşmaktadır. İlki, mobil aygıt üzerinde çalışacak olan J2ME ile geliştirilen uygulama. Bu uygulama mobil aygıttan gönderilen veriyi kullanılan RC4 şifreleme algoritması yardımı ile önceden paylaşılan şifreleme anahtarı ile şifreliyor ve mobil aygıtın isteğine yanıt olarak gelen verini de aynı yöntemle okuyarak düz metine çeviriyor. İkincisi, ise J2EE ile gerçekleştirilen web uygulamasıdır. Bu uygulama kendisine gelen istekleri RC4 şifreleme algoritmasını kullanarak okuyor ve aynı şekilde şifreleyip karşı tarafa gönderiyor.

Uygulamanın mobil kısmında MIDP 2.0 profili kullanılmıştır. MIDP 2.0 cep telefonları, çağrı cihazları ve kişisel dijital ajandalar gibi küçük kapasiteli araçlarda için uygulama geliştirmede başlangıç noktasını oluşturan bir profildir. Sun tarafından, mobil cihazlarda çalıştırılacak uygulamalar için geliştirilmiştir.

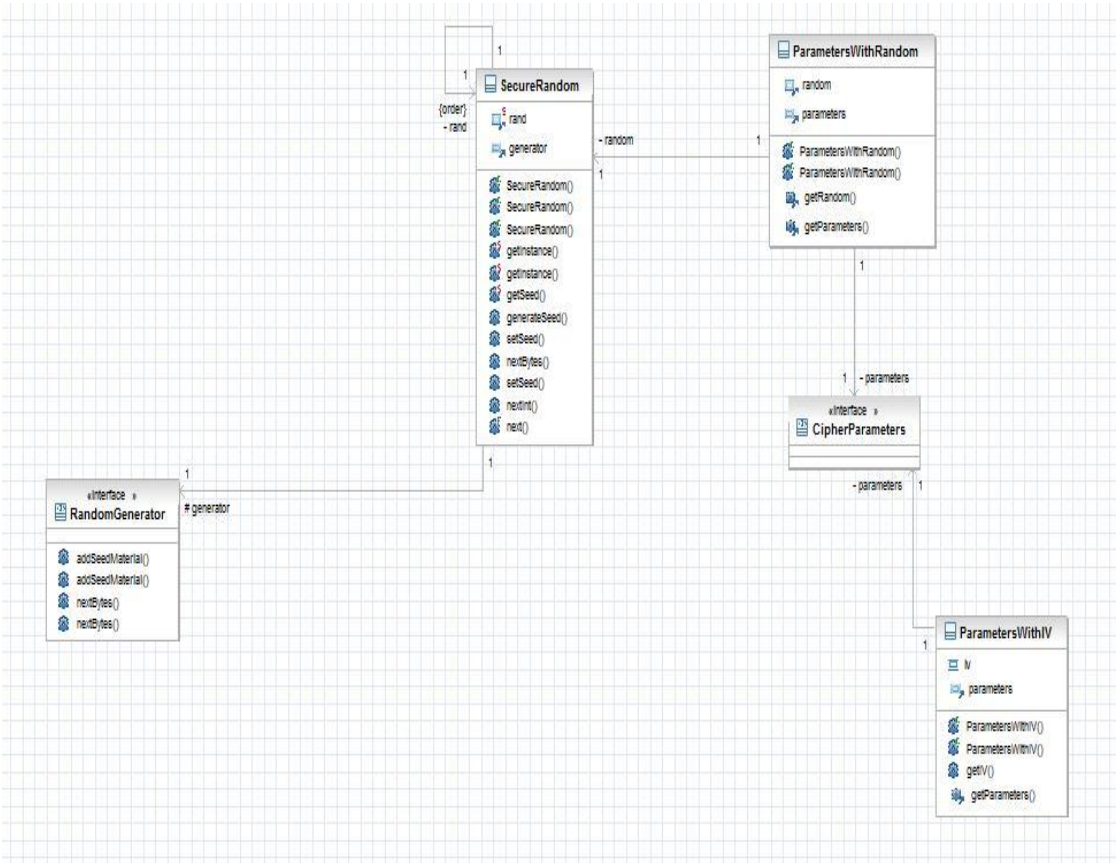
MIDlet yazarken en önemli konu, yapılan uygulama ile mobil kısımın nasıl konuşacağıdır. Uygulamanın, mobil kısımı ile HTTP protokolü kullanarak iletişim kurmasını sağlandı. Bunun için bir web uygulaması geliştirildi ve bu uygulamaya mobil taraftan iletişim kurabilmek için bir http bağlantısı kullanıldı [29]. MIDlet yazarken en önemli 2. konu ise, şifreleme algoritmasını uygulamaktır. Bunun için açık kaynak olan bouncycastle kütüphanesini kullanıldı [30]. Uygulamanın mobil kısmında Sun JavaTM Wireless Toolkit for CLDC Version 2.5.2 yardımcı aracını kullanıldı [31]. Bu araç J2ME geliştiriciileri için MIDP uyumlu uygulama geliştirmek için örnekler, dokümanlar ve simülasyon ortamını sunmaktadır.

Uygulamanın web kısmında ise JBoss web sunucusunu kullanıldı. Bu sunucu java ile yazıldığından tercih sebebi oldu. Böylelikle, tüm uygulama Java 2 platrormu üzerinde gerçekleştirimini sağlamış oldu.

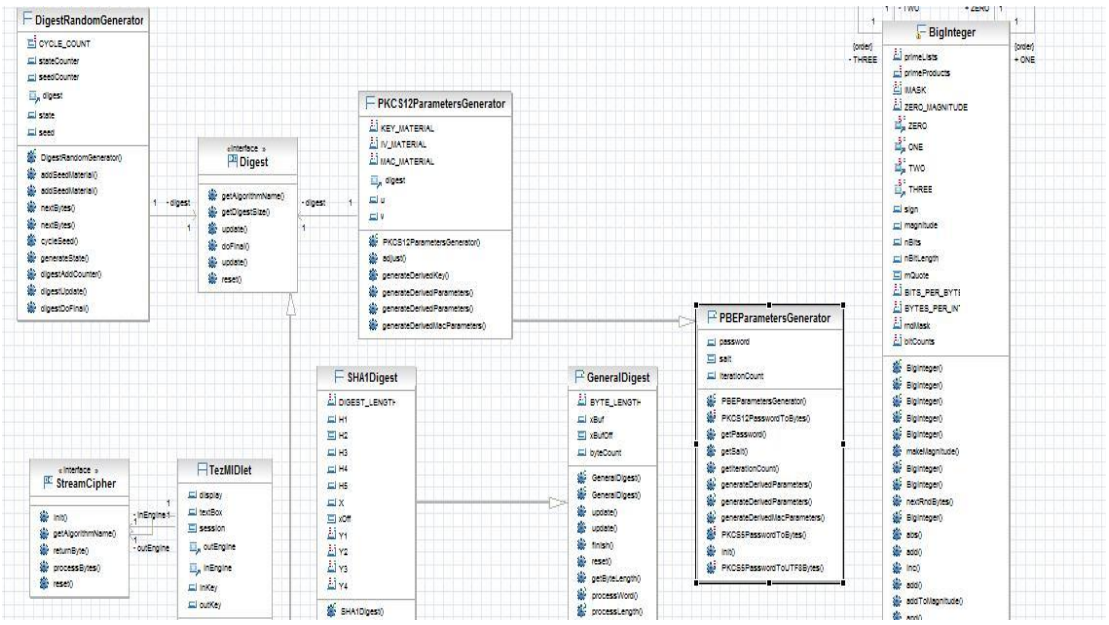
Uygulamada kullandığım class'ların UML class diagramları aşağıdaki resimlerdeki gibidir.



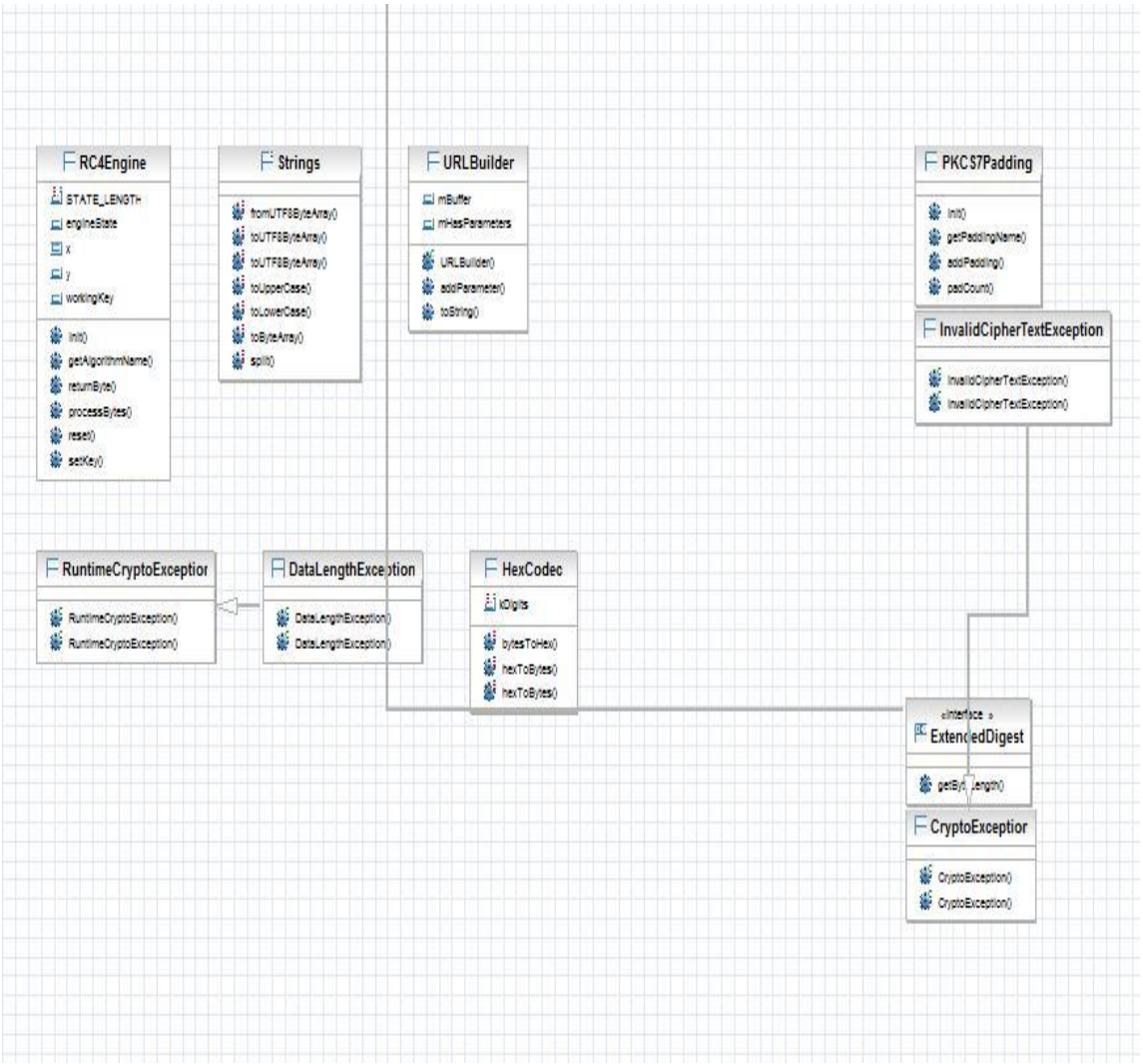
Sekil 4.2 :Uygulamanın class diagramı-1



Sekil 4.3 :Uygulamanın class diagramı-2



Sekil 4.4 :Uygulamanın class diagramı-3



Sekil 4.5 :Uygulamanın class diagramı-4

Kullanılan Bouncycastle kütüphanesinde şifreleme StreamCipher arayüzü tarafından sağlanıyor. İlklemek için init() metodunu kullanmak gerekli. Şifreleme ve deşifreleme için processBytes metodu kullanmak gerekli. init() fonksiyonuna ilk parametre “true” olmalıdır, eğer şifreleme yapıyorsa, deşifreleme için ise “false” kullanılıyor. İkinci önemli parametre ise anahtardır, bu KeyParameter nesenesinde tanımlanıyor.

Veriyi şifrelemek için, bizim sadece şifrelenmiş metin dizisini tutmak için bir dizi oluşturmanız gerekiyor. Şifreleme gerçekleştirmek için processBytes () fonksiyonu çağırmak gerek ve processBytes () fonksiyonuna düz metin dizisi, onun başlangıç

indeksi, işlenecek baytların sayısı, şifrelenmiş metin dizisi ve şifrelenmiş metin dizisine nereden başlayacağı indeks verilmeli.

Daha sonra veri 16'lık tabanda şifrelenmiş metine çevrilerek karşı tarafa gönderiliyor. Sistem hem veri gizliliği hem de kimlik doğrulama sağlıyor. RC4 simetrik şifreleme algoritması olduğu için, şifreleme ve deşifreleme için aynı anahtar kullanılıyor. MIDlet ve Servlet iki anahtar kullanıyorlar. Her bir yön için bir anahtar kullanıyorlar. Bir anahtar veriyi MIDlet'de şifrelemek ve Servlet'de deşifrelemek için kullanıyor. Diğer anahtar ise veriyi Servlet'de şifrelemek ve MIDlet'de deşifrelemek için kullanılıyor.

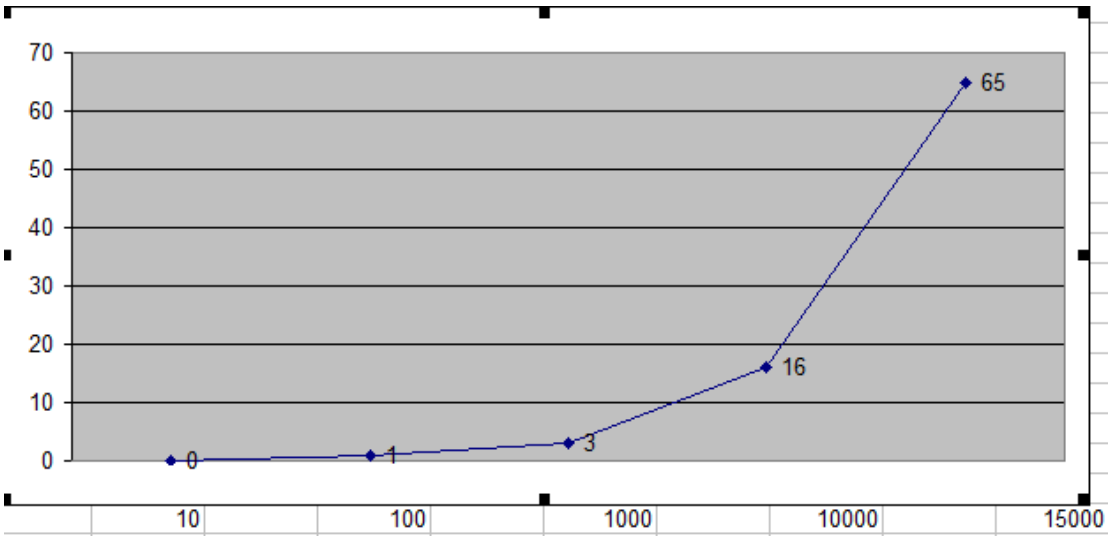
5. TARTIŞMA VE SONUÇ

Tezin uygulama kısmında Java Platformunun 2 bileşenini kullanıldı. J2ME ve J2EE. Önce test için java uygulamalarını destekleyen cep telefonuna mobil uygulama yüklenildi. Mobil uygulamayı cep telefonuna kurmanın 2 yöntemi vardır:

1. Aygıttan aygıta aktarımla,
2. İnternetten indirme ile.

Bu tezdeki mobil uygulama için internetten indirme yöntemi kullanıldı.Test için Nokia 6300 markalı cep telefonunu kullanıldı. Daha sonra cep telefonundan verilerin şifrelenerek java web uygulamasına gönderilip-alınması için java web uygulamalarını destekleyen bir java web sunucusuna ihtiyaç duyuldu. Bunun için java web uygulamasını destekleyen Apache Tomcat web sunucusunu kullanıldı.

RC4 algoritmasının yararlarına geldikte ise günümüzde cep telefonu kullanımı çok olduğu için, cep telefonlarından gönderilen verilerin miktarı çok oluyor ve bu da şifreleme ve şifre çözmede önemli bir etken oluyor. Aşağıdaki resimde RC4 algoritmasının şifreleme çıktısı görülmektedir.



Sekil 5.1 : RC4 algoritmasının şifreleme çıktısı

Yukarıdakı resimde, X ekseni şifrelenmek istenen verinin büyüklüğünü (byte olarak) ve Y ekseni ise verinin şifrelenmek için geçen zamanını gösteriyor (ms olarak). Şifreleme algoritmasının çıktısını incelediğimizde RC4 algoritmasının ne kadar hızlı çalıştığı anlaşılıyor.

Günümüzde birçok uygulama geliştirilirken Java ortamı kullanılmakta, uygulamanın kullanılacağı alana göre yararlanılan teknolojiler farklılık göstermektedir.

Bu yüksek lisans tezinde, Java ortamı kullanılarak uçtan uca güvenli iletişim geliştirmenin ne kadar kolay olduğu görülmektedir. Bu tezde J2ME ortamı ile kriptografik kütüphaneler iç içe kullanılarak mobil aygıtla sunucu arasında güvenli bir iletişim gerçekleştirilmiştir.

Bu tez çalışmasında mobil ticarette noktadan noktaya güvenlik sorunları tartışıldı ve ona uygun güvenlik yöntemleri uygulanarak soruna çözüm getirildi. Bu çalışmada kullanılan yöntemin, benzer diğer çalışmalardan üstün yönü bu çalışmada kullandığım şifreleme algoritması çok daha hızlı ve kolay uygulanabilirliğidir.

Sekil 5.2 : Uygulamanın çalışmasını gösteren resim (Adım-1)

Sekil 5.3 : Uygulamanın çalışmasını gösteren resim (Adım-2)

Sekil 5.2'de uygulamanin mobil kısmından girilen verilerin şifrelenmeden önceki halini gösteriliyor. Veri şifrelendikten sonra uygulamanın ikinci kısmı olan web uygulamasına istemci tarafından veri gönderliyor, daha sonra sunucu tarafında kimlik doğrulama ve oturum izleme yapılıyor, ardından gönderilen verinin şifresi çözülüyor ve gereken işlemler yapılıyor. Sekil 5.3'de ise web uygulamasından gönderilen şifreli verinin şifresi çözüldüten sonraki halini gösteriliyor.

6. KAYNAKLAR

1. WASSİM ITANİ, AYMAN I. KAYSSİ, 2003, J2ME End-to-End Security for M-Commerce, *IEEE Communication Magazine*, 3.
2. BRİNGEL FİLHO, WİNDSON VİANA, REİNALDO BRAGA, ROSSANA ANDRADE, 2005, FRAMESEC: A Framework for the Application Development with End-to-End Security Provision in the Mobile Computing Environment, *Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop*, Lizbon, Portekiz, 72 – 77.
3. ANDERS CERVERA, 2002, *Analysis of J2ME™ for developing Mobile Payment Systems*, Yüksek Lisans Tezi, IT University of Copenhagen.
4. MİGUEL SORİANO AND DİEGO PONCE, 2002, A security and usability proposal for mobile electronic commerce, *IEEE Communication Magazine*, 40(8), 62–67.
5. SUJATA BANERJEE, PUNEET GUPTA, 2004, *Managing Security in Mobile and Wireless Services* [online], http://www.infosys.com/offerings/IT-services/infrastructure-services/white-papers/Documents/security-mobile-wireless.pdf [Ziyaret Tarihi: 5 Ağustos 2009].
6. VİPUL GUPTA, SUMİT GUPTA, 2001, Securing the Wireless Internet, *IEEE Communication Magazine*, 39 (12), 68-74.
7. VİPUL GUPTA, SUMİT GUPTA, SHEUELİNG CHANG, 2002, Performance Analysis of Elliptic Curve Cryptography for SSL, *Proceedings of the 1st ACM workshop on Wireless security*, Association for Computing Machinery, New York, NY, USA, 87 – 94.
8. LARRY HUGHES, ABDUL-KADHİM HAY'AWİ, 2004, Java-Based End-to-End Security for Wireless Internet, *International Association for Development of the Information Society*.
9. SHERALİ ZEADALLY, NİCOLAS SKLAVOS, MOGANAKRİSHNAN RATHAKRİSHNAN, SCOTT FOWLER, 2007, End-to-End Security Across Wired-Wireless Networks for Mobile Users, *Information Systems Security*, Association for Computing Machinery, Bristol, PA, USA, 16 (5), 264 – 277.
10. MOURAD DEBBABİ, MOHAMED SALEH, CHMSEDDİNE TALHİ, SAMİ ZHİOUA, 2006, Security Evaluation of J2ME CLDC Embedded Java Platform, *Journal of Object Technology*, 5 (2), 125 – 154.
11. ANDRE N.KLİNGSHEİM, VEBJORN MOEN, AND KJELL J. HOLE, 2007, Challenges in Securing Networked J2ME Applications, *Computer*, 40 (2), 24 – 30.
12. GEMPLUS, 2005, *A smarter approach to Wireless JavaTM* [online], http://www.gemplus.com/pss/telecom/download/java_j2me_white_paper_feb05.pdf [Ziyaret Tarihi: 15 Temmuz 2009].
13. Baris Kayayurt, Tugkan Tuglular, 2004, *End-to-end security implementation for mobile devices using TLS protocol*, Yüksek Lisans Tezi, İzmir Institute of Technology.

14. JOSANG, A., SANDERUD, G., 2003, Security in mobile communications: challenges and opportunities, *Association for Computing Machinery*, 34 (21), 43 – 48.
15. OTTO KOLSİ, TEEMUPEKKA VİRTANEN, 2004, MIDP 2.0 Security Enhancements, *IEEE Communication Magazine*, 9.
16. MİCHAEL YUAN, JU LONG, 2002, *Securing wireless J2ME* [online], University of Texas, Austin, http://www.ibm.com/developerworks/java/library/wi-secj2me.html [Ziyaret Tarihi: 17 Temmuz 2009].
17. SUN MİCROSYSTEMS, 2009, *Introduction to the Java ME Platform* [online], http://java.sun.com/javame/technology/index.jsp [Ziyaret Tarihi: 25 Temmuz 2009].
18. TURHAN ÇOBAN, 2001, Java *Programlama Dili*, Alfa yayınevi, Istanbul, ISBN 975-316-631-1.
19. MUCHOW, J.W., 2001, Configurations and Profiles, *Core J2ME Technology and MIDP*, Prentice Hall PTR , New Jersey - ABD , 1 - 26.
20. WİRELESS APPLİCATİON PROTOCOL FORUM, LTD, 2001, *Application Protocol Architecture Specification* [online], http://www.openmobilealliance.org/tech/affiliates/wap/wap-210-waparch-20010712-a.pdf [Ziyaret Tarihi: 7 Ağustos 2009].
21. ERÇİN GÜDÜCÜ, 2010, *Security & Privacy of Data* [online], http://efe.ege.edu.tr/~eguducu/bim406_MIS/MIS_systems_CH3.ppt [Ziyaret Tarihi: 25 Temmuz 2010].
22. ALFRED J. MENEZES, PAUL C. VAN OORSCHOT AND SCOTT A. VANSTONE, 1999, *Handbook Of Applied Cryptography*, CRC Press, 0-8493-8523-7.
23. Murat Eren, 2003, *Açık Anahtarlı Kriptografi* [online], http://cekirdek.pardus.org.tr/~meren/belgeler/pkc/pkc.html [Ziyaret Tarihi: 30 Temmuz 2009].
24. SCOTT FLUHRER, ADİ SHAMİR AND ITSİK MANTİN, 2001, Weaknesses in the Key Scheduling Algorithm of RC4, *Association for Computing Machinery*, 1 – 24.
25. VOCAL TECHNOLOGIES, LTD, 2003, *RC4 Encryption Algorithm* [online], Buffalo, New York, http://www.vocal.com/data_sheets/RC4.pdf [Ziyaret Tarihi: 10 Ağustos 2009].
26. RALPH EDDİE, RİSE SUK-HYUN CHO DEVİN KAYLOR, 2002, *Pres_RC4 Encryption* [online], University of Washington, Seattle, http://www.math.washington.edu/~nichifor/310_2008_Spring/Pres_RC4%20Encryption.pdf [Ziyaret Tarihi: 13 Ağustos 2009].
27. SEEMA NAMBIAR, LU, C.-T., LIANG, L.R, 2004, Analysis of payment transaction security in mobile commerce, *IEEE Communication Magazine*, 475 – 480.
28. JEFFREY M. CAPONE, 2001, *Java 2ME Bridging Wireless Gap?* [online], http://onjava.com/pub/a/onjava/2001/08/14/j2me_wireless.html [Ziyaret Tarihi: 17 Ağustos 2009].
29. MİCHAEL JUNTAO YUAN, 2003, *Enterprise J2ME: Developing Mobile Java Applications*, Prentice Hall, 0-13-140530-6.

30. BOUNCYCASTLE, 2008, *Bouncy Castle Documentation* [online], http://www.bouncycastle.org/documentation.html [Ziyaret Tarihi: 25 Ağustos 2008].
31. SUN MİCROSYSTEMS, 2007, *J2ME Wireless Toolkit 2.5.2 User's Guide* [online], http://java.sun.com/products/sjwtoolkit/download.html [Ziyaret Tarihi: 20 Ağustos 2008].

ÖZGEÇMİŞ

Ben, Ilgar Mammadov 25.03.1986 yılında Azerbaycanda doğdum.Lisansımı 2003 – 2007 yılları arasında İzmirde Ege Üniversitesinde Bilgisayar Mühendisliğinde yaptım. 2007 yılında İstanbulda İstanbul Üniversitesi Bilgisayar Mühendisliğinde yüksek lisans yapmaya hak kazandım. Gelecek hedefim, aldığım eğtim doğrultusunda iş hayatımda başarılar elde etmektir.