MÜZİKLE ŞİFRELEME-VERİ GİZLEME SİSTEMİ TASARIMI VE GERÇEKLENMESİ

MUHAMMET HAMDİ YAVUZ

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

EYLÜL 2010

ANKARA

Fen Bilimleri Enstitü onayı

_____

Prof. Dr. Ünver KAYNAK

Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Prof. Dr. Erdoğan DOĞDU

Anabilim Dalı Başkanı

Muhammet Hamdi YAVUZ tarafından hazırlanan MÜZİKLE ŞİFRELEME-VERİ GİZLEME SİSTEMİ TASARIMI VE GERÇEKLENMESİ adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Yrd. Doç. Dr. Oğuz ERGİN

Tez Danışmanı

Tez Jüri Üyeleri

Başkan : Doç. Dr. Kemal BIÇAKÇI _____

Üye : Yrd. Doç. Dr. Oğuz ERGİN _____

Üye : Yrd. Doç. Dr. H. Taha SENCAR _____

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Muhammet Hamdi YAVUZ

Üniversitesi : **TOBB Ekonomi ve Teknoloji Üniversitesi**
Enstitüsü : **Fen Bilimleri**
Anabilim Dalı : **Bilgisayar Mühendisliği**
Tez Danışmanı : **Yrd. Doç. Dr. Oğuz ERGİN**
Tez Türü ve Tarihi : **Yüksek Lisans – Eylül 2010**

Muhammet Hamdi YAVUZ

MÜZİKLE ŞİFRELEME-VERİ GİZLEME SİSTEMİ TASARIMI VE GERÇEKLENMESİ

ÖZET

Gizli kalması gereken verilerin hızlı bir şekilde verinin kaynağından hedefe ulaşması için günümüzde internet gibi herkese açık ve güvensiz kanallar kullanılmaktadır. Bu verilerin güvenliğini sağlamak kimi zaman hayati önem taşıdığından farklı şifreleme yöntemleri dünyanın dört bir yanında geliştirilmekte ve kullanılmaktadır. Bu duruma bir başka yaklaşım ise; veri güvenliğini veriyi gizleyerek sağlayan steganografik yöntemlerdir. Bu çalışmada, güvenlik seviyesini arttırmak için şifreleme ve veri gizleme bir arada kullanılmaktadır. Şifreleme için öncelikle AES-256 algoritması kullanılmakta ve algoritmanın çıktısı olan veriler, gizlenirken kullanılacak taşıyıcı ses dosyasındaki müzik eserinin notalarının kullanılan özelliklerinden elde edilen değerlerle sonlu alanda çarpma yapmak suretiyle şifrelenmektedir. Daha sonra, yine bu nota değerlerinden oluşturulan kurala göre, şifrelenmiş veriler, taşıyıcı ses dosyası içerisine gizlenerek ortaya şüphe çekmeyen ve şifrelenmiş veri taşıyan ses dosyaları çıkarılmaktadır. Taşıyıcı ses dosyasındaki müzik eserinden elde edilen değerler, hem şifreleme aşamasında hem de veri gizleme aşamasında anahtar niteliğinde kullanıldığından, kaynaktan hedefe yapılacak dosya aktarımından önce herhangi bir veri aktarımına ve anahtar paylaşımına ihtiyaç duyulmamakta ve güvensiz kanaldan anahtar paylaşımı yapılması sorununu ortadan kaldırmaktadır. Tasarlanan sistem, girdiler ne olursa olsun oluşacak çıktıdan aynı nota değerlerinin elde edilebileceğini ve yapılan işlemlerin matematiksel olarak tersinin yapıldığı veri ayıklama ve çözme aşamalarının ardından doğru veriye kesin olarak ulaşılacağını garantilemektedir. Tasarlanan sistemin gerçeklenmesi birbirleriyle haberleşen yazılım ve donanım birimleri tasarlanarak yapılmıştır. Hız kısıtları göz önünde bulundurularak sistemin yazılım ve donanımda çalışacak alt sistemleri belirlenmiş ve donanımda 2524 Mbps gibi bir şifreleme hızına ulaşılmıştır.

Anahtar Kelimeler: Kriptografi, Steganografi, Müzik, FPGA, Ses İşeme, AES.

University : TOBB University of Economics and Technology
Institute : Institute of Natural and Applied Sciences
Science Programme : Computer Engineering
Supervisor : Associate Professor Dr. Oğuz ERGİN
Degree Awarded and Date : M.Sc. – September 2010

Muhammet Hamdi YAVUZ

CRYPTOGRAPHY-STEGANOGRAPHY SYSTEM DESIGN AND IMPLEMENTATION USING MUSIC

ABSTRACT

Nowadays, internet like public and unsecure channels are being used to transfer data, which must stay hidden, from source to destination rapidly. For providing the security of these data, which are sometimes vital, different cryptographic techniques are being developed and used all over the world. Another approach to this situation is steganography which provides security of the data by hiding it. In this work, cryptography and steganography are being used together to raise the security level. At first, AES-256 algorithm is used for encryption and then the output of the algorithm is being encrypted by multiplying it in finite field by the values which are obtained from the interested features of musical notes in the carrier audio file which will be used in data hiding process. After that, encrypted data is embedded in the carrier audio file by obeying the rules generated from the musical notes and as a result of that, there appears an unsuspicious audio file which includes encrypted data. Because of the values obtained from the music in the carrier audio file are used both in cryptography process and steganography process as key, there is no need to transfer data and share keys before the file transfer and thanks to that the key sharing through an unsecure channel problem is disappeared. This system ensures that, values obtained from the output file remain the same as obtained from the input file and after mathematical inverse of encryption and hiding processes the result data is definitely the same as the input data of the encryption process. Implementation of the system designed is being done with designing software and hardware units that communicates each other. Considering speed constraints, implemented subsystems are selected whether they will work on hardware unit or software unit and 2524 Mbps encryption throughput is observed in the hardware unit.

Keywords: Cryptography, Steganography, Music, FPGA, Audio Processing, AES.

TEŞEKKÜR

Kasırga Mikroişlemciler Laboratuvarı'nda bulunduğum süre boyunca ve tez çalışmalarımda bana yol gösterip destek veren değerli hocam, tez danışmanım Yrd. Doç. Dr. Oğuz ERGİN'e teşekkürlerimi sunarım.

Hayatımın her anında bana destek olan, beni seven, hayatını benimle paylaşan, her şey onun varlığında olan, bana hayat veren müstakbel eşim, sevgili nişanlım Rabia Burçin SARICA'ya sonsuz teşekkürlerimi sunarım.

Ayrıca, her an desteklerini yanımda hissettiğim, var oluşlarıyla bana güç veren, bana her zaman gerekli değeri, ilgiyi gösteren ailem Recai Nurhan YAVUZ, Ayşe Yavuz, Canan YAKIN, Mustafa YAVUZ, Meryem YAVUZ, Ferit YAKIN, Efehan YAKIN, Dorukhan Yakın, Ömer SARICA, Raziye SARICA ve Osman SARICA'ya sonsuz teşekkürlerimi sunarım.

Birlikte büyüdüğüm, birlikte çalışmalar yaptığımız, hayatı birlikte öğrendiğimiz, hep birbirimize destek olduğumuz kardeşlerim Yılmaz Fırat KAYA, Metin Sabri KAZKAYASI, bütün bunların yanı sıra tez çalışmalarım için bana uygun ortam sağlayan Onur TOSUN ve yukarıda yazdığım şeylerin yanı sıra donanım dünyasına adım atmamı sağlayan Özcan YURT'a sonsuz teşekkürlerimi sunarım.

Son olarak; üzerinde çalışmalarımızı sürdürebildiğimiz bu kutsal vatanı bize bağışlayan, bu vatanın kurulmasında ve gururla yaşatılmasında emeği geçmiş herkese, biz çalışmalarımızı yürütürken her türlü hava şartında, yaz, kış, yağmur, çamur demeden vatanımızı ve bizi koruyan şanlı Mehmetçik'e sonsuz teşekkürlerimi sunarım.

İÇİNDEKİLER

ÖZET iv
ABSTRACT v
TEŞEKKÜR vi
İÇİNDEKİLER vii
ÇİZELGELERİN LİSTESİ ix
ŞEKİLLERİN LİSTESİ x
KISALTMALAR xii
1. GİRİŞ 1
2. SAHADA PROGRAMLANABİLİR KAPI DİZİLERİ 3
2.1. Taramalı Çizelge Tabanlı Mantıksal Hücre 5
2.2. Makro Hücre 5
2.3. FPGA ile Tasarım Süreci 6
3. ŞİFRELEME BİLİMİ 8
3.1. Şifre Sistemleri 8
3.1.2. Açık Anahtarlı (Bakışımsız) Sistemler 10
3.1.3. Gizli Anahtarlı (Bakışımlı) Sistemler 11
3.1.4. Özetleme Fonksiyonları 23
4. AES ALGORİTMASI 30
4.1. Anahtar Genişletme 31
4.2. Dönüşüm İşlemleri 33
4.2.1. Bayt Değiştirme Dönüşümü 34
4.2.2. Satır Kaydırma Dönüşümü 36
4.2.3. Sütun Karıştırma Dönüşümü 36
4.2.4. Devir Anahtarı Ekleme Dönüşümü 37
4.3. AES Şifreleme Algoritmasının Akışı 38
5. MATEMATİKSEL ÖN BİLGİ 41
5.1. Sonlu Alan Teorisi 41
5.1.1. Değişmeli Grup 41
5.1.2. Halka 42
5.1.3. Alan 42

5.1.4. Sonlu Alan 43
5.2. Galois Alanı 43
5.2.1. GF(p) 43
5.2.2. GF(p^m) – Genişletilmiş Alan 43
5.2.3. GF(2^m) 44
5.2.4. Genişletilmiş Alanda Elemanların Gösterimi 44
5.2.5. Galois Alanında Çarpma İşlemi 46
6. HIZLI FOURIER DÖNÜŞÜMÜ 47
7. MÜZİKAL ÖN BİLGİ 51
7.1. Perde, Frekans, Nota ve Oktav Kavramları 51
7.2. Uzunluk, Süre, Tempo ve Ölçü Kavramları 52
8. VERİ GİZLEME 55
9. TASARIM ve GERÇEKLEME 59
9.1. AES-256 Algoritmasının Gerçeklenmesi 61
9.1.1. Gerçekleştirmenin Yapıldığı FPGA Geliştirme Kartı 61
9.1.2. Anahtar Genişletme Aşamasının Gerçeklenmesi 62
9.1.3. Bayt Değiştirme Dönüşümünün Gerçeklenmesi 63
9.1.4. Satır Kaydırma Dönüşümünün Gerçeklenmesi 65
9.1.5. Sütun Karıştırma Dönüşümünün Gerçeklenmesi 65
9.1.6. Çözme Algoritmasının Gerçeklenmesi 67
9.1.7. Sistemin Birleştirilmesi ve Kullanımı 67
9.1.8. Değerlendirme ve Karşılaştırma 70
9.2. AES Sonrası Şifreleme 72
9.3. Taşıyıcı Ses Dosyasından Özniteliklerin Elde Edilmesi 73
9.4. Veri Gizleme 76
10. SONUÇ 82
KAYNAKLAR 85
ÖZGEÇMİŞ 87

ÇİZELGELERİN LİSTESİ

Çizelge 3.1 SHA Algoritmalarının Özellikleri 24
Çizelge 3.2 SHA-256 Sabitleri 26
Çizelge 3.3 SHA-256 Başlangıç Özet Değerleri 27
Çizelge 4.2 AES algoritması için anahtar-kütük-devir bileşimleri 31
Çizelge 4.3 AES-256 için Örnek Anahtar Genişletme 33
Çizelge 4.4 AES Algoritması S-kutusu 35
Çizelge 9.1 AES-256 Gerçeklemesinin Farklı FPGAlar Üzerindeki Değerleri 70
Çizelge 9.2 AES Gerçeklemesinin Diğer Gerçeklemelerle Karşılaştırılması 71
Çizelge 9.3 AES Gerçeklemesinin Diğer Çalışmalarla Alan ve Veri Akışı Yönünden Karşılaştırılması 71
Çizelge 9.4 Notalara Karşılık Gelen Değerler 75

ŞEKİLLERİN LİSTESİ

Şekil 2.1 Bir FPGA Aygıtının Kavramsal Yapısı 4
Şekil 2.2 Programlanabilir Ara Bağlantılar 4
Şekil 2.3 Üç Girişli Taramalı Çizelge Tabanlı Mantıksal Hücre 5
Şekil 2.4 FPGA ile Tasarım Süreci Akış Şeması 7
Şekil 3.1 Şifre Sistemi Genel Yapısı 8
Şekil 3.2 Açık Anahtarlı(Bakışımsız) Sistemler 10
Şekil 3.3 Kütük Şifre Sistemi 13
Şekil 3.4 Elektronik Kod Kitabı Kipi Gösterimi 14
Şekil 3.5 ECB Kipi Kullanılarak Şifrelenen Bir Resim 15
Şekil 3.6 Şifrelenmiş Kütük Zincirleme Kipi Gösterimi 16
Şekil 3.7 Şifrelenmiş Veri Geri Besleme Kipi Gösterimi 18
Şekil 3.8 Çıktı Geri Besleme Kipi Gösterimi 20
Şekil 3.9 Dizi Şifre Sistemleri Genel Gösterimi 21
Şekil 4.1 AES Algoritması durum dizisi, girdi ve çıktılar. 30
Şekil 4.2 AES Algoritması Bayt Dönüşümünün Durum Matrisine Uygulanması 35
Şekil 4.3 AES Algoritması Satır Kaydırma Dönüşümü 36
Şekil 4.4 Sütun Karıştırma Dönüşümü 37
Şekil 4.5 Devir Anahtarı Ekleme Dönüşümü 38
Şekil 4.6 AES Şifreleme Algoritması Akış Şeması 39
Şekil 6.1 Ayrık Fourier Dönüşümü ve Hızlı Fourier Dönüşümü için Gereken Çarpma Sayılarının Karşılaştırılması 48
Şekil 6.2 8 Nokta AFD'nin 2 adet 4 Nokta AFD'ye İndirgenmesi 50
Şekil 7.1 Notaların Müzik Yazımındaki Gösterimleri 51
Şekil 7.2 Notaların Uzunluk Değerleri ve Gösterimleri 53
Şekil 7.3 Müzikte Ölçü, Vuruş, Uzunluk ve Süre Kavramları 54
Şekil 8.1 Veri Gizleme Sistemi 55
Şekil 8.2 Sabit Örnek Atlamalı Sese Veri Gizleme Sistemi 56
Şekil 8.3 Rastgele Örnek Atlamalı Sese Veri Gizleme Sistemi 57
Şekil 9.1 Müzikle Şifreleme Sistemi Üst Seviye Görünümü 60
Şekil 9.2 Altera Bemicro FPGA Geliştirme Kartı 61
Şekil 9.3 Kelime Değiştirme Dönüşümünün Gerçeklenmesi 63
Şekil 9.4 Bayt Değiştirme Dönüşümünün Gerçeklenmesi 64
Şekil 9.5 Sütun Karıştırma Devresinden Bir Kesit 66
Şekil 9.6 AES-256 Gerçeklemesinin Durum Makinesi 68
Şekil 9.7 Baskın Notaların Bulunması 74
Şekil 9.8 Veri Gizleme Sistemi 77
Şekil 9.9 Farklı Oktavlar İçin Ortalama Örnek Atlama Değerleri 78
Şekil 9.10 Taşıyıcı Dosyanın Veri Gizlenmeden Önceki ve Sonraki İzge Çözümlemesi 79

Şekil 9.11 Ses Kalitesi Ölçümü 80
Şekil 9.12 Şifreleme-Veri Gizleme Akış Şeması 81
Şekil 9.13 Veri Ayıklama-Çözme Akış Şeması 81
Şekil 10.1 Yazılım Ekran Görüntüsü 82
Şekil 10.2 Taşıyıcı Dosyanın Taşıyabileceği Veri Miktarları 84

KISALTMALAR

Kısaltmalar	Açıklama
AES	Şifreleme Standardı (Advanced Encryption Standard)
AFD	Ayrık Fourier Dönüşümü
CBC	Şifrelenmiş Kütük Zincirleme (Cipher Block Chaining)
CFB	Şifrelenmiş Veri Geri Besleme (Cipher Feedback)
DES	Veri Şifreleme Standardı (Data Encryption Standard)
DFF	Veri Kapanı (Data Flip Flop)
ECB	Elektronik Kod Kitabı (Electronic Codebook)
FIPS	Federal Bilgi İşleme Standardı (Federal Information Processing Standard)
FPGA	Sahada Programlanabilir Kapı Dizileri (Field Programmable Gate Array)
GF	Galois Alanı (Galois Field)
HFD	Hızlı Fourier Dönüşümü
LUT	Taramalı Çizelge (Look-up Table)
NSA	Ulusal Güvenlik Kurumu – ABD (National Security Agency)
OFB	Çıktı Geri Besleme (Output Feedback)
ROM	Yalnızca Okuma Belleği (Read Only Memory)
RTL	Direnç Tranzistör Mantığı (Resistor Transistor Logic)
SHA	Güvenli Özetleme Algoritması (Secure Hash Algorithm)

1. GİRİŞ

Gelişen teknolojiyle birlikte günümüz dünyasında, gizli kalması gereken bilgilerin dışarıdan erişilebilir sistemler vasıtasıyla, bilginin kaynağından hedefe aktarılması, bilginin sahipleri için şüphesiz ciddi sorunlar oluşturmakta ve bilginin gizli kalmasının etkileyeceği kitleler için çok önemli bir tehdit unsuru olmaktadır.

Çağımızın ihtiyaçlarından bir tanesi olan hız, bu alanda da önemli bir unsur olarak yerini almaktadır. Gizli kalması gereken verilerin çok kısa sürede yerine ulaştırılması kimi zaman hayati önem taşımaktadır. Bu sebeple, hızla gelişen teknolojinin en önemli ve etkileşimli ürünlerinden biri olan internet, bilindiği gibi tüm dünyada yaygın olarak kullanılmaktadır. Gizli verilerin iletildikleri internet gibi kanallara erişimin herkese açık olması, bu verilerin aktarılırken güvenliğinin sağlanması ihtiyacını beraberinde getirmektedir. Bahsedilen tehlikeye karşı birçok şifreleme algoritması geliştirilmiş ve hala bütün dünyada yaygın olarak kullanılmaktadır. Veriler şifrelendiklerinde yine veri sıfatındadırlar, yalnızca başka bir türe dönüşmüş olarak nitelendirilebilirler. Şifreli veri aktarımını izleyen bir kişi çoğunlukla şifrelenmiş bir verinin kaynaktan hedefe iletildiğini anlamakta ve şifreyi çözmek için çeşitli yollara başvurmaktadır. Bu durumu ortadan kaldırmak için ise veri gizleme bir diğer adıyla steganografi üzerinde de tüm dünyada çalışmalar yapılmaktadır. Veri gizlemede esas, aktarımı izleyen bir kişinin aktarımda göreceği masum dosyanın içerisinde gizli bir veri olduğunu anlayamamasıdır. Veri gizleme için en önemli güvenlik katmanı, veri gizlenen dosya içerisinde gömülü veri olduğunun ispat edilememesinin sağlanmasıdır.

Bu çalışmada güvenlik seviyesinin arttırılması amacıyla şifreleme ve veri gizleme birlikte kullanılarak bir sistem tasarlanmış ve birbirleriyle haberleşen yazılım ve donanım birimleri ile gerçekleştirilmiştir. Şifreleme amacıyla öncelikle AES-256 algoritması kullanılmış daha sonra algoritmanın çıktısı ile sonlu alanda bir çarpma

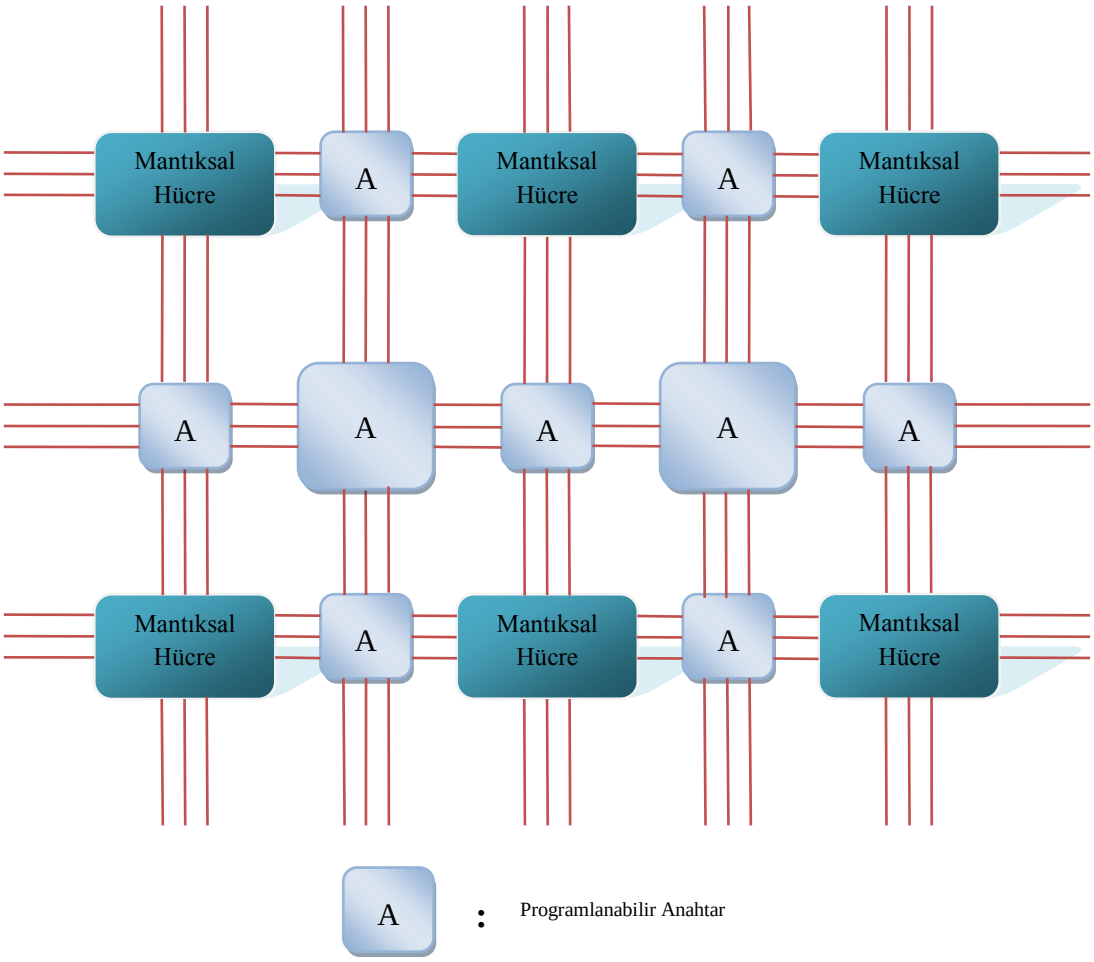
işlemi yapılarak ek bir şifreleme sistemi tasarlanmıştır. Daha sonra yeni bir yöntem ile veri gizleme yapılarak ortaya şüphe çekmeyen ve gizli veri taşıyan ses dosyaları çıkarılmıştır. Çalışmanın çıkış noktası şu cümlelerle özetlenebilir: "Mademki bir müzik eseri içeren ses dosyasına veri gizleniyor, neden bu veri ses dosyasındaki müzik eserinin notalarıyla şifrelenip yine bu notalara göre bir kural oluşturulup ses dosyası içerisine gömülmesin?". Böylelikle hem şifreleme için hem de veri gizleme için gönderici ile alıcı arasında fazladan bir veri aktarımına ve anahtar anlaşmasına gerek kalmamaktadır. Her farklı müzik eseri için sonlu alan çarpımında anahtar niteliği taşıyan çarpanlar değişecek ve aynı zamanda veri gizleme ses dosyasının farklı noktalarına uygulanacaktır. Yani her girdi ses dosyası için aynı güvenlik seviyesinde farklı bir şifreleme-veri gizleme yolu izlenecektir.

Müzikte aslında her şey matematiksel olarak ifade edilebildiği için yukarıda anlatılan sistem hem yeterli müzik bilgisine sahip bir kişi tarafından el ile hem de sayısal ses işleme yapılarak gerçekleştirilebilmektedir. Yapılan veri gizlemenin ayıklanabilmesi ve şifrelemenin çözülebilmesi için bir kişi notaların hangi özelliklerinin kullanıldığını, notalardan kullanılan değerlerin nasıl elde edildiğini, bu değerlerle veri gizlemenin ve çarpma işleminin nasıl yapıldığını ve son olarak AES-256 algoritmasında kullanılan 256 bitlik anahtarın ne olduğunu bilmesi, çözümü el ile yapacaksa yeterli müzik bilgisine sahip olması, eğer el ile yapmayacaksa müzik değerlerinin çıkarımlarını sayısal ses işlemeyle yapan bir sisteme sahip olması gerekmektedir.

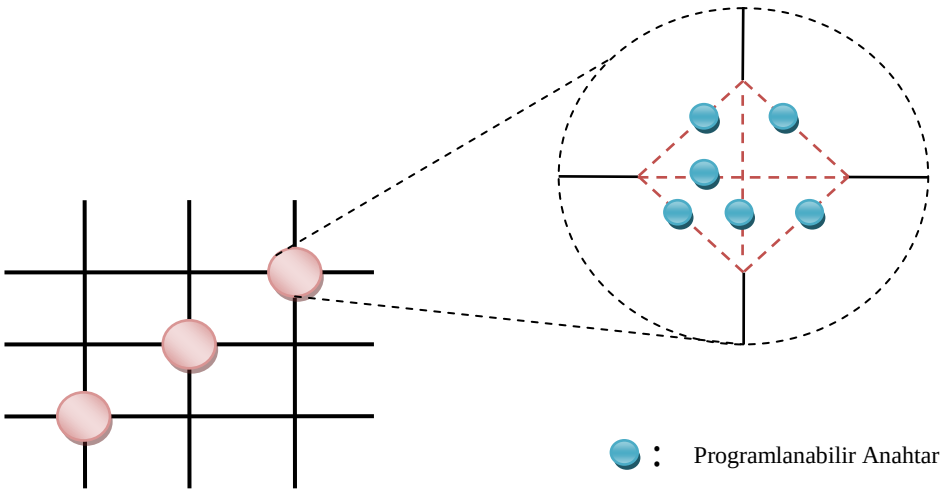
Tasarımın gerçeklenmesi birbirleriyle haberleşen bir yazılım birimiyle bir FPGA birimi yardımıyla yapılmıştır. FPGA üzerinde çalışan AES-256, hız kısıtlarıyla özgün olarak tasarlanmış ve yüksek hızlı veri şifreleme ve çözme sağlanmıştır. Burada kullanılacak veriler bilgisayardan alınıp çıktılar bilgisayara gönderildiğinden hızı kısıtlayan tek etmen iletişimdeki aktarım hızı olmaktadır.

2. SAHADA PROGRAMLANABİLİR KAPI DİZİLERİ

Sahada programlanabilir kapı dizileri, iki boyutlu üreysel mantık hücreleri dizisi ve programlanabilir anahtarlarları içeren mantıksal aygıtlardır. FPGAlarda mantıksal bir hücre, basit bir işlevi yerine getirmek için programlanabildiği gibi programlanabilir anahtarlar ise mantıksal hücreler arasındaki ara bağlantıları sağlayacak şekilde düzenlenebilmektedir. Şekil 2.1'de bir FPGA aygıtının kavramsal yapısı, şekil 2.2'de ise programlanabilir anahtarların oluşturduğu ara bağlantıların gösterimi sunulmaktadır. Özel bir tasarım, her mantıksal hücrenin işlevinin belirtilmesiyle ve her programlanabilir anahtarın bağlantılarının ayarlanmasıyla gerçeklenebilmektedir. FPGA aygıtları geliştirilen donanım tanımlama dilleri sayesinde üst seviye dillerle tasarlanan donanımlar tanımlanabilmektedir. Tasarım ve sentez aşaması tamamlandıktan sonra oluşan mantıksal hücre ve anahtar yapılandırması basit bir bağdaştırıcı kablo yardımıyla FPGA aygıtına gönderilir ve tasarlanan devre çalışmaya hazır hale gelir. Bu işlemler ister sahada ister fabrika çıkışında yapılabilmektedir. Günümüzde, FPGA aygıtlarının bir çoğunda mantıksal hücrelere ve programlanabilir anahtarlara ek olarak bellek birimleri de bulunmaktadır. Bu birimler bellek kütüklerinden veya ayrık kapan (flip-flop) yapılarından oluşabilmektedirler [1].



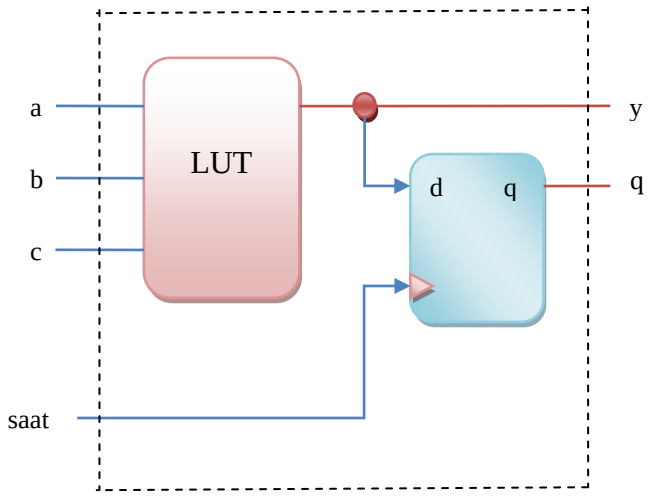
Şekil 2.1 Bir FPGA Aygıtının Kavramsal Yapısı



Şekil 2.2 Programlanabilir Ara Bağlantılar

2.1. Taramalı Çizelge Tabanlı Mantıksal Hücre

Bir mantıksal hücre genellikle D tipi kapanlı (DFF) küçük ve ayarlanabilir bir bileşimli devre içerir.Ayarlanabilir bir bileşimli devrenin gerçeklenmesinin en yaygın yöntemi taramalı çizelge (look-up table) yöntemidir. Bir n girişli LUT, 2^nx1'lik küçük bir bellek olarak düşünülebilir. Bir LUT, bellek içeriği uygun bir şekilde yazılarak , n adet girişli bir bileşimli işlevi gerçeklemek için kullanılabilmektedir. Şekil 2.3'te üç girişli taramalı çizelge tabanlı bir mantıksal hücre ve a ⊕ b ⊕ c işleminin üç girişli LUT gerçeklemesinin örnek çizelgesi sunulmaktadır. Burada LUT çıkışı doğrudan kullanılabildiği gibi bir DFF'ye de kaydedilebilmektedir.



a	b	c	y
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**
1	0	0	**1**
1	0	1	**0**
1	1	0	**0**
1	1	1	**1**

Şekil 2.3 Üç Girişli Taramalı Çizelge Tabanlı Mantıksal Hücre

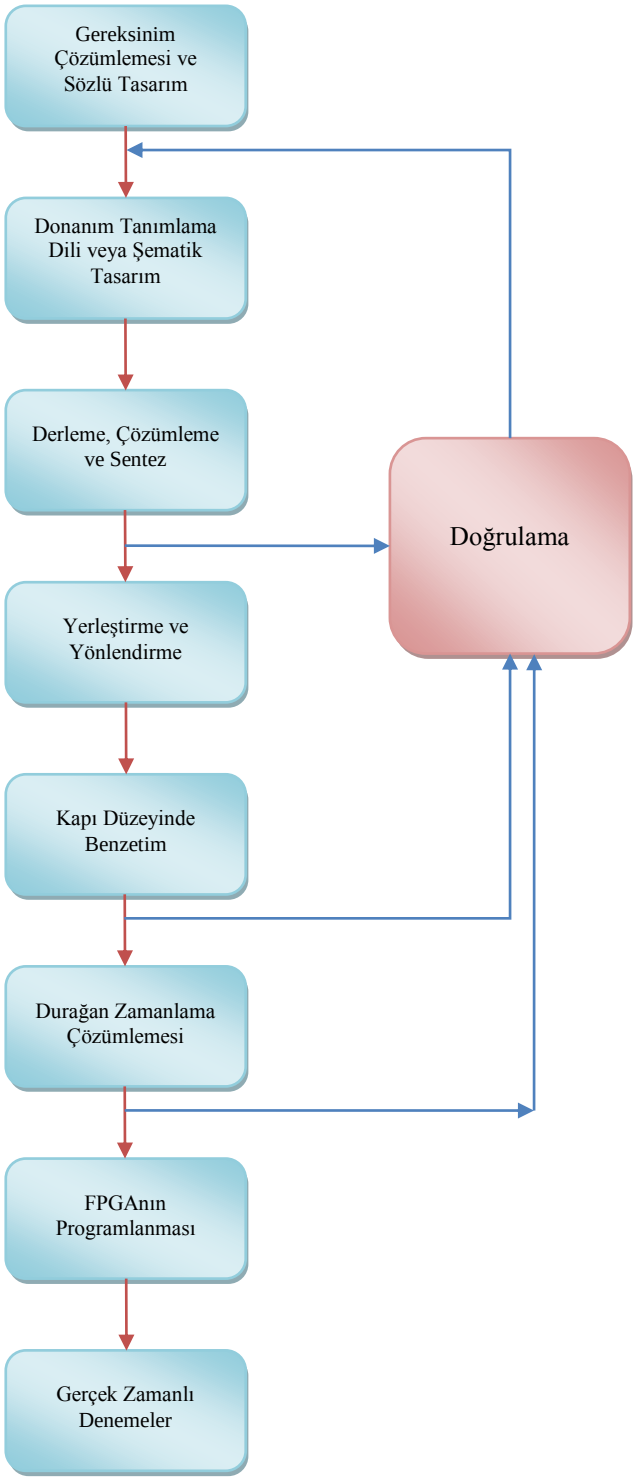
2.2. Makro Hücre

Bir çok FPGA aygıtında, aygıtın içine gömülü belli makro hücreler veya makro kütükler bulunmaktadır. Bu hücreler, transistör seviyesinde tasarlanmış ve üretilmiş olup işlevleri genel mantık hücrelerini tamamlayıcı niteliktedir. Yaygın olarak kullanılan makro hücreler arasında bellek kütükleri, bileşimli çarpıcılar, saat yönetim

devreleri ve giriş-çıkış arayüz devreleri bulunmaktadır. Gelişmiş FPGA aygıtları, bir veya daha çok önceden hazırlanmış işlemci çekirdeği içerebilmektedirler.

2.3. FPGA ile Tasarım Süreci

Bir tasarım FPGA ile gerçeklenmek istendiğinde, tasarımın çalışan bir devreye dönüşmesi için bazı aşamalardan geçmesi gerekmektedir. İlk olarak, gereksinim çözümlemesi yapılmalı ve sistem sözlü olarak tanımlanıp tasarlanmalıdır. Daha sonra, donanım tanımlama dilleri veya şematik tasarım amaçları yardımıyla sistemin daha üst seviyede tasarımı veya kodlanması yapılmalıdır. Bu aşamadan sonra işlevsel RTL benzetimi tamamlanır ve tasarımın sentezlenme aşamasına gelinir. Sentezin ardından seçilen FPGA için yerleştirme ve yönlendirme işlemi yapılır ve sonrasında kapı düzeyinde benzetim gerçekleştirilir. Eğer kısıtlar sağlanıyor ve bir sorun yok ise bu aşamaların ardından, durağan zamanlama çözümlemesi yapılarak ilk örnek hazırlanır. FPGA programlanarak gerçek zamanlı denemeler yapılır. Donanım tanımlama dilleriyle tanımlanmış tasarımın, tüm bu aşamalara geçmeden önce benzetim araçları ile denenmesi ve tasarımın iyileştirilmesi daha verimli tasarımlar ortaya koyacaktır. Şekil 2.4'te FPGA ile tasarım süreci akış şeması sunulmaktadır.



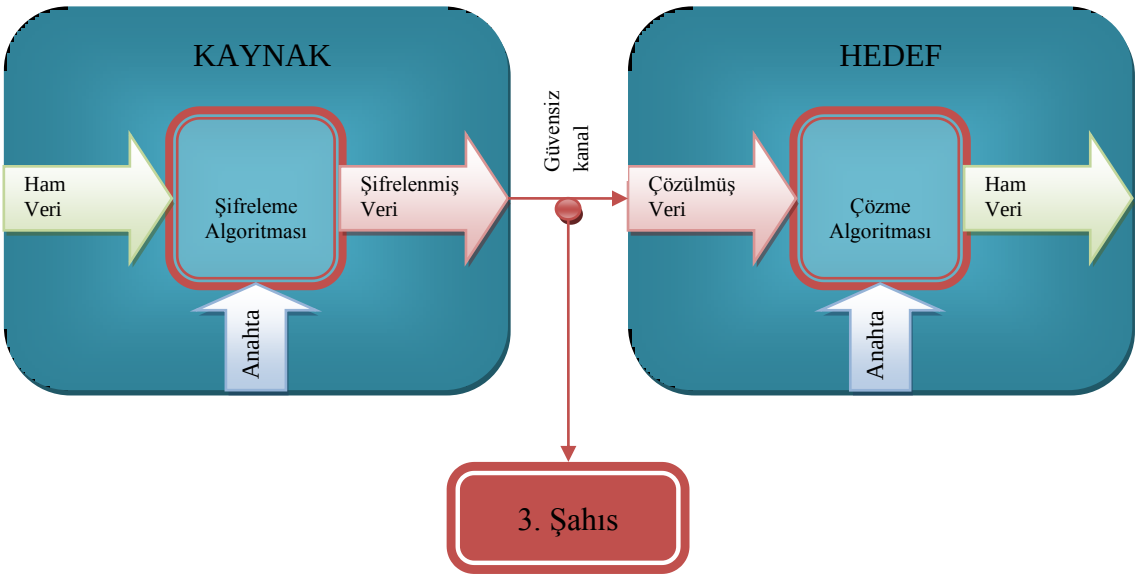
Şekil 2.4 FPGA ile Tasarım Süreci Akış Şeması

3. ŞİFRELEME BİLİMİ

Kriptografi olarak da anılan şifre bilimi; gizli olarak iletilmesi gereken herhangi bir bilginin kaynağından hedefe iletilirken 3. şahıslar tarafından çözümlenip anlaşılmamasını engellemeye çalışan, eski çağlardan beri gelişmekte olan bir bilim dalıdır. Günümüzde güvenliği sağlanması gereken bilgilerin güvensiz bir kanaldan(örneğin internet) iletilmesi gerekliliği oldukça yüksektir. Bu sebeple matematiksel şifre sistemleri üzerinde tüm dünyada çalışmalar yapılmakta, yeni sistemler geliştirilmekte ve var olan sistemlerin güvensiz yönleri ortaya çıkarılmaya çalışılmaktadır.

3.1. Şifre Sistemleri

Yukarıda bahsedildiği gibi güvensiz bir kanaldan gönderilecek bir verinin güvenliğinin sağlanması için çeşitli şifre sistemleri geliştirilmektedir. Bu sistemler temelde şekil 3.1'de gösterilen şekilde çalışmaktadırlar.



Şekil 3.1 Şifre Sistemi Genel Yapısı

Şekil 3.1'de görüldüğü üzere; kaynakta bulunan, güvenliği sağlanmak istenen ham veri, şifreleme anahtarını ve ham veriyi girdi olarak alan belli bir şifreleme algoritmasıyla şifrelenerek şifrelenmiş veri haline dönüştürülür. Şifrelenen veri, güvensiz kanal üzerinden hedefe gönderilir ve hedef çözme algoritmasına aldığı şifrelenmiş veriyle çözme anahtarını besleyerek ham veriye ulaşır. Bu tür sistemlerde kaynak ve hedef arasında şifreleme ve çözme algoritmalarının belirlenmiş ve anahtarların önceden paylaşılmış olması gerekmektedir. Şifre sistemlerinin ortaya çıkış sebeplerinden en önemlisi şekilde görülen 3. şahıstır. Güvensiz kanaldaki veri iletişimini dinleyerek kaynağın hedefe gönderdiği verinin ne olduğunu anlamaya çalışır. Algoritma ne kadar kuvvetli ise, 3. şahısın ham veriye ulaşma ihtimali o kadar düşük olacaktır. 3. şahıs aynı zamanda kriptoanalist olarak da nitelendirilebilir. Kriptoanalist, var olan şifre sistemlerini inceleyerek, sistemin açıklarını, eksiklerini bulan kişidir.

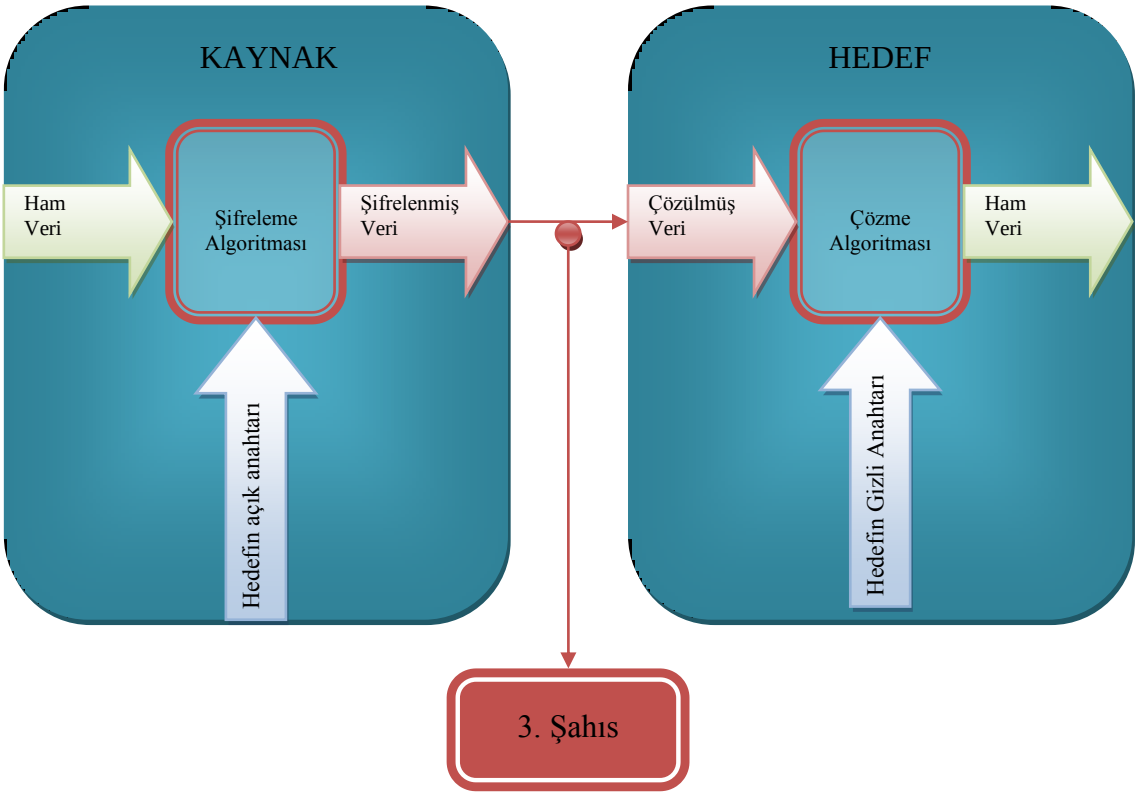
Şifre sistemlerini daha resmi ve genel bir şekilde ifade etmek gerekirse; bir şifre sistemi aşağıdaki koşulları sağlayan 5 bileşenden oluşur (H, K, A, Ş, Ç).

1. $H, sonlu\ sayıda\ olası\ ham\ verilerin\ kümesidir$
2. $K, sonlu\ sayıda\ olası\ şifrelenmiş\ verilerin\ kümesidir$
3. $A, sonlu\ sayıda\ olası\ anahtarların\ kümesidir$
4. $Her\ \mathrm{A} \in A\ için\ ş_a \in$
   $\mathrm{Ş}\ olmak\ üzere\ bir\ şifreleme\ kuralı\ ve\ buna\ ilişkin\ ç_a \in$
   $\mathrm{Ç}\ olmak\ üzere\ bir\ çözme\ kuralı\ vardır. Her\ ş_a : H \to K\ ve\ ç_a :$
   $K \to H\ fonksiyonları\ öyledirler\ ki\ her\ ham\ veri\ x \in$
   $H\ için\ ç_a(ş_a(x)) = x'tir.$

Şifre sistemleri genel olarak açık anahtarlı(bakışımsız) ve gizli anahtarlı(bakışımlı) sistemler olarak iki alt grupta incelenebilirler.

3.1.2. Açık Anahtarlı (Bakışımsız) Sistemler

Bu tür şifre sistemlerinde ham veriyi şifrelemek için farklı, şifrelenmiş veriyi çözmek için farklı anahtarlar kullanılmaktadır. Açık anahtarlı sistemlerde, birbirleriyle iletişime geçecek herkesin birer açık, birer de özel anahtarı vardır. Bu anahtarlar birbirlerine bağlı olarak üretilirler. Örneğin kaynak A, hedef B'ye veri göndermek isterse; hedef B'nin herkesle paylaştığı açık anahtarını kullanarak veriyi şifreler ve hedef B'ye gönderiri. Hedef B ise aldığı şifrelenmiş veriyi çözmek için kendi özel anahtarını kullanır ve veriye ulaşır. Bu tür sistemlerde açık anahtardan özel anahtar üretmek günümüz teknolojisiyle imkansıza yakın derecede zordur. Bu zorluk aslında açık anahtarlı sistemlerin temel güvenlik ihtiyacına cevap verir. Şekil 3.2'de bakışımsız sistemlerin temel yapısı sunulmaktadır.



Şekil 3.2 Açık Anahtarlı(Bakışımsız) Sistemler

Açık anahtarlı sistemlerin çözüm sunduğu en büyük sorun gizli anahtarlı sistemlerdeki anahtar paylaşma sorunudur. Gizli anahtarlı sistemlerde şifreleme ve çözme için tek bir anahtar kullanıldığından bu anahtarın hedef ve kaynak arasında güvenli bir kanal yardımıyla paylaşılması gerekmektedir. Açık anahtarlı sistemlerde ise paylaşılan yalnızca açık anahtar olduğundan paylaşım güvensiz kanallar yardımıyla sorunsuz bir şekilde yapılabilmektedir. Şifreli iletiyi çözmek için gerekli anahtar hedefin gizli anahtarı olduğundan ve bu anahtarın şifreleme esnasında bilinmesi gerekmediğinden anahtar paylaşımı esnasında güvenli bir kanal ihtiyacı ortadan kalkmış olur [2]. Bu yaklaşımı, gizli anahtarlı sistemlerdeki sıkıntıları ortadan kaldırmak için 1976 yılında Whitfield Diffie ve Martin E. Helman ortaya atmışlardır. Diffie ve Helman güvensiz bir kanal üzerinde güvenli bir anahtar paylaşımı yapılabilmesi amacıyla günümüzde de hala kullanılan Diffie-Helman protokolünü geliştirmişlerdir. Protokol ayrık logaritma probleminin çok büyük sayılar için çözümsüzlüğüne dayanmaktadır. Protokole göre şifreli iletişime geçecek iki kişinin öncesinde anahtar paylaşımı yapmasına gerek yoktur. Şifreli iletinin gönderiminden önce güvensiz bir kanalda birbirlerine kısıtlı bilgi gönderen kullanıcılar bu bilgiler ve ellerindeki göndermedikleri gizli bilgiler yardımıyla aynı anahtarı üretebilmektedir. Bu sayede kısıtlı veri alış verişi ile güvensiz kanalda güvenli bir paylaşım gerçekleştirilmiş olur [3].

3.1.3. Gizli Anahtarlı (Bakışımlı) Sistemler

Bakışımlı sistemlerde, şifreleme için de çözme için de gizli anahtar adı verilen yalnızca bir anahtar kullanılmaktadır. Sistemin tam güvenliğinin sağlanabilmesi için söz konusu gizli anahtar yalnızca hedef ve kaynak tarafından bilinmelidir. Güvensiz kanaldan yapılan veri aktarımının 3. Şahıslar tarafından izlenebilir olması sebebiyle algoritmanın yeterince güçlü ve anahtar genişliğinin yeterince büyük olması güvenliği sağlamada esastır.

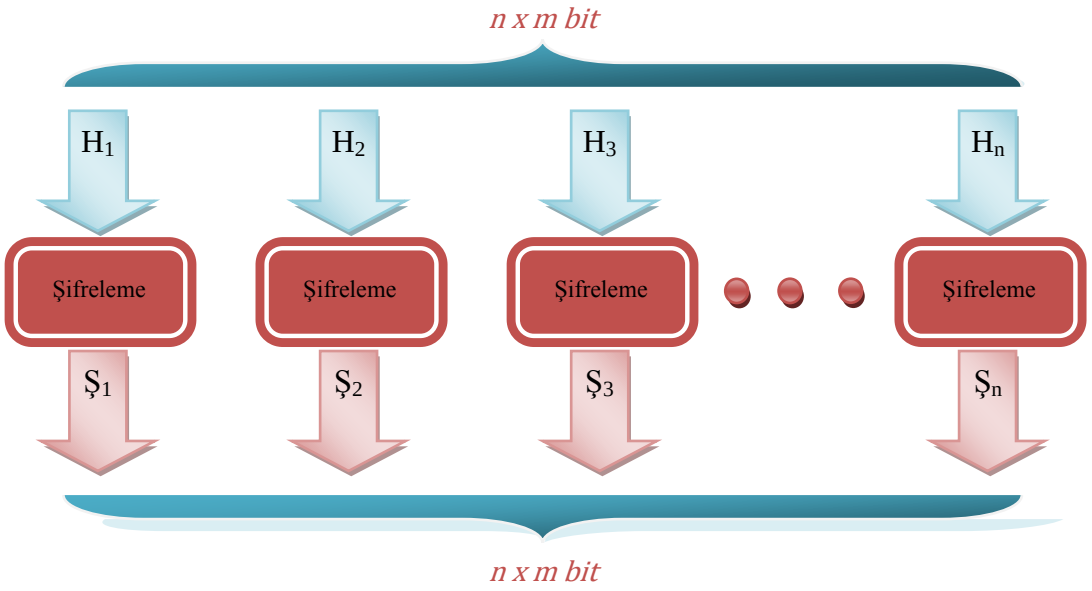
Gizli anahtarlı sistemler genel olarak beş bileşenle ifade edilebilmektedir. Bunlardan birincisi ham veridir. Ham veri açık metin (plain text) olarak da ifade edilebilir. Ham veri asıl paylaşılmak istenen gizli veriyi içerir ve şifreleme algoritmasının girdisidir. İkinci bileşen gizli anahtardır. Gizli anahtar hem şifreleme hem de çözme algoritmasının girdisi olarak kullanılır ve ham veriden tamamen bağımsızdır. Aynı ham veri için farklı gizli anahtarlar farklı çıktılar üretecektir. Üçüncü bileşen şifreleme algoritmasıdır. Gizli anahtarı da kullanarak ham veri üzerinde çeşitli dönüşüm işlemleri uygular ve dördüncü bileşen olan şifrelenmiş veriyi üretir. Şifrelenmiş veri kaynaktan hedefe güvensiz kanaldan gönderilecek olan veridir. Son bileşen ise çözme algoritmasıdır. Girdi olarak şifrelenmiş veriyi ve gizli anahtarı alarak algoritmanın içerdiği dönüşümleri uygular ve ham veriye ulaşır.

Gizli anahtarlı sistemlerin açık anahtarlı sistemlere kıyasla en büyük sorunu kaynak ve hedefin veri paylaşımından önce ortak bir anahtarda anlaşmasıdır. Şifreleme ve çözme için yalnızca tek bir anahtar kullanıldığından her iki taraf da bu anahtarı bilmek zorundadır. Güvensiz bir kanalda yapılacak anahtar paylaşımı gizli anahtarlı sistemlerin güvenliği için büyük bir tehdit oluşturmaktadır. Veri paylaşımını dinleyen 3. Şahıs eğer gizli anahtara sahip olur ise; algoritmayı bildiği takdirde gizli kalması gereken veriyi (ham veri) rahatlıkla öğrenebilecektir. Buna nazaran açık anahtarlı sistemlere karşı üstün yönleri ise çok daha hızlı çalışmalarıdır. Bakışımlı sistemler, kütük şifre sistemleri ve dizi şifre sistemleri olmak üzere iki alt sınıfta incelenebilirler.

3.1.3.1. Kütük Şifre Sistemleri

Kütük şifre sistemlerinde, girdi olarak alınan ham veri sabit genişlikli kütüklere ayrılarak algoritmanın gerektirdiği değişim ve dönüşüm işlemleri bu kütüklerin hepsine ayrı ayrı uygulanır ve sonuçta kütük sayısı kadar kütük genişliğinde şifrelenmiş veri ortaya çıkar. Şekil 3.3'te bir kütük şifre sistemi için kütük yapısı görülmektedir. Şekilde H ile ifade edilen her öğe şifreleme algoritmasının girdisi

olan ham verinin m bitinden oluşan kütüklerdir. Görüldüğü üzere; her kütüğe şifreleme işlemi ayrı ayrı uygulanır ve sonuçta Ş ile ifade edilen m bitlik n tane şifrelenmiş veri kütüğü elde edilir.

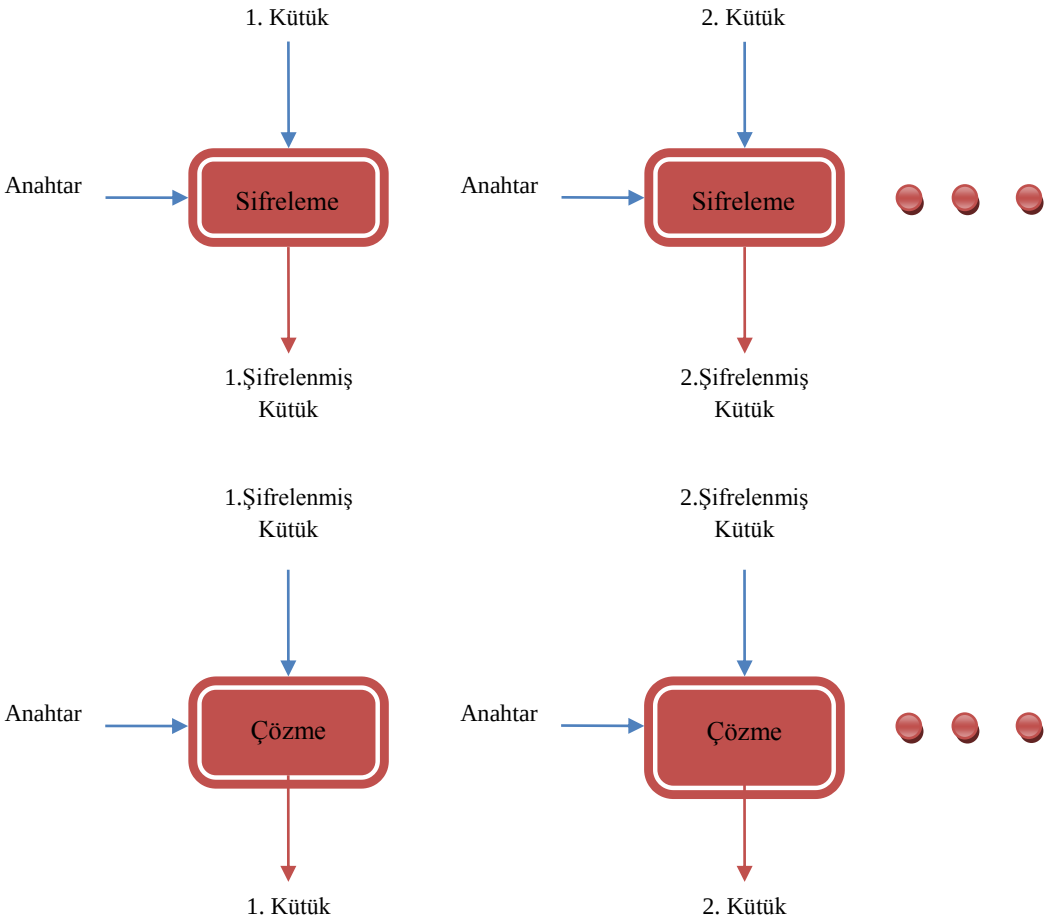


Şekil 3.3 Kütük Şifre Sistemi

Şifrelenecek verinin boyutu, türü, kütüklerin tekrar durumu gibi değişkenlerin farklı uygulamalarda kütük şifrele sistemlerinin güvenliğini azaltmaması için çeşitli kütük şifre çalışma kipleri tasarlanmıştır. Bu kiplerden ECB, CBC, CFB ve OFB'nin tanımları aşağıdaki gibidir [4].

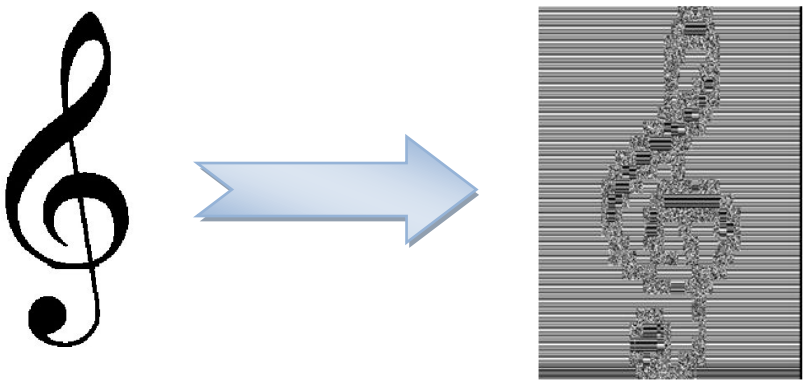
Elektronik Kod Kitabı Kipi :

ECB kipi, kütük şifre çalışma kipleri arasında en basit yapıda olan kiptir. Bu kipte, şifrelemeye her seferinde bir kütük tabi tutulur ve bu işlem hiçbir değişikliğe uğratılmadan sonraki kütükten devam eder. Şekil 3.4'te ECB kipinin şematik gösterimi sunulmaktadır.



Şekil 3.4 Elektronik Kod Kitabı Kipi Gösterimi

Şifrelenecek verinin yapısına göre bu kip çok ciddi sorunlar ortaya çıkarabilmektedir. Özellikle ham verinin tekrarlı yapılar oluşturduğu veya bir örüntüyü takip ettiği verilerde bu kipin kullanılmaması gerekmektedir. Bu kip kullanılırken yapılan şifrelemede bütün aynı kütüklerin şifrelenmiş kütükleri aynı olacağından bazı durumlarda ham verideki örüntü bozulmamış olacaktır. Buna örnek olarak bir resim girdisinin ECB kipi kullanılarak yapılan şifreleme sonucundaki görüntüsü şekil 3.5'te sunulmaktadır.



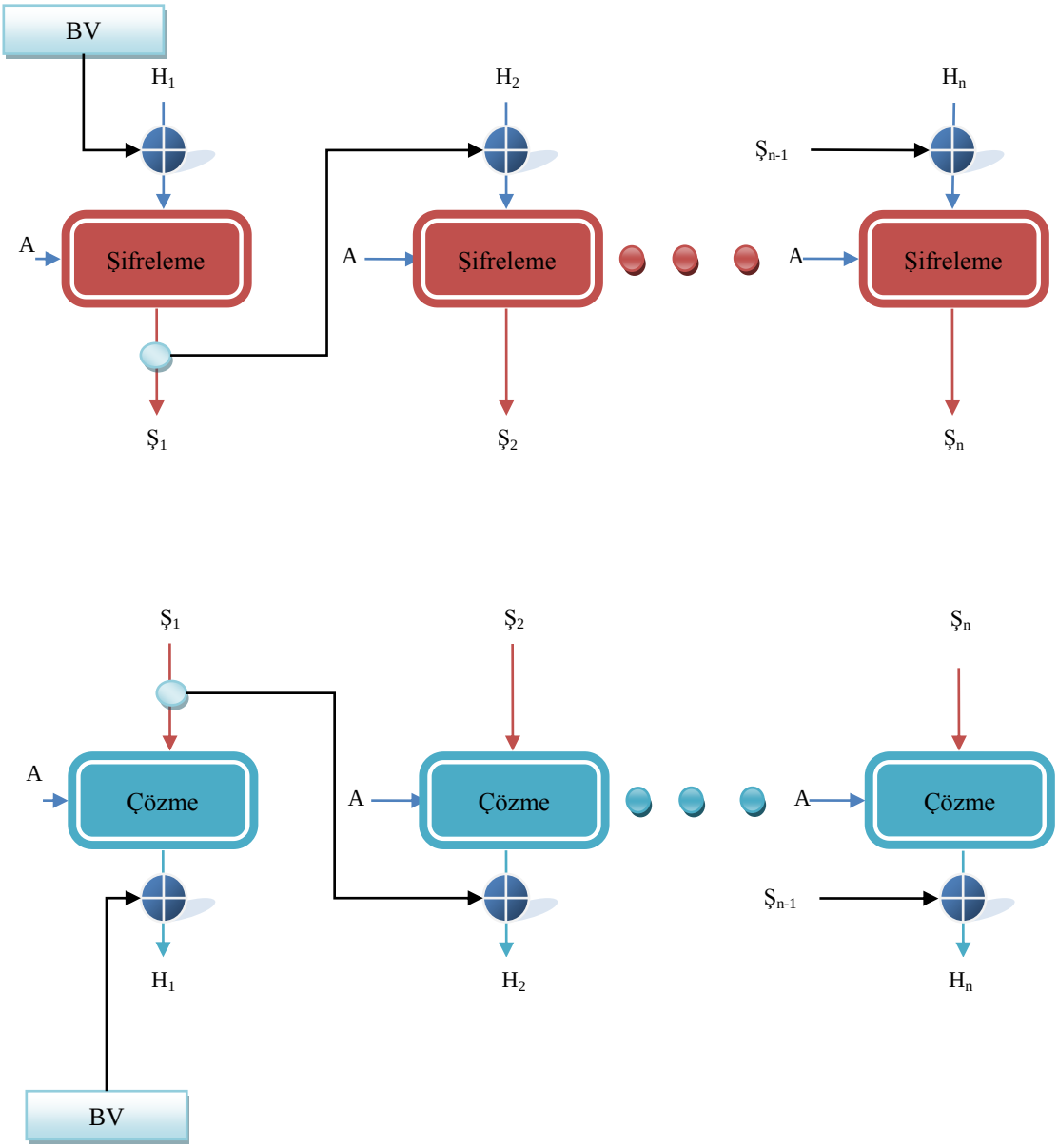
Ham Veri Şifrelenmiş Veri

Şekil 3.5 ECB Kipi Kullanılarak Şifrelenen Bir Resim

Şekil 3.5'te görüldüğü gibi ham veri olarak sisteme girilen resim verisi şifrelendikten sonra benzer bir örüntüyü korumakta ve şifrelendikten sonra da ne olduğu anlaşılmaktadır. Bu örnek, ECB kipinin bu tür veriler için güvensiz bir kip olduğunu göstermektedir.

Şifrelenmiş Kütük Zincirleme Kipi :

CBC kipi, ECB kipinde bahsedilen aynı kütük için aynı şifrelenmiş kütüğün ortaya çıkması sorunundan kurtulmak için 1976 yılında IBM tarafından geliştirilmiştir [5]. CBC kipinde, şifreleme algoritmasına doğrudan ham veri kütüğünün verilmesi yerine ham veri kütüğü kendisinden bir önce şifrelenmiş olan şifrelenmiş veri kütüğüyle özel veya işlemine tabi tutularak şifreleme algoritmasına beslenir. Şekil 3.6'da CBC kipinin çalışması gösterilmektedir.



Şekil 3.6 Şifrelenmiş Kütük Zincirleme Kipi Gösterimi

Şekil 3.6'da H, Ş, A, BV sırasıyla ham veri kütüğü, şifrelenmiş veri kütüğü, anahtar ve başlangıç vektörünü ifade etmektedir. Bu kip kullanılırken; işlemin başında hiç şifrelenmiş veri kütüğü bulunmadığından ham veri kütüğüyle özel veya işlemine katılacak bir veriye ihtiyaç duyulmaktadır. Başlangıç vektörü bu işe yaramaktadır. Başlangıç vektörünün gizli olması şart değildir, ancak; dikkat edilmesi gereken en önemli hususlardan bir tanesi başlangıç vektörünü birden fazla kullanmamaktır. Birden fazla kullanılan başlangıç vektörü, şifreleme işlemini saldırılara karşı zayıf bir

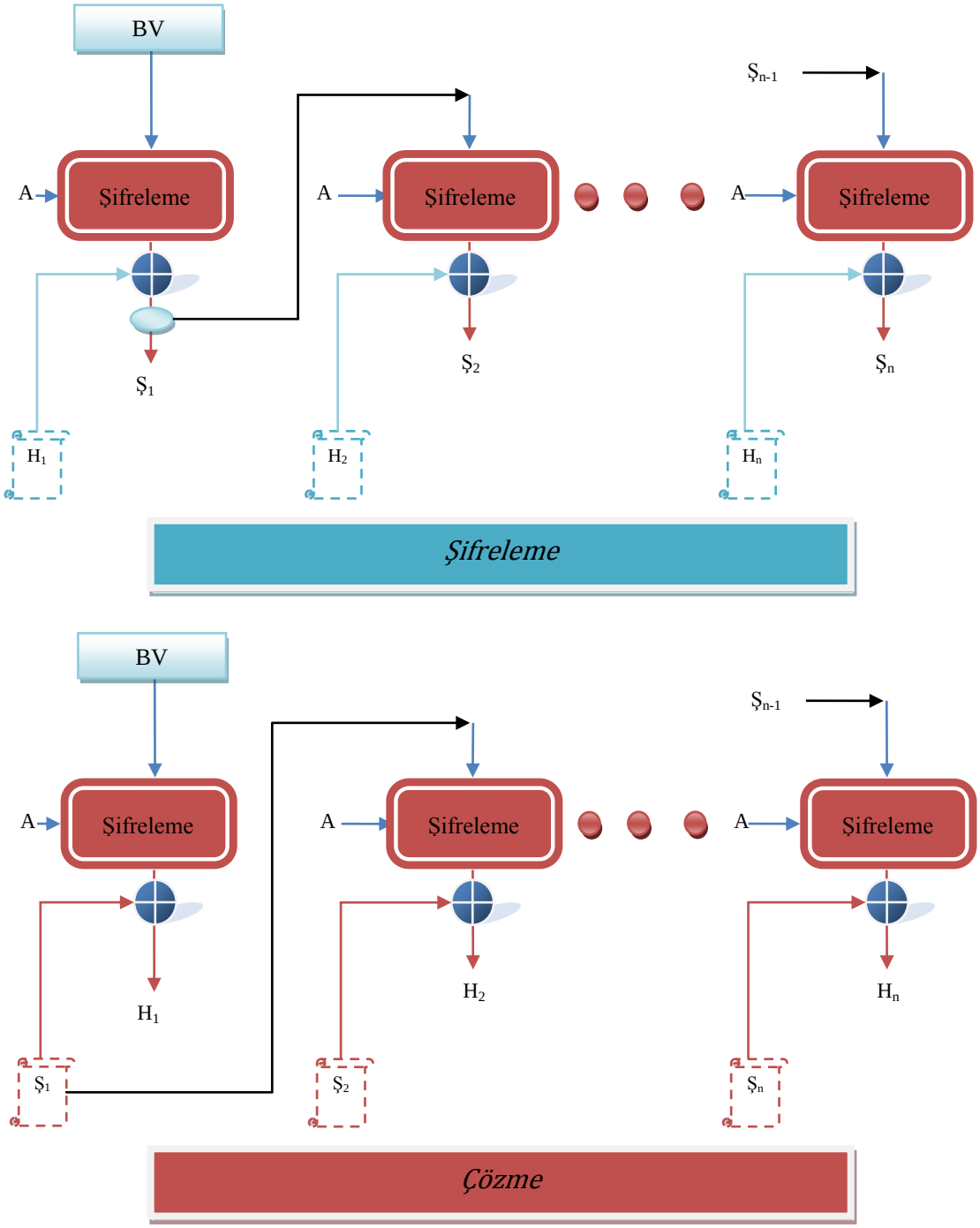
hale getirir. CBC kipi kullanılarak şifrelenmiş kütük oluşturmanın matematiksel ifadesi denklem 3.1'de, çözme algoritmasıyla ham veriye ulaşmanın matematiksel ifadesi ise denklem 3.2'de sunulmaktadır.

$$Ş_i = Şifrele(A, [Ş_{i-1} \oplus H_i]) \quad (3.1)$$

$$H_i = Ş_{i-1} \oplus Çöz(A, Ş_i) \quad (3.2)$$

Şifrelenmiş Veri Geri Besleme Kipi :

CFB kipinde sistem, kendini zamanlayan bir dizi şifreleme sistemi gibi davranmaktadır. Şifrelemenin başlangıç aşamasında, başlangıç vektörü şifrelenir ve ortaya çıkan veri ilk ham veri kütüğüyle özel veya işlemine tabi tutularak ilk şifrelenmiş veri kütüğü elde edilir. Bir sonraki aşamada; elde edilen ilk şifrelenmiş veri kütüğü şifrelenir ve daha sonra ikinci ham veri kütüğüyle özel veya işlemine tabi tutularak ikinci şifrelenmiş veri kütüğü elde edilir ve sistem kütük sayısı kadar devam ettirilir. Şekil 3.7'de CFB kipinin şematik gösterimi sunulmaktadır.



Şekil 3.7 Şifrelenmiş Veri Geri Besleme Kipi Gösterimi

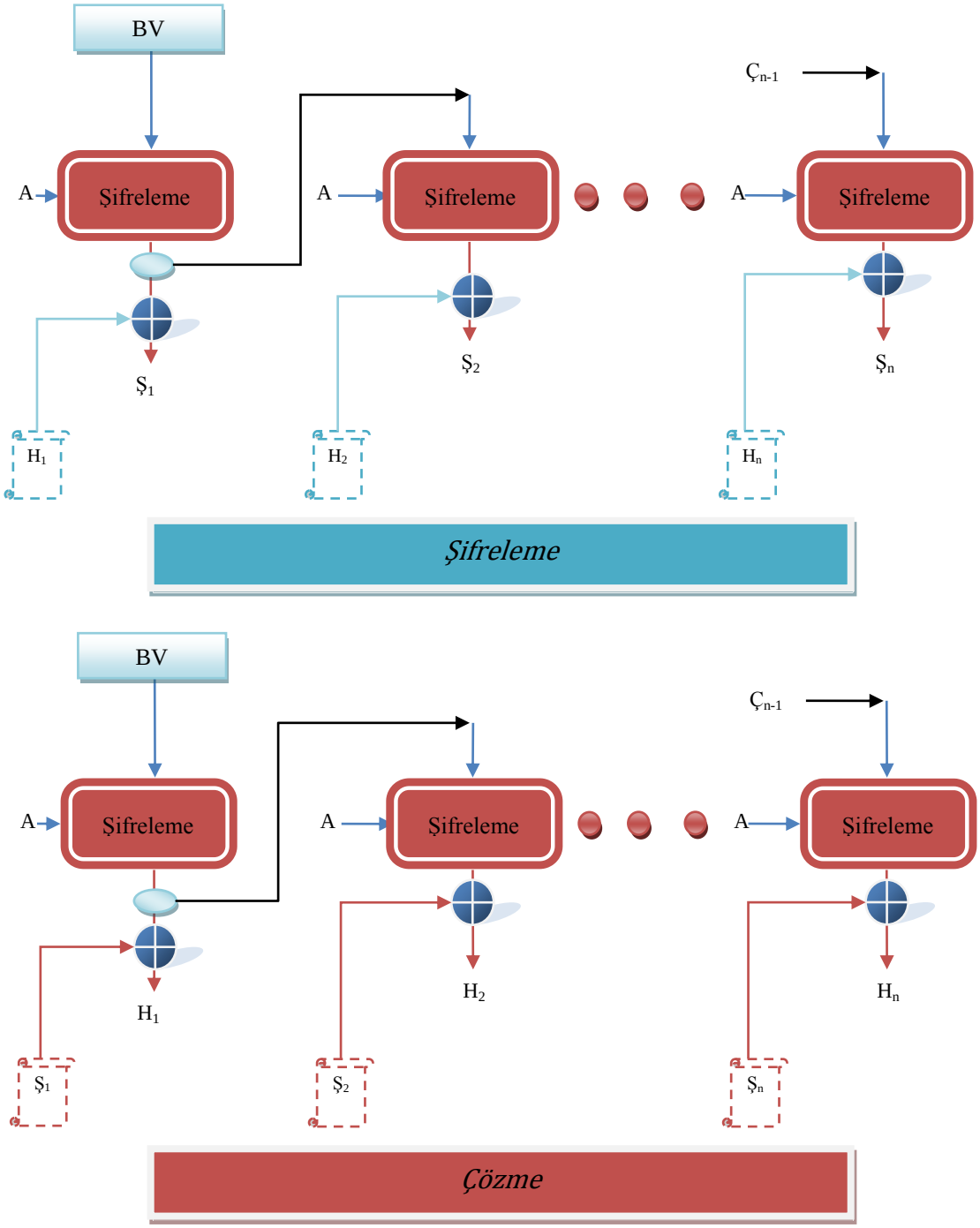
CFB kipinde ECB ve CBC kipinden farklı olarak çözme aşamasında da şifreleme algoritması kullanılmaktadır. CFB'nin sağlayabileceği en büyük olumlu etkilerden bir tanesi herhangi bir sebeple iletilememiş olan şifrelenmiş bitlerin verinin tamamını

etkilememesidir. Bu sebeple algoritmanın girişinde çoğunlukla bir kaydırmalı yazmaç kullanılmaktadır. Bu yöntemde, her kütük şifrelendiğinde oluşan veri kaydırmalı yazmaca sağdan girerek yazmacı kendisi kadar kaydırır. Veri aktarımında oluşabilecek eksik iletim gibi sorunlar bu kipte çözülebilmektedir. Bir başka olumlu yönü ise dizi şifre sistemleri gibi çalıştığından; bu kipte şifrelenecek veri boyutu kütük boyutunun tam katı değil ise ham veriyi kütük boyutunun tam katı haline getirmek gerekmemektedir.

Çıktı Geri Besleme Kipi :

CFB kipinde olduğu gibi OFB kipinde de sistem, kendini zamanlayan bir dizi şifreleme sistemi gibi davranmaktadır. Her aşamada şifreleme algoritmanın çıktısı bir sonraki aşamanın girdisi olarak kullanılmaktadır. Özel veya işleminin bakışımlı olmasından ötürü şifreleme ve çözme işlemi aynı değişim ve dönüşümlerden ibarettir. Yine CFB kipine benzer olarak bu kipte de iletim hatalarına karşı bir olumlu yön bulunmaktadır. Herhangi bir şifrelenmiş veri kütüğünde yanlış bit veya bitler söz konusu olduğunda yalnızca karşılığındaki ham veri kütüğü hatalı olarak elde edilmekte, geri kalan kütüklerde eğer bit hatası yok ise ham verinin kalanı doğru olarak elde edilmektedir. Şekil 3.8'de OFB kipinin şematik gösterimi sunulmaktadır. Şekilde Ç harfi ile ifade edilen veri, şifreleme algoritmasının çıktısıdır.

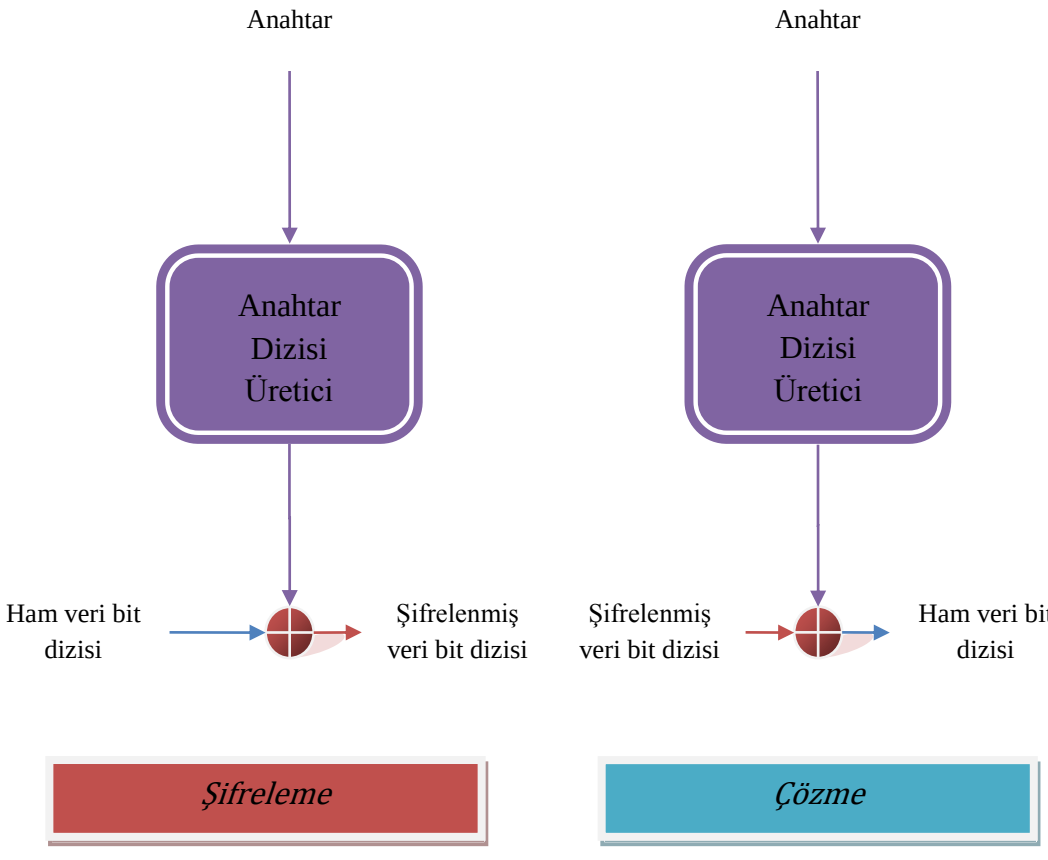
Dizi şifre sistemlerinin kütük şifreleme sistemlerine göre üstün yönleri, çok daha hızlı çalışmaları ve çok daha kolay gerçeklenmeleridir. Buna karşın zayıf yönü ise; farklı anahtar gereksinimleridir.



Şekil 3.8 Çıktı Geri Besleme Kipi Gösterimi

3.1.3.2. Dizi Şifre Sistemleri

Dizi şifre sistemleri, ham veriyi kütük şifreleme sistemlerinde olduğu gibi büyük kütükler halinde işlemek yerine bitler veya baytlar halinde işlemektedir. Tek seferde şifrelenecek veri boyutu dizi şifre sisteminin tasarımına göre değişmektedir. Dizi şifre sistemlerinin genel yapısında bir anahtar üretici bulunur. Anahtar üretici rastgele veya belli bir kurala göre(sözde rastgele) anahtar üretir ve oluşan anahtar ham veriyle özel veya işlemine tabi tutularak şifrelenmiş veri elde edilir. Üretilen anahtar veri genişliğinde olduğu için ham verinin belli bir genişlikte olması gerekmemektedir. Şekil 3.9'da dizi şifre sistemlerinin genel yapısı sunulmaktadır.



Şekil 3.9 Dizi Şifre Sistemleri Genel Gösterimi

Dizi şifre sisteminde her defasında bir birim veri işleme tabi tutulur. Bu sebeple aktarımdan kaynaklı bit hataları yalnızca ilgili bitleri etkilemektedir. Dizi şifre sistemlerinin ifadesi aşağıdaki gibidir:

Ham veri : $h = h_1\ h_2\ h_3 \ldots\ h_n$

Anahtar : $a = a_1\ a_2\ a_3 \ldots\ a_n$

Şifrelenmiş veri : $ş = ş_1\ ş_2\ ş_3 \ldots\ ş_n$ olmak üzere şifrelenmiş veri denklem 3.3'teki gibi ifade edilir.

$$ş_i = h_i \oplus a_i \tag{3.3}$$

Dizi şifre sistemlerinde, genellikle iki tip anahtar üretici bulunmaktadır. Bunlardan ilkli gerçek rastgele anahtar üreticidir. Bu tipte, üretilen anahtar dizisindeki her bit bağımsız olarak elde edilir. Yani anahtar dizisi kuralsız olarak, gerçekten rastgele bir şekilde üretilir. Bu tipin olumsuz yönü anahtarın karşı tarafa iletilmesindeki güçlüktür. Anahtar karşı tarafta üretilemeyecek bir yapıda olduğundan aynen gizli anahtar gibi hedefle kaynak arasında paylaşılmalıdır. Ham veri boyutu kadar anahtar boyutu gerektiğinden anahtarlar kimi zaman oldukça uzun olabilmekte ve anahtar iletimi ciddi sıkıntılara yol açabilmektedir. Buna karşın; olumlu yönü ise tamamen rastgele bir üretim olduğu için ataklara karşı son derece güvenlidir. Anahtar dizisinin bilinmediği durumda şifrelenmiş verinin olası tüm anahtarlarla denenmesi ve elde edilen tüm ham verilerin gözden geçirilmesi gerekmektedir. Anahtar boyutunun veri boyutuna bağlı olduğu düşünüldüğünde bu işlem imkansıza yakın derecede zordur. Bir diğer tip ise sözde rastgele anahtar üreticidir. Bu tipte üretilen anahtar dizisinin her biti kendisinden önce gelen bite bağlıdır. Sözde rastgele anahtar üreticinin iyi tasarlanması oldukça önemlidir. İyi tasarlanmış bir anahtar üreticiye sahip bir dizi şifre sistemi, aynı anahtar genişliğine sahip bir kütük şifre sistemi kadar güvenli olabilmektedir [2]. Dikkat edilmesi gereken en önemli hususlardan bir tanesi anahtar

dizisinin birden fazla kullanılmaması gerekliliğidir. Bu durumun doğuracağı sıkıntı denklem 3.4'te gösterilmektedir.

$$h_1 \oplus a = ş_1 \; ve \; h_2 \oplus a = ş_2 \; ise; \; ş_1 \oplus ş_2 = h_1 \oplus h_2 \tag{3.4}$$

Denklem 3.4'te h harfi ham veri dizisini, a harfi anahtar dizisini ve ş harfi de şifrelenmiş veri dizisini ifade etmektedir. Buna göre; eğer iki farklı ham veri dizisi aynı anahtar dizisi kullanılarak şifrelenecek olursa oluşacak iki farklı şifrelenmiş veri dizisi özel veya işlemine tabi tutulduğunda ham veri dizilerinin özel veya işlemine tabi tutulmasıyla elde edilecek sonucun aynısı elde edilir. Bu durum ciddi bir güvenlik tehdidi oluşturabilmektedir.

3.1.4. Özetleme Fonksiyonları

Özetleme fonksiyonu, çoğunlukla değişken genişlikteki uzun verileri sabit bir genişlikte ifade edebilmek için yapılan matematiksel dönüşümdür. Başka bir deyişle özetleme fonksiyonları verinin bütünlüğünü garantileyen bir parmak izi oluşturmaktadır. Bir özetleme ailesi aşağıdaki koşulları sağlayan dört bileşenden (H, Ş, A, Ö) oluşmaktadır.

1. *H,* olası iletilerin kümesidir.
2. *Ş,* olası özetlenmiş iletilerin sonlu kümesidir.
3. *A,* anahtar uzayı, olası anahtarların sonlu kümesidir.
4. Her A $\in A \; için$ ö$_a$: $H \rightarrow Ş \; ve$ ö$_a$ $\in$
   Ö $olmak$ üzere bir özetleme $fonksiyonu \; vardır.$

Yukarıda H sonsuz elemanlı Ş ise sonlu elemanlı birer küme olarak belirtilmektedir. H kümesinin de sonlu olarak tanımlandığı durumlarda fonksiyon sıkıştırma fonksiyonu olarak adlandırılmaktadır. Özet fonksiyonlarının girdi olarak değişken genişlikte veri alıp çıktı olarak sabit genişlikte veri elde etmelerinden ve oldukça

hızlı çalışmalarından ötürü günümüzde kimlik doğrulama, elektronik imza, bütünlük kontrolü, internet uygulamaları gibi birçok alanda kullanılmaktadırlar [6].

3.1.4.1. SHA-256 Özetleme Algoritması

SHA-256 özetleme algoritması 2001 yılında NIST tarafından yayınlanan güvenli özetleme standartlarından (SHA-2) bir tanesidir [7]. Aynı standart içerisinde farklı parça boyutları içeren SHA-224, SHA-384 ve SHA-512 de bulunmaktadır. Daha önce geliştirilen SHA-1 algoritmasına benzerlik göstermesine karşın çarpışmalardan kaynaklanan matematiksel zayıflığı bünyesinde barındırmayarak SHA-1 algoritmasına yapılan ataklara karşı güvenlidir.

Standartta yayınlanan algoritmalar temel olarak "ön işlem" ve "özet hesaplama" olmak üzere iki aşamadan oluşmaktadırlar. Ön işlemde girdi veri belli bir genişliğe tamamlanır, uygun genişlikli kütüklere ayrılır ve özet hesaplama aşamasında kullanılacak ilk değerler ayarlanır. Özet hesaplama aşamasında ise genişletilmiş veriden bir ileti listesi üretilir ve bu listeye göre gerekli sabitler, fonksiyonlar ve işlemler kullanılarak özet veri oluşturulur. SHA algoritmaları, işlem yapılan kütük genişlikleri girdi ve çıktı genişlikleri ve güvenlik seviyeleri ile birbirinden farklılık göstermektedir. Çizelge 3.1'de SHA algoritmalarının özellikleri sunulmaktadır.

Çizelge 3.1 SHA Algoritmalarının Özellikleri

Algoritma	Girdi Boyutu (bit)	Kütük Boyutu (bit)	Kelime Boyutu (bit)	Özet Boyutu (bit)
SHA-1	$< 2^{64}$	512	32	160
SHA-224	$< 2^{64}$	512	32	224
SHA-256	$< 2^{64}$	512	32	256
SHA-384	$< 2^{128}$	1024	64	384
SHA-512	$< 2^{128}$	1024	64	512

SHA-256 Fonksiyonları :

SHA-256 algoritması 32 bitlik kelimeler üzerine uygulanan aşağıdaki altı fonksiyonu içermektedir. Gösterimde x, y ve z harfleri 32 bitlik kelimeleri ifade etmektedir.

$$Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z) \quad (3.5)$$

$$Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \quad (3.6)$$

$$\sum_0^{\{256\}}(x) = SDÖN^2(x) \oplus SDÖN^{13}(x) \oplus SDÖN^{22}(x) \quad (3.7)$$

$$\sum_1^{\{256\}}(x) = SDÖN^6(x) \oplus SDÖN^{11}(x) \oplus SDÖN^{25}(x) \quad (3.8)$$

$$\sigma_0^{\{256\}}(x) = SDÖN^7(x) \oplus SDÖN^{18}(x) \oplus SKAY^3(x) \quad (3.9)$$

$$\sigma_1^{\{256\}}(x) = SDÖN^{17}(x) \oplus SDÖN^{19}(x) \oplus SKAY^{10}(x) \quad (3.10)$$

Yukarıdaki denklemlerde SDÖN fonksiyonu dairesel olarak sağa kaydırma, SKAY fonksiyonu ise sağa kaydırmayı ifade etmektedir. Bu fonksiyonların matematiksel ifadeleri denklem 3.11 ve 3.12'de sunulmaktadır.

$x, k\ bitlik\ kelime\ ve\ 0 \leq n < k\ olmak$ üzere;

$$SDÖN^n(x) = (x \gg n) \vee (x \ll k - n) \quad (3.11)$$

$$SKAY^n(x) = x \gg n \quad (3.12)$$

SHA-256 Sabitleri :

SHA-256 algoritmasında 64 adet 32 bitlik sabit kullanılmaktadır. Bu sabitler, ilk 64 asal sayının küp köklerinin kesirli kısımlarının ilk 32 bitlerinden oluşmaktadırlar. Sabitler $S_0^{\{256\}}$, $S_1^{\{256\}}$, …, $S_{63}^{\{256\}}$ şeklinde ifade edilmektedirler. Çizelge 3.2'de bu sabitler onaltılık tabanda gösterilmektedir.

Çizelge 3.2 SHA-256 Sabitleri

indisler (8 tabanında)	0	1	2	3	4	5	6	7
0	428a2f98	71374491	b5c0fbcf	e9b5dba5	3956c25b	59f111f1	923f82a4	ab1c5ed5
1	d807aa98	12835b01	243185be	550c7dc3	72be5d74	80deb1fe	9bdc06a7	c19bf174
2	e49b69c1	efbe4786	0fc19dc6	240ca1cc	2de92c6f	4a7484aa	5cb0a9dc	76f988da
3	983e5152	a831c66d	b00327c8	bf597fc7	c6e00bf3	d5a79147	06ca6351	14292967
4	27b70a85	2e1b2138	4d2c6dfc	53380d13	650a7354	766a0abb	81c2c92e	92722c85
5	a2bfe8a1	a81a664b	c24b8b70	c76c51a3	d192e819	d6990624	f40e3585	106aa070
6	19a4c116	1e376c08	2748774c	34b0bcb5	391c0cb3	4ed8aa4a	5b9cca4f	682e6ff3
7	748f82ee	78a5636f	84c87814	8cc70208	90befffa	a4506ceb	bef9a3f7	c67178f2

SHA-256 Ön İşlemler :

SHA-256 algoritmasında girdi verisini genişletip parçalamak için bir takım ön işlemler uygulanmaktadır. Girdi verisini genişletmek için verinin sonuna "1" biti ve denklem 3.13'ten elde edilecek sayı kadar "0" biti eklenir.

$$girdi\ verisinin\ uzunluğu\ u\ ve\ eklenecek\ sıfır\ sayısı\ k\ olmak\ \text{üzere};$$

$$u + 1 + k \equiv 448\ (mod\ 512) \tag{3.13}$$

Yukarıdaki denklemde k, denklemi sağlayacak en küçük pozitif tam sayıdır. Daha sonra verinin sonuna, u sayısının ikilik tabandaki karşılığı olan 64 bitlik veri eklenir. Bu işlemlerin sonucunda genişletilmiş mesaj 512 bitin tam katı uzunluğunda olmaktadır.

Bu işlemden sonra genişletilen verinin ayrıştırılması işlemi gerçekleştirilir. SHA-256 algoritması ayrıştırma işleminde genişletilmiş veri 512 bitlik kütüklere ayrılır. Her kütük 16 tane 32 bitlik kelimeden oluşmaktadır. n. kütük için bu kelimeler $G_0^{(n)}$, $G_1^{(n)}$, …, $G_{15}^{(n)}$ şeklinde ifade edilmektedir.

SHA-256 algoritmasının ön işlemlerinin son basamağında ise başlangıç özet değerleri ayarlanmaktadır. Sekiz adet 32 bitlik kelimeden oluşan bu değerler, ilk sekiz asal sayının kare köklerinin kesirli kısımlarının ilk 32 bitlerinden elde edilmektedirler. Çizelge 3.3'te başlangıç özet değerleri onaltılık gösterimde sunulmaktadır.

Çizelge 3.3 SHA-256 Başlangıç Özet Değerleri

Başlangıç özet kelimesi	Değer
$Ö_0^{(0)}$	6a09e667
$Ö_1^{(0)}$	bb67ae85
$Ö_2^{(0)}$	3c6ef372
$Ö_3^{(0)}$	a54ff53a
$Ö_4^{(0)}$	510e527f
$Ö_5^{(0)}$	9b05688c
$Ö_6^{(0)}$	1f83d9ab
$Ö_7^{(0)}$	5be0cd19

SHA-256 Özet Hesaplama :

Ön işlemler tamamlandıktan sonra, $G^{(1)}$, $G^{(2)}$, …, $G^{(N)}$ veri kütükleri aşağıdaki adımlarda gösterildiği gibi işlenir. Tüm adımlar, i indis olmak üzere; i = 1'den N'ye kadar tekrarlanır. (Toplama işlemleri modulo 2^{32}'de yapılmaktadır.)

1. Denklem 3.14'te gösterildiği gibi ileti listesi (L_z) hesaplanır.

$$L_z = \begin{cases} G_z^{(i)} & 0 \leq z \leq 15 \\ \sigma_1^{\{256\}}(L_{z-2}) + L_{z-7} + \sigma_0^{\{256\}}(L_{z-15}) + L_{z-16} & 16 \leq z < 63 \end{cases} \quad (3.14)$$

2. Sekiz adet çalışma değişkeni (a, b, c, ç, d, e, f, g), (i-1). özet değeriyle başlatılır.

$a = Ö_0^{(i-1)}$

$b = Ö_1^{(i-1)}$

$c = Ö_2^{(i-1)}$

$ç = Ö_3^{(i-1)}$

$d = Ö_4^{(i-1)}$

$e = Ö_5^{(i-1)}$

$f = Ö_6^{(i-1)}$

$g = Ö_7^{(i-1)}$

3. z = 0'dan 63'e kadar aşağıdaki işlemler yapılır.

$T_1 = g + \sum_1^{\{256\}}(d) + Ch(d,e,f) + S_z^{\{256\}} + L_z$

$T_2 = \sum_0^{\{256\}}(a) + Maj(a,b,c)$

$g = f$

$e = d$

$d = ç + T_1$

$ç = c$

$c = b$

$b = a$

$a = T_1 + T_2$

4. i. ara özet değerleri aşağıdaki gibi hesaplanır.

$Ö_0^{(i)} = a + Ö_0^{(i-1)}$

$Ö_1^{(i)} = a + Ö_1^{(i-1)}$

$Ö_2^{(i)} = a + Ö_2^{(i-1)}$

$Ö_3^{(i)} = a + Ö_3^{(i-1)}$

$Ö_4^{(i)} = a + Ö_4^{(i-1)}$

$Ö_5^{(i)} = a + Ö_5^{(i-1)}$

$Ö_6^{(i)} = a + Ö_6^{(i-1)}$

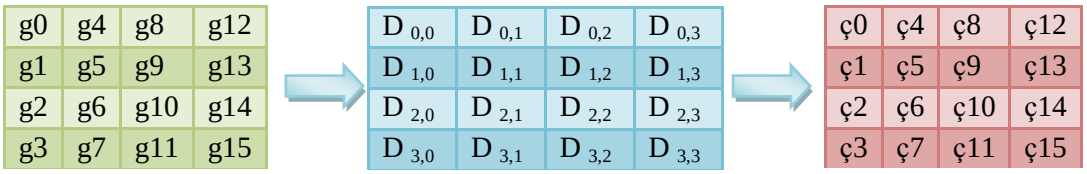
$Ö_7^{(i)} = a + Ö_7^{(i-1)}$

Yukarıdaki dört adım N kez tekrarlandıktan sonra, başka bir deyişle son veri kütüğü işlendikten sonra SHA-256 algoritmasının çıktısı olan 256 bitlik özet ($Ö_0^{(N)}$, $Ö_1^{(N)}$, …, $Ö_7^{(N)}$) oluşmaktadır [7].

4. AES ALGORİTMASI

AES algoritması, geliştiricilerinin isminden türetilen Rijndael ismiyle de anılan bir kütük şifreleme algoritmasıdır. AES' den önce var olan DES algoritmasının iyi bir algoritma olduğunun kabul edilmesine rağmen; düşük devir sayısı , düşük şifre parça boyutu sebebiyle ve dönemin meşhur EFF paralel bilgisayarı Deep Crack'e yapılan test saldırılarının sonucunda DES algoritması yerini AES algoritmasına bırakmıştır [8]. AES algoritması, NIST(National Institute of Standards and Technology) tarafından düzenlenen yarışma kapsamında Joan Daeman ve Vincent Rijmen tarafından geliştirilmiştir. Doğrulama ve standartlaştırma aşamaları sonrasında NIST, 26 Kasım 2001 yılında FIPS(Federal Information Processing Standard) 197 standardı adı altında AES algoritmasını yayınlamıştır [9].

FIPS 197 standardında belirtilen Rijndael algoritması, 128 bitlik veri kütüklerini, 128, 192 veya 256 bitlik genişliklerde şifreleme anahtarları kullanarak işleyen bir bakışımlı(simetrik) kütük şifreleme algoritmasıdır. 128 bitlik yalın veri algoritmaya beslenir. Algoritma içerisinde bu veri 4x4'lük bir matrisle ifade edilir ve algoritmanın dönüşüm işlemleri bu veri üzerinde uygulanır [10]. Dönüşüm işlemlerinin uygulandığı bu matrise durum matrisi adı verilir. Algoritmanın çıktısı ise; işlenmiş(şifrelenmiş) 128 bitlik bir veridir. Bu yapı Şekil 4.1'de sunulmaktadır.



Şekil 4.1 AES Algoritması durum dizisi, girdi ve çıktılar.

AES Algoritmasının girdisi Şekil 4.1'de gösterildiği gibi durum dizisine denklem 4.1'deki formül uygulanarak kopyalanır. Kopyalama işleminden sonra algoritma süresince dönüşüm işlemleri hep bu durum dizisi üzerine uygulanır.

$$D[s,u] = g[s+4u] \quad ; \quad 0 \leq s < 4 \; ve \; 0 \leq u < 4 \tag{4.1}$$

AES algoritması 128, 192 ve 256 bit olmak üzere farklı anahtar genişlikleriyle çalışabilmektedir. Anahtar genişliğine göre algoritmanın dönüşüm işlemlerinin devir sayısı değişmektedir. Farklı anahtar genişlikleri için gerekli devir sayıları Çizelge 4.1'de sunulmaktadır.

Çizelge 4.2 AES algoritması için anahtar-kütük-devir bileşimleri

AES tipi	Anahtar Genişliği (bit)	Kütük Boyutu (bit)	Devir Sayısı
AES-128	128	128	10
AES-192	192	128	12
AES-256	256	128	14

AES algoritması genel olarak iki kısımda incelenebilir. Bu kısımlar devir anahtarı oluşturma(anahtar genişletme) ve devir dönüşüm işlemleridir. Girdi anahtarından devir anahtarları oluşturulduktan sonra anahtar genişliğine bağlı olarak devir dönüşüm işlemleri durum dizisine farklı sayılarda uygulanırlar. Algoritma tamamlanırken son durum çıktıya kopyalanarak sonuç elde edilir.

4.1. Anahtar Genişletme

AES algoritmasının üstün yönlerinden bir tanesi de girilen anahtardan farklı sayıda anahtarlar üreterek şifreleme devirlerinde farklı anahtarları kullanmasıdır. Bu sayede, tekrarlı verilerden farklı şifrelenmiş veriler üreterek karmaşıklık arttırılmış olur. AES kütük boyutu 128 bit olduğu için girdi anahtarının boyutu 128 de olsa, 192 de olsa, 256 da olsa üretilen devir anahtarları 128 bit genişliğinde olacaktır. Üretilen bu anahtarlar her bir devirin sonunda özel veya işlemi kullanılarak durum dizisine eklenmektedirler. Devir anahtarı sayısı algoritmanın devir sayısına bağlı olarak

değişir. Genel olarak AES algoritmasında üretilen toplam anahtar boyutu kelime cinsinden devir sayısının bir fazlasının dört katı olarak ifade edilebilir. Başka bir deyişle AES-128 için 176, AES-192 için 208, AES-256 için ise 240 baytlık anahtar üretilmektedir.

Anahtar genişletme aşaması temel olarak üç ana işlevi bünyesinde barındırır. Bunlardan birincisi "kelime döndürme" işlevidir. Kelime döndürme 32 bitlik bir değerin en soldaki baytını en sağa alma, döngüsel permutasyon işlemidir. Örneğin $[v_0, v_1, v_2, v_3]$ dört baytlık değeri döndürülerek $[v_1, v_2, v_3, v_0]$ dört baytlık değeri elde edilir.

Bir diğer işlev "kelime değiştirme" işlevidir. Bu işlevde girdi olarak alınan 4 baytlık değer S-kutusu yardımıyla bir başka değere dönüştürülerek 4 baytlık çıktı verilir.

Üçüncü işlevde ise "devir sabiti ekleme" işlevidir. Bu işlevdeki sabit dizisi kısaca "Dsabit" şeklinde adlandırılacak olursa, sabitin elemanları denklem 4.2'deki gibi bulunabilmektedir.

$$Dsabit[i] = [x^{i-1}, \{00\}, \{00\}, \{00\}] \tag{4.2}$$

Yukarıdaki denklemde x değeri onaltılık tabanda {02} olarak ifade edilmekte ve x^{i-1} değeri $GF(2^8)$ alanında hesaplanmaktadır. Devir sabiti işlevinin girdisiyle uygun Dsabit değeri özel veya işlemine tabi tutularak işlevin çıktısı hesaplanır. Çizelge 4.2'de örnek bir anahtar için AES-256 anahtar genişletme işlevinden bir kısım sunulmaktadır.

Çizelge 4.3 AES-256 için Örnek Anahtar Genişletme

k_0 = 603deb10 **k_1 = 15ca71be** **k_2 = 2b73aef0** **k_3 = 857d7781**
k_4 = 1f352c07 **k_5 = 3b6108d7** **k_6 = 2d9810a3** **k_7 = 0914dff4**

Ak = 8

i	Geçici Değer	Kelime Döndürme Çıkışı	Kelime Değiştirme Çıkışı	Döngü Sabiti	Döngü Sabiti Ekleme	k[i - Ak]	k[i]
8	0914dff4	14dff409	fa9ebf01	01000000	fb9ebf01	603deb10	9ba35411
9	9ba35411					15ca71be	8e6925af
10	8e6925af					2b73aef0	a51a8b5f
11	a51a8b5f					857d7781	2067fcde
12	2067fcde		b785b01d			1f352c07	a8b09c1a
13	a8b09c1a					3b6108d7	93d194cd
14	93d194cd					2d9810a3	be49846e
15	be49846e					0914dff4	b75d5b9a

Çizelge 4.2'de [k_0 – k_7] aralığı girdi anahtarını ifade etmektedir. Görüldüğü üzere kelime döndürme ve döngü sabiti ekleme işlevleri her 8 dönüşte bir kez uygulanmaktadır. Kelime değiştirme işlevi ise her üretilen 128 bitlik anahtar için bir kez uygulanmaktadır.

4.2. Dönüşüm İşlemleri

AES algoritması Bayt Değiştirme, Satır Kaydırma, Sütun Karıştırma ve Devir Anahtarı Ekleme olmak üzere dört adet dönüşüm işlemi içermektedir. Algoritmanın devir sayısı, anahtar genişliğine bağlı olarak değişmekte ve her bir devirde bu işlemler durum matrisine uygulanarak tekrar etmektedir.

4.2.1. Bayt Değiştirme Dönüşümü

Bayt değiştirme dönüşümü, durum matrisinin her baytına ayrı ayrı uygulanan ve doğrusal olmayan bir dönüşüm işlemidir. Dönüşümü gerçekleştirmek için S-kutusu adı verilen tersinir bir dönüşüm tablosu kullanılmaktadır. S-kutusu dönüşüm tablosunun oluşturulabilmesi için öncelikle GF(2^8) alanında girdinin çarpmaya göre tersi alınmalıdır. AES algoritmasında, Galois alanında kullanılan indirgeme polinomu P(x) = $x^8 + x^4 + x^3 + x + 1$ şeklinde ifade edilmektedir. Bu aşamadan sonra denklem 4.3'te ifade edilen formül yardımıyla bir baytın dönüşümü tamamlanır.

$$b_i^{'} = b_i \oplus b_{(i+4)mod\,8} \oplus b_{(i+5)mod\ 8} \oplus b_{(i+6)mod\ 8} \oplus b_{(i+7)mod\ 8} \oplus c_i \quad (4.3)$$

Denklem 4.3'te $0 \leq i \leq 8$ olmak üzere; b_i ve c_i, b ve c baytlarının i. bitlerini temsil etmektedir. "c" baytı ise ikilik tabanda 01100011 değerine eşittir. Bu denklemdeki dönüşüm GF(2) alanında uygulanmaktadır. Bahsedilen dönüşümün matris halinde gösterimi denklem 4.4'te sunulmaktadır.

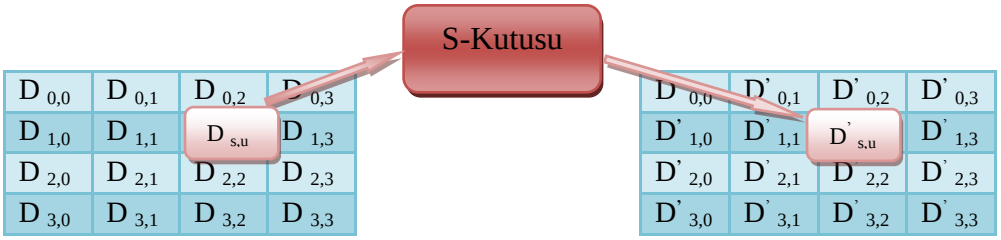
$$\begin{bmatrix} b_0^{'} \\ b_1^{'} \\ b_2^{'} \\ b_3^{'} \\ b_4^{'} \\ b_5^{'} \\ b_6^{'} \\ b_7^{'} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (4.4)$$

Yukarıda bahsedilen adımlar uygulanarak 8 bit ile ifade edilebilecek tüm sayılar için bir dönüşüm tablosu(S-kutusu) hazırlanır. AES algoritması için S-kutusu çizelge 4.3'te sunulmuştur.

Çizelge 4.4 AES Algoritması S-kutusu

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	C7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

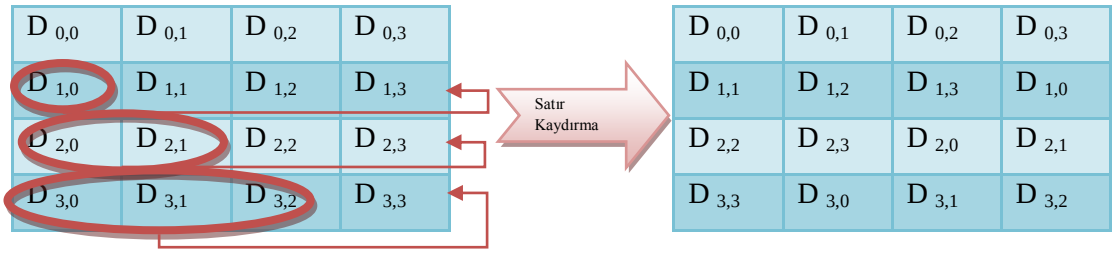
Çizelge 4.3'e bakılarak bir baytın dönüşümü rahatlıkla yapılabilmektedir. Örneğin; onaltılık gösterimi {61} olan baytın dönüşümden sonraki hali tablodan {ef} olarak bulunur. Dönüşüm, durum matrisinde bulunan her bayt için uygulanmakta ve durum matrisi güncellenmektedir. Bu durum şekil 4.2'de gösterilmektedir.



Şekil 4.2 AES Algoritması Bayt Dönüşümünün Durum Matrisine Uygulanması

4.2.2. Satır Kaydırma Dönüşümü

Satır kaydırma dönüşümü durum matrisinin her satırına ayrı ayrı uygulanan dairesel bir kaydırma işlemidir. Her satır farklı sayılarda kaydırılmak suretiyle durum matrisi güncellenir. İlk satır sabit bırakılır, ikinci üçüncü ve dördüncü satırlar dairesel olarak sırasıyla 1, 2 ve 3 bayt kaydırılarak yeni veriler elde edilir. Şekil 4.3'te sütun karıştırma dönüşümünün şematik gösterimi sunulmuştur.



Şekil 4.3 AES Algoritması Satır Kaydırma Dönüşümü

AES algoritmasının her devrinde yukarıda görülen satır kaydırma dönüşümü durum matrisine uygulanmaktadır.

4.2.3. Sütun Karıştırma Dönüşümü

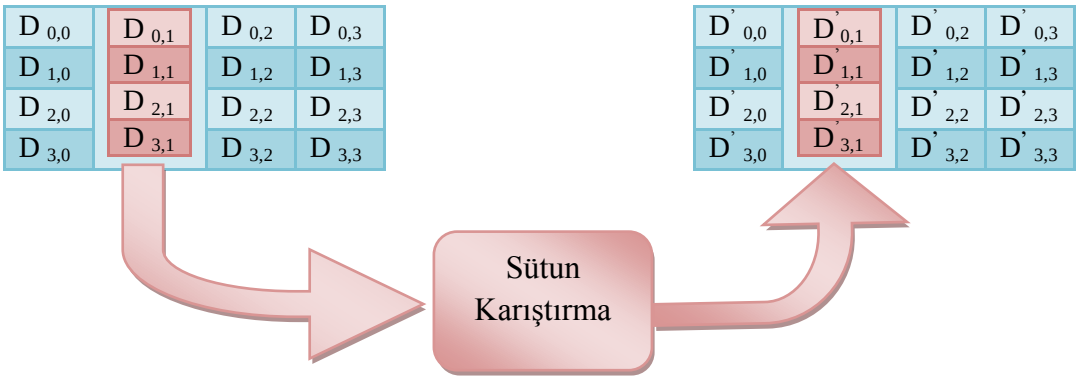
Sütun karıştırma dönüşümü; durum matrisinin sütunlarına, $GF(2^8)$ alanında birer dört terimli polinom gibi davranılarak uygulanır. Sütunlar, denklem 4.5'te verilen sabit polinomla modulo $x^4 + 1$'de çarpılarak dönüşüm gerçekleştirilir.

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \quad (4.5)$$

Bahsedilen çarpımın matris gösterimi denklem 4.6'da sunulmaktadır.

$$\begin{bmatrix} d'_{0,u} \\ d'_{1,u} \\ d'_{2,u} \\ d'_{3,u} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} d_{0,u} \\ d_{1,u} \\ d_{2,u} \\ d_{3,u} \end{bmatrix} \quad ; \quad 0 \leq u < 4 \tag{4.6}$$

Sütun karıştırma işlemi durum matrisinin her sütunu için uygulanırken durum matrisi güncellenir ve işlem tamamlandığında bir sonraki aşamaya geçilir. Sütun karıştırma dönüşümünün şematik gösterimi şekil 4.4'te sunulmuştur.

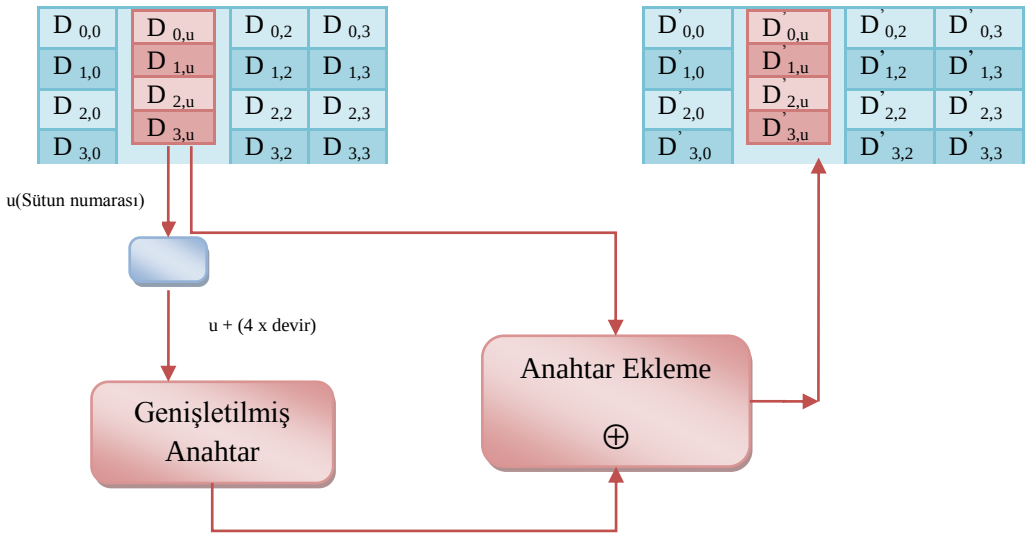


Şekil 4.4 Sütun Karıştırma Dönüşümü

4.2.4. Devir Anahtarı Ekleme Dönüşümü

Bu dönüşümde, bölüm 4.2'de anlatılan anahtar genişletme işlemi sonucunda oluşan anahtarlarlar durum matrisinin sütunlarına özel veya işlemi ile eklenir. Anahtar genişletme aşamasında oluşturulan anahtarlar 32'şer bit olarak paketlenir, içinde bulunulan devir sayısına ve durum matrisinin hangi sütunuyla işlem yapıldığına göre doğru anahtar seçilerek durum matrisinin ilgili sütunu ile özel veya işlemine tabi

tutulur. Şekil 4.5'te devir anahtarı ekleme dönüşümünün şematik gösterimi sunulmaktadır.

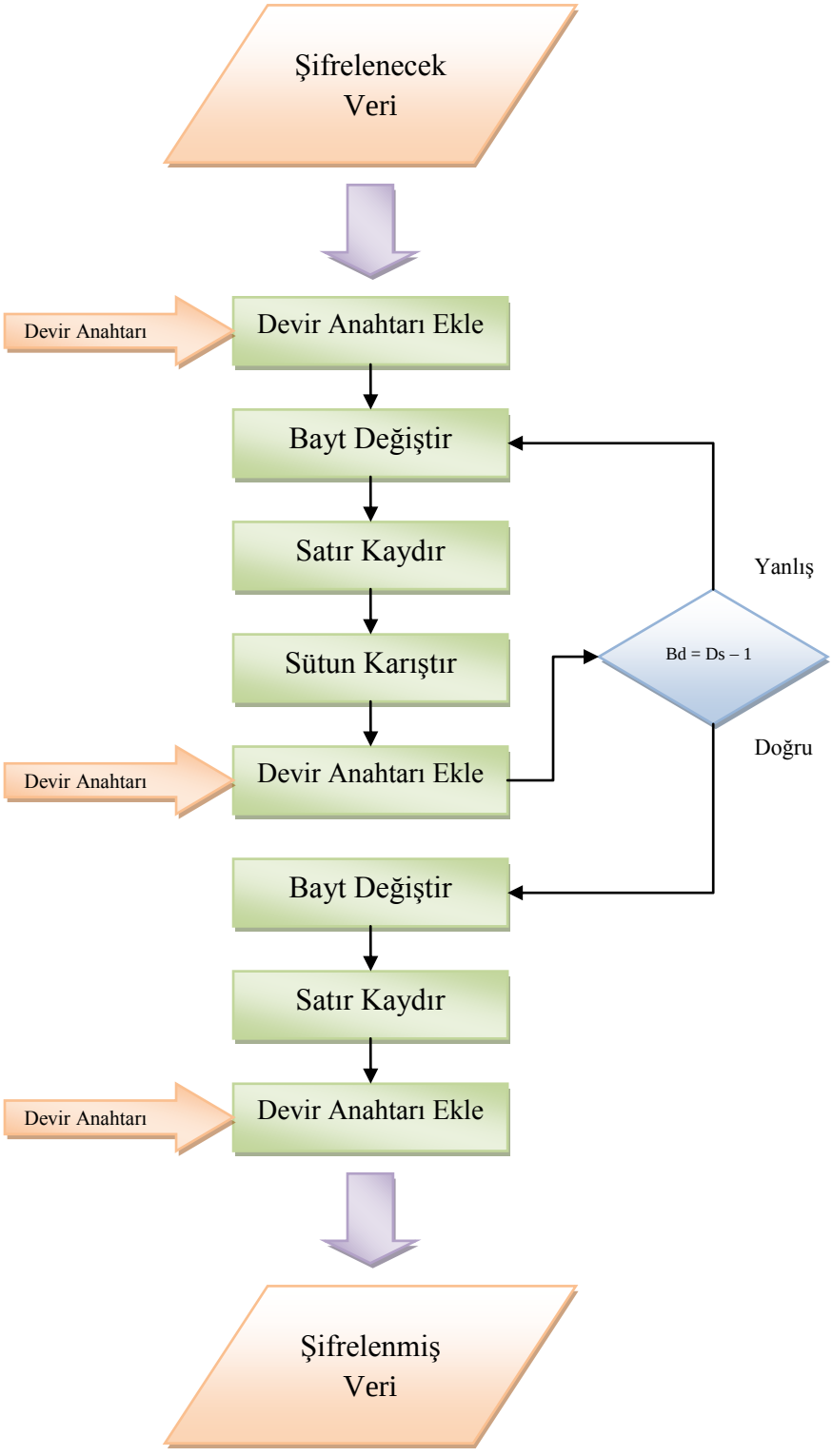


Şekil 4.5 Devir Anahtarı Ekleme Dönüşümü

Şekil 4.5'te görüldüğü gibi; genişletilen anahtarlardan uygun olanını seçmek için içinde bulunulan devir sayısı ve durum matrisinin işlem yapılan sütun numarasına ihtiyaç duyulmaktadır. İçinde bulunulan devirin 4 katına sütun numarası eklenerek kaçıncı 32 bitlik anahtar parçasının kullanılacağı seçilir ve durum matrisinin ilgili sütunu ile bu değer özel veya işlemine tabi tutularak durum matrisi güncellenir.

4.3. AES Şifreleme Algoritmasının Akışı

Bölüm 4.2, ve 4.3'te bahsedilen işlemler AES şifreleme algoritmasının temelini oluşturmaktadır. Algoritmanın genel akış şeması şekil 4.6'da sunulmaktadır.



Şekil 4.6 AES Şifreleme Algoritması Akış Şeması

Şekil 4.6'da görüldüğü gibi algoritmanın başlangıcında bir kez devir anahtarı ekleme işlemi yapılır. Daha sonra devirler başlar ve son devire kadar her devirde sırasıyla bayt değiştirme, satır kaydırma, sütun karıştırma ve devir anahtarı ekleme dönüşümleri uygulanır. Son devirde ise sütun karıştırma dönüşümü uygulanmayıp yalnızca bayt değiştirme, satır kaydırma, devir anahtarı ekleme dönüşümleri uygulanarak durum matrisi çıkışa kopyalanır ve 128 bitlik şifrelenmiş veri elde edilir.

AES şifreleme algoritmasıyla şifrelenen 128 bitlik verinin çözülmesi işlemi, AES şifreleme algoritmasının matematiksel olarak ters işleyeni olan AES çözme algoritması kullanılarak gerçekleştirilir. Çözme algoritmasında anahtar genişletme kısmı şifreleme algoritmasıyla aynı çalışmaktadır. Bayt değiştirme dönüşümünde; S-kutusunun yerine ters S-kutusu kullanmak dönüşüm tablosuyla gerçekleme açısından yeterli olmaktadır. Satır kaydırma işlemi yerine, satır kaydırmanın tersini yapan ters satır kaydırma dönüşümü kullanılır. Son olarak sütun karıştırma işleminde; çarpanların sonlu alanda çarpmaya göre tersleri kullanılarak ters sütun karıştırma dönüşümü gerçekleştirilir.

5. MATEMATİKSEL ÖN BİLGİ

Bu bölümde, çalışmada kullanılan matematiksel kavramlar için gerekli ön bilgiler sunulmaktadır.

5.1. Sonlu Alan Teorisi

Sonlu alanlar ve alt kümeleri, günümüzde birçok şifreleme algoritmasında kullanılmakta ve birçok algoritmanın temelini oluşturmaktadırlar. Sonlu alan teorisini ifade edebilmek için aşağıdaki tanımlar yapılmaktadır [11].

5.1.1. Değişmeli Grup

$$+ : G\ x\ G \rightarrow G : (a, b) \rightarrow a\ + b \tag{5.1}$$

G kümesi ve denklem 5.1'de ifade edilen + işleminin oluşturduğu <G, +> yapısı aşağıdaki beş özelliği sağladığı takdirde bir değişmeli gruptur.

1. $Kapalılık$: $\forall\ a, b\ \in G : (a + b) \in G$ (5.2)
2. $Birleşme$: $\forall\ a, b, c\ \in G : (a + b) +\ \mathrm{c} = \mathrm{a} + (\mathrm{b} + \mathrm{c})$ (5.3)
3. $Değişme$: $\forall\ a, b\ \in G : \mathrm{a} + \mathrm{b} = \mathrm{b} + \mathrm{a}$ (5.4)
4. $Etkisiz\ Eleman$: $\exists\ \mathrm{e}\ \in \mathrm{G}, \forall\ \mathrm{a}\ \in \mathrm{G} : \mathrm{a} + \mathrm{e} = \mathrm{a}$ (5.5)
5. $Ters\ Eleman$: $\forall\ a \in G, \exists\ b\ \in G : \mathrm{a} + \mathrm{b} = 0$ (5.6)

5.1.2. Halka

Halka, bir R kümesi ve bu küme üzerinde tanımlanmış olan "+" ve "*" olmak üzere iki adet işlemden oluşur. <R, +, *> yapısının bir halka oluşturabilmesi için aşağıdaki üç koşulun sağlanması gerekmektedir.

1. <R, +> yapısı bir değişmeli grup oluşturmalıdır.
2. * işlemi, R kümesi üzerinde, birleşme ve kapalılık özelliklerine sahip olmalı ve bu işlem için bir etkisiz eleman tanımlanabilmelidir.
3. * işleminin, + işlemi üzerinde dağılma özelliği olmalıdır:

$$\forall\, a, b, c \ \in R : c * (a + b) = (c * a) + \ (c * b) \tag{5.7}$$

"*" işlemi değişme özelliğini sağladığı takdirde; <R, +, *> halkası, değişmeli halka olarak adlandırılır.

5.1.3. Alan

F kümesi ve F kümesi üzerinde tanımlanmış "+" ve "*" işlemleri, aşağıdaki koşulları sağladığı takdirde bir alan oluştururlar.

1. <F, +> bir değişmeli grup olmalıdır.
2. <F, *> bir değişmeli grup olmalıdır ancak; yalnızca toplama işleminin etkisiz elemanı için bir ters eleman bulunmayabilir.
3. <F, +, *> bir halka olmalıdır.

5.1.4. Sonlu Alan

Bir alan, eğer sonlu sayıda eleman içeriyorsa o alan; sonlu alan olarak isimlendirilir. Bir sonlu alanın eleman sayısı, o sonlu alanın derecesini ifade etmektedir. Eğer iki sonlu alanın derecesi aynı ise bu alanlara eş yapılı alanlar denmektedir. Başka bir deyişle, matematiksel olarak aynı yapıya sahiptirler ancak elemanlarının gösterilişleri farklıdır. F_q şeklinde gösterilen bir alanın sonlu alan olabilmesi için denklem 5.8'de ifade edilen eşitliği sağlaması gerekmektedir.

$$p\ bir\ asal\ sayı\ ve\ m\ bir\ tamsayı\ olmak\ üzere;\ q = p^m \tag{5.8}$$

5.2. Galois Alanı

Sonlu alanların alt kümesi olan Galois alanları aşağıdaki ifadelerle tanımlanmaktadır.

5.2.1. GF(p)

p bir asal sayıyı ifade etmek üzere; eleman sayısı p olan $Z_p = \{0, 1, \ldots, (p - 1)\}$ kümesi üzerinde modulo p toplama ve modulo p çarpma işlemlerinin tanımlanmasıyla, p karakteristikli GF(p) şeklinde adlandırılan, p sayıda elemana sahip bir sonlu alan oluşur.

5.2.2. GF(p^m) – Genişletilmiş Alan

GF(p) alanı üzerinde p^m elemanlı yeni bir alan oluşturulabilmektedir. Bu durumda $q = p^m$ olmak üzere oluşturulan yeni alan GF(q), GF(p) alanının genişletilmiş alanıdır

denir ve GF(p^m) ile gösterilir. Bu gösterimde; p, yeni alanın karakteristiği, m ise alanın genişleme derecesi olarak ifade edilmektedir.

Yukarıdaki tanım sonucu oluşan GF(p^m) alanının bir Galois alanı olabilmesi için, m. dereceden bir indirgeme değerine ihtiyaç duyulmaktadır. Bunun sebebi, toplama ve çarpma işlemleri sonucunda elde edilecek yeni değerlerin sonlu elemanlar içerisinde bulunmama ihtimalidir. Elde edilecek yeni değer m. dereceden indirgeme değeriyle indirgenir ve bu değerin Galois alanını ifade eden kümenin bir elemanı olması sağlanır.

5.2.3. GF(2^m)

Yukarıda bahsedilen genişletilmiş alan tanımında p karakteristiğinin iki seçilmesiyle oluşturulmuş alanlardır. p karakteristiği iki seçilerek, matematiksel işlemlerin yazılımsal ve donanımsal olarak gerçekleştirilmesi ikilik tabanda çalışıldığından ötürü oldukça kolaylaştırılmış olur. Bu Galois alanları, ikili sonlu olarak ta anılmaktadırlar. GF(2^m) alanlarının elemanları, katsayıları sıfır ve birden oluşan polinomlardır.

5.2.4. Genişletilmiş Alanda Elemanların Gösterimi

Sonlu alanlarda, elemanların gösterimi için kullanılan en yaygın yöntemler polinomsal baz ve normal baz gösterimleridir.

5.2.4.1. Polinomsal Baz Gösterimi

f(x) polinomu denklem 5.9'u sağlamak üzere GF(2) alanında tanımlı m.dereceden bir polinom olsun. Eğer bu polinom aynı alan üzerinde tanımlanan diğer polinomların çarpımı şeklinde yazılamıyorsa *f(x)* polinomu indirgenemez bir polinomdur ve alanın indirgeme polinomu olarak isimlendirilmektedir.

$$f_i \in \{0,1\} olmak\ üzere;\ f(x) = x^m + \sum_{i=0}^{m-1}(f_i * x_i) \tag{5.9}$$

Bir a elemanının, $a_i \in \{0,1\}$ ve $\forall a \in GF(2^m)$ olmak üzere, polinomsal baz gösterimi denklem 5.10'da sunulmaktadır.

$$a = (a_{m-1}\ a_{m-2} \dots\ a_1 a_0) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \dots + a_1 x + a_0 \tag{5.10}$$

5.2.4.2. Normal Baz Gösterimi

$\beta \in GF(2^m)$ olmak üzere; $(\beta, \beta^2, \beta^4, \dots, \beta^{2m-1})$ kümesi, GF(2^m) alanının GF(2) üzerinde tanımlı normal bazıdır. Bir a elemanının $a_i \in \{0,1\}$ ve $\forall a \in GF(2^m)$ olmak üzere, normal baz gösterimi denklem 5.11'de sunulmaktadır.

$$a = (a_{m-1}\ a_{m-2} \dots\ a_1 a_0) = \sum_{i=0}^{m-1}(a_i * \beta^{2i}) \tag{5.11}$$

5.2.5. Galois Alanında Çarpma İşlemi

GF(p^m) alanında çarpma işlemi, GF(p^m) alanı üzerinde polinom çarpımının yapılması anlamına gelmektedir. Ancak; çarpım ortaya çıkacak polinomun derecesi sonlu alanın derecesinden yüksek olabileceği için daha önce bahsedilen indirgeme polinomu kullanılır. İki polinom çarpıldıktan sonra indirgeme polinomuna göre mod alınarak sonuç Galois alanında tanımlı bir polinoma indirgenir. Çarpma işlemi denklem 5.12'deki gibi ifade edilebilir.

$$c(x) = a(x)b(x) = \left(\sum_{i=0}^{m-1} a_i\, x^i\right)\left(\sum_{j=0}^{m-1} b_j\, x^j\right) = \sum_{i=0}^{m-1}\sum_{j=0}^{m-1}\left(a_i b_j \bmod p\right)x^{(i+j)} = \sum_{k=0}^{2m-2} c_k x^k \quad (5.12)$$

Galois alanında bir elemanın çarpmaya göre tersi denklem 5.13'te ifade edilmektedir.

$$a(x) * b(x) = 1\ (mod\ m(x)) \quad (5.13)$$

Yukarıdaki denklemde, a(x) ve b(x) GF(2^m) alanında tanımlı birer polinom ve m(x) bu alanda tanımlı indirgeme polinomu olmak üzere; a(x) ve b(x) polinomları birbirlerinin bu alanda tanımlanan çarpma işlemine göre tersidirler.

6. HIZLI FOURIER DÖNÜŞÜMÜ

Doğrusal dönüşümler, doğrusal sistemlerdeki problemlere çözüm sağlamasıyla bilinen tekniklerdir. Bu dönüşümler, doğrusal sistemlerdeki doğrudan çözülemeyen problemlerin üstesinden gelmek için adeta matematiksel veya fiziksel birer araç olarak kullanılırlar [13].

Fourier dönüşümü bilimin birçok alanında önemli bir rol oynamaktadır. Diğer dönüşümler gibi saf matematiksel işlev davranışı sergilediğinin kabul edilmesinin yanı sıra Fourier dönüşümünün fiziksel bir anlam ifade etmesi birçok alanda bu dönüşümü vazgeçilmez kılmaktadır. Optik, elektrik veya akustik bir dalga biçimi veya izgesi fiziksel olarak ölçülebilir ve gözlemlenebilir niceliklerdir. Örneğin; bir osiloskop yardımıyla elektrik dalga biçimi veya bir spektroskop yardımıyla optik veya elektrik izgeleri gözlemlenebilmektedir. Akustik açıdan bakıldığında ise insan kulağı gibi daha da doğrudan bir sistem göze çarpmaktadır. İnsan kulağı herhangi bir araca ihtiyaç duymadan izgeleri duyabilmektedir. Fourier dönüşümü fiziksel bir ilişki ifade etmektedir, öyle ki; dalga biçimi ve izge birbirlerinin Fourier dönüşümleridirler.

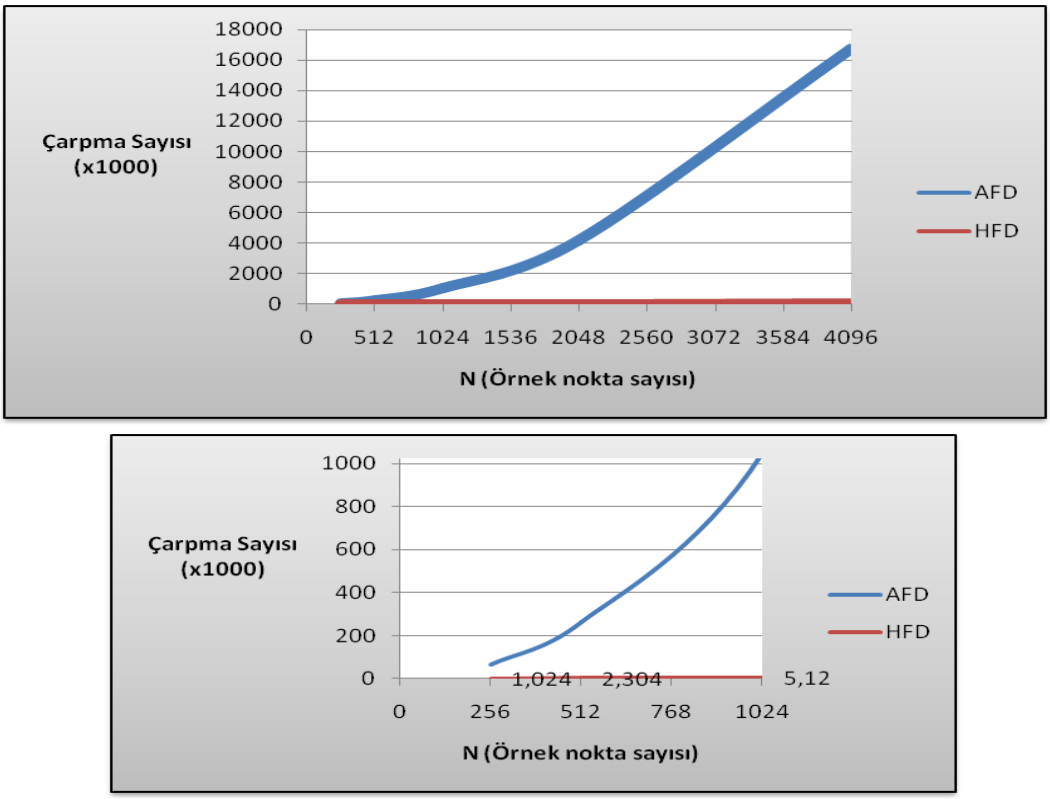
Ayrık Fourier dönüşümü kullanılarak bir sinyal biçiminin izgesi hesaplanırken 2Π periyodundaki ayrık zamanlı frekansın bir periyodunun hesaplamaya katılması sonuca ulaşmak için yeterli olacaktır. N adet frekans değeri için 2Π temel frekans bölgesinde N adet eşit aralıklı frekans değeri denklem 6.1 yardımıyla hesaplanabilmektedir.

$$\omega = \frac{2\pi}{N} k \qquad 0 \leq k < N \tag{6.1}$$

N adet ayrık frekans değeri için ayrık Fourier dönüşümü ise denklem 6.2'deki gibi hesaplanabilmektedir.

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j\omega n} \qquad 0 \leq k < N \tag{6.2}$$

Ayrık Fourier dönüşümünün yüksek karmaşıklığından ötürü (N^2) gerçek zamanlı uygulamalarda kullanılması oldukça zaman kayıplarına sebep olmaktadır. Bu sebeple hızlı Fourier dönüşümü geliştirilmiştir. Hızlı Fourier dönüşümü daha düşük işlem yükü sayesinde sinyal işleme alanında yoğun olarak kullanılmaktadır. Ayrık Fourier dönüşümü ve hızlı Fourier dönüşümünün çarpma sayılarının karşılaştırılması Şekil 6.1'de sunulmaktadır.



Şekil 6.1 Ayrık Fourier Dönüşümü ve Hızlı Fourier Dönüşümü için Gereken Çarpma Sayılarının Karşılaştırılması

Ayrık Fourier dönüşümünde katsayılar N nokta için denklem 6.3'teki gibi ifade edilebilmektedir.

$$W_N = e^{-j(2\pi/N)} \tag{6.3}$$

Faz faktörü, N'ye göre simetrik ve N/2'ye göre periyodik olduğundan denklem 6.4 ve denklem 6.5 yazılabilir.

$$W_N^{k+N/2} = -W_N^k \tag{6.4}$$

$$W_N^{k+N} = W_N^k \tag{6.5}$$

Hızlı Fourier dönüşümünde bu özelliklerden faydalanılarak ayrık Fourier dönüşümüne göre çok daha hızlı hesap yapılabilmektedir. Bu aşamadan sonra N noktalı bir Fourier dönüşümü için denklem 6.6 yazılabilmektedir.

$$x[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \tag{6.6}$$

Girdi sinyali tek (2n + 1) ve çift (2n) indisli örneklere ayrılarak dönüşüm denklem 6.7'deki gibi ifade edilebilir.

$$x[k] = \sum_{n=0}^{\left(\frac{N}{2}\right)-1} x[2n] W_N^{k(2n)} + \sum_{n=0}^{\left(\frac{N}{2}\right)-1} x[2n+1] W_N^{k(2n+1)} \tag{6.7}$$

W_N^k, giriş sinyaline bağlı olmadığından toplamın dışına çıkarılabilir. Ayrıca $W_N^{k(2n)}$, $W_{N/2}^k$'ye eşit olduğundan dolayı katsayılar N/2 noktalı dönüşüme göre düzenlenir ve x[2n] yerine $x_1[2n]$, x[2n + 1] yerine de $x_2[n]$ yazılırsa denklem 6.8 elde edilir.

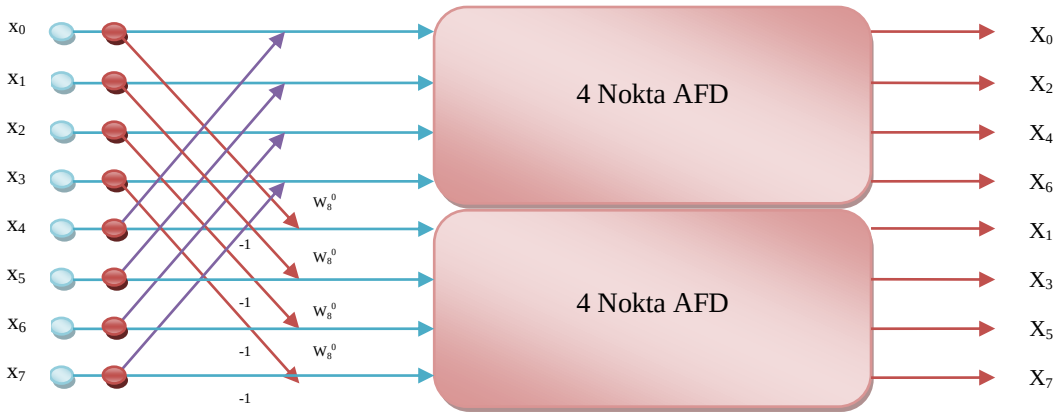
$$x[k] = \sum_{n=0}^{\left(\frac{N}{2}\right)-1} x_1[n] W_{N/2}^{kn} + \sum_{n=0}^{\left(\frac{N}{2}\right)-1} x_2[n] W_{N/2}^{kn} \tag{6.8}$$

Denklem 6.4'teki simetri özelliği kullanılarak ilk N/2 girdi için ve son N/2 girdi için dönüşüm denklemleri sırasıyla denklem 6.9 ve 6.10'daki gibi elde edilir.

$$x[k] = x_1[k] + W_N^k x_2[k] \tag{6.9}$$

$$x[k] = x_1[k] - W_N^k x_2[k] \tag{6.10}$$

Yukarıdaki eşitlikler yardımıyla N noktalı bir dönüşüm zincirleme olarak indirgenerek daha az işlem yapılmak suretiyle çözüme ulaşılır. Örneğin 8 nokta AFD önce 2 tane 4 noktalı AFD'ye indirgenir daha sonra 4 noktalı dönüşümler 2 noktaya indirgenir ve devam edilir [14]. Şekil 6.2'de örnek bir 8 nokta AFD'nin 2 adet 4 noktalı AFD'ye indirgenmesi sunulmaktadır.



Şekil 6.2 8 Nokta AFD'nin 2 adet 4 Nokta AFD'ye İndirgenmesi

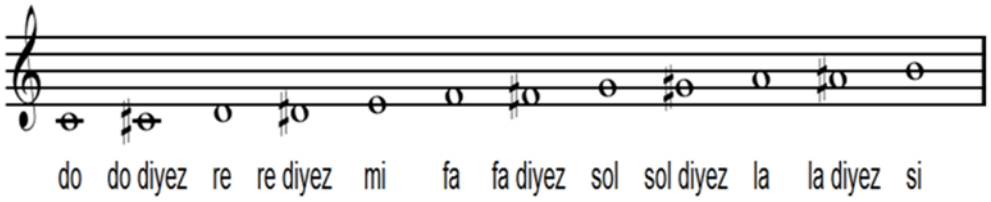
7. MÜZİKAL ÖN BİLGİ

Bu bölümde, çalışmada kullanılan bazı müzikal kavramlar hakkında bilgiler sunulmaktadır [15].

7.1. Perde, Frekans, Nota ve Oktav Kavramları

Bir müzik terimi olan perde kavramı, bilimsel bir kavram olan frekansa çok yakın bir anlama sahiptir. Perde, birim zamanda duyulan sabit frekanslı tek bir ses anlamında kullanılmaktadır.

Nota ise iki farklı şekilde ifade edilebilmektedir. Tek bir perde üzerinde duyulan bir ses veya tek bir sesi ifade eden sembol notanın iki farklı kavram olarak ifade edilişidir. Batı müziğinde arızalı sesler (diyez ve bemol) dahil kullanılan 12 ses bulunmaktadır. Bunlar; do, do diyez (re bemol), re, re diyez (mi bemol), mi, fa, fa diyez (sol bemol), sol, sol diyez (la bemol), la, la diyez (si bemol) ve si'dir. Yukarıda yazılı her ardışık sesin arası yarım ses olarak tanımlanmaktadır. Örneğin; re sesi yarım ses inceltildiğinde re diyez sesi elde edilir. Şekil 7.1'de notaların müzik yazımındaki gösterimleri sunulmaktadır.



Şekil 7.1 Notaların Müzik Yazımındaki Gösterimleri

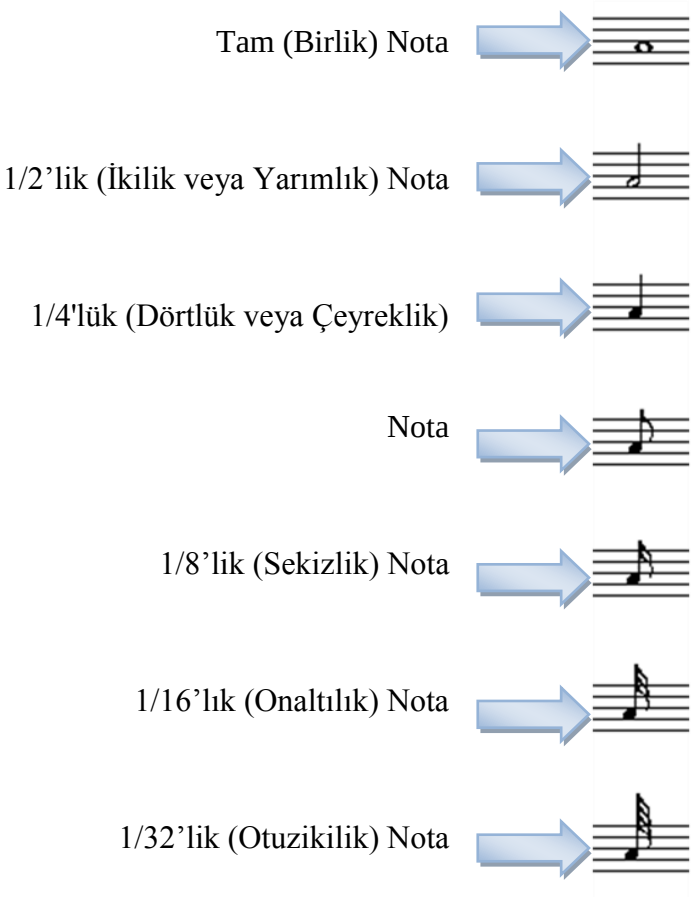
Müzikteki seslerin her birinin birer frekans değeri vardır. Bir sesin frekans değeriyle, o değerin iki katı veya yarısı arasına bir oktav denir. Örneğin; genellikle hesaplamalarda bağıl eleman olarak kullanılan dördüncü oktav la sesinin frekansı 440 Hz'dir. Buradan yola çıkarak, herhangi bir oktavdaki notanın frekansı denklem 7.1 yardımıyla hesaplanabilmektedir.

$$f = 2^{n/12} \, x \, 440 \, Hz \tag{7.1}$$

Denklem 7.1'de verilen eşitlikte "*f*" frekansı "*n*" ise frekansı hesaplanmak istenen nota ile dördüncü oktavdaki la (La4) notası arasındaki yarım seslerin sayısıdır. Eğer frekansı hesaplanmak istenen nota La4'ün üzerindeyse (La4'ten inceyse) n pozitif, altındaysa (La4'ten kalınsa) n negatiftir.

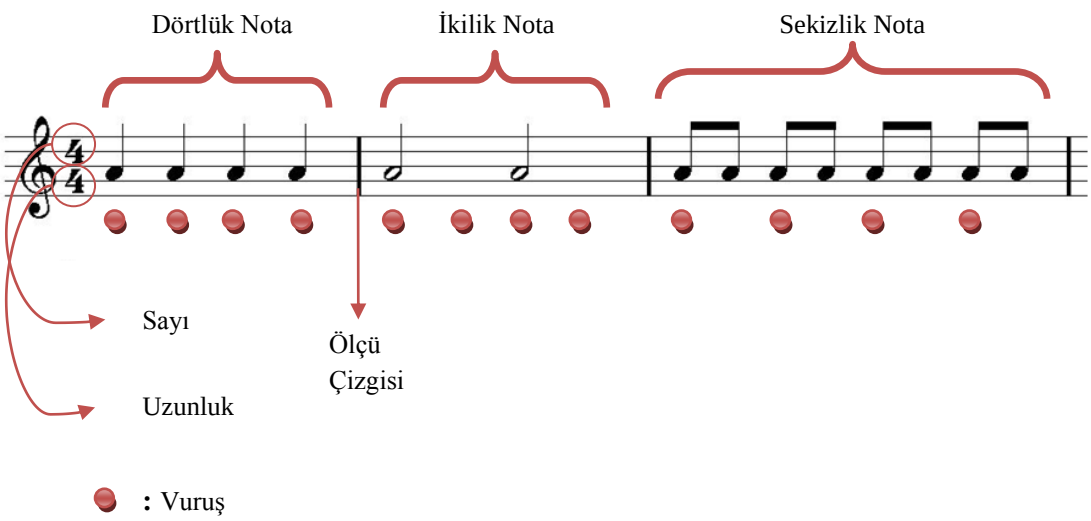
7.2. Uzunluk, Süre, Tempo ve Ölçü Kavramları

Müzik yazımında her bir notanın bir uzunluk değeri vardır. Uzunluk değeri tek başına tam olarak zaman ifade etmez. Dakikadaki vuruş sayısına göre farklı zaman karşılıkları vardır. Müzikte, dakikadaki vuruş sayısı tempo olarak adlandırılır. Uzunluk değeri ise; bir notanın kaç vuruş sürdüğünü ifade etmektedir. Müzikte, notaların uzunluk değerleri ve gösterimleri şekil 7.2'de sunulmaktadır.



Şekil 7.2 Notaların Uzunluk Değerleri ve Gösterimleri

Bir müzik eseri, ölçü adı verilen eşit uzunlukta parçalara bölünmüştür. Tüm bu parçalar içerisindeki vuruş sayıları eşittir. Başka bir deyişle, her ölçüdeki nota uzunluklarının ve boşluk (sus) uzunluklarının toplamları eşittir. Süre ise; bir ölçünün uzunluğunu belirleyen değişkendir. Süre, bir ölçüde hangi uzunlukta kaç tane nota veya sus olduğunu, başka bir deyişle bir ölçüde kaç adet vuruş olduğunu ifade eder. Şekil 7.3'te ölçü, vuruş, uzunluk ve süre kavramları gösterilmektedir.



Şekil 7.3 Müzikte Ölçü, Vuruş, Uzunluk ve Süre Kavramları

Şekil 7.3'te sayı olarak açıklanan, notaların en başına eklenen sayı, bir ölçü içerisinde, altında yazan sayıdaki uzunlukta notalardan veya suslardan kaç tane olması gerektiğini söylemektedir. Burada taneden kasıt, tam olarak nota sayısı değildir. Örneğin; yukarıdaki şekilde bir ölçüde dört adet dörtlük nota olduğu ifade edilmektedir. Bu ifade, dört adet dörtlük nota süresine karşılık gelmektedir. Bu süre farklı şekillerde sağlanabilmektedir. Şekilde ilk ölçüde dört adet dörtlük nota bulunurken ikinci ölçüde iki adet ikilik nota, üçüncü ölçüde ise sekiz adet sekizlik nota bulunmaktadır. Asıl anlatılmak istenen ölçü içerisindeki nota sayısı değil toplam nota uzunluğudur. Örneğin bir adet ikilik nota iki adet dörtlük notaya ve aynı zamanda dört adet dörtlük notaya eşit uzunluktadır. Yine aynı şekil üzerinde kırmızı nokta ile gösterilen yerler vuruşların başlangıcıdır. Her ölçüdeki sürelerin eşit olduğu buradan da gözlemlenebilir. Bir dörtlük nota başına bir vuruş geçerken, bir ikilik nota başına iki vuruş geçmektedir. Örneğin yukarıdaki şekilde temponun 120 olduğu var sayılırsa; bir dakikadaki vuruş sayısı 120 olacaktır. O halde; 1 vuruş 0.5 saniyeyi ifade edecektir. Dörtlük nota 0.5 saniye sürerken ikilik nota bir saniye, sekizlik nota ise 0.25 saniye sürecek ve her ölçüdeki süreler toplamı ayrı ayrı iki saniyeye eşit olacaktır.

8. VERİ GİZLEME

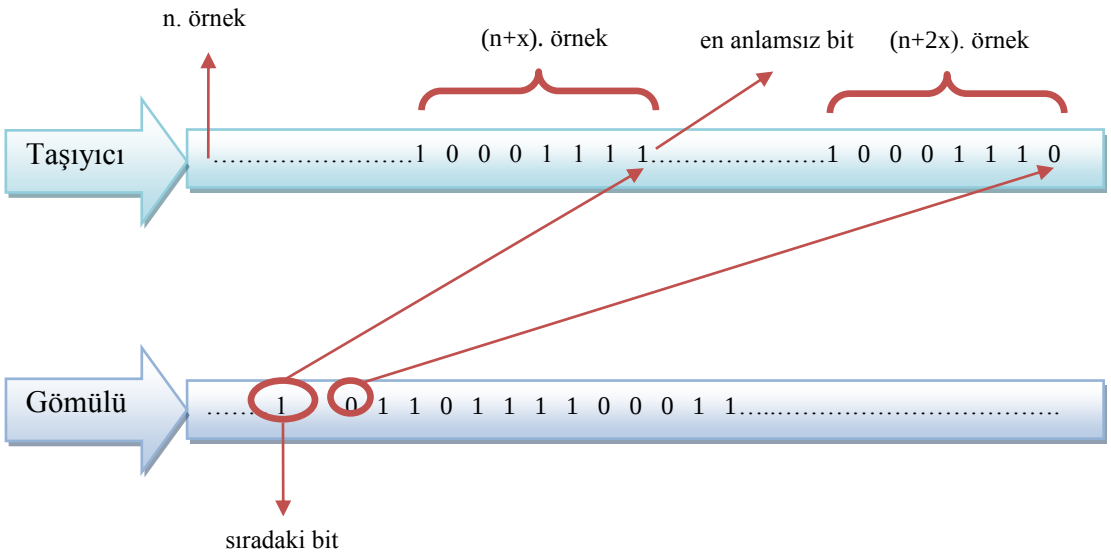
Veri gizleme, kullanılan ismiyle steganografi, ses, resim, video veya uygun bir biçimdeki sayısal dosyanın içerisine veri gizlemek için kullanılan bir bilgi güvenliği yöntemidir. Başka bir deyişle steganografi çok eski bir, görünüş olarak zararsız bir mesajın içerisine gizli bir mesajı gömme sanatıdır. Üstelik bu sanat, gizli mesajı karıştırmak için kriptografide kullanılan şifreleyicileri, kodları v.s. içermemektedir [16]. Steganografideki temel amaç bilgi güvenliğinin sağlanmasıdır. Steganografide bilgi güvenliğinin sağlanması için elde bulunan en büyük koz, kaynaktan hedefe veri aktarımı yapılırken, bu aktarımda gizli bir veri olduğunun anlaşılmamasıdır. Şüphesiz ki steganografiyi kriptografi ile birlikte kullanmak, yani gizlenen verinin açık veri olmayıp şifrelenmiş veri olması, bilgi güvenliğini bir seviye daha arttıran bir etmen olmaktadır.

Veri gizleme sistemlerinde, gizlenecek veriyi saklamak için kullanılan dosyaya taşıyıcı, gizlenecek veriye gömülü, gizleme işlemi sonucunda ortaya çıkan dosyaya ise stego isimleri verilmektedir. Bu yapının şekilsel gösterimi, şekil 8.1'de sunulmaktadır.

Şekil 8.1 Veri Gizleme Sistemi

Veri gizleme sistemlerinde esas; stegonun şüphe çekmemesi ve şüphe çektiği takdirde içerisinde gizli veri taşındığının anlaşılmasının zorluğunun en yüksek seviyeye taşınmasıdır [17]. Pooyan ve Delforouzi, yapmış oldukları çalışmada şüpheyi en düşük seviyeye indirmek adına, taşıyıcı ses dosyasındaki en anlamsız bitleri saptayarak gizli verilerin bu anlamsız bitlere gömülmesinin daha iyi olacağını

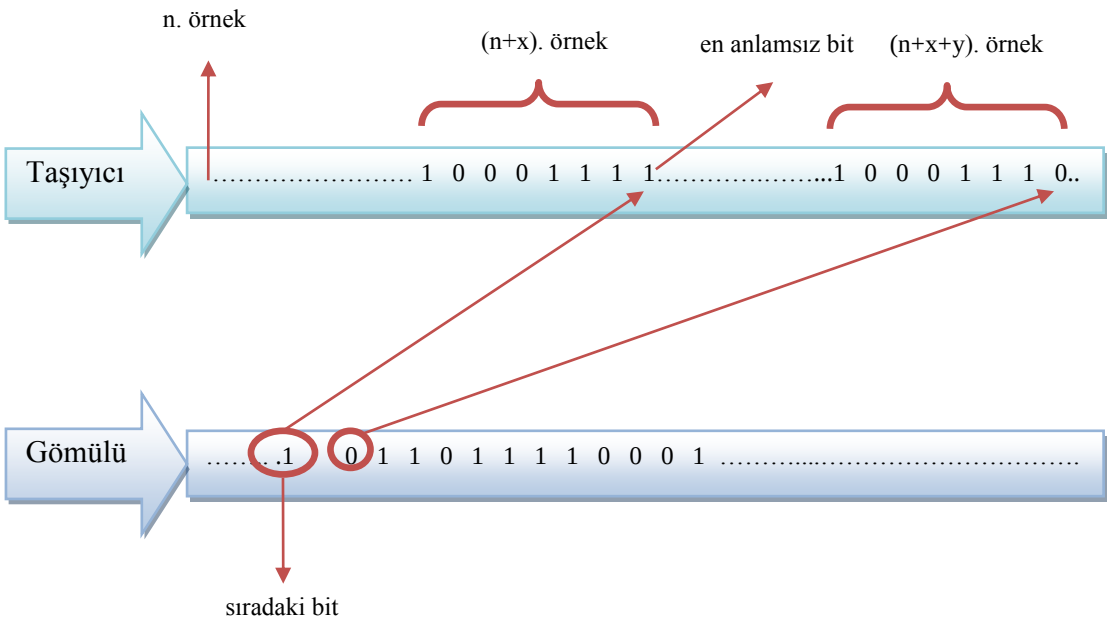
belirtmişlerdir [17]. Gopalan ise yapmış olduğu çalışmada, ses dosyasının her örneğindeki bir biti, gizleyeceği verinin sıradaki bitiyle değiştirmiştir [18]. Taşıyıcı dosyanın en anlamsız bitlerinin değiştirilmesi yaklaşımı, veri gizlemede yoğun olarak kullanılan yaklaşımlardan bir tanesidir. Örneğin bir ses dosyasında örneklerin belli aralıklarla en anlamsız bitlerinin değişmesi, uygun örnek seçimi yapıldığında insan kulağının algılama yapısı dahilinde dinleyici için hiçbir fark oluşturmamaktadır. Uygun örneklerin seçimi kimi zaman sabit olarak kimi zaman ise rastgele olarak yapılmaktadır. Sabit yaklaşımda; gizli mesajın her biti, sabit sayıda örnek atlanarak ulaşılan örneğin en anlamsız biti ile değiştirilmekte, oluşan verinin değiştirilen bitleri ilk halinden ortalama %50 farklılık göstermektedir[19]. Bu yöntemin kötü yönü, sabit sayıda veri atlandığında belli bir örüntüyü izlemesi ve uygulanacak steganaliz yöntemlerine karşı diğer yönteme kıyasla daha dayanıksız olmasıdır. Sabit örnek seçiminin şekilsel gösterimi şekil 8.2'de sunulmaktadır.



Şekil 8.2 Sabit Örnek Atlamalı Sese Veri Gizleme Sistemi

Rastgele örnek atlamalı sistemde ise atlanacak örnek sayısı her seferinde farklıdır. Kimi zaman bir kurala göre seçilir kimi zaman ise rastgele seçilmektedir. Sabit bir örüntü korunmadığı için steganalize karşı daha dayanıklıdır. Ayrıca bu yöntemde rastgelelik belirli bir kurala göre yapılarak taşıyıcı dosyadaki en anlamsız bitlerin gömülecek verideki bitlerle aynı olmaları büyük ölçüde sağlanarak taşıyıcı dosya

üzerinde çok az değişiklik yapılabilmektedir. Bu sistemin şekilsel gösterimi şekil 8.3'te sunulmaktadır.



Şekil 8.3 Rastgele Örnek Atlamalı Sese Veri Gizleme Sistemi

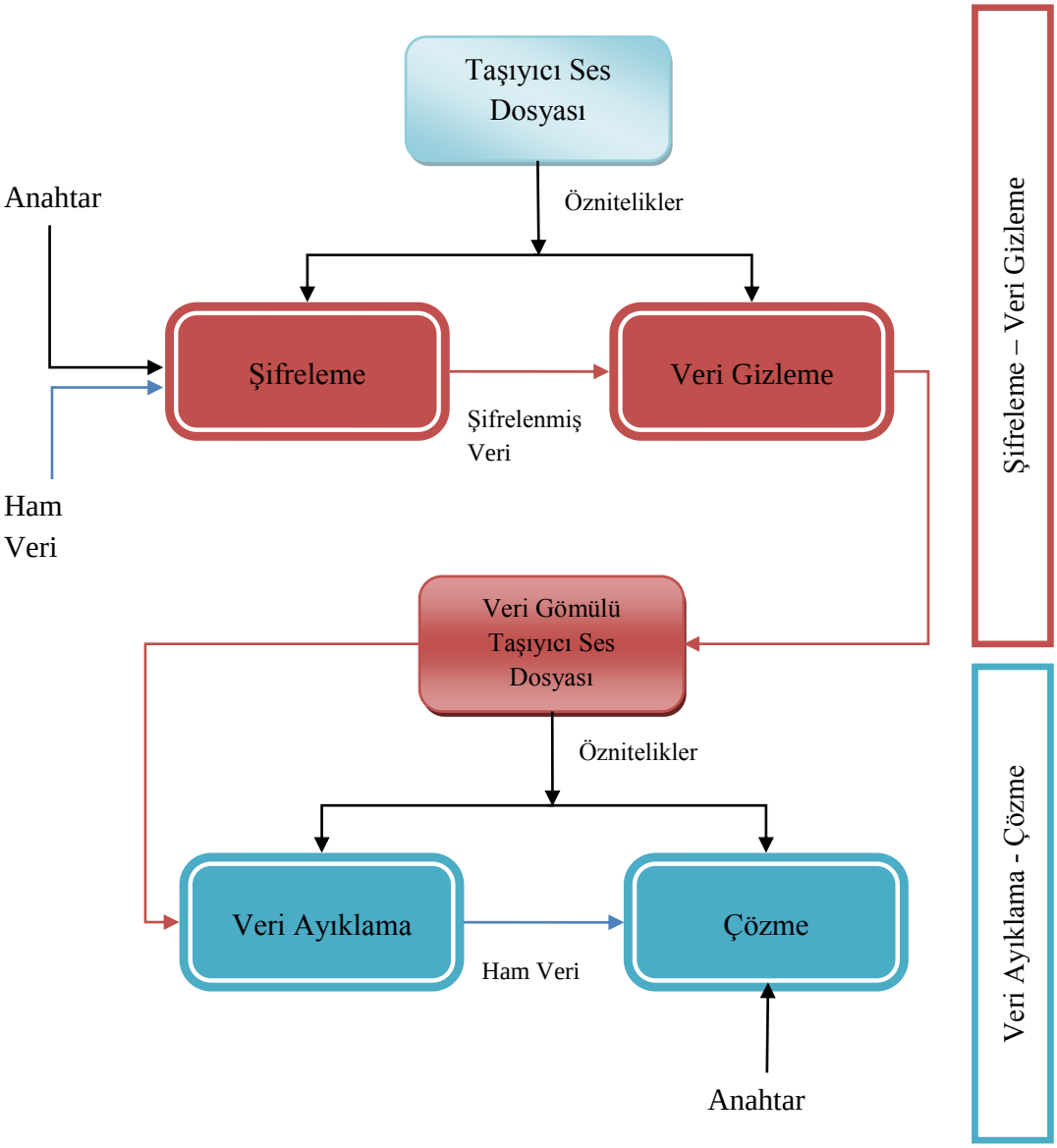
Bu yöntemin en kötü yönü ise; atlanacak örnek sayıları veriyi gömme aşamasında belirlendiği için, veriyi gönderen kaynak ile veriyi alan hedef arasında kimi zaman çok yüksek boyutları bulabilecek bir veri aktarımının gerekli olmasıdır. Hedef, verinin hangi bitlere saklandığını bilmediği için şekil 8.3'te gösterilen x ve y değerleri ve bunları takip eden diğer atlama değerleri güvenli bir kanalda kaynaktan hedefe aktarılmalıdır.

Anlamsız bitleri kullanarak (düşük bit kodlaması) veri gizleme yönteminin haricinde kullanılan faz kodlaması, yayılmış izge ve yankı veri gizleme yöntemleri de ses içerisine veri gizlemede yaygın olarak kullanılmaktadır. Faz kodlamasında ses dosyası bölütlenmekte ve ardından her bölüme ait faz değeri içerisine veri gizlenmekte yani faz değerleri yeniden oluşturulmaktadır. Yayılmış izge yönteminde ise; sesin frekans izgesi elde edilip, bu izge üzerine veri gömülmektedir. Bu yöntem,

gürültüye karşı zayıf bir yöntemdir. Son olarak, yankı veri gizleme yönteminde ise sesin üzerine yankı eklenmekte ve eklenen yankının gecikme değerleri gömülü veriyi ifade etmektedir. Bu yöntem ise bilinen bir yöntem olduğu için steganografinin şüphe çekmeme ilkesiyle kimi zaman ters düşmektedir.

9. TASARIM ve GERÇEKLEME

Gelişen teknolojiyle birlikte bilgi güvenliği ihtiyacı günden güne artmakta olup, tüm dünyada bu alanda çalışmalar yapılmaktadır. Güvenliği sağlanmak istenen bilgiler kişisel bilgiden ziyade toplumları etkileyecek nitelikte ve kaynaktan hedefe çok kısa zamanda iletilmesi gereken bilgiler olduğu düşünüldüğünde, bu bilgilerin güvensiz bir ortamda aktarılması kaçınılmaz olmaktadır. İşte bu durumda, bilginin güvenliğini sağlamak adına bilgiler şifrelenmekte veya gizlenmektedir. Kimi zaman yapılan çalışmalarda bu iki sistem birlikte kullanılmaktadır. Yani; güvenliği sağlanmak istenen bilgi önce şifrelenmekte daha sonra ise sayısal bir taşıyıcı dosya içerisine belli yöntemlerle gizlenerek iletim yapılmaktadır. Bu durumda; bilgiyi riske atmamak için şüphesiz ki çok güçlü bir şifreleme algoritmasının ardından, içerisinde veri taşındığı anlaşılmayan veya anlaşılsa dahi verinin ayıklanmasının olabildiğince zorlaştıran sistemler tercih edilmektedir. Taşıyıcı dosyada yeterli güvenilirliği sağlamak için yapılan çalışmalarda, çoğunlukla aktarım öncesinde bir bilgi paylaşımına ihtiyaç duyulmaktadır. Bu çalışmada öncelikle, var olan AES algoritmasına ek yapılmış ve bu ek şifreleme algoritmasında taşıyıcı olarak kullanılan ses dosyasından türetilen bazı veriler girdi olarak kullanılmıştır. Daha sonra ise, aktarım öncesi veri paylaşımını ortadan kaldırmak ve veri gizleme sistemini güçlendirmek adına taşıyıcı ses dosyasında çalan müziğin belirli notaları anahtar olarak kullanılmıştır. En üst seviyeden bakıldığında tasarlanan şifreleme-veri gizleme ve veri ayıklama-çözme sistemleri Şekil 9.1'de sunulmaktadır.



Şekil 9.1 Müzikle Şifreleme Sistemi Üst Seviye Görünümü

Şekil 9.1'deki sistemi gerçekleştirmek için yapılan tasarımlar ve gerekli donanımsal ve yazılımsal tüm alt sistemlerin gerçeklenmesi ile ilgili detaylar bu bölümde sunulmaktadır.

9.1. AES-256 Algoritmasının Gerçeklenmesi

Müzikle şifreleme ve veri gizleme sisteminin yüksek güvenilirliğe sahip olabilmesi için sistemin şifreleme ve çözme aşamalarında güvenilirliği günümüzde kabul edilmekte olan AES-256 algoritması kullanılmaktadır. Sistemdeki tüm şifreleme işlemi AES-256 ile yapılmamaktadır. AES-256 yalnızca şifreleme aşamasının başında ve çözme aşamasının sonunda kullanılmaktadır. AES-256 algoritması, yüksek hızda çalışabilmesi için donanımsal olarak FPGA üzerinde gerçeklenmiştir. Gerçeklenme aşamasında kullanılan FPGA geliştirme kartı ile ilgili bilgiler bölüm 9.1.1'de sunulmaktadır.

9.1.1. Gerçekleştirmenin Yapıldığı FPGA Geliştirme Kartı

AES-256 algoritmasının donanımsal gerçeklemesi için Altera Bemicro FPGA geliştirme kartı kullanılmıştır. Bu kartın seçilmesinin sebepleri; kartın yeterli donanıma sahip olup ucuz olması ve boyutlarının oldukça küçük olmasından kaynaklanan kullanım kolaylığıdır. Şekil 9.2'de bu geliştirme kartının resmi sunulmaktadır.

Şekil 9.2 Altera Bemicro FPGA Geliştirme Kartı

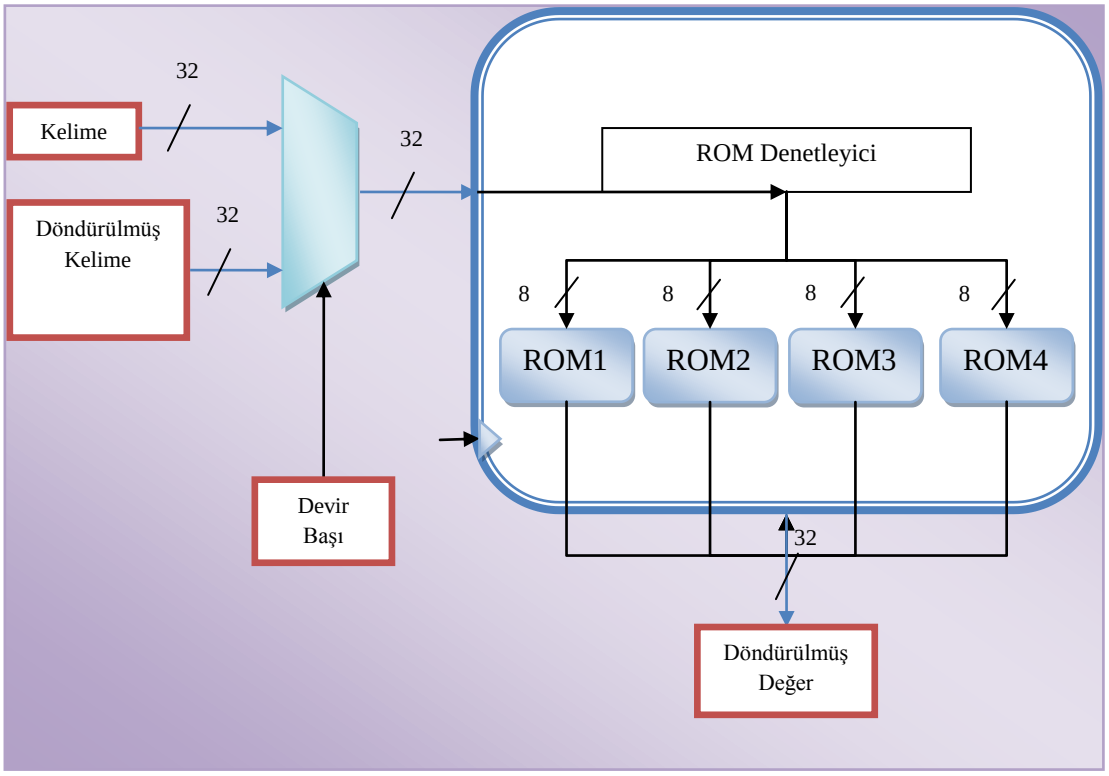
Altera Bemicro FPGA geliştirme kartının özellikleri aşağıdaki gibidir [20]:

- Cyclone III EP3C16F256C8N FPGA
- 4 MB SRAM
- 16 MHz Osilatör
- 3 adet durum belirten led
- 8 adet kullanıcı kontrollü genel amaçlı led
- 80 genel amaçlı iğneli bağlantı elemanı

9.1.2. Anahtar Genişletme Aşamasının Gerçeklenmesi

Bölüm 4.1'de anlatılan anahtar genişletme işleminin gerçeklenmesi için tasarlanan durum makinesi, kaçıncı anahtar kelimesinin (32 bit) türetildiğine göre durumlar arasında geçiş yapmaktadır. Bölüm 4.1'de üç temel işlemden bahsedilmektedir. Bunlar; kelime döndürme, kelime değiştirme ve devir sabiti ekleme işlemleridir. Algoritmaya göre, kelime döndürme ve devir sabiti ekleme işlemleri yalnızca devir başlarında, kelime değiştirme işlemi ise devir başlarında ve devir ortalarında yapılmaktadır. Kelime döndürme işlemi için fazladan bir tasarıma ihtiyaç duyulmamaktadır. Gerekli değer, ilgili değerin bitlerini farklı sırada işleme tabi tutmaktan ibarettir. Devir sabiti , yedi adet birer baytlık değerden ve bu değerler birbirleriyle ilişkili olduğu için 8 bitlik bir yazmaç devir sabitini tutmak için kullanılmış ve başlangıç değeri olarak ilk devir sabiti olan {01} baytı yazmaca kaydedilir. Daha sonraki devirlerde bu yazmaç, sağa veya sola kaydırılarak devir sabitleri elde edilmektedir. Algoritmada kelime değiştirme işlemi için S-kutusu kullanılmaktadır. S-kutusunun gerçeklenmesi ROM kullanılarak taramalı çizelge usulüyle yapılmıştır. ROM, saate bağlı çalıştığı ve taramalı çizelgede birer baytlık değerler tutulduğu için, saat vuruşu kaybetmemek adına 4 adet ROM paralel olarak çalışması için tasarlanmıştır. Kelime dönüşümü dört baytlık değerlere uygulanacağından, ROMlar bir çatı altına toplanarak 4 baytlık adres girişinin her baytı paralel olarak adres değeri olarak kullanılmış ve dört romdan alınan çıktılar bir saat vuruşunda birleştirilerek 32 bitlik veri yoluyla dışarıya verilmiştir. Algoritmaya

göre kelime dönüşümü yapılacak değer devir başlarında ve devir ortalarında değişmektedir. Bu sebeple, adres girişinin önüne bir adet çoklayıcı koyularak seç girişine devirin başında mı, yoksa ortasında mı olunduğunu ifade eden bir girdi yapılmaktadır. Anahtar genişletme aşamasının kelime değiştirme dönüşümünü ifade eden gösterim şekil 9.3'te sunulmaktadır.

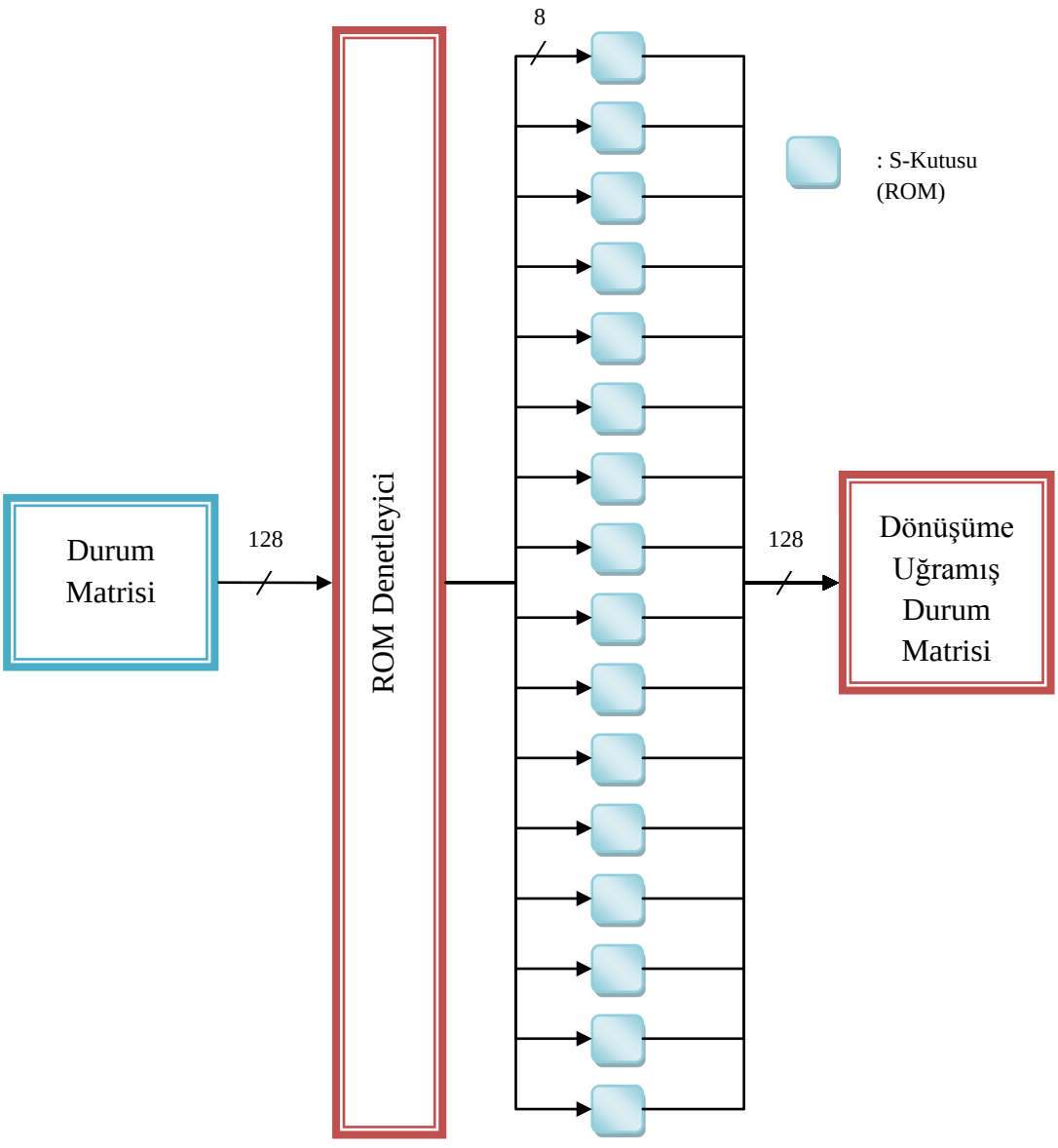


Şekil 9.3 Kelime Değiştirme Dönüşümünün Gerçeklenmesi

9.1.3. Bayt Değiştirme Dönüşümünün Gerçeklenmesi

Dördüncü bölümde anlatılan bayt değiştirme dönüşümü saat vuruşundan kayıp olmaması adına taramalı çizelge yöntemiyle gerçeklenmiştir. 128 bitlik durum matrisinin her baytının ayrı ayrı dönüşüm geçirmesi için tek bir ROM kullanılırsa 16 saat vuruşu beklemek gerekmektedir. Ancak; bu çalışmada alan veya güç tüketimi bir kısıt arz etmediğinden, tek kısıt çalışma zamanı olduğu için 16 adet S-kutusu yani

16 adet ROM paralel olarak gerçeklenmiş ve 128 bitlik durum matrisinin tamamının dönüşümünün bir saat vuruşunda gerçekleşmesi sağlanmıştır. Şekil 9.4'te bayt değiştirme dönüşümünün gerçeklenmesi gösterilmektedir.



Şekil 9.4 Bayt Değiştirme Dönüşümünün Gerçeklenmesi

9.1.4. Satır Kaydırma Dönüşümünün Gerçeklenmesi

Satır kaydırma dönüşümünde durum matrisinin hiçbir baytı farklı bir bayta dönüşmez, ancak matrisin her satırı kendi içerisinde dairesel kaydırma işlemine tabi tutulur. Satır kaydırma dönüşümü her 128 bit için bir saat vuruşunda gerçekleştirilmektedir. Bayt değiştirme dönüşümünün çıktısını girdi olarak alır ve çıktısı olan yeni durum matrisi sütun karıştırma dönüşümüne gönderilir.

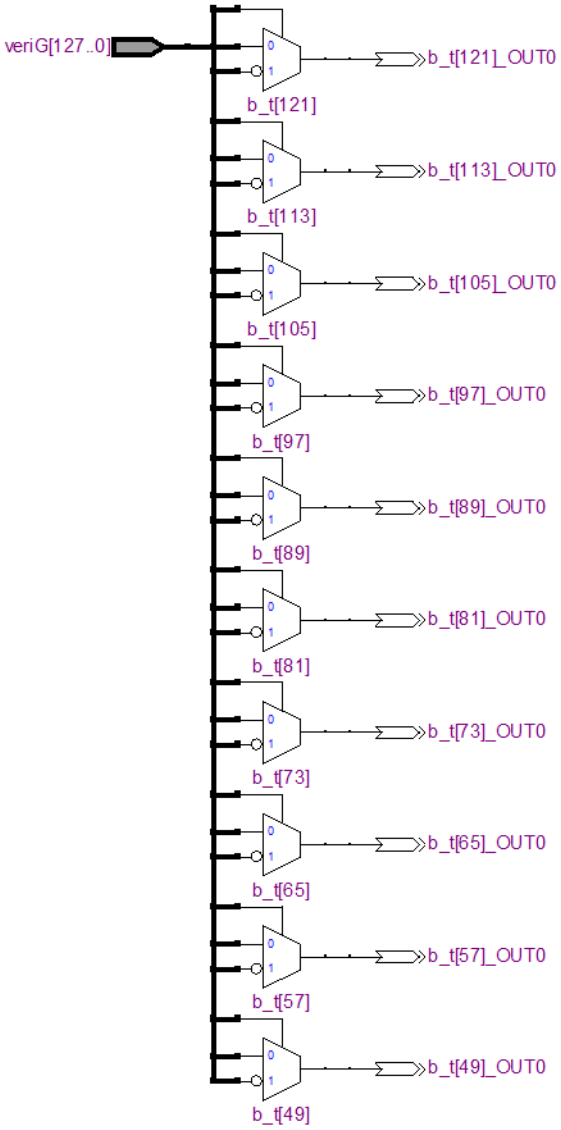
9.1.5. Sütun Karıştırma Dönüşümünün Gerçeklenmesi

Sütun karıştırma dönüşümü saatten bağımsız olarak gerçeklenmiştir. Sütun karıştırma modülünün girdisi ve çıktısı 128 bit olarak ayarlanmıştır. Yani durum matrisine sütun karıştırma dönüşümü paralel olarak uygulanmaktadır. Gerçekleme için 128 bitlik yardımcı bir tel dizisi kullanılmıştır. Girdi verisi "veriG", yardımcı tel "b_t" şeklinde ifade edilerek öncelikle denklem 9.1'de verilen işlem uygulanmıştır.

$$b_{t[n:n-7]} = \begin{cases} \{veriG[n-1:n-7],0\} \oplus \{1B\}, & veriG[127] = 0 \\ \{veriG[n-1:n-7],0\}, & veriG[127] \neq 0 \end{cases}$$

$$n = \{7, 15, 23, \dots, 127\} \tag{9.1}$$

Burada gelen verinin her baytının en üst bitine bakılır. Bir bayt için, eğer en üst bit bire eşit ise yardımcı telin ilgili baytına gelen baytın altıncı ve sıfırıncı bitleri alınarak sonuna sıfır eklenir ve {1B} onaltılık değeriyle özel veya işlemine tabi tutularak yazılır. Bu işlemin şeması şekil 9.5'te sunulmaktadır.



Şekil 9.5 Sütun Karıştırma Devresinden Bir Kesit

Bu işlemin ardından denklem 9.2'deki eşitlik dizisinde olduğu gibi her sütun için hesaplanan değer sütun karıştırma dönüşümünün çıktısıdır.

$$çıktı[n:n-7] = b_{t[n:n-7]} \oplus veriG[n-24:n-31] \oplus veriG[n-16:n-23] \oplus b_{t[n-8:n-15]} \oplus veriG[n-8:n-15]$$

$$çıktı[n-8:n-15] = b_{t[n-8:n-15]} \oplus veriG[n:n-7] \oplus veriG[n-24:n-31] \oplus b_{t[n-16:n-23]} \oplus veriG[n-16:n-23]$$

$$çıktı[n-16:n-23] = b_{t[n-16:n-23]} \oplus veriG[n-8:n-15] \oplus veriG[n:n-7] \oplus b_{t[n-24:n-31]} \oplus veriG[n-24:n-31]$$

$$çıktı[n-24:n-31] = b_{t[n-24:n-31]} \oplus veriG[n-16:n-23] \oplus veriG[n-8:n-15] \oplus b_{t[n:n-7]} \oplus veriG[n:n-7]$$

$$n = \{31, 63, 95, 127\} \quad (9.2)$$

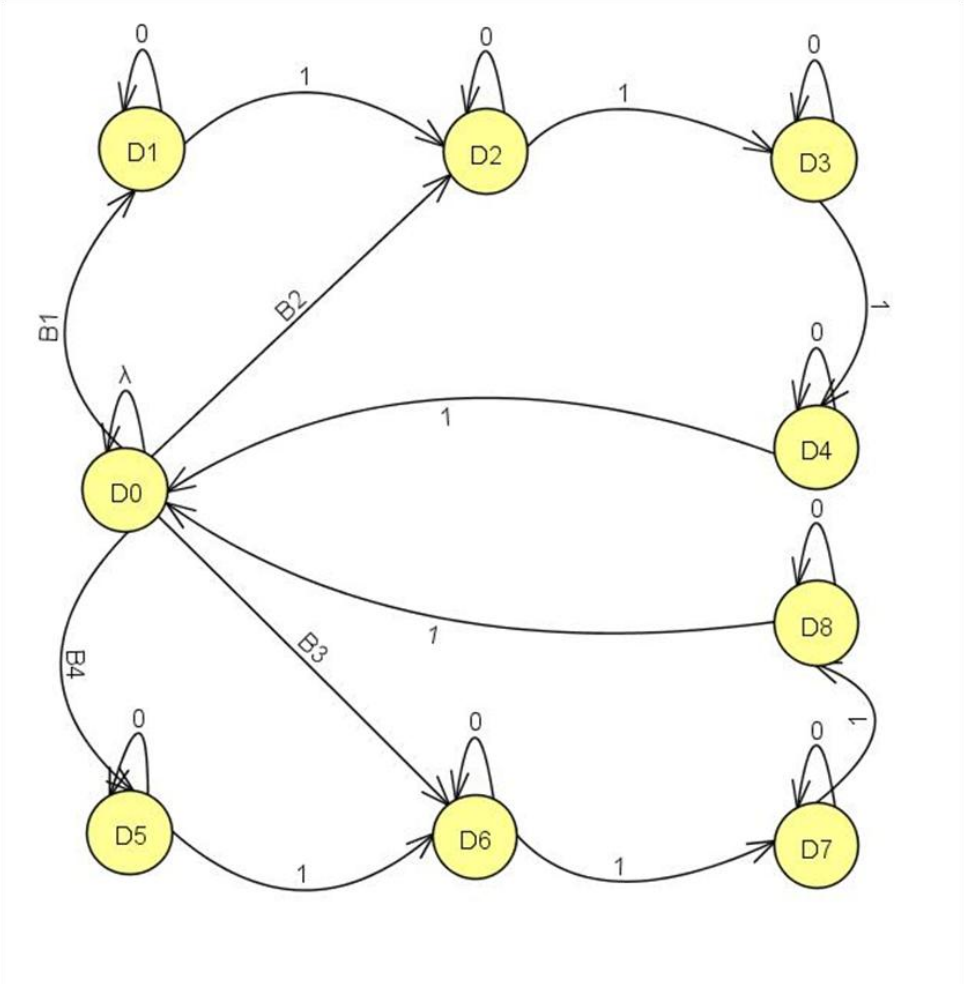
9.1.6. Çözme Algoritmasının Gerçeklenmesi

AES-256 algoritmasının çözümü, şifrelenirken yapılan işlemlerin tersinden ibarettir. Çözme algoritması gerçeklenirken sütun karıştırma dönüşümü hariç tüm işlemler aynı mantıkla ele alınmıştır. Çözme algoritmasının sütun karıştırma dönüşümünde çarpanlar değiştiği için, gerçekleme saat vuruşuna bağlı olarak yapılmıştır. Her çarpan için olası tüm sonuçlar taramalı çizelge yöntemiyle gerçeklenir ve 128 bitlik durum matrisi için dönüşüm bir saat vuruşunda gerçekleştirilir.

9.1.7. Sistemin Birleştirilmesi ve Kullanımı

Gerçeklenen AES-256 şifreleme ve çözme sistemi bir bilgisayarla birlikte kullanılması için tasarlanmıştır. Bilgisayar ile seri kapı yardımıyla veri iletişimi gerçekleştirmektedir. Tüm sistem birleştirildiğinde, farklı seçeneklerin kullanılabilmesi için bir durum makinesi oluşturulmuştur. Kullanıcının donanımın ne yapacağını belirlemesi için donanımda kullanılmak üzere dört adet buyruk geliştirilmiştir. Bu dört seçenek, anahtar değeri değiştirerek şifreleme, son girilen anahtarla şifreleme, anahtar değeri girilerek çözme ve son anahtar değeriyle çözmedir. Seri kapı iletişim modülü, bilgisayardan gönderilecek buyrukları sürekli

dinlemekte ve aldığı verilere göre sistemi uygun şekilde başlatmaktadır. Şekil 9.6'da AES-256 sisteminin durum makinesi sunulmaktadır.



Şekil 9.6 AES-256 Gerçeklemesinin Durum Makinesi

Şekil 9.6'da durumlar aşağıdaki gibidir:

- D0 : Boşta
- D1 : Şifreleme anahtarı alma
- D2 : Şifrelenecek veri alma
- D3 : Şifreleme
- D4 : Şifrelenmiş veri gönderme

- D5 : Çözme anahtarı alma
- D6 : Çözülecek veri alma
- D7 : Çözme
- D8 : Çözülmüş veri gönderme

Şekilde B1, B2, B3 ve B4 ise yukarıda bahsedilen buyrukları ifade etmektedir. Durum makinesinin durumlarının işlevleri ise şu şekildedir:

D0 : Seri kapıdan buyruk dinlenir. Buyruk gelmemişse aynı durumda kalınır. Eğer B1 gelmiş ise D1 durumuna, B2 gelmiş ise D2 durumuna, B3 gelmiş ise D6 durumuna, B4 gelmiş ise D5 durumuna geçilir.

D1 : Bu durumda seri kapıdan 256 bitlik şifreleme anahtarı değeri alınır. Anahtar alma işlemi bittikten sonra D2 durumuna geçilir.

D2 : Seri kapıdan 128 bitlik şifrelenecek veri değeri alınır. 128 bit okunduktan sonra D3 durumuna geçilir.

D3 : Bu durumda; alınan girdilerle şifreleme modülü çalıştırılır. Şifreleme işlemi bitene kadar bu durumda kalınır. Şifreleme tamamlandığında D4 durumuna geçilir.

D4 : Şifreleme modülünün çıktısı olan 128 bitlik şifrelenmiş veri seri kapıdan gönderilir. Gönderim bittikten sonra D0 durumuna geçilir.

D5 : Seri kapıdan 256 bitlik çözme anahtarı değeri alınır. Anahtar alma işlemi bittikten sonra D6 durumuna geçilir.

D6 : Seri kapıdan 128 bitlik çözülecek veri değeri alınır. 128 bitin tamamı okunduğunda D7 durumuna geçilir.

D7 : Bu durumda çözme modülü alınan girdilerle çalıştırılır. Çözme tamamlandığında ise D8 durumuna geçilir.

D8 : Bu durumda ise, çözme modülünün çıktısı olan çözülmüş veri seri kapıdan gönderilir. Gönderim tamamlandıktan sonra durum D0'a yani boşta durumuna geçilir.

9.1.8. Değerlendirme ve Karşılaştırma

AES-256 algoritması daha önce de belirtildiği gibi yalnızca hız kısıtı göz önünde bulundurularak gerçeklenmiştir. Bunun sonucu olarak şifreleme modülü aldığı 128 bitlik girdiyi 19 saat vuruşu sonunda şifreleyerek dışarı vermektedir. Altera Bemicro FPGA kartı üzerindeki osilatör 16MHz'dir. Bu saate bağlı olarak şifreleme aşamasının veri akışı 107.79 Mbps olarak hesaplanmıştır.

AES-256 algoritmasının gerçeklenmesi tamamlandıktan sonra Altera Bemicro FPGA geliştirme kartı üzerinde denenmesinin haricinde farklı FPGA aygıtları için denemeler yapılmış ve şifreleme için çizelge 9.1'deki sonuçlar elde edilmiştir.

Çizelge 9.1 AES-256 Gerçeklemesinin Farklı FPGAlar Üzerindeki Değerleri

FPGA	Frekans (MHz)	Veri Akışı (Mbps)	Kapladığı Alan (%)
Cyclone III EP3C16F256C8N	122.35	824.25	15
Spartan III XC3S2000-5	87.1	586.78	5
Virtex 6 XC6VLX550T-2	374.76	2524.70	< 1

Yapılan gerçeklemenin, diğer bazı gerçeklemelerle karşılaştırılması çizelge 9.2'de sunulmaktadır.

Çizelge 9.2 AES Gerçeklemesinin Diğer Gerçeklemelerle Karşılaştırılması

Gerçekleme →	**Chodowiec[21]**	**Rouvroy[22]**	**Good[23]**	**Bu çalışma**
FPGA	XC2S30-6	XC3S50-4	XC2S15-6	XC3S2000-5
Frekans (MHz)	60	71	67	87
Veri Akışı (Mbps)	166	208	2.2	586.8

Çizelge 9.3'te diğer çalışmalarla alan ve veri akışı karşılaştırması sunulmaktadır.

Çizelge 9.3 AES Gerçeklemesinin Diğer Çalışmalarla Alan ve Veri Akışı Yönünden Karşılaştırılması

Gerçekleme →	**Standaert[24]**	**Jarvinen[25]**	**Zambreno[26]**	**Saggesse[27]**	**Bu çalışma**
FPGA	Virtex-E XCV3200E-8	Virtex-E XCV1000E-8	Virtex-E XC2V4000	Virtex-E XCV3200E-8	Virtex 6 XC6VLX550T-2
Frekans (MHz)	145	129.2	184.1	158	374.76
Dilim	15112	11719	16938	18600	2044
Veri Akışı (Mbps)	18560	16500	23570	20300	2524.70
Veri Akışı / Dilim	1.228	1.408	1.391	1.091	1.235

9.2. AES Sonrası Şifreleme

Bu aşamada, AES-256 ile şifrelenen veriler, anahtarı doğrudan paylaşılmayan bir güvenlik katmanı daha oluşturmak adına bir şifreleme işlemine tabi tutulmaktadırlar. AES çıktı kütüklerinin her baytı yeni bir bayt ile Rijndael Galois alanında çarpma işlemine tabi tutularak yeni değerler elde edilir. Çarpanlar ise, önceden paylaşılmayan, sistemin son aşamasında verinin gizleneceği taşıyıcı ses dosyasından elde edilirler. Taşıyıcı ses dosyasının içerisinde bulunan müziğin, her ölçüsündeki en uzun süreye sahip notalara karşılık gelen bayt değerleri çarpan olarak kullanılır. Çarpanların nasıl elde edildiği daha sonraki bölümlerde anlatılmaktadır.

Bu aşamanın gerçekleştirilmesi, FPGA ile bilgisayar arasındaki veri aktarım boyutunu arttırmamak için yazılım tarafında yapılmıştır. Sonlu alanda çarpma işlemi aşağıda adımları yazılı algoritma yardımıyla gerçekleştirilmiştir.

1. Başla
2. Sayı a ve sayı b'yi oku ve sonuç ve tekrar değişkenini 0 ile başlat
3. Eğer b'nin en alt biti bir ise sonuç değişkenini a ile özel veya işlemine al
4. a'nın en üst bitini kaydet
5. a'yı bir sola kaydır
6. Eğer a'nın 4. adımda kaydedilen en üst biti bir ise a'yı 1B (onaltılık) ile özel veya işlemine tabi tut
7. b'yi bir sağa kaydır
8. tekrar değişkenini bir arttır
9. Tekrar sekize eşit değilse 3. adıma dön
10. Sonuç değişkenini kaydet ve bitir

Sistemin çözme aşamasında ise çarpanların Galois alanında çarpmaya göre tersleriyle çarpma işleminin yapılması gerekmektedir. Çarpmaya göre ters almayı gerçekleştirmek için taramalı çizelge yöntemi kullanılmıştır. Her defasında çarpanların tersini hesaplamak yerine olası tüm çarpanların tersleri hesaplanıp

kaydedilmiş ve ihtiyaç duyulduğunda bu çizelgeden çarpmaya göre tersler okunmaktadır.

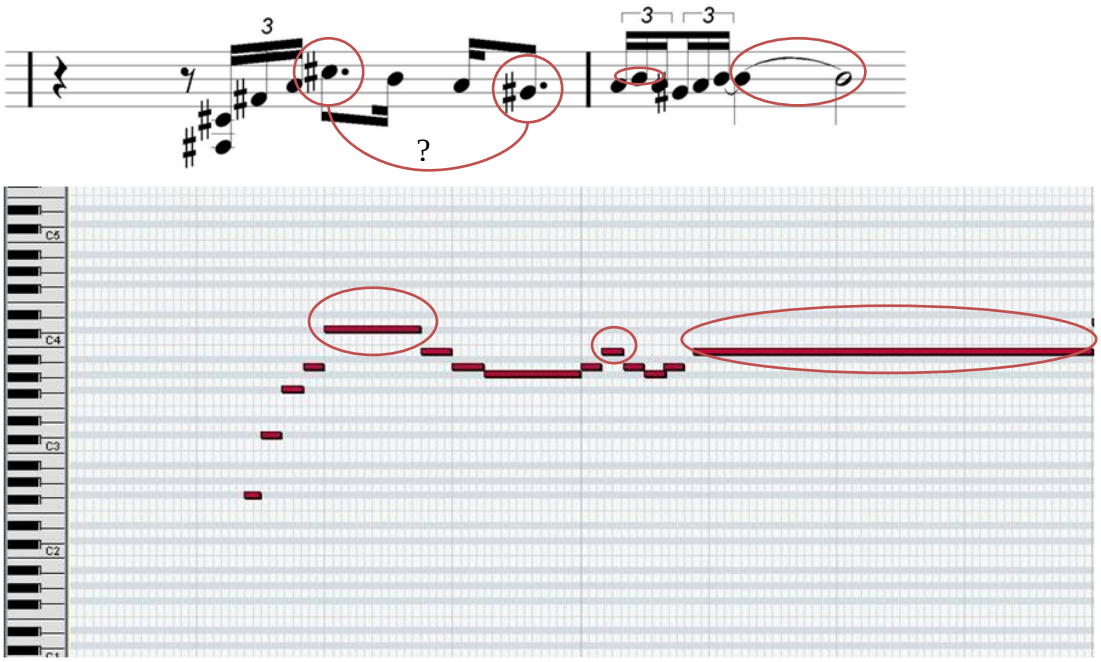
9.3. Taşıyıcı Ses Dosyasından Özniteliklerin Elde Edilmesi

Bir müzik eseri içeren taşıyıcı ses dosyası bu sistemde WAV dosyasıdır. Bu aşamada yapılacak işlemlerin hepsi, ses işleme gerektirmeden müzik bilgisine sahip birisi tarafından el ile de yapılabilmektedir. Ses dosyasından elde edilmek istenen veriler, her ölçüdeki en uzun süreye sahip notanın hangi nota olduğu bilgisidir.

Bu aşamada, ölçülerdeki en baskın başka bir deyişle uzunluğu en yüksek olan nota değerlerini elde etmek için öncelikle ses dosyasının genlik değerleri okunarak zaman alanından frekans alanına geçmek için Hızlı Fourier Dönüşümü Cooley-Tukey algoritması yardımıyla hesaplanır [28]. HFD hesaplamasının taşıyıcı ses dosyasındaki müzik eserinin her ölçüsü için ayrı ayrı yapılması gerektiği için müzik eserinin tempo ve süre bilgileri sisteme girdi olarak verilmelidir. Zaman aralığı herhangi bir değer olabileceğinden, eğer sistem ölçü uzunluğuna bağlı kalmadan çalıştırılmak istenirse (örneğin önceden seçilmiş rastgele zaman aralıkları ile) sistem yine doğru çalışacaktır. Çünkü esas olan belirli bir süre içerisindeki en baskın notayı bulmaktır. HFD sonucunda izgeler hesaplandıktan sonra izge vektörü içerisindeki en yüksek değerin indisi tespit edilir. İndis tespit edildikten sonra, dönüşümü yapılan zaman aralığındaki en uzun süre devam eden frekans denklem 9.3'teki gibi hesaplanır.

$$Baskın\ Frekans = \frac{Örnekleme\ \ Frekans ı}{İzgenin\ \ Eleman\ \ Sayısı} \times indis \tag{9.3}$$

Bu işlem sonucunda istenen tüm aralıklardaki en baskın frekanslar hesaplanmış olur. Eğer müzik eserinin bir ölçüsünde birden fazla aynı uzunluğa sahip nota var ise ve bu notaların uzunlukları diğerlerinden fazla ise, seçilen nota tiz olan (frekansı yüksek olan) notadır. Şekil 9.6'da el yordamı ile bir ölçüdeki baskın notanın nasıl bulunduğu gösterilmektedir.



Şekil 9.7 Baskın Notaların Bulunması

Şekil 9.7'de iki örnek ölçü için baskın notaların nasıl tespit edildiği gösterilmektedir. Üst tarafta müzik yazımı alt tarafta ise soldaki piyano resminin notalarına karşılık gelen uzunluk değerleri gösterilmiştir. Her iki şekle de bakarak iki ölçü için de doğru değerler bulunabilir. Örneğin ilk ölçüde dördüncü oktav do diyez notasının ve üçüncü oktav la bemol notasının aynı uzunluğa sahip olduğu görülmektedir. Böyle durumlarda frekans değeri daha yüksek olan notanın seçileceğinden daha önce bahsedilmişti. O halde birinci ölçü için baskın nota dördüncü oktav do diyez notasıdır. İkinci ölçüde ise üçüncü oktav si notasının çalınma süresi ölçüdeki diğer

tüm notalardan fazladır. O halde ikinci ölçü için baskın frekans üçüncü oktav si notası olarak bulunur.

Bu aşamada bulunan notalar bir çizelge yardımıyla birer baytlık verilere dönüştürülürler ve notalara karşılık gelen bu veriler hem şifreleme aşamasında çarpan hem de veri gizleme aşamasında anahtar niteliğinde kullanılmaktadırlar. Notalara karşılık gelen değerler çizelge 9.4'te sunulmaktadır.

Çizelge 9.4 Notalara Karşılık Gelen Değerler

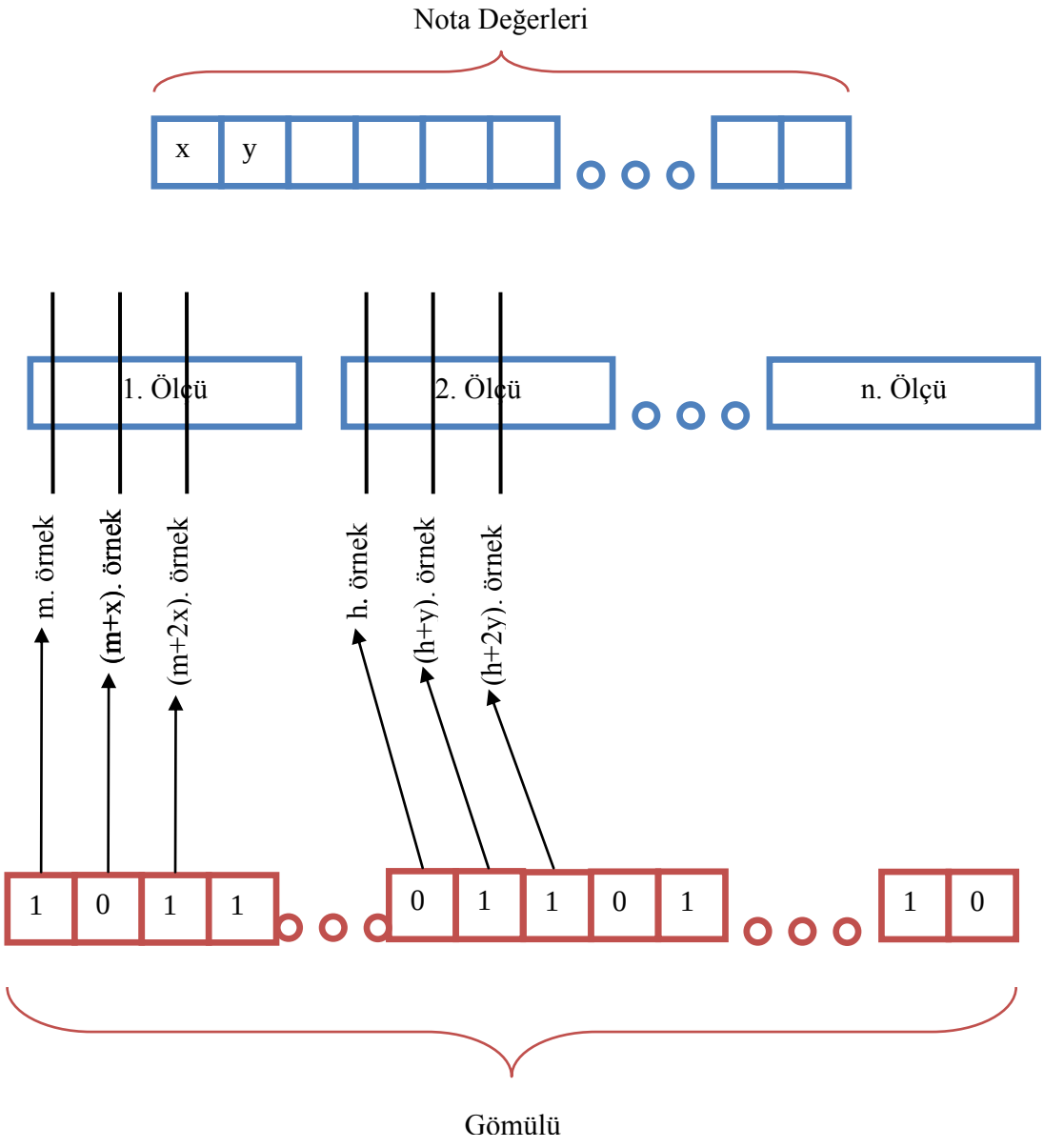
Nota	Değer (Onaltılık)	Nota	Değer (Onaltılık)	Nota	Değer (Onaltılık)
DO0	9C	**LA#2**	D1	**SOL#5**	EA
DO#0	9D	**Sİ2**	D3	**LA5**	E8
RE0	9E	**DO3**	D5	**LA#5**	E6
RE#0	9F	**DO#3**	D7	**Sİ5**	E4
Mİ0	A0	**RE3**	D9	**DO6**	E2
FA0	A1	**RE#3**	DB	**DO#6**	E0
FA#0	A2	**Mİ3**	DD	**RE6**	DE
SOL0	A3	**FA3**	DF	**RE#6**	DC
SOL#0	A4	**FA#3**	E1	**Mİ6**	DA
LA0	A5	**SOL3**	E3	**FA6**	D8
LA#0	A6	**SOL#3**	E5	**FA#6**	D6
Sİ0	A7	**LA3**	E7	**SOL6**	D4
DO1	A8	**LA#3**	E9	**SOL#6**	D2
DO#1	A9	**Sİ3**	EB	**LA6**	D0
RE1	AA	**DO4**	ED	**LA#6**	CE
RE#1	AB	**DO#4**	EF	**Sİ6**	CC
Mİ1	AD	**RE4**	F1	**DO7**	CA
FA1	AF	**RE#4**	F3	**DO#7**	C8
FA#1	B1	**Mİ4**	F5	**RE7**	C6
SOL1	B3	**FA4**	F7	**RE#7**	C4
SOL#1	B5	**FA#4**	F9	**Mİ7**	C2
LA1	B7	**SOL4**	FB	**FA7**	C0
LA#1	B9	**SOL#4**	FD	**FA#7**	BE
Sİ1	BB	**LA4**	FF	**SOL7**	BC
DO2	BD	**LA#4**	FE	**SOL#7**	BA
DO#2	BF	**Sİ4**	FC	**LA7**	B8
RE2	C1	**DO5**	FA	**LA#7**	B6
RE#2	C3	**DO#5**	F8	**Sİ7**	B4
Mİ2	C5	**RE5**	F6	**DO8**	B2
FA2	C7	**RE#5**	F4	**DO#8**	B0
FA#2	C9	**Mİ5**	F2	**RE8**	AE
SOL2	CB	**FA5**	F0	**RE#8**	AC
SOL#2	CD	**FA#5**	EE		
LA2	CF	**SOL5**	EC		

Yukarıdaki çizelge, müzikte yoğun kullanılan notalara daha yüksek değerler verilerek oluşturulmuştur. Bunun en önemli getirisi, veri gizleme aşamasında taşıyıcı dosya üzerinde daha seyrek değişiklik yapmak olacaktır. Bu çizelge, amaca göre istenilen şekilde tasarlanabilmektedir. Örneğin; eğer Geleneksel Türk Müziği eserleri kullanılmak istenirse, buradaki nota sayısı artacak, şifreleme aşamasındaki Galois alanının derecesi büyüyecek ve veri gizleme aşaması daha karmaşık hale gelecektir.

9.4. Veri Gizleme

Tüm sistemin son aşaması olan veri gizleme aşamasında; şifrelenen veri taşıyıcı ses dosyası içerisine gömülerek gizli veri içeren ses dosyası oluşturulur. Veri gizleme işlemi, şekil 8.3'te gösterildiği şekilde yapılmaktadır. Ancak; her bit gömme işleminden sonra atlanacak örnek sayısı rastgele olarak seçilmemektedir. Bölüm 9.3'te anlatılan işlemler sonucu elde edilen değerler veri gizleme alt sistemine gönderilir. Bu değerler örnek atlama miktarlarını belirlerler. Belirlenen her zaman dilimi için, yani her ölçü için, o ölçüden elde edilen değer, yine o ölçüye veri gizlenirken örnek atlama miktarı olarak kullanılır.

Bu aşamada kullanılan dosya biçimi WAV dosya biçimidir. Veri gizleme işlemi başlamadan hemen önce, dosya içerisine gizlenecek veri boyutu ve bu boyutta verinin taşıyıcı dosyaya sığıp sığmayacağı hesaplanır. Daha sonra dosyanın ön anlaşmalı başlangıç noktasından itibaren dosyaya ilk önce ne kadar gizli veri taşıdığı gömülür. Yüksek boyutlu bir dosya ile düşük boyutlu bir veri gönderilmek istenebileceği için alıcının dosyanın içerisinde ne kadar gizli veri taşıdığını bilmesi gerekmektedir. Bu işlem, bu sebeple yapılmaktadır. Şekil 9.8'de veri gizleme sisteminin şekilsel gösterimi sunulmaktadır.

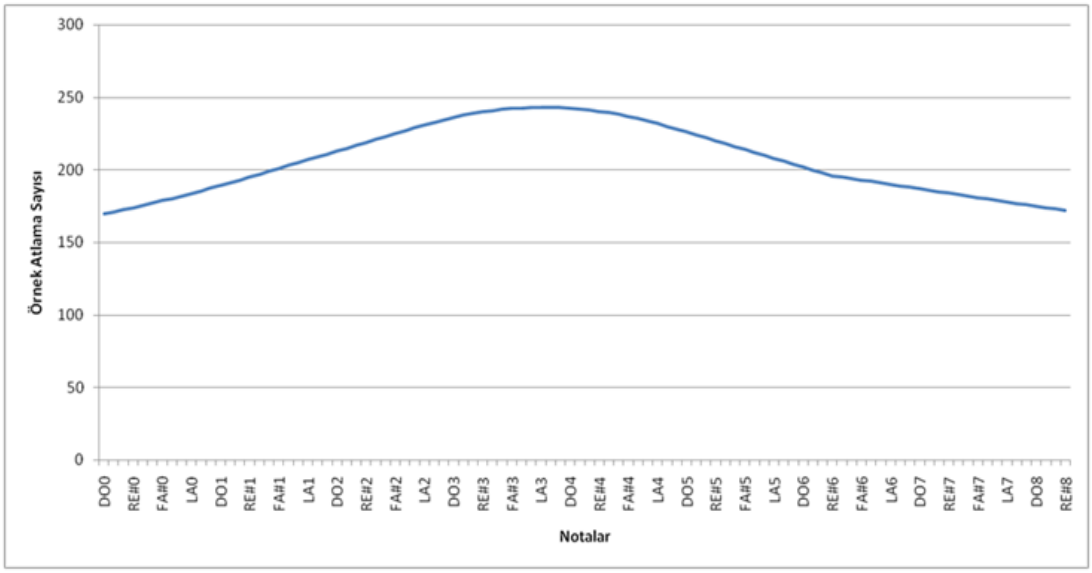


Şekil 9.8 Veri Gizleme Sistemi

Yukarıdaki yöntemle veri gizleme yapıldığında, kaynak (gönderici) ve hedef (alıcı) arasında dosya aktarımından önce herhangi bir verinin paylaşılması gerekmemektedir. Ölçüler içerisindeki örnek atlama sıklıklarını anahtar olarak kabul edersek, anahtar taşıyıcı dosya içerisindeki şarkının kendisi olmaktadır. Bu durum, dosyaya işlevsellik kazandırmaktadır. Aynı zamanda; iki farklı şarkıyla yapılan veri gizleme işlemlerinde, veri gizleme sıklığı tamamen değişiklik göstereceğinden bu

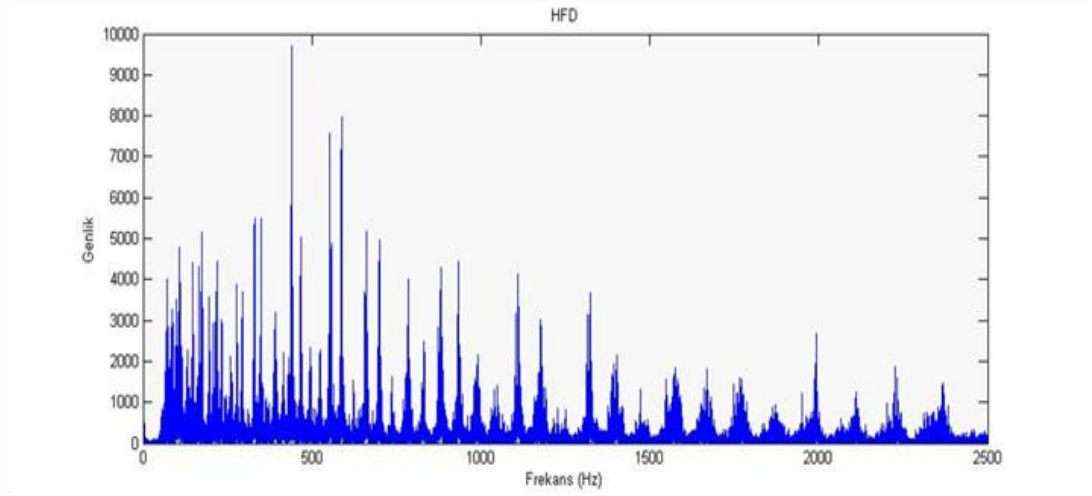
sistem, iki örnek dosyanın karşılaştırıldığında herhangi bir sonuca varılmasını engellemektedir.

Her şarkı için farklı bir veri gizleme yapılacağı için ortaya bir şarkı seçimi kavramı çıkmaktadır. Örneğin çizelge 9.4, en sık kullanılan notaların en yüksek değerlere sahip olması düşünülerek tasarlanmıştır. En iyi durumlardan birinde bir şarkının dördüncü oktav la notası çevresindeki iki oktava yayıldığı düşünülürse; çizelge 9.4'e göre ortalama örnek atlama değeri 243 olarak hesaplanır. Bu demek olur ki; taşıyıcı ses dosyasının örneklerinin 1/243'ünün en anlamsız bitleri %50 ihtimalle değişmektedir. Bu durum da ortalamaya katılırsa, en iyi durumda dosyadaki örneklerin ortalama 1/486'sının en anlamsız bitleri değişecektir. Bu değer sabit olmamakla birlikte oldukça yüksek olduğu için steganalizi zorlaştırabilmektedir. Bir şarkının iki oktav olduğu varsayılarak şekil 9.9'daki grafik oluşturulmuştur. Burada yatay eksendeki notalardan itibaren iki oktav sayılarak ortalama örnek atlama değerleri hesaplanmıştır. Şekilde görüldüğü gibi en kötü durumda ortalama örnek atlama sayısı 150'den büyük olacaktır.

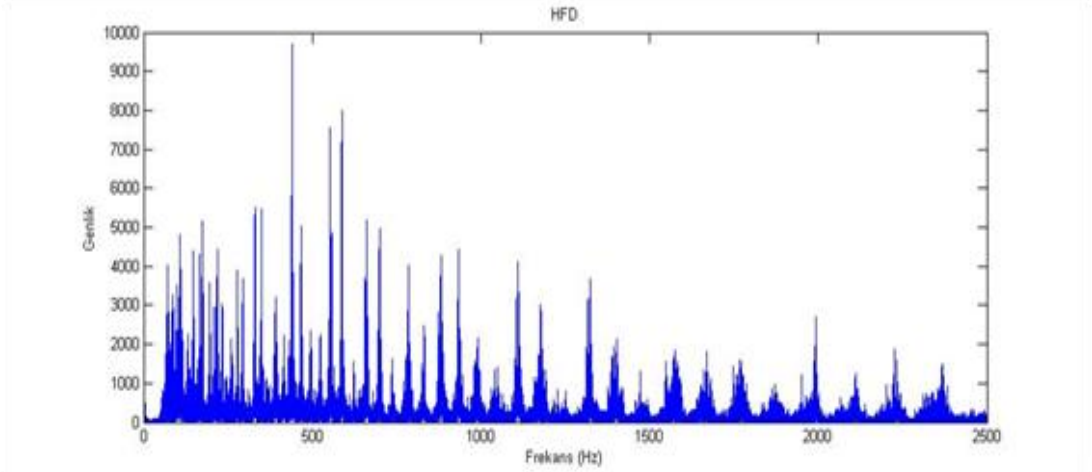


Şekil 9.9 Farklı Oktavlar İçin Ortalama Örnek Atlama Değerleri

Şekil 9.10'da içerisine taşıyıcı ses dosyası ile içerisine veri gömülmüş aynı ses dosyasının izge çözümlemelerinin karşılaştırması sunulmaktadır.



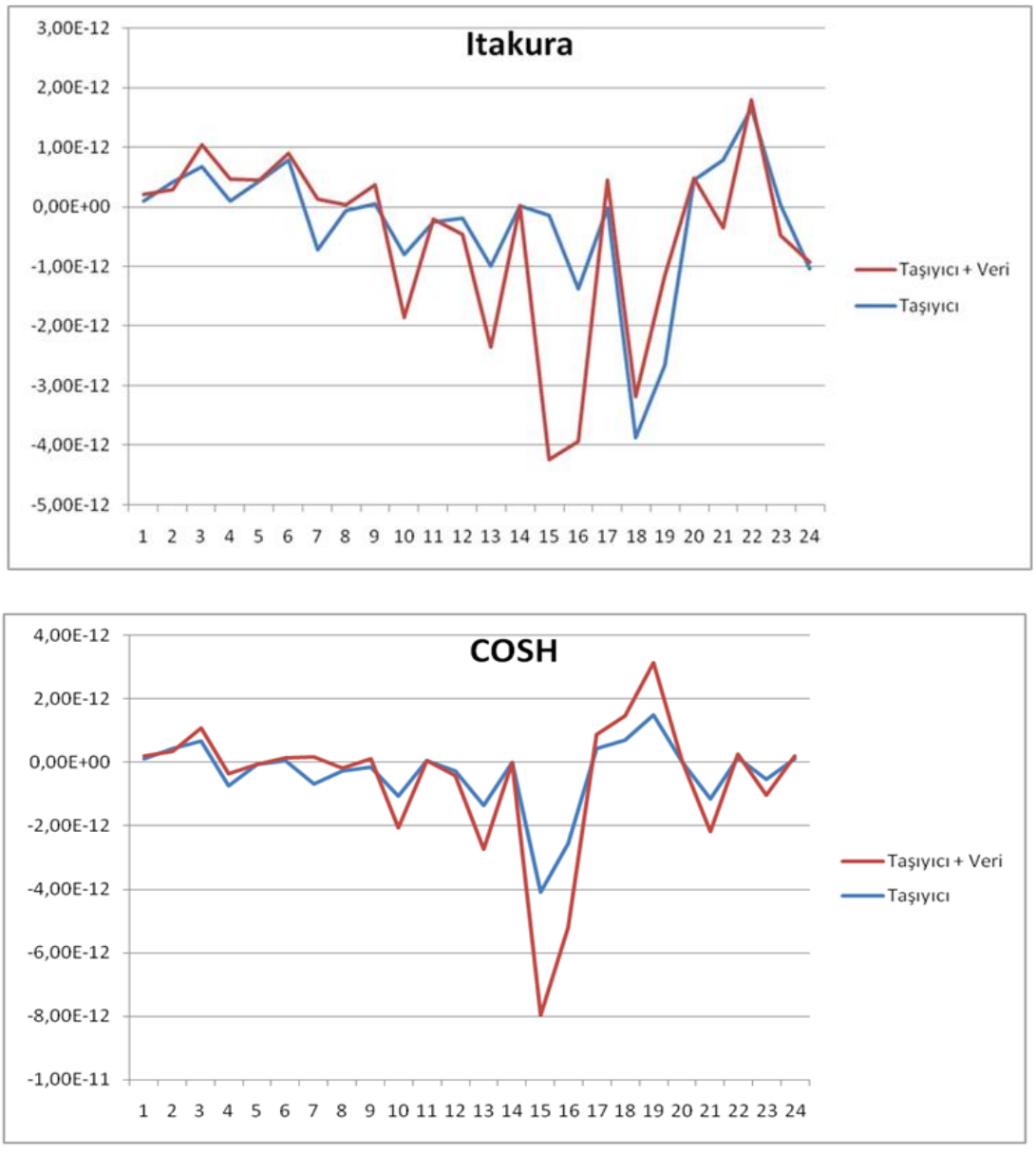
Taşıyıcı Ses Dosyası



Veri Gömülü Taşıyıcı Ses Dosyası

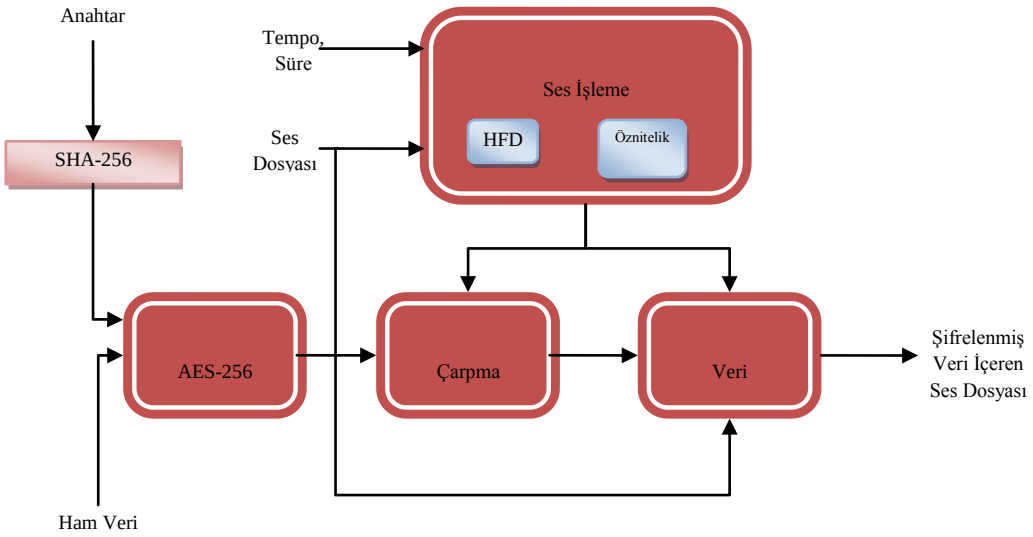
Şekil 9.10 Taşıyıcı Dosyanın Veri Gizlenmeden Önceki ve Sonraki İzge Çözümlemesi

Çizelge 9.4'te sunulan notalara karşılık gelen değerlerin amaca uygun olarak farklı şekilde tasarlanabileceğinden daha önce bahsedilmişti. Bu değerler olabilecek en düşük değerlere çekildiğinde veri taşıma sığası artacaktır. Böyle bir tasarım için veri taşımayan 24 adet ses dosyasının ve veri gömülü aynı 24 dosyanın, ses kalitesi ifadelerinde yoğun olarak kullanılan Itakura ve Cosh mesafesi değerleri[29] hesaplanmış ve sonuçlar Şekil 9.11'de sunulmuştur.

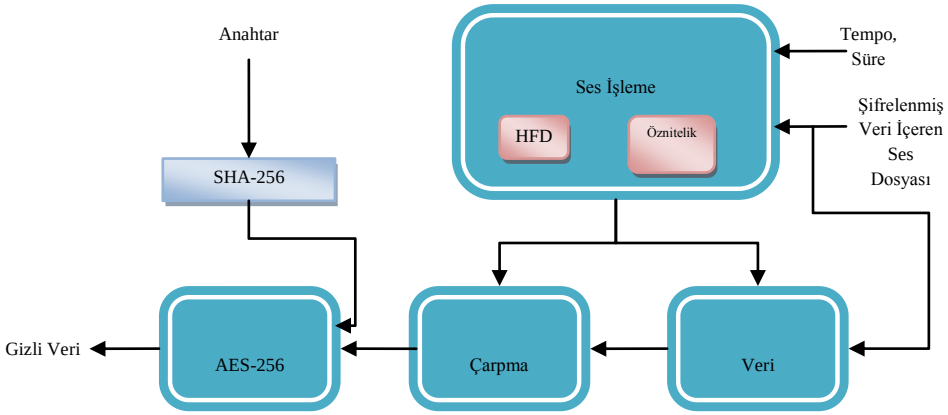


Şekil 9.11 Ses Kalitesi Ölçümü

Dokuzuncu bölümde anlatılan tasarıma ulaşılana dek, farklı tasarımlar gerçeklenip denenmiş ve başarımı en yüksek tasarımın bu çalışmada anlatılan tasarım olduğunda karar verilmiştir. Şekil 9.12'de tasarımın şifreleme-veri gizleme akış şeması, şekil 9.13'te ise tasarımın veri-ayıklama çözme akış şeması sunulmaktadır.



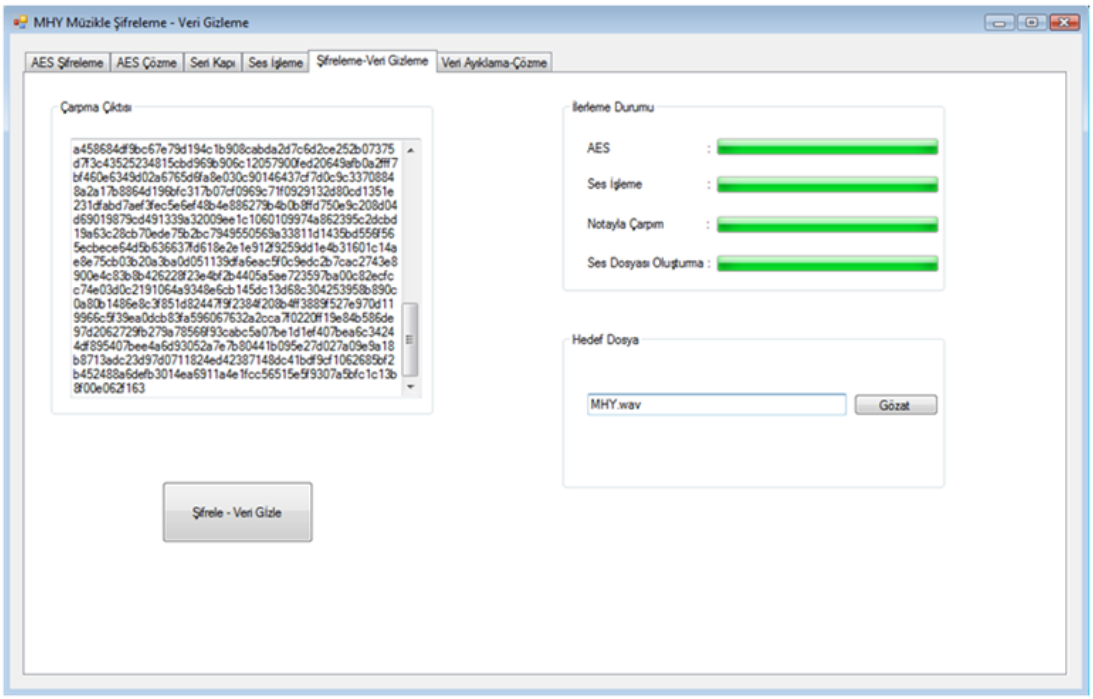
Şekil 9.12 Şifreleme-Veri Gizleme Akış Şeması



Şekil 9.13 Veri Ayıklama-Çözme Akış Şeması

10. SONUÇ

Dokuzuncu bölümde anlatılan gerçekleme, sistemin çalışma süresini kısaltmak adına birbirleri ile haberleşen yazılım ve donanım birimleri halinde yapılmıştır. Çalışmanın son halinde; yazılım ve donanım arasındaki veri aktarımını düşürüp zaman kaybını engellemek için yalnızca AES-256 şifreleme ve çözme algoritmaları FPGA üzerinde çalıştırılmaktadır. Tasarımın diğer tüm birimleri geliştirilen yazılım üzerinde çalışmaktadır. Şekil 10.1'de geliştirilen yazılımın bir ekran görüntüsü sunulmaktadır.



Şekil 10.1 Yazılım Ekran Görüntüsü

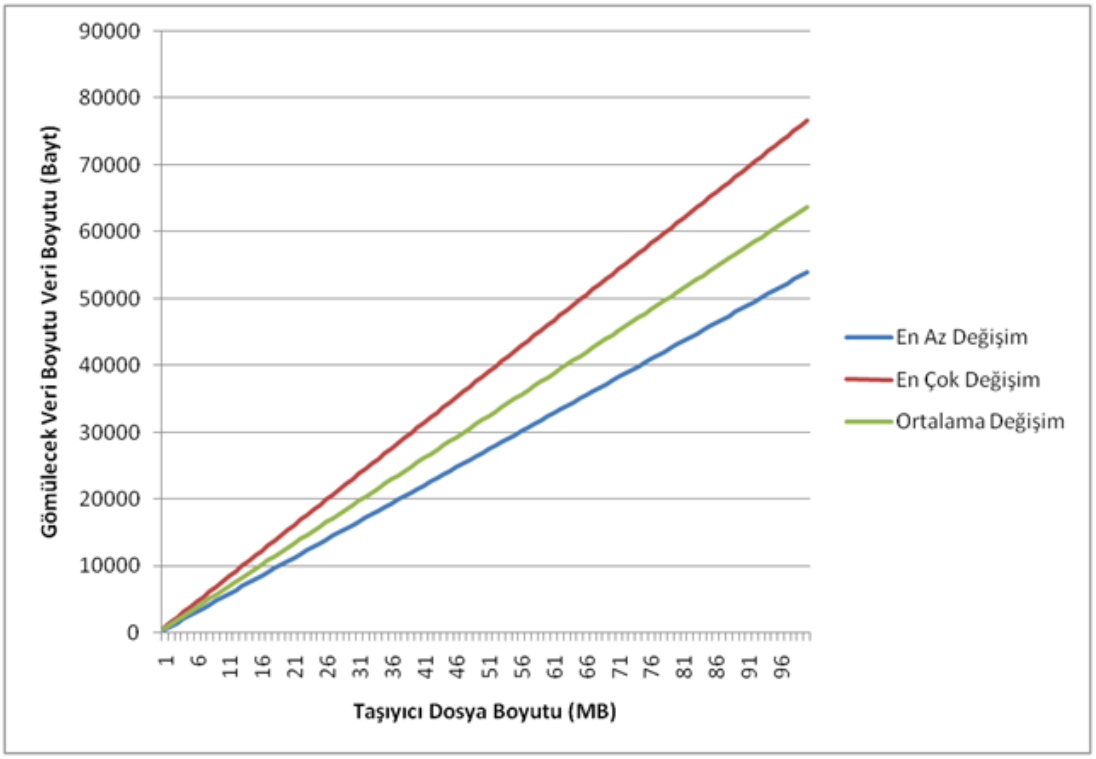
Gerçeklenen şifreleme-veri gizleme sistemi, girdi olarak şifrelenecek veriyi, şifreleme anahtarını, ses dosyasını, müziğin temposunu ve süre bilgisini almakta ve çıktı olarak şifrelenmiş gizli veri içeren ses dosyası vermektedir. Çıktı ses dosyasının kalite çözümlemeleri, önceki bölümde belirtildiği gibi, girdi ses dosyasının çözümlemeleri ile çok yakın değerleri işaret etmektedirler. Bu durum, insan kulağıyla zaten ayırt edilemeyen şifreli veri içeren ses dosyasının güvenliği açısından

önemli bir durumdur. Ayrıca AES-256 algoritmasının güvenilirliği üzerine anahtarı ses dosyası olan bir şifreleme işlemi daha eklenmiş ve güvenlik katman sayısı arttırılmıştır. Bu sistem yapısı bozulmadan, bir çok farklı şekilde tasarlanarak ortaya çeşitli farklı sistemler çıkarılabilmektedir. Örneğin; tüm notaları kullanmak yerine tüm notaların alt kümesi olan herhangi bir nota kümesini kullanmak veya Geleneksel Türk Müziği'nde kullanılan notaları kullanarak (batı müziğinden daha fazla) Galois alanının derecesini arttırıp çarpanların bit sayısını yükseltmek çizelge 9.4'teki nota değerlerini değiştirmek veya ses işlemedeki zaman birimini ölçü bazında değil de belirli bir zaman birimi bazında düşünmek, aynı güvenilirlikte, sisteme çok fazla çeşitlilik sunmaktadır. Bu durum, algoritmayı bilen bir düşman için bile denenmesi günümüz teknolojisiyle imkansız sayıda farklı seçenek ortaya koymaktadır. Yapılacak tüm değişiklikler için, yapı bozulmadan veri ayıklama-çözme sistemi de işlemlerin tersi tasarlanarak gerçeklenebilmektedir.

Veri gizleme aşamasında önerilen yeni yöntem sonucunda taşıyıcı ses dosyasında verinin gömülü olduğu örnekleri karşı tarafa bildirme gerekliliği ortadan kalkmıştır. Çünkü; algoritmayı bilen karşı taraf, hangi notaların kullanıldığını, bu notaların hangi özelliklerinin kullanıldığını ve bu özelliklere karşılık gelen değerleri de bilecek ve aldığı gizli veri içeren ses dosyası üzerinde ister müzik bilgisi yardımıyla el ile, ister sayısal ses işleme yaparak veri gömülü örnekleri bulacak ve gizli veriyi elde edecektir. Bu yöntem uygulandığında, bir ses dosyasının veri taşıma sığası şekil 10.2'deki grafikte sunulmaktadır.

Şekil 10.2'deki grafikte, 8 bitlik örnek boyutuna sahip bir ses dosyası ele alınmaktadır. Bu grafik oluşturulurken çizelge 9.5'te sunulan nota değerleri kullanılmıştır. Buna göre, müzikte en yaygın olarak kullanılan notalara, taşıyıcı ses dosyası üzerinde daha az etkiye sebep olmak adına yüksek değerler verilmiştir. Bu durum grafikte en az değişim ile ifade edilmektedir. Çizelgedeki en düşük değerlerden elde edilen durum ise en yüksek değişim olarak adlandırılmaktadır. Ortalama durum ise tüm değerlerin ortalaması kullanılarak hesaplanmıştır. Ortalama durumda bir megabaytlık bir ses dosyasında 637.82 bayt şifrelenmiş veri

taşınabilmektedir. İstenen durumda ise bu sığa yaklaşık olarak 540 bayttır. Sığanın en yüksek olduğu durumda ise (sıfırıncı oktav do diyez sesinden itibaren iki oktavlık eserlerin ortalaması) bir megabaytlık taşıyıcı dosya, 767.58 bayt şifrelenmiş veri taşıyabilmektedir. Çizelge 9.5'te verilen tasarım amaca uygun olarak, algoritmada hiçbir değişikliğe sebep olmadan değiştirilebilmektedir.



Şekil 10.2 Taşıyıcı Dosyanın Taşıyabileceği Veri Miktarları

Tüm sistem el ile gerçekleştirilebileceği gibi tasarlanan donanım ve yazılım kullanılarak çok daha kısa sürelerde gerçekleştirilebilmektedir. Örneğin, donanımda çalışan AES-256, 374.76 MHz'te, saniyede 2524.70 megabit şifreleme sığasına sahiptir. Sistem, amaca yönelik olarak değiştirilebilecek esnek bir yapıya sahiptir. Bu sistem sayesinde, ek bir veri aktarımına ihtiyaç duyulmadan üç katmanlı güvenlik ile çok kısa sürede veriler şifrelenip, dosyaya gömülüp, karşı tarafa iletilebilmekte ve aynı hızda veriler dosyadan ayıklanıp çözülerek gizli bilgiye ulaşılabilmektedir.

KAYNAKLAR

[1] Chu, P.P., FPGA Prototyping by Verilog Examples, *Wiley & Sons*, New Jersey, 2008.

[2] Stallings, W., Cryptography and Network Security Principles and Practices, *Prentice Hall*, USA, 2005.

[3] Diffie, W., Hellman, M. E., New directions in cryptography, IEEE Transactions on Information Theory, 22(6), 644-654, 2006

[4] Delfs, H., Knebl, H., Introduction to Cryptography : Principles and Applications, *Springer*, Berlin, 2002.

[5] William F. Ehrsam, Carl H. W. Meyer, John L. Smith, Walter L. Tuchman, Message verification and transmission error detection by block chaining, US Patent 4074066, 1976.

[6] Stinson, D. R., Cryptography Theory and Prcatice, *Chapman & Hall/CRC*, USA, 2002.

[7] National Institute of Standards and Technology, Secure Hash Standard, *FIPS PUB*, 180-3, USA, 2008.

[8] Forte, D., The future of the advanced encryption Standard, Network Security, 1999(6), 10-13, 1999.

[9] National Institute of Standards and Technology, Advanced Encryption Standard, *FIPS PUB*, 197, USA, 2001.

[10] Bruen, A. A., Forcinito, M. A., Cryptography, Information Theory, and Error-Correction : A Handbook fort he 21st Century, *Wiley-Interscience*, USA, 2005.

[11] Lidl, R., Niederreiter, H., Introduction to finite fields and their applications, *Cambridge University Press*, Cambridge, 1986.

[12] Daemen, J., Rijmen, V., The Design of Rijndael, *Springer*, Berlin, 2002.

[13] Bracewell, R., N., The Fourier Transform and Its Applications, *Mc Graw Hill*, Singapore, 2000.

[14] Brigham, E. O., The Fast Fourier Transform and Its Applications, *Prentice Hall*, New Jersey, 1988.

[15] Cangal, N., Armoni, *Arkadaş Yayınları*, Ankara, 1999.

[16] Caldwell, J., Steganography, CROSSTALK The Journal of Defense Software Engineering, 25-27 , 2003.

[17] Pooyan, M., Delforouzi, A., LSB-based Audio Steganography Method Based on Lifting Wavelet Transform, IEEE International Symposium on Signal Processing and Information Technology, 600-603, 2007.

[18] Gopalan, K., Audio Steganography Using Bit Modification, International Conference on Multimedia and Expo (ICME '03) Proceedings, 629-632, 2003.

[19] Yavuz, M.H., Ergin, O., Verileri Nota Kullanarak Şifreleme ve Ses Dosyası İçerisine Gizleme, 3. Uluslararası Katılımlı Bilgi Güvenliği ve Kriptoloji Konferansı, Ankara, 2008.

[20] Altera BeMicro Embedded Control Made Easy, erişim adresi: http://www.arrownac.com/offers/altera-corporation/bemicro/, erişim tarihi: 2 Eylül 2010.

[21] Chodowiec, P., Gaj, K., Very Compact FPGA Implementation of the AES Algorithm, Cryptographic Hardware and Embedded Systems, 2779, 319-333-Springer-Verlag, 2003.

[22] Rouvroy, G., Standaert, F. X., Quisquater, J. J., Legat, J. D., Compact efficient encryption/decryption module for FPGA implementation of the AES Rijndael very well suited for small embedded applications, Procedings of the International Conference on Information Technology: Coding and Computing 2004 (ITCC '04), 2, 583-587, 2004.

[23] Good, T., Benaissa, M., AESon FPGA from the Fastest to the Smallest, Cryptographic Hardware and Embedded Systems (CHES 2005), 3659, 427-440, 2005.

[24] Standaert, F., Rouvroy, G., Quisquater, J. J., Legat, J. D., Efficient Implementation of Rijndael encryption in reconfigurable hardware: Improvements & design tradeoffs, CHES 2003, Germany, 2003.

[25] Jarvinen, K. U., Tommiska, M. T., Skytta, J. O., A fully pipelined memoryless 17.8 Gbps AES-128 encryptor, Procedings of the International Symposium on Field Programmable Gate Arrays, 2003.

[26] Zambreno, J., Nguyen, D., Choudhary, A., Exploring Area/Delay Trade-offs in an AES FPGA Implementation, Procedings of FPL '04, 2004.

[27] Saggese, G. P., Mazzeo, A., Mazocca, N., Strollo, A. G. M., An FPGA based performance analysis of the unrolling, tiling and pipelining of the AES algorithm, Procedings of FPL '03, Portugal, 2003.

[28] Cooley, J. W., Tukey, J. W., An algorithm fort he machine calculation of complex Fourier series, Mathematics of computation, 1965.

[29] Özer, H., Avcıbaş, İ., Sankur, B., Memon, N., Steganalysis of Audio Based on Quality Metrics, Security and Watermarking of Multimedia Contents V, SPIE, Santa Clara, CA, 2003

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : YAVUZ, Muhammet Hamdi

Uyruğu : T.C.

Doğum tarihi ve yeri : 1985 - Erzurum

Telefon : (312) 292 42 90

Belgegeçer : (312) 292 42 90

e-mail : mhyavuz@etu.edu.tr

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	Gazi Üniversitesi Kimya Mühendisliği Bölümü	2009

İş Deneyimi

Yıl	Yer	Görev
2009-2010	TOBB Ekonomi ve Teknoloji Üniversitesi	Araştırma Görevlisi
2008-2010	Kasırga Mikroişlemciler Laboratuvarı	Ar-Ge Görevlisi

Yabancı Dil

İngilizce

Yayınlar

Yavuz, M.H., Ergin, O., Verileri Nota Kullanarak Şifreleme ve Ses Dosyası İçerisine Gizleme, 3. Uluslararası Katılımlı Bilgi Güvenliği ve Kriptoloji Konferansı, Ankara, Aralık 2008.

Yavuz, M.H., Ergin, O., İleri Bilgisayar Mimarisi Dersinde Ptlsim Mikroişlemci Benzetim Aracının Kullanımı, Mühendislik Eğitimi Uluslararası Konferansı, Antalya, 2010. (Kasım 2010'da yayınlanacak)