GÜVENLİ ELEKTRONİK POSTA SİSTEMİ PGP'NİN FPGA ÜZERİNDE TASARIMI VE GERÇEKLENMESİ

Vijlan Çelik, Berna Örs Yalçın

***Özet*— Gün geçtikçe özel haberleşme yöntemleri ağlar üzerinden kullanılmaya başlanmıştır. Elektronik posta da haberleşme aracı olarak ağ üzerinde kullanılan en yaygın yöntemlerden biridir.**

Kullanılan bilgi iletişim kanallarının herkesin erişimine açık olması bilginin taşınması sırasında güvenliğin sağlanmasını zorunlu kılmaktadır. Bu nedenle bilginin güvenli bir şekilde ağda taşınabilmesi için kriptografiye başvurulur.

PGP (Pretty Good Privacy) ile e-postaların güvenli şekilde iletilebilmesi işlemi dört aşamadan ibarettir. Buna göre PGP mesajı öncelikle imzalanır, sonra sıkıştırılır, sıkıştırıldıktan sonra mesaj şifrelenir ve en sonunda iletim için hazırlanır.

PGP de kullanılan kriptografik algoritmalar ise MD5, RSA ve IDEA olarak bilinir. Bu çalışmada IDEA algoritması yerine AES kullanılmıştır. MD5 algoritmasının çalışması incelenmiş ve gerçekleştirme aşamasındaki adımlar ayrıntılı bir şekilde anlatılmıştır.

Devamında PGP'nin gerçekleştirilmesi sırasında verinin öncelikle MD5, sonra RSA ve en son AES'ten geçmesi sağlanacak şekilde bloklar birleştirilmiş, sonuç test edilmiş ve devre FPGA üzerinde gerçeklenmiştir.

Anahtar Kelimeler—PGP, MD5, AES, RSA, FPGA

I. GİRİŞ

Elektronik posta ya da e-mail, bilgisayar ağlarında kullanıcıların birbirleriyle haberleşmesini sağlayan ve kullanımı en yaygın olan yöntemlerden biridir. 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 ortadan kaldırmak için kullanılan temel yöntem kriptografidir ve e-posta için günümüzde kullanılan en yaygın protokollerden biri PGP (Pretty Good Privacy) dir.

PGP güvenlik ve kimlik doğrulamayı sağlayan e-posta güvenlik protokolüdür [1]. Açık anahtarlı şifreleme kullanarak kullanıcıya gizlilik ve kimlik doğrulaması sağlaması, PGP'nin tercih edilmesini arttırmış ve 1991'de ilk olarak piyasaya çıktığından bu yana en popüler e-posta güvenlik protokollerinden biri haline gelmiştir.

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

Güvenilirlik: E-postanın içeriği yetkilendirilmemiş kullanıcılara karşı korunur. Yetkisi olan kullanıcı hariç kimse postayı okuyamaz. Posta alıcının posta kutusuna gelene kadar şifreli halde bulunur. Posta kutusuna geldikten sonra şifreli olmasına gerek yoktur ve alıcı korumasız postaya 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. Postayı 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.

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

II. PGP

A. Temel Yapısı

PGP e-posta güvenlik protokolü, kriptografik algoritma olarak açık anahtar şifrelemesini ve anahtar yönetimi için sayısal imzayı, veri şifrelemesi için özel anahtar kriptografisini ve sayısal imza için tek yönlü özet 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 kullanıldığı görülür. PGP e-posta güvenlik protokolünün FPGA üzerinde gerçekleştirilmesinde IDEA yerine AES algoritması kullanılmıştır.

PGP protokolünün 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 paylaşmak zorunda kalmaz. Mesajın gönderici tarafından gönderilebilmesi için alıcının açık anahtarına sahip olması yeterlidir.

B. PGP Mesajının Gönderilmesi

Gönderilen PGP mesajı dört adımdan oluşur. İlk adım imzalama adımıdır [1]. Sıkıştırma kısmı mesajın boyutunun azalması için gerçekleştirilen kısmıdır ve algoritması sıkıştırma programı ZIP 2.0 ile aynıdır. Sıkıştırma sonrası

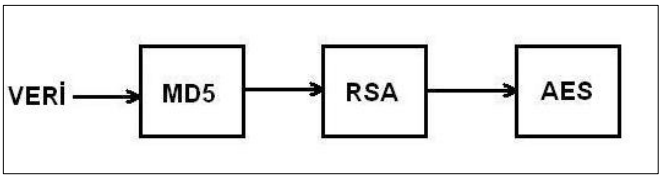
gerçekleştirilen adım şifreleme kısmıdır.. PGP sisteminin şifreleme kısmı FPGA üzerinde gerçeklenmiştir. Şifreleme kısmında kullanılan algoritmalar imzalama kısmındaki algoritmalarla aynıdır. Sadece imzalama algoritmalarına ek olarak AES algoritması ile imzalanan metin şifrelenir.

1) PGP Mesajlarının İmzalanması

PGP mesajları postayı 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 özeti bulunmuş mesaj gönderenin özel anahtarı ile şifrelenir ve şifrelenmiş özet mesaja eklenir. Bu aşamada aşağıda anlatılan PGP mesajlarının şifrelenmesi adımından farklı olarak imzalanmış mesaj, şifrelenmeden RSA bloğunun çıkışında elde edilmektedir.

PGP ile bir mesaj şifrelenebilir ve imzalanabilir. 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



Şekil 2.1: PGP'nin imzalama aşamasında kullandığı bloklar

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:

III. KRİPTOGRAFİ VE PGP'DE KULLANILAN KRİPTOGRAFİK YÖNTEMLER

A. 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 ve asimetrik anahtarlama kriptografisi.

B. RSA Algoritması

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

Modüler üs alma işlemi 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. Matematiksel algoritma aşağıdaki gibidir:

Hem asal p ve q seçilir. İkisinin çarpımı k bitlik katsayı N'yi meydana getirir: $N = pq$, $p \neq q$, $2^{k-1} < N < 2^k - 1$.

Bir E seçilir. Bu seçime göre E ve $\phi(N)$'nin en büyük ortak böleninin 1 olması ve E'nin N'den küçük olması gerekir: $gcd(E, \phi(N)) = 1$, $E \in \{1, \ldots, N-1\}$. N'nin Euler totient fonksiyonu uygulanır: $\phi(N) = (p-1)(q-1)$

Özel anahtar D hesaplanır: $D = E^{-1} mod(\phi(N))$.

$mod\ N$ 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: $C = M^E\ mod\ N$ $C, M, E \in \{0, 1, \ldots, N-1\}$

RSA'da şifre çözme işlemi de şifreleme işleminde kullanılan aynı algoritma ile gösterilir: $M = C^D\ mod\ N$ $C, M, E\ \varepsilon\ \{0, 1, \ldots, N-1\}$

C. AES Algoritması

AES (Advanced Encryption Standard) algoritması bir simetrik anahtarlama teknolojisi olup verinin şifrelenmesi ve şifrelenmiş bilginin çözülmesi için kullanır [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 işleminden geçirilir ve 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.

D. MD5 Algoritması

MD5 128 bitlik özet değerinden oluşan kriptografik özet fonksiyonudur [5]. Algoritma keyfi uzunluktaki bir mesajı alıp bu mesajdan 128 bitlik bir mesaj özeti çıkarır. Farklı iki mesajdan aynı mesaj özetinin oluşması ya da verilmiş bir mesaj özetinden mesajın kendisinin hesaplanması matematiksel olarak imkansızdır.

MD5 algoritması, dijital imza uygulamalarında, RSA gibi bir açık anahtar kriptografisiyle dosyanın özel anahtarla şifrelenmeden önce sıkıştırılması gerektiğinde kullanılır.

MD5 algoritmasının içeriği şu şekildedir:

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

Çıkış *x*'in 128 bit uzunluklu bir özet fonksiyonu olur.

Rakamsal Gösterimler: MD5'in gerçekleştirilmesi sırasında kullanılan fonksiyonlar aşağıda tanımlanmıştır: $f(u,v,w) = (u \wedge v) \vee (u` \wedge w)$ $g(u,v,w) = (u \wedge w) \vee (v \wedge w`)$, $h(u,v,w) = u \oplus v \oplus w$, $k(u,v,w) = v \oplus (u \vee w`)$.

Sabitlerin Tanımlanması: Algoritmanın gerçekleştirilmesinde kullanılan değerlerden olan 32 bitlik 4 adet sabit değer tanımlanır: $h_1 = 0x67452301$, $h_2 = 0xefcdab89$, $h_3 = 0x98badcfe$, $h_4 = 0x10325476$.

32 bitlik bir *y[j]* sabiti $0 \le j \le 63$ 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]
z[16..31] = [1,6,11,0,5,10,15,4,9,14,3,8,13,2,7,12]
z[32..47] = [5,8,11,14,1,4,7,10,13,0,3,6,9,12,15,2]
z[48..63] = [0,7,14,5,12,3,10,1,8,15,6,13,4,11,2,9]

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]
s[16..31] = [5,9,14,20,5,9,14,20,5,9,14,20,5,9,14,20]
s[32..47] = [4,11,16,23,4,11,16,23,4,11,16,23,4,11,16,23]
s[48..63] = [6,10,15,21,6,10,15,21,6,10,15,21,6,10,15,21]

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 $b \bmod 2^{64}$'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≥0)* adet 0 biti ile doldurulur. Bu şekilde 512'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: $x_0x_1x_2....x_{(16m-1)}$.

En son başlangıçtaki sabitler yeni bir yere yüklenir: $(H_1, H_2, H_3, H_4) \leftarrow (h_1, h_2, h_3, h_4)$.

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: $X[j] \leftarrow x_{(16i+j)}$.

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

Birinci dizi $0 \le j \le 15$ olması durumunda gerçeklenir: $t \leftarrow (A+f(B,C,D)+X[z[j]]+y[j])$, $(A,B,C,D) \leftarrow (D,B+(t \lll s[j]),B,C)$.

İkinci dizi $16 \le j \le 31$ olması durumunda gerçeklenir: $t \leftarrow (A+g(B,C,D)+X[z[j]]+y[j])$, $(A,B,C,D) \leftarrow (D,B+(t \lll s[j]),B,C)$.

Üçüncü dizi $32 \le j \le 47$ olması durumunda gerçeklenir: $t \leftarrow (A+h(B,C,D)+X[z[j]]+y[j])$, $(A,B,C,D) \leftarrow (D,B+(t \lll s[j]),B,C)$.

Dördüncü dizi $48 \le j \le 63$ olması durumunda gerçeklenir: $t \leftarrow (A+k(B,C,D)+X[z[j]]+y[j])$, $(A,B,C,D) \leftarrow (D,B+(t \lll s[j]),B,C)$.

Bu aşamalar bittikten sonra zincir güncellenir: $(H_1,H_2,H_3,H_4) \leftarrow (H_1+A,H_2+B,H_3+C,H_4+D)$.

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.

IV. PGP'NİN GERÇEKLENMESİ

A. MD5 Algoritmasının Oluşturulması

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 $|sin(j+1)|$'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' modülünde gerçeklenmiştir.

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}'e gore sonucu, 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 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 işlemler sonucunda MD5'in üçüncü adımı da bitmiş olur. Bu aşama daha sonraki aşamanın giriş işareti olarak kullanılacaktır.

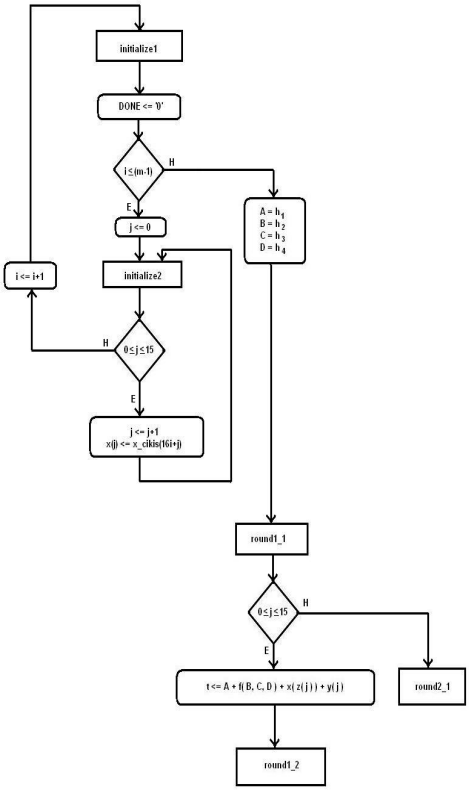
3) Durum Makinesi Modülünün Oluşturulması

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 VHDL ile en iyi şekilde gerçekleştirebilmek için değişik süreçlerin gerçekleştirileceği bir durum makinesinin tasarlanması uygun görüldü. Durum makinesinin tasarlanacağı modüle 'durum makinesi (state machine)' adı verildi.

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



Şekil 4.1: Durum makinesi modülündeki durumlardan ilk üçü

bir düzende kullanılacak şekilde oluşturulmasıdır. Burada *0≤i≤m-1* şeklinde *i* değeri tanımlanırken, *0≤j≤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 durumunda 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.

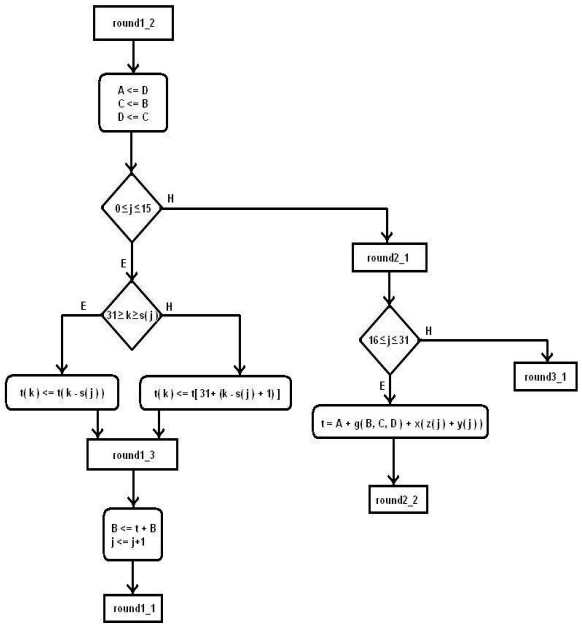
Şekil 4.1'de verilen blok şemasına göre bu işlemler initialize1 ve initialize2'de gerçekleştirilmektedir:

Ara bağlantı olarak kullanılan *A, B, C, D* güncellenip çıkış değerine aktarılması için öncelikle 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.

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 '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 ve bu işlemlerin hepsinin 1 saat darbesi kadar sürede yapılması sağlandı.

Bir sonraki saat darbesinde yapılması istenen *A, C* ve *D* değerlerine sırasıyla *D, B* ve *C* değerlerinin aktarılması ve *B*'ye ise, *t* değerine *s[j]* değerinin temsil ettiği değer kadar öteleme yapıldıktan sonra, ötelenmiş t'nin önceki B değeri ile mod 2^{32} işlemine sokulup 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 sayıda sola kaydırılma işlemi yapıldı. Bunun devamında kaydırılan kısımları doldurabilmek için en sondaki sayılar bu boşluklara yerleştirildi. Bu şekilde kaydırma işlemi bir saat darbesi süresinde tamamlanmış oldu. Şekil 4.2'de durum



Şekil 4.2: Durum makinesinde dizilerin gelme durumu makinasımda dizilerin gelişi gösterilmiştir.

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) 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.

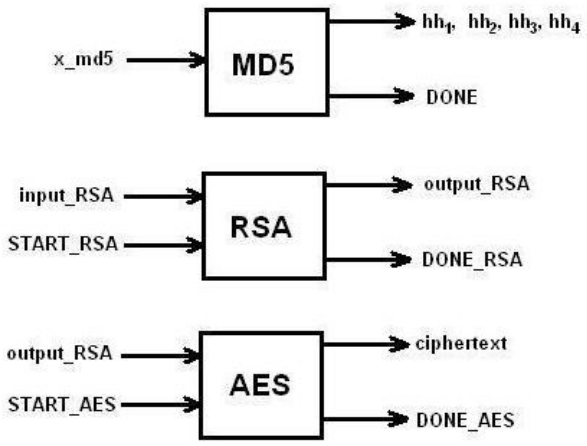
Bu işlemlerden sonra sıra algoritmanın test aşamasına gelmiştir. Bunun için öncelikle test için yeni bir modül yazıldı. Modülde ilk önce tüm devrenin 50ns boyunca resetlenmesi sağlandı. Bu sayede kaydedilmiş olabilecek sayıların sıfırlanması amaçlandı ve devre başlangıç durumuna getirilmiş oldu. Daha 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 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.

B. 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 4.3'te verilmiştir.



Şekil 4.3: PGP bloklarında kullanılan modüllerin giriş ve çıkışları

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. Şekil 4.3'te bu blokların giriş ve çıkışları gösterilmiştir. Tüm bu aşamalar farklı saat darbelerinde gerçekleştiğinden bunların bir durum makinesi içerisinde tanımlanması gerekir. Durum makinesinin zaman diyagramı Şekil 4.4'teki gibidir:

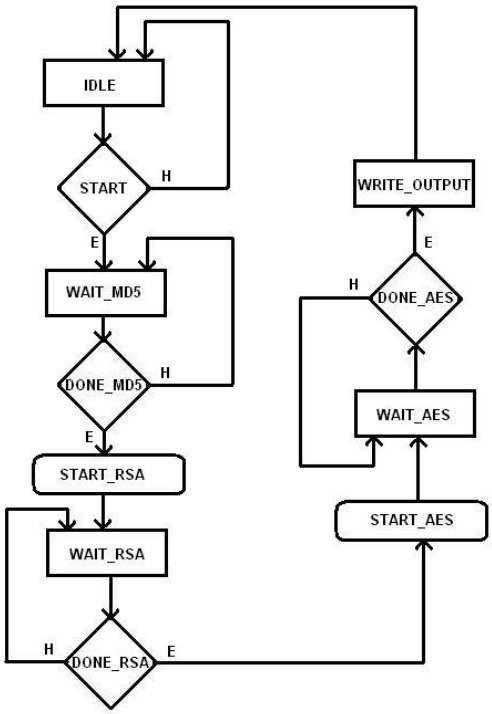
PGP ile güvenli e-posta gönderim sistemi FPGA üzerinde gerçekleştirilmiştir. Bu gerçekleştirmeler sonucunda 19347 dilimlik, 35558 LUT'luk bir alan kullanılmıştır. Devrenin gerçeklenmesi suresinde harcanan maksimum saat frekansı da 63.731MHz olarak gözlenmiştir.

V. 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 hangi kriptografik algoritmaların ne şekilde kullandığı ve bu algoritmaların çalışmaları sırasında gerçekleştirilen aşamalar incelenmiştir. Bu sayede bilginin ne kadar güvenli bir şekilde gönderilmiş olduğu, bir başkasının sadece bize ait olan bilgiyi biz istemedikçe göremeyeceği öğrenilmiştir. Bunun yanında MD5 algoritması ayrıntılı bir şekilde incelenmiş ve donanımsal olarak gerçeklenebilmesi için gereken çalışmalar yapılmıştır.

Yapılan çalışmalar ileride bu konuda yapılacak olan çalışmalara kaynak olacak niteliktedir. Bu konu hakkında şimdiye kadar sadece bir adet makalenin yazılmış olması



Şekil 4.4: PGP bloğu durum akış diyagramı

nedeniyle bu bitirme çalışması daha da önem kazanmaktadır.

VI. 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.