1. GİRİŞ

1.1. Giriş

Elektronik posta ya da e-mail, bilgisayar ağlarında kullanıcıların birbirleriyle haberleşmesini sağlayan ve kullanımı en yaygın olan yöntemlerden biridir. Günümüzde internet kullanımının artması ile birlikte e-posta teknolojisi de gün geçtikçe popülerleşmekte, e-posta yolu ile sadece yazılı metin değil, aynı zamanda resim, video, sunum gibi verilerin de gönderiliyor olması e-posta teknolojisini diğer yöntemlere oranla daha tercih edilir kılmaktadır. Diğer yandan internetten gönderilen e-postalar dışarıya açık ağlardan geçmekte, gizli ve başkaları tarafından ulaşılması istenmeyen bilgilerin de bu yolla iletilmek istenmesi ciddi güvenlik sorunlarını da beraberinde getirmektedir. Bu güvenlik sorunlarını da ortadan kaldırmak için kullanılan temel yöntem kriptografidir ve e-posta için günümüzde kullanılan en yaygın teknoloji de PGP (Pretty Good Privacy) teknolojisidir.

PGP kriptografik güvenlik ve kimlik doğrulamayı sağlayan elektronik posta güvenlik programıdır. PGP güvenlik sisteminin kaynak kodu açıktır ve kaynak kodunun açık olması kodun istenilen kişi tarafından incelenebileceği anlamına gelir ki bu da kodun gerçekten güvenilir olduğunu kanıtlar niteliktedir. Program internetten ücretsiz olarak edinilebilmesinin yanında çok çeşitli donanımlarda çalışır ve kullanımı oldukça kolaydır [1]. Açık anahtar şifrelemesini kullanarak kullanıcıya gizlilik ve kimlik doğrulaması sağlaması, PGP'nin tercih edilebilirliğini arttırmış ve 1991'de ilk olarak piyasaya çıktığından bu yana en popüler elektronik mail güvenlik programlarından biri haline gelmiştir. PGP'nin ilk sürümü Philip Zimmerman tarafından Haziran 1991'de halka tanıtılmış ve o günden günümüze gelene kadar değişik sürümleri çıkarılmış ve geliştirilmiştir.

PGP e-posta için birçok şekilde güvenliği sağlamayı hedefler [1]:

Güvenilirlik: E-postanın içeriği yetkilendirilmemiş kullanıcılara karşı korunur. Yetkisi olan kullanıcı hariç kimse maili okuyamaz. Mail alıcının mail kutusuna gelene kadar şifreli halde bulunur. Mail kutusuna geldikten sonra şifreli olmasına gerek yoktur ve alıcı korumasız maile istediğini yapabilir.

Merkezi Kimlik Doğrulama: Sadece mesajı almaya yetkisi olan alıcının, mesajı gönderenin kimliğini doğru şekilde tespit etmeye yetkisi vardır. Hattı gizlice dinleyen bir kişi bu bilgiye erişemez. Hattı izinsiz bir şekilde dinleyen kişi sadece gönderenin adresini edinebilir. Bunun doğruluğunu kanıtlayamaz. Maili gönderen kişi gönderen adresini gerçek ismini saklayarak gönderirse hattı izinsiz dinleyen kişi hiçbir bilgiye ulaşamaz.

Mesaj Bütünlüğü: Bu özellik, yetkisi olan kullanıcıya mesajın iletim sırasında değişmediğinin garantisini sunar.

Kimlik doğrulama ve mesaj bütünlüğü aynı şeyin iki ayrı kısmıdır. Bu iki özellik bir mesajı alıcıya kimin gönderdiğini söyler. Mesajın kimden geldiği bilinmeden mesajın içeriğinin değiştirilip değiştirilmediği bilinemez. Aynı şekilde mesajın iletim sırasında değiştirilmediğinin garantisi olmazsa mesajın kimden geldiğinin bilinmesi ihtimali yoktur.

Kabullenme: Bu özellik gönderilmiş olan bir ileti (forward) mesajının gerçekte kimin tarafından gönderildiğinin tespit edilmesini sağlar.

2. PGP

2.1. Temel Yapısı

PGP e-posta güvenlik programı kriptografik algoritma olarak açık anahtar (public-key) şifrelemesini ve anahtar yönetimi için sayısal imzayı, veri şifrelemesi için özel anahtar kriptografisini ve sayısal imza için tek yönlü hash fonksiyonunu kullanır [1].

PGP'de güvenliği sağlamak için tercih edilen algoritmalar araştırıldığında veri şifrelemesi için IDEA algoritması, anahtar yönetme algoritması olarak RSA, mesaj bütünlüğü kontrolünün sağlanması ve sayısal imza algoritması için de MD5 ve RSA kullanılır. PGP e-posta güvenlik programının FPGA ile gerçekleştirilmesi projesinde IDEA algoritmasının VHDL ile gerçeklenmiş hali bulunmadığından başka bir veri şifreleme algoritması olan AES algoritması projenin gerçekleştirilmesi aşamasında kullanılmıştır.

PGP programının birincil amacı imzalı ve şifreli bir şekilde güvenli mesajın gönderilmesidir. Fakat mesaj gönderilirken mutlaka şifrelenmek zorunda değildir. PGP ile sadece imzalanmış mesajların da gönderilmesine izin verilir. Şifrelenmiş mesaj sadece belirlenmiş alıcı tarafından okunabilir. Açık anahtar kriptografisi kullanılarak, gönderici alıcı ile özel anahtarını değiş tokuş etmek zorunda kalmaz. Mesajın gönderici tarafından gönderilebilmesi için alıcının açık anahtarına sahip olması yeterlidir.

2.2. PGP Mesajının Gönderilmesi

Gönderilen PGP mesajı dört adımdan oluşur. İlk adım imzalama adımıdır [1]. Proje kapsamında bu adımın gerçekleştirilmesi için gereken bloklar mevcuttur. Sıkıştırma kısmı mesajın boyutunun azalması için gerçekleştirilen kısmıdır ve algoritması sıkıştırma programı ZIP 2.0 ile aynıdır. Projede gerçekleştirilmesi düşünülmediğinden bu kısımdan bahsedilmeyecektir. Sıkıştırma sonrası gerçekleştirilen adım şifreleme kısmıdır. . PGP sisteminin FPGA ile gerçeklenmesi

projesinde üzerinde çalışılan ve gerçeklenen kısım şifreleme kısmıdır. Şifreleme kısmında kullanılan algoritmalar imzalama kısmındaki algoritmalarla aynıdır. Sadece imzalama algoritmalarına ek olarak AES algoritması ile imzalanan metin şifrelenir. Mesaj gönderiminin son aşaması iletimin şifrelenmesi kısmıdır. Bu kısım bitirme konusunun kapsam ve amacının dışında olduğu için bu konuya değinilmemiştir.

2.2.1. PGP Mesajlarının İmzalanması

PGP mesajları maili gönderen kişi tarafından sayısal olarak imzalanabilir. PGP imzası, alıcının mesajı gönderen kişinin kimliğini belirlemesinde ve mesaj içeriğinin kurcalanmadığının doğruluğunun kanıtlamasında kullanılır.

PGP'nin sayısal imzası MD5 hash fonksiyonunu ve RSA açık anahtar şifreleme algoritmasını içerir. Sayısal imza yaratmak için PGP öncelikle MD5 algoritmasını kullanarak mesajın hashlenmiş halini oluşturur. Daha sonra hashlenmiş mesaj gönderenin özel anahtarı ile şifrelenir ve şifrelenmiş hash mesajı mesaja eklenir. Bu aşamada aşağıda anlatılan PGP mesajlarının şifrelenmesi adımından farklı olarak imzalanmış mesaj, şifrelenmeden RSA bloğunun çıkışında elde edilmektedir.

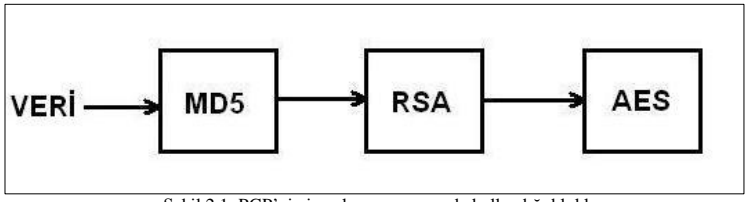
2.2.2. PGP Mesajlarının Şifrelenmesi

PGP iletilecek mesajların veya yüklenecek veri dosyalarının şifrelenmesiyle güvenilirlik sağlar. PGP mesajları şifrelemek için IDEA algoritmasını kullanır ve bu şifreleme süreci çok aşamalı bir süreçtir [1]. Öyle ki, öncelikle mesaj 8 byte uzunluğun katı olarak şekilde doldurulur. Daha sonra PGP mesaj şifre anahtarı ve bir IV (Initialization Vector) oluşturur. Bu anahtarlar her mesaj için rastgele seçilir ve eşsizdir. Her yeni mesaj şifrelendiğinde 128 bitlik rastlantısal IDEA anahtarı ve rastlantısal 64 bitlik IV yaratılır.

Bu işlemlerin devamında PGP rastlantısal mesaj şifreleme anahtarını, RSA algoritmasını ve alıcının açık anahtarını kullanarak şifreler. Birden fazla alıcının olması durumunda PGP bu anahtarın çok sayıda kopyasını her alıcının açık anahtarını kullanarak şifreler.

Buraya kadar yapılan açıklamalardan anlaşıldığı üzere PGP'nin bir mesajı hem şifreleyebildiği hem de imzalayabildiği söylenebilir. Buna göre öncelikle mesaj için imza oluşturulur ve mesaja eklenir. Devamında mesaj ve imza rastlantısal mesaj şifreleme anahtarı ile şifrelenir. En son aşama olarak rastlantısal şifreleme anahtarı RSA ile şifrelenir ve şifrelenmiş bloğa eklenir. Mesajın imzasının doğrulanması

işlemi şifre çözme işlemi yapılmadan gerçekleştirilememektedir. Bu işlemlerin gerçekleştirilmesi sırası Şekil (2.1)'deki blok şemasında gösterilmiştir:



Şekil 2.1: PGP'nin imzalama aşamasında kullandığı bloklar

Bitirme raporunun bundan sonraki bölümünde PGP devresinin tasarlanması için kullanılacak olan MD5, RSA ve AES blokları incelenmiştir. Bunlardan gerçeklenmiş olan MD5 bloğu ayrıntılı bir şekilde açıklanmış ve kullanıma hazır hale getirilmiştir.

3. KRİPTOGRAFİ VE PGP'DE KULLANILAN KRİPTOGRAFİK YÖNTEMLER

3.1. Kriptografik Sistemler

Günümüzde verilerin elektronik haberleşme kanalları tarafından sıklıkla bir yerden başka bir yere iletilmesi ve bu iletim sırasında haberleşme kanallarının herkesin erişimine açık olması ciddi güvenlik açıklarını beraberinde getirmektedir. Haberleşme kanalları kullanılarak iletilen verilerin istenmeyen kişiler tarafından ulaşılmasının engellenmesi için kriptografiden yararlanılmıştır [2]. İki çeşit kriptografik sistem vardır: Simetrik anahtarlama kriptografisi (symmetric key encryption) ve asimetrik (açık) anahtarlama kriptografisi (public key encryption).

3.1.1. Simetrik Anahtarlama Kriptografisi

Simetrik anahtarlama kriptografisinde gönderici bir mesajı elinde bulundurduğu bir anahtar ile şifreler ve şifreli metin herkesin kullanımına açık olan bir hattan iletilir. Alıcı aynı anahtar ile şifreli metini çözer ve metni okuyabilir. Her iki kullanıcının da metni şifrelemek ve okumak için kullandıkları anahtarın aynı olduğu bu sistem simetrik anahtarlama teknolojisidir [2].

Simetrik anahtarlama teknolojisinde anahtarın kolaylıkla istenmeyen kişilerin eline geçebilecek olması tehlikesi anahtarın sürekli olarak kullanılmasını engeller ve her işlem için yeni anahtar oluşturulması zorunluluğunu doğurur. Bunun yanında göndericinin ve alıcının metni şifrelerken aynı anahtarı kullanıyor olması metni kimin şifrelediğinin anlaşılmasını zorlaştırır ve bu durum da sayısal imzalamanın simetrik anahtarlama teknolojisi ile kullanılamaz hale gelmesine sebep olur. Bu sorunun üstesinden gelebilmek için Diffie ve Hellman açık anahtar kriptografisini geliştirmişlerdir.

3.1.2. Asimetrik Anahtarlama Kriptografisi

Açık anahtarlama kriptografisinde göndericide kullanılan şifreleme anahtarı ile alıcıdaki şifreleme anahtarı birbirinden farklıdır [2]. Her iki kullanıcıda da iki anahtar vardır ve anahtarlardan biri kullanılarak diğeri elde edilmediği sürece anahtarlardan birinin açıklanmasında bir sakınca yoktur.

Bilgiyi şifrelemekte kullanılan anahtara açık anahtar (public key) denir ve herkese yayınlanabilir. Şifrelenmiş bilgiyi açmak için kullanılan anahtar da özel anahtar (private key) olarak adlandırılır ve gizli tutulur. Birisine bir mesaj gönderildiğinde alıcının açık anahtarı ile mesaj şifrelenir. Alıcı bu mesajı kendi özel anahtarı ile açıp okuyabilir. Mesajı cevaplamak için ise karşı taraftakinin açık anahtarını kullanması gerekir.

Özel anahtar yalnızca mesajı gönderen kişide olduğundan, özel anahtar ile şifrelenmiş bir mesaj, o kişinin alıcıdaki genel anahtarı ile uyumluysa gönderenin kimliği onaylanmış olur. Bu, bir kişinin, başka birisinin kimliğini kullanarak işlem yapmasını engellediği gibi gönderenin yaptığı işlemi inkar etme olasılığını da büyük ölçüde ortadan kaldırır. Alıcının açık anahtarı ve gönderenin özel anahtarının kombinasyonu ile yapılan bir kriptolama mesajın hem doğruluğunu hem de gizliliğini sağlayacaktır. Bunun yanında asimetrik anahtarlama kriptografisinde anahtarın yaratılması rastgele bir sayının seçimi ile gerçekleştirildiğinden bu sayı bulunması ne kadar güç bir sayı olursa algoritma bir o kadar güçlü olur ve istenmeyen dinleyicilerin anahtarı bulması bir o kadar zorlaşır.

Günümüzde en çok kullanılan açık anahtarlama sistemi RSA'dır. Bitirme projem esnasında RSA algoritmasını kullanmamın nedeni PGP'nin açık anahtarlama teknolojisini tercih etmesinin yanında, RSA algoritmasının asal çarpanlarının bulunmasındaki zorluk sebebiyle özel anahtarının açık anahtardan bulunamaması ve bu nedenle oldukça güvenli olmasıdır.

3.2. RSA Algoritması

1977 yılında Diffie ve Hellman'ın yazmış olduğu açık anahtar kriptografisine uygun bir yöntem olan RSA, Rivest, Shamir ve Adleman tarafından gerçekleştirildi [2].

RSA kriptografisi şifreleme, şifre çözme algoritmalarına ve sayısal imzaya hizmet eden bir açık anahtarlama kriptografisidir. Modüler şifreleme RSA'da şifreleme ve

şifreyi çözme işlemi olarak kullanır. Bu işlem tek yönlü bir fonksiyondur, bir yönden hesaplanması kolay fakat ters yönden hesaplanması imkansızdır. Bu da izinsiz hattı dinleyen kişilerin özel anahtarı bulmasını engeller. RSA algoritmasının güçlü bir matematiksel algoritması olduğu için kullanımı oldukça yaygındır. RSA şifreleme ve şifreyi çözme için hep aynı algoritmayı kullanır. RSA'nın kullandığı matematiksel algoritma aşağıdaki gibidir:

Hem p'nin hem de q'nun asal olduğu p ve q seçilir. İkisinin çarpımı k bitlik katsayı N'yi meydana getirir.

$$N = p*q \text{ , } p \neq q \text{ , } 2^{k-1} < N < 2^k - 1 \quad (3.1)$$

Bir E seçilir. Bu seçime göre E ve ϕ (N)'nin en büyük ortak böleninin 1 olması ve E'nin N'den küçük olması gerekir.

$$\gcd(E, \phi(N)) = 1, E \in \{1, \ldots, N-1\} \quad (3.2)$$

N'nin Euler totient fonksiyonu uygulanır.

$$\phi(N) = (p-1)*(q-1) \quad (3.3)$$

Özel anahtar D hesaplanır.

$$D = E^{-1} \bmod(\phi(N)) \quad (3.4)$$

Mod N ve E ilan edilirken D, p ve q gizli tutulur. M mesaj ve C de şifrelenmiş mesaj olarak alınırsa RSA'da şifreleme şu şekilde gösterilir:

$$C = M^E \bmod N \quad C, M, E \in \{0, 1, \ldots, N-1\} \quad (3.5)$$

RSA'da şifre çözme işlemi de şifreleme işleminde kullanılan aynı algoritma ile gösterilir.

$$M = C^D \bmod N \quad C, M, E \varepsilon \{0, 1, \ldots, N-1\} \quad (3.6)$$

Bu iki denklem kombine edilirse eğer;

$$C^D \bmod N = M^{ED} \bmod N \quad (3.7)$$

Elimizde aşağıdaki denklem de vardır.

$E*D = 1 \bmod (\phi(N))$ (3.8)

Tüm bu denklemlerden yola çıkılarak elimizde kalan denklem sonucu doğru ise işlem doğru, yanlış ise yanlıştır.

$C^D \bmod N = M \quad \gcd(M,N)=1$ (3.9)

RSA algoritmasında N'in asal çarpanları olan p ve q'nun bulunması ciddi bir güvenlik sorununu oluşturur. Ancak p ve q çok büyük asal sayılar olduğundan bulunmaları neredeyse imkansızdır. Zaten sadece genel anahtar belli iken D yi, p ve q olmadan bulmak çok zordur. Bu nedenle RSA algoritması oldukça güvenlidir.

Devrenin gerçekleştirilmesi sırasında bu blok önceden hazır olarak bulunduğu için hazır olan blok [2] numaralı tezden kullanılmıştır.

3.3. AES Algoritması

AES (Advanced Encryption Standard) algoritması simetrik anahtarlama teknolojisi olup verinin şifrelenmesi ve şifrelenmiş bilginin çözülmesi için kullanır. Rijndael tarafından geliştirilen AES algoritmasında blok ve anahtar uzunlukları 128 bitten 256 bite kadar 32'lik araklıklarla birbirinden bağımsız olarak değişebilmekte olup, 128 bitlik blok boyunu ve 128, 192 veya 256 bitlik anahtar uzaylarını destekleyen bir simetrik blok şifreleyici olarak kullanlmaktadır [3,4].

AES algoritması içerisinde tekrarlanarak oluşturulan yapıya tur adı verilir. Algoritmada tekrarlanan tur sayısı anahtar uzunluğuna bağlıdır. Her tur 4 katmandan oluşur ve ilk olarak 128 bitlik veri 4x4 bayt matrisine dönüştürülür. Bu işlemden sonra gerçekleştirilecek adımlar baytların yer değiştirmesi, satırların ötelenmesi, sütunların karıştırılması ve tur için belirlenen anahtar ile XOR'lama işleminin yapılmasıdır.

Durum Atanması: 16 bayt uzunluğunda olan veri giriş bloğu için işlemler 4x4 matris haline getirilir.

Bayt Yer Değiştirme: Bu işlem 16 baytlık tüm veri bloğunun baytları için ayrı ayrı yeni bayt değerleri yaratılması ile gerçekleştirilir. Algoritma içerisinde doğrusal olmayan tek işlemdir .

Bayt değiştirme dönüşümü, girişindeki durumun her bir bayt'ını, 'S-Kutusu ' adı verilen bir değiştirme tablosu kullanarak, başka bir bayta dönüştürür.

Satır Öteleme İşlemi: Satır öteleme işlemi son üç satır üzerinde işlem yapar. Buna göre ikinci satır döngüsel olarak sağdan sola bir pozisyon, üçüncü satır iki pozisyon, dördüncü satır da üç pozisyon öteleme işlemini gerçekleştirir.
Sütun Karıştırma İşlemi: Sütun karıştırma işlemi 4x4'lük matrisin sütunları üzerinde gerçekleştirilir her sütun dördüncü dereceden polinom gibi kullanılır. Giriş durum matrisinin her sütunu, sütunları karıştırma dönüşümünden geçirilerek, çıkış durum matrisi elde edilir.

Tur Anahtarı ile Toplama İşlemi: Bu dönüşümde, her tur sonunda elde edilen matris, tur anahtarına karşı düşen baytlarıyla karşılıklı XOR işlemine sokulur.

Anahtar Üreteci Oluşturma İşlemi: Şifreleme ve şifreyi çözmede kullanılacak anahtar dizisi bu aşamada üretilir. Anahtar üreteci, her turun son adımında, üretilen ara değerle toplanacak olan tur anahtarlarını üretmektedir. Anahtar üretmenin temel prensibi bir satır ile dört önceki satırın XOR işlemini gerçekleştirmesidir.

PGP bloğunun gerçekleştirilmesi için şifreleme algoritması olarak AES'ten yararlanılmıştır. Bu bloğun hazır hali [4] numaralı tezden alınmış ve proje kapsamında bu hazır blok kullanılmıştır.

3.4. MD5 Algoritması

Kriptografik hash* fonksiyonları günümüz kriptografisinde oldukça önemli bir role sahiptir. Hash fonksiyonu bir mesajı alır, o mesajı fonksiyonuna sokar ve mesaj hashlenmiş bir şekilde çıkar. MD5 kriptografide oldukça yaygın şekilde kullanılan, 128 bitlik hash değerinden oluşan kriptografik hash fonksiyonudur. Güvenlik uygulamalarında oldukça sık şekilde kullanılan MD5 fonksiyonu özellikle dosyaların güvenilirliği ve doğruluğunu kontrol etmek için kullanılır [5].

MD5 algoritması, daha önceden yaratılmış olan MD4 algoritmasının yeterli bulunmaması sebebiyle, yerine kullanılmak üzere geliştirilmiştir.

Algoritma keyfi uzunluktaki bir mesajı alıp bu mesajdan 128 bitlik bir mesaj özeti çıkarır. Ayrıca farklı iki mesajdan aynı mesaj özetinin oluşması ya da verilmiş bir mesaj özetinden mesajın kendisinin hesaplanması matematiksel olarak imkansızdır.

* karmakarışık anlamına gelir, Türkçe olarak yerine geçecek birebir karşılık bulunamadığından bu kelime tercih edilmiştir.

MD5 algoritması, dijital imza uygulamalarında, RSA gibi bir açık anahtar kriptografisinde özel anahtarla şifrelenmeden önce dosyanın sıkıştırılması gerektiği durumlarda kullanılır.

MD5 algoritmasının içeriği şu şekildedir:

Girişe sıfırdan büyük olan b bit uzunluklu x dizisi uygulanır [5].

Çıkış x'in 128 bit uzunluklu bir hash fonksiyonu olur.

Rakamsal Gösterimler: MD5'in gerçekleştirilmesi sırasında kullanılan fonksiyonlar aşağıda tanımlanmıştır:

$$f(u,v,w) = (u \wedge v) \vee (u' \wedge w) \quad (3.10)$$

$$g(u,v,w) = (u \wedge w) \vee (v \wedge w') \quad (3.11)$$

$$h(u,v,w) = u \oplus v \oplus w \quad (3.12)$$

$$k(u,v,w) = v \oplus (u \vee w') \quad (3.13)$$

Sabitlerin Tanımlanması: Algoritmanın gerçekleştirilmesinde kullanılan değerlerden olan 32 bitlik 4 adet sabit değer tanımlanır:

$$h_1 = 0x67452301 \quad (3.14)$$

$$h_2 = 0xefcdab89 \quad (3.15)$$

$$h_3 = 0x98badcfe \quad (3.16)$$

$$h_4 = 0x10325476 \quad (3.17)$$

32 bitlik bir y [j] sabiti $0 \leq j \leq 63$ olması durumunda |sin (j+1)|'in ilk 32 bitinin ikilik düzendeki değeri şeklinde hesaplanır.

Kaynak kelimelere ulaşmak için 16'şar bloklar halinde kullanılacak olan sabitler tanımlanır.

$$z[0..15] = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15] \quad (3.18)$$

$$z[16..31] = [1,6,11,0,5,10,15,4,9,14,3,8,13,2,7,12] \quad . (3.19)$$

$$z[32..47] = [5,8,11,14,1,4,7,10,13,0,3,6,9,12,15,2] \quad (3.20)$$

$$z[48..63] = [0,7,14,5,12,3,10,1,8,15,6,13,4,11,2,9] \quad (3.21)$$

Sola kaç bit kaydırılacağının belirlenmesinde kullanılan sayılar tanımlanır.

$$s[0..15] = [7,12,17,22,7,12,17,22,7,12,17,22,7,12,17,22] \quad (3.22)$$

$$s[16..31] = [5,9,14,20,5,9,14,20,5,9,14,20,5,9,14,20] \quad (3.23)$$

$$s[32..47] = [4,11,16,23,4,11,16,23,4,11,16,23,4,11,16,23] \quad (3.24)$$

$$s[48..63] = [6,10,15,21,6,10,15,21,6,10,15,21,6,10,15,21] \quad (3.25)$$

Önsüreç aşaması: Bu aşamadaki amaç x'in bit uzunluğunu 512'nin katı olacak şekilde doldurulmasıdır. Bunun için yapılması gereken işlem şöyledir: Öncelikle en yüksek anlamlı bitlerin b bit kadarlık kısmı x giriş verisi ile doldurulur. En düşük anlamlı bitlere 2 adet 32 bitlik kelimeden oluşan b mod 2^{64}'ün 64 bitlik değeri eklenir. Geri kalan bitlerden en yüksek anlamlı bit tek bir 1 biti ile doldurulur. Boşta kalan diğer bitler toplamı 512'nin katından küçük olan r-1 ($r-1 \geq 0$) adet 0 biti ile doldurulur. Bu şekilde 521'nin katı olacak şekilde bir çıkış elde edilmiş olur. Bu işlemlerin gerçekleşmesinden sonra verinin 512'lik bit bloklarının sayısını temsil etmek için m değeri belirlenir. Bu işlemler sonucunda giriş 16m adet 32 bitlik kelimelerden oluşur:

$$x_0 x_1 x_2 \ldots x_{(16m-1)} \quad (3.26)$$

En son başlangıçtaki sabitler yeni bir yere yüklenir:.

$$(H_1, H_2, H_3, H_4) \leftarrow (h_1, h_2, h_3, h_4) \quad (3.27)$$

Süreç aşaması: 0'dan m-1 değerine kadar her i değeri için 16 adet 32 bit kelimelerin i. bloğu geçici bir hafızaya şu şekilde kopyalanır:

$$X[j] \leftarrow x_{(16i+j)} \quad (3.28)$$

Burada j değeri $0 \leq j \leq 15$ şeklinde tanımlanır. Bu işlemin devamında 4 adet 16 adımlık dizi, zincirin güncellenmesinden önce gerçeklenir.

Birinci dizi $0 \leq j \leq 15$ olması durumunda gerçeklenir:

$$t \leftarrow (A + f (B,C,D) + X[z[j]] + y[j]) \quad (3.29a)$$

$$(A,B,C,D) \leftarrow (D, B + (t \lfloor\!\lfloor s[j]), B, C) \quad (3.29b)$$

İkinci dizi $16 \leq j \leq 31$ olması durumunda gerçeklenir.

$$t \leftarrow (A + g (B,C,D) + X[z[j]] + y[j]) \quad (3.30a)$$

$$(A,B,C,D) \leftarrow (D, B + (t \lfloor\!\lfloor s[j]), B, C) \quad (3.30b)$$

Üçüncü dizi $32 \leq j \leq 47$ olması durumunda gerçeklenir.

$$t \leftarrow (A + h (B,C,D) + X[z[j]] + y[j]) \quad (3.31a)$$

$$(A,B,C,D) \leftarrow (D, B + (t \lfloor\!\lfloor s[j]), B, C) \quad (3.31b)$$

Dördüncü dizi $48 \leq j \leq 63$ olması durumunda gerçeklenir.

$$t \leftarrow (A + k (B,C,D) + X[z[j]] + y[j]) \quad (3.32a)$$

$$(A,B,C,D) \leftarrow (D, B + (t \lfloor\!\lfloor s[j]), B, C) \quad (3.32b)$$

Bu aşamalar bittikten sonra zincir güncellenir.

$$(H_1, H_2, H_3, H_4) \leftarrow (H_1 + A, H_2 + B, H_3 + C, H_4 + D) \quad (3.33)$$

Bitiş: Son hash değeri ortaya çıkar. Bu değer en son güncellenmiş $H_1||H_2||H_3||H_4$ değerlerinin bağlanmış halidir.

Bir sonraki aşamada MD5 algoritmasının ve onun devamında PGP'nin ne şekilde gerçeklendiği anlatılmakta ve gerçeklenme sırasında karşılaşılan problemlere değinilmektedir.

4. PGP'NİN GERÇEKLENMESİ

4.1. MD5 Algoritmasının Oluşturulması

MD5 algoritmasının anlaşılması ve gerçekleştirilmesi PGP ile güvenli e-posta gönderim sisteminin FPGA üzerinde gerçeklenmesi projesinin en önemli aşamasını oluşturmaktadır. Bunun başlıca nedenlerinden biri MD5 algoritmasının günümüzde oldukça yaygın bir şekilde kullanılan tek yönlü güçlü bir hash algoritması olmasının yanında, bu algoritmanın şimdiye kadar donanımsal ortamda sadece 1 kez gerçeklenmiş olması, MD5 algoritmasının bu proje kapsamında gerçekleştirilmiş olmasını, gelecekte bu konuda geliştirilebilecek diğer projelere örnek olması açısından, daha önemli kılmaktadır [6].

PGP ile güvenli e-posta gönderim sisteminin FPGA üzerinde gerçeklenmesi projesinin ilk aşaması olarak öncelikle PGP hakkında bilgi toplandı. PGP'nin gerçeklenmesi için kullanılacak bloklar MD5, RSA ve AES olarak belirlendi. MD5 algoritmasının elimizde VHDL (Very high speed integrated circuit Hardware Description Language) ile gerçeklenmiş hali bulunmadığından öncelikle MD5 ile ilgili bilgi toplandı. MD5 algoritması yeterince anlaşıldıktan sonra gerçeklemek için VHDL dilinin öğrenilmesi aşamasına geçildi. Bilinen en güncel betimleme dillerinden biri olan VHDL'in öğrenilmesinde [7] numaralı kaynaktan yararlanılmıştır. VHDL dilinin kullanılacağı Xilinx ISE adlı program bilgisayara kurularak programın özellikleri incelenmiştir. Ayrıca simülasyonların yapılacağı ModelSim adlı program hakkında da bilgi edinilmiş ve bilgisayara kurularak kullanıma hazır hale getirilmiştir.

Hem Xilinx programının kullanımına hem de VHDL diline alışmak için MD5 algoritmasının gerçekleştirilmesi gereken (3.10), (3.11), (3.12), (3.13) numaralı denklemlerdeki fonksiyonlar birer bit için gerçeklendi. Bunun devamında MD5'te bu fonksiyonlar 32 bit olarak kullanıldığı için fonksiyonlar bileşen (component) olarak gösterilip 32 bitlik devre olacak şekilde gerçeklenmesi sağlanmıştır.

4.1.1. Sabitler Modülünün Oluşturulması

MD5 algoritması incelendiğinde algoritmanın belirli yerlerinde bir ya da birden daha çok aşamasında kullanılmak üzere tanımlanan sabit değerler olduğu görülür. Her yeni modül oluşturulduğunda bu değerlerin tekrar tekrar yazılmasını engellemek ve kod içindeki kalabalığı önlemek için 'sabitler' adı altında tek bir modülde hepsinin toplanmasına ve bu modülün gerekli olduğu durumlarda ilgili diğer modüllere kütüphane şeklinde eklenmesine karar verildi. Öncelikle h_1, h_2, h_3, h_4 değerleri onaltılık sayı tabanında 'sabitler' içine eklendi. Kaynak kelimelere ulaşmak için onaltışar dört blok halinde tanımlanmış olan z[j] dizisinin değerleri ve sola kaç bit kaydırılacağının belirlenmesinde kullanılan onaltışar dört blok halinde tanımlanmış olan s[j] dizisi 'sabitler' içerisinde tanımlandı. İlk 32 bitinin ikilik düzendeki değerinin kullanılacağı lsin (j+1)l'nin değeri işlem adımlarını kolaylaştırmak ve hataya yer vermemek için MATLAB'da hesaplandı ve bulunan sonuçlar 'sabitler' modülüne eklendi. Bu sayede MD5 algoritmasının ilk iki adımının gerçeklenmesi tamamlanmış oldu.

Algoritmanın üçüncü aşaması olan önsüreç aşaması sonunda çıkacak verinin bit uzunluğunun 512'nin katı olması gerektiğinden bu değer 0 ile 10000000 arasında değiştirilebilir bir değer olarak tanımlanıp 1024 değeri bu uygulama için tercih edildi. Çıkışın kaç adet 512'lik bit bloğundan oluştuğunu gösteren m sayısı iki olarak hesaplanıp bu aşamanın çıkışının 32 bitlik kelimelerden oluşan 16*m adet bloktan oluştuğunun gösterilmesi sağlanmıştır. Bu işlemler sabitler dosyasının altında tanımlanırken önsüreç aşaması ile ilgili olan diğer adımlar 'Önsüreç (preprocess)' modülünde gerçeklenmiştir.

4.1.2. Önsüreç Modülünün Oluşturulması

Gönderilmeye çalışılan mesajın bit uzunluğunun çıkış bit uzunluğundan daha kısa olduğu bilgisinden yola çıkılarak uygulamada kullanılmak üzere giriş veri uzunluğu 800 bit olarak alındı. Matematikte genel olarak bir tamsayının sürekli olarak ikiye göre modunun alınması işlemi sayının ikilik tabandaki karşılığına denk geldiği bilinen bir gerçektir. Buna göre giriş bit uzunluğu b'nin mod 2^{64} işlemi b tamsayısının ikilik tabana göre yazılmış halinin en düşük anlamlı 64 bitinin alınmış olması anlamına gelir. 'Önsüreç' modülünde de bu işlem için bir tam sayının 64 defa

2'ye göre modülünün alınarak b mod 2^{64} işleminin gerçeklenmesini sağlayan bir fonksiyon yazılmıştır.

'Önsüreç' modülünün çıkışının oluşturulması için öncelikle 1024'lük çıkışın en anlamlı b biti giriş mesajı ile doldurulur. En düşük anlamlı 64 biti b mod 2^{64} işlemine göre doldurulduktan sonra geri kalan bitlerin tanıma göre r adet olması gerekir. Bu kalan bitlerden en anlamlı olanı 1 biti ile geri kalan r-1 adet bit ise 0 biti ile doldurulur. Bu işlemden sonra bu değer her biri 32 kelimelik olan 16*m adet bloğa aktarılır. Bu şekilde MD5'in üçüncü adımı da gerçeklenmiş olur. Bu aşama daha sonraki aşamanın giriş işareti olarak kullanılacaktır.

4.1.2. Durum Makinesi Modülünün Oluşturulması

Algoritmanın süreç aşaması zincirin güncellendiği ve algoritmanın son aşaması gerçekleştirilmeden önceki ara adımların gerçekleştirildiği kısımdır. MD5 algoritmasının süreç aşaması birbiri ile bağlantılı adımlardan oluşmuş bir aşamadır. Yani bir aşamanın gerçekleşmesi için diğer aşamadaki sonucun öncelikle olması gerekir ki devamındaki aşamada o aşamadaki sonuç kullanılabilsin. Bu nedenle bu durumu en iyi şekilde VHDL ile gerçekleştirebilmek için değişik süreçlerin gerçekleştirileceği bir durum makinesinin tasarlanması uygun görüldü. Durum makinesinin tasarlanacağı modüle 'durum makinesi (state machine)' adı verildi.

VHDL'in öğrenilme aşamasında yazılmış olan 32 bitlik (3.10), (3.11), (3.12), (3.13) numaralı denklemlerdeki fonksiyonlar ara bağlantıda kullanılmak üzere f1, g1, hf1 ve k1 adları ile koda bileşen olarak eklendi.

Durum makinesi gerçekleştirilirken kullanılan saat yükselen kenar tetiklemeli olarak tasarlandı ve resetin lojik 1 olması durumunda devrenin tamamen sıfırlanmasına karar verildi. Bunun yanında donanımsal olarak proje gerçekleştirildiğinde isteğimiz dışında olabilecek küçük voltaj değişimlerinin devrenin resetlenmesine sebep olmasının önüne geçilmesi amacıyla reset senkron olarak tasarlandı.

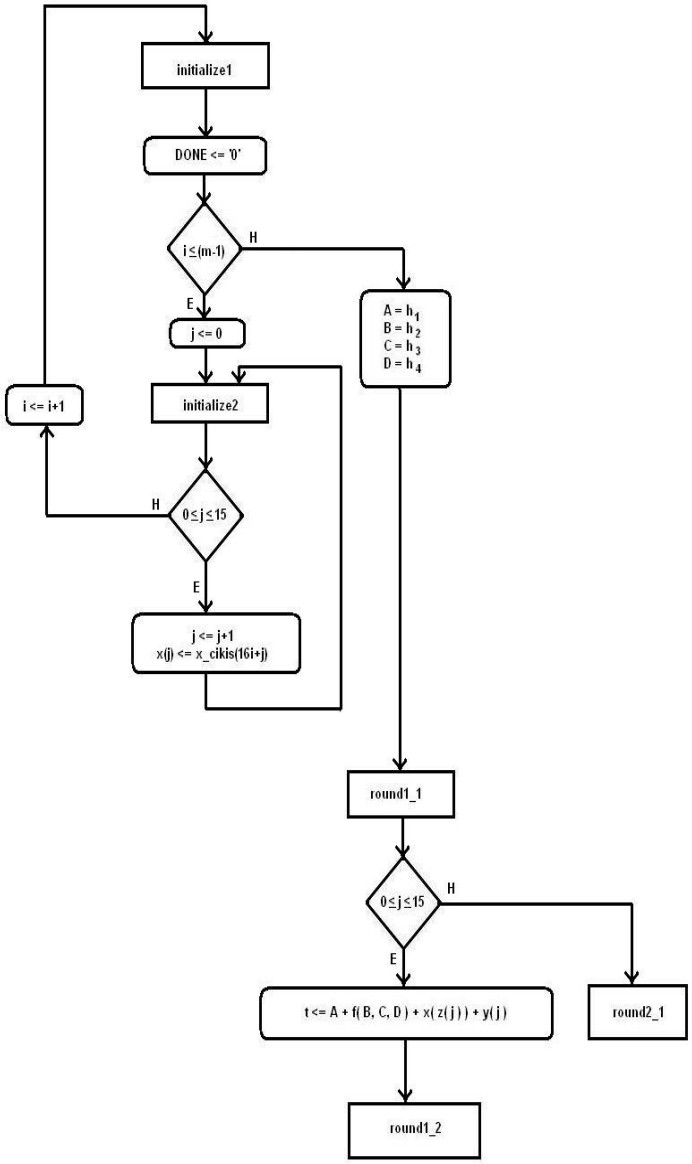
Bunların dışında devre sürekli çalışır durumda olduğundan algoritmanın tamamlandığı zaman çıkan sonucun daha sonraki aşamalarda diğer bloklarda kullanılabilmesi için işlemin bittiğini haber veren 'tamamlandı' anlamına gelen bir çıkış tanımlandı. Böylece tamamlandı işareti lojik sıfır iken diğer bloklara bilgi aktarılmayacak, tamamlandı işareti lojik bir olduğunda diğer bloklara bilgi aktarılacaktır.

Süreç aşamasında gerçekleştirilen ilk durum önsüreç aşamasından gelen veri bloğunun X[j] $\leftarrow x_{(16i+j)}$ şeklinde yeni bir düzende kullanılacak şekilde oluşturulmasıdır. Burada $0 \leq i \leq m-1$ şeklinde i değeri tanımlanırken $0 \leq j \leq 15$ j değerinin tanım aralığıdır.. Gelen verilerin sırayla ve bit bit aktarılması istendiğinden ilk önce x_0 bitinin aktarılması sağlanacak şekilde bir algoritma oluşturulur. Durum makinesinin ilk iki durumu veri aktarımının doğru ve eksiksiz bir şekilde x_0'dan başlayarak $x_{(16\ i\ +\ j)}$'de son bulması sağlanmıştır. Veri aktarım işlemi bittikten sonra sabitler modülünde tanımlanmış olan h_1, h_2, h_3, h_4 değerleri ara bağlantı olarak kullanılan A, B, C, D değerlerine aktarılıp kullanılabilir hale getirilmişlerdir.

Şekil (4.1)'de verilen blok şemasına göre bu işlemler initialize1 ve initialize2'de gerçekleştirilmektedir:

VHDL'de tasarım yapılırken genel olarak İngilizce kelimeler kullanılmıştır. Bunun nedeni [5] numaralı kaynakta algoritma tanımlanırken kullanılan kelimelerin aynısının seçilmeye çalışılmasından kaynaklanmaktadır. Kelimelerin aynısının seçimi kodun yazılım aşamasında takip edilebilirliği açısından kolaylık sağlamıştır. Ayrıca bu kodu kullanıp geliştirmek isteyenlerin de algoritma ile koddaki ortak kelimelerden yararlanmaları sağlanarak daha çabuk çözüme ulaşmalarını sağlamak hedeflenmiştir.

Ara bağlantı olarak kullanılan A, B, C, D güncellenip çıkış değerine aktarılması için öncelikle kablo ya da diğer bir deyişle ara bağlantı olarak tanımlanan t değerinin güncellenmesi ve bu güncellemenin diğer ara bağlantıları sağlayan A, B, C, D değerlerine aktarılması gerekir. Bu iki olaydan birincinin sonucu diğerinin de sonucunu etkileyeceğinden durum makinesinde farklı durumlarda tanımlanmaları gerekir. Yani farklı saat darbelerinde bu olayların gerçekleştirilmesi gerekir. Teorikte ara bağlantıyı sağlayan t değerinin güncellenmesi 1 saat darbesi kadar zamanda, bu güncellenmenin A, B, C, D değerlerine aktarılması da bir saat darbesi kadar zamanda gerçekleştirilebilir. Fakat pratikte 3 saat darbesinde bu işlemlerin hepsinin gerçekleştirilmesi uygun görülmüştür.



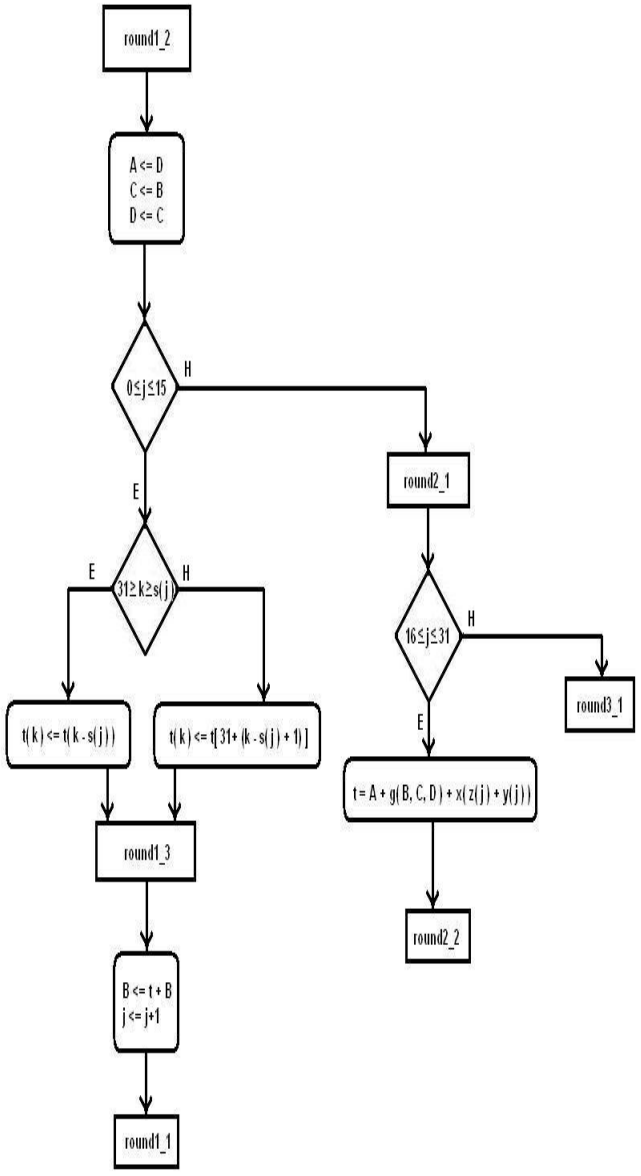
Şekil 4.1: Durum makinesi modülündeki durumlardan ilk üçü

Buna göre ilk saat darbesinde (3.29a) denklemi gerçekleştirilir. MD5 algoritmasında kullanılan + işlemi 2^{32}'ye göre modulo işlemini temsil etmektedir. VHDL'de ise + işareti kullanıldığında yaptığı iş tam olarak n bit uzunluğundaki bir verinin 2^n'ye göre modulo işlemini gerçekleştirmektir. VHDL'de + işaretinin kullanılması için de kütüphane olarak 'ieee.std_logic_unsigned' kütüphanesinin çağırılması yeterlidir. Bu işlemin gerçekleştirilmesi sırasında karşılaşılan diğer bir sorun + işlemi yapılırken bir değişkenin değerinin başka bir değişken tarafından kontrol edilmesi işleminin VHDL tarafından kabul edilmemesidir. Bu nedenle X[z[j]] işlemi içerisindeki z[j] değerinin her yeni değeri için X değeri tek tek yazılarak devre oluşturuldu. Bu işlemlerin hepsinin 1 saat darbesi kadar sürede yapılması sağlandı.

Bir sonraki saat darbesinde gerçekleştirilen devre (3.29b) denklemidir. Burada yapılması istenen A, C ve D değerlerine sırasıyla D, B ve C değerlerinin aktarılması ve B değerine t değerine s[j] değerinin temsil ettiği değer kadar öteleme yapıldıktan sonra bu sonucun önceki B değeri ile mod 2^{32} işlemine sokulup B'ye aktarılmasıdır. Fakat t değerinin s[j] değerine kaydırılması işlemi ile A, B, C, D değerlerinin güncellenmesiyle aynı saat darbesi içerisinde gerçeklenmesi mümkün değildir. Çünkü aynı anda gerçekleştirilmek istenirse kaydırılma işlemi bir saat darbesi sonunda sonuca etkileyeceği için B değerinin güncellenmesi diğerlerinden bir saat darbesi sonra olacak bir işlem olurdu ve B değerinden istenilen sonuç elde edilemezdi. Bu nedenle A, C, D güncellenirken aynı saat darbesinde kaydırma işlemi yapılmış, bir sonraki saat darbesinde de B'nin değeri güncellenmiştir.

T değerinin s[j] değeri kadar kaydırılması işlemi iki ayrı kısımda gerçekleştirildi. Öncelikle kaydırılmak istenen miktar kadar sayı sola kaydırılma işlemi yapıldı. Daha sonra kaydırılan kısımlara doldurulmak üzere en sondaki sayılar bu boşluklara yerleştirildi. Bu şekilde kaydırma işlemi bir saat darbesi süresinde tamamlanmış oldu.

Şekil 4.2'de gösterilen grafik anlatılanların daha iyi anlaşılması açısından çizilmiş durum akış diyagramıdır. Geri kalan durumların durum akış diyagramları Şekil A.1, Şekil A.2 ve Şekil A.3'te bulunmaktadır.



Şekil 4.2: Durum makinesinde dizilerin gelme durumu

Tüm bu güncelleştirme işlemleri tamamen bittikten sonra yeni gelen değerlerle işlemlerin tekrarlanması kısmına geçilir. İlk dizide bu adımlar j değerinin 0'dan 15'e kadar değiştiği her değer için ayrı ayrı yapılır. Bu dizi bittiğinde diğer dizilere geçilir ve geri kalan üç dizide de aynı işlemler tekrarlanır. Dizilerde ilk aşamada değişen tek şey kullanılan fonksiyondur. İlk dizide 32 bitlik f (u,v,w) devresi kullanılırken ikinci dizide 32 bitlik g (u,v,w), üçüncü dizide 32 bitlik h (u,v,w) ve dördüncü dizide 32 bitlik k(u,v,w) devresi kullanılır.

Devrede süreç aşaması her $0 \leq j \leq 63$ arasındaki her sayı için üç saat darbesi süresince gerçekleştirilir. Sadece diziler arası geçişlerde bir saat darbesi fazla işlem yapılır. Bunun nedeni dizinin bittiğini kontrol etmek için harcanan zaman yüzündendir. Bu da beklenen bir durumdur.

Dördüncü dizinin sonunda dizi tamamlandığında yapılan işlem zincir değerlerinin güncellenmesi kısmıdır. Var olan h_1, h_2, h_3, h_4 değerlerine sırasıyla güncellenmiş A, B, C, D değerleri toplanarak hh_1, hh_2, hh_3, hh_4 değerlerine aktarılır. Aynı zamanda algoritmanın bittiğini diğer bloklara haber verecek olan tamamlandı işareti lojik 1'e çekilir. Bu sayede algoritmanın bittiği anlaşılır. Elde edilen son değerler durum makinesinin yani MD5'in çıkışıdır.

4.1.4. MD5 Modülünün Oluşturulması

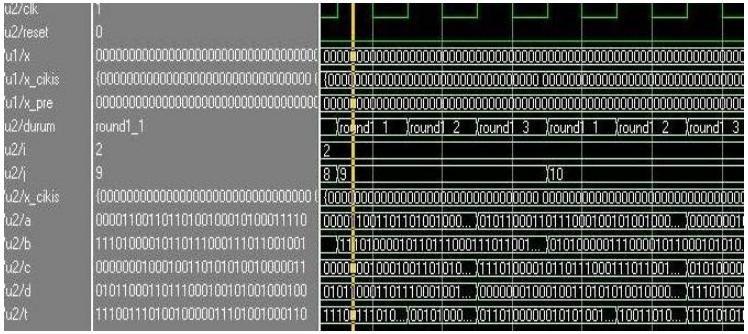
Algoritmanın tamamlanması ve test edilebilmesi için sıradaki adım MD5'i oluşturan blokların birleştirilmesidir. Bu nedenle öncelikle MD5 adı altında yeni bir modül oluşturuldu ve MD5 algoritmasının gerçekleştirilmesi sırasında kullanılan sabit değerlerin bulunduğu sabitler modülü kütüphane olarak eklendi.

Algoritmanın VHDL'de yazılması sırasında kullanılan iki modül olan Önsüreç ve durum makinesi modülleri bileşen olarak eklendi. 'Önsüreç' modülünün girişi MD5'in girişi, çıkışı durum makinesi modülünün girişi olarak atandı. Durum makinesi modülünün çıkışı da MD5'in çıkışı olarak atandı. Algoritmanın bittiğini gösteren tamamlandı işareti de devrenin çıkışına atandı.

Bu işlemlerden sonra sıra algoritmanın test aşamasına gelmiştir. Bunun için öncelikle test için yeni bir modül yazıldı. Modülde ilk önce tüm devrenin 50ns boyunca resetlenmesi sağlandı. Bu sayede kaydedilmiş olabilecek sayıların sıfırlanması

amaçlandı ve devre başlangıç durumuna getirilmiş oldu. Daha sonra devreye ASCII karakter kodu olan 'a' değeri girildi ve bu işlem sonucu oluşan çıkış takip edildi.

Test sonucu incelendiğinde her döngüye istenilen zamanda girildiği ve istenilen sürede döngüden çıkıldığı gözlendi. Girilen değer ve bu değer sonucu oluşması beklenen ara bağlantıların değerleri elle hesaplandığında test sonucu çıkan değer ile birebir örtüştüğü gözlendi. Ayrıca çıkış istenildiği gibi 32 bitlik dört ayrı parçadan oluştuğundan yani çıkışın 128 bit olmasından dolayı yapılan kontroller sonucunda MD5 algoritmasının istenildiği şekilde çalıştığı söylenebilir. MD5 algoritmasının test edilmesi sonucu belli bir zaman diliminde elde edilen değer Şekil (4.3)'teki gibidir:

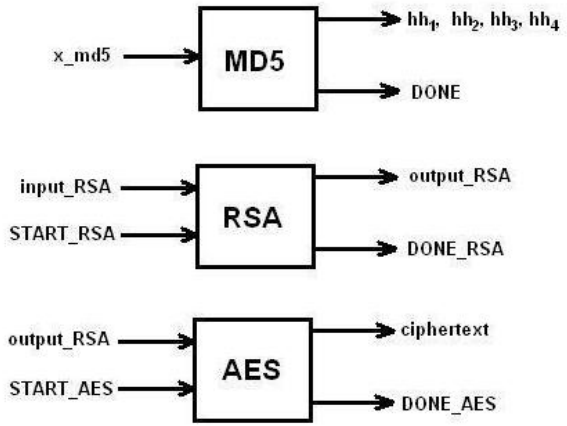


Şekil 4.3: MD5'in çıkışı

MD5 algoritmasının gerçeklenmesi aşaması bittikten sonra sıra PGP'de şifreleme aşamasında kullanılmış olan diğer iki bloğun yani RSA ve AES'in MD5 ile birleştirilmesi kısmına gelmiştir. RSA ve AES kısımları elimizde hazır şekilde bulunduğundan bu üç ayrı bloğun birleştirilmesi için PGP adı altında bir modülün yazılması yeterli olacaktır.

4.2. PGP'nin Oluşturulması

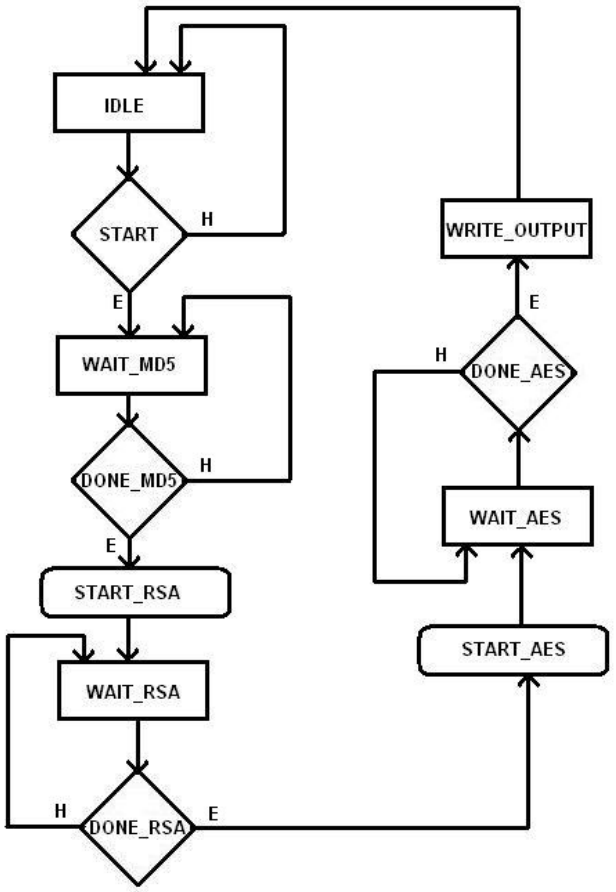
Bu modülün gerçeklenmesi için kullanılan üç algoritmanın sırayla MD5, RSA ve AES şeklinde bağlanması gerekmektedir. Çünkü PGP'nin şifrelenmesi sırasında algoritmaların çalışma sırası bu şekildedir. Modüllere hangi verilerin bağlı olup hangilerinin çıktığı Şekil (4.4)'te verilmiştir:



Şekil 4.4: PGP bloklarında kullanılan modüllerin giriş ve çıkışları

Öncelikle RSA'nın gerçekleştirilmesi sırasında kullanılan RSApackage_1024 kütüphanesi ile MD5'in gerçekleştirilmesinde kullanılan sabitler kütüphanesi eklendi.

Kullanılan MD5 algoritmasının girişi 800 bit olup çıkışı 128 bittir. Bunun yanında kullanılan RSA ve AES algoritmasının giriş ve çıkışı 1024 bittir. MD5'in çıkışı RSA'nın girişi olması gerektiğinden RSA'nın ilk 128 biti MD5'ten gelen veri ile doldurulur ve geri kalan bitlere lojik 0 değeri aktarılır. Bunun yanında kullanılan RSA ve AES algoritmalarının da MD5'te olduğu gibi işlemin bittiğini haber veren işaretlerinin olmasının yanında bir de işleme başlayacaklarını haber veren başlangıç işaretleri vardır. RSA'nın çalışması için MD5 algoritmasından bittiğini belirten işaretin gelmesi gerektiği gibi AES'in çalışmaya başlaması için de RSA'dan bitti işaretinin gelmesi gerekir. Aynı zamanda MD5 algoritmasının girişi devrenin girişi, MD5 algoritmasının çıkışı RSA algoritmasının girişi, RSA algoritmasının çıkışı AES algoritmasının girişi, AES algoritmasının çıkışı da devrenin çıkışını oluşturur. Tüm bu aşamalar farklı saat zamanlarında gerçekleştiğinden bunların bir durum makinesi içerisinde tanımlanması gerekir. Durum makinesinin zaman diyagramı şekildeki gibidir:



Şekil 4.5: PGP bloğu durum akış diyagramı

Durum makinesinde devre saat girişinde görülen bir yükselen kenar ile tetiklenir ve reset lojik 1 durumuna geldiğinde devre sıfırlanır. Devre ilk önce boşta bekleme durumundadır. Bu durumda iken başlama işaretinin gelmesini bekler. Başla işareti gelmediği sürece devre boşta bekleme durumundadır. Bu durumda PGP bloğunun bittiğini kontrol eden tamamlandı işareti lojik 0 durumuna getirilir. Başla işareti geldiğinde MD5 işleminin kontrol edildiği sürece gidilir.

MD5 algoritmasının tamamlandığının kontrol edildiği bu durumda MD5'den işlemin tamamlandığını gösteren tamamlandı işaretinin gelmesi beklenir. Bu işaret gelene kadar bu durumda kalmayı sürdürür.

MD5 algoritmasının kontrol edildiği süreçte işlemin bitmesi RSA algoritmasını tetikler. Bu işlemden sonra RSA'dan işlemin bittiğini gösteren tamamlandı işaretinin gelmesi beklenir. Gelmediği sürece RSA'nın durumunu kontrol eden süreçte beklenir.

RSA'dan işlemin bittiğini gösteren işaret geldiğinde AES'in başlaması için de gereken işaret verilir ve AES'in durumunun takip edildiği AES durumuna gidilir. Bu durumda da AES'in işleminin bittiğini gösteren işaretin gelmesi beklenir. Bu işaret gelene kadar beklenir, geldiğinde de gerçekleştirilecek adımlar bitmiş demektir. AES'in çıkışı yazdırılır ve devrenin bitiğini gösteren bitti işareti lojik 1 durumuna getirilir. Bu sayede PGP algoritması ile şifreleme işlemi tamamlanmış olur.

4.3. FPGA'de Geçeklenmesi

FPGA (Field Programmable Gate Array - Alanda Programlanabilir Kapı Dizileri), çok sayıda lojik blok dizisi içeren, giriş çıkış birimleri ve bütün bu birimleri bağlayan programlanabilir sayısal tümleşik devrelerdir. Arzu edilen devrenin oluşturulması için temel bloklar birleştirilerek karmaşık bloklar elde edilebilmekte ve bu karmaşık blokların işlevselliği arttırılabilmektedir. Düşük maliyetli olması ve tasarım sırasında kullanıcıya esneklik sağlaması sebebiyle kullanımı gittikçe yaygınlaşmıştır.

PGP ile güvenli e-posta gönderim sistemi FPGA üzerinde gerçekleştirilmiştir.

5. SONUÇLAR

PGP ile güvenli e-posta sisteminin FPGA ile gerçeklenmesi projesinde, PGP hakkında genel bir bilgi verilmiş ve MD5 algoritması üzerine yoğunlaşılmıştır. PGP güvenlik programının e-posta gönderilirken kullanılması durumunda kullanıcıya ne gibi faydalar sağlayacağı ayrıntılı bir şekilde anlatılmıştır. Bunun yanında PGP programının çalışırken ne gibi kriptografik algoritmaların ne şekilde kullandığı ve bu algoritmaların çalışmaları sırasında gerçekleştirilen aşamalar incelendi. Bu sayede bilginin ne kadar güvenli bir şekilde gönderilmiş olduğu, bir başkasının sadece bize ait olan bilgiyi biz istemedikçe göremeyeceği öğrenildi. Bunun yanında MD5 algoritması ayrıntılı bir şekilde incelendi ve donanımsal olarak gerçeklenebilmesi için gereken çalışmalar yapıldı. Yapılan çalışmalar ileride bu konuda yapılacak olan çalışmalara kaynak olacak niteliktedir. Bu konu hakkında şimdiye kadar sadece bir adet makalenin yazılmış olması nedeniyle bu bitirme çalışması daha da önem kazanmaktadır.

KAYNAKLAR

[1] **Schneier, B.,** 1995, E-mail Security, How to Keep Your Electronic Messages Private, John Wiley and Sons, United States of America.

[2] **Bayam, K.A.,** 2007. Differential Power Analysis Resistant Hardware Implementation of the RSA Cryptosystem, *M. Sc. Thesis*, İstanbul Technical University Institue of Science and Technology, İstanbul.

[3] **Kayış, M.,** 2006. AES Uygulaması'nın FPGA Gerçeklemelerine Karşı Güç Analizi Saldırısı, *Yüksek Lisans Tezi,* İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.

[4] **Ordu, L.,** 2006. AES Algoritmasının FPGA Üzerinde Gerçeklemesi ve Yan Kanal Analizi Saldırılarına Karşı Güçlendirilmesi, *Yüksek Lisans Tezi,* İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.

[5] **Menezes, A., Oorschot, P. ve Vanstone, S.,** 1997. Handbook of Applied Cryptography, CRC Press.

[6] **Jarvinen, K., Tommiska, M. Ve Skytta, J.,** 2005. Hardware Implementation Analysis of the MD5 Hash Algorithm, *The 38th Hawaii International Conference on System Sciences*, Otakaari, Finland.

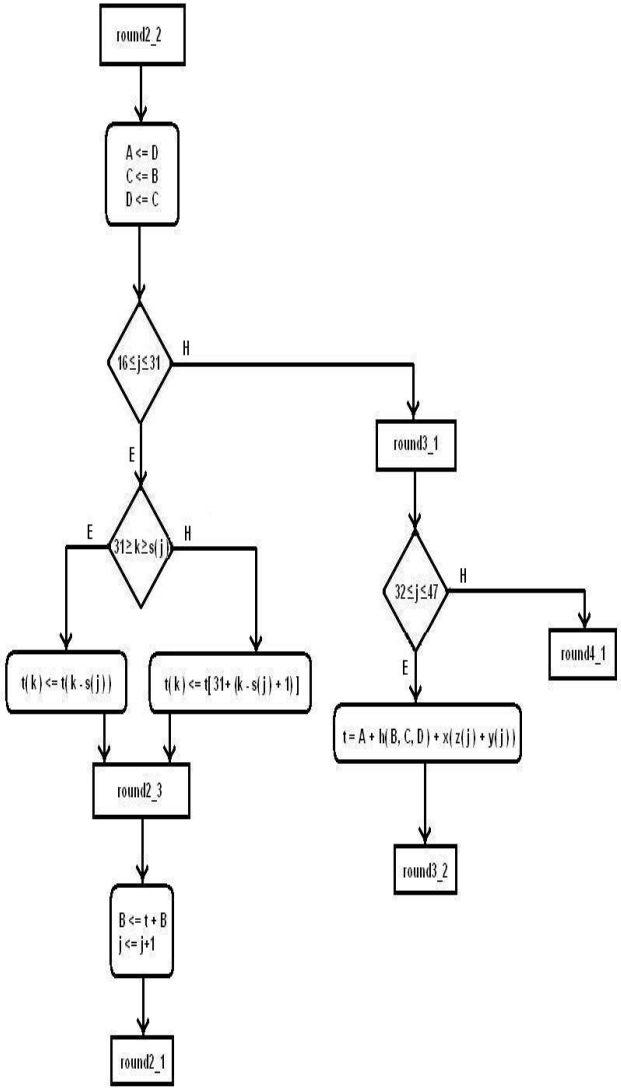
[7] **Pedroni, V.A.,** 2004. Circuit Design with VHDL, MIT Press, London, England.

[8] **Wikipedia,** 2008. http://tr.wikipedia.org/wiki/FPGA.

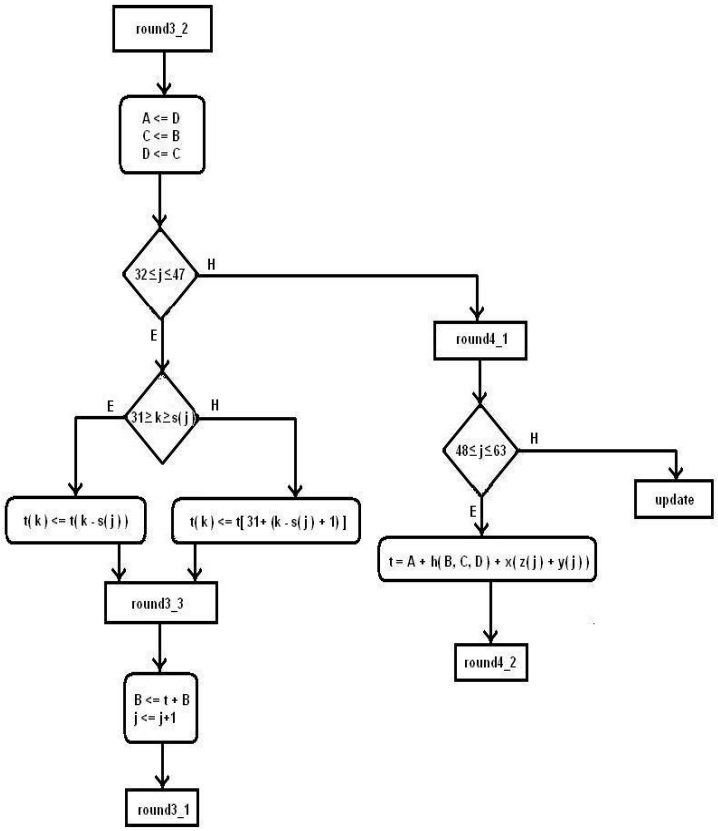
EKLER

EK-A : Durum makinesindeki durum diyagramları (Arka sayfalardadır.)

EK-B : Uygulamanın kaynak kodları ve bitirme projesi (CD'ye kaydedilmiştir).



Şekil A.1: Durum makinesi akış diyagramı 1



Şekil A.2: Durum makinesi akış diyagramı 2

ÖZGEÇMİŞ

1985 yılında doğan Vijlan ÇELİK, Tekirdağ Anadolu Lisesi'nden 2003 yılında mezun olduktan sonra lisans öğrenimini İTÜ Elektronik Mühendisliği'nde görmeye başladı. 2006 yılında, İstanbul Teknik Üniversitesi Bilgi İşlem Daire Başkanlığında 2 yıl süreyle yarı zamanlı öğrenci olarak çalışmıştır. Birkaç kez, İTÜ Onur ve Yüksek Onur listesine giren Vijlan ÇELİK, 2003 senesinde İTÜ başarı bursuna layık görülmüş ve 5 yıl süresince bu bursu almaya devam etmiştir.