İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

TASARSIZ AĞLAR İÇİN ELİPTİK EĞRİYE DAYALI DIFFIE HELLMAN GRUP ANAHTAR YÖNETİMİ

YÜKSEK LİSANS TEZİ

Müh. Seha SOYLU

Anabilim Dalı: Bilgisayar Mühendisliği

Programı: Bilgisayar Mühendisliği

Tez Danışmanı: Prof. Dr. Ali Emre Harmancı

OCAK 2007

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

TASARSIZ AĞLAR İÇİN ELİPTİK EĞRİYE DAYALI DIFFIE HELLMAN GRUP ANAHTAR YÖNETİMİ

YÜKSEK LİSANS TEZİ
Müh. Seha SOYLU
504031524

Tezin Enstitüye Verildiği Tarih : 25 Aralık 2006
Tezin Savunulduğu Tarih : 31 Ocak 2007

Tez Danışmanı : **Prof. Dr. Ali Emre HARMANCI**
Diğer Jüri Üyeleri **Yrd. Doç. Dr. Sıddıka Berna ÖRS YALÇIN**
Yrd. Doç. Dr. Deniz Turgay ALTILAR

OCAK 2007

ÖNSÖZ

Tez danışmanım sayın Prof. Dr. Ali Emre Harmancı'ya bilimsel katkısı ve çalışmam süresince gösterdiği anlayış ve destek için teşekkürlerimi sunarım.

Aileme, hayatımın her evresinde gösterdikleri ilgi, destek ve çabaları için, eşime varlığı ve sevgisi için, teşekkür ederim.

Ocak, 2007 Seha SOYLU

İÇİNDEKİLER

KISALTMALAR v
TABLO LİSTESİ vi
ŞEKİL LİSTESİ vii
SEMBOL LİSTESİ viii
ÖZET ix
SUMMARY x

1. GİRİŞ 11
1.1 Önbilgi 11
1.2 Amaç 12
1.3 Bölümler 12

2. TASARSIZ AĞLARIN GÜVENLİĞİ 13

3. KRİPTOGRAFİK KAVRAMLAR 16
3.1 Simetrik Kriptografi 16
3.2 Asimetrik(Açık Anahtarlı) Kriptografi 17
3.3 Diffie Hellman Anahtar Anlaşması Yöntemi 17
3.4 Eliptik Eğri Diffie - Hellman Anahtar Anlaşması Yöntemi 18

4. GRUP ANAHTAR YÖNETİMİ 20
4.1 Önceden Anahtar Dağıtımı 20
4.2 Merkezi Anahtar Dağıtımı 21
4.3 Anahtar Anlaşması 21
4.3.1 Grup iletişimi 22
4.3.2 Grup Üyelik İşlemleri 23
4.4 Literatürdeki Grup Anahtar Yönetim Protokolleri 24
4.4.1 2^d – Küp 25
4.4.2 2^d – Ahtapot 26
4.4.3 Asokan-Ginzboorg Yöntemi 28
4.4.4 Cliques 30
4.4.5 STR Yöntemi 38
4.4.6 TGDH 50

5. LİTERATÜRDEKİ PROTOKOLLERİN DEĞERLENDİRİLMESİ 60

6. TASARSIZ AĞLAR İÇİN ELİPTİK EĞRİYE DAYALI DH GRUP ANAHTAR YÖNETİMİ 62
6.1 Anahtar Ağacının Oluşturulması 62

6.2 Grup Anahtarının Hesaplanmasında Eliptik Eğri Diffie Hellman Yönteminin Kullanılması 62

7. TASARSIZ AĞLAR İÇİN ELİPTİK EĞRİYE DAYALI DH GRUP ANAHTAR YÖNETİMİ SİMÜLASYONU 65
7.1 Temel Sınıflar 65
7.2 Performans 71

8. SONUÇ 74

KAYNAKLAR 76

ÖZGEÇMİŞ 79

KISALTMALAR

ADM : Anahtar Dağıtım Merkezi
AS : Asıllama Sunucusu
BK : Körlenmiş Anahtar (Blinded Key)
BOS : Bilet Onaylama Sunucusu
DH : Diffie Hellman
ECDH : Eliptik Eğri Diffie Hellman
DOS : Hizmet İnkarı (Denial of Service)
EC : Eliptik Eğri (Elliptic Curve)
GDH : Grup DH
GE : Görünüm Eşzaman
GSE : Genişletilmiş Sanal Eşzaman
MAC : Mesaj Asıllama (Message Authentication)
MANET : Gezgin Tasarsız Ağ (Mobile Ad hoc Network
PK : Açık Anahtar (Public Key)
PKI : Açık Anahtar Altyapısı (Public Key Infrastructure)
TGDH : Ağaç Tabanlı Grup DH (Tree Based GDH)

TABLO LİSTESİ

		Sayfa No
Tablo 4.1.	2d Küp ve 2d Ahtapotun Karmaşıklığı ……...	29
Tablo 4.2.	Asokan-Ginzboorg En Kötü Durum Karmaşıklık Tablosu	30
Tablo 4.3.	GDH3 Kullanan IKA2 Protokolünün Karmaşık Tablosu …...	32
Tablo 4.4.	Ekleme Protokolünün Karmaşıklık Tablosu..……..	33
Tablo 4.5.	Ayrılma Protokolünün Karmaşıklık Tablosu..…….	34
Tablo 4.6.	Birleşme Protokolünün Karmaşıklık Tablosu.…….....	36
Tablo 4.7.	Bölünme Protokolünün Karmaşıklık Tablosu.…….....	37
Tablo 4.8.	Yenileme Protokolünün Karmaşıklık Tablosu.....……...	38
Tablo 4.9.	Kurulum Protokolünün Karmaşıklık Tablosu …….….....	40
Tablo 4.10.	STR Katılım Protokolünün Karmaşıklık Tablosu …….….....	42
Tablo 4.11.	STR Ayrılma Protokolünün Karmaşıklık Tablosu …….	44
Tablo 4.12.	STR Birleşme Protokolü Karmaşıklık Tablosu….……..	46
Tablo 4.13.	M Gruplu Birleşmenin Karmaşıklık Tablosu…….….	48
Tablo 4.14.	STR Parçalanma Protokolünün Karmaşıklık Tablosu……	50
Tablo 4.15.	TGDH Kurulum Safhası Karmaşıklık Tablosu……...	54
Tablo 4.16.	TGDH Katılma Protokolü Karmaşıklık Tablosu…….	55
Tablo 4.17.	TGDH Ayrılma Protokolü Karmaşıklık Tablosu…….....	57
Tablo 6.1.	EC ve DH Anahtar Uzunluklarının Karşılaştırılması……..	63

ŞEKİL LİSTESİ

		Sayfa No
Şekil 4.1	Grup Üyelik İşlemleri	24
Şekil 4.2	4 Üye İçin 24 – küp Yöntemi	25
Şekil 4.3	$d = 7$, $n = 2$ İçin 2^d Ahtapot Protokolü	28
Şekil 4.4	Cliques IKA2 Protokolü	31
Şekil 4.5	Cliques Katılma Protokolü	33
Şekil 4.6	Cliques Ayrılma Protokolü	34
Şekil 4.7	Cliques Birleşme/Çoklu Katılım Protokolü	35
Şekil 4.8	Cliques Bölünme Protokolü	36
Şekil 4.9	STR Ağacı	39
Şekil 4.10	4 Üyeli Ağaca 5. Üyenin Eklenmesi	41
Şekil 4.11	Geliştirilmiş STR Protokolünde Gruba Katılım İşlemi	42
Şekil 4.12	STR'de 4 Üyeli Bir Grup İçin Gruptan Bir Eleman Ayrılması	44
Şekil 4.13	STR Birleşme Protokolüne Örnek	45
Şekil 4.14	STR Parçalanma Protokolüne Örnek	49
Şekil 4.15	8 Elemanlı Grup İçin TGDH Anahtar Ağacı 1	51
Şekil 4.16	8 Elemanlı Grup İçin TGDH Anahtar Ağacı 2	52
Şekil 4.17	TGDH Kurulum Safhasına Örnek	53
Şekil 4.18	TGDH Üye Ekleme Örneği	55
Şekil 4.19	TGDH Üye Ayrılma Örneği	56
Şekil 4.20	TGDH Birleşme Örneği	58
Şekil 4.21	TGDH Parçalanma Örneği	59
Şekil 6.1	Grup Anahtarının ECDH Yöntemine Göre Hesaplanışı	64
Şekil 7.1	Anahtar Paketi Sınıf Diyagramı	66
Şekil 7.2	İletişim Paketi Sınıf Diyagramı	67
Şekil 7.3	Konfigürasyon ve Arayüz Paketi Sınıf Diyagramı	68
Şekil 7.4	Düğümü Temsil Eden Arayüz	68
Şekil 7.5	Güncellenmiş Anahtar Ağacı 1	69
Şekil 7.6	Güncellenmiş Anahtar Ağacı 2	69
Şekil 7.7	Anahtar Ağacı Paketi Sınıf Diyagramı	71
Şekil 7.8	İşlem Maliyetini Karşılaştırma Grafiği 1	72
Şekil 7.9	İşlem Maliyetini Karşılaştırma Grafiği 2	73

SEMBOL LİSTESİ

N	: Gruptaki üye sayısı
M_i	: i. grup üyesi
M_*	: Tüm grup üyeleri
G	: Çevrimsel cebir grubu
|G|	: G' nin derecesi
g	: G' nin üreteci (üstel taban)
p,q	: Büyük asal sayı
r_i	: Z|G| den, Mi tarafından seçilen gizli değerler.
K_g	: Grup anahtarı
H	: Ağaç derinliği
T_i	: Anahtar ağacının Ti görünümü
KEY_i^*	: Mi' nin anahtar yolundaki anahtarların kümesi
BK_i^*	: Mi'nin körlenmiş anahtarlarının kümesi
$KpBK_i^*$	: Mi' nin anahtar yolundaki körlenmiş anahtarların kümesi
$CoBK_i^*$	: Mi' nin yan yolundaki körlenmiş anahtarların kümesi
Q	: Cebirsel grubun derecesi
Ni	: Mi tarafından üretilen rastsal(gizli) üs
K_n	: n grup üyesi tarafından paylaşılan grup anahtarı

TASARSIZ AĞLAR İÇİN ELİPTİK EĞRİYE DAYALI DİFFİE HELLMAN GRUP ANAHTAR YÖNETİMİ

ÖZET

Son yıllarda tasarsız ağlar ile ilgili birçok araştırma yapılmış ve bu konuya ilişkin örnekler son yıllarda giderek artmıştır. Bu örnekler çoğunlukla askeri uygulamalarda görüldüğü için güvenlik başlıca problemlerden biridir. Bu çalışmada, güvenli iletişimde kullanılacak grup anahtarının oluşturulması için tasarsız ağlara uygun bir grup anahtar yönetimi protokolü önerilmiştir. Önerilen protokol ve orijinal protokolün simülasyonu görsel olarak gerçeklenmiş ve testler için kullanılmıştır.

Literatürde yer alan grup anahtar yönetimi protokolleri incelenerek TGDH yöntemi üzerinde iyileştirme yapılmıştır; çünkü işlem maliyeti, ölçeklenebilirlik ve iletişim maliyeti kısıtlarının tamamı göz önünde bulundurulduğunda TGDH yöntemi diğer yöntemlerden daha az maliyetlidir. Bu nedenle de sınırlı kaynaklara sahip tasarsız ağlar için uygundur.

TGDH yönteminde Diffie Hellman anahtar anlaşma yöntemi kullanılmıştır; fakat bu çalışmada işlem ve bellek maliyetini düşürebilmek için Diffie Hellman yerine Eliptik Eğri Diffie Hellman anahtar anlaşması yöntemi kullanılmıştır. Daha düşük anahtar boylarında aynı güvenliği sağlayan Eliptik Eğri Diffie Hellman yöntemi ile hem işlem maliyeti hem de bellek maliyeti düşürülmüştür. Böylece genellikle düşük güç tüketimi gerektiren düğümlerden oluşan tasarsız ağlara uygun bir grup anahtar yönetim protokolü önerilmiştir.

Önerilen protokolün simülasyonu için JGroups ve BouncyCastle yazılım kütüphaneleri kullanılarak JAVA masaüstü uygulaması yazılmıştır. JGroups kütüphanesi grup iletişim altyapısı için, BouncyCastle kütüphanesi ise anahtar anlaşması yöntemlerinin gerçeklenmesi için kullanılmıştır. Geliştirilen simülasyon programı üzerinde testler yapılarak maliyetlerdeki düşmeler gözlemlenmiştir.

Sonuç olarak Eliptik Eğri Diffie Hellman kullanılan ağaca dayalı grup anahtar yönetim protokolü, güvenlik ve maliyet kıstasları göz önünde bulundurulduğunda tasarsız ağlara uygundur.

TREE BASED ELLIPTIC CURVE DIFFIE HELLMAN GROUP KEY MANAGEMENT FOR AD HOC NETWORKS

SUMMARY

Ad Hoc Networks became a popular research topic and the applications covering this topic as been increased. As the military applications are the most encountered domain, security is one of the most important problems. In this thesis, a group key management protocol for generating and maintaining group key that will be used in secure group communication, is proposed for Ad Hoc Networks. The simulation of the proposed and the original protocol is implemented visually and used for the tests.

Group key management protocols in the literature are searched and analyzed and TGDH group key management protocol is chosen to be enhanced; because considering computation cost, scalability and communication cost, TGDH protocol is efficient than the other protocols. Therefore it is convenient for ad hoc networks that has limited resources.

TGDH protocol uses Diffie Hellman key agreement; but in this thesis for the purpose of reducing the computation and memory costs, Elliptic Curve Diffie Hellman key agreement is used. Computation and communication costs are reduced as Elliptic Curve Diffie Hellman key agreement method uses smaller key sizes at the same security level.

Simulation of the proposed protocol is implemented by using JGroups and BouncyCastle software libraries. JGroups library is used for the communication infastructure of the group and BouncyCastle library is used for the implementation of key agreement methods. Tests are run on the developed simulation program and the reduction of the costs are observed.

In conclusion, tree based group key management protocol using Elliptic Curve Diffie Hellman is convenient for ad hoc networks as considering security and cost criterias.

1. GİRİŞ

Tasarsız ağlar, bir altyapı olmadan kendi kendine haberleşebilen mobil cihazın oluşturduğu haberleşme ağlarıdır. Grup haberleşmesi, dağıtık ve katılımsal uygulamaları içine alan, çok sayıda gönderici ve alıcının bulunduğu bir iletişim yöntemidir[1].

Tasarsız ağlar grup haberleşme uygulamaları için ucuz ve kullanımı kolay bir uygulama platformudur; fakat böyle uygulamaları güvenli hale getirmek güçtür. Güvenliği sağlamaktaki temel zorluk kriptografik anahtarların yönetimidir[1]. Bu çalışmanın konusu tasarsız ağlarda grup anahtar yönetimi problemidir. Çalışmamız, tasarsız ağlar için geliştirdiğimiz katılımsal, ölçeklenebilir grup anahtar yönetimini içermektedir.

1.1 Önbilgi

Gezgin tasarsız ağların uygulamaya geçirilen örnekleri gün geçtikçe arttığı için popülerliği de artmaktadır. Yaygın olarak kullanılan Bluetooth teknolojisi bu uygulamalara örnek olarak verilebilir[2]. Tasarsız ağlar alt yapısız olduğu için kurulması kolaydır. Bu nedenle de kullanım alanı yaygınlaşmaktadır[1].

Grup iletişiminde birden çok gönderici ve alıcı mevcuttur[3]. Bu konuda yapılan çalışmalarda izlenen ortak yollardan biri de dosya paylaşımı, asıllama, elektronik posta gibi merkezi hizmetleri birden çok sistem ve ağa yayılmış dağıtık hizmetlere dönüştürmektir[3]. Dağıtık sistemlere ek olarak, video konferansları gibi işbirlikli uygulamalar da önemli grup iletişimi uygulamalarıdır.

Tasarsız ağların kullanım kolaylığı, tasarsız ağlardaki grup iletişimini, ilgi duyulan bir araştırma haline getirmiştir.

1.2 Amaç

Bu çalışmada, grup iletişimi güvenliğinin temelini oluşturan grup anahtar yönetimi problemi ele alınmıştır. Tasarsız ağlara uygun bir grup anahtar yönetim protokolü geliştirilmesi amaçlanmıştır.

1.3 Bölümler

Bu bölümde çalışmanın konusu ve amacı anlatılmıştır. Bölüm 2'de tasarsız ağların genel olarak güvenliği, Bölüm 3'te tezde geçen temel kriptografik kavramlar, Bölüm 4'te grup anahtar yönetimi, Bölüm 5'te literatürdeki yöntemler, Bölüm 6'da önerilen yöntem, Bölüm 7'da önerilen yöntemin simülasyonu ve Bölüm 8'de sonuçları anlatılmıştır.

2. TASARSIZ AĞLARIN GÜVENLİĞİ

Tasarsız ağlar farklı amaçlar için kullanılabilirler:

- Askeri operasyonlarda: Hareket halindeki cihazlar ve personel iletişim sağlayabilir.
- Kurtarma çalışmalarında: Mevcut iletişim altyapısının devre dışı kaldığı sel, deprem gibi doğal afetlerde, ilk yardım durumlarında.
- Toplantı veya konferanslarda: Katılanlar arasında
- Yasal zorunluluk nedeniyle: Altyapı kurulmasının yasal olarak mümkün olmadığı yerlerde

Tasarsız ağların genel özellikleri ve alt yapısı olan ağlar ile arasındaki farklar [4]'te açıklanmıştır. Tasarsız ağdaki düğümlerin kaynaklarının sınırlı olması yapılacak güvenlik çalışmalarını da sınırlandırmaktadır. Bu kısıtlar küçük merkezi işlem birimi (CPU), az bellek, kısıtlı band genişliği ve sınırlı batarya olarak özetlenebilir.

- Küçük merkezi işlem birimi ve az bellek: İşlemcinin hesaplama gücü kısıtlı olduğundan karışık hesaplamalar yavaş yapılır.
- Kısıtlı bant genişliği: İletişim telsiz ortamda gerçekleşir. Band genişliği kısıtlı olduğundan sabit kanal ayırma yöntemleri yerine band genişliğini daha etkin kullanan, dinamik kanal ayırma yöntemleri uygulanmalıdır.
- Sınırlı batarya: Batarya ömrünün sınırlı olması dikkat edilmesi gereken bir özelliktir. Yol atamadaki güç tüketimi en aza indirilmeye çalışılmalıdır. Düğümler bataryayı ekonomik olarak kullanmak için alıcılarını periyodik olarak açarlar. Alıcılar her zaman açık olmadığı için düğümlerin uyanık olduğu zamanı beklemek gerekebilir.

Tasarsız ağlarda sıklıkla topoloji değişiklikleri görülür. Düğümler, ağa istediği zaman katılır ve istediği zaman ağdan ayrılır. Bunun sonucu olarak sık sık yayınlanan güncelleme bilgileri ağ kaynaklarını meşgul eder. Bu değişiklikler ile ilgili bilginin gecikmesi ise yol atamada istikrarsızlığa sebep olur. Verimlilik ve tepki süresi arasındaki dengeyi sağlamak oldukça zordur[4].

İletimde genellikle kızılötesi ya da radyo frekansı kullanılmaktadır. Dolayısıyla iletim menzili kısadır. IrDA (Infrared Data Association), veri protokolünde iki aygıt arasındaki iletim mesafesini 2 m olarak belirtirken, düşük güçlü düğümlerde bu aralık 20–30 cm.'ye düşebilir. Menzil uzaklığını arttırmak için paketler komşu düğümlere gönderilir. Bu düğümler de hedefe ulaşana kadar paketleri ilerletirler[4].

Tasarsız ağların altyapısız olması, merkezi bir denetim biriminin olmaması, güvenlik politikalarının gerçekleştirilmesinde kısıttır. Rasgele ve ani yerleşen düğümler, dinamik olarak değişen ağ topolojisi, güvenlik çözümlerinin dinamik olmasını gerektirir. Saldırıya açık telsiz ortamda çoklu iletişim ve son derece kısıtlı özkaynaklar ile daha da zorlaşan güvenlik konusu üzerinde çalışılmaya devam edilmektedir.

Güvenlik konusunda ulaşılması beklenen nitelikler[5]:

Ulaşılabilme: Ağ hizmetlerinin hizmet aksatma (DoS) saldırılarına rağmen varlığını koruması hedeflenir. DoS saldırıları tasarsız ağların her katmanında gerçekleştirilebilir. Kötü niyetli kullanıcı ağa mesaj yığarak kanallarda çatışma yaratabilir. Ağ katmanında yol atama protokolünün çalışmasını aksatarak ağ bağlantısını koparabilir ve daha üst katmanlarda üst düzey hizmetleri çökertebilir.

Asıllama: Bir düğüm başka bir düğümün yerine geçerek yetkisi olmadığı halde kaynağa erişmemeli ve diğer düğümlerin işleyişine müdahale etmemelidir.

Bütünlük: Mesaj kötü niyetli olmayan bozucu bazı etkenlere maruz kalabilir ya da ağda kötü niyetli saldırılara uğrayabilir. Verinin bütünlüğü korunmalıdır.

Güvenirlik: Bilginin, yetkisi olmayan kullanıcılara açılmaması hedeflenir. Askeri ya da stratejik bilgiler gibi hassas bilgilerin ağda iletimi güvenilirliği gerektirir. Yol

atama bilgisinin güvenirliği de bazı durumlarda daha değerli olabilir. Örneğin savaş alanında düşman taraf bu bilgiye göre ateş edebilir.

İnkâr edememe: A düğümü B düğümünden hatalı bir mesaj aldığında bu mesajı kullanarak B'yi suçlayabilmeli ve diğer düğümleri bu konuda ikna edebilmelidir.

3. KRİPTOGRAFİK KAVRAMLAR

Bu bölümde tez çalışması için gerekli temel kriptografik bilgi ve yöntemler açıklanacaktır.

Kriptografi, Romalılar zamanından beri var olup 2. dünya savaşı gibi modern savaşlarda önemli rol oynamıştır. Son zamanlarda elektronik ticaretin de kullanılmaya başlanılmasıyla kriptografi alanına büyük çapta ticari ilgi oluşmuştur[5].

Kriptografi; şifreleme ve deşifreleme işlemleri üzerine kuruludur. Şifreleme, mesajı düz metin olarak okunamayacak şekle sokma işlemidir. Bu işlem şifreleme anahtarı ile gerçekleştirilir. Deşifreleme, mesajı orijinal şekline çevirme işlemidir. Bu da bir deşifreleme anahtarı ile gerçekleştirilir[5].

3.1 Simetrik Kriptografi

Simetrik kriptografi, şifrelemede ve şifreyi çözmede aynı şifreleme anahtarını kullanır. Simetrik kriptografi aynı zamanda Özel Anahtar Kriptografi olarak da bilinir[5].

Simetrik kriptografi, bilgiyi şifrelemek ve deşifrelemek için tek bir anahtar kullandığından, kriptolamada kullanılan anahtar kullanılarak mesaj açılabilir.

Simetrik kriptografinin blok ve akış şifreleme olarak bilinen iki tip şifreleme algoritması vardır. Blok şifreleyici bilgiyi bloklar halinde işler. Akış şifreleyici bilgiyi bitlere veya bazı durumlarda sekizlilere göre işler[5].

Blok şifreleme şifrelenecek bir blok bilgiyi alır ve anahtarı ile önceden belirlenmiş fonksiyonu kullanarak onu aynı boyuttaki başka bir bloğa dönüştürür. Akış şifreleme farklı uzunluklardaki girişlerle çalışabilir[5].

3.2 Asimetrik(Açık Anahtarlı) Kriptografi

Açık anahtarlı şifreleme/deşifreleme algoritmaları, şifreleme için bir anahtarı, deşifreleme için de bu anahtardan türetilen ikinci bir anahtarı kullanırlar. Bir anahtar şifreleme için kullanılmışsa, deşifreleme için diğer anahtar kullanılır[5].

1. Ağdaki alıcı düğüm, mesaj alındığında şifreleme ve deşifreleme için kullanacak olduğu anahtar parçasını yaratır.

2. Her düğüm, şifreleme anahtarını herkes tarafından erişilebilen bir dosya ya da saklayıcıya kaydederek paylaşır. Bu anahtar açık anahtardır. Gizli anahtar saklı tutulur.

3. A, B`ye bir mesaj yollamak isterse, mesajı B`nin açık anahtarını kullanarak şifreler.

4. B, mesajı aldığında, bu mesajı kendi özel anahtarını kullanarak çözer. Diğer hiçbir alıcı mesajı çözemez, çünkü mesajı çözebilen anahtarı sadece B bilir.

3.3 Diffie Hellman Anahtar Anlaşması Yöntemi

Diffie - Hellman, anahtar dağıtma problemine ilk pratik çözümdür [6]. "Üs alarak anahtar değiştirme" olarak da bilinen bu sistem daha önce hiç haberleşmemiş iki tarafın açık kanal üzerinden mesajlarını birbirlerine göndererek ortak bir anahtar yaratmasını sağlar. Diffie – Hellman algoritması ayrık logaritmaların hesabının zorluğuna dayanır.

p yeteri kadar büyük bir asal sayı olsun öyle ki Z_p^* de ayrık logaritma problemini çözmek mümkün olmasın. g, Z_p^*'de primitif bir kök olsun. p ve g herkes tarafından bilinsin[6].

A ve B haberleşen 2 taraf olmak üzere, aşağıdaki yolu izleyerek ortak bir anahtar yaratabilirler:

- A, 0 ≤a≤p–2 eşitsizliğini sağlayan rasgele bir a doğal sayısı seçer. $c = g^a \bmod p$ 'yi hesaplar ve B'ye gönderir.

- B, 0 ≤b≤p-2 eşitsizliğini sağlayan rasgele bir b doğal sayısı seçer. $d = g^b \bmod p$ 'yi bulur ve A'ya gönderir.

A, ortak anahtar k'yı şu şekilde hesaplar: $k = d^a = (g^b)^a \bmod p$

B, ortak anahtar k'yı şu şekilde hesaplar: $k = c^b = (g^a)^b \bmod p$

Böylelikle, A ve B aralarında ortak bir anahtar olan k için anlaşmış olurlar.

3.4 Eliptik Eğri Diffie - Hellman Anahtar Anlaşması Yöntemi

Eliptik eğrilerin kriptografide kullanılması 1985 yılında Neal Koblitz[7] ve Victor Miller[8] tarafından önerilmiştir. ECDH yönteminde kullanılan eğriler reel sayılar üzerindeki eliptik eğriler değildir. Genelde 2 sonlu cisim üzerinde tanımlanır[7-8]:

- F_p sonlu cismi: p eleman içeren asal sonlu cisim.
- F_2^m sonlu cismi: 2^m eleman içeren sonlu cisim.

Bu çalışmada F_p asal sonlu cismi ele alınmıştır. F_p, sonlu asal cisimdir öyle ki p tek asal sayıdır. $a,b \in F_p$, $4a^3 + 27b^2 \neq 0 (\bmod p)$ eşitliğini sağlasın. Bu durumda, F_p üzerinde $E(F_p)$ eliptik eğrisi, $a,b \in F_p$ parametreleri ve $x,y \in F_p$ olmak üzere $y^2 = x^3 + ax + b (\bmod p)$ denklemini sağlayan $P = (x,y)$ çözüm kümeleri ile tanımlanır. $y^2 = x^3 + ax + b (\bmod p)$ denklemi $E(F_p)$ eliptik eğrisinin tanım denklemidir. Verilen bir $P = (x_p, y_p)$ noktası için x_p, P noktasının x koordinatı, y_p ise P noktasının y koordinatıdır[7-8].

Eliptik eğri şifreleme kullanan kriptografik yöntemler eliptik eğri noktalarının skaler çarpımlarına dayanır. k tamsayısı ve $P \in E(F_p)$ noktası için skaler çarpım, P'yi

kendisine k defa ekleme işlemidir. Bu skaler çarpımın sonucu $k \times P$ veya kP ile gösterilir[7-8].

A ve B haberleşen 2 taraf olmak üzere, bir eliptik eğri ve bu eğri üzerinde bulunan bir P noktası üzerinde başlangıçta anlaşırlar.

- A, A_S rasgele sayısını seçer. A_S*P 'yi bulur ve B'ye yollar.
- B, B_S rasgele sayısını seçer. B_S*P 'yi bulur ve A'ya yollar.

A, ortak anahtar k'yı şu şekilde hesaplar:

$k = A_S*B_S*P$

B, ortak anahtar k'yı şu şekilde hesaplar:

$k = B_S*A_S*P$

Böylelikle, A ve B aralarında ortak bir anahtar olan k için anlaşmış olurlar. Gizli anahtar olan k, eliptik eğri üzerinde bir noktadır. Bu gizli anahtar oturum anahtarı olarak kullanılacaksa, bir tamsayı türetilmelidir. Bu tamsayı türetme işlemi için iki yöntem vardır:

1. Eğer oturum anahtarının eliptik eğri üzerinde tekrar elde edilebilir olması gerekmiyorsa x-koordinatı veya x-koordinatının özü oturum anahtarı olarak kullanılabilir. Bu yöntem oturum anahtarından gizli anahtar k' nın geri elde edilebilmesi mümkün değildir.
2. Eğer oturum anahtarının eliptik eğri üzerinde tekrar elde edilebilir olması gerekiyorsa nokta belirlenecek bir formata göre kodlanır, tekrar kullanılmak istendiğinde bu kodlama biçiminde göre çözülerek elde edilir.

4. GRUP ANAHTAR YÖNETİMİ

Grup anahtar yönetimindeki amaç, grupta yer alanlar arasında paylaşılacak anahtarı oluşturmaktır. Bu anahtar güvenli grup iletişimi için şifrelemede kullanılacaktır. Bu amacı gerçekleştirmek üzere yapılan çalışmalar [3]'e göre aşağıdaki şekilde sınıflandırılır:

- Grup üyelerinin her birine iletişim ortamına girmeden önce dağıtılır.
- Grupta yer alan bir lider tarafından oluşturulup tüm grup üyelerine dağıtılır
- Üyeler arasında belirlenmiş bir protokolle oluşturulup kullanılabilir. (anahtar anlaşması)

Bu tezde ele alınan yöntemde, katılımsal olarak grup anahtarı oluşturulmaktır. Bu bölümde, önceden anahtar dağıtımı, merkezi anahtar dağıtımı ve anahtar anlaşması kavramları açıklanacaktır.

4.1 Önceden Anahtar Dağıtımı

Önceden anahtar dağıtımı yönteminde, iletişim başlamadan önce, anahtar bilgisi tüm düğümlere dağıtılır. Grup anahtarı her düğümde saklanan bir bilgi olduğu için, düğümlerden biri ele geçirildiğinde grup iletişiminin güvenliği tehlike altındadır[3].

Duyarga ağları alanındaki çalışmalarda, önceden anahtar dağıtımı, her duyarga düğümünün $N-1$ anahtar sakladığı (N , ağdaki düğüm sayısı) ve bu anahtarlarının her birinin sadece ağdaki diğer bir duyarga düğümü tarafından bilindiği bir yöntem olarak düşünülür [9].

Ağdaki bir grup düğüm ele geçirilirse, ele geçirilemeyen düğümlerin oluşturduğu grubun güvenliği korunur. Bu yöntemin, saklanacak anahtar sayısı bakımından bellek

ihtiyacı yüksek olduğundan çok sayıda düğüm içeren gruplar için uygulanabilirliği yoktur. Bunun için duyarga ağlarında bu sorunu çözmeye yönelik yöntemler geliştirilmiştir [9].

4.2 Merkezi Anahtar Dağıtımı

Grup yöneticisi veya anahtar sunucusu, gizli anahtarları üreterek grup üyelerine dağıtır. Bu anahtarların güvenli olarak iletilmesi için önceden paylaşılmış bir gizli anahtar veya PKI olmalıdır. Anahtarları üreten düğüm diğer düğümlere oranla çok fazla işlem yüküyle karşı karşıyadır[3].

Bu sınıfa giren birkaç yöntem tasarlanmıştır [10 - 12]. Bu yöntemlerden bazıları uygulamalara ve standartlara öncülük etmiştir. Örneğin, simetrik anahtar teknolojisine dayanan Kerberos [13, 14], ANSI X9.17 [15] ve KryptoKnight[16] veya açık anahtar teknolojisini kullanan ISO-CCITT X.509 Dizin Asıllama[17] yöntemi.

Kerberos, Needham - Schroeder protokolüne [11] dayanır. Güvenilir üçüncü bir kaynak olan Anahtar Dağıtım Merkezi (ADM) kullanır. ADM iki bölümden oluşur; Asıllama Sunucusu(AS) ve Bilet Onaylama Sunucusu(BOS). Ağdaki her düğüm Kerberos sunucusu ile bir gizli anahtarı paylaşır. Bu anahtar, düğümün kimliğini onaylamada kullanılır. Örneğin A düğümü B düğümü ile konuşmak istediğinde, A ADM'den B ile konuşmak için gerekli anahtarı elde etmelidir. ANSI X9.17 'de ise A, önce B ile iletişime geçerek B'nin ADM'den gerekli anahtarı almasını sağlar. KryptoKnight, Kerberos ile aynı hizmetleri sunar; fakat KryptoKnight'ta kullanılan asıllama ve anahtar dağıtım protokolleri esnek ve ölçeklenebilirdir. KryptoKnight asıllama için mesaj özü fonksiyonu kullanır.

4.3 Anahtar Anlaşması

Anahtar anlaşması yöntemlerinde, grup üyeleri ortak anahtarı hesaplamada kullanılacak protokol üzerinde anlaşır[3]. Bu çalışmada ele alınan protokol bu kategoride olduğu için ayrıntılı olarak açıklanmıştır.

4.3.1 Grup iletişimi

Güvenilir bir grup iletişim ortamı yok ise grup anahtarı belirlemek zordur. Güvenli grup anahtar oluşturma protokolü, ağ katmanında çalışan protokol mesajlarının iletimi için kullanılan grup iletişim semantiklerine bağlıdır. Yaygın olarak iki güçlü grup iletişim sistemi kullanılır: Genişletilmiş Sanal Eşzamanlama(GSE) [18, 19] ve Görünüm Eşzamanlaması(GE) [20]. İki sistemin de sağladığı ortak hizmetler:

1. Grup üyeleri, iki ardışıl üyelik işlemi arasında aynı mesaj kümesini görür.
2. Gönderenin istediği mesaj sırası korunur.

GE daha kuralcı bir hizmet sunarken, GSE uygulamaları genelde daha verimli çalışır.

Bu iki sistem arasındaki temel fark, görünüm değişikliklerini nasıl ele aldıklarıdır. GSE' de görünüm değişiklikleri istekçilerden habersiz olarak gerçekleştirilirken, GE' de yeni görünüm ancak istekçinin izniyle yüklenir. Gerçek uygulamalarda istekçinin kontrolüyle doğru görünüm değişikliklerinin yapılması mümkün değildir.

GE grup iletişim sisteminde, katılma, ayrılma, bölünme, birleşme işlemleri sonucunda görünüm değişebilir. Grup iletişim sistemi görünümün değiştiğini ve yeni görünümün yüklenmesi gerektiğini bildirir. Üye, grup iletişim sistemine yeni görünümü yüklemesi için izin verir. Bu ana kadar, üye, gruptakilerden mesaj alabilir; fakat gruba hiçbir mesaj gönderemez.

GSE grup iletişim sisteminde, üyelik durumunda değişim olduğunda yeni görünüm, üyelere mesaj olarak gönderilir. GSE sistemi, daha hızlı ve görünüm değişikliklerine karşı daha ölçeklenebilirdir. GSE' de üyenin kontrolü daha az olduğu için, GSE altında algoritma programlamak, SE altında algoritma programlamaktan daha zordur. Örneğin GSE grup iletişim sistemlerinde, mesaj, gönderene geri gelene kadar, gönderici başka hangi üyelerin kendi mesajını almış olabileceği konusunda bilgi edinemez. Görünüm çok sık değişebileceğinden dolayı algoritma tüm görünüm değişikliği birleşimlerini göz önünde bulundurmalıdır.

GE grup iletişim sistemlerinde, gönderen düğüm, hangi düğümlerin mesajını alacağını bilir. Görünüm değişikliği de uygulamanın istediği hızda gerçekleşir. Bu

özellikler göz önüne alındığında, dağıtık, hata toleranslı grup anahtarı üretme protokol uygulaması için GE grup iletişim sistemi gerekir. GSE üzerinde gerçeklenen grup anahtarı oluşturma protokolü, gönderilenlerin sırasını tutabilmek için GE' nin gerçekleştirdiklerini gerçeklemek durumunda kalacaktır.

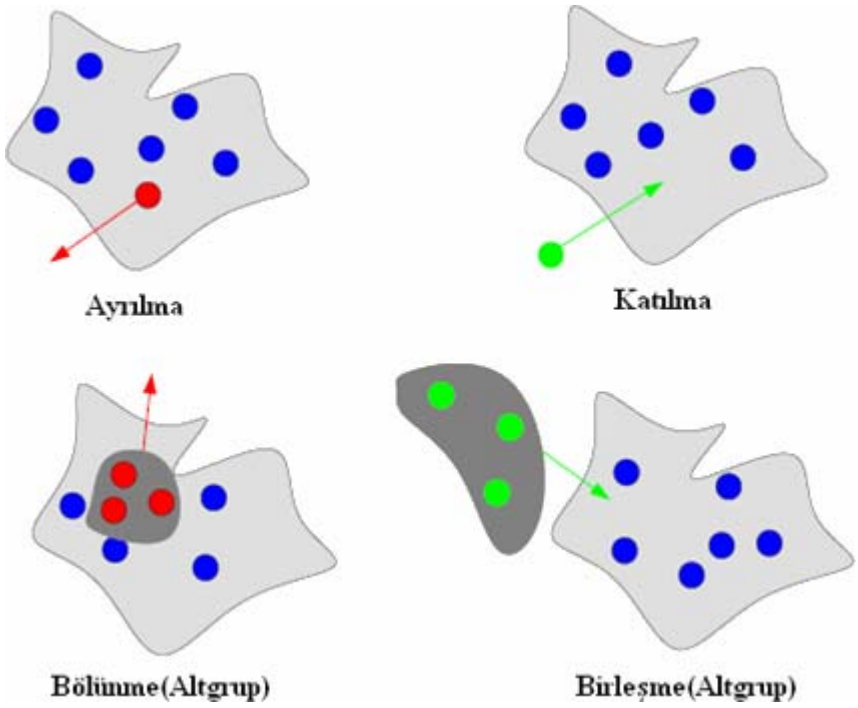
4.3.2 Grup Üyelik İşlemleri

Dinamik bir grupta, görüntü değişikliği çok sık gerçekleşebilir. Grup belirli bir amaç için oluşturulabilir. Yeni üyeler gruba dâhil olmak isteyebilir. Bazı üyeler isteyerek ya da iletişimde meydana gelen hatalar nedeniyle gruptan ayrılabilir. Grup anahtar anlaşması, bu dinamik yapıyı göz önünde bulundurmalıdır[3].

Kurulum işlemi: Grup oluşturulması sırasındaki işlemdir. Bu aşamada, önceden üzerinde anlaşılan protokol kullanılarak grup anahtarı oluşturulur. Tasarsız ağlar için birçok protokol vardır. Anahtar ağacına dayanan grup anahtarı anlaşması protokollerinde, düğümler çeşitli kıstaslara göre sıralanarak ağaç oluşturulur. Grup anahtarı anlaşması, güvenli grup iletişimine zemin hazırlar. Bu aşamadaki protokol yükü, mümkün olduğunca düşürülmelidir.

Dinamik grup işlemleri: Dinamik bir grupta, ağ katmanındaki grup iletişim sistemindeki görünümler dinamiktir. Bu nedenle, grup oluşturulduktan sonra grup işlemleri devam eder.

İlk grup anahtar anlaşması evresi, protokolün sadece bir bölümünü oluşturur. Grup anahtar anlaşması protokolü, üyelik durumunun değişmesi halinde grup anahtarının güncellenmesinden de sorumludur. Bu aşamadaki üyelik işlemleri üye katılımı ve ayrılmasıdır. Ekleme işlemleri bir veya daha çok üyenin gruba dâhil edilmesi, ayrılma işlemleri de bir veya daha çok üyenin gruptan çıkarılması işlemleridir. İşlemler, üye sayısına göre de sınıflandırılabilir. Bu durumda, işlemler tekil veya çoğul üyelik işlemleri olarak ayrılır. Tüm üyelik işlemleri Şekil 4.1'de gösterilmiştir.



Şekil 4.1: Grup Üyelik İşlemleri

4.4 Literatürdeki Grup Anahtar Yönetim Protokolleri

Bu bölümde, incelenen grup anahtar yönetim protokollerinden DH kullanan grup anahtarı yönetim protokolleri bu bölümde anlatılmaktadır. Ayrık logaritmalar problemi nedeniyle güvenliği ispatlanmış DH anahtar anlaşması yöntemini kullanan bu protokollerin güvenliğinin ispatı da tümevarımla yapılmaktadır.

n üyeli bir ağda amaç grup anahtarı olan $g^{N1*N2*....Nn}$'yi hesaplamaktır.

Karmaşıklık ölçümleri için kullanılan kavramlar:

Çevrim: Toplam çevrim sayısı.

Üs/M_i[1]: Mi tarafından hesaplanan toplam modüler üs sayısı.

Tüm üsler: Tüm üyelerce hesaplanan toplam modüler üs sayısı.

Tekil iletim(Unicast): Tüm üyelerin toplam tekil aktarımlı mesaj sayısı.

Yayın(Broadcast): Tüm üyelerin toplam yayın mesaj sayısı.

Tekil iletim boyutu: Kümülâtif tekil iletim mesaj boyu. (Noktadan noktaya)

Yayın boyutu: Kümülâtif yayın mesaj boyu.

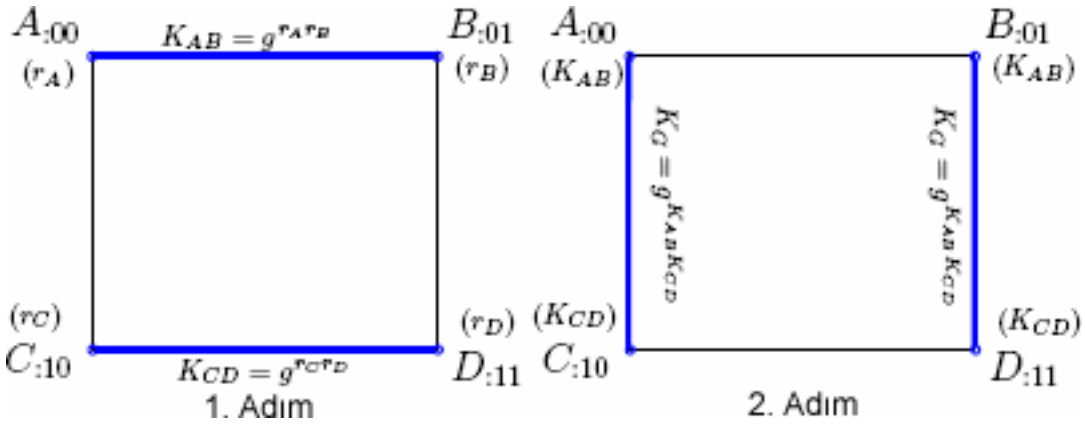
Mesaj uzunluğu: Kümülâtif tekil aktarım ve yayın mesaj boyu.

Tüm protokoller için kullanılan notasyonlar aşağıda verilmiştir. STR[21] ve TGDH[22] için anahtar yolu(key path) ve yan yol(co path) terimleri sıkça kullanılır.

- Anahtar yolu (Key path): M_i üyesinin anahtar yolu, Mi' den köke olan yoldur.
- Yan yol (Copath): M_i üyesinin anahtar yolu üzerinde bulunan düğümlerin kardeş düğümlerinden oluşan yoldur.

4.4.1 2^d - Küp

2^d küp protokolünde[23], 2^d grup üyesi, d boyutlu (d-küp) hiperküpe yerleştirir. Grup üyeleri, d-boyutlu vektör uzayı F_2^d'nin vektörleri olarak ifade edilir. Örnek olarak '2 - küp'ü (2 boyutlu) ele alırsak: 4 üye → A,B,C,D ve 4 üyenin için 2'şer bitlik adresleri sırasıyla: 00, 01, 10 ve 11. Örnek durum Şekil 4.2'degösterilmiştir.



Şekil 4.2: 4 Üye İçin 2^4 – küp Yöntemi

Bu 4 üye arasında ortak olarak kullanılacak grup anahtarını oluşturmak için 2 çevrim gerekir.

- İlk çevrimde ‘A - B’ ve ‘C – D’, DH anahtar değiş tokuş yöntemiyle ortak birer anahtarda(K_{AB}, K_{CD}) anlaşırlar; ortak bir anahtar oluştururlar.
- İkinci çevrimde ‘A-C’ ve ‘B – D’ ilk çevrimde elde ettikleri K_{AB}, K_{CD} anahtarlarını kullanarak K grup anahtarını oluşturur.

$n = 2^d$ grup üyesi olması durumunda;

- Her katılımcının d-bitlik adresi olur. Bu adresler 0’dan 2^d-1’e kadar değer alır.
- Protokolün sonuca ulaşması için d tane çevrim yürütülmesi gerekir.
- j. çevrimde adresi A olan üye ile, adresi $A \oplus 2^{j-1}$ olan üye, DH ile aralarında anlaştıkları bir ortak anahtar oluştururlar. Başka bir ifade ile, j. çevrimde adresleri sadece (j-1). düşük anlamlı bitte farklı olan 2 üye, aralarında DH yöntemi ile anahtar oluşturur.
- j tane çevrim bittiğinde (n-j) yüksek anlamlı bitleri aynı olan düğümler aynı anahtarlıdır; 2^{d-j} anahtar çifti vardır.
- d çevrim sonrasında, tüm üyeler aynı anahtarı paylaşır.

Bu protokol, gruptaki üye sayısı 2’nin kuvveti olduğunda çalışır. Bu protokolü genelleştirmek için aynı makalede[23] 2^d ahtapot protokolü önerilmiştir.

4.4.2 2^d – Ahtapot

Bu protokol[23], 2^d - küp yöntemindeki $n = 2^d$ koşulunun genelleştirilerek $n \in Z^+$ olmak üzere n grup üyesi tarafından yürütülebilmesi için önerilmiştir.

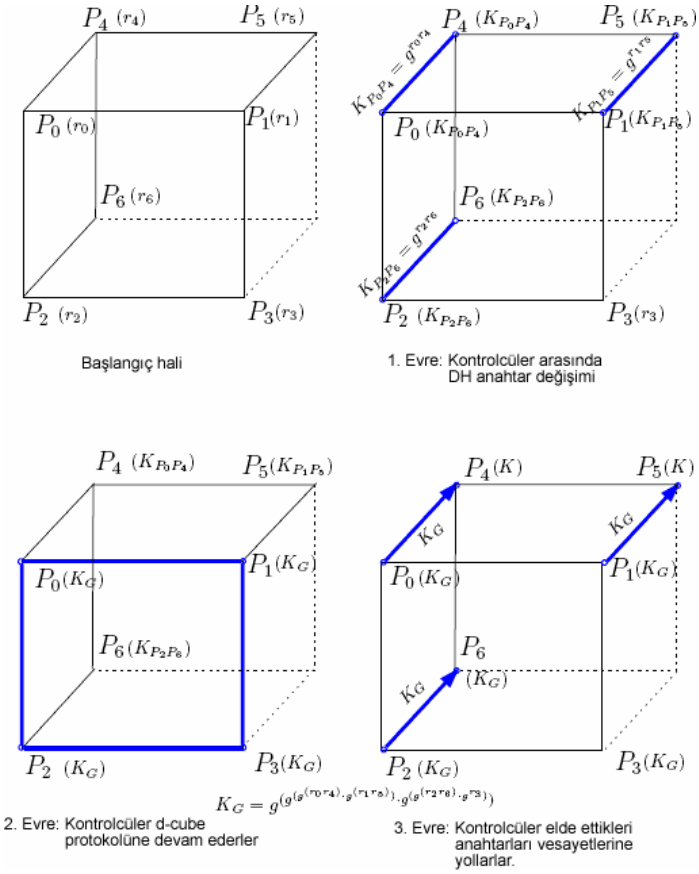
n tane grup üyesinin tamamı, $2^d < n < 2^{d+1}$ olmak üzere d + 1 bitlik adreslerle ifade edilir. Bu adresler desimal olarak [0 , (n–1)] arasında değer alır. İlk 2^d grup üyesi, merkezi denetimci olarak görev yapar. Geri kalan n-2^d katılımcı, 2^d denetimcinin vesayeti olur. A adresli bir vesayet, A>2^d olmak üzere, A-2^d adresli kontrolcünün vesayetidir.

İlk evrede denetimciler vesayetleri ile DH anahtar değiş tokuşu yapar. Daha sonra d-küpte, denetimciler, ilk çevrimde elde ettikleri anahtarı kullanarak protokolü yürütmeye devam ederler. Üçüncü ve son çevrimde ise, denetimciler, ikinci çevrimde elde ettikleri anahtarları, ilk çevrimde elde ettikleri anahtarla şifreleyerek vesayetlerine yollarlar.

Grup anahtarını, vesayetlere şifreleyerek yollamada çeşitli yöntemler izlenebilir. Örneğin;

- İlk evrede, denetimci ve vesayeti K_{AB} anahtarını oluşturur.
- Açık anahtar $PK = g^{K_{AB}}$, grup anahtarı K_G 'dır.
- Denetimci t rasgele sayısını seçer. C_1 ve C_2'yi hesaplar ve vesayete yollar.
- $C_1 = g^t$, $C_2 = K_G.(g^{K_{AB}})^t$ 'dır.
- Vesayet $\frac{C_2}{(C_1)^{K_{AB}}}$ işlemini uygulayarak K_G' yi elde eder. Vesayet K_{AB}' yi bildiği için $(g^t)^{K_{AB}} = C_1^{K_{AB}}$ 'yi hesaplayabilir.
- $\frac{C_2}{(C_1)^{K_{AB}}} = \frac{K_G.(g^{K_{AB}})^t}{(g^t)^{K_{AB}}} = \frac{K_G.g^{K_{AB}.t}}{g^{t.K_{AB}}} = K_G$

Protokolün n = 2, d = 7 parametreleri için yürütülmesi örneği Şekil 4.3'te gösterilmiştir.



Şekil 4.3: $d = 7$, $n = 2$ İçin 2^d Ahtapot Protokolü

4.4.3 Asokan-Ginzboorg Yöntemi

2^d küp ve 2^d ahtapotta asıllama olmadığından güvenliği sağlamanın bir yöntemi sayısal imza kullanmaktır. Sayısal imza için PKI altyapısı gerekir[5]. Tasarsız ağlarda böyle bir altyapının olduğu varsayılamayacağı için [24] makalesinde tasarsız ağlara uygun bir grup anahtar yönetim protokolü önerilmiştir.

2^d küp ve 2^d ahtapotta, yürütülen her 2 taraflı DH yöntemi yerine, 4 bölümlü 2 taraflı DH yöntemi yürütülebilir. Ağda oluşabilecek hatalara karşı, herhangi bir grup üyesi herhangi bir çevrim sırasında seçtiği eşin canlı olup olmadığını bilebilmelidir. 2^d küp ve 2^d ahtapotta böyle bir mekanizma yoktur; bu tip hatalar için çözüm önerilmemiştir. 2^d küp ve 2^d ahtapot yöntemlerinin genişletilmiş ve iyileştirilmiş hali olan Asokan - Ginzboorg yönteminde bu tip durumlar göz önüne alınarak çözüm üretilmiştir.

Eğer bir grup üyesi, bir çevrimde eşinin canlı olmadığını fark ederse canlı olan başka bir eş seçer. Eş seçiminde ise yakın olanın tercih edilmesi önerilir. 2^d küp ve 2^d ahtapotun karmaşıklıkları Tablo 4.1'de karşılaştırılmıştır.

Tablo 4.1: 2^d Küp ve 2^d Ahtapotun Karmaşıklığı

	2^d küp	2^d ahtapot
Çevrimler	d	$d+2$
Tekil iletim boyutu	Nd	$3\left(n-2^d\right)+2^d \bullet d$
Yayın boyutu	0	0
Tüm üsler	$n(2d-1)$	$2^d \bullet (2d+1)$
n: gruptaki üye sayısı, $d=\log_2 n \rightarrow n=2^d$		

Temel hiperküpten farklı olarak, iletişim maliyetini belirlemek için alt çevrim sayısı ölçülür. Katılımcı sayısı olan n, 2'nin kuvveti değil ise; $2^d < n < 2^{d+1}$, canlı eş bulmak için $(d+1)$ çevrim yürütülür. $n=2^d+1$ iken alt çevrimlerin sayısı en fazla 2^d'dir. Katılımcılar farklı şekilde etiketlendirilirse, karmaşıklık önemli ölçüde düşürülür. Örneğin m tane ölü grup üyesi bulunsun. $[0,2(m-1)]$ arası çift sayılar ölü üyeler ile ilişkilendirilsin. Geri kalan adresler, sırayla canlı grup üyelerine verilsin. Kalan her canlı grup üyesi, en fazla 3 denemede 2 taraflı DH yöntemini yürütebileceği bir eş bulur. Burada bahsedilen tekrar etiketleme yöntemiyle, en kötü durum karmaşıklığı $2(d+1)$ çevrim olur.

[24] makalesinde bahsedilen, tekrar etiketlemenin olmadığı bir senaryo göz önüne alınırsa: $2^d < n \le 2^{d+1}$ olmak üzere toplam n adet grup üyesi mevcuttur. m adet grup üyesi ölüdür, $2^k \le m < 2^{k+1}$. $2^k - 1$ ölü grup üyesi k-küp C_1'dedir. Geri kalan ölü üyeler ise k-küp C_2'dedir. C_1 ve C_2, $(k+1)$. çevrimde anahtar değiş tokuşu yapacaklardır. Ana çevrim i' deki alt çevrim sayısı $(1 \le i \le k+1)$, en fazla 2^{i-1}' dır. (k+1). çevrimde C_1'deki ölü katılımcıların sayısı $2^k - 1$'dir. C_2'deki bazı katılımcılar, tek canlı katılımcıyı bulmadan önce, C_1'deki tüm ölü katılımcıları seçmeyi deneyebilir. Bu nedenle, ana çevrim $(k+1)$ için, alt çevrim sayısının maksimumu k' dır. Çevrimde, $1 \le i < k$, tek canlı katılımcı pes etmeden önce tüm olası katılımcıları deneyecektir. Ana çevrim $(k+2)$' den $(d+1)$'e kadar, maksimal alt çevrim $(m+1)$'dır; çünkü bu çevrimlerin her birinde, her zaman m adet ölü grup

üyesi içeren bir alt küp vardır. Aynı canlı grup üyesi, m adet grup üyesinden her birini kendi DH yönteminin yürütülmesi sona ermeden önce seçebilir. Böylece $(m+1)$ çevrimde tamamlayabilir. Bu nedenle, en fazla, $\sum_{i=1}^{k+1} 2^{i-1} + (d-k)(m+1) = 2^{k+1} - 1 + (d-k)(m+1)$ çevrim gerekir. Bu sayı, m hatalı grup üyesinin konumlarından bağımsız olarak üst sınırdır. Canlı katılımcılar için en fazla $2^{k+1} - 1 + (d-k)(m+1)$ tekil aktarımlı mesaj gerekir. $(n-m)$grup üyesi canlıdır. Bu nedenle grup anahtarının oluşturulması süresince en fazla $(n-m)(k^{k+1} - 1 + (d-k)(m+1))$ tekil aktarımlı mesaj yollanır. Her canlı grup üyesi $(d+1)$başarılı anahtar değiş tokuş işlemi yapar. Her canlı grup üyesi en fazla $2^{k+1} - 1 + (d-k)(m+1) - d - 2$ başarısız anahtar değiş tokuşu yapar. Üs almanın, sadece başarılı DH değiş tokuşunda yapıldığı kabul edilirse; her canlı grup üyesi tarafından d adet üs hesaplanacaktır. En kötü durumda, her ana çevrimde en fazla $\max(n-m, \#altçevrim)$ seri üs alma vardır. Toplam seri üs alma sayısı $\min(n-m, \sum_{i=1}^{k+1} \max(n-m, 2^{i-1}) + (d-k).\max(n-m, n+m))$. Asokan – Ginzboorg protokolünün en kötü durum karmaşıklığı Tablo 4.2'de gösterilmiştir.

Tablo 4.2: Asokan-Ginzboorg En Kötü Durum Karmaşıklık Tablosu

Çevrim	$2^{k+1} - 1 + (d-k)(m+1)$
Tekil aktarımlı mesajlar	$(n-m)(k^{k+1} - 1 + (d-k)(m+1))$
Yayın mesajları	0
Üs/üye	2d+1
∑tekil aktarım	$(n-m)(k^{k+1} - 1 + (d-k)(m+1))$
∑yayın	0
∑üs	(n-m)(2d-1)

4.4.4 Cliques

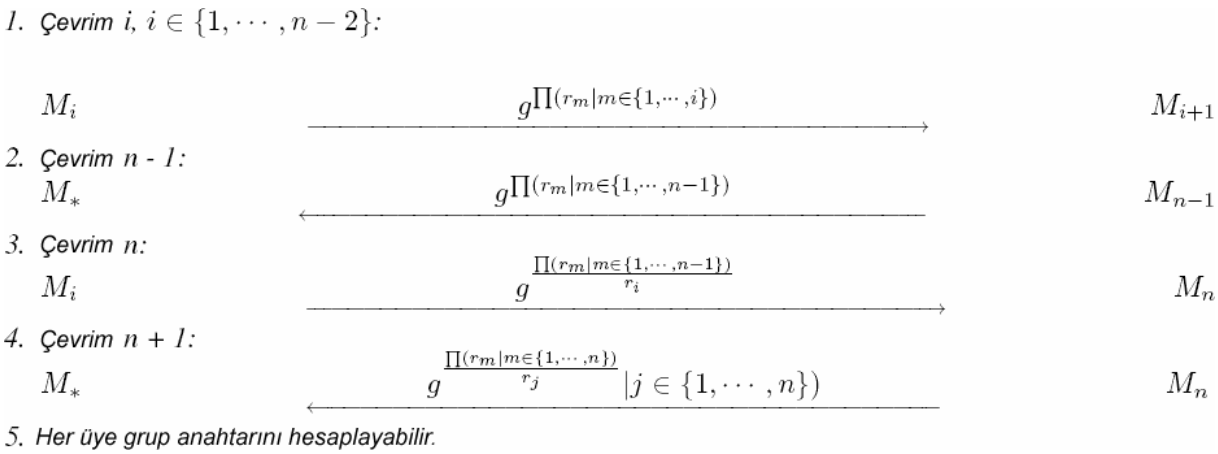
Bu protokolde[25], ilk anahtar anlaşması oluşturulması(IKA), üye katılma ve ayrılma işlemleri göz önüne alınmıştır. DH anahtar değiş tokuş yöntemini kullanır. n üyeden oluşan bir grupta her M_C üyesini bilmesi gereken bilgiler: $g^{\Pi(r_m | m \in \{1,...n\} \{j\})} \mid j \in \{1,...n\} \{c\}$, $g^{\Pi(r_m | m \in \{1,...n\} \{c\})}$, $g^{\Pi(r_m | m \in \{1,...n\})}$, kendi seçtiği rasgele sayı r_c ve grup anahtarıdır.

Her üyenin bellek maliyeti 2 gizli anahtar (r_c ve grup anahtarı) ve $(n+1)$ açık anahtardır.

4.4.4.1 IKA Protokolü

IKA protokolü için 2 alternatif sunmuştur. Birinde GDH2 diğerinde ise GDH3 kullanılmaktadır [25] . GDH3 kullanılan IKA protokolü geniş gruplar veya sınırlı kaynağı olan düğümlerden oluşan gruplar için geliştirilmiştir. GDH2 kullanılan IKA protokolünde, M_i üyesinin $(i+1)$ üs alma işlemi yapması gerekir. GDH3 kullanan IKA' da ise bu sayı: $i \le (n-2)$ için 4, M_{n-1} için 2, M_n n ' dir.

İşlem maliyeti düşük olduğu için burada GDH3 kullanılan IKA2 Şekil 4.4'te açıklanmıştır.



Şekil 4.4: Cliques IKA2 Protokolü

IKA2 protokolü gerçekleştirildikten sonra tüm grupça paylaşılan bir grup anahtarı elde edilir: $K_G = g^{\prod_{i=1}^{n} r_i}$. GDH3 kullanan IKA2 protokolünün karmaşıklığı Tablo 4.3'te gösterilmiştir.

Tablo 4.3: GDH3 Kullanan IKA2 Protokolünün Karmaşık Tablosu.

Çevrim	n+1
Tekil iletim	2n-3
Yayın	2
Üs/M_i	$3, i \in \{1,...,n-2\}$ $2, i = n-1$ $n, i = n$
Tekil iletim boyutu	2n-3
Yayın boyutu	n+1
Tüm üs almalar	5n-6

4.4.4.2 Katılma Protokolü

Ekleme işleminin yapılabilmesi için bir grup yöneticisine ihtiyaç vardır. Gruptaki herhangi bir üye bu görevi yerine getirebilir. Grup yöneticisi, diğer üyelere nazaran daha yüksek kapasiteye sahip bir düğüm olmalıdır. Diyelim ki M_c, grup yöneticisi; M_{n+1} de gruba eklenecek bir üye olsun.

İlk adımda, M_c yeni bir rasgele $\hat{r}_c$ sayısı seçer. Daha önce seçtiği r_c' nin yerini $\hat{r}_c\, r_c \bmod |G|$ alır; böylece anahtar bağımsızlığı sağlanmış olur. M_{n+1} üyesine M_c tarafından mesaj yollanır. M_{n+1}, rastsal olarak kendi gizli anahtarı r_{n+1}'i üretir; bu bilgiyi mesajın içine gömerek grup üyelerine yayın yapar. Cliques katılma protokolü Şekil 4.5'te çevrim sırası ile açıklanmıştır.

1. Çevrim 1: M_c kontrolcüsü

- Gizli anahtarını güncelleyerek yeni mesaja dahil eder
- bu mesajı M_{n+1}'e yollar.

$$M_c \xrightarrow{\substack{(g^{\hat{r_c}} \prod(r_m | m \in \{1,\cdots,n\}\setminus\{j\}) | j \in \{1,\cdots,n\}), \\ g^{\hat{r_c}} \prod(r_m | m \in \{1,\cdots,n\}}} M_{n+1}.$$

2. Çevrim 2: M_{n+1},

- r_{n+1} gizli anahtarını oluşturur
- gizli anahtarı yeni mesaja gömer,
- mesajı yayınlar.

$$M_* \xleftarrow{(g^{\hat{r_c}} \prod(r_m | m \in \{1,\cdots,n+1\}\setminus\{j\}) | j \in \{1,\cdots,n+1\})} M_{n+1}.$$

3. Her eleman grup anahtarını hesaplar.

Şekil 4.5: Cliques Katılma Protokolü

Protokolü yürüttükten sonra, tüm üyeler yeni grup anahtarını elde etmiş olur. $K_{G_{NEW}} = g^{\hat{r_C} \prod_{i=1}^{n+1} r_i}$. Ekleme protokolünün karmaşıklığı Tablo 4.4'te gösterilmiştir.

Tablo 4.4: Ekleme Protokolünün Karmaşıklık Tablosu

Çevrim	2
Tekil iletim	1
Yayın	1
Üs/M_i	n+1 grup yöneticisi ve yeni üye 1 diğer üyeler
Tekil iletim boyutu	n+1
Yayın boyutu	n+1
Tüm üs almalar	3n+1

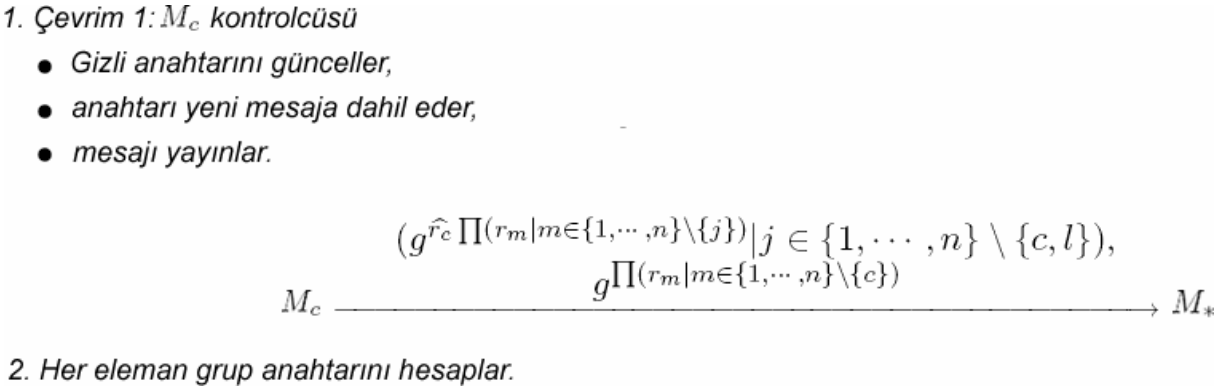
Tablo 4.4'den de görüldüğü üzere, işlem yükü, yeni üye ve grup yöneticisi üzerinde yoğunlaşmıştır. Adil bir dağılım söz konusu değildir.

4.4.4.3 Ayrılma Protokolü

Ayrılma protokolü tek çevrimde bittiğinden diğer CLIQUES protokollerine göre basit kalır. M_c grup yöneticisi, M_l de gruptan ayrılacak üye olsun.

1. M_c, kendi gizli anahtarı r_c'yi $\hat{r_c}$ r_c ile günceller.
2. M_c, yeni anahtarını da barındıran mesajı yayınlar.

Ayrılma protokolü Şekil 4.6'da ayrıntılı olarak açıklanmıştır.



Şekil 4.6: Cliques Ayrılma Protokolü

Ayrılma protokolü yürütüldükten sonra, geri kalan tüm üyeler grup anahtarını hesaplayabilir. $K_{G_{NEW}} = g^{\hat{r}_c \prod_{i=1}^{n} r_i}$. l üyesinin anahtara katılımı olan r_e, yeni grup anahtarında da çarpan olarak bulunmaktadır; fakat M_e, grup anahtarını hesaplayamaz; çünkü alt anahtarlardan $g^{\hat{r}_c \prod(r_m|m\in\{1,\ldots n\}\ \{l\})}$ alt anahtar kümesinde artık yoktur. Ayrılma protokolünün karmaşıklığı Tablo 4.5'te gösterilmiştir.

Tablo 4.5: Ayrılma Protokolünün Karmaşıklık Tablosu

Çevrim	1
Tekil iletim	0
Yayın	1
Üs/M_i	$(n-1)$, grup yöneticisi için 1 , diğer üyeler için
Tekil iletim boyutu	0
Yayın boyutu	$(n-1)$
Tüm üsler	$(2n-3)$

4.4.4.4 Birleşme/Çoklu Katılım Protokolü

CLIQUES de birleşme protokolü tanımlanmamıştır. 2 grubu birleştirmek için 2 yöntem önerilmiştir.

1. Üye sayısı küçük olan grubun üyeleri diğer gruba eklenecek yeni üyeler olarak görülebilir.
2. IKA uygulayarak, baştan yeni bir grup oluşturmak.

N üyeli bir grup için, k üye eklenecekse, bu üyeler M_{n+1}'den M_{n+k}'ya kadar etiketlenir. İlk çevrim ekleme protokolündeki ile aynıdır. Cliques birleşme protokolü Şekil 4.7'de ayrıntılı olarak açıklanmıştır.

1. Çevrim 1: M_c *kontrolcüsü*

- *Gizli anahtarını güncelleyerek yeni mesaja dahil eder*
- *bu mesajı* M_{n+1}*'e yollar.*

$$M_c \xrightarrow{\begin{array}{c}(g^{\widehat{r_c}\prod(r_m|m\in\{1,\cdots,n\}\setminus\{j\})}|j\in\{1,\cdots,n\}),\\ g^{\widehat{r_c}\prod(r_m|m\in\{1,\cdots,n\}}\end{array}} M_{n+1}.$$

2. Çevrim 2'den k'ya: i. çevrim, $2 \le i \le k, M_{n+i-1}$ *üyesi*

- *Gizli anahtarını yeni mesaja dahil eder,*
- M_{n+i} *üyesine mesajı gönderir.*

$$M_{n+i-1} \xrightarrow{\begin{array}{c}(g^{\widehat{r_c}\prod(r_m|m\in\{1,\cdots,n+i-1\}\setminus\{j\})}|j\in\{1,\cdots,n+i-1\}),\\ g^{\widehat{r_c}\prod(r_m|m\in\{1,\cdots,n+i-1\}}\end{array}} M_{n+i}$$

3. Çevrim k+1: M_{n+k} *üyesi*

- *Gizli anahtarını yeni mesaja dahil eder,*
- *mesajı yayınlar.*

$$M_* \xleftarrow{(g^{\widehat{r_c}\prod(r_m|m\in\{1,\cdots,n+k\}\setminus\{j\})}|j\in\{1,\cdots,n+k\})} M_{n+k}$$

4. Her eleman grup anahtarını hesaplar.

Şekil 4.7: Cliques Birleşme/Çoklu Katılım Protokolü

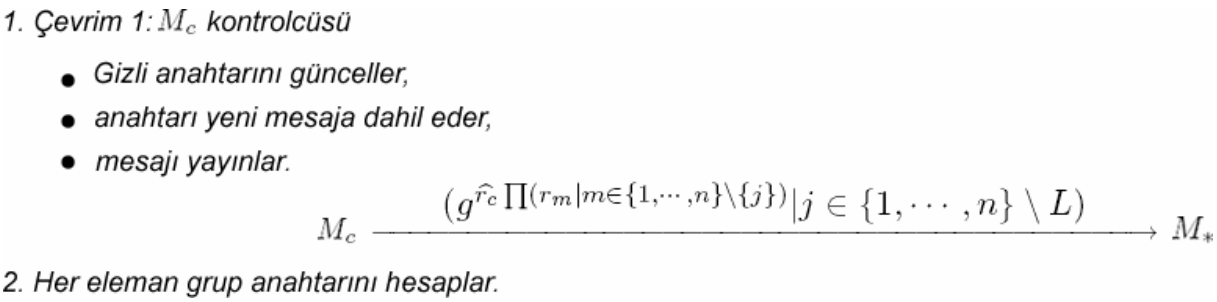
Bu protokolü yürüttükten sonra tüm üyeler yeni grup anahtarını hesaplayabilir. $K_{G_{NEW}} = g^{\hat{r_C} \prod_{i=1}^{n+k} r_i}$. Cliques birleşme protokolünün karmaşıklığı Tablo 4.6'da gösterilmiştir.

Tablo 4.6: Birleşme Protokolünün Karmaşıklık Tablosu

Çevrim	k+1
Tekil iletim	K
Yayın	1
Üs/M_i	$(n+2)$, grup yöneticisi için 1 , i∈ {1,...,n}\{c} için i+1 , i∈ {n+1,...,n+k} için
Tekil iletim boyutu	$(\frac{k^2+2nk+k}{2})$
Yayın boyutu	$(k+n)$
Tüm üsler	$(\frac{k^2+2nk+3k+4n+2}{2})$
k: yeni üye sayısı	

4.4.4.5 Bölünme Protokolü

Bölünme protokolü, ayrılma protokolüne benzer. Tek farkı, grup yöneticisi daha az anahtar hesaplar. L, gruptan ayrılan tüm üyelerin kümesi olsun. Buna göre bölünme protokolü Şekil 4.8'de ayrıntılı olarak açıklanmıştır.



Şekil 4.8: Cliques Bölünme Protokolü

Grup yöneticisi, yeni anahtarını hesaplar. Yeni alt anahtarları (L kümesindeki üyelerin alt anahtarlarını hesaplamaz) gruba yayınlar. Her üye grup anahtarını

hesaplar. Protokolün tamamı yürütüldükten sonra, grupta kalan üyeler yeni grup anahtarını hesaplayabilir. $K_{G_{NEW}} = g^{\hat{r}_C \prod_{i=1}^{n} r_i}$

Bir grubun birden çok küçük gruplara bölünmesi durumu için, diğer küçük gruplara ait olan elemanların grubu terk ettiğini düşünebiliriz. Her küçük grup kendi başına işler. Örneğin, $M_1,M_2,\ldots,M_6$ üyelerinden oluşan bir grup, G_1 grubu (M_1,M_3,M_5) ve G_2 grubu (M_2,M_4,M_6), şeklinde 2 gruba ayrılmak istesin. G_1 grubunda M_3, G_2 grubunda M_4 grup yöneticisi olsun. Yeni seçilen gizli anahtarlar da r_3 ve r_4 olsun. G_1 grubunda yöneticinin yayın mesajı;

$(g^{r_2(r_3\hat{r}_3)r_4r_5r_6}; g^{r_1r_2(r_3\hat{r}_3)r_4r_6}; g^{r_1r_2\hat{r}_3r_4r_5r_6})$ ve grup anahtarı $K_{G_{NEW}} = g^{r_1r_2(\hat{r}_3r_3)r_4r_5r_6}$,

G_2 grubunun yayın mesajı, $(g^{r_1r_2(r_4\hat{r}_4)r_5r_6}; g^{r_1r_2r_3(r_4\hat{r}_4)r_5}; g^{r_1r_2r_2\hat{r}_4r_5r_6})$ ve grup anahtarı $K_{G_{NEW}} = g^{r_1r_2r_3(\hat{r}_4r_4)r_5r_6}$ olur. Bölünme protokolünün karmaşıklığı Tablo 4.7'de gösterilmiştir.

Tablo 4.7: Bölünme Protokolünün Karmaşıklık Tablosu

Çevrim	1
Tekil iletim	0
Yayın	1
Üs/M_i	$(n-p)$, grup yöneticisi için 1 , geri kalan tüm üyeler için
Tekil iletim boyutu	0
Yayın boyutu	1
Tüm üsler	$(2n-2p-1)$

4.4.4.6 Yenileme Protokolü

Yenilemeyi yapacak eleman M_r (gizli anahtarını en eski güncelleyen), yeni bir gizli anahtar seçer: $\hat{r}_r$. Bu yeni bilgiyi gruba yollar. Protokolü yürüttükten sonra, her üye grup anahtarını hesaplar. $K_{G_{NEW}} = g^{\hat{r}_r \prod(r_m | m \in \{1,\ldots,n\})}$. Yenileme protokolünün karmaşıklığı Tablo 4.8'de gösterilmiştir.

Tablo 4.8: Yenileme Protokolünün Karmaşıklık Tablosu

Çevrim	1
Tekil iletim	0
Yayın	1
Üs/M_i	n , güncelleyici için 1 , geri kalan tüm üyeler için
Tekil iletim boyutu	0
Yayın boyutu	N
Tüm üsler	$(2n-1)$

4.4.5 STR Yöntemi

[26] yöntemi, [21] yöntemini dinamik grup işlemlerini kapsayacak şekilde genişletmiştir. STR[21], iletişim gecikmesine karşı iyileştirildiğinden, gecikmesi yüksek ağlarda kullanılması uygundur. STR için kullanılacak notasyon:

br_i: M_i'nin açık, körlenmiş oturum anahtarı, ÖR: $g^{r_i} \bmod p$

ki: $M_1 \ldots M_i$ arasında paylaşılan anahtar.

bk_i: açık körlenmiş k_i, $g^{k_i} \bmod p$

N_i: ağaç düğümü i

IN_i: i. Seviyedeki ara düğüm

LN_i: M_i'ye karşı düşen yaprak düğüm

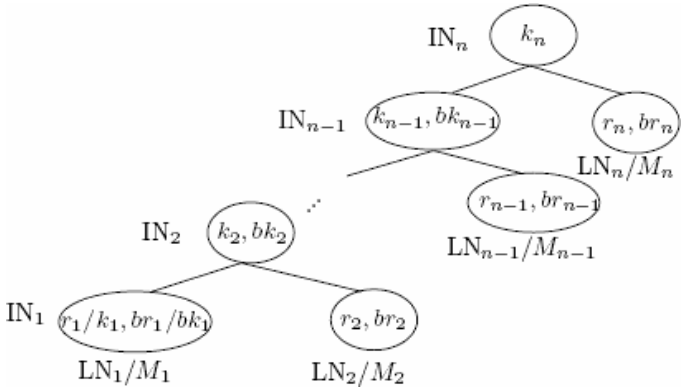
BR_i^*: M_i^*'nin körlenmiş oturum anahtarları kümesi

STR, grubu yönetmek için bir anahtar ağacı kümesi kullanır. Ağaçta 2 tip düğüm vardır; yaprak ve ara düğüm. Her M_i grup üyesi bir yaprak düğüm LN_i ile ilişkilendirilir. Ara düğüm ise IN_i ile gösterilir. Her ara düğümün 2 çocuğu vardır. Her yaprak düğüm LN_i rasgele bir r_i oturum gizli anahtarı yaratır ve bu gizli anahtara karşı düşen körlenmiş oturum anahtarı üretir; $b_{r_i} = g^{r_i}$. Bir ara düğüm IN_i için de

aynı şekilde, gizli anahtar k_i ve buna karşı düşen körlenmiş anahtar $b_{k_i} = g^{k_i}$ vardır. Farklı olarak; k_i rasgele seçilmez, 2 çocuğu arasındaki DH anahtar değiş tokuş işleminin sonucudur. $k_i = (b_{k_{i-1}})^{r_i} \equiv g^{k_{i-1}r_i} \bmod p, i > 1\ (b_{k_{i-1}})^{r_i} = (b_{r_i})^{k_{i-1}}$

En yüksek indisli ara düğüm kök düğüm olarak adlandırılır. Kök düğümün gizli anahtarı paylaşılan grup anahtarıdır. N üyeli bir grup için, kök düğüm IN_n, grup anahtarı k_n olmak üzere, grup anahtarı aşağıdaki şekilde hesaplanır.

$(N = 4)$, grup anahtarı, $K_G = k_4 = g^{(r_4 g^{(r_3 g^{(r_2 r_1)})})}$. Örnek bir STR ağacı Şekil 4.9'da gösterilmiştir.



Şekil 4.9: STR Ağacı

n üyeli yukarıdaki anahtar ağcına bakılırsa M_c üyesi, $1 \leq i \leq n$, kendi oturum gizli anahtarını ve tüm körlenmiş anahtarları ve ebeveyn düğümünden köke kadar olan yoldaki anahtarları bilmelidir. Saklaması gereken bilgiler:

- r_c, br_i, bk_i (i=1,…,n (br_1=bk_1) için)
- k_i (i=c,...,n için)

M_c'nin bellek maliyeti, $c \geq 2$ için n-c+2 anahtar, c=1 için n anahtar, tüm üyeler için 2n-2 anahtardır.

4.4.5.1 Kurulum Protokolü

Grup oluşturmak isteyen üyeler birtakım kıstaslara göre sıralanabilir. Anahtar ağacının yapısı bu sıradan yola çıkılarak oluşturulabilir. Her i üyesi (yaprak düğüm LN_i) için gizli anahtar r_i, körlenmiş anahtar da br_i olsun. İşlemler aşağıdaki şekilde gerçekleşir:

1. Çevrim 1:

   Her M_i, $i \in \{1,\ldots,n\}$ körlenmiş oturum anahtarını yayınlar.

   M_i —— $(br_i) \rightarrow M^{*k}$

2. Çevrim 2:

   Üye M_1 rekürsif olarak $k_i = (br_i)^{k_{i-1}}$ ve $bk_i = g^{k_i}, \forall i \in \{2,3,\ldots n-1\}$ ve $k_n = (br_n)^{k_{n-1}}$ hesaplar. Güncellenmiş ağacı tüm körlenmiş anahtarlarla birlikte yayınlar. M_1 —— $(\hat{T}_{1(BK_1^*)}) \rightarrow M_*$

3. Her M_i üyesi, $i \geq 2$, $k_i = (bk_{i-1})^{r_j}$ ve sonra rekürsif olarak $k_j = (br_{ji})^{k_{j-1}}$ $\forall j \in [i+1, n]$ hesaplar.

Kurulum protokolünün karmaşıklığı Tablo 4.9'da gösterilmiştir.

Tablo 4.9: Kurulum Protokolünün Karmaşıklık Tablosu

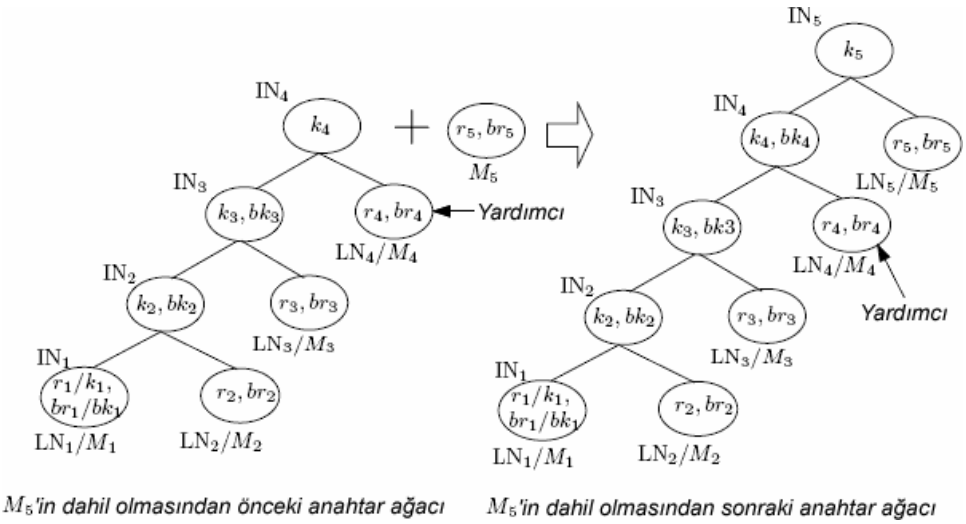
Çevrim	2
Tekil iletim	0
Yayın	n+1
Üs/M_i	$(2n-1)$, $i=1$ için $n-i+1$, $2 \leq i \leq n$
Tekil iletim boyutu	0
Yayın boyutu	$2n-1$
Tüm üsler	$\frac{n^2+3n-2}{2}$

4.4.5.2 Katılma Protokolü

n üyeli bir gruba M_{n+1} elemanının katılması durumunda anahtar ağacı, $n = n + 1$ yapılarak ve yeni bir ara düğüm olan IN_n eklenerek güncellenir. IN_n'in 2 çocuğu vardır;

1. Sol çocuğu, önceki T_i ağacının kök düğümü IN_{n-1}
2. Sağ çocuğu yeni yaprak düğüm (M_{n+1}'e karşı düşen) LN_n

Bu yeni ara düğüm yeni ağacın kök düğümü olacaktır. Şekil 4.10'da örnek olarak 4 üyeli bir ağaca beşinci bir üyenin eklenmesi gösterilmiştir.



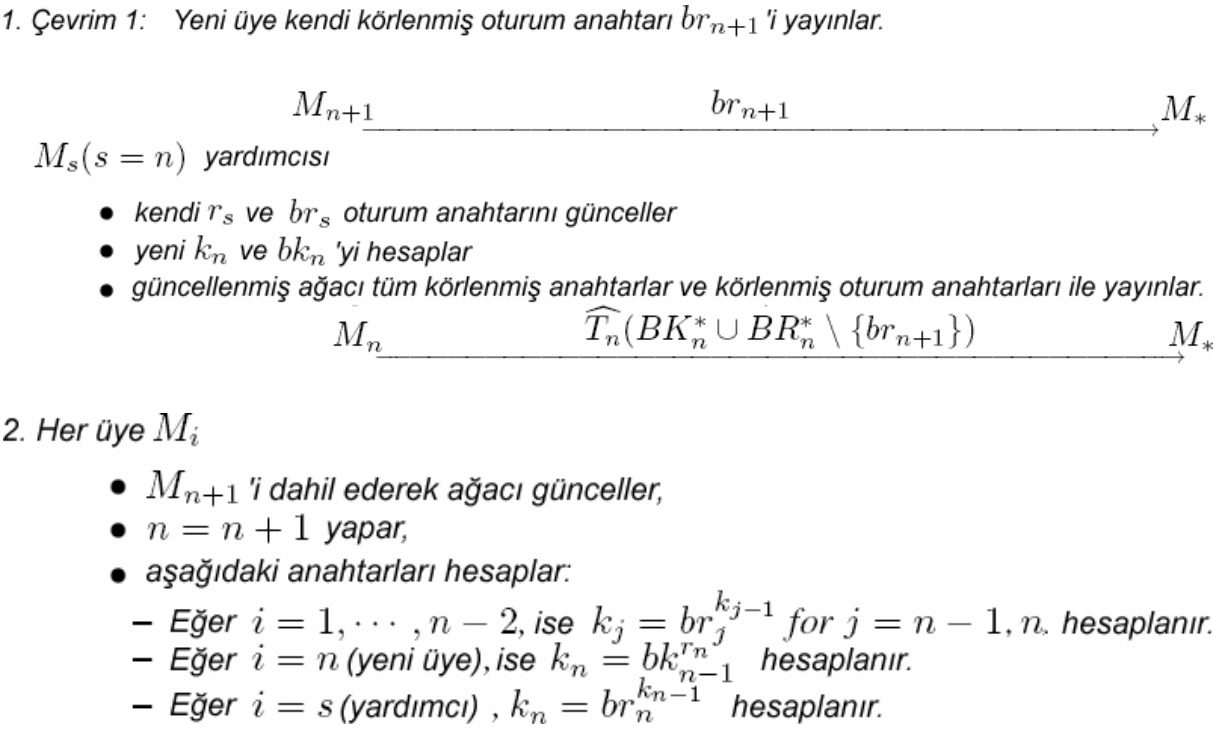
Şekil 4.10: 4 Üyeli Ağaca 5. Üyenin Eklenmesi

Ekleme işlemiyle ilgili olarak M_n üyesi yardımcı düğüm olarak seçilir. Yeni üye M_{n+1}, kendi körlenmiş anahtarını(bk_{n+1}) içeren katılma istek mesajını gruba yayınlar. Tüm M_i üyeleri $(1 \le i \le n)$ yeni grup anahtarını hesaplayabilir. Yardımcı düğüm, M_{n+1}'e tüm körlenmiş anahtarları yollar. Bu mesaj sayesinde, yeni düğüm M_{n+1} de grup anahtarını hesaplayabilir. Bu katılma protokolü, anahtar bağımsızlığı ilkesini sağlamaz. Önceki grup anahtarını biliyorsak, yeni grup anahtarını hesaplayabiliriz. 4 üyeli bir gruba 5. elemanın eklenmesi durumunu göz önüne alırsak,

Eski grup anahtarı $= g^{(r_4 g^{(r_3 g^{(r_2 r_1)})})}$

Yeni grup anahtarı $= g^{(r_5 (r_4 g^{(r_3 g^{(r_2 r_1)})}))} = g^{r_5 (\text{eski grup anahtarı})} = (bk_5)^{(k_4)}$

Bu problemin üstesinden gelmek üzere [26], yardımcı düğümün kendi gizli anahtarını güncellemesini önermektedir. Anahtar bağımsızlığı sağlanan STR protokolünde gruba üye katılması işlemi Şekil 4.11'de açıklanmıştır.



Şekil 4.11: Geliştirilmiş STR Protokolünde Gruba Katılım İşlemi

Tablo 4.10'da STR katılım protokolünün karmaşıklığı gösterilmiştir.

Tablo 4.10: STR Katılım Protokolünün Karmaşıklık Tablosu

Çevrim	2
Tekil iletim	0
Yayın	2
Üs/M_i	4, yardımcı düğüm için 1, yeni üye için 2, diğer üyeler için
Tekil iletim boyutu	0
Yayın boyutu	2n

4.4.5.3 Ayrılma Protokolü

STR'deki ayrılma protokolü sadece 1 çevrim gerektirir. n üyeli bir grup olsun. M_ℓ üyesi $(1 \le \ell \le n)$ gruptan ayrılmak isterse, kendi oturum anahtarını güncelleyecek yardımcı bir M_s düğümü gerekir. $\ell > 1$ ise; yardımcı düğüm ayrılan üyenin altındaki düğümdür $(M_{\ell-1})$. Aksi durumda yardımcı düğüm M_2 düğümüdür. Ağaç yeniden etiketlendirilip güncelleneceği için, eğer M_1 grubu terk ediyorsa, yardımcı düğüm yeniden etiketlendirme işlemi bittikten sonra yine M_1'dir.

Grup iletişim sistemi tarafından bildirilen ayrılma işleminden sonra, geri kalan üye LN_l ve IN_l düğümlerini silerek anahtar ağaçlarını günceller. M_ℓ 'nin üzerinde yer alan düğümleri yeniden numaralandırırlar. Protokolün işleyişi:

1. Çevrim 1:

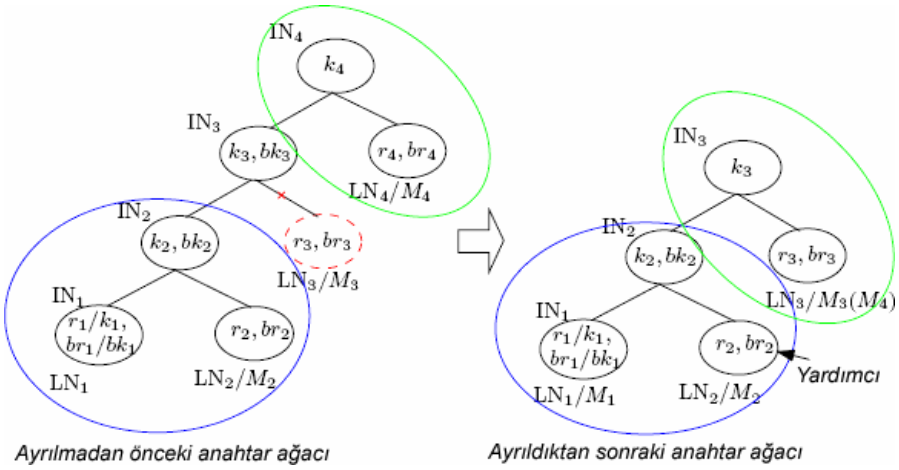
Her üye ayrılan üye M_l'yi silerek ağacı günceller ve n = n − 1 yapar. Yardımcı M_s ek olarak,

- kendi r_s ve br_s oturum anahtarını günceller
- k_n, k_i ve bk_i'yi hesaplar, $\forall i \in [\max(2,s), n-1]$
- güncellenmiş ağacı tüm körlenmiş anahtarlar ve br_s ile yayınlar.

2. Her üye Mi,

- i < s ise, $k_j = br_j^{k_{j-1}}, \forall j \in [s, n].$
- i > s ise, $k_i = bk_{i-1}^{r_j}$, ve $k_i = bk_{i-1}^{r_j}, \forall j \in [i+1, n].$

Şekil 4.12'de 4 üyeli bir STR ağacından bir üyenin ayrılması açıklanmıştır.



Şekil 4.12: STR'de 4 Üyeli Bir Grup İçin Gruptan Bir Eleman Ayrılması.

Şekil 4.12'de de görüldüğü gibi ayrılacak düğümün üstündekiler tekrar numaralandırılır. STR ayrılma protokolünün karmaşıklığı Tablo 4.11'de gösterilmiştir.

Tablo 4.11: STR Ayrılma Protokolünün Karmaşıklık Tablosu

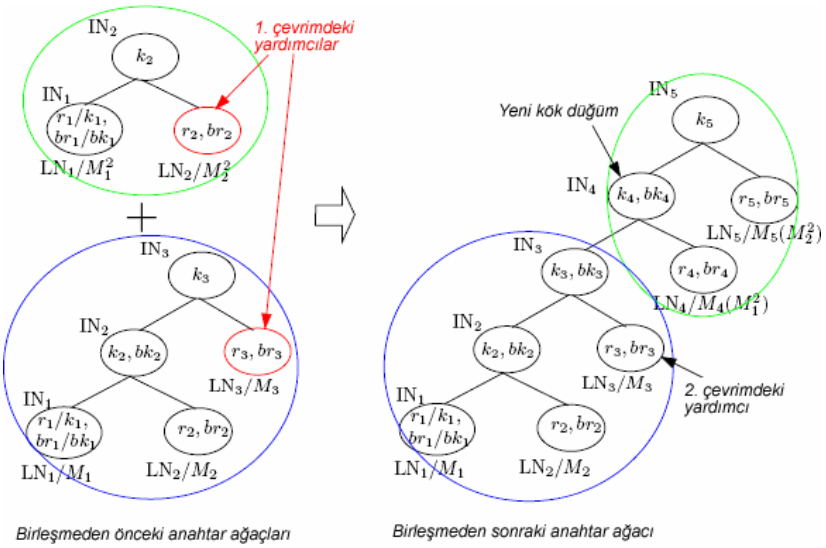
	$s=1$	$s \geq 2$
Çevrim	1	1
Tekil iletim	0	0
Yayın	1	1
Üs/M_i	$2n-4$, yardımcı düğüm için $n-i$, diğer üyeler için	2(n-s) , yardımcı düğüm için $n-i$, M_i için, i > s $n-s$, M_i için, i < s
Tekil iletim boyutu	0	n-1
Yayın boyutu	n-2	
Tüm üsler	$\frac{n^2+n-6}{2}$	$\frac{n^2+n-s-s^2}{2}$
LN_s	Yardımcı düğüm	

4.4.5.4 Birleşme Protokolü

Birleşme protokolü aynı zamanda çoklu katılım için de bir çözüm teşkil etmektedir. Daha küçük olan grup, büyük grupla birleştirilir. Daha küçük olan anahtar ağacı, büyük olanın üstüne doğrudan eklenir. Grup üye sayısı eşit ise, belirleyeceğimiz bir kıstasa göre sıralama yapabiliriz. 2 çocuğu olan yeni bir ara düğüm yaratılır. Büyük

ağacın kökü, bu N düğümün sol çocuğu olur. Küçük ağaçların en küçük sayı etiketli yaprak düğümü N düğümünün sağ çocuğu olur. Küçük ağacın kökü, yeni ağacın da kök düğümü olur.

Her grup için 1 yardımcı düğüme ihtiyaç vardır. En üstteki yaprak düğüm yardımcı düğüm olarak seçilir. Her iki yardımcı düğüm de anahtar ağaçlarındaki körlenmiş anahtarları birbirlerine yollar. Daha büyük olan anahtar ağacındaki yardımcı düğüm ikinci çevrimde yardımcı düğüm olur. Bu yeni yardımcı düğüm öncelikle kendi oturum anahtarını yeniler. Daha sonra tüm gizli ve körlenmiş anahtar çiftlerini köke kadar hesaplar. Hesaplama bittikten sonra, yeni ağacı tüm körlenmiş anahtar ve körlenmiş oturum anahtarları ile birlikte yayınlar. Şekil 4.13'te STR birleşme protokolüne bir örnek verilmiştir.



Şekil 4.13: STR Birleşme Protokolüne Örnek

2 ağaç birleştirilecek olsun. Daha büyük olanı, T^1, n_1 üyeli ve diğeri ise, T^2, n_2 üyeli olsun. Bu ağaçların yardımcı düğümleri ise M_{s1} ve M_{s2} olsun. Bu 2 grubu birleştirmek için yapılacak işler:

1. Çevrim 1:

Her iki yardımcı düğüm M_{s_1} ve M_{s_2} T^1 ve T^2 ağacını tüm körlenmiş anahtar ve oturum rastsalları ile birlikte sırayla değiş tokuş eder.

2. Çevrim 2: Yardımcı düğüm M_s (M_{s_1})

- Kendi oturum anahtarını günceller,
- $(k_i, bk_i), i = n_1 - 1, \cdots, n_1 + n_2 - 1$ *ve* $k_{n_1+n_2}$ 'yi hesaplar
- Güncellenmiş ağacı tüm körlenmiş anahtarlarla yayınlar.

3. Her M_i üyesi,

- $i < s$ *ise* , $k_j = br_j^{k_j - 1}, \forall j \in [s, n]$ 'i hesaplar.
- $i > s$ *ise, önce* $k_i = bk_{i-1})^{r_i}$ *'yi, sonra* $k_j = br_j^{k_j - 1}, \forall j \in [i+1, n]$ 'i hesaplar.

Birleşme protokolü karmaşıklığı Tablo 4.12'de gösterilmiştir.

Tablo 4.12: STR Birleşme Protokolü Karmaşıklık Tablosu

Çevrim	2
Teki iletim	2
Yayın	1
Üs/M_i	$2k+2$, M_n için $k+1$, M_i için, i < n $n-i+1$, diğer M_i'ler için
Tekil iletim boyutu	$2(n+k-2)$
Yayın boyutu	$2(n+k-1)$
Tüm üsler	$\frac{k(2n+k+3)}{2}+n+1$

Birleşme protokolü birden fazla grubu birleştirmek üzere genelleştirilebilir. k tane grup birleştirilecek olsun. Tüm k tane grup büyükten küçüğe doğru T^1-T^k şeklinde sıralanmış olsun. Üyeleri ise n_1-n_k şeklinde gösterilsin. Her t_i ağacının en üstteki yaprak düğümü M_s ile ifade edilen yardımcı düğüm olarak belirlenir.

2 çeşit çoklu birleşim düşünülebilir. İlki, ardışıl 2-grup birleşmesidir. Bu yöntemde ilk 2 grup, birleşme protokolünü yürüterek güncellenmiş $\hat{T}_M$ anahtar ağacını oluşturur. $\hat{T}_M$ grubu daha sonra 3. grupla 2 gruplu birleşme protokolünün yürütür. Tüm gruplar birleştirilene kadar bu işleme devam edilir. Bu yöntemin en önemli avantajı alt protokol olarak 2 gruplu birleşmeyi çağırabilmesidir; fakat yeni grup (tüm ağaçların birleşiminden oluşan) , $2(k-1)$ çevriminden sonra oluşturulur.

Diğer yöntem de, m-gruplu birleşmedir:

1. Çevrim 1:

Her yardımcı düğüm M_{s_i} anahtar ağacını tüm körlenmiş anahtarlarıyla birlikte yayınlar. $i, 1 \le i \le m$ *ile* :

2. Çevrim 2: Her eleman

- ağacı tüm ağaçları ekleyerek günceller,
- $n = \sum_{i=1}^{m} n_i$ yapar

Ayrıca yardımcı düğüm M_s (M_{s_i}),

- Oturum rastsalını günceller,
- $(k_i, bk_i), \forall i \in [n_1, n-1]$ *ve* k_n 'i hesaplar,
- Anahtar ağacını tüm körlenmiş anahtarlarıyla birlikte yayınlar.

3. Her M_i üyesi,

- $i < s$ *ise* , $k_j = br_j^{k_j - 1}, \forall j \in [s, n]$ 'i hesaplar.
- $i > s$ *ise, önce* $k_i = bk_{i-1})^{r_i}$ '*yi, sonra* $k_j = br_j^{k_j - 1}, \forall j \in [i+1, n]$ 'i hesaplar.

M_{s1}, 2. çevrimde M_s ile gösterilen yeni yardımcı düğüm olur. 2 gruplu birleşmeden farklı olarak, ağaçlar yardımcı düğümler arasında tekil iletim ile değiş tokuş edilmez. Bunun yerine yayınlanır; çünkü tekil iletim ile değiş tokuş işleminde m grup arasında $m(m-1)$ mesaj gidip gelir. M gruplu birleşmenin karmaşıklığı Tablo 4.13'te gösterilmiştir.

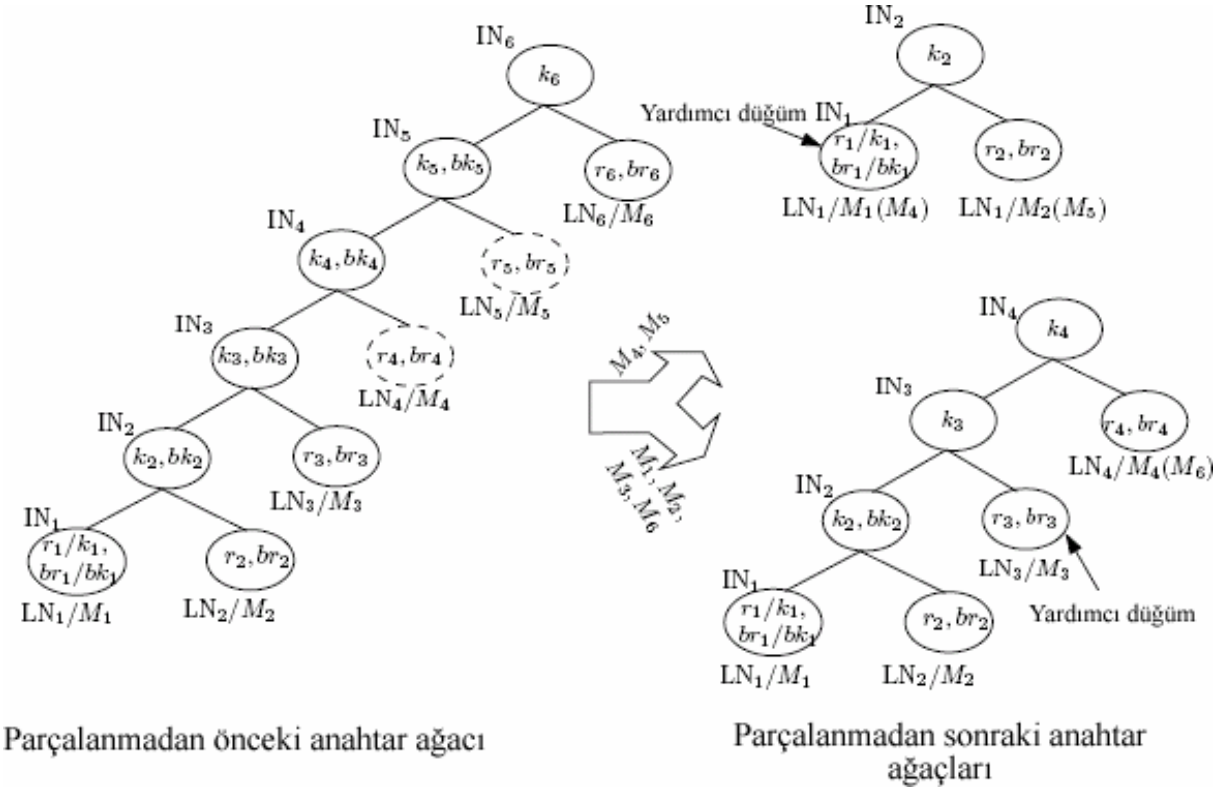
Tablo 4.13: M Gruplu Birleşmenin Karmaşıklık Tablosu

Çevrim	2
Tekil iletim	0
Yayın Boyutu	$M+1$
Üs/M_i	$2k+2$, M_n için $k+1$, M_i için i < n $n+k-i+1$, diğer M_i'ler için
Tekil iletim boyutu	0
Yayın boyutu	$4(n+k)-2m-1$
Tüm üsler	$\frac{k(2n+k+3)}{2}+n+1$
m	Birleşen grup sayısı
k	Diğer gruplardaki eleman sayısı

Birleşme protokolü geriye doğru gizlilik sağlar; çünkü tüm üyeler diğer grupların sadece körlenmiş anahtarlarını bilir; fakat birleşme işlemi anahtar bağımsızlığı sağlamaz, çünkü T_1 ağacının grup anahtarını bilerek birleşme işleminden sonra oluşan ağaçtaki grup anahtarı hesaplanabilir. Bu problem, ikinci çevrimde yardımcı düğümün kendi gizli oturum anahtarını güncellemesi ile çözülür.

4.4.5.5 Parçalanma Protokolü

Ayrılma protokolüne benzer. Yardımcı düğümün seçiminde farklılık vardır. Şekil 4.14'te parçalanma protokolüne ilişkin örnek açıklanmıştır.



Şekil 4.14: STR Parçalanma Protokolüne Örnek

Şekilde de görüldüğü gibi ayrılan üyenin altındaki en düşük sayı etiketli üye yardımcı olarak seçilir. Böyle bir yaprak düğüm yoksa (M_1 grubu terk ediyorsa), en küçük sayılı yaprak düğümü yardımcı olarak seçeriz.

N üyeli bir grup olsun. p tanesi gruptan ayrılacaksa, işlemler listesi aşağıda verilmiştir:

1. Çevrim 1:

Her üye

- Ayrılan üyeleri silerek ağacı günceller,
- $n = n - p$ yapar.

Ayrıca yardımcı düğüm M_s (M_{s_i}),

- Oturum rastsalı r_s'yi ve körleştirilmiş br_s'yi günceller,
- $(k_i, bk_i), \forall i \in [\max(2, s), n-1]$ *ve* k_n'i hesaplar,

- Güncellenmiş anahtar ağacını tüm körlenmiş anahtarlarıyla ve yeni körlenmiş oturum rastsalı ile birlikte yayınlar.

STR parçalanma protokolünün karmaşıklığı Tablo 4.14'te gösterilmiştir.

Tablo 4.14: STR Parçalanma Protokolünün Karmaşıklık Tablosu

	$s=1$	$s \geq 2$
Çevrim	1	1
Tekil iletim	0	0
Yayın	1	1
Üs/M_i	$2m-2$, yardımcı düğüm için $m-i+1$, diğer üyeler için	$2(m-s)+2$, yardımcı için $m-i+1$, M_i için, $i>s$ $m-s+1$, M_i için, $i<s$
Tekil iletim boyutu	0	0
Yayın boyutu	$m-1$	M
Tüm üsler	$\frac{m^2+3m-4}{2}$	$\frac{m^2+3m-s-s^2+2}{2}$
p: Ayrılan üye sayısı	s: güncellenmiş ağaçtaki sponsorun indisi	

4.4.6 TGDH

Tree-based Group Diffie-Hellman (TGDH) [22] protokolü, grup anahtarı yönetimi için dengeli ikili anahtar ağacı kullanır. Bu yöntem, anahtar bağımsızlığını ve anahtar gizliliğini sağlamayı amaçlar. TGDH'de kullanılan notasyonlar:

$<\ell, \mathbf{v}>$: ağaçta, ℓ.seviyedeki v.düğüm.

$\mathbf{k}_{<\ell,v>}$: $<\ell,v>$ düğümünün anahtarı

$\mathbf{bk}_{<\ell,v>}$: $<\ell,v>$ düğümünün körlenmiş anahtarı

$\mathbf{T}_{<i,j>}$: <i,j> düğümünün kök olduğu alt ağaç.

TGDH anahtar ağacında <0,0>'daki düğüm kök düğümdür. Her grup üyesi bir yaprak düğüme karşı düşürülür. Diğer düğümler ara düğümlerdir; 2'şer çocukları vardır. $<\ell,v>$ düğümünün sol çocuğu $<\ell+1,2v>$, sağ çocuğu ise $<\ell+1,2v+1>$'dir.

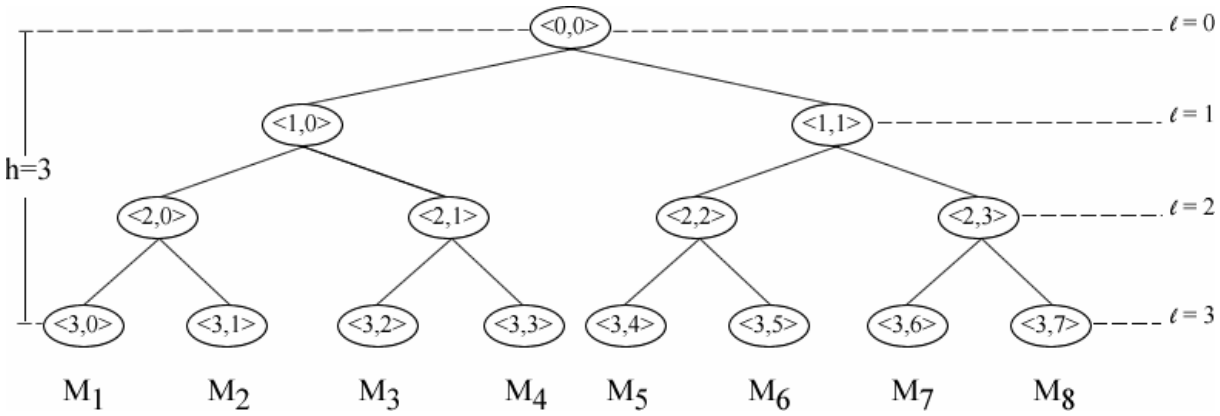
$<\ell,v>$ düğümünün körlenmiş anahtarı $\mathbf{bk}_{<\ell,\mathbf{v}>} = g^{k_{<\ell,v>}}$'dir. $<\ell,v>$ düğümüne ait anahtar rekürsif olarak şu şekilde hesaplanabilir:

$$\mathbf{k}_{<\ell,\mathbf{v}>} = \left(\mathbf{bk}_{<\ell+\mathbf{1},\mathbf{2v}>}\right)^{\mathbf{k}_{<\ell+1,2v+1>}} \mathbf{modp}$$

$$= \left(\mathbf{bk}_{<\ell+\mathbf{1},\mathbf{2v}+\mathbf{1}>}\right)^{\mathbf{k}_{<\ell+1,2v>}} \mathbf{modp}$$

$$= \mathbf{g}^{\left(\mathbf{k}_{<\ell+1,2v>}\mathbf{k}_{<\ell+1,2v+1>}\right)}\mathbf{modp}$$

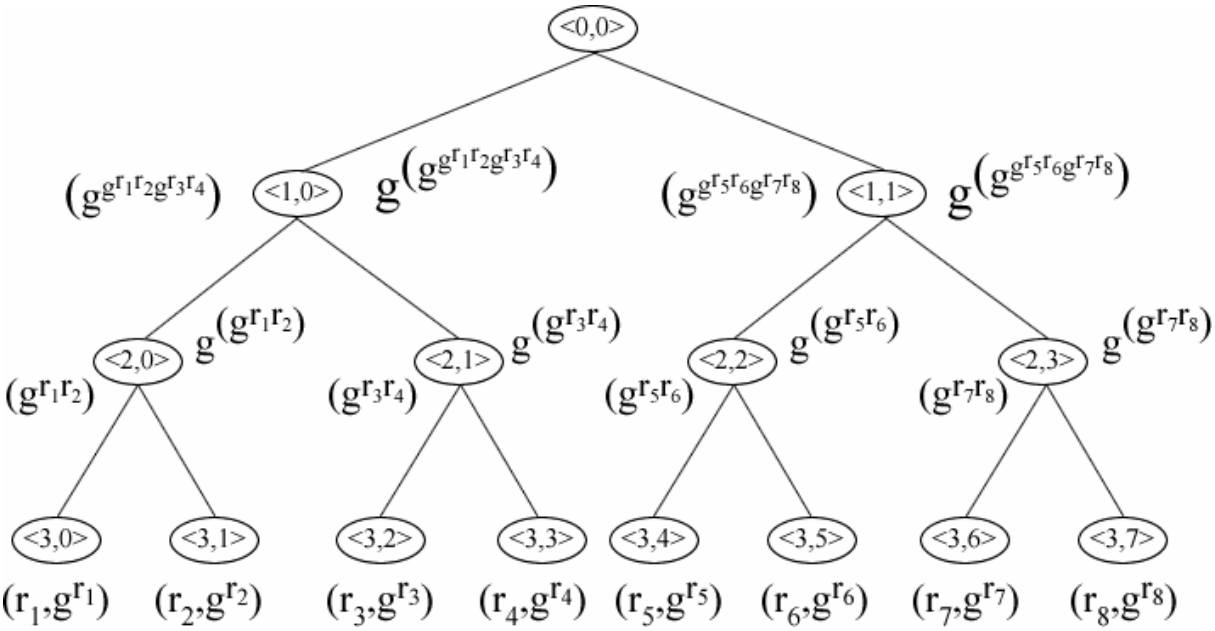
Şekil 4.15'te 8 elemanlı bir grup için TGDH ağacı gösterilmiştir.



Şekil 4.15: 8 Elemanlı Grup İçin TGDH Anahtar Ağacı 1

Her elemanın, anahtar ve körlenmiş anahtarı sırasıyla şöyledir: $M_1 : r_{1,} g^{r_1}$, $M_2 : r_{2,} g^{r_2}$, $M_3 : r_{3,} g^{r_3}$, $M_4 : r_{4,} g^{r4}$, $M_5 : r_{5,} g^{r_5}$, $M_6 : r_{6,} g^{r_6}$, $M_7 : r_{7,} g^{r_7}$, $M_8 : r_{8,} g^{r_8}$.

Şekil 4.16'da her düğümün sol tarafında yazılan değer gizli anahtar, sol tarafında yazılan değer açık anahtar olmak üzere anahtar çiftleri gösterilmiştir.



Şekil 4.16: 8 Elemanlı Grup İçin TGDH Anahtar Ağacı 2

Kök düğümün anahtarları ise: $k_{<0,0>} = g^{(g^{r_5 r_6} g^{r_7 r_8})} g^{(g^{r_1 r_2} g^{r_3 r_4})}$ ve $bk_{<0>} = g^k = g^{[g^{(g^{r_5 r_6} g^{r_7 r_8})(g^{r_1 r_2} g^{r_3 r_4})}]}$.

n üyeli bir TGDH anahtar ağacında, $(n-1)$ tane ara düğüm vardır. Her üye kök düğüm dışındaki tüm düğümlerin körlenmiş anahtarlarını, anahtar yolundaki düğümlerin de anahtarlarını bilmelidir.

4.4.6.1 Kurulum Protokolü

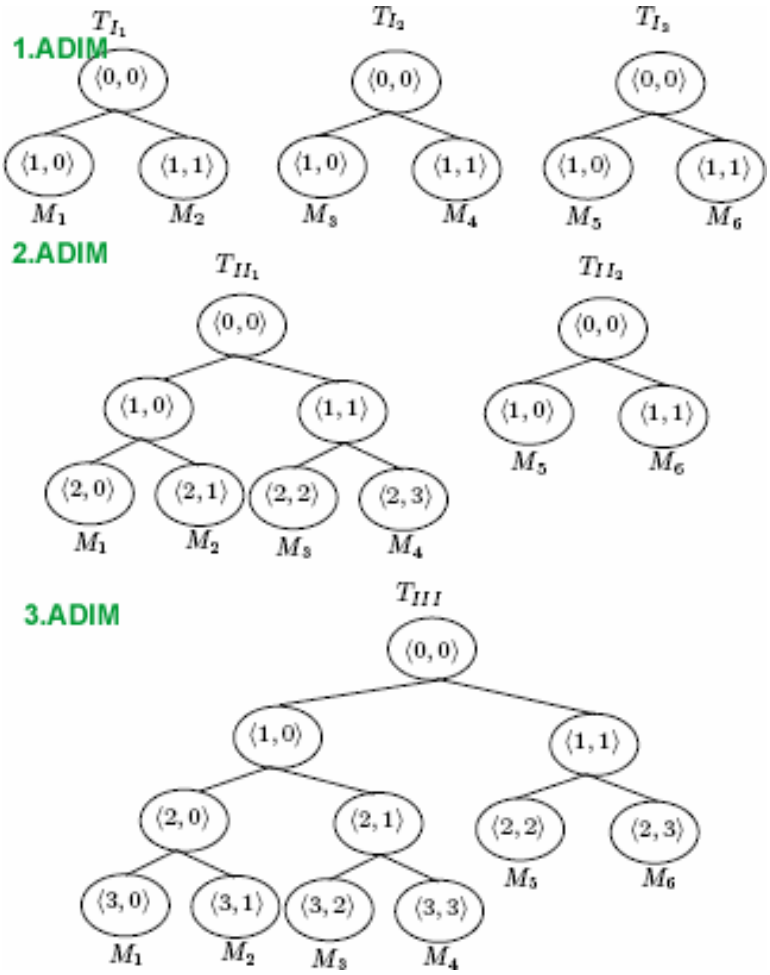
[22] makalesinde herhangi bir kurulum protokolü önerilmemiştir. Bu çalışmada gerçeklenen simülasyon için bir kurulum protokolü önerilmiştir. Grup oluşturmak isteyen n üye olsun. Grup oluşturabilmeleri için anahtar ağacı oluşturmaları gerekir.

1. Tüm üyeler belirli bir kritere göre sıralanır. (M_1, M_2,…., M_n).

2. Daha sonra ilk 2 üye (M_1 ve M_2) , aralarında 2 taraflı DH yürütürler. Aynı işlemi diğer üyeler de yapar.

3. Bir önceki adımın sonunda $\left\lceil \frac{n}{2} \right\rceil$ grup oluşmuş olur. Her yadımcı düğüm (anahtar ağacında en sağ konumdaki yaprak düğüm), kendi anahtar ağacını körlenmiş anahtarlarıyla birlikte yayınlar.

Bu işlemler, ağaçta tüm üyeler yer alana kadar devam eder. Grubun oluşumu $\left\lceil \log_2^n \right\rceil$ çevrim sonunda biter. Şekil 4.17'de kurulum safhasına ilişkin bir örnek gösterilmiştir.



Şekil 4.17: TGDH Kurulum Safhasına Örnek

Tablo 4.15'te kurulum safhasının karmaşıklığı gösterilmiştir.

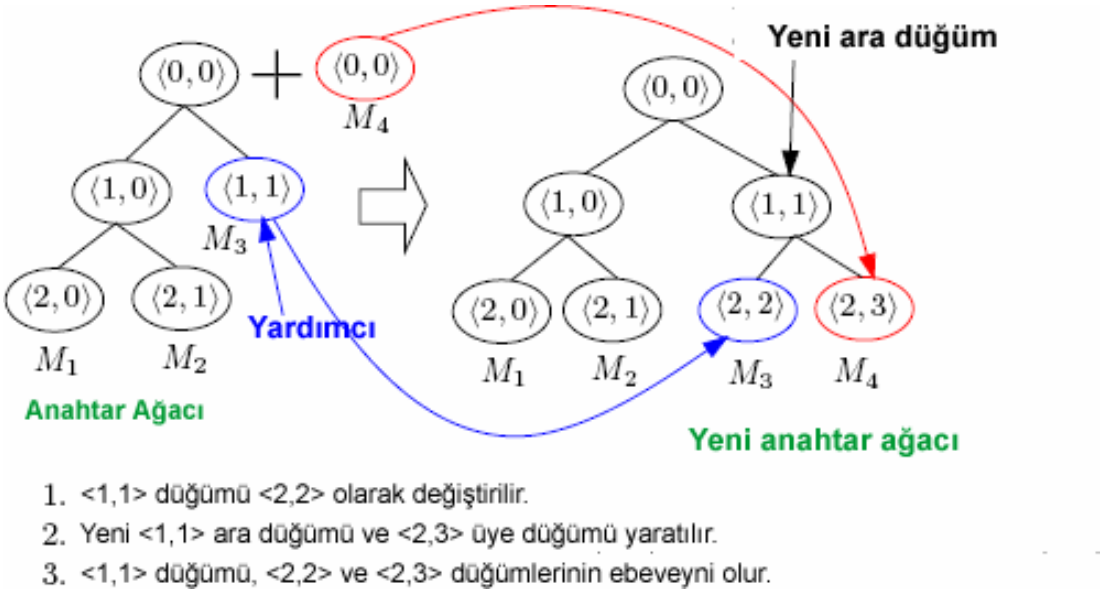
Tablo 4.15: TGDH Kurulum Safhası Karmaşıklık Tablosu

Çevrim	H
Tekil iletim	0
Yayın	$h^2 - h + n$
Üs/$<\ell, v>$	$\ell + i + 1$, $\left\lfloor \frac{v}{2^i} \right\rfloor$ çift ve i bunu sağlayan en küçük pozitif tamsayı. Eğer $<\ell, v>$, en sağdaki yaprak ise(2 farklı ev,yede bu koşulu sağlayan düğüm varsa seviyesi düşük olan) , $\ell + i$ 'dir.
Tekil iletim boyutu	0
Yayın boyutu	$(h-2)2^{h+2} + n + 8$

4.4.6.2 Katılma Protokolü

Yeni bir M_{n+1} elemanı, n elemanlı bir gruba katılmak isterse, bu yeni eleman katılma isteğini belirten mesajı, körlenmiş anahtarı ile birlikte yayınlar. Katılma isteğini alan üye, ekleme noktasını belirler. Ağaç tamamen dengeli ise, $(n = 2^h)$, yeni üye kök düğüme eklenir. Tam dengeli değil ise en sağdaki yaprak düğüm ekleme noktasıdır. Eğer 2 farklı seviyede sağda yaprak düğüm varsa köke daha yakın olan düğüm ekleme noktasıdır. Bu şekilde, anahtar ağacı olabildiğince dengeli tutulmuş olur.

Yeni üyenin eklenmesi işleminde, öncelikle yeni bir ara düğüm yaratılır. Ekleme noktasındaki düğüm(yardımcı) yeni ara düğümün sol çocuğu, yeni eklenen üye de sağ çocuğu olur. Şekil 4.18'de bir üye ekleme örneği gösterilmiştir.



Şekil 4.18: TGDH Üye Ekleme Örneği

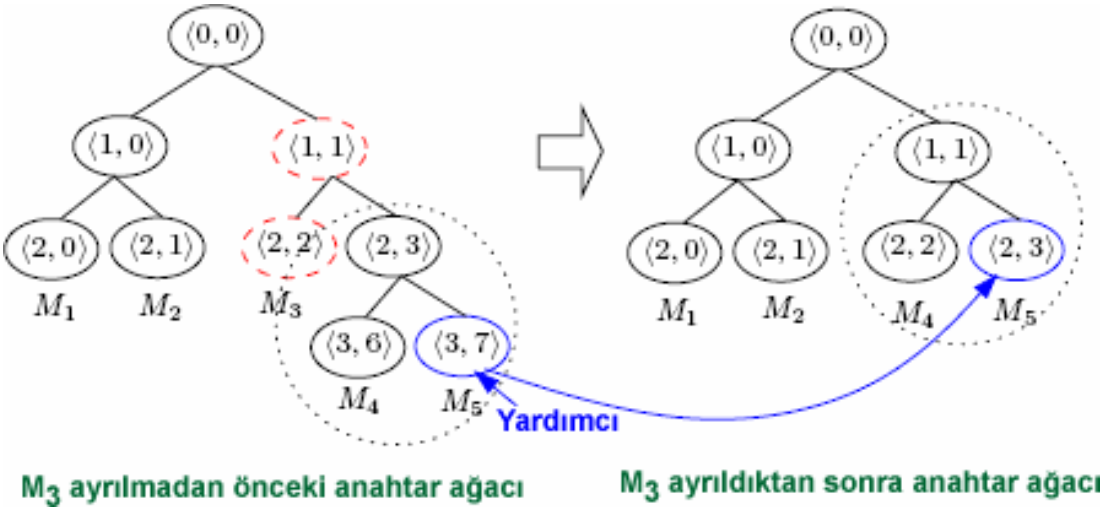
Bu örnekte M_3 yardımcı düğümdür. Eski ağaçtaki her düğüm M_4'ü anahtar ağaçlarına ekleyerek ağaçlarını güncellerler. M_3'ün anahtar yolundaki anahtar ve körlenmiş anahtarları silerler. M_3, kendi anahtarını güncelleyerek $k_{<1,1>}$, $bk_{<1,1>}$ ve $k_{<0,0>}$'ı hesaplar. Yardımcı düğüm, güncellenmiş anahtar ağacını, tüm körlenmiş anahtarlarla birlikte yayınlar. Yayın mesajından sonra, M_1 ve M_2 yeni $k_{<0,0>}$'ı hesaplar. M_4 ise $k_{<1,1>}$ ve $k_{<0,0>}$'ı hesaplar. Katılma protokolünün karmaşıklığı Tablo 4.16'da gösterilmiştir.

Tablo 4.16: TGDH Katılma Protokolü Karmaşıklık Tablosu

Çevrim	2
Tekil iletim	0
Yayın	2
Üs/M_j	$2\ell_s$, yardımcı için $\ell_s - i + 1$, eğer $\exists i \in [0, \ell_s]$ öyle ki $M_j \in T_{<\ell_s - i, v_s/2^i>} / T_{<\ell_s - i + 1, v_s/2^{i-1}>}$
Tekil iletim boyutu	0
Yayın boyutu	$2n+1$
$<\ell_s, v_s>$	Güncellenmiş ağaçtaki yardımcı

4.4.6.3 Ayrılma Protokolü

n üyeli bir grupta M_ℓ üyesi gruptan ayrılmak istesin. M_ℓ'nin kardeş alt ağacının en sağ yaprak düğümü yardımcı düğüm olarak seçilir. Grup iletişimi sisteminden ayrılma olayı olduğu haberi gelince, kalan üyeler, M_ℓ'yi silerek anahtar ağaçlarını günceller. M_ℓ'nin kardeşi, M_ℓ'nin ebeveyn düğümü ile yer değiştirir. Yardımcı düğüm kendi anahtar yolu üzerindeki anahtarları günceller. Şekil 4.19'da gruptan bir üyenin ayrılmasına ilişkin örnek verilmiştir.



Şekil 4.19: TGDH Üye Ayrılma Örneği

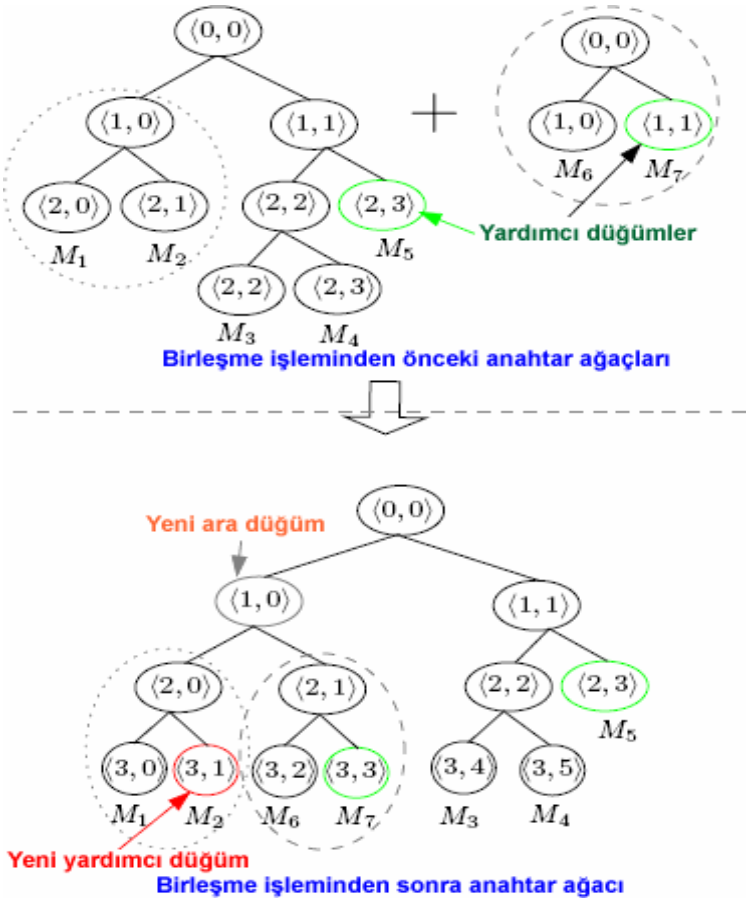
Gruptan ayrılan düğüm M_3'tür. Kalan düğümler (M_1, M_2, M_4, M_5), $<2,2>$ ve $<1,1>$ düğümlerini siler. M_5 kendi anahtarını güncelleyerek $k_{<1,1>}$, $bk_{<1,1>}$ ve $k_{<0,0>}$'ı hesaplar. Güncellenmiş anahtar ağacını tüm körlenmiş anahtarlarla birlikte yayınlar. Bu yayın mesajından sonra, M_1 ve M_2 yeni grup anahtarı $k_{<0,0>}$'ı hesaplar. Ayrılma protokolünün karmaşıklığı Tablo 4.17'de gösterilmiştir.

Tablo 4.17: TGDH Ayrılma Protokolü Karmaşıklık Tablosu

Çevrim	1
Tekil iletim	0
Yayın	1
Üs/M_j	$2\ell_s$, yardımcı için $\ell_s - i + 1$, eğer $\exists i \in [0, \ell_s]$ öyle ki $M_j \in T_{<\ell_s - i, v_s / 2^i>} / T_{<\ell_s - i + 1, v_s / 2^{i-1}>}$
Tekil iletim boyutu	0
Yayın boyutu	$2(n-4)$
$<\ell_s, v_s>$	Güncellenmiş ağaçtaki yardımcı

4.4.6.4 Birleşme Protokolü

G_1 grubuna m adet yeni üye katılmak istesin. TGDH kurulum safhasını yürüterek, m üye G_2 grubunu oluşturur. G_1 ve G_2 grupları birleştirilir. Birleşecek grup sayısı k olmak üzere, $k > 2$ için ikili grup birleşme işlemi $(k-1)$ defa tekrarlanır. Öncelikle birleştirilecek 2 grup derinliklerine göre büyükten küçüğe doğru sıralanır; T^1 ve T^2. Eğer aynı yükseklikte iseler, belirlenecek herhangi başka bir kıstasa göre sıralanırlar. T^2, T^1'e katılır. Birleşme işleminde kullanılacak ekleme noktası belirlenir. Eğer iki ağacın yüksekliği aynı ise, T^2, T^1'in kök düğümüne eklenebilir. Diğer durumda, ağaç yüksekliğinin artmamasını sağlayacak en sağdaki düğüm bulunur. Böyle bir düğüm mevcut değilse, ekleme noktası kök düğümdür. m adet ağaç birleşsin. Yüksekliklerine göre sıralansınlar: $T^1,.....,T^m$. Birleştirme işlemini gerçekleştirmek için her T^i ağacı en sağ yaprak düğümünü yardımcı düğüm olarak belirler. Şekil 4.20'de birleşme protokolüne ilişkin bir örnek gösterilmiştir.

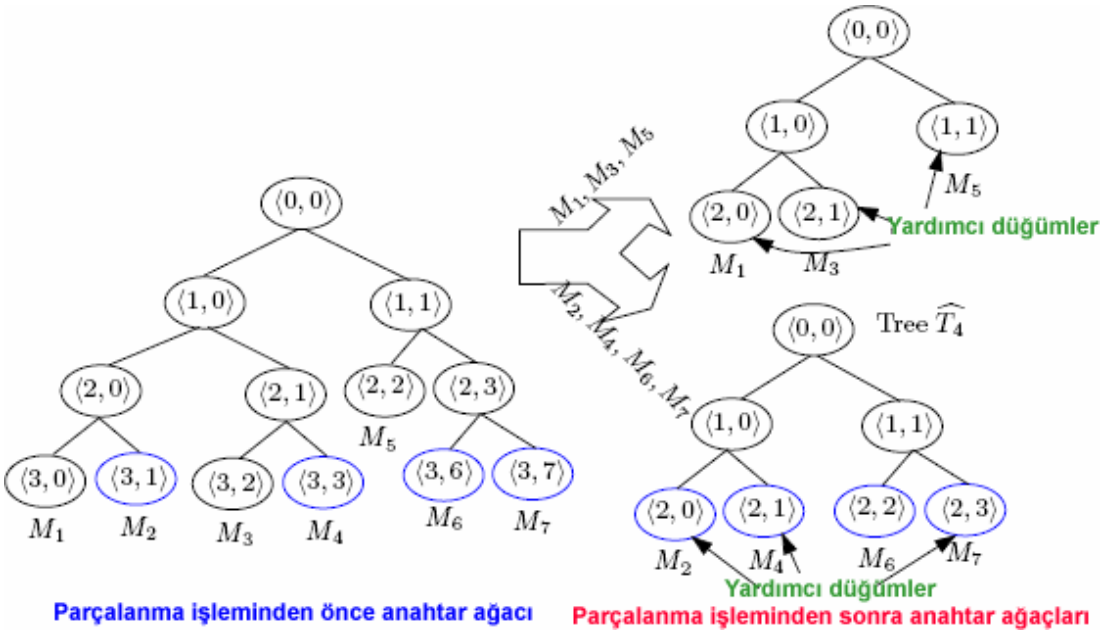


Şekil 4.20: TGDH Birleşme Örneği

M_5 ve M_7 yardımcı düğümleri anahtarlarını günceller. M_5 kendi anahtar ağacındaki yeni $k_{<1,1>}$, $bk_{<1,1>}$, $k_{<0,0>}$ ve $bk_{<0,0>}$'ı hesaplar. M_7 kendi anahtar ağacındaki $k_{<0,0>}$ ve $bk_{<0,0>}$'ı hesaplar. M_5 ve M_7, güncellenmiş ağaçlarını yayınlarlar. Her üye, ağaçları, birbirinden bağımsız olarak birleştirerek M_2'yi yardımcı düğüm olarak seçer. Tüm üyeler M_2'nin anahtar yolundaki tüm anahtar ve körlenmiş anahtarları siler. M_2 kendi anahtarını günceller. Yeni $k_{<2,1>}$, $bk_{<2,1>}$, $k_{<1,0>}$, $bk_{<1,0>}$ ve $k_{<0,0>}$'ı hesaplar. Daha sonra M_2, tüm körlenmiş anahtarlarla birlikte güncellenmiş ağacı yayınlar. Bu mesajla birlikte M_1, $k_{<2,0>}$, $k_{<1,0>}$ ve $k_{<0,0>}$'ı hesaplar. M_6 ve M_7, $k_{<1,0>}$ ve $k_{<0,0>}$'ı hesaplar. Diğer üyeler; M_3, M_4, M_5 sadece yeni grup anahtarı $k_{<0,0>}$'ı hesaplar. ' grubun birleştirilmesi sırasında sadece bir yardımcı düğüm bulunduğu için, M_2, yan yolundaki tüm anahtarları ve körlenmiş anahtarları bilir. Bu nedenle anahtar yolundaki tüm anahtarları ve körlenmiş anahtarları hesaplayabilir.

4.4.6.5 Parçalanma Protokolü

n üyeli bir gruptan k üye ayrılıyor olsun. İlk çevrimde, kalan tüm üyeler ağaçlarını, ayrılan üyeleri ve ebeveynlerini silerek günceller. Eğer bir alt ağacın tüm yaprakları gruptan ayrılıyor ise, bu alt ağacın kök düğümü gruptan ayrılacak düğüm olarak işaretlenir. Bu kök düğümün yaprak düğümleri ayrılan düğümler listesinden silinir. Her ayrılan düğüm için bir sponsor belirlenir.



Şekil 4.21: TGDH Parçalanma Örneği

5. LİTERATÜRDEKİ PROTOKOLLERİN DEĞERLENDİRİLMESİ

2d-küp ve 2d-ahtapot[23], dinamik grup işlemleri için hiçbir protokol önerilmemiştir.

Kurulum: STR[21] en verimli haberleşme sağlayan protokoldür. Yeni bir grup oluşturmak için iki çevrim gerekir. Cliques[25]'te aynı işlem için (n+1) sayıda çevrim gerçekleştirilir. TGDH, Cliques'ten daha verimlidir;çünkü gereken çevrim sayısı üye sayısına logaritmik bağlıdır.

Mesaj sayısı ve uzunluk göz önüne alınırsa; STR yalnızca (n+1) mesaj kullanır. Cliques ve TGDH iki misli sayıda mesaj kullanır. STR' nin toplam mesaj uzunluğu, mesajda yalnızca (2n–1) körlenmiş anahtar bulunduğu için, en küçüktür. Cliques, mesajda (3n–2) körlenmiş anahtar bulundurur. Dengedeki bir TGDH ağacı ele alınırsa, $h = \lceil \log_2 n \rceil \approx \log_2 n + 1$. Gönderilen toplam körlenmiş anahtar sayısı:

$n + (h-2)2^{h+2} + 8 = n + (\log_2 n - 1)2^{\log_2 n + 3} + 8 = 8n \log_2 n - 7n + 8$ olur.

Seri modüler üslere göre, TGDH en verimli hesaplama maliyetine sahiptir; çünkü seri modüler üs sayısı, üye sayısına logaritmik bağlıdır.

Katılma: Tüm protokoller[21–26] 2 çevrim ve 2 mesaj kullanır. STR ve TGDH, 2 çoklu iletim mesajı kullanır, Cliques 1 tane tekli iletim, 1 tane de çoklu iletim mesajı kullanır[25]. Bu nedenle Cliques protokolü haberleşmede daha verimlidir. Cliques hesaplamada maliyeti en yüksek protokoldür; grup büyüklüğüne göre lineer artan sayıda üs gerektirir. TGDH, seri modüler üs sayısının üye sayısına logaritmik bağlı olmasından dolayı, daha verimlidir. STR ise sabit sayıda modüler üs kullanır.

Ayrılma: Tüm protokoller[21–26], 1 çevrim ve 1 çoklu iletim mesaj kullanır. TGDH hesaplama bakımından en verimli protokoldür. STR ve Cliques protokollerinin kullandığı seri modüler üs sayısı grup büyüklüğü ile doğru orantılı iken; TGDH üye sayısı ile logaritmik orantılıdır.

Birleşme: Cliques protokolündeki haberleşme çevrim sayısı, gruba eklenen üye sayısı ile doğru orantılıdır. Bu durumda, STR, sabit sayıda çevrim gerektirdiği için daha verimlidir. TGDH, birçok sayıda çevrim kullanmasına rağmen, haberleşme maliyeti birleşen grup sayısına bağlıdır. Hesaplama maliyetinde ise TGDH, seri modüler üs sayısı en büyük grup büyüklüğü ve tüm üyelerin sayısının çarpımı ile logaritmik bağımlı olduğu için, en verimli protokoldür.

Ayrılma: Cliques ve STR bant genişliği bakımından verimlidir; yalnızca bir mesaj içeren bir çevrim kullanılır. TGDH protokolünde ayrılma için gerekli çevrim sayısı, en az, güncellenen anahtar ağacının boyu ve en çok, $\log_2 p + 1$ arasında[22] olmasına rağmen, maliyeti en fazla işlemdir. Ayrıca TGDH protokolünde mesaj uzunluğunun da dikkate alınması gerekir. TGDH hesap maliyeti bakımından en verimli protokoldür.

Yukarıda sayılan nedenlerden dolayı, STR yüksek ağ beklemelerinin çok olduğu ağlarda en kullanışlı yöntemdir; fakat hesap maliyeti çok yüksektir. Cliques, STR ve TGDH protokollerine göre daha az bellek maliyetine neden olur; fakat kurulum ve birleşmede daha fazla çevrim kullanır[25]. TGDH protokolünün hesap maliyeti tüm protokollerden daha düşüktür; fakat bölünmedeki kullanılan çevrim sayısı da bu protokolün olumsuz yanıdır.

6. TASARSIZ AĞLAR İÇİN ELİPTİK EĞRİYE DAYALI DH GRUP ANAHTAR YÖNETİMİ

Bu bölümde, tasarsız ağlar için önerilen eliptik eğriye dayalı DH grup anahtar yönetimi anlatılmıştır.

Bu çalışmada, TGDH protokolü [22] üzerinde çalışılarak tasarsız ağlar için uygun hale getirilmesi amaçlanmıştır. TGDH, ağaç tabanlı olduğu için ölçeklenebilirlik problemi yoktur; çünkü iletişim ve işlem maliyetleri grup üye sayısı ile doğru orantılı olarak logaritmik artar. Tasarsız ağlardaki düğümlerin düşük kapasiteli düğümler olması nedeniyle işlem ve iletişim maliyetlerinde yapılacak iyileştirmeler TGDH protokolü tasarsız ağlara uygun hale getirecektir.

6.1 Anahtar Ağacının Oluşturulması

TGDH yönteminde, anahtar ağacının nasıl oluşturulacağına dair bilgi verilmemektedir. Simülasyon safhasında anahtar ağacının oluşturulabilmesi için üyelere numaralar verilerek TGDH kurulum evresini gerçekleyebilmeleri sağlanmıştır.

6.2 Grup Anahtarının Hesaplanmasında Eliptik Eğri Diffie Hellman Yönteminin Kullanılması

Tasarsız ağları oluşturan düğümlerin düşük kapasiteli olması nedeniyle işlem ve bellek maliyeti, önerilen protokollerde önemli ölçütlerden biridir. Eliptik eğri Diffie Hellman yönteminin işlem maliyeti, Diffie Hellman[6] yöntemine göre daha düşüktür. Anahtar uzunluğu açısından da ECDH, DH'ye göre daha avantajlıdır. Eliptik Eğri Diffie Hellman yöntemi için kullanılan anahtar boyları aynı güvenlik

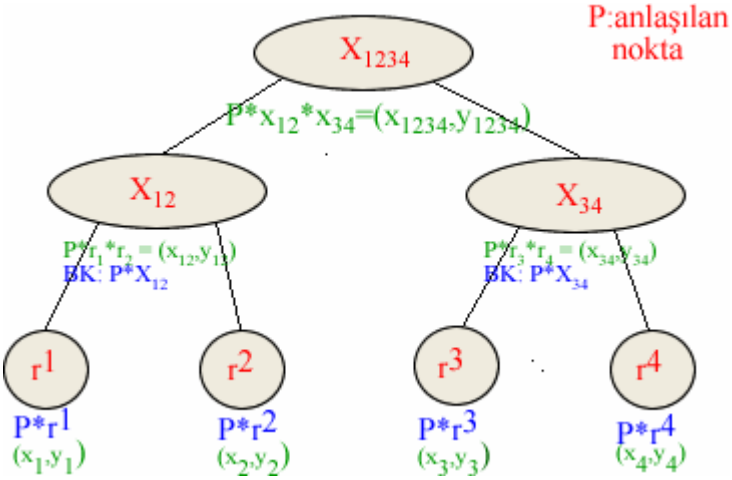
seviyesinde Diffie Hellman yöntemine göre daha düşüktür[27]. Tablo 6.1'de değişik güvenlik seviyeleri için anahtar uzunlukları gösterilmiştir:

Tablo 6.1: EC ve DH Anahtar Uzunluklarının Karşılaştırılması

Diffie Hellman Anahtar Uzunluğu(bit)	Eliptik Eğri Anahtar Uzunluğu(bit)
1024	160
2048	224
3072	256
7680	384
15360	521

Bu çalışmada yapılan testlerde, 112 bit – 512 bit ve 163 bit – 1024 bit, ECDH-DH anahtar çiftleri kullanılmıştır.

Grup anahtarının ECDH yöntemine göre hesaplanmasında, üyeler önceden bir P noktasında anlaşırlar. P noktası ile gizli anahtarların skaler çarpımı o düğüme ait açık anahtarı verir. Şekil 6.1'de açık anahtarlar düğümlerin altında BK etiketi ile gizli anahtarlar ise düğümlerin üzerinde gösterilmiştir. Skaler çarpımın sonucu bir (x,y) ikilisidir ve bu ikiliden x koordinatı düğümün gizli anahtarı olarak kullanılır. Sadece üyeler(yapraklar için), Şekil 6.1'de gösterildiği üzere, x koordinatı yerine kendi oluşturdukları gizli anahtar kullanılabilir. Bir düğümün anahtarı rekürsif olarak şekilde gösterildiği üzere hesaplanabilir. Buradan da açıkça anlaşılabileceği gibi ECDH kullanılan ağaca dayalı grup anahtarı hesaplamada seri üs hesapları yerine skaler çarpımlar kullanılmaktadır.



Şekil 6.1: Grup Anahtarının ECDH Yöntemine Göre Hesaplanışı

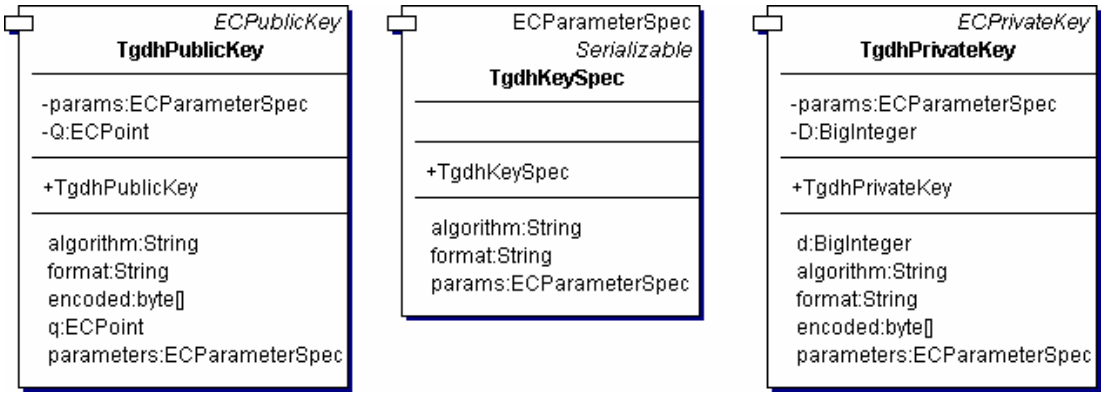
Her üye, anahtar ağacındaki diğer üyelerin körlenmiş anahtarlarını (şekil üzerinde BK ile gösterilmiştir) bildiği sürece grup anahtarını hesaplayabilir.

7. TASARSIZ AĞLAR İÇİN ELİPTİK EĞRİYE DAYALI DH GRUP ANAHTAR YÖNETİMİ SİMÜLASYONU

Bu bölümde TGDH protokolünün, DH ve ECDH yöntemleri ile görsel olarak izlenmesini sağlayan simülasyon programı anlatılmıştır. JAVA programlama dili ile geliştirilen simülasyon bir masaüstü uygulamasıdır. Grup iletişim alt yapısı için JGroups[28] ve kriptografik işlemler için Bouncycastle[29] yazılım kütüphaneleri kullanılmıştır.

7.1 Temel Sınıflar

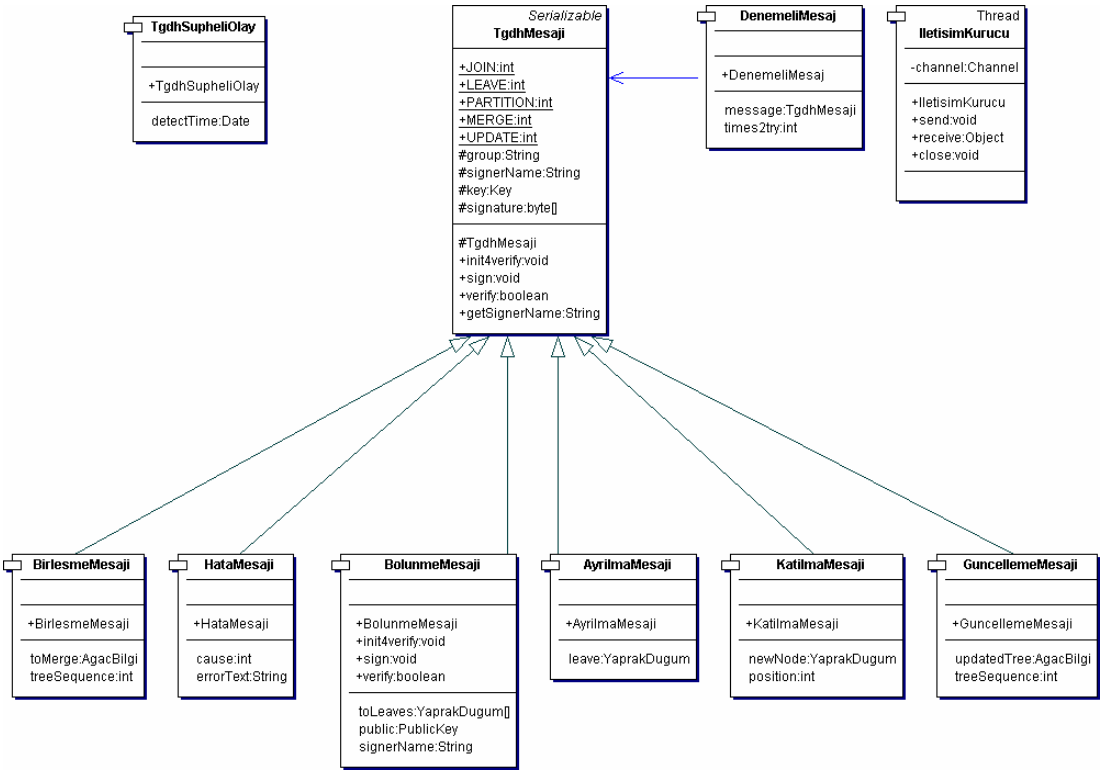
Şekil 7.1'de sınıf diyagramı gösterilen anahtar paketinin içinde anahtar parametrelerinin belirlendiği ve yardımcı metodların bulunduğu sınıflar yer alır. Bu sınıflar DH yönteminin kullanıldığı protokol simülasyonunda DH parametrelerine göre, ECDH yönteminin kullanıldığı protokol simülasyonunda ise ECDH parametrelerine göre düzenlenmiştir. ECDH için kullanılan [29] kütüphanesinde, ECDH anahtar parametreleri doğrudan kullanıcı tarafından verilebildiği gibi tanımlı eğriler tablosundaki eğriler de kullanılabilir. Tanımlı eğrilerin kullanılması durumunda sayısal hiçbir parametre verilmesi gerekmeksizin sadece eğrinin tablodaki adı parametre olarak verilir. Tabloda yer alan eğriler [29]'a ait sanal dokuda verilmiştir. [29] tarafından desteklenmeyen ama önerilen eğrilerden kullanmak için parametrelerin sayısal olarak verilmesi gerekmektedir.



Şekil 7.1: Anahtar Paketi Sınıf Diyagramı

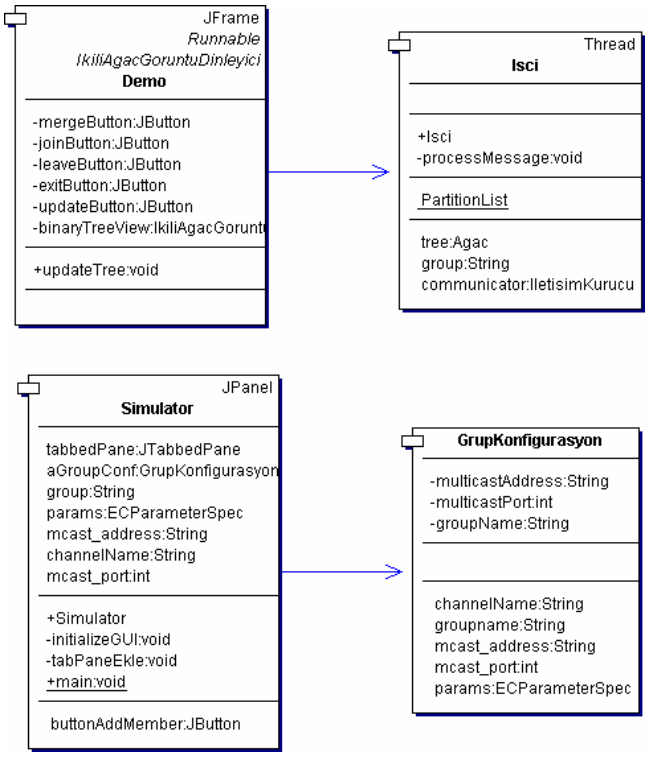
Grup üyelik işlemlerinin simülasyonunu gerçeklemek üzere mesaj tipleri tanımlanmıştır. Bu mesaj tipleri üyelik işlemlerine karşı düşen mesaj tipleridir; katılma, ayrılma, birleşme, parçalanma.

JGroups[28] kütüphanesi kullanılarak mesajların iletimi sağlanmıştır. Bir düğüme mesaj geldiğinde, düğüm, mesajın tipine göre gerekli işlemler yapar. Eğer kendisi yardımcı düğüm ise üyelik işlemini yürütür, değil ise güncelleme mesajını bekler. Böylece protokolde olduğu gibi üyelik işlemi sırasındaki güncel anahtar hesaplama işlemleri sadece yardımcı düğüm tarafından yapılır. İletişim paketine ait sınıf diyagramı Şekil 7.2'de gösterilmiştir.



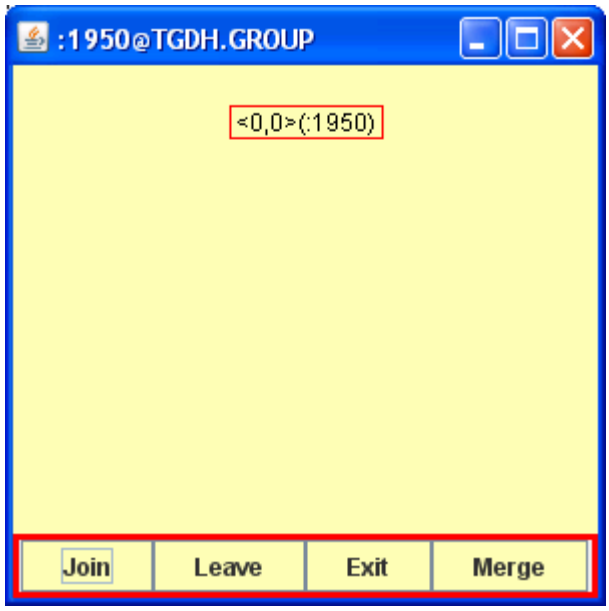
Şekil 7.2: İletişim Paketi Sınıf Diyagramı

Oluşturulacak grup için gerekli çoklu iletim parametrelerinin düzenlenip JGroups altyapısının doğru kullanılabilmesi için konfigürasyon sınıfı oluşturulmuştur. Kullanıcı gerekli parametreleri girerek çoklu iletişim grubu oluşturur. Üyelik işlemlerinin görsel olarak izlenebilmesi için düğümü temsil eden ekran gerçeklenmiştir. Bu ekran üzerindeki işlem butonları kullanılarak grup işlemlerinin yürütülmesi sağlanır. Bu işlem sonucu oluşan yeni anahtar ağacı da düğüm ekranından gözlemlenebilir. Şekil 7.3'te konfigürasyon ve arayüz paketinin sınıf diyagramı gösterilmiştir.



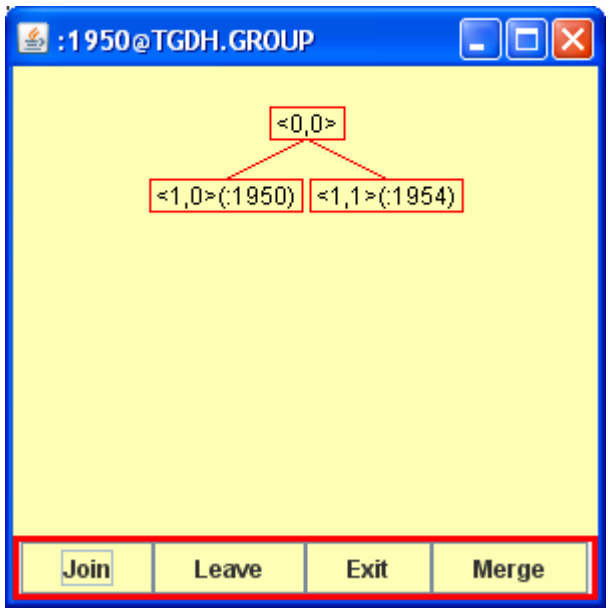
Şekil 7.3: Konfigürasyon ve Arayüz Paketi Sınıf Diyagramı

Demo sınıfı çalıştırılarak düğümü temsil eden Şekil 7.4'teki ekran görüntülenir.

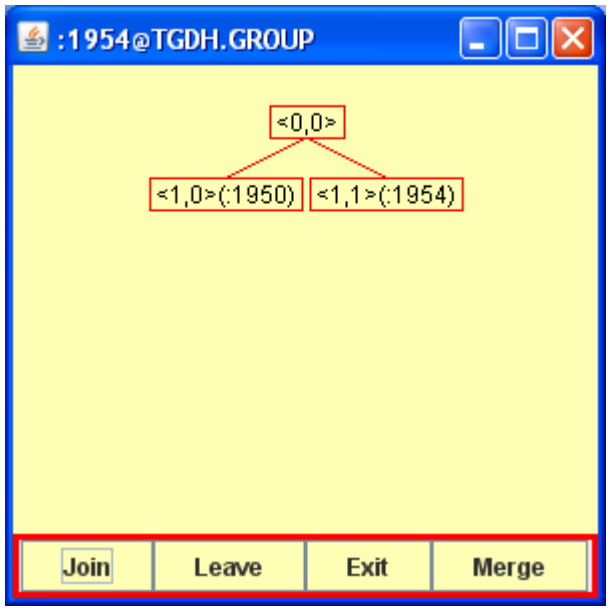


Şekil 7.4: Düğümü Temsil Eden Arayüz

Düğüm ağdaki gruba, imzasının geçerliliği doğrulanırsa, katılabilir. Bir düğüm daha yaratıp bir önceki düğümle beraber bir grup oluşturmak istersek düğümlerde görülen anahtar ağaçları ilk düğüm için Şekil 7.5 ve katılan ikinci düğüm için Şekil 7.6'daki gibi güncellenecektir.



Şekil 7.5: Güncellenmiş Anahtar Ağacı 1

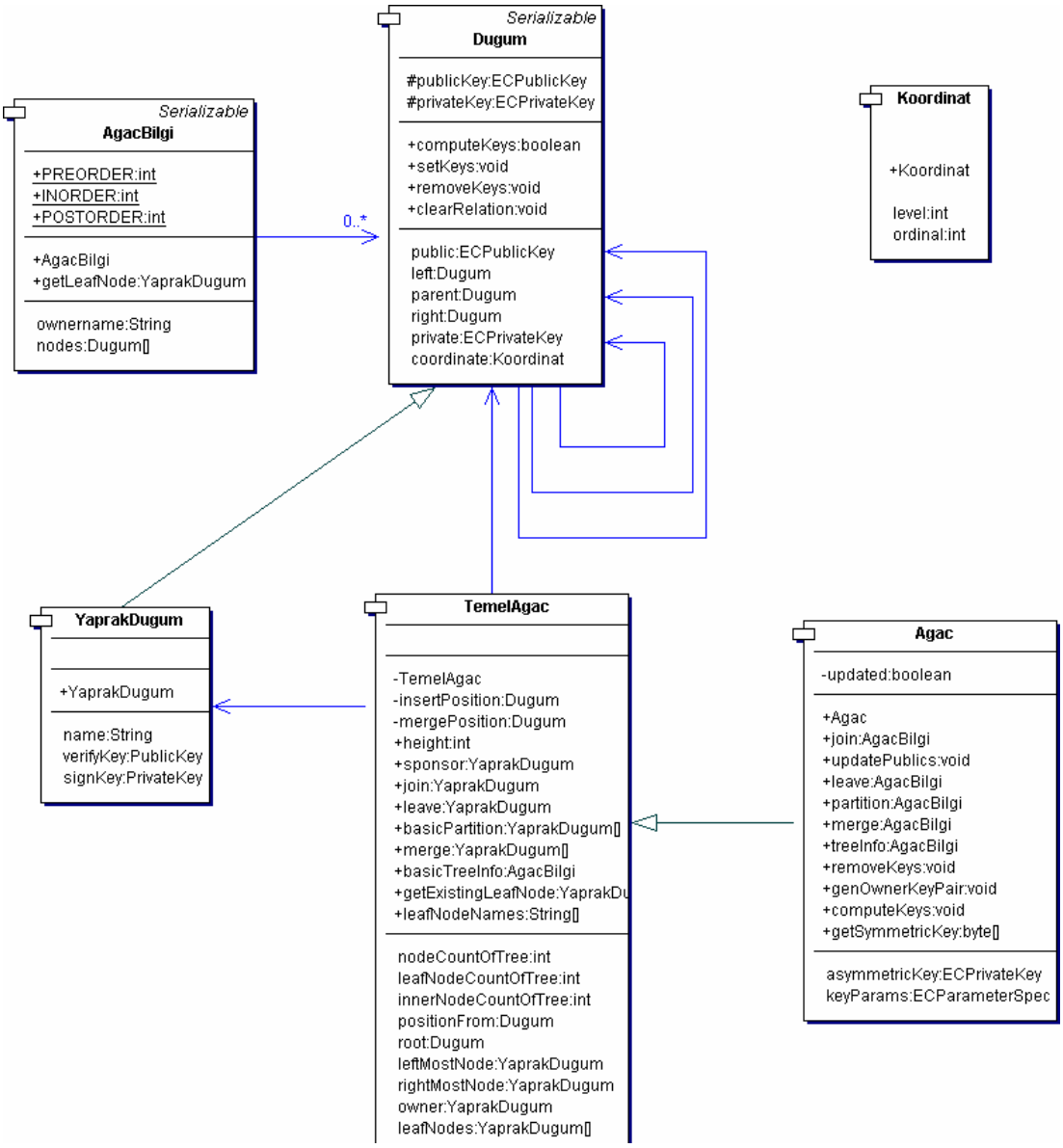


Şekil 7.6: Güncellenmiş Anahtar Ağacı 2

Bir grup oluşturdukları için güncellenmiş anahtar ağaçları aynıdır. Bu şekilde üye eklemeye devam edilebilinir. Bir üyenin gruptan ayrılması durumunda ise ayrılan üye dışındaki üyelerin ağaç görüntüleri aynı iken ayrılan üyenin ağaç görüntüsü farklıdır; çünkü gruptan ayrılan üye grubun yeni ağa. Görünümünden haberdar olamaz, yeni ağaçtan çıkarılmıştır.

Ayrılma, birleşme ve bölünme işlemleri de yürütülebilir. Bu işlemler ekrandan yürütülebileceği gibi test sınıfları kullanılarak ekran açmaksızın konsoldan da yürütülebilir.

DH veya ECDH yöntemi kullanıldığında aynı protokol uygulandığı için görsel olarak değişiklik olmamaktadır; fakat hesaplanan grup anahtarı ve hesaplama zamanında değişiklik olmaktadır. Anahtar ağacının yazılımsal gerçeklenmesi sırasında geliştirilen sınıflar ve birbiri ile ilişkileri Şekil 7.7'de gösterilmiştir.



Şekil 7.7: Anahtar Ağacı Paketi Sınıf Diyagramı

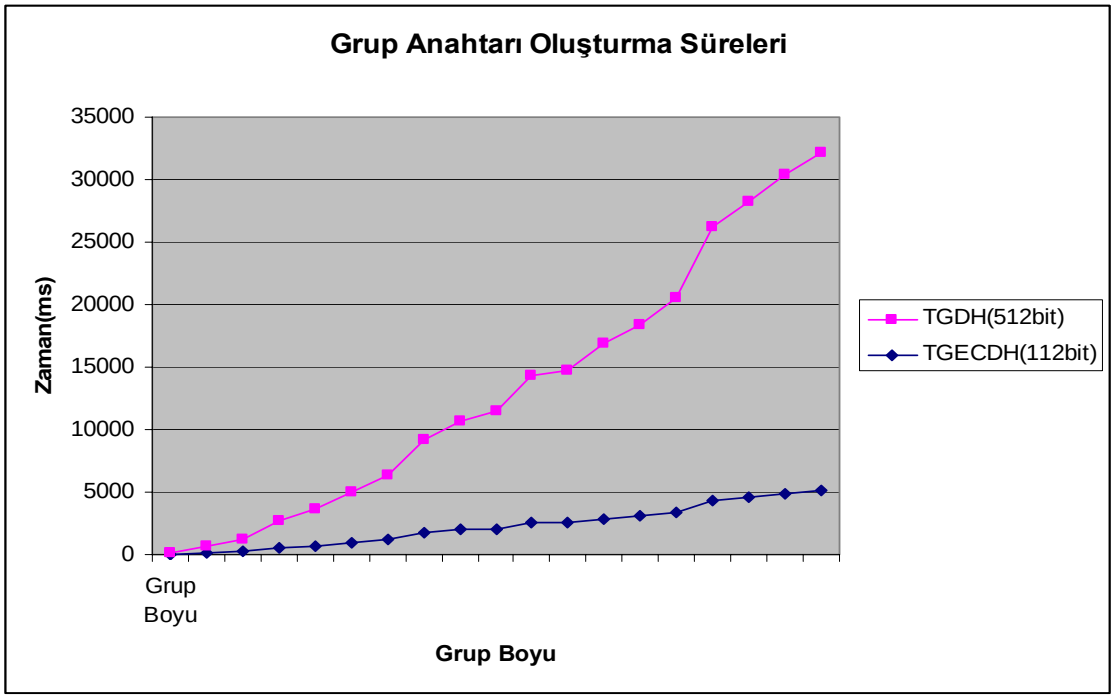
7.2 Performans

Simülasyon sistemi üzerinde 112 bit ECDH – 512 bit DH ve 163 bit ECDH -1024 bit DH anahtar çiftleri ile test yapılmıştır. Test ortamı:

- JVM heap space: min64M – max600M
- Pentium M 1.6 Ghz, 640Mb

- Windows XP Home Edition
- JDK 6.0

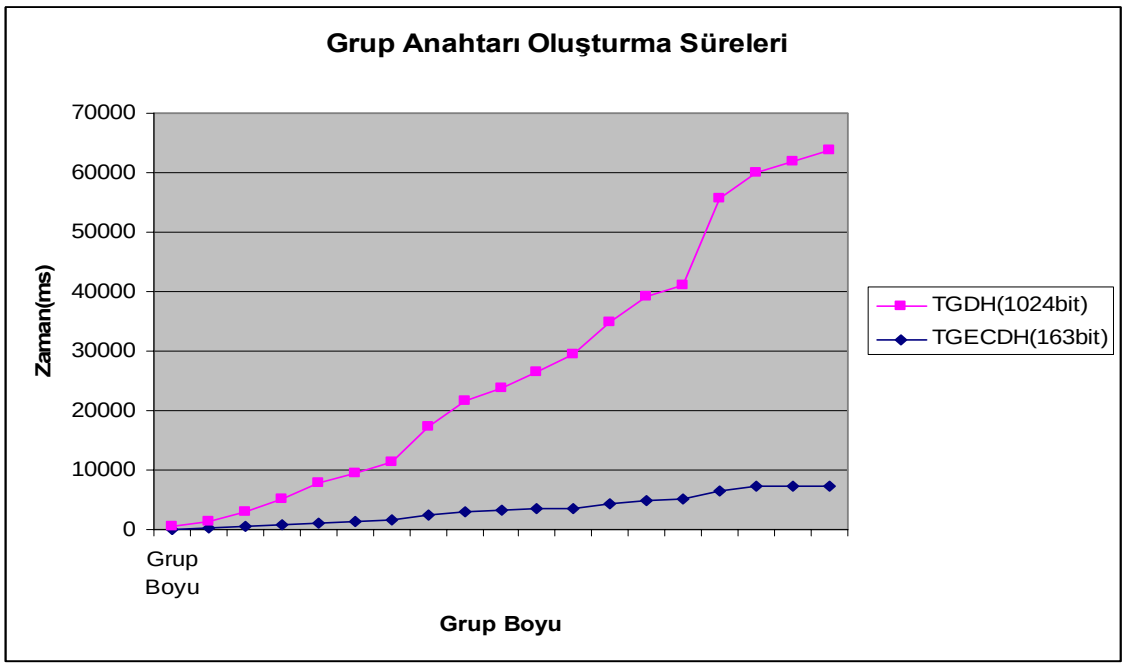
Simülasyon sistemi Java 6.0'dan düşük versiyonlar ile uyumlu değildir; çünkü Java 5.0'te bulunan nesne serializasyonu ile ilgili problem enkod edilmiş anahtarların aktarılmasına engel olmaktadır. Şekil 7.8'de 100 üyeli bir grubun 112 bit ECDH ve 512 bit DH grup anahtarı oluşturma sürelerinin test sonucu çıkarılan karşılaştırmalı grafiği görülmektedir.



Şekil 7.8: İşlem Maliyetini Karşılaştırma Grafiği 1

Şekil 7.8'deki grafiğe göre, ECDH kullanılan ağaca dayalı grup anahtarı oluşturma protokolünün yürütüldüğü grupta anahtar oluşturma işlemi, DH kullanan ağaca dayalı grup anahtarı oluşturma protokolünün yürütüldüğü gruba göre ortalama 4.4 kat daha hızlıdır.

Şekil 7.9'da 100 üyeli bir grubun 163 bit ECDH ve 1024 bit DH grup anahtarı oluşturma sürelerinin test sonucu çıkarılan karşılaştırmalı grafiği görülmektedir.



Şekil 7.9: İşlem Maliyetini Karşılaştırma Grafiği 2

Şekil 7.9'daki grafiğe göre, ECDH kullanılan ağaca dayalı grup anahtarı oluşturma protokolünün yürütüldüğü grupta anahtar oluşturma işlemi, DH kullanan ağaca dayalı grup anahtarı oluşturma protokolünün yürütüldüğü gruba göre ortalama 6.4 kat daha hızlıdır. Daha yüksek güvenlik seviyesinde ECDH yönteminin performansı da artmaktadır.

8. SONUÇLAR

Bu tezde, tasarsız ağlarda grup anahtar yönetimi problemi incelenmiş ve TGDH[22] protokolü ECDH yöntemi ile uygulanarak tasarsız ağlara uygun grup anahtar yönetim protokolü önerilmiştir.

Tasarsız ağ, aralarındaki bağlantıyı sağlayacak bir altyapısı olmayan hareketli düğümlerden oluşan kendi kendine kurulan ağdır. Tasarsız ağlar, kolayca kurulup kullanıldığı için popülerliği artmaktadır.

Grup iletişimi birden çok düğüm ile birden çok düğüm arasında yapılan haberleşmedir; birden çok gönderici ve alıcı vardır. Dosya paylaşımı, mail gibi geleneksel merkezi hizmetler de birden çok sistemde ve ağda çalışan dağıtık sistemler haline getirilmiştir. Bu uygulamalar, grup iletişimi kullanırlar. Bu dağıtık uygulamalara ek olarak, video ve ses konferansları gibi katılımsal uygulamalar, birden çok oyunculu oyunlar da popüler grup iletişim uygulamalarıdır.

Tasarsız ağlar, kullanımı kolay bir platform sunar; fakat bu tip uygulamaları güvenli hale getirmek tasarsız ağların yapısı nedeniyle zordur.

Geleneksel kriptografik anahtar yönetim mekanizmaları tasarsız ağlar için uygun değildir; çünkü tasarsız ağlarda dinamik ağ topolojisi, sınırlı band genişliği, enerji kısıtı vardır, belirli bir altyapı yoktur.

Grup iletişimini güvenli hale getirmek güvenilir bir anahtar yönetim protokolü ile mümkün olmaktadır. Grup anahtar yönetimi protokolünün güvenlik gerekleri grup anahtarının gizliliği, grup anahtarlarının birbirinden bağımsızlığı, ileriye ve geriye dönük anahtar gizliliğidir.

Bu çalışmada tasarsız ağlara uygun bir anahtar yönetim protokolü önerilmiştir. [22]'de önerilen TGDH protokolü DH ve ECDH ile gerçeklenerek test edilmiştir. Bu

protokol [22] üzerinde geliştirildiği için grup anahtarının gizliliği, grup anahtarlarının birbirinden bağımsızlığı, ileriye ve geriye dönük anahtar gizliliği gereklerini yerine getirir. Şekil 7.8 ve Şekil 7.9'da gösterilen grafikteki sonuçlar itibari ile ECDH yöntemi kullanılan TGDH protokolünün anahtar oluşturma sürecinde işlem maliyetinin DH yöntemi kullanılan TGDH protokolüne göre daha düşük olduğu gösterilmiştir. ECDH da kullanılan anahtar boyunun da DH yöntemine göre daha düşük olması nedeniyle düğümlerdeki bellek maliyeti de azalır.

ECDH kullanan TGDH yöntemi, 112 bit ECDH - 512 bit DH anahtar çiftleri için ortalama 4.4, 163 bit ECDH – 1024 bit DH anahtar çiftleri için ortalama 6.4 hızlanma göstermiştir. Daha yüksek güvenlik seviyelerinde DH'ye göre performansın artışı güvenlik gereği yüksek uygulamalarda tercih edilebilir olduğunu gösterir.

Bu sonuçlar göz önüne alındığında, ECDH yöntemi kullanılan TGDH protokolü düşük işlem ve bellek maliyeti ile tasarsız ağlara uygun bir protokoldür.

Bu çalışmanın devamında, geliştirilen sistem üzerinde çok çeşitli testler yapılarak protokol istatistiksel olarak daha detaylı incelenebilir. Bu durumda, hangi konumlardaki üyelik işlemlerinin daha az maliyetli olduğu bulunarak üyelik işlemlerine ait iyileştirmeler de yapılabilir.

KAYNAKLAR

[1]**NIST**, Wireless Ad Hoc Network Project, http://w3.antd.nist.gov/.

[2] **The Official BlueTooth Website,** http://www.bluetooth.com/.

[3] **Rafaeli, S. and Hutchison, D.,** 2003, A Survey of Key Management for Secure Group Communication, *ACM Computing Surveys (CSUR)*, **35(3),** 309 – 329.

[4] **Khalili, A. and Arbaugh, W. A.,** 2002. Security of Wireless Ad-hoc Networks, USA.

[5] **Menezes, A. J., Oorschot, P. C., and Vanstone, S. A,** 1996. Handbook of Applied Cryptography,CRC Press, October 1996, USA.

[6] **Diffie,W. and Hellman, M. E.,** 1976. New Directions in Cryptography, *IEEE Transactions on Information Theory,* **22(6),** 644-654.

[7] **Koblitz, N.,** 1987**,** Mathematics of Computation, American Mathematical Society**,** USA.

[8] **Miller, V.**, 1986. Uses of Elliptic Curves in Cryptography, *Proceeedings on Crypto'85,* Santa Barbara, California, USA, November 1986, 417 – 426.

[9] **Du, W., Deng, J., Han, Y. S. and Varshney, P. K.,** 2005. A Pairwise Key Predistribution Scheme for Wireless Sensor Networks, *ACM Transactions on Information and System Security (TISSEC)*, **8(2),** 228 - 258.

[10] **Bauer, R. K., Berson, T. A. and Feiertag, R. J.,** 1983. A Key Distribution Protocol Using Event Markers, *ACM Transactions on Computer Systems(TOCS),* **1(3)**, 249 - 255.

[11] **Needham, R. M. and Schroeder, M. D.,** 1978. Using Encryption for Authentication in Large Networks of Computers, *Communications of the ACM*, **21(12),** 993 - 999.

[12] **Needham, R. M. and Schroeder, M. D.,** 1987. Authentication Revisited, *ACM SIGOPS Operating Systems Review*, **21(1)**, 7 – 7.

[13] **RFC1510,** 1993**.** The Kerberos Network Authentication Service (V5), *Networking Working Group,* ABD.

[14] **RFC2712,** 1999. Addition of Kerberos Cipher Suites to Transport Layer Security(TLS), *Network Working Group,* ABD.

[15] **ISO 8732**, 1988. Banking - Key Management , *ISO/TC 68/SC 2 ,* ABD.

[16] **Molva, R., Tsudik, G., Herreweghen, E. V. and Zatti, S.,** 1995. KryptoKnight Authentication and Key Distribution System, *IEEE/ACM Transactions on Networking (TON)* , **3(1)**, 31 – 41.

[17] **ISO/IEC 9594-8,** 1994. Authentication framework, ABD.

[18] **Amir, Y.,** 1995. Replication Using Group Communication over a Partitioned Network, *PhD Thesis*, The Hebrew University of Jerusalem Institute of Computer Science, Israel.

[19] **Moser, L., Amir, Y., Melliar, P. and Agarwal, D.,** 1994**.** Extended Virtual Synchrony, *ICDCS'94*, Pozman, Poland, January 1994, 56 – 65.

[20] **Fekete, A., Lynch, N. and Shvartsman, A.,** 2001. Specifying and Using a Partitionable Group Communication Service , *ACM Transactions on Computer Systems (TOCS)*, **19(2)**, 171 - 216.

[21] **Steer, D. G., Strawczynski, L., Diffie, W. and Wiener, M.,**1988. A Secure Audio Teleconference System, *Proceedings on Advances in Cryptology*, Santa Barbara, California, USA, November 1998, 520 – 528.

[22] **Kim, Y., Perrig, A. and Tsudik, G.,** 2004. Tree-based Group Key Agreement, *ACM Transactions on Information and System Security (TISSEC),***7(1),** 60-96.

[23] **K. Becker, K. and Wille, U.,** 1998. Communication Complexity of Group Key Distribution, *Proceedings of the 5th ACM Conference on Computer and Communications Security*, San Francisko, California, USA, November 1998, 1 - 6.

[24] **Asokan, N. and Ginzboorg, P.,** 2001. Key Agreement in Ad Hoc Networks, C*omputer Communications*, **23(17)**, 1627-1637.

[25] **Steiner, M., Tsudik, G. and Waidner, M.,** 2000. Key Agreement in Dynamic Peer Groups, *IEEE Transactions Parallel Distributed Systems*, **11(8),** 769 – 780.

[26] **Kim, K. Y., Perrig, A. and Tsudik, G.,** 2001. Communication-Efficient Group Key Agreement*, Proceedings of the 16th international conference on Information security: Trusted information: the new decade challenge*, Paris, France, November 2001, 229 – 244.

[27] **Barker, E., Barker, W., Burr, W., Polk, W., and Smid**, **M.,** 2006. Recommendation for Key Management - Part 1, *NIST,* August 2005.

[28] **Bela, B.,** 1999, JGroups, Cornell University Department of Science, http://www.jgroups.org/ docs/user/index.html

[29] **Bouncycastle,** http://www.bouncycastle.org/

ÖZGEÇMİŞ

Seha SOYLU, 24.06.1981 tarihinde Muğla'da doğmuştur. 1999 yılında Muğla Anadolu Lisesi'nden, 2003 yılında İTÜ Bilgisayar Mühendisliği Bölümü'nden mezun olmuştur. 2003–2004 yılları arasında ELİAR Elektronik AŞ'de Yazılım Uzmanı olarak çalışmıştır. 2004 Eylül ayından beri TÜBİTAK Ulusal Elektronik ve Kriptoloji Araştırma Enstitüsü'nde Araştırmacı olarak çalışmaktadır.