

1. GİRİŞ

1.1 Giriş ve çalışmanın amacı

Gelişen teknolojiyle internetin kullanımı her geçen gün daha da yaygınlaşmakta olup, internette yollanan veri paketleri birçok dışarıya açık networklerden geçmektedir. Gizli ve başkaları tarafından ulaşılması istenilmeyen bilgilerin internette bir yerden bir yere nakledilmesi ciddi anlamda güvenlik kaygısını da beraberinde getirmektedir. Bu tür bilgileri korumak mümkün olmadıkça, gizli ve şahsi yazışmalarda bulunmak da mümkün olmayacaktır. Bilgi güvenliği, başkası tarafından dinlenme, bilginin değiştirilmesi, kimlik taklidi gibi tehditlerin ortadan kaldırılması ile sağlanır ve bu amaç için geliştirilen temel bilim kriptografidir. Kriptografi, bilgi güvenliğini sağlayan temel bilim dalıdır. Güvenilirlik, veri bütünlüğü, kimlik doğrulama gibi bilgi güvenliği konularıyla ilgilenen matematiksel yöntemler üzerine yapılan çalışmalar, kriptografinin önemli konularıdır.

Kriptografi genel olarak şu ana konularla ilgilenir:

Gizlilik: Bilgi istenmeyen kişiler tarafından anlaşılmalıdır.

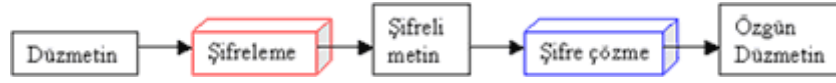
Bütünlük: Bir iletinin alıcısı bu iletinin iletim sırasında değişikliğe uğrayıp uğramadığını öğrenmek isteyebilir. Dışarıdan biri, doğru iletinin yerine yanlış bir ileti koyma şansına erişmemelidir. Saklanan veya iletilmek istenen bilgi farkına varılmadan değiştirilememelidir.

Reddedilemezlik: Bilgiyi oluşturan ya da gönderen, daha sonra bilgiyi kendisinin oluşturduğunu veya gönderdiğini inkâr edememelidir. Bir gönderici daha sonrasında bir ileti göndermiş olduğunu yanlışlıkla reddetmemelidir.

Kimlik belirleme: Gönderen ve alıcı, birbirlerinin kimliklerini doğrulayabilmelidir. Dışarıdan biri, bir başkasının kimliğine bürünme şansına erişmemelidir.

Bir kriptografik sistem, bilgi güvenliğini sağlamak için bir araya getirilmiş birçok küçük yöntemler bütünlüğü olarak görülebilir. Bu sistemler belirli bir algoritma kullanılarak şifre üretme yöntemleridir ve çok çeşitlidir. Bir algoritmaya dayanılarak üretilen şifre ile iletilmek istenen mesaj şifrelenir ve alıcı tarafa gönderilir. Alıcı tarafta ise yine bir şifre kullanılarak şifrelenmiş metin çözülür ve gönderilmesi istenen mesaj elde edilir. Gönderen tarafın mesajı şifreleme işlemine şifreleme

(encryption), alıcı tarafta yapılan şifre çözme işlemine ise deşifreleme (decryption) denir ve şekil 1.1 ile gösterilmiştir.



Şekil 1.1: Şifreleme ve şifreyi çözme işlemleri diyagramı

Şifreleme teknikleri, blok şifreleme ve dizi şifreleme olarak ikiye ayrılır. Dizi şifrelemede bir anda mesajın bir biti şifrelenirken, blok şifrelemede bir anda bit bütünü, blok halinde şifrelenir.

Dizi şifreleme de kendi içinde senkronize ve kendiyle senkronize olmak üzere ikiye ayrılır. Senkronize şifrelemede şifre algoritmasında anahtar ve başlangıç vektörü kullanılırken, kendiyle senkronize şifrelemede ise bunlara ek olarak daha önceden şifrelenmiş şifreli mesaj bitleri de kullanılmaktadır.

Şifreleme algoritmalarının gerçekleşmesinde yaygın olarak kullanılan sahada programlanabilir kapı dizileri (FPGA) lojik bloklardan ve flip-flop'lardan oluşan ve bunların arasında elektriksel programlanabilir bağlantılar bulunduran iki boyutlu bir dizidir. Aralardaki bağlantılar, elektriksel programlanabilir anahtarlardır. FPGA, kullanıcının, hem aradaki bağlantıları programlayabildiği, hem de içinde bulunan her bir lojik bloğu ayrı ayrı programlayabildiği bir tümdevre elemanıdır.

Mosquito algoritması, şifreleme yöntemlerinden biri olan “kendiyle senkronize dizi şifreleme” yöntemine örnek bir algoritmadır. Bu algoritmada, anahtar olan K ve başlangıç vektörü olan IV (initialization vector) kullanılarak yedi aşamada bir bit şifre üretilmektedir. Bu işlemler için kullanılmak üzere, devrenin girişini her defasında farklı kılacak bir ötelemeli kaydedici, giriş ile aldığı bilgiyi güncelleyebilmesi için bir koşullu tamamlayan ötelemeli kaydedici ve yedi aşamanın her biri için yedi adet kombinezonsal devre tasarlanmıştır.

Tezde, Mosquito algoritmasının çok yüksek seviyeli donanım tanımlama dilleri (VHDL) kullanılarak kodlanması ve FPGA üzerinde gerçekleşmesinin aşamaları anlatılacaktır.

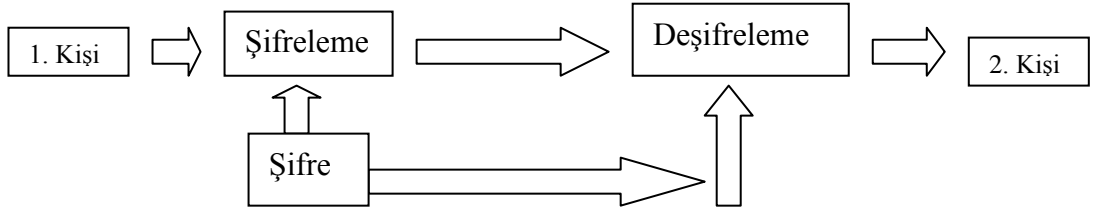
Tezin ikinci bölümünde kriptografi ve bir kriptografik sistem türü olan dizi şifreleme hakkında detaylı bilgi verilmiştir. Üçüncü bölümde algoritmanın gerçekleşmesinde kullanılacak FPGA'lar hakkında daha geniş bilgi verilmiş, yapıları ve programlanmaları anlatılmıştır. Tezin dördüncü bölümünde ise Mosquito algoritması, algoritmanın gerçekleştirme adımları ve sonuçları verilmiştir.

2. KRİPTOGRAFİ VE DİZİ ŞİFRELEME

2.1. Kriptografi

Kriptografi, okunabilir durumdaki bir bilginin istenmeyen taraflarca okunamayacak bir hale dönüştürülmesinde kullanılan tekniklerin tümü olarak görülebilir. Bu teknikler, bir bilginin iletimi esnasında karşılaşılabilecek aktif ya da pasif ataklardan bilgiyi, dolayısıyla bilgi ile beraber bilginin göndericisi ve alıcısını da koruma amacı güderler.

Bir başka deyişle kriptografi, iki kişinin, güvenli olmayan bir kanal üzerinden üçüncü bir kişinin anlayamayacağı bir şekilde haberleşebilmesi amacıyla tekniklerin geliştirilmesidir. Burada güvenli olmayan kanal telefon ya da şebeke hattı olabilir. Gönderilmek istenen mesaj, önceden belirlenmiş bir şifre ile şifrelenerek kanal üzerinden gönderilir ve alıcı tarafta aynı şifre ile deşifre edilir. İki tarafın da ihtiyaç duyduğu şifre ise ayrı olarak güvenli bir kanal üzerinden gönderilmelidir. Bu gönderim ve şifreleme işlemleri Şekil 2.1’de gösterilmiştir [2].



Şekil 2.1: Haberleşme kanalı

Şifreleme fonksiyonunun $e(x)$, Deşifreleme fonksiyonunun ise $d(y)$ olduğu düşünülürse şifrelenmiş metin;

$$Y = Y_1 Y_2 Y_3 \dots Y_n = e(X_1) e(X_2) e(X_3) \dots e(X_n) \quad (2.1)$$

deşifre edilmiş metin ise;

$$X = X_1 X_2 X_3 \dots X_n = d(Y_1) d(Y_2) d(Y_3) \dots d(Y_n) \quad (2.2)$$

şeklinde elde edilir.

Kullanılan şifreler belli bir algoritmaya göre oluşturulur. Kullanılan algoritmanın değerlendirilmesindeki kriterler; güvenlik seviyesi, fonksiyonellik, işlem metodu, performans ve gerçekleştirme kolaylığıdır [3].

Algoritmalar şifreleme ve çözme işlemlerini kontrol etmek için bir anahtar kullanırlar. Anahtar temelli algoritmaların iki çeşidi vardır; simetrik (gizli-anahtar) ve asimetrik (açık-anahtar) algoritmaları. Simetrik algoritmalar şifreleme ve çözme işlemleri için aynı anahtarı kullanırken, asimetrik algoritmalar şifreleme ve çözme için farklı anahtar kullanırlar ve çözme anahtarı şifreleme anahtarından elde edilemez.

Simetrik algoritmalar dizi şifreleme (stream cipher) ve blok şifreleme (block cipher) olarak ikiye ayrılabilir. Dizi şifrelemede belli bir anda bir bitlik metin şifrelenebilirken, blok şifrelemede ise pek çok biti- alınıp (tipik olarak 64 bit) bunlar tek bir birim olarak şifrelenirler. [7]

2.2 Dizi Şifreleme

Dizi şifreleme, mesajdaki her harfin özel bir algoritma kullanılarak anahtardan üretilen bir anahtar dizisi ile sırayla şifrlenmesidir [3].

Dizi şifrelemede mesaj metni $X_1 X_2 X_3 \dots X_n$ ve şifre dizisi $Z_1 Z_2 Z_3 \dots Z_n$ iken X_1 biti Z_1 ile, X_2 biti Z_2 ile şifrelenir.

En genel şifre üretme yönteminde şifre, belirli bir algoritma ile oluşturulur. Algoritma sonucunda belirli uzunluktaki K şifresinden bir anahtar dizisi oluşturulur ve bu sayede bit sayısı olarak daha kısa bir şifre (K), daha uzun ve güvenli bir şifre dizisine dönüşür. Bu dönüşüm sonucunda (2.3) denkleminde görülen n bitlik Z şifre dizisi elde edilir.

$$G(K) = Z_1 Z_2 Z_3 \dots Z_n \quad (2.3)$$

Birçok şifreleme algoritması, bir bitin daha önce üretilen bitlere bağlı olması mantığı üzerine kuruludur. Burada amaç, şifrelemenin daha güvenli olması için, periyodu daha büyük şifre dizileri elde etmektir.

Şifrelenecek metnin bitlerinin ve şifre bitlerinin 1 ya da 0 olduğu ve metin biti ve şifre arasında yapılan işlemin modül 2 ye göre toplama işlemi olduğu

düşünüldüğünde (2.4) ve (2.5) denklemleri gereği bit düzeyinde uygulanan işlemin “Ayrıcalıklı Veya” (XOR) işlemi olacağı açıkça görülür [1].

$$e(x) = x + z \quad (2.4)$$

$$d(y) = y + z \quad (2.5)$$

Bir şifreden bir algoritmaya dayanarak bir şifre dizisi oluşturmak ve bu dizinin bitleri ile mesaj metninin bitleri ayrıcalıklı veya işlemine tabi tutularak yapılan bir şifreleme yöntemi olan dizi şifreleme iki türe ayrılır. Bu türlerden birincisi olan senkronize dizi şifrelemede üretilecek şifre sadece anahtar K'ya ve başlangıç vektörü (initialization vector) olan IV'ye bağlıdır. Diğer tür olan kendiyle senkronize dizi şifrelemede (Self-synchronizing stream encryption) ise üretilecek şifre, K ve IV'nin yanı sıra önceden üretilmiş şifre ile şifrelenmiş metine de bağlıdır [7]. 4. bölümde anlatılacak olan Mosquito algoritması, kendiyle senkronize dizi şifreleme algoritmasına bir örnek niteliğindedir.

Bir t anında şifrelenecek mesaj m^t , üretilmiş şifre z^t iken şifrelenmiş metin denklem (2.6)'dan uyarınca c^t olsun.

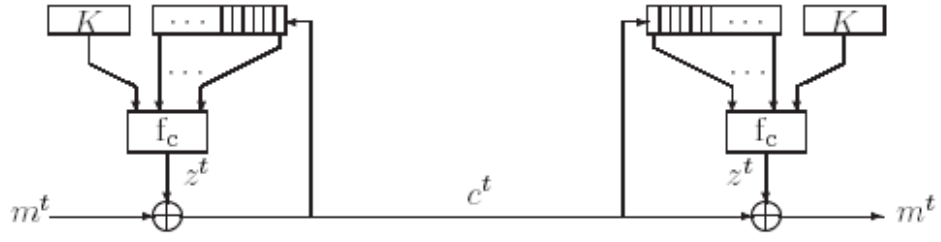
$$c^t = m^t \cdot z^t \quad (2.6)$$

Kendiyle senkronize dizi şifrelemenin en önemli özelliği, denklem 2.7 den de görüldüğü gibi, üretilen şifre biti olan z^t 'nin, şifre K'nın yanı sıra daha önceden şifrelenmiş mesaj bitleri olan $c^{t-1} \dots c^{t-n_m}$ ye bağlı bir fonksiyonu ile üretiliyor olmasıdır [1].

$$z^t = f_c[K](c^{t-n_m} \dots c^{t-1}) \quad (2.7)$$

Bir başka deyişle algoritma ile üretilecek şifre, yine o algoritma ile üretilmiş şifre ile şifrelenmiş metin ile üretilmektedir. Kendiyle deyişimi bu sebepten ötürüdür.

Eşitlik 2.7'deki n_m , şifre bitinin üretilmesi için gerekli olan şifrelenmiş metnin bit sayısıdır. Kendiyle senkronize dizi şifrelemenin blok diyagramı olan şekil 2.2'den de görüleceği gibi şifre, anahtar (K) ve daha önceki anlarda şifrelenmiş metnin bir fonksiyonudur.



Şekil 2.2: Kendiyle senkronize dizi şifreleme blok diyagramı

Eşitlik 2.7'den görüleceği gibi bir t anında z^t 'nin üretilmesi için daha önceki anlarda elde edilmiş n_m adet c 'ye ihtiyaç vardır. Fakat şifreleme işlemine başlanıldığında n_m adet c elde edilmiş değildir ve başlangıçta kullanılmak üzere, n_m uzunluğunda bir bit dizisi gerekir. Bu problemi karşılamak için, şifrelenmiş mesaj gibi davranan bir "başlangıç vektörü" (IV) denilen, n_m uzunluğunda bir bit dizisi kullanılır. Böylece şifreleme işlemi için gerekli olan n_m adet c sağlanmış olur. IV, sadece başlangıç ihtiyacını karşılamak için kullanılır. n_m adet şifre üretildiğinde ise IV'ye olan ihtiyaç biter ve üretilen şifreler ile şifrelenmiş mesaj (c) kullanılmaya başlanır [1]. Şifreleme ve deşifreleme işlemleri için IV'nin aynı şifre gibi, her iki tarafta da bulunması gereklidir fakat şifreden farklı olarak gizli tutulmasına ihtiyaç yoktur.

En genel haliyle bir mesaj m 'nin K anahtarı ve IV başlangıç vektörü ile şifrelenmesi aşağıdaki 6 adım ile anlatılabilir:[7]

1. Ayrıştırma: Bu aşamada mesaj m , $m_1m_2m_3\dots m_n$ formuna ayrıştırılır.
2. Başlangıç: Şifre üreticisine anahtar K ve başlangıç vektörü IV verilir. Sayıcı 0'a ayarlanır
3. Şifre üretme: Şifre üretici $z_1z_2z_3\dots z_n$ şifre bloğunu üretir.
4. Şifre bloğu ile mesaj şifrelenerek her m_i ve z_i için bir c_i elde edilir.
5. Döngü: $i < n$ ise 3. adıma dönülür ve $i = n$ oluncaya yani tüm mesaj şifreleninceye kadar adımlar tekrarlanır.
6. Çıkış: Şifrelenmiş mesaj olan $c_1c_2c_3\dots c_n$ elde edilir.

3. SAHADA PROGRAMLANABİLİR KAPI DİZİLERİ

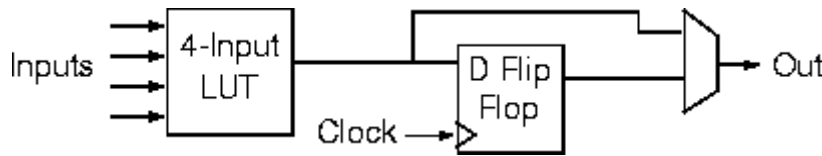
3.1. Sahada Programlanabilir Kapı Dizileri Hakkında Genel Bilgi

Sahada Programlanabilir Kapı Dizileri (Field Programmable Gate Array, FPGA) yaygın olarak kullanılan, geniş uygulama alanlarına sahip programlanabilir tümdevrelerdir. FPGA kullanımıyla, temel lojik kapıların (AND, OR) ya da çözücü gibi daha karmaşık yapıdaki devre elemanlarının fonksiyonelliği artırılmaktadır.

FPGA'lar, programlanabilir lojik bloklar ve ara bağlantılardan oluşur. Lojik blokların ve ara bağlantıların imalat aşamasından sonra tasarımcı tarafından programlanabilmesi sebebiyle sahada programlanabilir adını taşımaktadır [5]. Tasarımcının ihtiyacı olan lojik fonksiyonları gerçekleştirebilmesi amacıyla sahada programlanabilir olarak üretilmiştir. Kullanıcının tasarladığı lojik devreye göre, tümdevre üreticisi tarafından sağlanan bir yazılım sayesinde lojik bloklar ve aralarındaki bağlantılar programlanır [4]. Tasarım sırasında kullanıcıya sağladığı esneklik, düşük maliyet ve hızlı ilk üretme özelliği ile FPGA'lar sayısal tasarım ortamlarının vazgeçilmez yapıları haline gelmiştir. Uygulama alanlarından bazıları sayısal işaret işleme, kriptografi, uzay, savunma ve tıbbi görüntülemedir [5].

3.2. Sahada Programlanabilir kapı dizilerinin yapısı

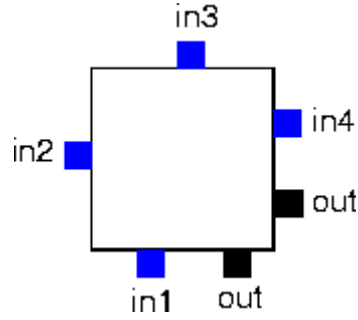
Tipik FPGA lojik bloğu şekil 3.1'de görüldüğü gibi 4 giriş (Look up table, LUT) ve bir flip-flop'tan oluşur. FPGA lojik bloğunun giriş-çıkışları ise şekil 3.2'de gösterilmektedir.



Şekil 3.1: FPGA lojik bloğu

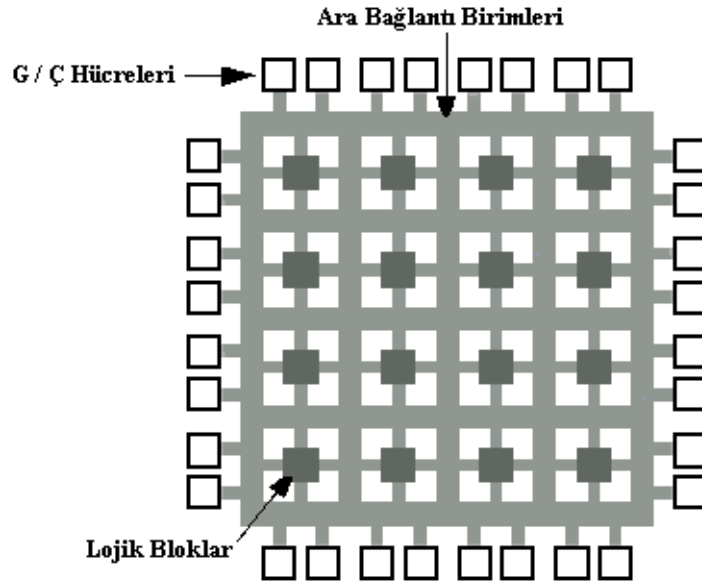
FPGA, düzenlenebilir lojik bloklar (configurable logic blocks, CLB), giriş / çıkış blokları (Input / Output Blocks, IOB) ve ara bağlantılar olmak üzere üç tane önemli

düzenlenebilir elemana sahiptir [3]. FPGA'ların programlanması ara bağlantıların nasıl olacağını belirleme ve FPGA bellek hücrelerinin programlanması işlemidir.



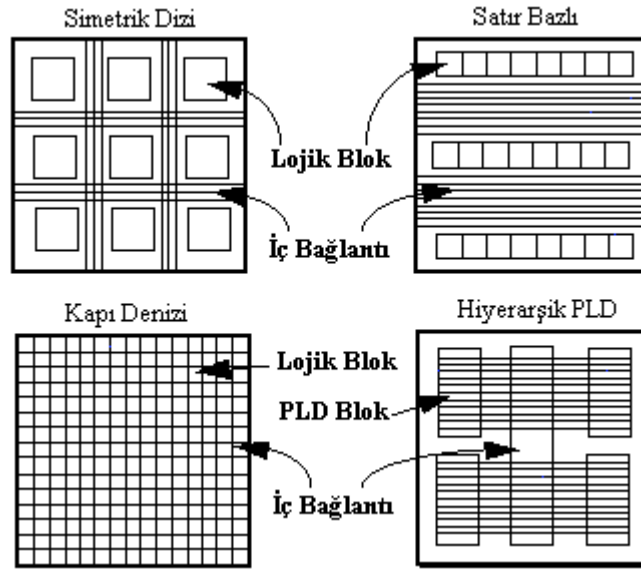
Şekil 3.2: FPGA bloğu (giriş- çıkış)

Şekil 3.3'te FPGA da bulunan lojik bloklar ve ara bağlantı birimleri görülmektedir.[3]



Şekil 3.3: FPGA genel yapısı

FPGA'lar bağlantı çeşitlerine göre simetrik dizi, sıra tabanlı, hiyerarşik PLD ve kapı denizi olmak üzere dörde ayrılırlar. (Bkz. Şekil 3.4). [6]



Şekil 3.4: Bağlantı şekillerine göre FPGA'lar

3.3. Sahada Programlanabilir kapı dizilerinin programlanması

FPGA'lar genellikle yüksek seviyeli donanım tanımlama dilleri (Hardware Description Language, HDL) kullanılarak programlanır. Yazılan program derlenerek simülasyonu yapılır. Simülasyon aşamasında beklenen sonuç alındıktan sonra sentezleme aşamasına geçilir. Sentezleyici, kullanıcının belirtmiş olduğu FPGA elemanına göre gerekli bağlantıları çıkartır ve kullanıcıya bu bağlantıları liste halinde sunar. Yerleştirme ve yönlendirme sonrası istenilen sonuçlar elde edildikten sonra FPGA'nın programlanması aşamasında kullanılacak bit dizisi, üreticinin sağladığı yazılımla elde edilir. FPGA'nın uygun donanım kullanılarak programlanmasıyla tasarım tamamlanır [3-4, 6]. En genel anlamda bir tasarım süreci ve tasarlanan fonksiyonun bir FPGA'ya entegre edilmesi beş başlık altında toplanabilir:

1. Sistem tasarımı: Bu aşamada tasarımcı FPGA'ya uygulanacak olan fonksiyona ve bu fonksiyonun sistemin geri kalan kısmıyla nasıl bütünleneceğine edileceğine karar verir.
2. Giriş-çıkış entegrasyonu: FPGA'nın giriş ve çıkışlarının sisteme yani devreye entegre edildiği aşamadır.
3. Tasarımı tanımlama: Bu aşamada tasarımcı, tasarladığı fonksiyonu şemalar ya da HDL kullanarak tanımlar.

4. Sentezleme: Bu aşamada optimizasyon, lojik blokların yerleşimi ve ara bağlantıların belirlenmesi işlemleri yapılır.
5. Tasarımı doğrulama: Bu aşama test aşamasıdır. Elde edilen sonuçların simülasyon sonuçlarıyla karşılaştırılması bu aşamada yapılır.

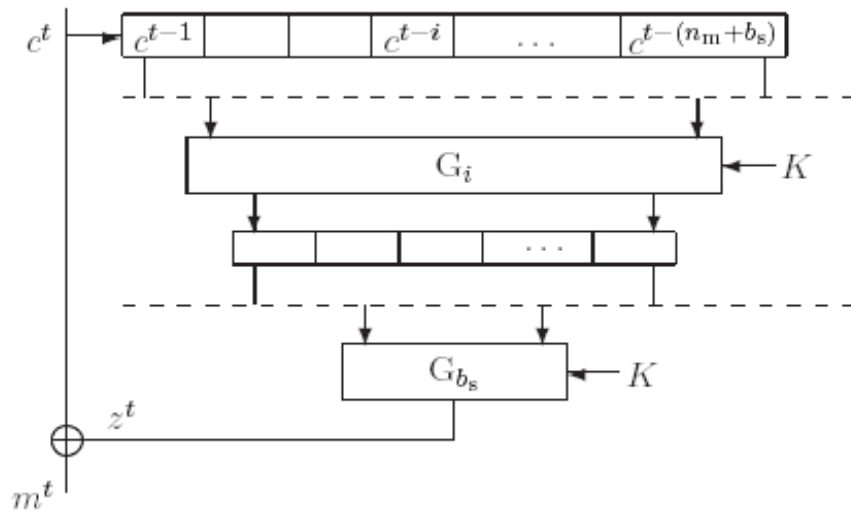
4. MOSQUITO ALGORİTMASI VE GERÇEKLENMESİ

4.1. Mosquito Algoritması

Kendiyle senkronize dizi şifreleme tipine bir örnek olan Mosquito algoritmasında, koşullu tamamlayan ötelemeli kaydedici (Conditional complementing shift register, CCSR) kullanılmaktadır ve ardışık düzen kullanılarak gerçekleştirilen yedi aşama vardır. Bu yüzden bu bölümde önce CCSR ve ardışık düzen hakkında genel bilgi verilecek, daha sonra Mosquito algoritmasında kullanılan CCSR ve uygulanan ardışık düzen anlatılacaktır.

4.1.1. Ardışık düzen (Pipelining)

Şifre, dayanıklı olması amacıyla birçok aşamadan geçirilerek üretilmek istenir. Bu aşamalar şekil 4.1’ de görüldüğü gibi, G_i ile gösterilen b_s adet aşamadır.



Şekil 4.1: Şifre oluşturuluş aşamaları

Donanımsal olarak, her aşama bir kombinezonsal devre ve ara sonucu saklayan bir bellekten oluşur. Devrenin b_s aşamadan oluşması, ilk aşama ile son aşama arasında b_s zaman farkı oluşmasına neden olur. Bu zaman farkı dolayısı ile üretilen z^t , 2. bölümde bahsedildiği gibi $c^{t-nm} \dots c^{t-1}$ 'e bağlı değil de $c^{t-nm} \dots c^{t-(bs+1)}$ 'e bağlı hale gelir. Ardışık zaman nedeniyle n_m uzunluğundaki giriş hafıza belleği, $n_m + b_s$ uzunluğuna çıkar. Fakat bu artış, üretilen z 'nin fazla olan b_s adet bitten bağımsız olacağı nedeniyle giriş uzunluğunu değiştirmez. b_s şifre fonksiyonunun gecikmesi olarak da bilinir [1].

Bu ardışık düzenin ilk aşamasında giriş, ötelemeli kaydedicide bulunan $n_m - b_s$ bitlik şifrelenmiş metindir. Böylece üretilecek olan z 'nin, sadece bu $n_m - b_s$ bitlik diziye ve anahtar olan K 'ya bağlı olması garantilenmiş olur. Ötelemeli kaydedicinin n_m uzunluğunda bir iç bellekli bir sonlu durum makinesi ile değiştirilmesi, devrenin propagasyon özelliklerini geliştirecektir. q bir durum ve G durum güncelleme fonksiyonu olarak düşünüldüğünde denklem 4.1'den de görüleceği gibi bir $t+1$ anındaki durum, t anındaki duruma ve yine t anındaki şifrelenmiş mesaj bitine bağlı olacaktır.

$$q^{t+1} = G(q^t, c^t) \quad (4.1)$$

Durum q , bağlı olduğu şifre mesaj biti sayısına eşit iç belleklerle ilişkilendirilmiş parçalara ayrılırsa, j adet şifrelenmiş metin bitine bağlı bir durum elemanı q^j ile gösterilir. q^j ise q^j 'nin i inci parçasıdır. Burada c , hiçbir bite bağlı olmayan, yani q durumunun $j=0$ indeksli parça olarak düşünülebilir. Sonuç olarak bir t anındaki q^j daha büyük j indeksli durum elemanlarından bağımsız, daha küçük j indeksli durum elemanlarına bağımlıdır. q durumu, 1 ile n_m arasındaki uzunluklarda belleklere sahip alt durumlara ayrılmıştır ve her durum parçası için durum güncelleme fonksiyonu, $j \leq n_m$ için denklem 2.2'deki halini alır.

$$q^{(j) t+1} = G [K]_i^{(j)}(c^t, q^{(1)t}, \dots, q^{(j-1)t}) \quad (4.2)$$

4.1.2. Koşullu tamamlayan ötelemeli kaydedici (CCSR)

Koşullu tamamlayan ötelemeli kaydedici, sonlu iç bellekli sonlu durum makineleri için geliştirilmiş bir çözümdür. Bu bellek ile durum güncelleme fonksiyonu denklem 2.3'teki halini alır.

$$q^{(j) t+1} = q^{(j) t+1} + E [K]^{(i)} (q^{(j-2)t}, \dots, q^{(1)t}, c^t) \quad (4.3)$$

Burada $t+1$ anındaki q^j , q^{j-1} in eski değerinin ve bir bole fonksiyonunun toplamına eşit olduğundan belleğe koşullu tamamlayan ötelemeli kaydedici ismi verilmiştir.

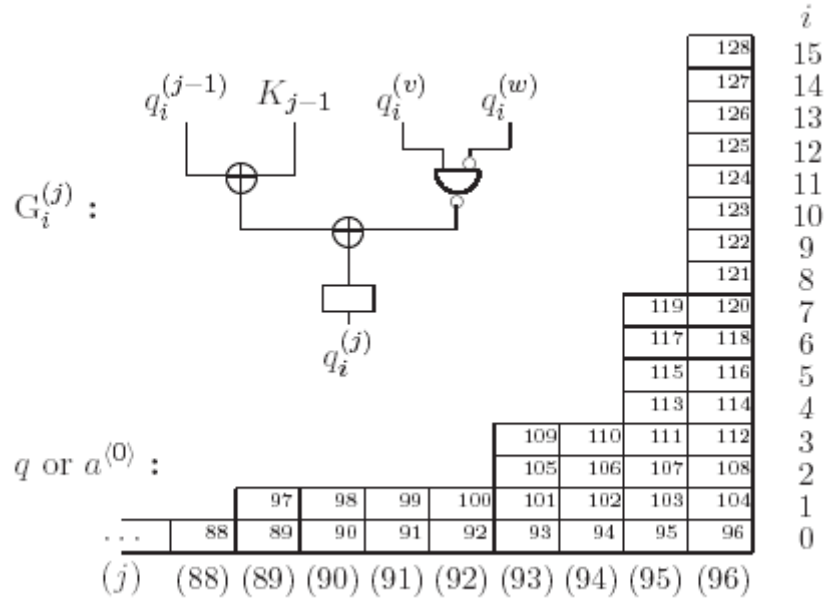
4.1.3. Mosquito CCSR

Mosquito, anahtar uzunluğu 96, bellek uzunluğu 105 ve fonksiyon gecikmesi 9 olan bir kendiyile senkronize dizi şifreleme fonksiyonudur. Mosquito algoritmasında IV 105 bit uzunluğunda olup, üretilecek olan şifre z 'nin genel fonksiyonu denklem 2.4'teki gibidir.

$$z^t = f_c [K] (c^{t-105} \dots c^{t-10}) \quad (4.4)$$

Mosquito CCSR'de 96 iç bellek olup, tüm CCSR'nin içeriği 128 bite genişletilmiştir. Bu genişleme, her iç belleğin $j = 89$ 'dan itibaren 2, $j = 93$ 'ten itibaren 4, $j = 95$ 'te 8 ve son olarak $j = 96$ 'da 16 bitlik bir belleğe genişletilmesiyle elde edilir. Bu durum şekil 4.1'de gösterilmektedir.

Basitlik, alan ve kapı gecikmeleri açısından verimli bir devre olması açısından durum güncelleme fonksiyonu basit bir bole fonksiyonu olarak düşünülmüştür ve şekil 4.1'de kombinezonalsal devre karşılığı görülen ve 4.4'te verilen fonksiyondur.



Şekil 4.1: Mosquito CCSR'in genişleme yapısı ve durum güncelleme fonksiyonu.

$$G[K]_i^{(j)}(q,c) = q_i^{(j-1)} + K_{j-1} + (q_i^{(v)})(q_i^{(w)} + 1) + 1 \quad (4.4)$$

Denklem 4.4'teki v ve w değerleri j-1'den küçük değerlerdir. Bu v ve w'nın alacağı değerler i ve j değerlerine bağlı olup tablo 4.1'e göre hesaplanmaktadır. $j \leq 4$ için v ve w değerleri 0 alınır. Tablo 1'den hesaplanan q_i^j 'nin mevcut bitlere karşılık gelmediği durumlarda ise var olan bir karşılık bulunana kadar i indeksi 2'nin en büyük kuvvetleri kadar azaltılır. Örneğin, q_{14}^{93} şekil 4.1'den de görüldüğü gibi

Tablo 4.1: $4 < j < 96$ indeksli G_i^j ve G_0^{96} için v ve w değerleri

	v	w
$(i + j) \bmod 3 = 0$	$j - 4 + (i \bmod 2)$	$j - 2$
$(i + j) \bmod 3 = 1$	$j - 6 + (i \bmod 2)$	$j - 2$
$(i + j) \bmod 6 = 2$	$j - 5 + (i \bmod 2)$	0
$(i + j) \bmod 6 = 5$	0	$j - 2$

bulunmamaktadır ve dolayısıyla i indeksi önce 14 ten küçük 2'nin en büyük kuvveti olan 8 azaltılır ve daha sonra da 6 ten küçük 2'nin en büyük kuvveti olan 4 azaltılarak var olan bir hücre olan q_2^{93} 'e ulaşılır ve durum güncelleme fonksiyonunda o kullanılır.

$i > 0$ için G_i^{96} olan 15 bit ise farklı bir denklem ile hesaplanır ve bu denklem 4.5'te verilmiştir.

$$G[K]_i^{96}(q,c) = q_i^{95} (q_0^{(95-i)} + 1) + q_i^{94} (q_i^{(94-i)} + 1) \quad (4.5)$$

4.1.4. Mosquito ardışık düzen aşamaları

Mosquito ardışık düzeninde yedi aşama vardır. İlk beş aşama 53'er bitlik belleklerden oluşmaktadır. Birinci aşamada CCSR' nin 128 biti kullanılarak 53'bitlik bir bellek doldurulur. Bu işlemde 4.6'daki fonksiyon kullanılır.

$$G_{4i \bmod 53}^{(1)} = \alpha_{128-i}^{(0)} + \alpha_{i+18}^{(0)} + \alpha_{113-i}^{(0)} (\alpha_{i+1}^{(0)} + 1) + 1 \quad (4.6)$$

2 ila 5'inci aşamalardaki 53'er bitlik bellek için fonksiyonlar ise 4.7'de verilmiştir.

$$G_{4i \bmod 53}^{(j)} = \alpha_i^{(j)} + \alpha_{i+3}^{(j)} + \alpha_{i+1}^{(j)} (\alpha_{i+2}^{(j)} + 1) + 1 \quad (4.7)$$

Her iki fonksiyon için de sağ taraftaki terimlerden herhangi birinin indeksinin 52'den büyük olduğu durumda bu terim 0 alınır. Altıncı aşamadaki bellek ise 12 bit uzunluğundadır ve güncelleme fonksiyonu denklem 4.8'deki gibidir.

$$G_i^{(6)} = \alpha_{4i}^{(5)} + \alpha_{4i+3}^{(5)} + \alpha_{4i+1}^{(5)} (\alpha_{4i+2}^{(5)} + 1) + 1 \quad (4.8)$$

Son aşama olan 7'inci aşamada ise 3 bitlik bir bellek vardır ve fonksiyonu 4.9'da verilmiştir

$$G_i^{(7)} = \alpha_{4i}^{(6)} + \alpha_{4i+1}^{(6)} + \alpha_{4i+2}^{(6)} + \alpha_{4i+3}^{(6)} \quad (4.9)$$

Algoritma sonucu olan şifre biti z ise bu son aşamadaki 3 bitlik belleğin bitlerine XOR işlemi uygulanması ile elde edilir (4.10).

$$z = a_0^7 + a_1^7 + a_1^7 \quad (4.10)$$

4.2. Gerçekleme Adımları

4.3. Gerçekleme Sonuçları