KABLOSUZ ALGILAYICI AĞLARDA ANAHTAR DAĞITIM YÖNTEMLERİ İÇİN PERFORMANS DEĞERLENDİRMESİ

ÖNDER AMEEN KHALİL

YÜKSEK LİSANS TEZİ
BİLGİSAYAR BİLİMLERİ

GAZİ ÜNİVERSİTESİ
BİLİŞİM ENSTİTÜSÜ

HAZİRAN 2011
ANKARA

Önder Khalil tarafından hazırlanan KABLOSUZ ALGILAYICI AĞLARDA ANAHTAR DAĞITIM YÖNTEMLERİ İÇİN PERFORMANS DEĞERLENDİRMESİ adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

Yrd. Doç. Dr. Suat ÖZDEMİR
Danışman

Bu çalışma jürimiz tarafından oy birliği ile Bilgisayar BilimleriAnabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir.

Başkan :

Üye :

Üye :

Üye :

Üye :

Tarih :

Bu tez, Gazi Bilişim Enstitüsü tez yazım kurallarına uygundur.

TEZ BİLDİRİMİ

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

Önder Khalil

KABLOSUZ ALGILAYICI AĞLARDA ANAHTAR DAĞITIM YÖNTEMLERİ İÇİN PERFORMANS DEĞERLENDİRMESİ

(Yüksek Lisans Tezi)

ÖNDER AMEEN KHALİL

GAZİ ÜNİVERSİTESİ
BİLİŞİM ENSTİTÜSÜ
Haziran 2011

ÖZET

Kablosuz Algılayıcı Ağları (KAA) her bir cihaza algılayıcı düğümü adı verilen ve batarya enerjisi ile çalışan çok sayıda küçük özerk cihazlardan oluşur. Bu cihazlar bütünleşik algılayıcılar, veri işleme özellikleri ve kısa erişimli radyo haberleşme sistemleri ile donatılmıştır. Güvenilirlik, aslına uygunluk, erişebilirlik ve bütünlük KAA' nın tipik güvenlik amaçlarındandır. KAA'lar, askeri amaçlara yönelik algılama ve izleme sistemleri, çevre görüntüleme vb çok çeşitli uygulamalarda kullanılmaktadır. Güvenliğin son derece önemli olduğu düşman ortamlarda, KAA'ların güvenli haberleşmeyi sağlayabilmesi kriptografik yöntemleri kullanılmasına bağlıdır. Kriptografik yöntemler kullanılırken her zaman bir anahtar dağıtım hizmeti gereklidir. Bu çalışmada literatürde KAA'lar için önerilmiş olan anahtar dağıtım yöntemleri incelenmiştir. KAA'ların özelliklerine göre anahtar dağıtım yöntemleri için bir sınıflandırma yapılmış ve incelenen yöntemlerin güvenilir iletişim, hafıza yükü, iletişim yükü, anahtar dağıtım süresi ve düğüm yakalanmasına karşı direnç açısından performans karşılaştırması yapılmıştır.

Bilim Kodu : 902.1.063

Anahtar Kelimeler: kablosuz algılayıcı ağları, anahtar dağıtım yöntemleri

Sayfa Adedi : 156

Tez Yöneticisi : Yrd. Doç. Dr. Suat Özdemir.

PERFORMANCE EVALUATION OF KEY DISTRIBUTION METHODS FOR WIRELESS SENSOR NETWORKS

(MS c Thesis)

ONDER AMEEN KHALIL

GAZI UNIVERSITY

INFORMATICS INSTITUTE

June 2011

ABSTRACT

Wireless sensor networks usually consist of a large number of small devices that are called sensor node. Each device is battery powered and equipped with integrated sensors, a data processing unit, and a short-range radio communication unit. Sensor nodes are significantly constrained in terms of energy, memory, and computational capacity. Wireless sensor networks are being deployed in wide variety of applications, including military sensing and tracking, environment monitoring, patient monitoring, smart environments. When a wireless sensor network is deployed in a hostile environment, security becomes an extremely important issue. Confidentiality, authenticity, availability, and integrity are typical security goals for wireless sensor networks. Providing these goals to secure communication among sensor nodes typically depends on the employment of cryptographic schemes which always require a key management service. In this thesis, key distribution methods of wireless sensor networks are investigated and a classification for these key distribution methods is proposed. In addition, performance evaluation of the analyzed key distribution schemes in terms of secure connectivity, memory overhead, communication overhead, time and resilience against node capture is presented.

Science Code : 902.1.063

Key Words : wireless sensor networks, key management methods

Page Number : 156

Adviser : Assist. Prof. Dr. Suat Özdemir

TEŞEKKÜR

Çalışmalarım boyunca, değerli yardım ve katkılarıyla zaman mefhumu gözetmeksizin beni yönlendiren ve sonuca ulaşmadaki azim ve kararlılığımı artıran yaklaşımları ile beni destekleyen Sayın Danışmanım Yrd. Doç. Dr. Suat Özdemir'e, şükranlarımı sunarım.

Benden öğretim hayatım boyunca her türlü maddi manevi desteğini esirgemeyen, eğitimin değerini bana aşılayan çok değerli anne ve babam, kardeşlerim Esin ve İnan'a ayrıca enişteme Dr. Ali Habib'e teşekkürlerimi sunmayı bir borç bilirim.

İÇİNDEKİLER

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış bazı simgeler kısaltmalar ve tanımlar, açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	**Açıklama**
Bit	Bilişimde kullanılan en küçük bilgi birimi (0 veya 1).
CMOS	Algılayıcı piksellerden oluşan, entegre devre içeren bir görüntü algılayıcısıdır
ID	Identification
KB	Kilobyte
KHz	Kilohertz (10^3 Hertz)
Mbps	Megabit (10^6 Bit) per second
MB	Mega Bayt
MHz	Megahertz (10^6 Hertz)
m	Metre

Kısaltmalar	**Açıklama**
ACK	Acknowledgement
DoS	Denial of Service
GPS	Global Positioning System
CMOS	Complementary Metal Oxide Semiconductor
MANET	Mobile Ad-hoc NETwork
RSA	Rivest Shamir Adleman algorithm
RAM	Random Access Memory
EEPROM	Electrically Erasable Programmable Read-Only Memory

Kısaltmalar	Açıklamalar
OS	Operating System
RF	Radio frequency
MAC	Media Access Control
MANET	Mobile Ad Hoc NETwork
KDC	Key Distribution Center
ECC	Ellitic Curve Cryptograph
ROM	Read-only memory
EBS	Exclusion Based System
LEAP	Localized Encryption and Authentication Protocol
QoS	Quality of Service
GUI	Graphical User Interface
NED	Network Description Language
CRT	Chain Remainder Theorem
pdf	probability density function
µTESLA	micro version of Time Efficient Stream Loss-tolerant Authentication

1. GİRİŞ

Kablosuz Algılayıcı Ağlar (KAA), çok sayıda ucuz ve sınırlı kaynaklı düğümlerden oluşan toz tanesi bilindiği gibi bir ad hoc ağın özel türü sayılabilir.

Kablosuz algılayıcı ağlar için ilk kablosuz algılayıcı düğümü 1996'da üretilmiştir. Bir madeni para büyüklüğünde olan bu algılayıcı düğümünün ömrü kısaydı. Algılayıcılar ve KAA'lar hayatımızın hemen hemen her noktasında yer almakta ve çeşitli alanlarda kullanılmaktadır. Düşman ordusundaki hareketliği ve nükleer saldırıları sezmek gibi askeri uygulamalarda, volkanik patlamalar gibi doğa olaylarını takip etmekte, uzaktan hasta gözetimi ve doktorların uzaktan ameliyat takibi gibi tıbbi uygulamalarda kullanılmakta ve büyük önem arz etmektedir [1]. KAA'larda veri iletişimi kablosuz ortamda gerçekleştiğinde KAA uygulamalarının birçoğu düşman ve kötücül kişilerin gizli bilgiye kulak misafiri olmasını ve sistemin çalışmasını engellemelerini önlemek için güvenlik önlemlerini sağlamak zorundadırlar.

Bu nedenle KAA'ların gerçek potansiyelini elde etmek için gerekli olan kritik etmen güvenliktir. KAA'ların özel altyapıya ihtiyaç duyması, sınırlı hesaplama, haberleşme kabiliyeti, algılayıcı düğümlerin enerji tüketimi, algılayıcılarda fiziksel güvenliğin bulunmaması ve büyük ölçekli yerleştirme gibi faktörler nedeniyle normal ağlar için tasarlanan klasik güvenlik teknikleri KAA'lar için uygun değildir [2]. Diğer düğümlerle iyi çalışan, güvenilir bir algılayıcı düğümü tasarlayıp üretmek için, dikkate alınması gereken en önemli kriterlerden birisi bu düğümlerin aralarındaki iletişim güvenliğini sağlamaktır.

Geleneksel ağlarda kullanılan güvenlik teknikleri iki sebeple doğrudan KAA'larda kullanılamaz. Birincisi, KAA'lar güç, hafıza, hesaplama ve iletişim kabiliyeti açısından sınırlıkaynaklara sahiptir, ikincisi ise KAA'lar düşman alanında yerleştirilir ve düşman tarafından bu algılayıcı düğümler fiziksel ataklara kalabilirler.

Bu nedenle KAA'lar için özel güvenlik uygulamalarına ve dolayısı ile bu güvenlik uygulamaları için de yeni anahtar dağıtım yöntemlerine ihtiyaç vardır.

Bu çalışmada KAA larda iletişim güveniğini sağlamasının ilk ve en önemli aşaması olan anahtar dağıtım yöntemleri incelenmiştir. Performans değerlendirmesinde güvenilir iletişim, hafıza yükü, iletişim yükü, anahtar dağıtım süresi ve düğüm yakalanmasına karşı çeşitli metrikler kullanarak tek anahtar, ikili anahtar, temel olasılıksal yöntem ve q- bileşim rastgele anahtar öndağıtım yöntemleri incelenmiştir.

Bu metrikleri ölçmek ve değerlendirmek için OMNeT++ "Objective Modular Network Test-bed in C++"kullanılmıştır. OMNeT++ açık kaynak kodlu ve ayrık olay tabanlı bir benzetim aracıdır. OMNeT++ ayrıca güçlü bir grafik kullanıcı arayüzüne sahiptir.

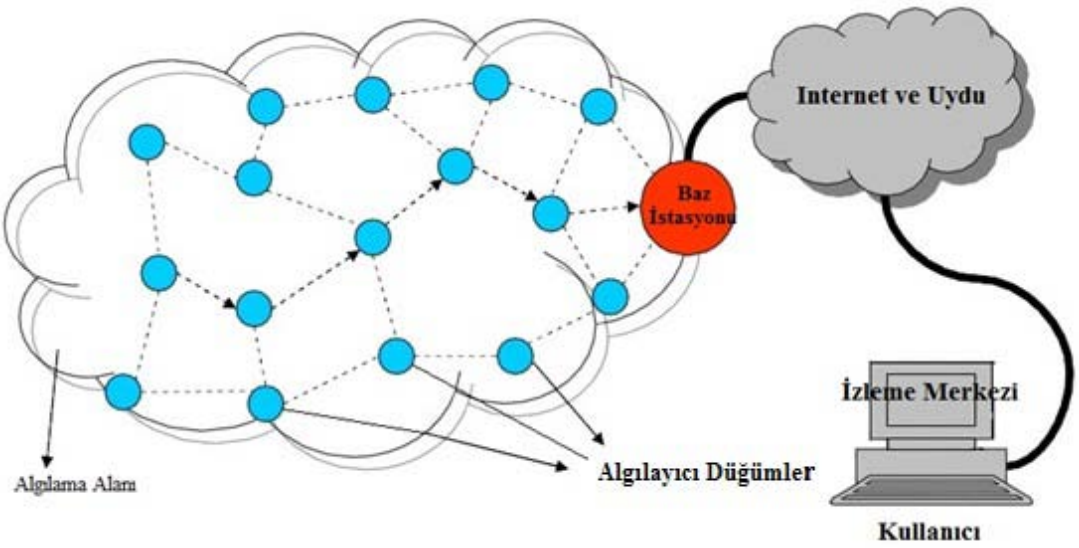
Tezin geri kalan kısmı şu şekilde organize edilmiştir. 2. Bölümde KAAlar hakında detaylı bilgi verilmiştir. KAA tasarımını etkileyen faktörler ve KAA güvenliği literatürdeki çalışmalar dayanarak verilmiş, bu bağlamda anahtar dağıtım yöntemlerine olan ihtiyaç bellirlenmiştir. 3. Bölümde, KAA literatüründe sıklıkla kullanılan anahtar dağıtım yöntemleri detaylı olarak anlatılmıştır. 4. Bölümde ilk olarak performans değerlendirmesinin yapıldığı anahtar dağıtım yöntemleri için üretilen senaryolar açıklanmıştır. Bu bölüm sonunda elde edilen sonuçlar göresel olarak verilmiş ve bu sonuçlar yorumlanmıştır . 5. Bölüm tez çalışmasından elde edilen sonuçları özetleyerek gelecek çalışmalar için öneriler sunmaktadır.

2. KABLOSUZ ALGILAYICI AĞLAR

Kablosuz Algılayıcı Ağlar (KAA) algılayıcı düğümü adı verilen ve batarya enerjisi ile çalışan çok sayıda küçük özerk cihazlardan oluşur [3]. Bu cihazlar bütünleşik algılayıcılar, veri işleme özellikleri ve kısa menzilli radyo haberleşme sistemleri ile donatılmışlardır. Algılayıcılar enerji, depolama ve hesaplama kapasitesi gibi kullanılabilir mevcut kaynaklar açısından belirgin bir biçimde kısıtlanmıştır.

Bir KAA tipik olarak çok az bir altyapıya sahiptir veya hiç bir altyapısı yoktur. Mantıksal olarak kendiliğinden organize edilen KAA' lar yoğun bir şekilde olayın içerisinde olan veya ona çok yakın bir şekilde konuşlandırılan ve ortam hakkında veri elde etmek için bir bölgeyi takip etmek amacıyla çalışan sayıları binlerle ölçülebilen çok sayıda algılayıcı ağından oluşmaktadır [3].

Şekil 2.1 KAA'ların yapısı için tipik bir örnek sunmaktadır. Algılama alanına birçok algılayıcı düğümü yerleştirilir; her düğüm kablosuz alıcı-verici, mikrodenetleyici ve güç kaynağı ile donatılmıştır. Bu düğümler vasıtasıyla ağ, fiziksel ortamı örnekler ve düğümden-düğüme baz istasyonuna iletir (her düğüm çok sekmeli yönlendirme algoritması tarafından desteklenir). Baz istasyonu düğümlerden verileri toplar ve internet gibi ana taşıyıcı altyapı vasıtasıyla izleme merkezine iletir. Uygulama alanlarına bağlı olarak KAA'lar çeşitli gruplara ayrılabilir [4].



Şekil 2.1. KAA'lara bir örnek [4].

2.1. KAA'ların Kategorileri

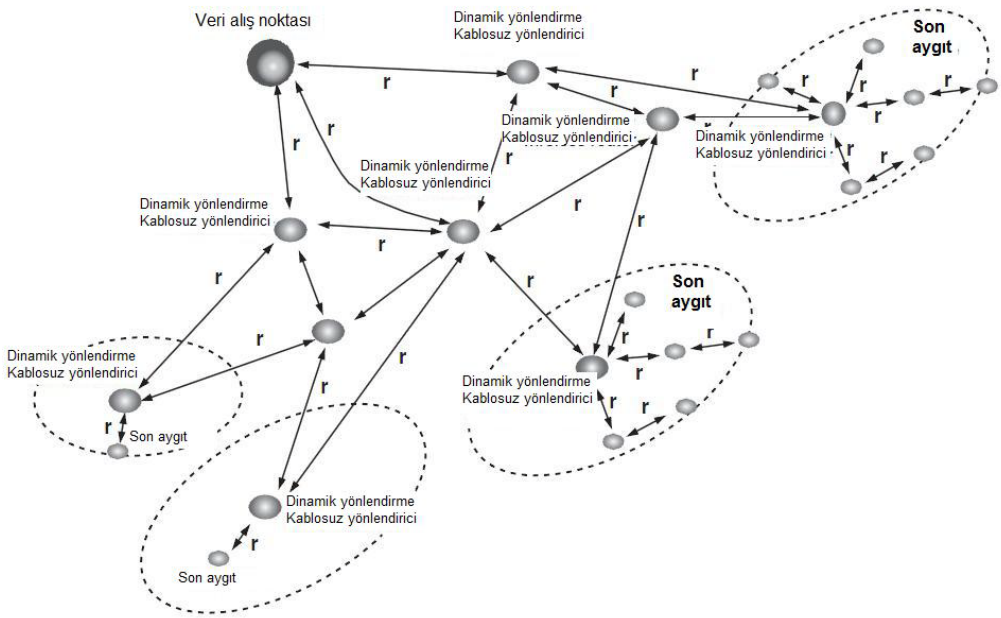
Uygulama alanına bağlı olarak KAA'lar çeşitli guruplara ayırabilir.

2.1.1. Kategori 1 KAA'lar

Ağın hem kablosuz hem de kablolu bölümlerinde dinamik yönlendirme kullanan ve kablosuz ağlar arasında çok sekmeli radyo bağlantısı içeren hemen hemen değişmeyen bir örgüsel (mesh) sistemdir. Gönderme düğümü dinamik yönlendirmeyi destekleyen kablosuz bir yönlendiricidir (örneğin; birden fazla olası yönlendirmeler dışında en iyi yönlendirmeyi bulmada kullanılacak bir mekanizmaya sahiptir); kablosuz yönlendiriciler genellikle kablosuz bağlantılar üzerinden bağlanır (Şekil 2.2.). Kategori 1 KAA'ların önemli özellikleri aşağıdaki gibidir [4]:

1. Algılayıcı düğümler yineleyici rolüne girerek diğer algılayıcı düğümler adına iletişimi destekleyebilir;

2. Gönderme düğümü dinamik yönlendirmeyi destekler. Ağın diğer kalan kısmı için de fiziksel ve mantık çerçevesinde birden fazla fiziki bağlantı yapma imkânı mevcuttur.
3. Radyo bağlantıları binlerce metre ile ölçülür.
4. Gönderme düğümü, algılayıcı düğümler adına veri işlemeyi ve azaltmayı destekleyebilir.



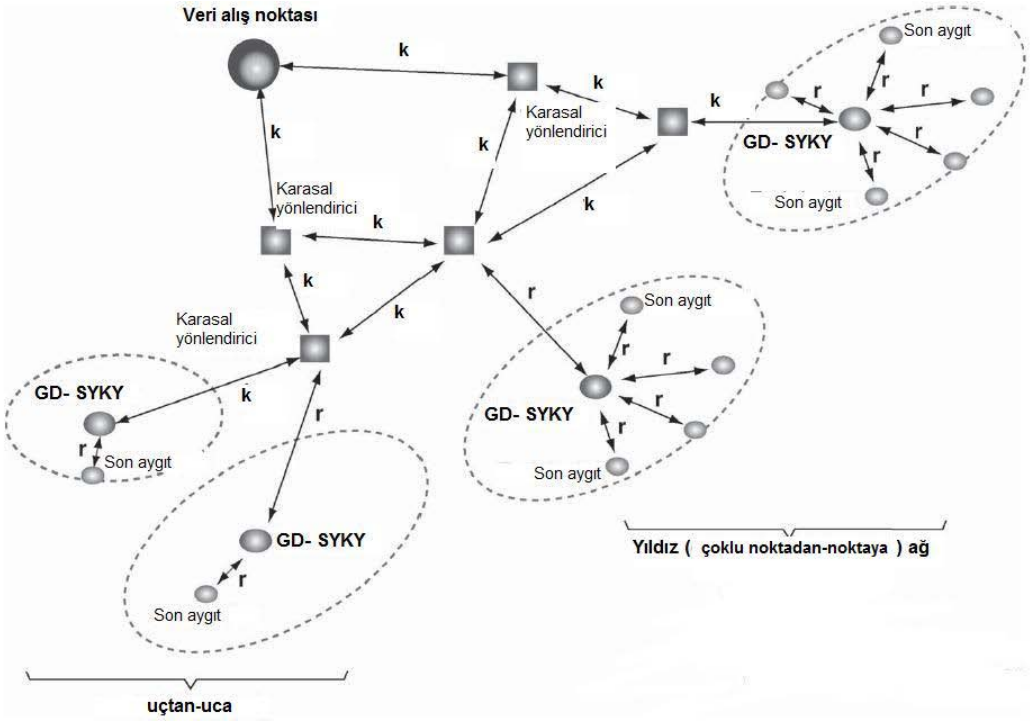
Şekil 2.2. Kategori 1 KAA. (çoklu sekme sistemleri, dinamik yönlendirmeyi destekleyen yönlendiricidir) [4].

2.1.2. Kategori 2 KAA'lar

Kablosuz ağ üzerinden sabit yönlendirme kullanan ve genellikle kablosuz ağlara yapılan tek sekmeli radyo bağlantısı ile uçtan uca veya çoklu noktadan noktaya (yıldız temelli) bir sistemdir. Genel yapısı itibarı ile kablosuz ağlardan ilgili karasal / kablolu gönderme düğümlerine sadece bir güzergâh olacaktır ve gönderme düğümü noktadan noktaya kablosuz bağlantı veya sabit bir hat yoluyla karasal ağa

bağlanacaktır. Şekil 2.3. bu kategorinin özelliklerini aşağıdaki gibi göstermektedir [5]:

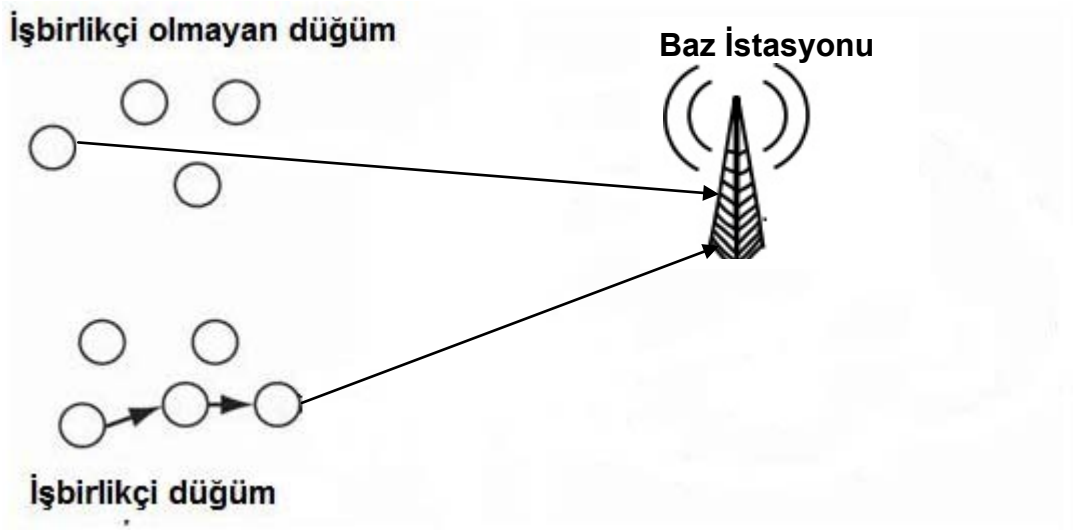
1. Algılayıcı düğümler diğer herhangi bir algılayıcı düğüm adına iletişimi desteklemez.
2. Gönderme düğümü karasal ağda sadece sabit yönlendirmeyi destekler ve / veya karasal ağda sadece bir tane fiziki bağlantı mevcuttur.
3. Radyo bağlantısı yüzlerce metre olarak ölçülür.
4. Gönderme düğümü, algılayıcı düğümler adına veri işleme ve azaltmayı desteklemez. Bu yapılar nispeten basit kablosuz sistemleridir.



Şekil 2.3. Kategori 2 KAA. statik yönlendirmeyi destekler [5]. (r:Radyo bağlantısı, k:karasal bağlantısı, GD-SYKY'ise Gönderme Düğümü-Statik Yönlendirme Kablosuz Yönlendirici).

Bazıları (i) işbirlikçi olarak (bir düğüm diğer bir düğüm adına bilgi gönderdiği zaman) (ii) işbirliksiz (bir düğüm sadece kendi iletişimini yürüttüğü zaman) olarak

iki türde özellik bünyesinde mevcuttur [6]. Şekil 2.4. bu iki tür iletişimide göstermektedir



Şekil 2.4. İşbirlikçi ve işbirlikçi olmayan düğüm.

Algılayıcı ve mikro algılayıcı kullanan KAA'ların ticari ve askeri sebeplerden ötürü fiziksel dünyayı algılamak ve kontrol etmek için uygulama çeşitliliği sağlaması beklenmektedir. Uygulamalar, bir yandan çevresel kontrol (örneğin toprak kirlenmesinin izlenmesi, yaşam alanı (habitat) denetimi), stok kontrolü ve sağlık bakımı öbür yandan da bilimsel ve askeri kullanımlar arasında değişiklik gösterir [7,8]. Araştırma seviyesindeki problemler üzerine yoğunlaşmış olan algılayıcı ağ biliminin gelişiminde KAA uygulamalarının bir sonraki adımı temsil ettiği inanılmaktadır [9].

Ticari kuruluşlar algılayıcı teknolojisise çoktan ilgi göstermiştir. Örneğin, sigorta şirketleri yangın tehlikelerini en aza indirmek amacıyla kreozot (katran ruhu) oluşumunu denetlemek için çatılarda algılayıcılar kullanma konusuyla ilgilendiklerini belirtmiştir [10]. Ayrıca bu şirketler buz hasarını önleme amacıyla su borularının sıcaklığını kontrol etme konusuyla da ilgilenilmektedir. Bu uygulamaların motivasyonu kayıpları ve ilgili masrafları azaltmaktır [10]. Tüketici

uygulamaları ciddi altyapı koruma ve güvenliği, sağlık bakımı, çevre, enerji, yiyecek güvenliği, üretim işleme ve yaşam kalitesi desteğini içermektedir [11]. KAA'ın uzaktan kumandalı ev ısıtma ve ışıklandırma, kişisel sağlık teşhisi ve otomatik otomobil bakım uzaktan ölçümü gibi bir takım yeni rahatlıkları tüketicilere sunması beklenmektedir. Ticari uygulamalar, endüstriyel ve inşaat denetlemeyi, cihaz kontrolünü, otomotiv algılayıcılar ve uyarıcıları, ev özdevinimlemesini, otomatik metre okuyucusunu, elektrik yük yönetimini, tüketici elektroniği ve eğlencesini ve mülk denetimini içerir. Çizelge 2.1.de ticari ve akademik araştırmalar için üretilen kablosuz algılayıcı düğüm platformları verilmiştir.

Çizelge 2.1.Ticari ve akademik araştırma için üretilen algılayıcı düğüm platformları [12]

Düğüm	Üretici	İşlemci	Güç	Hafza	Radyo
COTS Dust	Dust Network spun out of UC Berkeley	8 bit 150KHz ATMET MCU	3V	128B RAM 4Kb Flash	4.8Kbps
Rene Mote	Crossbow Corp.	ATMEL L8535	36μW sleep 60 μW active	512K RAM 8K Flash 32K EEPROM	10Kbps
Mica2	Crossbow Corp.	ATMEL ATmega 128L(8MHz)	< 1μA sleep 27ma active	128K Flash ROM 4K EEPROM	38.4Kbps
MicaZ	Crossbow Corp.	ATMEL ATmega 128L(8MHz)	<15μA sleep 8Ma active	128K Flash ROM 4K EEPROM	Chipcon 2420 (802.15.4ZigBEE) 250Kbps
BTnode	Art of Technology , Zurich	ATMEL ATmega 128L(8MHz)	9.9mW Idle 198mW active	64+180Kb RAM 128Kb Flash ROM 4Kb EEPROM	Bluetooth1.2 up to 723Kbps Chipcon CC1000 76.8Kbps
Sensoria2.0	Sensoria Corp.	Hitachi SH4(32-bit)	3V	64Mb RAM 32Mb Flash	208.11b
Rochwell-Hidra	Teledyne Scientific and Imaging	StrongARM 1100 (133MHz)	3V	4MB Flash 1MB SRAM	100Kbps

2.1. Algılayıcı Ağların Uygulamaları

KAA uygulamaları iki kategoriye ayrılabilir: ***denetleme ve izleme***. Denetleme uygulamaları iç/dış çevresel denetleme, sağlık denetleme, biyomedikal uygulamalar, güç denetleme, konum denetleme, fabrika ve işleme otomasyonu ve depremsel ve

yapısal denetlemeleri içerir. Bu uygulamalar aşağıdakiler gibi çok çeşitli durumlar için kullanılabilir [13]:

- Sıcaklık,
- Nem,
- Araç taşıma,
- Işıklandırma durumu,
- Basınç,
- Zemin düzeni,
- Ses seviyeleri,
- Belli türdeki objelerin varlığı veya yokluğu,
- Ekli objelerin mekanik stres seviyeleri ve
- Bir objenin hız, yön ve büyüklüğü gibi mevcut özellikler.

İzleme uygulamaları ise radyasyon ve nükleer tehdidi saptama sistemleri, gemiler için ufuk ötesi silah algılayıcıları gibi yüksek kaliteli uygulamalar bağlamında kullanılan objeleri, hayvanları, insanları ve araçları izlemeyi içerir.

Çok yakın zamanda ulusal güvenlik uygulamaları için ağ tabanlı biyolojik ve kimyasal algılayıcılar üzerine ilgi yoğunlaşmıştır [14]. Dahası doğrudan tüketici uygulamalarında giderek artan bir ilgi vardır. Algılayıcı ağların mevcut ve potansiyel uygulamaları askeri algılama, fiziksel güvenlik, hava trafik kontrolü, trafik denetimi, video denetimi, endüstriyel ve üretim özdevinimlemesi, işlem kontrol, kontrol yönetimi, dağıtımlı robot bilim, hava algılama, çevre denetleme, ulusal sınır denetleme ve inşaat ve yapı denetlemeyi içerir [14].

Bir dizi uygulama aşağıdaki gibidir:

- Askeri uygulamalar [15].
    - Düşman kuvvetlerini denetleme

    - Dost kuvvetleri ve ekipmanları denetleme
    - Askeri alanını veya savaş meydanını denetleme
    - Hedef alma
    - Savaş zarar değerlendirmesi
    - Nükleer, biyolojik ve kimyasal saldırı tespiti
- Çevresel uygulamalar
    - Mikro iklim
    - Orman yangın tespiti
    - Sel tespiti
    - Hassas tarım
- Sağlık uygulamaları
    - Fizyolojik verinin uzaktan denetlenmesi
    - Bir hastanede doktor ve hastaların izlenmesi ve denetlenmesi
    - İlaç yönetimi
    - Yaşlı bakımı
- Ev uygulamaları
    - Ev otomasyonu
    - Otomatik metre (ölçü) okuma
- Ticari uygulamaları
    - Endüstriyel ve ofis binalarında çevresel kontrol
    - Stok kontrolü
    - Araç izleme ve tespit
    - Trafik akış denetimi

2.1. KAA'ların Tasarımını Etkileyen Faktörler

Bir algılayıcı ağ tasarımı aşağıdakileri içeren bir çok faktör tarafından etkilenmektedir.

2.3.1. Hata toleransı

Hata toleransı, algılayıcı düğüm arızalarından dolayı, herhangi bir kesinti olmaksızın algılayıcı ağ işlevselliklerini sürdürme özelliğidir [16]. Bazı algılayıcı düğümleri, güç eksikliğinden dolayı hata verebilir veya durdurabilir ve fiziksel hasar veya çevresel etkileşime sahip olabilirler [17]. Algılayıcı düğümlerinin arızası algılayıcı ağının tüm görevini etkilememelidir. Bu güvenilirlilik veya arıza tolerans konusudur.

Hata toleransı mümkün olduğu zaman KAA'lar genellikle kendi kendini onarırlar [18]. Bu yüzden, ağ içinde karmaşık hata yönetimi yöntemleri uygulayarak ağ hizmetlerinin kalitesini devam ettiren ve bireysel düğümlerin enerjisini koruma yolu arasında ağ ömrünü uzatan bir dönüşümü vardır.

Bir algılayıcı düğümünün güvenilirliği $R_k(t)$ veya hata toleransı, zaman aralığı $(0, t)$ içerisinde bir hata yapmama olasılığını yakalamak için Poisson dağılımını kullanılabilir [19]:

$$R_k(t)=exp(-\lambda_k t)$$

λ_k ve t, k düğüm algılayıcısının hata oranı ve zaman periyodudur. Protokollerin ve algoritmaların algılayıcı ağlar tarafından istenen hata tolerans seviyesini karşılamak için tasarlanabilirler. Algılayıcı düğümlerin yerleştirildiği ortamın az bir etkileşimi varsa, protokoller daha gevşek olabilir.

2.3.2. Ölçeklenebilirlik

Ölçeklenebilirlik, ağın büyüklüğünde meydana gelen ayrılığın artışın ağın ve performansını etkilenmesidir.Tasarlanan protokol artan düğüm sayısı altında çalışabilir özellikte olmalıdır. Ayrıca algılayıcı ağlarının yüksek yoğunluk doğasını kullanmalıdır. Yoğunluk birkaç algılayıcı düğümünden bir bölgedeki birkaç yüz algılayıcı düğümüne kadar değişebilir. Düğüm yoğunluğu algılayıcı düğümlerinin yerleştirildiği uygulamaya bağlıdır. Makine teşhis uygulaması için düğüm yoğunluğu

bir 5×5 m^2 bölgede yaklaşık 300 algılayıcı düğümüyken, araç takip uygulaması için yoğunluk bölge başına yaklaşık 10 algılayıcıdır [19]. Genel olarak, yoğunluk 20 algılayıcı düğümü/m^3 kadar yüksek olabilir [19]. Ortam izleme uygulaması için, algılayıcı düğümlerin sayısı bölge başına 25 ila 100 arasında değişmektedir.

2.3.3. Güvenlik

Algılayıcı ağlar tehlikeli bir ortamda da faaliyet gösterebileceği için, güvenlik sistemi önceden düşünülmeli ve tasarım içine yerleştirilmiş olmalıdır. Saldırı ve kandırmalara karşı ağ korunmalıdır [20]. Üstelik KAA'lar tehlike ve risklere karşı savunmasızdır. Yabancı bir tehlike bir algılayıcı düğümü kandırabilir, veri bütünlüğünü değiştirebilir, mesajları dinleyebilir, sahte mesajlar gönderebilir ve ağ kaynağını boşa harcayabilir. Kablolu ağların aksine, kablosuz düğümler mesajlarını ortama yayınlarlar. Bu yüzden, KAA'da mutlaka güvenlik sisteminin bulunması gerekir. Güvenlik gerekliliklerinin uygulama seviyesini karşılamak için, bireysel düğümler, karmaşık şifreleme ve kimlik doğrulama algoritmalarını yürütecek kapasitede olmalıdır [21].

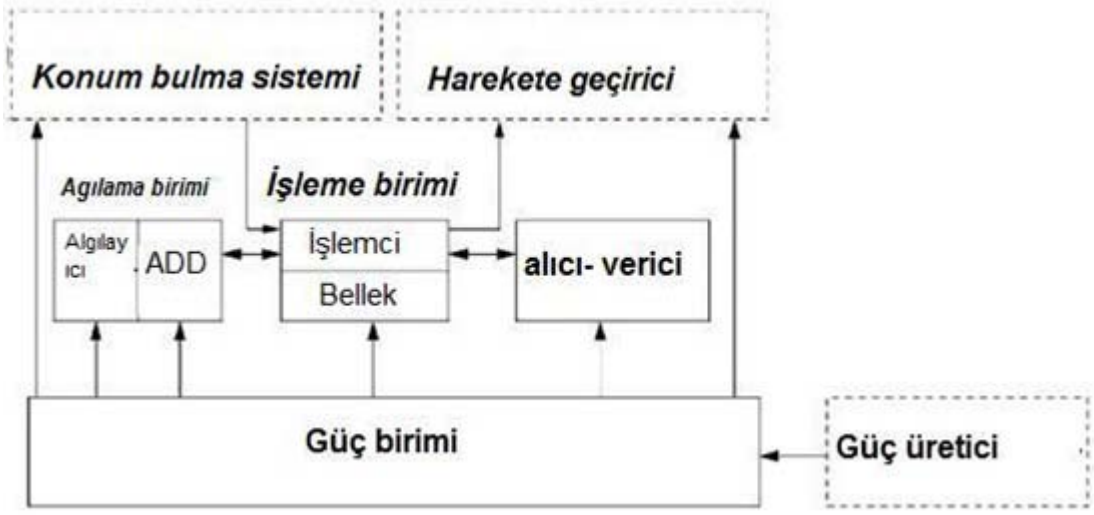
2.3.4. Maliyet

Maliyet genellikle uygulama ve üretim maliyetine bölünür. Sınırlı sayıda pahalı algılayıcıların uygulanması gerektiğinde uygulama maliyeti genellikle önemlidir ve üretim maliyeti ile doğru orantılı şekilde artar [22]. Algılayıcı ağların geniş sayıdaki algılayıcı düğümlerden oluşmasından dolayı, tek bir düğümün fiyatı, ağların toplam maliyetini doğrulamak için çok önemlidir. Sonuç olarak, her bir algılayıcı düğümünün maliyeti düşük tutulmak zorundadır. Bir algılayıcı düğümünün maliyeti, algılayıcı ağının uygun olması amacıyla 1 amerikan dolarından az olmalıdır [23]. Bir düğüm algılayıcısının aynı zamanda, algılama ve işletme üniteleri gibi bazı ek üniteleri vardır. Ek olarak, algılayıcı ağlarının uygulamalarına bağlı olarak bir konum bulma sistemi, nakledici (harekete geçirici) veya güç üreticiyle donatılabilir. Sonuç

olarak, bir dolardan daha az bir fiyat sınırı işlevsel özelliklerinden dolayı bir algılayıcı düğümünün maliyeti için son derecede zorlu bir konudur.

2.3.5. Donanım kısıtları

Bir algılayıcı düğümü Şekil 2.5'de gösterildiği gibi dört temel bileşenden oluşmaktadır: *algılama birimi, işleme birimi, alıcı-verici birimi* ve *güç birimi.* Ayrıca uygulamaya bağlı olarak, bir konum bulma sistemi, bir güç üreticisi ve bir nakledici (harekete geçirici) gibi ek bileşenlere sahip olabilirler. Algılama üniteleri genel olarak iki tane alt kümeden oluşmaktadırlar: algılayıcılar ve analog dijital dönüştürücüler (ADD' ler). Gözlemlenen fenomene dayalı olarak algılayıcılar tarafından oluşturulan analog sinyaller ADD tarafından dijital sinyallere dönüştürülmektedir ve ardından işletme birimine gönderilmektedir. Genelde küçük bir depolama birimiyle ilgili olan işletme birimi, tayin edilen algılama görevlerini gerçekleştirmek için algılayıcı düğümünün diğer düğümlerle işbirliği yapmasını sağlayan prosedürleri yönetmektedir. Bir alıcı verici birimi düğümü ağ'a bağlar [24]. Bir algılayıcı düğümünün en önemli bileşenlerinden birisi güç birimidir. Güç üniteleri güneş pilleri gibi bir güç tarama birimi tarafından desteklenebilir.



Şekil 2.5. Bir algılayıcı düğümün bileşenleri [24]

Bir harekete geçirici, tayin edilen görevleri gerçekleştirmesi istendiğinde, bazen algılayıcı düğümleri taşıması gerekebilir.

Tüm bu alt üniteler bir kibrit kutusu büyüklüğündeki bir modülün içerisinde sığmalıdır. Ayrıca algılayıcı düğümleri için başka kesin kısıtlar da mevcuttur. Bu düğümler [25]:

- Son derecede az güç tüketmelidirler.
- Değişik yoğunluklarda çalışmalıdırlar.
- Düşük üretim maliyetlerine sahip olmalıdırlar ve zorunlu olmamalıdırlar.
- Bağımsız olmalıdırlar ve gözetim gerektirmeden çalışabilmelidirler.
- Ortama uyabilmelidirler.

Algılayıcı düğümlerinin erişimi güç yada olanaksız bölgelerde olmalarından dolayı, bir algılayıcı ağının kullanım ömrü, düğümlerin güç kaynaklarının kullanım ömrüne bağlıdır. KAA düğümleri tipik lityum (Li) pillerle çalıştırılmaktadır [26]. Enerji taramasıyla (üretimi) algılayıcı ağlarının kullanım ömrünü uzatmak mümkündür, bunun anlamı güneş pilleri gibi ortamdan enerjinin emilmesidir [27].

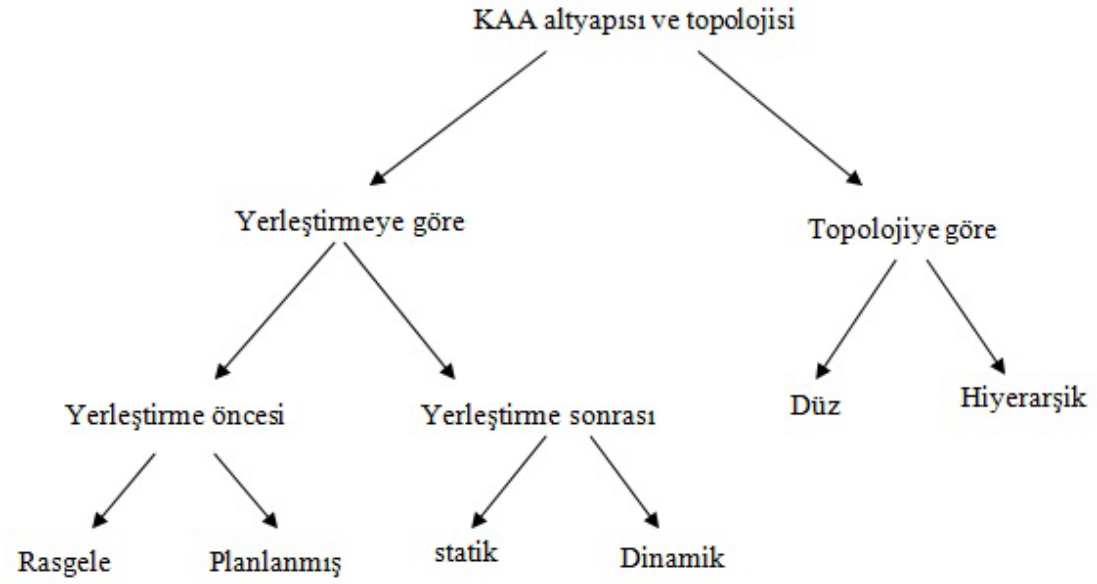
Daha yüksek programlama güçlerinin gitgide daha küçük işlemciler şeklinde mevcut oldukları düşünülürse, algılayıcı düğümlerinin işletme ve bellek üniteleri hala zor bulunan kaynaklardır. Örneğin küçük bir **Smart Dust Mote** esas modelinin işletme birimi bir 4 Mhz' lik Atmel AVR8535 mikro-kontrolörlü 8 KB işlemli flaş belleği, 512 bitlik RAM ve 512 bitlik EEPROM' dur [28] . 3500 bit işletim sistemi kod alanı ve 4500 bit mevcut kod alanına sahip olan bu işlemci üzerinde TinyOS işletim sistemi kullanılmaktadır [29] .

Algılama görevlerinin çoğu konumun bilinmesini gerektirmektedir. Algılayıcı düğümlerinin genel olarak rastgele yerleştirmelerinden ve gözetimsiz

çalışmalarından dolayı, bir konum bulma sistemiyle birleşmeleri gerekir. Konum bulma sistemleri ayrıca teklif edilen birçok algılayıcı ağ yönlendirme protokolleri tarafından talep edilmektedirler. Her bir algılayıcı düğümünün en az 5m kesinliğe sahip olan bir global konumlandırma sistemine "GPS" sahip olduğu sık sık varsayılmaktadır [30] .

2.3.6. Algılayıcı ağı altyapısı ve topolojisi

Tipik bir KAA topolojisi yoktur ve çoğunlukla uygulamaya bağlıdır [31]. Topoloji fiziksel koordinatlardan, ağ yapısı ve komşu düğümlerden etkilendiği için. Bu yüzden KAA için topolojiler tasarımı edilirken şekil 2.6. da gösterildiği gibi bazı dağıtım ve topoloji kontrol sorunları ortaya çıkar ve bu sorunların göz önünde tutulmaya ihtiyacı vardır.



Şekil 2.6. KAA'nın altyapısı ve topolojisi

Birçok algılayıcı düğümü hiçbir şekilde altyapısı olmayan bölgelerde yerleştirilmektedir. Bir ormanda yerleştirme yapmanın tipik bir şekli algılayıcı

düğümlerinin bir uçaktan havaya savrulmalarıdır. Yüksek sayıdaki düğümlerin yoğun bir şekilde yerleştirilmesi topoloji bakımının dikkatli bir şekilde yapılmasını gerektirir [29]:

a) Yerleştirme öncesi ve yerleştirme aşaması

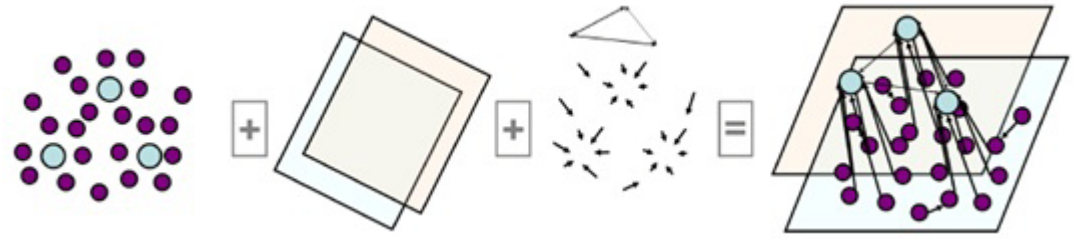
Düğümler, bir uçaktan savrularak, bir silah mermisi kovanında, roket veya füze içerisine yerleştirilerek, bir sapanla fırlatılarak (bir geminin güvertesinden vs.), fabrikada yerleştirilerek ve bir insan veya bir robot tarafından tek yerleştirilerek yerleştirilebilir. Bu nedenle asıl yerleştirme yöntemleri kurulum maliyetlerini azaltmalı, herhangi bir ön organizasyon ve ön planlama gereksinimini ortadan kaldırmalı, düzenleme esnekliğini artırmalı ve kendiliğinden organizasyonu ve arıza toleransını geliştirmelidir.

Algılayıcıların işleyeceği uygulama ve çevreye göre, ön-dağıtım tasarım aşaması, temel olarak rastgele (savaş alanı takibi ve orman yangını tespiti ve benzeri uygulamalar içindeki) ve planlanmış (binalardaki sıcaklık ve ışık gözlemi, köprüler, binalar ve benzerinin depremsel olay gözlemi) iki türe ayırılır [32]. Rastgele/gelişigüzel dağılım, büyük ölçekli KAA uygulamalarında daha mantıklıdır. Bu durumda, algılayıcı cihazlar verilen alan içinde genellikle uniform olarak dağılırlar. Bu strateji, dağılmanın kolaylığı sebebiyle tercih edilebilir. Fakat yerleştirme sırasında algılayıcı ağlar zarar görebileceğinden dolayı, pek güvenilir bir yöntem değildir (örneğin, havadan düşmesi) ve bilinmeyen bölgedeki engellerden dolayı, kapsama uniform olmayabilir. Başka bir seçenek ise, KAA tasarımının performans hedeflerine ulaşmak için algılayıcıları belirli bir şekilde yerleştirmektir. Bu durumlarda, dikkatli planlama ile başlama zamanında topoloji kurulur [18].

Ağ yapısının mutlaka belirlenmesi gerekmektedir. Buradaki türlerde genellikle düzgün ağlar homojen algılayıcılardan ve baz istasyonundan oluşur, daha basit yapıya sahiptirler ve oldukça küçük algılama alanlarının uygulamalarına uyum

sağlarlar. Hiyerarşik yapılar ise, farklı görevleri olan ve homojen olmayan algılayıcılardan oluşurlar ve daha yüksek karmaşıklık pahasına daha geniş uygulamalar desteklerler. Şekil 2.7. (a) ve (b).

(a) KAA'larda düz ağ yapısı



(b) KAA'larda hiyerarşik ağ yapısı

Şekil 2.7. KAA'ların ağ yapıları [18].

b) Yerleştirme sonrası aşaması

Yerleştirme sonrasında, topoloji değişiklikleri algılayıcı düğümlerinin sıkışma, gürültü, hareket eden engeller vs. dolayısıyla konum ulaşılabilirliliği, mevcut enerji ve görev detaylarında değişiklik meydana gelir [33,34]. Algılayıcı düğümleri statik bir şekilde yerleştirilebilirler. Bununla birlikte, enerjinin azalması veya yok olma nedeniyle cihaz arızası sıradan bir durumdur. Bunun yanı sıra, algılayıcı düğümleri ve ağ farklılık gösteren görev dinamiklerini tecrübe edinirler ve sıkışmayı açmak için bir hedef olabilirler. Bu nedenle algılayıcı ağ topolojileri yerleştirme sonrasında sık görülen değişikliklere meyillidirler.

c) Ek düğümlerin yeniden yerleştirme aşaması

Ek algılayıcı düğümler, arızalı düğümleri değiştirmek için veya görev dinamiklerindeki değişikliklerden dolayı herhangi bir zamanda yeniden yerleştirilebilirler. Yeni düğümlerin eklenilmesi ağın yeniden organize edilme gereksinimini doğurmaktadır.

2.3.7. Ortam

Algılayıcı düğümleri, gözlemlenecek olan fenomene çok yakın veya doğrudan onun içine yoğun bir şekilde yerleştirilmektedir. Bu nedenle genel olarak uzak coğrafik alanlarda gözetimsiz bir şekilde çalışmaktadırlar. Bir okyanusun dibindeki yüksek basınç altında, enkaz veya bir savaş alanı gibi zorlu ortamlarda, bir savaş donanım motorunun ağızlığı veya kutup bölgeleri gibi aşırı sıcak ve soğuk altında ve kasti sıkışma altında olduğu gibi son derecede gürültülü ortamlarda çalışmaktadırlar [34].

2.3.8. İletim ortamı

Düğümler arasındaki link radyo, kızıl ötesi veya optik ortam tarafından oluşturulabilir. Algılayıcı düğümleri için çoğu akım donanım RF devre tasarımına dayalıdır. μAMPS kablosuz algılayıcı düğümü, entegre bir frekans sentezleyicisiyle (birleştirici) birlikte bir Bluetooth uyumlu 2.4 GHz alıcı verici kullanmaktadır [35]. Ayrıca, bir algılayıcı anteni diğer kablosuz cihazlarındakilerin yüksekliğine ve radyasyon gücüne sahip olmayabilir. Bu nedenle, iletim ortamının seçimi, son derecede farklı olan bu kanal özelliklerini etkin bir şekilde modelleyen dayanıklı kodlama ve modülasyon yöntemleri tarafından desteklenmelidir.

2.3.9. Güç tüketimi

Bir algılayıcı düğümünün bir algılayıcı alanındaki ana görevi, olayları tespit etmek, hızlı lokal veri işletimi yapmak ve ardından verileri iletmektir. Güç tüketimi bu nedenle üç alana bölünebilir: algılama, iletme ve veri işletme [36].

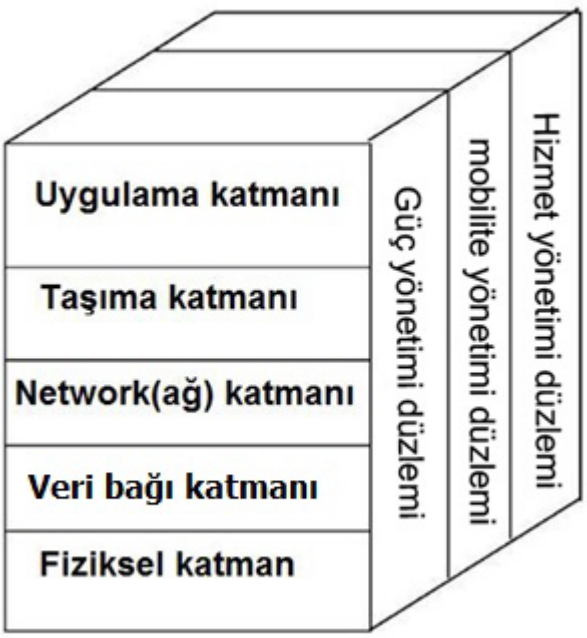
2.4. Algılayıcı Ağların İletişim Mimarisi (Protokol Yığını)

Baz istasyonu ve algılayıcı düğümler tarafından kullanılan protokol yığını, fiziksel katman, veri bağlantısı, ağ katmanı, taşıma katmanı ve uygulama katmanından oluşur [37]. Hiyerarşik yapı içerisinde, her bir katman yalnızca altındaki katmanın fonksiyonlarını kullanır ve fonksiyonu üstteki katmana ihraç eder. Fakat güç yönetimi ve güvenlik her katmanda dikkat edilmesi gereken konulardır. Örneğin, algılayıcı düğüm fiziksel katmanda tıkanıklık tehlikesine girerse, diğer katmanlarda bazı etkili güvenlik yöntemleri olmasına rağmen, bütün ağın güvenlik durumu bozulur. Eğer, bir katman için tek bir güvenlik çözümü yeterli olmaz ise, güvenliği sağlamak için tüm katmanları kapsayan bir bütünsel yaklaşım tercih edilebilir [38].

Ek olarak, güç, mobilite "hareketlilik" ve görev yönetimi düzlemleri, gücü, hareketi ve algılayıcı düğümleri arasındaki görev dağılımını takip etmektedir [37]. Bu düzlemler algılayıcı düğümlerinin algılama görevini koordine etmelerine ve tüm güç tüketiminin azaltılmasına yardımcı olmaktadır.

Güç yönetim düzlemi bir algılayıcı düğümünün kendi gücünü nasıl kullanacağını yönetir. Örneğin, algılayıcı düğümü kendi komşularından birisinden bir mesaj aldıktan sonra kendi alıcısını kapatabilir. Bunun nedeni çoğaltılmış mesajların alımını önlemektir. Ayrıca, algılayıcı düğümünün güç seviyesi düşük olduğunda, algılayıcı düğümü kendi gücünün zayıf olduğunu ve yönlendirme mesajlarına katılmayacağını kendi komşularına yayınlar. Kalan güç algılama için saklanılmaktadır. Mobilite yönetim düzlemi algılayıcı düğümlerinin hareketini tespit ve kayıt eder ve böylece kullanıcıya geri giden bir yol her zaman için

sürdürülmektedir ve algılayıcı düğümleri kendi komşuları olan algılayıcı düğümlerinin kimin olduğunu takip edebilir. Komşu olan algılayıcı düğümlerinin kimin olduğunu bilerek, algılayıcı düğümler kendi güçlerini ve görev kullanımlarını dengeleyebilirler. Görev yönetim düzlemi belirli bir bölgeye verilen algılama görevlerini dengeler ve planlar. Söz konusu bölgedeki tüm algılayıcı düğümlerinin algılama görevini aynı zamanda yapmaları gerekmez. Sonuç olarak, bazı algılayıcı düğümleri, kendi güç seviyesine bağlı olarak, görevi diğerlerinden daha fazla gerçekleştirmektedirler. Bu yönetim düzlemleri, algılayıcı düğümlerinin güç açısından etkili bir şekilde birlikte çalışabilmeleri, verileri bir hareketli algılayıcı ağda yönlendirebilmeleri ve algılayıcı düğümleri arasındaki kaynakları paylaşabilmeleri için gereklidir. Bunlar olmadan, her bir algılayıcı düğümü kendi başına çalışacaktır. Algılayıcı düğümlerinin birbirleriyle işbirliği yapabilmeleri çok daha etkili olur ve bu şekilde algılayıcı ağların kullanım ömürleri uzatılabilir [37].



Şekil 2.8. Algılayıcı ağlardaki protokol yığını [36].

- **Fiziksel katman:** Frekans seçimi, taşıyıcı frekans oluşturulması, işaret tespiti, modülasyon ve veri şifreleme işlemlerinden sorumlu katmandır.

Fiziksel Katman güvenli erişimi, frekans hoplaması ve yayılı spektrum kullanılarak sağlanabilir.

- **Veri bağı katmanı**: Gürültülü bir çevrede ve algılayıcı düğümlerin hareketi içinde, komşu katmanlarının erişimi ile iletim ortamına çatışma yöneltir.

  Ortam Erişim Kontrol Protokolleri (MAC), gücün farkında ve servis dışı bırakma (denial of service) saldırılara karşı dayanıklı olmalıdır [39]. TinySec [38], algılayıcı ağlar için önerilen bir bağlantı katmanı güvenlik protokolüdür.

- **Ağ katmanı:** Verinin varış noktalarına yönlendirmesi ile ilgilenir. Yönlendirme protokollerinin güvenli bir şekilde tasarlanması gerekir. Güvenlik yöntemlerinin sonraki aşamada birleştirilmesi çok etkili olmayabilir.

- **Taşıma katmanı:** Son kullanıcılar arasında veri akışını sağlar. Güvenilir veri aktarımı, uç uca bağlantılar üzerinden başarılabilir.

- **Uygulama katmanı:** Bu katman, uygulamalar yoluyla kullanıcılara bilgi erişimi sağlar. Kullanıcılar, algılayıcı okumalar toplamak için uygulama yoluyla sorgu ve istek gönderebilirler.

2.5. Algılayıcı Ağlardaki Güvenlik Tehlikeleri ve Saldırılar

KAA'lara karşı pek çok tehlike ve saldırılar, geleneksel ağlarınkilere benzemektedir. Yeni saldırılar ise, kablosuz iletim ortamı, özgül altyapı eksikliği ve son derece kısıtlı algılayıcı düğüm kapasitesi gibi algılayıcı ağların özelliklerinden dolayı şiddetlenmektedir.

Saldırılar, yetkisiz kullanıcılar trafından veya ele geçirilmiş yetkili kullanıcılar tarafından algılayıcı ağalara özel erişim olmadan başlatılabilir. Saldırılar, fiziksel katmandan uygulama katmanına kadar uzanmayı amaçlayabilir. Örnek olarak, fiziksel katmanda kanal tıkanıklığı [40], veri bağlantı katmanında MAC protokollerini engellemek [41-43], ağ katmanındaki yönlendirme protokollerine saldırı [44], taşıma katmanında sel basma, yerelleştirme hizmetlerini tehlikeye atmak [45,46] veya uygulama katmanındaki algılama okuyucu [47]. Tipik tehditler ve KAA'larda uygun savunma teknikleri Çizelge 2.2'de özetlenmiştir.

KAA'lara karşı saldırılar, iki farklı açıdan düşünülebilir: biri, güvenlik mekanizmalarına karşı yapılan saldırılar; diğeri ise, yönlendirme protokolleri gibi temel mekanizmalara yapılan saldırılardır. Bir sonraki bölümde KAA'lara karşı yapılan önemli saldırıları özetleyeceğiz.

Çizelge 2.2. KAA'larda tipik tehditler [48]

Tehdit	Katman	Savunma tekniği
Tıkanıklık (jamming)	Fiziksel	• Yayılmış spectrum • Düşük görev döngüsü
Tahribat (tampering)		• Tahribatı düzeltmek (tamper-proofing) • Etkili anahtar dağıtımı
Tükenme (exhausting)	Veri bağlantı	• Hız sınırlaması (rate limitation)
Çarpışma (collision)		• Hata düzeltme kodu (error correction code)
Yönlendirme bilgilerini değiştirmek	Ağ	• Kimlik doğrulama (authenticaton) • Şifreleme
Seçmeli ileti (selective forwarding)		• Yedekleme (redundancy) • İnceleme (probing)
Sybil atağı		• Kimlik doğrulama (authenticaton)
Sinkhole atağı		• Kimlik doğrulama • İzleme • Yedekleme
Solucan deliği (wormhole)		• Esenek yönlendirme • İzleme
Hello seli (hello flooding)		• İki yönlü kimlik doğrulaması • Üç yönlü hand shaking
Sel basma (flooding)	Taşıma	• Sınırlı bağlantı sayısı • Müşteri bulmacası (client puzzle)
Düğüm kopyalama atağı (colne attack)	Uygulama	• İki anahtar dağıtımı

2.5.1. Servis dışı bırakmak (denial of service)

Servis Dışı Bırakmak "DoS" [49], bir kaynağın ya da hizmetin hedef kullanıcıların ulaşamaması için yapılmış belirgin bir girişimdir. Normalde bu terimi yabancı bir

tehlikenin bir ağ fonksiyonunu yıkmak, bozmak veya yok etmek girişimi olarak kullansak da, DoS saldırısı, ağ kapasitesinin beklenen fonksiyonlarını azaltan veya hizmet dışı bırakan herhangi bir olay olabilir. Donanım arızaları, yazılım hataları, kaynak tükenmesi, aşırı yüklenme veya bu faktörlerin herhangi başka bir etkileşimi DoS saldırılarına sebep olabilir.

Algılayıcı ağlarda, her katman farklı DoS saldırılarına maruzdur [39]: fiziksel katmandaki tıkanıklık ya da tahribat; bağlantı katmanında çarpışma, tükenme ve haksızlık; ağ katmanında ihmal, hırs, özgüdüm, yanlış yönlendirme ve kara delikler; taşıma katmanında sel basma ve eş zamanlılığın bozulması (desenkronizasyon).

2.5.2. Aktarmadaki aldatma, değişme veya tekrar bilgi

Bir algılayıcı ağa en doğrudan saldırı, bir düğümden baz istasyonuna veya düğümler arasında yönlendirme bilgisinin değiştirilmesi, aktarma sırasında düğüm okuyucuları dahil olmak üzere, bilginin hedef alınmasıdır. Kablosuz aktarma, gizli dinlemelere maruz olduğu için, saldırganlar, kablosuz trafiği izleyebilir, değiştirebilir, aldatabilir, yeniden yayınlayabilir veya aktarımdaki paketleri atabilirler. Böylece, baz istasyonuna veya ağdaki diğer düğümlere uzlaşmış tehlikeli bilgi sağlanmış olur.

2.5.3. Sinkhole saldırıları

Kara delik atağı (blackhole) olarak da bilinen Sinkhole Saldırıları [50,53], ağın içinde bir sinkhole veya kara delik oluşturarak belirli bir alandan hemen hemen tüm trafiği çekmeyi amaçlarlar. Sinkhole Saldırıları, pek çok fırsatla paketlerin yönünü karıştırarak bir tehlike oluştururlar ve seçici yönlendirme gibi diğer saldırıların başlamasına olanak sağlarlar.

Sinkhole saldırıları tipik olarak diğer düğümlere yönlendirmeyi düşman yoluyla özel çekici olduğunu göstermek ile yönlendirme protokollerini kandırılar. Sinkhole saldırılarının algılayıcı ağlara karşı etkileyiciliği, onarlın (KAA) bütün düğümlerin

paketlerini baz istasyonuna göndermek istediği özel iletişim modellerinin sonucudur. Tehlikeli bir düşmanın komşu düğümleri yalnızca düşman yoluyla baz istasyonu için belirlenen paketleri sevk etmeyecek ayrıca bu çekici yönü komşularına da tanıtacaktır. Sonuç olarak bir düşman, düğümlerden baz istasyonuna kadar, baz istasyonu için belirlenen tüm trafiği etkili bir şekilde çeker.

2.5.4. Sybil atağı

Sybil atağı [51,52] KAA'lar için önemli bir yönlendirme atağıdır. Sybil atakta bir kötücül algılayıcı düğümü kendisini ağdaki diğer düğümlere birden fazla kimlikle tanıtır. Bu durumda, kurban olarak seçilmiş olan düğüm bu kötücül düğümden gelen mesajları farklı düğümlerden geliyormuş gibi algılar. Kötücül düğüm bu şekilde kurban olarak seçtiği düğümlerin mesaj alıp vermesini engelleyebilir. Dahası, Sybil atak yolu ile bir kötücül düğüm sürekli yanlış bilgi göndererek ağda toplanan bilgiyi fazlasıyla değiştirebilir. Böylece baz istasyonunda yanlış bilgi toplanmasına neden olarak karar verme mekanizmasını yanıltabilir [52,54].

2.5.5. Solucan deliği atağı (wormholes)

Solucan deliği atağında [55], düşman bir yerdeki paketleri alır ve kontrol ettiği düşük gecikmeli bağlantı ile ağ içindeki başka bir bölgeye tünel yolu ile aktarır. Solucan deliği atağı, algılayıcı ağlar için çok önemli bir tehlike oluşturmaktadır. Çünkü yönlendirme protokolleri şiddetli bir şekilde parçalayabilir. Solucan deliği atağına karşı savunma yapmak için bazı mekanizmalar olmasaydı, pek çok mevcut yönlendirme protokolü düğüm çiftleri arasındaki istikrarlı yönleri yapılandıramazdı. Şayet düşman solucan deliği yoluyla tüm paketleri tünellerle gerçekten ve güvenilir bir şekilde ağın bir yerinden başka bir yerine taşırsa, düşmanın paketleri daha etkili bir şekilde aktararak yararlı bir hizmet sağladığı sanılır. Ancak, bir düşman bu yolla normalde baz istasyonundan çok uzağa sıçrayan düğümleri ikna eder ve düğümler

baz istasyonuna daha fazla yakınlaşır. Bu durum sinkhole yaratır ve böylece muhtemelen seçici yönlendirme veya gizli dinlemeleri ile birlikte kullanılabilir.

Solucan deliği ataklarında, saldırganın ağdaki meşru bir düğümle uzlaşması gerekmez. Dahası, hem iç hem de dış kullanıcılar tarafından başlatılabilirler. Bu çeşit bir saldırı, iki algılayıcı düğümün komşu olduklarını inandırmak için başlangıç ağ kurulum evresinde bile yapılabilir.

2.5.6. Düğüm kopyalama (node clone) atağı

Tehlikeli ortamlarda dağıtılan algılayıcı düğümler, yakalanmaya ve kandırılmaya karşı maruzdurlar. Böylece düşman, bu düğümlerden gizli bilgiyi alabilir, kopyalayabilir ve çeşitli iç saldırılar başlatmak için onarlı zeki bir şekilde ağ içinde dağıtabilir [56].

2.6. Saldırılara Karşı Önlemler ve Güvenlik Yöntemleri

Bir önceki bölümde verilen saldırıları önlemek için çeşitli yöntemler geliştirilmiştir.

2.6.1. Şifreleme

Şifreleme yöntemi, paketlerin gizli dinlenmelerine, enjeksiyona, kandırılmalarına ve değiştirilmelerine karşı standart bir savunmadır. Şifreleme ve şifreleme bazlı doğrulama mekanizmaları, algılayıcı ağlar için güvenlik protokollerinin yapı taşlarıdır.

Bazı MANET ağ güvenlik mekanizmaları, açık anahtar şifrelemeye dayanmaktadır [38, 57, 58]. KAA'larda açık anahtar şifreleme mevcut ağ düğümleri için çok pahalıdır ve muhtemelen de öyle kalacaktır.

KAA'lar için güvenlik protokolleri, özellikle etkili simetrik şifrelemeye güvenmelidir [59]. Asimetrik şifreleme aksine, simetrik şifreleme, iki ya da dört

sefer arası hesaplamayı daha fazla hızlandırır ve daha az enerji sarf eden fonksiyonlara sahiptir.

Şifreleme yöntemi, algılayıcı düğümler için seyrek bulunan kaynaklar olan hesaplama, paket ebatı, hafıza ve güç tüketimi bakımından masraflı olurlar. Şifrelenmiş hesaplamalar aktarım ile birlikte çakışabileceği için, genel performansın çoğu paket ebadının arttırılmasına dayandırılmıştır [61]. Şifreleme, şifreleme-bazlı güvenlik mekanizmalarından sonuçlanan ekstra uçların aktarımı ve işletiminin güç tüketimi sebebiyle, algılayıcı ağlar için güç yönetimi problemini şiddetlendirmektedir.

2.6.2. Anahtar dağıtımı

Anahtar dağıtımı, yetkili taraflar arasındaki anahtarlama ilişkilerinin bakımı ve kurulumunu destekleyen teknik ve yordamlar takımıdır ve aşağıdaki maddeleri kapsamaktadır [60, 61]:

- Sistem kullanıcılarının bir etki alanı içerisinde kullanıma başlatması.
- Anahtarlama materyalinin oluşturulması, dağılımı ve kurulumu.
- Anahtarlama materyalinin kullanımının kontrolü.
- Anahtarlama materyalinin güncellemesi, iptali ve yürürlükten kaldırılması.
- Anahtarlama materyalinin depolama, yedekleme/kurtarma ve arşivlemesi.

Anahtarlama ilişkisi, şifreleme (anahtarlama) teknikleri içindeki devrelerin ortak veriyi paylaştığı durumdur [62]. Bu veri, açık veya özel anahtar çiftlerini, gizli anahtarları, başlangıç değerlerini ve gizli olmayan parametreleri (katsayıları) içerebilir.

Genel anahtar dağıtım problemi, gizlilik ve doğrulama gibi güvenlik özelliklerinin sağlanması için iletişimde olan taraflar arasında dağıtılan gizli anahtarların taslağını ifade eder.

Anahtar dağıtımı, KAA'ların tehlikeli veya yabancı bir dış ağ ile uzlaşmalarını önlemek amacıyla gizlilik, bütünlük ve doğrulamanın güvenlik hedeflerini gerçekleştirmesi açısından çok önemlidir. Çünkü güvenli iletişimin temeli doğrulamadır. Güçlü bir kimlik doğrulama mekanizması olmayan bir sistemde, güvenlik hedefleri (gizlilik, veri bütünlüğü ve tanımlama gibi) pek çok durumda başarılamaz [63]. Kimlik doğrulama, yalnızca bilinen bir şeyin kimlik verisi ile ilişkili olduğunu doğrulama yoluyla anlaşılabilir. Elektronik bir ortamda, kimlik sahibi, kimliği ile ilişkili kamuya doğrulanabilir bir parolaya sahip olmalıdır. Aksi takdirde, düğüm kişiselleştirilebilir [62].

Anahtar dağıtımı yöntemlerinin temel işlevi, sırayla anahtar üzerinde bölünebilmeyle ve bu anahtarının taşınması ile oluşan anahtarlama materyalinin kurulumudur [62]. Anahtar uyumu iki ya da daha fazla tarafın veya ilişkilendirme ile her bir protokol katılımcının ortak bilginin işlevi olarak paylaşılan anahtarlama materyalini sağlamasına olanak sağlar. Öyle ki, hiçbir taraf sonuçlanan değeri önceden tahmin edemez [62]. Anahtar taşıma protokollerinde, bir taraf anahtarlama materyalini oluşturur veya elde eder ve böylece güvenli bir şekilde diğer tarafa ya da taraflara onu aktarır. Hem anahtar uyumu hem de anahtar taşınması ya simetrik ya da asimetrik teknikler kullanılarak başarılabilir. Karışık bir anahtar dağıtım yöntemi her iki teknikten de fayda sağlamak için kullanılır.

2.7. KAA için Anahtar Dağıtımı Yaklaşımları

KAA'ın güvenliği için temel sorun, iletişimde bulunan düğümler arasında gizli anahtarlar kurarak, algılayıcı düğümler arasında güvenli iletişimi başlatmaktır. Bu işlem anahtar dağıtımı olarak adlandırılır.

Bir anahtar dağıtım protokolü, güvenli altyapıyı sağlamak için yalnızca yeni dağıtılan algılayıcı düğümleri değil, ayrıca, sonradan dağıtılan düğümlere de güvenli bir şekilde ağa katılması için yetki vermelidir.

Üç çeşit anahtar dağıtım yöntemi vardır [64, 65]: güvenilir-sunucu yöntemi (trusted-server), kendi kendini güçlendiren yöntem (self enforcing) ve anahtar ön-dağıtım yöntemi (key pre-distribution). Güvenilir- sunucu yöntemi, güvenilir bir sunucuya bağlıdır örneğin, Kerberos [66]. Kendi kendini güçlendiren yöntem (self-enforcing), açık anahtarlar kullanarak asimetrik şifrelemeye bağlıdır. Diffe-Hellman [67] ve RSA [68] gibi açık anahtar algoritmaları, yüksek hesaplama kaynakları gerektirir. Anahtar ön dağıtım yönteminde. Anahtarların düğümlere ağ kurulumundan önce verilerek olasılıksal bir biçimde anahtar paylaşımı yapmaktadır.

2.8. Anahtar Dağılım Protokolünün Gereklilikleri

Belirli bir KAA'da uygulanan anahtarlama teknikleri, verimli olması için bazı gereksinimleri karşılamalıdır.

- **Düğüm ele geçirilmesine karşı esneklik:** Algılayıcı ağdaki düğümler dağıldıktan sonra, saldırgan düğümlere karşı fiziksel bir saldırıda bulunabilir ve hafızasındaki gizli bilgiyi okuyabilir. Algılayıcı ağında bir düğüm ele geçirilmişse, anahtarlama teknikleri gizli bilginin diğer düğümlere aktarılmasını engellemelidir. Bir yöntemin esneklik durumu, tehlikeye maruz kalan toplam ağ iletişimlerinin kesiti ile toplam düğüm sayılarını karşılaştırma ile hesaplanır. Fakat bu evre, uzlaşan ağların

doğrudan dâhil edildiği iletişimleri kapsamaz. Hata toleransı ayrıca yeni eklenen ağların güvenli iletişime katılmasını sağlamaktadır.

- **Düğüm Kopyalamasına Karşı Direnç:** Ağa giren saldırgan bazı düğümlerle uzlaşarak ve onları taklit ederek ağa zarar verebilir. Bu atağı kullanarak bir virüs tüm ağı tehlike altına alabilir ve böylece tüm ağın kontrolünü ele geçirebilir. İyi bir anahtarlama yöntemi sayesinde böyle saldırılara karşı direnç sağlanabilir.

- **İptal/Yürürlükten Kaldırma:** Bir algılayıcı ağı saldırgan tarafından saldırıya uğramışsa ya da, ağda uyumsuz hareket eden düğümler tespit edildiyse, anahtarlama teknikleri bu düğümleri iptal etmeli ve doğrudan sistemden uzaklaştırmalıdır.

- **Aslına Uygunluk/ Kimlik Doğrulama:** Anahtarlama teknikleri, ağ içindeki iletişim düğümlerinin diğer düğümlerin kimlik doğrulaması yaptığını garantilemelidir. Örneğin, alıcı düğüm, gönderici düğümün veri doğrulamasını onaylamalıdır.

- **Gizlilik:** Anahtarlama teknikleri, gizli bilginin ya da verinin açığa çıkmasını yetkisiz taraflardan korunmasını sağlamalıdır. Yabancı bir tehlike ya da virüs, veri elde etmek için, gizli anahtarları elde etme yoluyla tüm ağa saldırabilir. İyi bir anahtarlama tekniği, verinin sonradan ortadan çıkmasını önlemek amacıyla düğümleri kontrol eder.

- **Bütünlük:** Bütünlük, aktarma sırasında veri yanılgısı veya tahribatı olmamasıdır. Burada, anahtarlama teknikleri bakımından, anlamlar aşağıdaki gibi genellenmektedir. Yalnızca ağdaki düğümler anahtarlara erişmelidir ve yalnızca tanımlı baz istasyonu anahtarları değiştirme yetkisine sahip olmalıdır. Böylece, kullanılan anahtarlar hakkında yetkisiz

düğümlerin bilgi edinmesi ve dış kaynakların güncellemeleri etkili bir şekilde önlenecektir.

- **Ölçeklenebilirlik:** Algılayıcı ağlarının, ağın tipik boyutu içindeki çeşitlenmelerine izin vermesi için, ölçeklenebilir anahtarlama tekniklerinden faydalanması ile verimlilik, gerçekleşir. Anahtarlama teknikleri küçük ağlar için yüksek güvenlik duvarları sağlamalı ve bu özellikleri daha büyük ağlar için de devam ettirmelidir.

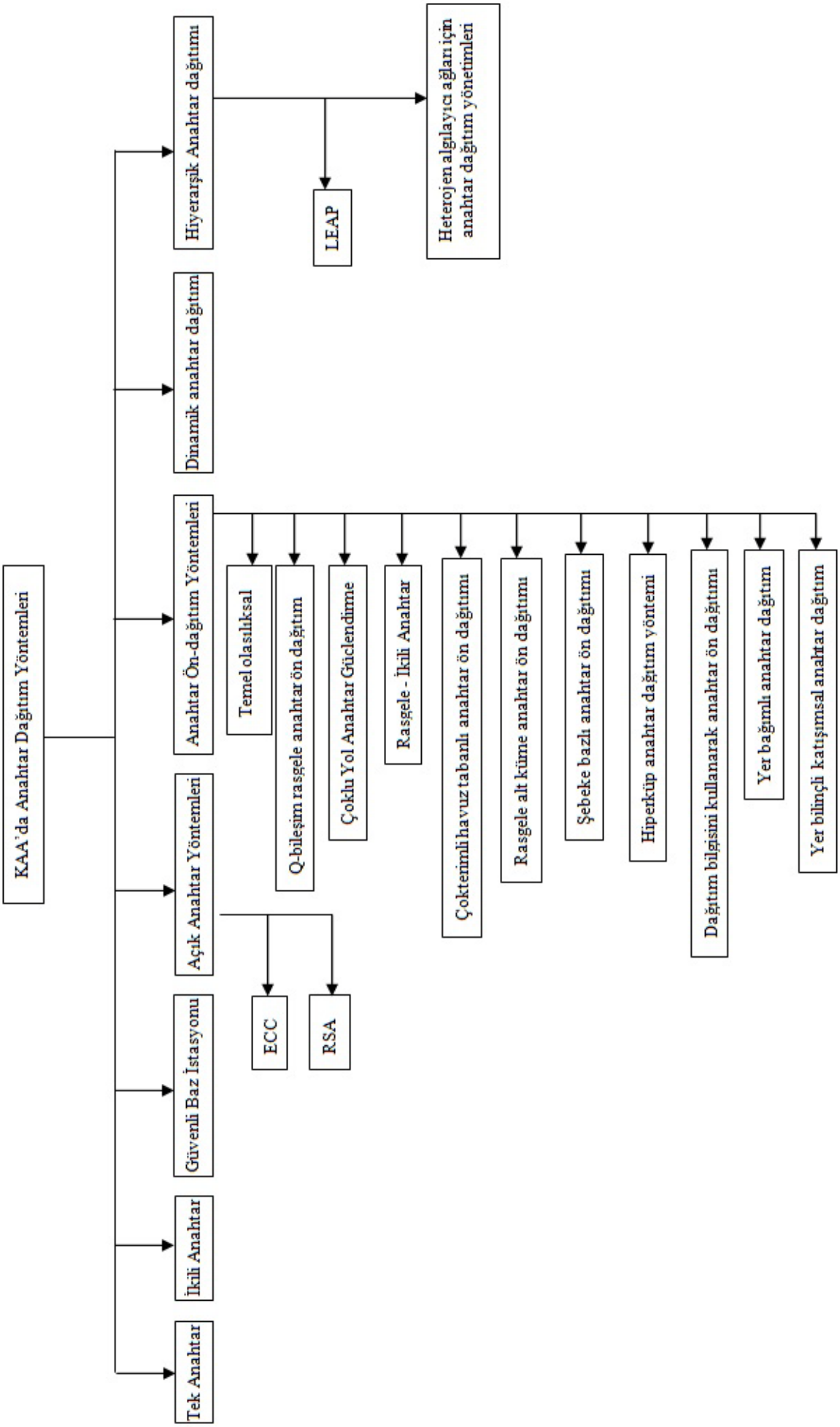
- **Bağlanırlık:** Dağılan algılama ağları, birbiriyle son derece bağlantılı olmalıdır. Anahtar bağlantısı, yalnızca şu şartlardan ikisi gerçekleştiği takdirde bağlantılı olduğu düşünülen iki düğüm anlamına gelir : (1) fiziksel komşudurlar ve (2) en azından bir gizli anahtar paylaşırlar.

- **Esneklik:** Anahtarlama teknikleri her çevresel ortamda ve sistemde iyi çalışmalı ve düğümlerin dinamik dağılımını desteklemelidir. Örneğin, bir anahtarlama tekniği çoklu uygulamalarda yararlı olmalı ve herhangi bir zamanda başka düğümlerin eklenmesini sağlamalıdır.

- **Verimlilik:** Tasarlanan yöntemlerin sınırlı hafızadaki algılama ağları, hesaplama ve iletişim kapasitesi için verimli olması gereklidir.

3. MEVCUT ANAHTAR DAĞITIM YÖNTEMLERİ

Bu bölümde KAA'da yaygın olarak kullanılan anahtar dağıtım yöntemleri detaylı olarak açıklanmaktadır.



Şekil 3.1. KAA'da anahtar dağıtım yöntemleri

3.1. Tek Anahtar Yöntemi

Tek anahtar yöntemi en basit anahtar belirleme tekniğidir. Bu tekniğin başlatma aşamasında bir tekli bir anahtar ağın tüm düğümleri içerisine önceden yüklenir. Dağıtımdan sonra ağ içindeki her bir düğüm bu anahtarı iletileri şifrelemek ve iletilerin şifreleri çözmek için kullanabilir [60,68]. Minimum düzeyde depolama gerektirmesi ve kompleks protokollerden kaçınması bu tekniğin sunduğu avantajlardan bazılarıdır. Düğümlerin belleği içinde sadece tek bir anahtar depolanacaktır ve bir defa ağ içine dağıtım gerçekleştikten sonra, iletişim aralığı içindeki tüm düğümler daha önceden paylaşmakta oldukları anahtarı kullanarak iletilerin aktarımını gerçekleştirebilecekleri için herhangi bir düğümün anahtar keşfi ya da anahtar alış verişi yapmasına gerek yoktur [68].

Her ne kadar tek anahtar avantajlı görünebilse de bunun temel sakıncası tek bir düğümün güvenliğinin tehlikeye düşürülmesinin paylaşılan anahtar dolayısıyla tüm ağın güvenliğinin tehlikeye düşürülmesine neden olmasıdır. Bu yöntem bazı sıkıntılara daha az hesaplama ve azalan bellek kullanımı ile cevap vermekteyse de saldırıda bulunmayı amaçlayan bir düşman için bunu kolaylaştırarak bir algılayıcı ağının temel gerektirimlerini yerine getirmede başarılı değildir [68].

3.2. İkili Anahtar Yöntemi

İkili anahtar yöntemi düğümden düğüme belgeleme ve düğüm ele geçirilmesine karşı yüksek direnç sunmaktadır. n düğümden oluşan KAA'da, toplam $\binom{n}{2}$ benzersiz anahtar bulunur. Her düğüm $(n-1)$ anahtar depolar ağda her düğüm için bir anahtar ayırılır.

İkili anahtar yöteminde n düğümden oluşan ağda, her bir düğüme oluşturabileceği her düğüm çifti için ikili anahtar atanarak anahtar öndağıtımı gerçekleşir. Her düğüm hafızasında n-1 İkili anahtar tutulur. Böylece her düğüm kapsama alanındaki diğer düğümlerle bağlanabilir [68].

Bu yöntem de düğüm ele geçirilmesine karşı artan direnç sunmaktadır ki düğüm hasım tarafından yakalanmışsa gizliliği bozulan düğüm kendisiyle doğrudan iletişimi olmayan düğümle ilgili bilgi vermez. Artan direnç sayesinde bu yöntem düğümün kopyalarak çoğaltma şansını en aza indirir [68].

İkili anahtar dağıtım yöntemin dezavantajları, ağdaki diğer düğümlerle $(n-1)$ benzersiz anahtar kurmak için ve bu anahtarları kendi hafızasında tutmak için gereken ek yükttür. Bu yöntemi kullanmak ağ boyutunu engellemek demektir, çünkü ağda düğüm sayısı arttığı zaman her düğüm hafızasında anahtar sayısı da artmak demektir. örneğin 10.000 düğümden oluşan ağ her düğüm belleğinde 9999 anahtar tutmalıdır [68].

3.3. Güvenli Baz İstasyonu

Anahtar dağıtımının bu yöntemi, algılama ağlarına bağlantı anahtarları sağlamak için belirleyici olarak güvenli baz istasyonu kullanmaktadır. Algılayıcı düğümleri, baz istasyonu bir bağlantı anahtarı oluşturduktan ve bu anahtarı her iki tarafa güvenli bir şekilde aktardıktan sonra, kendilerini baz istasyonuna onaylatırlar. Protokolun özeti şu şekilde olabilir. Dağıtımdan önce, ağdaki her bir düğüm için özgün simetrik bir anahtar oluşturulur. Bu düğüm anahtarı düğümün hafızasında saklanır ve düğüm ve baz istasyonları arasındaki anahtarlı iletişimleri kolaylaştırmasının yanı sıra, düğüm için kimlik doğrulayıcı olarak da hizmet eder. Baz istasyonu bütün düğüm anahtarlarına ya doğrudan (hafızası içinde saklanmış) veya dolaylı olarak (baz istasyonu tüm bağlantıları güvenli bir iş istasyonuna aktarır) erişime sahiptir [69].

Bu yöntem, herhangi iki düğüm arasındaki iletişim için oturum anahtarlarını göndererek, ikili yöntem sorununun üstesinden gelir. Bu yöntem ayrıca, KDC, Anahtar Dağıtım Merkezi ya da *Centralized Key Distribution Center* yaklaşımı olarak da adlandırılır [69].

Yöntem, küçük hafıza gereksinimlerine ve mükemmel kontrol edilen düğüm taklidine veya bütün ağa nüfuz etme özelliklerine sahiptir. Sistem, düğüm yakalanmasına müsait olmakla birlikte, anahtar eşleşmelerini iptal etmek mümkündür. Dezavantajlar ise, sistemin ölçeklenebilir olamaması ve baz istasyonunun saldırılarının hedefi haline gelmesidir [69].

3.4. Açık Anahtar Yöntemleri

Birçok simetrik anahtar ve Açık olmayan anahtar kriptolama yöntemleri hafıza, hesaplama ve iletişim kapasitesi ve güç kaynağı açılarından sınırlıdır, Açık anahtar yöntemleri kadar esnek değildir. Algılayıcı düğümleri ileri (sofistike) kriptografi teknolojilerini kullanamaz.

Dağılımdan önce, ana açık/özel anahtar çifti,(K_M , K^{-1}_M) öncelikle oluşturulur. Sonra, her A düğümü için Açık/özel anahtar çifti (K_A , K^{-1}_A) oluşturulur. Bu anahtar çifti, ana açık anahtar K_M ve A'nın açık anahtarı üzerindeki ana anahtarının imzası ile birlikte A düğümün hafızasında saklanır.Bu yolla tüm düğümler başlatıldığı zaman, dağılım için hazır hale gelirler [70].

Düğümler dağıldıktan sonra, anahtar değişimi gerçekleşir. Düğümler kendi açık anahtarlarını ve ana anahtar imzalarını değiştirirler. Her düğümün açık anahtarı, esas açık anahtar kullanılarak ana anahtarının imzasını doğrulama yoluyla yasal olarak doğrulanır. Bir düğümün açık anahtarı alındığı zaman, simetrik bir bağlantı anahtarı oluşturulabilir ve ona gönderilir, açık anahtar ile anahtarlanır. Oturum anahtarının alınması üzerine, anahtar dağıtımı tamamlanır ve simetrik bağlantı anahtarı kullanılarak iki düğüm arasında iletişim sağlanabilir [70].

Yaygın olarak kullanılan iki açık anahtar algoritması bulunmaktadır: ECC ve RSA

a. ECC (Elliptic Curve Cryptography): 1k RAM ve 34k ROM kullanarak 8 MHz işlemcilerle KAA'da kullanılmıştır [71,72 ve 73]. İki terimli (binary polynomial),

yavaş işlemciler tarafından yeterli kadar desteklememesi nedeniyle ECC asal sayı (prime integer) olarak KAA'da uygulanmaktadır [72].

b. RSA (Rivest Shamir Adlewan algoritması): RSA yöntemi genellikle 512-2048 bitlik anahtarlar üretir [70,72]. RSA yöntemi (M) Mesajı alır ve (K)anahtarı kullanarak bir (c) şifre metni üretir, RSA'yı hızlandırmak için (CRT) yöntemi kullanılabilir. böylece CRT hesaplama süresini ¾ oranında arttırır.

Hem ECC hem de RSA adanmış özel yardımcı-işlemciler araclıyla hızlandırılabilir. ECC'de hesaplama süresi daha hızlıdır, anahtarlar daha küçüktür ve RSA'dan daha az hafıza ve bant genişliği kullanır. ECC'de işlemler doğrosal (linear) olarak ölçülür. Bu da küçük kelimeli işlemcilerde (small word size) ECC'ye RSA karşısında avantaj sağlar. ECC küçük anahtar kullanarak RSA ile aynı güvenlik düzeyini elde edebilir. RSA ve ECC yöntemlerinin örnekleri,160 bitlik ECC anahtarı 1024 bitlik RSA anahtarıyla aynı güvenlik düzeyine sahiptir. 224 bitlik ECC anahtarı 2048 bitlik RSA anahtarına eşdeğerdir [73].

3.5. Anahtar Ön-dağıtım Yöntemleri

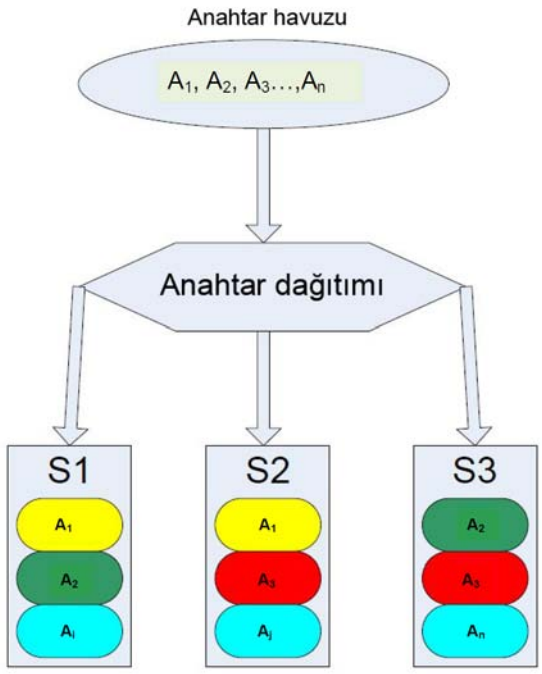
Bu tip yöntemlerde ilk olarak birkaç anahtar her bir algılayıcı düğümüne yüklenir, sonra bu algılayıcı düğümleri dağıtım alanına yerleştirilir. Yerleştirmeden sonra, düğümler arasında ortak anahtarları bulmak için bir keşif işlemi yapılır. Bu yöntem, herhangi iki algılayıcı düğümü arasında iletişim kurabilmek için ikili anahtar yönteminden faydalanabilir.

3.5.1. Anahtar ön-dagıtım aşamaları

Anahtar dağıtımı üç aşamadan geçmektedir [74].

a) Anahtar ön-dağıtımı ve düğümler arasında depolama aşaması.

Anahtar ön dağıtım aşaması, algılayıcı düğümler dağıtılmadan önce meydana gelir. Bu aşamada, anahtarlar ya da anahtar materyalleri her bir düğümün içine yerleştirilir. Büyük bir P anahtarlar havuzu üretmek, her bir algılayıcının hafızasına anahtar halkası yüklemek ve anahtar kimliklerini (ID) anahtar halkasında saklamaktan oluşur [75]. Şekil 3.2. anahtar ön dağıtımını göstermektedir.

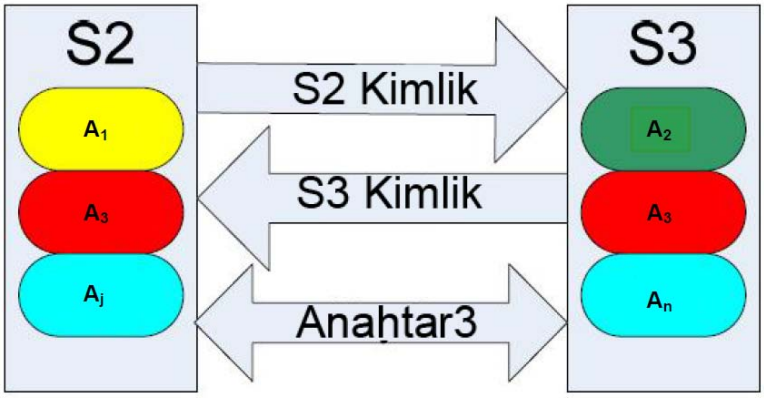


Şekil 3.2. Düğümler arasında anahtar dağıtımı ve depolanması (S: algılayıcı düğümü).

b) Paylaşılan anahtar keşfi aşaması

Paylaşılan anahtar keşfi aşaması, algılayıcı düğümler dağıtıldıktan sonra meydana gelir. Bu aşamada, her düğüm kablosuz haberleşme mesafesinde anahtar paylaştığı kendi komşusunu keşfeder. Sadece bir anahtar paylaştıklarında iki algılayıcı düğüm arasında bir bağlantı oluşur ve eğer iki düğüm arasında bir

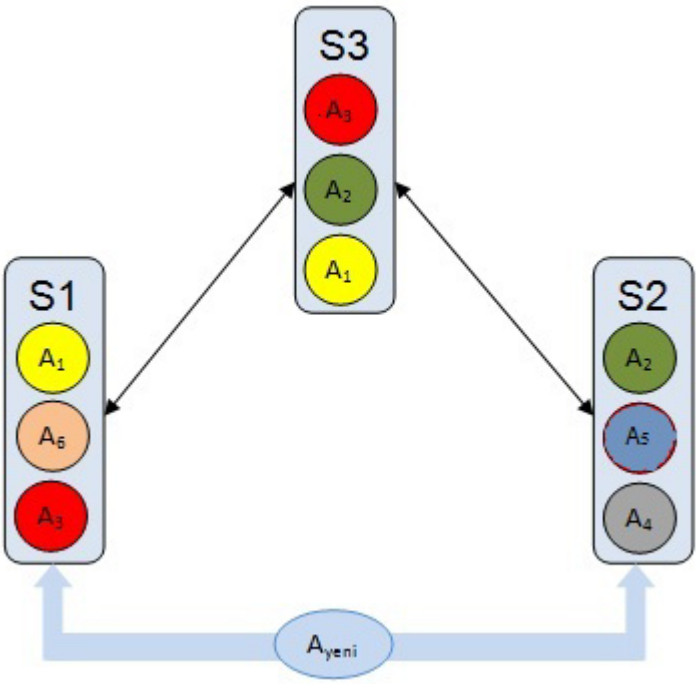
bağlantı oluşmuş ise, bütün haberleşme, şifre ve kimlik denetlemeleri yapılarak gerçekleşir [75]. Şekil 3.3. anahtra keşfi aşamasını göstermektedir.



Şekil 3.3. İki düğüm arasında ortak anahtar keşfi (S: algılayıcı düğümü).

c) Güvenilir yol anahtarı oluşturma aşaması

Birbirinin kablosuz haberleşme alanı içerisinde olan ve ortak anahtar bulma aşamasında ortak anahtar bulamamış olan iki düğüm ara düğümler yoluyla bir anahtar oluştururlar [75]. Şekil 3.4'te görüldüğü gibi iki düğüm ortak anahtar A_{yeni} kullanarak güvenilir yol anahtarı oluştururlar.



Şekil 3.4. İki düğüm arasında güvenli kanal iletişimi (S: algılayıcı düğümü).

3.5.2. Temel olasılıksal yöntem

Önceki yöntemlerde hala her bir algılayıcı düğüme büyük sayılarda anahtar yüklenmesini gerekmektedir. Büyük ölçekli KAA'lar için yukarıdaki şartların tümünü karşılayan temel olasılıksal yöntem önerilmiş. Bu yöntemde her algılayıcı kullanımdan önce büyük bir $|S|$ anahtar havuzundan bir K anahtar alt kümesi depolar (Başka bir ifadeyle $|S|$ havuzundaki anahtarların sayısı S kümesi içindeki K büyüklüğündeki iki rastgele alt kümenin P olasılığıyla en az bir anahtar paylaşacağı şekilde seçilir) [76]. Anahtar dağıtımı üç aşamadan geçer: *anahtar ön dağıtım aşamasında* algılayıcı ağı içindeki her bir düğüm halkası çok sayıda anahtara sahip olmak yerine, birkaç anahtar her bir düğümün belleğinde depolanması gerekir ve bu az sayıdaki anahtar seçilen olasılığa dayanarak iki düğümün ortak bir anahtar paylaşmasını temin etmek için yeterlidir. *Paylaşılan anahtar keşfi aşamasında* her düğüm telsiz iletişim aralığında anahtarları paylaştığı komşularını keşfeder. İki düğümün anahtar paylaşıp paylaşmadıklarını belirlemek için birçok yol vardır, en

basit yöntem de, düğüm kendi anahtar kimlik listesini diğer düğümlere yayınlar. Anahtar kimlik listesini karşılaştırılarak komşu düğümler ortak anahtara sahip olup olmadıklarını anlarlar. *Yol anahtarı oluşturma aşaması,* yol anahtarı oluşturmaya ve sonrasında telsiz iletişim aralığında ortak bir anahtarı paylaşmayan ama paylaşılan anahtar keşif fazının sonunda iki veya daha fazla linkle bağlanan seçilmiş iki düğüm arasındaki bağlantıyı oluşturmaya yarar, böylece, iki düğümün iletişimi için bir aracıdır veya anahtar dağıtım merkezi rolü gibidir [76].

P olasılığını hesaplamak için bu temel olasılıksal yöntem rastgele graf teorisini kullanır. Rastgele graf teorisi, $G(n,p)$, n tepe noktasından (vertics) oluşan bir graftır. Graf teorisinde iki düğüm arasındaki bağlantı olasılığı p'dir. P_c graf bağlanabilirliği şöyle hesaplanır [76]:

$$P_c = \lim_{n\to\infty} P_c[G(n,p)\ is\ connected] = e^{e^{-c}}$$

Burada, $p = \frac{ln(n)}{n} + \frac{c}{n}$,c sabittir.

n bellirlenmiş ise, $d = p * (n-1)$ ile d ve beklenen düğüm derecesi hesaplanabilinir. Sonuçta oluşan graf gereken P_c olasılığıyla bağlanır. Ayrıca bağlanabilirlik komşu sayısını ($n' \ll n$)'ye sınırlandırır, dolayısıyla iki komşu arasındaki anahtar paylaşım olasılığı:

$$p' = \frac{d}{n'-1} \gg p$$

Temel olasılıksal yöntemde işlenen P_c bağlanabilirlik olasılığı c seçilir ki $P_c = e^{e^{-c}}$ p, $p = \frac{\ln(n)}{n} + \frac{c}{b}$ ile hesaplanır.

Anahtar havuz boyutu S ve anahtar halkası boyutu k olmak üzere $1-\frac{((S-k)!)^2}{(S-2k)!S!} \geq p$ şartını sağlamalıdır.

Böylece, *S* boyutlu anahtar havuzu tanımlanır, ve her düğüm anahtar havuzundan k anahtarı rastgele alabilir ve komşuları ile P olasılığı ile en az bir anahtar paylaşabilir.

3.5.3. Q-bileşim rastgele anahtar ön dağıtım yöntemi

Bu yöntem temel olasılıksal yöntemin ağ güvenliğini ve direncini artırmak için genişletilmiş halidir. İki düğüm arasında güvenli iletişim kurabilmesi için sadece bir anahtar yerine (q>1) ortak anahtar gerektirir. Q-bileşim yöntemi büyük ölçekli saldırılarda savunmasız olurken, küçük ölçekli saldırılara karşı güvenliliği sağlayabilir. Q-bileşim rastgele anahtar ön dağıtım yönteminde güvenli bir iletişim bağlantısı sağlamak ve düğüm ele geçirilmesine dirençli olmak için iki düğümün en az q sayıda ortak anahtarı olmalıdır [68]. Çünkü iki düğüm arasında örtüşen anahtar miktarı çok büyük olduğunda ağın bunların iletişim bağlantılarını kırması zorlaşır. Aynı zamanda anahtar havuzu boyutunu azaltırsak bu da şebeke güvenliğinin ihlali tehlikesini getirir çünkü düşmanın anahtar havuzunu büyük bir bölümüne erişim elde edebilmek için birkaç düğümü ele geçirmesi yeter. Q-bileşim yönteminin büyük zorluğu q ortak anahtarları için optimal sayının seçilmesi ve bunu yaparken güvenliğin feda edilmemesidir. ***Anahtar ön dağıtım aşamasında,*** $|S|$ Anahtar havuzundan rastgele k anahtar seçilir ve her anahtar halkasında (saklanılır) ilk kullanıma hazırlatırılır. ***Paylaşılan anahtar keşif aşamasında,*** her düğüm komşularının her biriyle ortak olarak sahip olduğu tüm anahtarları keşfetmelidir. Bu, düğümün sahip olduğu tüm anahtar tanımlayıcıların ID (kimliklerin) yerel olarak yayınlanmasıyla yağılabilir. Daha güvenli, ama yavaş bir anahtar keşifi Merkle bulmaca (Merkle's Puzzle) yöntemi gibi **"**müşteri bulmaca**"** (Client Puzzle) yöntemi kullanılmasıdır [77].

S öyle seçilmelidir ki, en azından q anahtar paylaşan iki düğüm olasılığı p'den büyük veya p'ye eşit olmalıdır. S'yi aşağıdaki gibi hesaplayabiliriz [77]:

$$p(i) = \frac{\binom{|S|}{i}\binom{|S|-i}{2(m-i)}\binom{2(m-i)}{m-i}}{(m^{|S|})^2}$$

Burada, $p(i)$ her iki düğüm arasındaki i ortak anahtar olma olasılığıdır. m, düğüm halkasının kapasitesidir. i'yi seçmek için $\binom{|S|}{i}$ yol vardır. $|S|-i$, i'yi seçtikten sonra

havuzda geri kalan anahtar sayısıdır. m'yi seçmek için $\binom{|S|}{m}$ farklı yol bulunmaktadır. $\binom{|S|}{m}^2$ her iki düğüm için m'yi seçmek toplam yol sayısıdır. Her düğüm için geri kalan anahtarları seçmek için anahtar havuzundan $2(m-i)$ farklı anahtar seçilir. Bunun da $\binom{|S|-i}{2(m-i)}$ kadar yolu vardır. Her iki düğüm arasında eşit miktarda anahtarları bölmek için $2(m-i)$ yol vardır. P_c, iki düğüm arasında güvenilir iletişim kurmak için ortak düğüm sayısı olasılığıdır varsayalım. Dolayısı ile $P_c = 1 - (p(0) + p(1) + \cdots + p(q-1))$ ki $P_c \geq p$ seçilmelidir.

Bu yöntemin avantajları, mükemmel eseneklik ve kimlik doğrulamasına sahip olmasıdır. Başka bir değişle, bir düğüm ele geçirilse bile diğer düğümlerdeki anahtarların gizlilikleri ihlal edilemez. Bu yöntemin dezavantajları, her düğüm ağdaki düğüm sayısı kadar çok anahtar depolanmasını gerektirdiği için bu yöntem ölçeklenebilirliği çok iyi şekilde destekleyemez. Bunlara ek yeni düğüm ağa bağlanması için giğer düğümler yeni anahtarı depolamak zorunda kaldığı için yeni bir düğüm eklemek zor olur.

3.5.4. Çoklu yol anahtar güçlendirme yöntemi

Bu metot, bazı ağ iletişimlerinin ek yüklerinin feda etmek yoluyla düğüm yakalama saldırılarına karşı büyük ölçüde artmış bir dirençlilik ve esneme elde etmek için temel rastgele yöntem ile bağlantılı olarak uygulanabilir [77].

Temel yöntemde, anahtarların anahtar havuzundan rastgele seçilmesi bir ağ içindeki düğümlere aynı anahtarlardan bazılarını paylaşma olanağı verir ve bu yolla da olası olarak sadece birinin güvenliği tehlikeye düşürüldüğünde çoklu düğümleri tehdit eder.

Bu problemi çözmek için düğümler arasındaki haberleşme anahtarı güvenli bir bağlantı oluştuktan sonra bir tanesinin güvenliği tehlikeye düşürüldüğü zaman

mutlaka güncellenmelidir [77]. Bu, önceden tesis edilmiş bağlantı vasıtası ile yapılmamalıdır, çünkü, düşman yeni anahtarı elde etmek için bağlantıyı çözebilir. Daha fazla güvenlik için bağımsız çoklu yollar kullanılarak koordine edilmelidir [77].

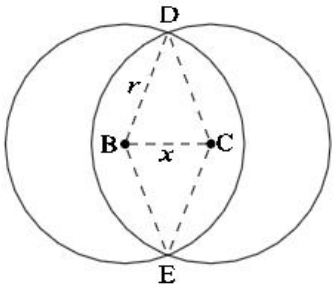
Artık çeşitli düğümlerin anahtar halkalarındaki ortak anahtarlar sayesinde oluşturulmuş birçok güvenli link mevcuttur. A'nın, anahtar kurulumundan sonra B'ye ait güvenli bir link olduğunu kabul edelim. İlk anahtar kurulumunun temel rastgele anahtar yöntemini kullanarak tamamlanmış olduğunu var sayalım. Bu düğümlerden herhangi biri yakalanırsa A ve B arasındaki bağlantının güvenliği tehlikeye girer. Bu sorunu ele almak için anahtar güncellemesini bağımsız çoklu yollar üzerinde koordine etmemiz zorunludur [77].

A düğümünün B düğümü ile güncellenmiş bir iletişim anahtarına ihtiyaç duyması durumunda B düğümüne ait tüm olası ayrık yollarının kullanılması zorunludur. Bu yöntemde A ve B düğümü arasında yeterli miktar da yönlendirme bilgileri olmalıdır. A düğümünden B düğümüne giden h kadar bu ayrık yol olduğunu kabul edelim. Buna göre A düğümü h kadar rastgele $(g_1, g_2, .. , g_h)$ değer üretir, bunların her biri, bir şifreleme anahtarının büyüklüğüne eşittir.A düğümü, rastgele değerleri ayrık yol boyunca B düğümüne gönderir. B düğümü h rasgtele değer almış olduğunda, A düğümü tarafından, k asıl anahtar olmak üzere $k' = k \oplus g_1 \oplus g_2 \oplus \ldots \oplus g_h$ ile yeni ve güvenli bir iletişim linki oluşturulması ile aynı zamanda yeni bir şifreleme anahtarı hesaplar [69,77].

Yeni bağlantının uygulamaya konulması ile, herhangi bir düşman tarafından iletişimlerin şifresinin çözülebilmesinin tek yolu anahtarın biçimlendirilmesinde rol oynayan tüm düğümlerin güvenliğini tehlikeye düşürmektir. h ne kadar büyük olursa, olaya daha çok yol ve düğüm dahil olur ve yeni bağlantının güvenliği de o kadar artar [77]. Ağ iletişimlerindeki bu artış iki düğüm arasında ayrık çoklu yolların bulunmasında aşırı ek yüklere neden olur. Ayrıca, bir yolun büyüklüğü artarken bir

düşmana gizlice dinleme yapma şansı verdiği sürece büyüyebilir de ve bu da yolun tamamını güvensiz kılar.

İki sekmeli çoklu yol anahtar güçlendirme



Şekil 3.5. İki düğüm arasındaki uzaklık ve iletişim aralığı

Şekil 3.5. hesaplamamızda kullanılacak olan parametreleri göstermektedir. B ve C iletişimde bulunan iki algılayıcı düğümü simgelemektedir. *r,* her bir algılayıcı düğümünün iletişim aralığıdır. Her bir düğümün alma ve iletme için aynı aralığa sahip olduğunu kabul ediyoruz. *x*, iki düğüm arasındaki uzaklıktır.

Chan, Perrig ve Song, bir ağ içindeki iki düğüm arasındaki iletişim yarıçapının bindirmesini (örtüşmesini) düzlemsel bir dağıtımda $0.5865\pi r^2$ olarak hesaplamıştır [77].

Her iki düğümünde de güvenli bir bağlantıyı paylaştığı ortak komşular olarak güvenli bir linki paylaşan iki düğümün *güçlendirici komşuları* terimini tanımlarız. Tek bir iletişim yarıçapının umulan bindirme (örtüşme) alanı 0.5865 olduğundan, güçlendiren komşuların umulan sayısı da bundan dolayı $0.5865p^2n'$ olur [66]. Burada p iletişimde bulunacak yeterli anahtarların paylaşılması olasılığı ve n' de her bir düğümün komşu sayısıdır. Bu, aynı zamanda $0.5865d^2/n = k$ olarak da ifade edilebilir, her iki düğüm de umulan k sayıda komşu ile güvenli bir link paylaşımı yapar. Örnek olarak, $d = 20$ ve $n' = 60$ için güçlendiren komşu sayısı $k = 3.83$'tür.

İki sekmeli yaklaşımın etkililiğini değerlendirmek için iki düğüm arasındaki linkin uzlaştırılması için yeni bir olasılığın türetilmesi gereklidir.

Genel olarak, eğer bir link *k* sayıda ortak komşu tarafından güçlendirilirse, bu durumda hasmın bu bağlantı üzerinde ve aynı zamanda da *k* sayıdaki 2 sekmeli yolların her biri üzerindeki en az bir bağlantıda gizli dinleme yapabilir olması zorunludur.

Hasmın bir bağlantıyı tehlikeye düşürebilmesi baz olasılığı *b* ise, buna göre verilen herhangi bir iki sekmeli yol üzerinde en az bir sekmede tehdit yaratabilmesi olasılığı yol içindeki 1 sekmesini tehlikeye düşürmesi olasılığı artı yol içindeki 2 sekmesini tehlikeye düşürmesi olasılığı eksi yol içindeki her iki sekmeyi de tehlikeye düşürmesi olasılıdır ve bu da $2b - b^2$ 'e eşittir. Böylece, bir linkin kesilmesinin son olasılığı : $b' = b(2b - b^2)^k$ 'dır [77].

Örneğin; güçlendirmeden önce, verilen herhangi bir link üzerinde b=0.1'lik bir gizlice dinleme baz şansına sahipse, 3 komşu (k komşu sayısı) tarafından güçlendirilen bir link için güçlendirmeden sonraki gizlice dinleme şansı 6.86×10^{-4} ya da yaklaşık 1,458'de 1 olarak artar.

Chan, Perrig ve Song, umulan güçlendirici komşu sayısından, 2 sekmeli çoklu yol güçlendirme yönteminin umulan iletişim ek yükünü hesaplamaktadır. Güçlendiren her bir komşu, verili bir 1 sekmeli bağlantının güçlendirilmesine yardımcı olmak için ilaveten 2 sekmeli bir iletişimi temsil etmektedir. Böylece, ortalama olarak, anahtar güçlendirmede toplam ek iletişimlerin ek yükü, ortak komşu keşfi için ek iletişimler dahil olmamak kaydıyla, ağın temel anahtar kurulumu için gerek duyduğu ek yükün en az $2\times0.5865p^2n'$ katıdır [74]. Örneğin; p = 0.33 ve n' = 60 için, anahtar kurulumu tamamlandıktan sonra en az 7.66 kat ek ağ trafiği görmeyi umabiliriz. Ortak komşu keşfi de dahil olmak üzere, son yötemin bu durumda ağ iletişimlerinde temel

yöntemden yaklaşık olarak 10 kat daha pahalı olacağını tahmin ederiz [64]. Gizli dinleme olasılıklarının 0.1'den 6.86×10^{-4}'e yükseltilebileceğini (146 kat daha yüksek) göz önünde bulundursalar bu iyi bir alış veriş olabilir [77].

Çoklu yol anahtar güçlendirme ve iki Sekmeli Çoklu yol yöntemleri düğümden-düğüme doğrulama sağlamaz. Daha önceden de belirttiğimiz gibi, ikili anahtar dağıtım yöntemi en etkili anahtarlama yöntemidir. Çünkü diğer yöntemlerle karşılaştırıldığında, düğümden-düğüme doğrulama ve düğüm kopyalamasına tolerans dâhil olmak üzere, çok sayıda ilave özellik sunmaktadır. Bunun yanı sıra, algılama düğümlerinin kısıtlı kaynakları olduğu için, bu önemli ek yük yöntemin uygulanabilirliğini sınırlar. Fakat daha küçük ağlar için etkili bir şekilde kullanılabilir [77].

3.5.5. Rastgele ikili anahtar yöntemi

Rastgele ikili anahtar yöntemi tüm $n-1$ anahtarlarının yüksek olasılıklı bağlanmış rastgele bir grafiğe sahip olmak için düğümün anahtar halkası içinde depolanmasına ihtiyaç duymadıkları yerlerde ikili anahtar yönteminin bir modifikasyonudur. *n*: ağ içindeki düğümlerin sayısı ve p de iki düğümün güvenli bir biçimde iletişimde bulunabilme olasılığı olmak üzere her bir düğümün tüm $n-1$'nın hepsini birden depolamak yerine sadece *np* ikili anahtardan oluşan rastgele bir kümeyi depolaması gerekecektir. Bir düğüm *k* kadar anahtarı depolayabilirse, buna göre desteklenebilir maksimum ağ büyüklüğü $n=\frac{k}{p}$'dir [76]. Temel olasılıksal yöntemde görüldüğü gibi, düğüm bağlantıları bir takım arzu edilen P_c olasılığı karşıladığı takdirde tüm düğümlerin bağlantı kurmalarına gerek duyulmaz, dolayısı ile , anahtar halkasında sadece *np* anahtar depolanmasını gerektirir.

Rastgele ikili anahtarlar yönteminde başlatma ve anahtar kurulumu [77]

a. Dağıtım öncesi *başlatma* aşamasında, toplam olarak $n = k/p$ sayıda benzersiz düğüm kimlikleri üretilir. Her bir düğüm kimliği m kadar diğer rastgele seçilmiş ayrık düğüm tanıtıcı ile eşleştirilir ve her bir düğüm eşi için bir ikili anahtar üretilir. Ağın gerçek büyüklüğü n'den daha küçük olabilir. Kullanılmayan ek tanıtıcılar, ileride ağ'a ek düğümler ilave edilirse kullanılacaktır ve bu da ağa ölçeklenebilirlik özelliği verir. Anahtar, aynı zamanda bu anahtarı da bilen diğer düğümün tanıtıcısı ile birlikte her ikin düğümün anahtar halkaları içinde depolanır.

b. Dağıtım sonrası *anahtar kurumu* aşamasında her bir düğüm önce hemen yakınındaki komşulara kendi kimliğini yayınlar, eğer komşu düğümler yayın düğümü ile iletişim için ortak ikili anahtarı paylaşır ise komşu düğümler arasında bir kriptografik el sıkışma gerçekleşir ve böylece de güvenli bir iletişim linki oluşur.

c. Bu yöntemi destekleyen Ek özellik Çoklu sekmeli aralık uzantımıdır. Bu yayınlama işlemi aynı zamanda, ara düğümlerin düğüm kimliğini belli sayıda sekmeye yeniden yayınlaması sağlanarak bir düğümün aralığını aşacak biçimde de uzatılabilir. Düğüm kimliğinin etkili bir biçimde yeniden yayınlandığı her bir düğüm iletişim yarıçapının aralığını uzatır ve böylece de yayını duyabilecek düğümlerin sayısı artmış olur.

Çizelge 3.1.'de iletişim aralığı içindeki umulan komşu sayısının 60 olduğu durumdaki ulaşılabilir düğümlerin sayısı için bazı seziler gösterilmiştir [77].

Çizelge 3.1. Beklenen komşu sayısı [77]

yerel (0 sekme)	1 sekme	2 sekme	3 sekme
60	240	540	960

Etkili iletişim yarıçapını arttırarak, aynı zamanda *n'* komşu sayısını da arttırırız ki bu da ağın maksimum izin verilebilir büyüklüğünün arttırılmasında yardımcı olur [77].

3.5.6. Çokterimli havuz tabanlı anahtar ön dağıtımı

Bu yöntemde anahtarlar her algılayıcı düğümünde çokterimli alt küme havuzundan üretilir. Eğer iki düğüm aynı çokterimliden üretilen anahtarları sahipseler, çokterimli-tabanlı anahtar ön-dağıtım yöntemine dayalı ikili anahtar kurabilirler.

Çokterimli tavuz tabanlı yöntemin başlangıç aşaması, çokterimli tabanlı anahtar ön dağıtımına dayanır ki ikili anhtarlar dağıtır. Sistem sunucusu bir F_q alanı üzerinde iki değişkenli rastgele *t-dereceli* bir çokterimli üretir: $f(x, y) = \sum_{i,j}^{t} aij\ x^i\ y^j$ oysa q kriptografik bir anahtar dağıtmaya yetecek kadar geniş bir asal sayıdır ve *f(x, y) = f(y, x)* özelliğine sahibtir [75].

Her algılayıcının benzersiz bir kimliğe sahip olduğu kabul edilmektedir. Her bir *i* algılayıcısı için sistem sunucusu, *f(x, y)*'nin yani *f(i, y)*'nin *çokterimli payını* hesaplar. *i* ve *j* gibi herhangi iki algılayıcı düğümü için düğüm *i*, *j* noktasında *f(i, y)*'yi değerlendirerek *f(i, j)* genel anahtarını ve düğüm *j* ise *i* noktasında *f(i, j)*'yi değerlendirerek aynı *f(j, i) = f(i, j)* anahtarını hesaplayabilir [75].

Bu yaklaşımda her bir *i* algılayıcı düğümü , *(t+1) log q* depolama alanını işgal eden *f(i, x) t-dereceli* çokterimlini saklaması gerekmektedir. İkili anahtar oluşturmak için her iki algılayıcı düğümünün de diğer algılayıcı düğümü kimliğindeki çokterimlini değerlendirmesi gerekmektedir.

Bu nedenle bu yöntem, *t* değerinin algılayıcı düğümlerinda bulunan bellekle sınırlandırıldığı durumda yalnızca *t* kadar ele geçirilmiş düğümlerden başkasını tolare edemez. Aslında algılayıcı ağı ne kadar büyük olursa, düşman da *t* algılayıcı

düğümlerinden ve sonra da bütün ağın gizliliğini ihlal eder, bu nedenle güvenli ve pratik anahtar kurulumu tekniklerine sahip olmak için "*çokterimli havuz tabanlı anahtar ön dağıtımı*" adı verilen bir genel program daha geliştirilir, tek bir *t-dereceli* çokterimli kullanımı yerine yeni programda bir çokterimliler havuzu kullanılmaktadır. Başlangıç aşamasında her bir düğümün belleğine rastgele seçilen çokterimliler yerleştirilir. Havuzda sadece bir çokterimli kaldığında program, çokterimli havuz tabanlı anahtar dağıtımının arkasında kalır, fakat bütün çokterimliler *0-derece* olursa dağıtım, çokterimli tabanlı anahtar ön dağıtım programına benzer.

İkili anahtar oluşumu üç aşamada gerçekleştirilir [75]: *anahtar ön dağıtımı, doğrudan anahtar oluşumu* ve *yol anahtarı oluşumu*.

Anahtar ön dağıtım aşamasındaki ana sorun, her bir algılayıcı düğümü için çokterimlilerin alt kümesinin F_q'den nasıl toplanacağıdır.
anahtar keşif aşamasındaki ana sorun,olduğu genel iki değişkenli Çokterimlinin nasıl bulunacak.

Bu sorunu çözmek için iki tür teknik vardır: *ön dağıtım* ve *gerçek zamanlı keşif*. **Ön dağıtım tekniği:** Bu, her bir düğüm çokterimlini paylaşan düğümlerin kimliklerini taşır. Sistem sunucusu, belli bilgileri algılayıcılara önceden dağıtır, böylece başka bir algılayıcının kimliği verilen bir algılayıcı düğümü, diğer algılayıcıyla bir çift anahtar oluşturup oluşturamayacağını belirleyebilir [75].

Ancak bu yöntemin, ağa katılan, ağa yeni düğümleri ekleme esnekliğini sunmayan ve ağı saldırılara karşı açık bırakan algılayıcılarla ilgili zorlukları vardır. Sonuç olarak saldırgan, özel bir iki değişkenli çokterimlinin çokterimli paylarını öğrenme girişiminde bulunabilir ve depolanan verilere erişim elde edebilir.

Gerçek zamanlı keşif tekniği: Gerçek zamanlı keşif, iki algılayıcının anında keşfedilmesini gerektirir. Bu keşif, öncelikle her ikisinin de paylara sahip olduğu çokterimli kimliklerini değiştirebilen ve düğümlerin paylaştığı, sonra da genel çokterimlini tanımlamaya çalışan ve düğümler doğrudan çokterimli kimliklerini açıklamak. Gerçek zamanlı keşfin geri çekilmesi, ön dağıtım yaklaşımlarında görünmeyen, yöntem seçimini kritik hale getiren ek iletişim ek yükünü göstermesidir [75].

Yol Anahtarı Oluşturma aşamasındaki ana sorun, iki algılayıcı düğümü arasında bir yolun nasıl bulunacağını belirten *yol keşfi* sorunudur ve düğümler, her iki düğümle da iletişim kurabilmelidir. bu soruna yönelik olarak iki tür teknik bulunmaktadır [75]:

a) **Ön dağıtım:** sistem sunucusu belli bilgileri algılayıcılara önceden dağıtır, böylece başka bir algılayıcının kimliği verilen bir algılayıcı düğümü, diğer algılayıcıya doğrudan bir anahtar yolu bulabilir. Bu tekniğin dezavantajı, saldırının ölçülebilirliğinin ve hassasiyetinin bulunmamasıdır.

b) **Gerçek zamanlı keşif**: Gerçek zamanlı keşif teknikleri, anında algılayıcı keşfi anahtar yoluna sahiptir. Algılayıcı, oluşturulan çift anahtarlara sahip bir dizi ara düğümü toplar. Kaynak düğüm, bütün bu ara düğümlere istek gönderebilir. Bu ara düğümlerden biri doğrudan hedef düğümle bir çift anahtar oluşturabilirse anahtar yolu keşfedilir. Bu süreç, istek gönderen ara düğümlerle devam edebilir. Geri çekme, bu yöntemlerin temel iletişim ek yükünü gösterebilmesidir.

Bu yöntem, Dağıtımdan sonra ağın daha geniş boyuta büyümesine izin vermek avantajını içerir.Ve bu yöntemin sakıncası t-çarpışma direnci (t-cokterimliden fazla uzlaşmak ağın uzlaşmasına neden olur) içermektedir.

Rastgele altküme kullanarak çok terimli havuz tabanlı anahtar ön dağıtım yöntemi diğer yöntemlerle karşılaştırıldığı zaman belli bir sayıya kadar (%60 ele geçirilmiş düğümler) mükemmel güvenlik ve esneklik sunar. Düğüm yakalanmasına karşı ağdaki hiçbir anahtar iki defa kullanılmadığından mükemmel direnç sunan rastgele ikili anahtar yöntemle karşılaştırıldığında, çokterimli havuz tabanlı eğer bir çokterimli t defadan fazla kullanılmamışsa aynı direnci sunar.

3.5.7. Rastgele alt küme anahtar ön dağıtımı

Bu yöntem, temel olasılıklı yöntemin bir uzantısı sayılabilir. Büyük bir anahtar havuzundan rastgele anahtar seçip bunları algılayıcılara tayin etmek yerine bu yöntem, çokterimli havuzundan çokterimliler seçip bu çokterimlilerin paylarını algılayıcılara tayin eder [75].

Temel olasılıklı yöntemde aynı anahtar, birden fazla algılayıcı tarafından paylaşılabilir. Bunun tersine bu yöntemde her bir algılayıcı çifti arasında benzersiz bir anahtar bulunmaktadır.

Anahtar oluşumu için üç aşama vardır [75]:

ön dağıtım aşaması (**Alt küme tayini)**: Sistem sunucusu, *s* iki değişkenli *t-dereceli* çokterimlilerin $\mathcal{F}$ kümesini rastgele oluşturur. Her bir algılayıcı düğümü için sistem sunucusu, $\mathcal{F}$'den s' çokterimlilerinin alt kümesini rastgele toplar ve bu s' çokterimlilerinın paylarını algılayıcı düğümüne tayin eder.

Anahtar keşfi aşaması (**Çokterimli payı keşfi):** Ortak bir anahtarı paylaşarak düğümleri oluşturması gereken algılayıcılar, gerçek zamanlı keşif teknikleriyle ortak bir Çokterimli bulmak zorundadırlar. İki değişkenli ortak bir Çokterimli keşfetmek için algılayıcı düğümü bir Çokterimli kimlikleri listesi yayınlayabilir.

Yol anahtarı oluşturma aşaması (**Yol keşfi)**: İki algılayıcı ortak bir anahtarı paylaşamazsa yol anahtarı oluşturma aşamasını başlatmak zorundadırlar. Bu aşama sırasında bir kaynak algılayıcısı düğümü, hedef düğümle bir ortak anahtar oluşturmaya yardımcı olabilecek başka bir düğüm bulmaya çalışır. Kaynak düğümü, bir istek mesajı yayınlar. Bu isteği alan düğümlerden biri, hem kaynak düğümü ile hem de hedef düğümle ortak bir anahtar oluşturabilir, sonra ikisinin arasında bir iletişim yolu oluşur.

İki algılayıcının doğrudan ortak anahtarı paylaşabilme olasılığı olan aynı iki değişkenli çokterimlini paylaşma olasılığı şu şekilde tahmin edilebilir $p = 1 - \prod_{i=0}^{s'-1} \frac{s-s'-i}{s-i}$ [75].

d ile, her bir algılayıcı düğümünün temas edebileceği ortalama komşu düğüm sayısını göstermiş olsak. Hem kaynak hem de hedef düğümle bir çift anahtarı paylaşma olasılığı p^2'dir. Algılayıcı düğümlerin anahtar keşfinde veya yol anahtarı oluşturma aşamasında bir çift anahtar oluşturma olasılığı şöyle olacaktır $P_s = 1 - (1-p)(1-p^2)^d$ [75].

Çokterimlinin uyumlu olma olasılığı ise şöyledir: $P_c = 1 - \sum_{i=0}^{t} p(i)$.

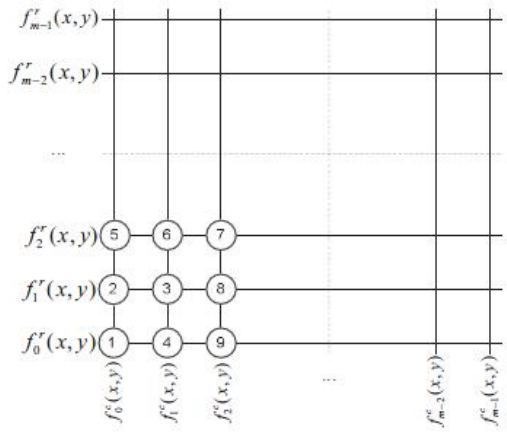
Saldırgan algılayıcı düğümleri üzerinde çokterimlilerin dağıtımını biliyorsa özel bir çokterimliden türetilen anahtarları uydurmak için belli algılayıcıları hedefleyebilir. Yukarıdaki tehdidi yok etmek için kolay bir yöntem, her bir çokterimlinin kullanımını en fazla *t+1* defa sınırlandırmaktır. [75] Bu yöntem, rastgele alt kümeler uygulandığında maksimum algılayıcı sayısına da limit uygular.

3.5.8. Şebeke bazlı anahtar ön dağıtımı

Bu yöntem bir dizi çekici özelliğe sahiptir. Bu yöntem, ilk önce ele geçirilmiş herhangi bir algılayıcı olmadığında ve algılayıcıların birbirleri ile haberleşmede

bulunabilmesi şartıyla herhangi iki algılayıcının bir ikili anahtar tesis edebilmesini garanti eder [74]. İkinci olarak bu yöntem düğümün ele geçirilmesi durumuna karşı dayanıklı ve bunu atlatılıp eski duruma gelmesi konusunda esnektir. Bazı algılayıcıların ele geçirilmesi durumunda bile, hala ele geçirilmemiş algılayıcılar arasında ikili bir anahtar tesis etme gibi yüksek bir olasılık vardır. Üçüncü olarak, bir algılayıcı bir başka düğüm ile ikili bir anahtar belirleyip belirleyemeyeceğini doğrudan doğruya belirleyebilir ve eğer bunu yapabilirse hangi çokterimlinin kullanılmasının gerektiğini de belirleyebilir. Bunun bir sonucu olarak, çokterimli paylaşımı keşfi sırasında hiçbir haberleşme ek yükü yoktur [74].

Şekil 3.6'da görülmekte olduğu üzere, şebeke içindeki her *i* satırı bir $f_i^r(x,y)$ ile birleştirilmiştir ve her bir sütun ise bir $f_i^c(x,y)$ çokterimli ile birleştirilmiştir.



Şekil 3.6. Şebeke bazlı anahtar ön dağıtımı [74].

Anahtar belirlemenin üç aşaması vardır [74], anahtar belirlemenin ilk aşaması **alt küme atama**'dır. Kurulum sunucusu çokterimliler $\{f_i^c(x,y,f_i^r(x,y)\}, i=0,\dots,m-1$ ve $m=\sqrt{N}$ olmak üzere N sayısı algılayıcı düğümüne sahip bir algılayıcı ağ için ve 2m'lik bir kümeye sahip bir $m \times m$ şebeke yapılandırır. Her bir algılayıcı için kurulum sunucusu şebeke içinde işgal edilmemiş ve benzersis bir *(i, j)* ara kesiti seçer ve bunu düğüme atar. Bundan dolayı algılayıcının kimliği ID =$\langle i,j \rangle$ 'dir.

Sunucu her düğüme kendi kimliğini sağlar ve satır ve sütun Çokterimlini bu şebeke ara kesitini paylaşır.

Bundan dolayı, kurulum sunucusu bu algılayıcı düğümüne$\{1D, f_i^c(x,y), f_i^r(x,y)\}$ dağıtımı yapar. Yol keşfini kolaylaştırmak için algılayıcılara tahsis edilen ara kesitlerin şebeke içindeki bir dikdörtgen alan içerisinde yoğun bir biçimde seçilmiş olmaları gereklidir Bunun bir sonucu olarak, algılayıcı düğümleri bu bilgiye dayalı olmak üzere paylaşım keşfi ve yol keşfini gerçekleştirebilirler. İkinci aşamada (**çokterimli paylaşım keşfi)**: *i* düğümü *j* düğümü ile bir ikili anahtar belirleyecektir, *i* düğümü *j* ile ortak satır ya da sütunlar olup olmadığını yani $c_i = c_j$ veya r_i=r_j olup olmadığını kontrol eder. Eğer $c_i = c_j$ ise buna göre hem *i* hem de *j* düğümleri satır ya da sütun çokterimli paylaşımlara sahiptir ve doğrudan doğruya ikili bir anahtar belirleyebilirler. Eğer bu koşullardan hiçbiri doğru değilse, *i* ve *j* düğümleri yeniden bir ikili anahtar belirlemek üzere yol keşfinden geçerler. **yol keşfi aşaması:** *S* düğümü ilk önce, üzerinden, *D* düğümü ile bir ikili anahtar belirleyebileceği bir ara düğüm bulur. Eğer güvenliği tehlikeye düşürülmüş herhangi bir düğüm yoksa herhangi iki algılayıcı arasında bir ara düğüm olarak kullanılabilecek en az bir düğümün var olması garanti edilir.

İlgili düğümleri ele geçirmeden sadece ikili anahtarı ele geçirmek için, saldırgan, iki düğüm arasındaki paylaşılan çokterimlini ele geçirebilir. Bu da en az $(t+1)$ düğümü ele geçirmek gerektirir. Saldırgan çokterimlini ele geçirse bile bağlı düğümler yine de bir ikili anahtar kurabilirler [74].

İki değişkenli çokterimlini ele geçirmek olasılığı: $P_c = 1 - \sum_i^t p(i)$'dir. Her bir düğüm 2 adet iki değişkenli t–dereceli çokterimli ve düğüm kimliği depolamak zorunda olmasından dolayı her bir düğümün ek yükü en fazla $2(t+1)\log(q) + 2(t+1)l'$ dir, burada *l* çokterimli havuzunda ifşa edilen bazı çokterimlilerdir. Düğümler arasında doğrudan doğruya bir anahtar belirleme mevcut olduğu zaman, buna göre ek yük 0'dır. İki düğümün yol anahtarı belirleme yolu ile ortak bir anahtar

bulmasının zorunlu olduğu durumda küçük bir yük vardır ve bu ek yük, güvenliği tehlikeye düşürülmüş her bir ek düğüm ile artar.

Bunlara ek, şebeke bazlı anahtar dağıtımı yöntemi, diğer yöntemlerde sağlanmayan bazı özelliklere sahiptir. Ağda gizliliği ihlal edilmiş düğümler bulunmazsa, doğrudan yoksa ara düğüm yardımıyle her hangi iki algılayıcı arasında anahtar kurmasına garanti eder. İletişim yükü büyük ölçüde önceki yöntemlerden daha alçaktır, ayrıca anahtar yolu belirlemek verimlidir. Ikinci olarak, ağda ele geçirilmiş düğümlerin bulunmasına rağmen hala ele geçirilmiş iki düğüm arasında ikili anahtar belirleme olasılığı yüksektir. Şebeke bazlı yöntem iki algılayıcı düğüm arasında mevcut ikili anahtar ele geçirilse bile bu düğümler ele geçirilmedikçe, yüksek bir ihtimalle tekrar başka bir ikili anahtar kurabilecekleri anlamında izinsiz girmelere karşı toleranslıdır. Bu yöntem düğüm ele geçirilmesine karşı iyi direnci vardır ve hem ağ haberleşmesinde ve hem de hesaplamalarda daha az ek yük sunar.

3.5.9. Hiperküp anahtar dağıtım yöntemi

Ağdaki toplam N algılayıcı düğümlerini dikkate alan hiperküp tabanlı yöntem, öncelikle m^{n-1} iki değişkenli çokterimli *n*-boyutlu bir hiperküp oluşturur. Sonra sistem sunucusu, ağdaki her düğüme bu *n*-boyutlu alanda benzersiz bir koordinat tayin eder. Belirli bir koordinatta bulunan algılayıcı düğümü için sistem sunucusu, bu düğüme çokterimli paylarını önceden dağıtır [75].

Sistem sunucusu, sonlu bir F_q alanı üzerinden rastgele $n \times m^{n-1}$ *t-dereceli* iki değişkenli çokterimliler üretir. Her algılayıcı düğümü için sistem sunucusu, *n*-boyutlu alanda işgal edilmemiş bir koordinat seçer ve bu koordinatı bu düğüme tayin eder. Bu koordinat daha sonra bu düğümün kimliği (ID) olarak kullanılır. Sonra sistem sunucusu, kimlik ile çokterimli paylarını bu algılayıcı düğümüne dağıtır [75].

j düğümüyle bir çift anahtar oluşturmak için *i* düğümü, n – 1 boyutlarında aynı alt indekslere sahip olup olmadıklarını kontrol eder. *i* ve *j* düğümleri, ortak bir çokterimlini paylaşırlar ve çokterimli tabanlı anahtar ön dağıtım yöntemini kullanarak doğrudan bir anahtar oluşturabilirler.

İki düğüm ortak bir çokterimlini paylaşıyorsa doğrudan bir bağlantı oluşturabilir ve iletişim için bir çift anahtar hesaplayabilirler. *i* ve *j* düğümleri doğrudan bir anahtar oluşturamazsa hiperküpte bir biri arasında bir anahtar yolu bulması gerekir. Ele geçirilmiş düğümler yoksa ve iki düğüm birbiriyle haberleşebiliyorsa iki düğüm arasında bir oturum oluşturmak için kullanılabilen en az bir anahtar yolunun bulunmasını garanti eder.

Dinamik Anahtar Yolu Keşfi

Aşağıdaki algoritma, dinamik olarak *S* ve *D* düğümleri arasında bir anahtar yolu bulur. Temel düşünce kaynak düğüme sahip olmaktır ve hedef düğüme "yakın" olan ele geçirilmemiş bir düğüm ile iletişime geçer. Başarı şansını artırmak için aşağıdaki algoritma birçok devrelerle gerçekleştirilebilir. Ağda ele geçirilmiş düğümler yoksa bu yöntem, iki düğüm iletişimde olduğu sürece daima çalışacaktır [64].

Başlangıçta *D* düğümüyle dolaylı bir anahtar oluşturmak için *S* düğümü rastgele *r* sayısını üretir ve c sayısını 0'la başlatır. Her devrede c'yi artırır ve $K_c = F(r, c)$'yi hesaplar. Sonra bayrak = 1 ile *M* mesajını oluşturur. Bu mesaj, kaynak ve hedef düğüm kimliklerini, K_c, *c* ve bayrak değişkeni olan "$M = \{S,D,K_c, c, bayrak\}$"yi içerir. Bayrak değişkeni, keşfedilen yolun uzunluğunu ve gönderilen mesaj sayısını kontrol etmek için kullanılır [64].

$M = \{S,D,Kc, c, bayrak\}$ mesajına sahip *u* algılayıcı düğümünü dikkate alan *u* düğümü, önce ele geçirilmemiş bir çokterimli kullanarak *u* ile doğrudan bir anahtar

oluşturabilen gizliliği ihlal edilmemiş bir *v* düğümü bulmaya çalışır. Bu başarılı olursa *u*, *M*'deki *bayrağı* 1'e ayarlar ve değiştirilmiş *M* mesajını *v*'ye gönderir.

u, böyle bir düğüm bulamazsa ve *M*'deki *bayrak* 0 ise yol keşfi durur. Aksi taktirde uyumsuz bir çokterimli kullanarak *u* ile doğrudan bir anahtar oluşturabilen ele geçirilmiş bir *v* düğümü seçer ve *u* bu tür bir *v* düğümü bulursa *M* mesajındaki bayrağı 0'a ayarlar ve değiştirilen *M* mesajını *v*'ye gönderir. Bu tür bir düğüm bulamazsa bu düğümdaki yol keşfi protokolü sona erer.

D hedef düğüm anahtar oluşturma talebini aldığında *S* düğümünün kendisiyle bir ikili anahtar oluşturmak istediğini bilir. *D* düğümü daha sonra $K_{S,D} = K_c$ olarak ikili anahtar oluşturur ve karşı *c* değeri konusunda *S* düğümünü bilgilendirir. Sonuç olarak her iki algılayıcı düğümü da aynı ikili anahtarı paylaşır [64].

Performansı, ek yükü ve güvenliği değerlendirirsek, ağ boyutu veya ağdaki düğüm sayısı artarsa iki düğümün doğrudan bir ikili anahtar oluşturabilme olasılığının azaldığını söyleyebiliriz. Bu yöntem, doğrudan veya dolaylı olarak iki düğüm arasında bir ikili anahtar oluşumunu garanti eder [64].

3.5.10. Dağıtım bilgisini kullanarak anahtar ön dağıtımı

Yukarıda bahsedilen bütün yöntemler, anahtar ön dağıtım sorununa uygun çözümler sunmasına rağmen performanslarını önemli ölçüde iyileştirebilecek bir bilgi parçasını ortaya çıkarmamıştır. Bu bilgi parçası, uygulamada düğümlerin yerleştirildiği yoldan türetilebilen *düğüm yerleşim bilgisidir*.

İlk olarak algılayıcılar, daha küçük gruplar dizisiyle önceden düzenlenir ve bu gruplar bir uçaktan atılır. Yerleşim bilgisi, tek biçimli olmayan olasılık yoğunluk fonksiyonlarını (tek biçimli olmayan pdf) kullanılarak modellenebilir [64].

Dağıtım bilgisinin modellenmesi

Bu, algılayıcı düğümlerinin yerleştirilir yerleştirilmez statik olduklarını farz eder. Dağıtım modelinde iki terim vardır, *dağıtım noktası*, algılayıcıun yerleştirileceği nokta yeridir. Bu, algılayıcının nihai olarak kalacağı yer değildir. Algılayıcı düğümü, belli bir pdf'ye göre bu nokta etrafındaki noktalara yerleşebilir ve bu nokta pdf ortalamasıdır. Ve *kalma noktası*, algılayıcının nihai olarak yerleşeceği nokta yeridir [64].

Düğümler arasında anahtar dağıtımı

Bir yere yerleştirilen N algılayıcı düğümleri, $t \times n$ eşit boyutlu gruplara bölünür, böylece $i = 1. . . t$ ve $j = 1, . . . , n$ için her bir $G_{i,j}$ grubu, (i, j) indeksli dağıtım noktasından yerleştirilir.(x_i, y_j), bu grup için dağıtım noktasını göstermektedir.

Dağıtım noktaları, şebeke halinde düzenlenir. Ve dağıtım modeli, şekil 3.7.de görülgüğü gibi şebeke tabanlı bazlı yöntemini takip eder, gerçekçi senaryolarda oldukça yaygın olduğundan bu strateji seçilir.

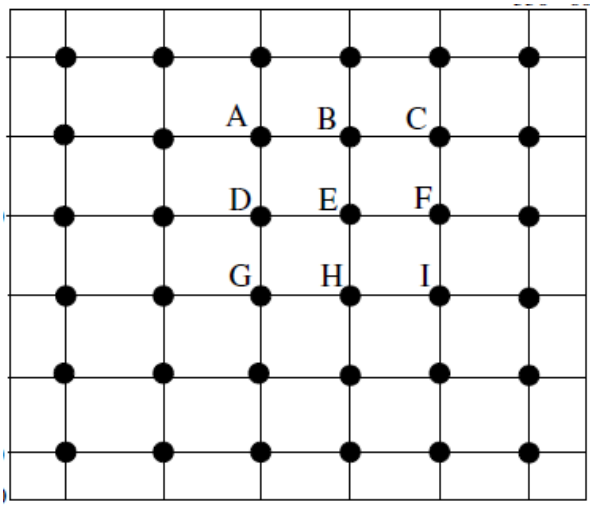
Bu yöntemin amacı, algılayıcı düğümlerinin dağıtımdan sonra komşuların her biriyle genel bir gizli anahtarı bulmasına olanak sağlamaktır. Bu yöntem üç aşamadan oluşur: anahtar ön dağıtımı, paylaşılan anahtar keşfi ve yol anahtarın oluşturulması [64]. Son iki aşama, temel yöntemyla tamamen aynıdır fakat dağıtım bilgisinden dolayı ilk aşama, temel yöntemden önemli ölçüde farklıdır.

Adım 1: Anahtar Ön Dağıtım aşaması: Öncelikle S anahtar havuzu $t \times n$ anahtar havuzlarına bölünür $S_{i,j}$ (ki $i = 1. . ., t$ ve $j = 1, . . . , n$). Her $S_{i,j}$ anahtar havuzunun yöndaşı bir $G_{i,j}$ grubudur. İlgili dağıtım grupları komşu (ya da yakındaki) yerlere yerleştirilirse iki anahtar havuzu komşu olur (ya da birbirine yakın olur). Anahtar havuzlarını bölmenin amacı, komşu anahtar

havuzlarının genelde daha fazla anahtara sahip olmasını sağlamaktır. Anahtar havuzları oluşturulduktan sonra dağıtım grubunda her bir $G_{i,j}$ algılayıcı düğümü için ilgili $S_{i,j}$ anahtar havuzundan *m* anahtarlar seçilir ve bu anahtarlar düğümün belleğine yüklenir.

Adım 2: Paylaşılan anahtar keşif aşaması. Dağıtımdan sonra her düğümün komşularıyla herhangi bir anahtarı paylaşıp paylaşmadıklarını keşfetmesi gerekmektedir. Bunu yapmak için her düğüm, taşıdıkları anahtarların indislerini içeren bir mesaj yayınlar. Her komşu düğüm, yayıncı düğümle paylaştıkları ortak bir anahtar bulunup bulunmadığını öğrenmek için bu yayın mesajlarını kullanabilir. Böyle bir anahtar varsa komşu düğüm, bu anahtarı yayıncı düğümle iletişim kanalını güvenceye almak için kullanır Şekil 3.7.

Adım 3: Yol anahtarı oluşturma aşaması. İki komşu düğümün aralarında ortak bir anahtar bulamaması muhtemeldir. Bu durumda ortak bir anahtar konusunda anlaşmaya varmak için güvenli bir yol bulmaları gerekir. U düğümü V düğümüyle iletişime geçmek isterse ve ikisi ortak bir anahtarı paylaşmıyorsa U düğümü, V düğümüyle iletişime geçmek istediğini belirterek komşu I düğümüyle iletişim kurmalıdır. Sonra U düğümü, I düğümünekendi kimliğini ve gizli bir anahtarı gönderir ve I düğümü V düğümüyle ortak bir anahtarı paylaşıyorsa V düğümüne paylaşılan bu anahtarla şifrelenmiş bir mesaj gönderir. Bu U → I → V veya V → I → U yolu sayesinde hem U hem de V düğümü gizli bir anahtar kullanarak birbirileriyle haberleşebilirler.



Şekil 3.7. Dağıtım noktası (her bir boncuk dağıtım noktasını göstermektedir) [64].

3.5.11. Yer bağımlı anahtar dağıtım yöntemi

Yer bağımlı anahtar dağıtım yöntemi, ortamdaki yerlerine bağlı olarak hangi anahtarları her bir düğüme yerleştirileceğine karar verir [78].

Bu yöntemde düğümler statik olarak belirlenir. Yalnızca şifrelenmiş kanallar üzerinden iletişim kurabilirler. Bu yöntemdeki düğümlerin de farklı aktarım alanları dikkate alınarak farklı güç seviyelerinde aktarım kapasitesine sahip olduğu kabul edilir. Bu yöntem, düğümler dağıtımından sonra küçük bir zaman aralığı boyunca herhangi bir algılayıcı düğümüne zorla giremez. Bu yöntemin sorunu ağdaki bütün düğümlerin aynı zamanda yerleştirilmesi gereğidir. Bu yöntem, daha önce yerleştirilmiş olan düğümlerin anahtarı bozduğundan ve böylece daha sonra yerleştirilecek olan düğümlerle iletişim kuramayacağından algılayıcı düğümlerinin, zaman içinde farklı noktalara yerleştirildiği senaryolarda kullanılamaz. İki algılayıcı tipini, yani Tutturucu Düğüm (TD) (Anchor node) ile birlikte düzenli algılayıcı düğümlerini kabul ediyoruz. Tutturucu düğümün özelliği, farklı güç seviyelerinde aktarım yapabilmesidir. Bu, mevcut algılayıcı düğümlerinde halen bulunan bir kabiliyettir [78].

N_s algılayıcı düğümlü ve N_a tutturucu düğümlü bir ağ varsayalım. Bu algılayıcı düğümlerinin ömründe üç aşamayı dikkate alınmaktadır. Bunlar[78]:

- ön dağıtım aşaması.
- başlangıç aşaması.
- iletişim aşaması.

Ön dağıtım aşamasında bir anahtar suncusu, düğümler tarafından kullanılacak olan bir anahtar kümesi hesaplar. Anahtarları, anahtar havuzuna yerleştirir. Her bir algılayıcı düğümü, daha sonra her düğümün paylaştığı tek bir ortak anahtarla birlikte bu anahtarların alt kümesiyle yüklenir. Tutturucu düğümleri, anahtar havuzundan anahtarları almaz. Bu aşamadan sonra düğümler ortama yerleştirilir.

Başlangıç aşamasında bütün düğümler ve tutturucualar rastgele dağıtılır. Şimdi tutturucu düğümleri farklı güç seviyelerinde bir işaret yayınlar. Algılayıcı düğümleri bu işaretleri alır ve eski anahtarları ve tutturucu düğümünden gelen işareti kullanarak yeni anahtarları hesaplar. Anahtarların orijinal alt kümesi, yeni anahtarları hesapladıktan sonra algılayıcı düğümünün belleğinden silinir. Aynı şekilde tutturucu düğümler hariç bütün algılayıcılar, hepsinin paylaştığı ortak bir anahtarı, ağda fiziksel yerleriyle doğrudan bağlı olan algılayıcı belleklerinde bulunan anahtarları siler.

İletişim aşamasında: düğümler, aralarında güvenli bir iletişim oluşturmak için ikili anahtarları hesaplarlar.

Bu yöntemde ele geçirilmiş düğümler, ağda farklı bir yerde bulunan düğümleri etkilemez. Bir rakip bir düğümü ele geçerse tutturucu düğümlerine bağlanan diğer düğümlerle iletişim kuramazlar. Bu nedenle ele geçirilmiş düğümler, bütün ağı değil yalnızca diğer lokal düğümleri etkilerler.

Dört öge, yere bağımlı yöntem performansını değerlendirir [78]:

1. Anahtar havuzundaki anahtarların sayısı *P,* ve her bir düğümdeki anahtar sayısı *R* öyle seçilir ki yüksek bir bağlanabilirlik oranı elde eder. P*/R* oranı arttığında uyumlu düğüm etkisi azalır.

2. Ortak anahtar eşiği N_c arttırılırsa, R düşük olduğunda bağlanabilirlik oranı azalır.

3. Tutturucu aktarım aralığı. Tutturucu düğümünün maksimum aralığı arttırılırsa düğüm ele geçirme oranı azalır ve sonra tutturma düğümü aralığı büyürken artar.

4. Algılayıcı aktarım aralığı. Algılayıcı düğümleri aktarım aralığı tutturucu düğümü aktarım aralığından daha geniş olduğunda bağlanabilirlik oranı ve uzlaşma oranı azalır.

3.5.12. Yer bilinçli katışımsal anahtar dağıtımı

Bu yöntem, KAA'larda anahtar dağıtım sorununu çözmek için ortaya çıkmıştır. SHELL adı verilen kümelenmiş KAA'lar için anahtar yönteminin hafif birleştirilmiş bir yapısıdır. SHELL'de anahtarların hesaplanmasında düğümlerin fiziksel yerleri kullanılarak gizli anlaşma azaltılır. Bu yöntem, bütün ağa uygulanacak bir komut düğümü (command node) kullanır. Komut düğümü, doğrudan ayrı kümelerin sorumlusu olan ağ geçidi (Gateway) düğümleriyle iletişim kurar. Düğümler, herhangi bir zamanda bu ağa eklenebilir. Ağ geçidi düğümleri, komut düğümüyle iletişim kuracak kadar güçlüdür ve gerekli anahtar dağıtım işlevlerini yerine getirir [79].

SHELL anahtar dağıtım yönteminde, her ağ geçidi düğümü, ağda en az diğer iki ağ geçidi düğümüyle iletişim kurabilir ve üç tür anahtara sahiptir. İlk anahtar tipi, ağ geçidinin doğrudan komut düğümüyle iletişim kurmasını sağlayan ön yüklemeli

anahtardır. İkinci tip, farklı ağ geçidi düğümlerinin iletişim kurmasına olanak sağlar. Üçüncü anahtar tipi, ağ geçidinin kümesindeki bütün algılayıcı düğümleriyle iletişim kurmasını sağlar [79]. Düğüm yerleşiminden sonra ağ geçidi düğümleri, komut düğümüne geri dönen bir bağlantı kurar. Komut düğümü bütün ağ geçidi düğümleriyle bağlantılara sahip olduğunda ağ geçitleri arasındaki iletişim için bağlantıya özel anahtarları hesaplar. Daha sonra komut düğümü, bu anahtarları ağ geçidi düğümlerinin her birine gönderir. Komut düğümü de kendi özel kümelerinde algılayıcı düğümleriyle iletişim için kullanmak üzere anahtarları ağ geçidi düğümlerine gönderir. Bütün ağ geçidi düğümleri oluşturulduğunda ve bağlandığında kümeler arası algılayıcı keşfine başlar.

Özel bir kümeyi yöneten ağ geçidi düğümü, kendi kümesi için anahtarlar üretmez. Komut düğümü, bu küme için anahtarları üretmek üzere diğer iki küme başlıklarını görevlendirir. Anahtarlar hesaplanınca tayin edilen idari anahtarlar (administrative node) kümesi her düğüme bildirilir. Daha sonra küme başlıkları, kendi kümelerinde algılayıcı düğümleri için üretilen anahtarları yayınlarlar [79].

Ağa yeni bir düğüm eklendiğinde komut düğümü, ağ geçidi düğümünün kendi güç aralığına genişlemesine izin verir ve yeni düğüm ile iletişim kurması için anahtarını verir. Ağ geçidi düğümleri, yeni düğümün hangi kümeye katılacağına karar verir.

3.6. Dinamik Anahtar Dağıtımı

Dinamik anahtar dağıtım yöntemleri, istek üzerine ya da düğümün yakalandığının saptanması üzerine periyodik olarak idari anahtarları değiştirebilir. Dinamik anahtarlamanın en büyük avantajı, yakalanan anahtar(lar) tekrar anahtarlama olarak bilinen işlemde zamanında değiştirildiğinden artırılmış ağ kurtarılabilirliğidir. Dinamik anahtarlamanın diğer avantajı ise sabit anahtar havuzu kullanan statik anahtarlamadan farklı olarak yeni düğümlerin eklenmesi üzerine ağ için en iyi desteği sağlamasıdır [80].

Statik ve dinamik anahtar dağıtımı yöntemleri arasındaki farklar: Statik anahtar dağıtım yöntemleri, tipik olarak kısa süreli ağ ömrüne, daha büyük anahtar havuzlarına sahiptirler ve anahtar havuzu büyük olduğundan büyük bir tekrar anahtarlama masrafına sahip olabilir. Dinamik yöntemler genellikle daha uzun ağ ömrüne, daha küçük anahtar havuzuna ve her düğüm üzerinde daha az anahtara sahiptirler ve tekrar anahtarlama için yalnızca birkaç mesaj gerektirirler [80].

Dinamik anahtarlamada en büyük sorun, güvenli etkili bir yeniden anahtarlama mekanizmasının tasarlanmasıdır [80].

DTS'de hile sorununa yönelmek için yeni teklif edilen sisteme verilen isim SHELL'dir. SHELL, her bir küme içinde tekrar anahtarlama işlemini gerçekleştirmek için DTS çerçeve çalışmasını kullanır [80]. Küme ağ geçitleri, anahtar atama izlerini korur fakat gerçek anahtarların izlerini korumaz, bu anahtarlar diğer kümelerin ağ geçitleri tarafından saklanır. SHELL hileye karşı dirençlidir. SHELL, anahtar atamasında dağıtım sonrası yer bilgilerini kullanır; birbirilerine yakın yerleştirilen düğümler, birbirinden daha uzağa yerleştirilen düğümlerden daha fazla sayıda anahtarı paylaşır.

LOCK (LOcalized Combinational Keying) , ek yükü azaltmak için yerel yeniden anahtarlama (localized rekeying) işlemini gerçekleştirir. LOCK, kümelenmiş algılayıcı ağı için anahtar dağıtım yöntemidir [80]. Fiziksel ağ modeli, küme lideri (cluster leader) düğümlerle sonra da düzenli algılayıcı takip edilen üstte baz istasyonlu üç bağlayıcılı bir algılayıcı ağıdır. LOCK'ta düğümlerin beklenen yerleri için hiçbir ön dağıtım bilgisi kabul edilmez. LOCK, anahtarların üretiminde konum bilgilerini kullanmaz. Başlangıçta düğümler ortama serbest bırakıldıklarında bir dizi yedekleme anahtarı oluştururlar. Bu yedekleme anahtarı kümeleri, yerel küme lideri düğümlerle değil yalnızca baz istasyonuyla paylaşılır. Bir düğüm yakalanırsa diğer düğümler, ele geçirilmiş düğümler aralarında iletişim kuramayacağı şekilde yerel olarak yeniden anahtarlandırılır. Küme lideri yakalanmışsa baz istasyonu, küme başı

seviyesinde tekrar anahtarlama işlemini başlatır. Aynı şekilde yakalanmış küme lideri tarafından yönetilen grup içindeki düğümler baz istasyonuyla tekrar anahtarlandırılırlar. LOCK'ta bir rakip herhangi bir düğümü gizliliğin ihlal etmişse diğer kümelerdeki diğer düğümlerin çalışmalarını etkilemez [80].

3.7. Hiyerarşik Anahtar Dağıtımı

3.7.1. LEAP: Büyük ölçekli KAA'lar için etkili güvenlik

LEAP (Localized Encryption and Authentication Protocol) ağ içi (in-network) işlemlemeyi desteklemek için tasarımlanmış olan algılayıcı ağlarına yönelik bir anahtar yönetim protokolüdür.

LEAP büyük ölçekli KAA'larda güvenli haberleşmeleri destekleyecek biçimde tasarımlanmıştır; bu nedenle de gizlilik, kimlik doğrulama ve asıl kaynağı belgeleme gibi temel güvenlik hizmetleri sağlamaktadır.

LEAP bazı düğümlerin ele geçirilmesi nedeniyle tüm ağın durmasına engel olacak biçimde sisteme hayatta kalabilirlik özelliği kazandırır.

LEAP, farklı güvenlik gerektirimlerini yerine getirmek için düğümler arasında değişilen farklı ileti türlerinin gerekli olduğu teorisine dayanmaktadır. Bir algılayıcı ağı içinde aktarılan tüm paketlerin, bir düşman tarafından herhangi bir anda yanlış veri ile bir KAA'ya saldırılabileceğinden dolayı bir algılayıcı düğümünün verinin kim tarafından gönderildiğini bildiği durumda her zaman doğrulanması gereklidir.

LEAP, her bir düğüm için kişisel, parçalı, küme ve grup olmak üzere dört çeşit anahtarın belirlenmesini destekler [81].

- *Özel anahtar:* Bütün düğümlerin her birinin baz istasyonu ile paylaştıkları benzersiz bir anahtarları vardır. Bu anahtar, düğüm ile baz istasyonu arasındaki güvenli iletişim için kullanılır. Baz istasyonu, anahtarlama malzemesi ya da bir özel düğüme gönderdiği özel talimat gibi herhangi bir hassas bilgiyi şifrelemek için bu anahtarı kullanabilir. Bir u düğümüne ait K^m_u özel anahtarı aşağıdaki gibi üretilir: $K^m_u = fK^m(u)$. Burada f sözde rastgele fonksiyon ve K^m sadece denetleycinin bildiği asıl anahtardır. Özel bir u düğümü ile haberleşmede bulunmaya ihtiyaç duyduğunda K^m_u anında hesaplar.

- *İkili anahtar:* İkili anahtarlar sadece düğümlerle onların yakın komşuluklarında bulunan düğümler arasında (ör: bir sekmeli komşu düğümler) paylaşılır. Komşu ilişkisi önceden belirlenmiş olan düğümler için ikili anahtar belirleme basitçe algılayıcı düğümler kendilerine karşılık gelen ikili anahtarlarla önceden yüklenerek yapılabilir.

- *Küme anahtarı:* Bu, bir düğüm ile onun komşuluğunda bulunan düğümler arasında paylaşılan bir anahtar olup ağ içi işlemlemeyi desteklemesinden dolayı son derece önemlidir. Bir düğüm, komşuluğunda bulunan bir düğüm tarafından aynı iletinin daha iyi bir sinyalle gönderilmesi durumunda baz istasyonun bir ileti göndermemeyi seçebilir.

  Küme anahtar belirleme yöntemi ikili anahtar belirleme yöntemini takip eder. u düğümünün yakın komşuluğunda bulunan tüm $v_1, v_2, ..., v_m$. düğümleri ile bir küme anahtarı tesis etmek istediği durumda, u düğümü ilk önce bir K^c_u rastgele anahtarı üretir, sonra bu anahtarı komşuluğunda bulunan her bir düğümle paylaşılan ikili anahtarla şifreler ve sonra da şifrelenen bu anahtarı her bir v_i komşu düğümüne iletir. v_i düğümü K^c_u anahtarının şifresini çözer.

- *Grup anahtarı:* Bu, tüm gruba yayınlanan iletileri şifrelemek için baz istasyonu tarafından kullanılan, küresel olarak paylaşılan bir anahtardır. Grup anahtarı ağ

içindeki tüm düğümler arasında paylaşıldığı için, grup anahtarı bir düğümün iptal edildiği her defasında mutlaka güncellenmelidir.

Aşağıda tanımı verilmiş olan dört adım yeni ilave edilmiş bir *u* düğümünün daha önce dağıtılmış olan kendi komşuluğu içindeli düğümlerin her biri ile ikili bir anahtar belirlemesinin yöntemini göstermektedir [81].

a) **Anahtar Ön – dağıtımı:** Denetleyici bir K_I başlatma anahtarı üretir ve her bir düğüm bu anahtarla yükler. Her bir *u* düğümü bir $K_u = f_{K_I}(u)$ asıl anahtarı türetir.

b) **Komşu Keşfetme:** *u* düğümü ilk önce kendi kimliğini içeren bir MERHABA iletisi yayınlar ve her bir komşusunun *v* düğümünün kimliğini içine alan bir ACK (alındı) iletisi ile yanıt vermesini bekler. Bütün *v* komşularının her birinden gelecek olan ACK iletisi *v* düğümünün K_v asıl anahtarı ile doğrulanır ve bu da $K_v = f_{K_I}(v)$ olarak türetilmiş bulunmaktadır. *u* düğümü K_I'yi tanıdığından K_v'yi türetebilir ve daha sonra da *v*'nin kimliğini doğrulayabilir. $u \longrightarrow *: \quad u., v \longrightarrow u: \quad v, MAC(K_v, u|v)$.

c) **İkili anahtar belirleme:** *u* düğümü *v* ile olan ikili anahtarını K_{uv}, bu da $K_{uv} = f_{K_u}(u)$ biçiminde hesaplar. *v* düğümü, aynı zamanda K_{uv}'yi de aynı yolla hesaplayabilir ve bu onların ikili anahtarı olarak işlev görür. Bu adımda *u* ile *v* arasında hiçbir ileti alış verişi gerçekleşmez.

d) **Anahtar silme:** Zamanlayıcısının süresi dolduğu zaman, *u* düğümü, komşu keşfetme aşamasında hesaplamış olduğu K_I 'yı ve kendi komşuluğunda bulunan tüm K_v anahtarlarını siler. *u* düğümünün kendi asıl anahtarı olan K_u'yi silmediğine dikkat edin. Her bir düğüm kendi asıl anahtarını muhafaza eder.

Yukarıdaki adımlardan sonra, *u* düğümü kendi komşuluğunda bulunan her bir düğüm ile ikili paylaşımlı bir anahtar belirlemiş olacaktır ve bu ikili anahtar bunların kendi aralarında değişilen verilerin güvenilir bir şekilde alınması için kullanılacaktır. İki düğümün bir doğrultu içinde bir ikili anahtar ve bunların güvenli haberleşmesi sırasında ters doğrultuda diğer bir anahtar kullanması gerekli değildir.

Bu yöntemin avantajları [81], farklı tip ileti yayılması için dört çeşit anahtar dağıtımı yöntemi sağlamak, batarya kullanımını ve haberleşme yükünü azaltmak, yerel yayın belgeleme için değişik µTESLA kullanmak sağlar. Bu yöntemin sakıncaları da, her düğüm dört tip anahtar depolar bundan dolayı aşırı bellek gerekir, ve tek-yol anahtar zinciri. Haberleşme ve hesaplama yükü ağın yoğunluğuna bağlıdır (daha fazla ağ yoğunluğu daha fazla yük demektir.

3.7.2. Heterojen algılayıcı ağları için anahtar dağıtım yönetimleri

Heterojen algılayıcı ağları (HAA) az sayıda güçlü Üst – uç algılayıcılardan (Ü – algılayıcılar) ve çok sayıda Alt–uç algılayıcılarından (A–algılayıcıları) oluşur [80].

HAA için etkili anahtar dağıtım yöntemleri HAA'lerin AP (Asymmetric Predistribution) yöntemi olarak bilinir. Ana fikir, her bir Ü – algılayıcısı çine göreli olarak büyük sayıda anahtarı önceden yüklerken her bir A –algılayıcısı içine sadece az sayıda anahtarı önceden yüklemektir; çünkü bir Ü – algılayıcısı bir A – algılayıcısından çok daha fazla depolama alanına sahiptir [82]. Bir Ü – algılayıcısı içinde önceden dağıtılmış anahtarların sayısı bir A – algılayıcısı içindekinden son derece farklı olduğundan biz bu yöntemi asimetrik bir ön dağıtımlı anahtar dağıtım yöntemi olarak adlandırılır. AP yöntemi üç aşamadan oluşur [82]:

Anahtar ön dağıtım aşaması: İlk önce P anahtar ve buna karşılık gelen anahtar kimliklerinden oluşan bir havuz üretilir. Daha sonra, her bir A – algılayıcısı, anahtar havuzundan değiştirmeden rastgele seçilen *l* anahtarları ile ön yüklenir. *l* anahtarları

her bir A – algılayıcısı içinde bir anahtar halkası oluşturur. Aynı zamanda, anahtar havuzundan değiştirmeden rastgele olarak seçilen *M* anahtarları ile de Ü – algılayıcısı ön yüklenir, burada $M \gg l$. Buna ek olarak, her bir Ü – algılayıcı özel bir anahtar olan $K_{Ü}$ ile ön yüklenir.

Paylaşılan anahtar keşif yöntemi: paylaşılan anahtar keşif yöntemi küme oluşumundan sonra başlar. Merkezileştirilmiş yöntemde her bir A – algılayıcı kendi komşuları ile haberleşir ve paylaşılan anahtarları tespit eder. Düğüm halkasında bulunan anahtar kimlik listesini yayınlamak ile her hangi anahtar paylaşan iki A – algılayıcı düğümleri keşfedilir. Dağıtılmış yöntem olan bir alternatif yöntemde her bir *u* A – algılayıcısı, kendisinin küme başı H'ye *u* A – algılayıcısı kimliği, *u*'daki anahtar kimliklerini ve *u*'nun yerini içeren şifrelenmemiş bir Anahtar – listesi gönderir. Daha sonra H, komşu A – algılayıcılarının her bir çifti arasında paylaşılan anahtarları keşfeder. *u* ve *v* arasındaki uzaklık bir A – algılayıcısının aktarım aralığından daha az ise H, *u ve v*'nin komşu olduklarını kabul eder. H, komşu A – algılayıcılarının her bir çifti arasında paylaşılan anahtarları keşfettikten sonra paylaşılan anahtarları kullanarak paylaşılan anahtar bilgisini A – algılayıcılarına dağıtır (serpiler). Eğer *u* ve *v* birden fazla paylaşılan anahtara sahip ise, ek yükü azaltmak için bunlardan sadece biri paylaşılan anahtar içine dahil edilir. Küme içindeki A – algılayıcılarının sayısı çok büyük değil ise bir adet paylaşılan anahtar iletisi tüm komşu çiftleri için paylaşılan anahtar kimliklerini, *u*'yu *ve v*'yi içerebilir ya da Ü – algılayıcısı kısa bir paylaşılan anahtar iletisi gönderebilir ya da sadece *u* ve *v*'yi çoklu noktaya yayınla gönderebilir.

<u>İkili anahtar kurulumu aşamasına dayalı Ü - algılayıcısı</u>

Bazı A –algılayıcıları komşularla ön yüklenmiş hiçbir anahtarı paylaşmayabilir; H ilk önce her biri ile ayrı ayrı olmak üzere H ile A – algılayıcılarının her bir çifti arasında paylaşılan bir anahtar elde eder ve sonra da H her bir çift için bir ikili anahtar üretir ve anahtarı bunlara güvenli bir biçimde gönderir. Bir A – algılayıcısı *x*'in küme başı

H ile ön yüklenmiş herhangi bir anahtara sahip olmaması durumunda H, 1'nci~ n'ynci dereceden komşunun *x* ile herhangi bir anahtarı paylaşıp paylaşmadığını kontrol eder. Eğer bunlardan hiçbiri *x* ile paylaşılan bir anahtara sahip değilse, H, baz istasyonuna *x* düğümünün bir adet anahtar kimliği içeren bir *İstek* iletisi gönderir; ve sonra da baz istasyonu, K_H tarafından şifrelenmiş olaran H'ye karşılık gelen anahtarı gönderir. Anahtarları baz istasyonundan aldıktan sonra H, *x* ve *y* A – algılayıcılarının her bir çifti için bir $K_{x,y}$ ikili anahtarını üretebilir ve $K_{x,y}$ anahtarını *x* ve *y* düğümüne tek noktaya yayın yaparak gönderebilir. Daha sonra, *x* ve *y* $K_{x,y}$ çift paylaşılmış anahtarına sahip olacaktır ve güvenli iletişime başlayabileceklerdir [82].

4. PERFORMANS DEĞERLENDİRMESİ

4.1. Benzetim Ortamları

Anahtar dağıtımının benzetmesini yaparken enerji verimliliği, hata toleransı, QoS, eşzamanlama, planlama stratijileri, system topolojisi, protokollerin uyumluluğu ve haberleşmeleri gibi KAA'ların tasarımını etkileyen birçok faktorlerin göz önüne alınması gerekir [83].

Ticari olan ve ticari olmayan olmak üzere, KAA'ların iki tür benzetme ortamı yazılımları vardır. Ticari olmayan benzetme ortamları örneğin şu yazılımları içerir (SMURPH, NS2, Ptolemy, NetSim++, C++Sim, CLASS, omnet++) [84]. Ticari benzetme ortamları da şunları içermektedir (OPNET, EmStar, Atemu, Sense, Avrora) bu yazılımların ticari olmayan en ünlüleri NS2' dur, OPNET' ise ticari olan yazılımların en ünlüleridir [84].

OPNET, OMNeT++ gibi göreli iç içe derinli hiyerarşişk modüllere izin verir. Hiyerarşik modüller ve ağ topolojisi açısından, OMNeT++ NS2 dan daha iyidir, OPNET çok sınırlı ve NS2de bu tür yetenekte yetersizdir [85]. OMNeT++ (thread/coroutine-based programming model, ve FSMs built upon a message-receiving function), her iki program biçimini de destekler, ama OPNET ve NS2 sadece son biçimi destekler. OMNeT++ hata düzeltme amacıyle üç araç sunmaktadır(otamatik animasyon, modül, amaç denetleyicisi).

Modüller arasında önemli olmayan birçok bağlılık uygulayan nesne yönelimli (object-oriented) yapı, NS2'nun problemlerinden biridir. Bunun gibi bağlılıklar zaman zaman yeni protokol modüllerini eklemeyi çok zor kılar ve sadece simülatürü çok iyi bilen deneyimli insanlar tarafından yapılabilir [85]. Ayrıca, yapılan çalışmalar. NS2'de hafıza kullanımının çok yüksek olduğunu göstermektedir [85].

Bunlara ek olarak benzetme senaryolarını belirleme veya benzetme izlemesi için inceleme aracı yoktur.

NS2 tarafından çözülmeyen problemleri, JavaSim, SSFNeT, Glomosim gibi çözmeyi deneyen birçok ağ simulator bulunmaktadır. Bunların arasında, nesne yönelimli tasarımının sakıncalarını farkına varan JavaSim geliştiricileriydi ve bileşen altyapısını (component-oriented architecture) kullanarak bu eksiklerin çözmesini denemiştirler. Paralel simülasyonla ilgilenen SSFNeT ve Glomosim tasarımcıları daha sonra kablosuz ağlarla odaklamışlar.

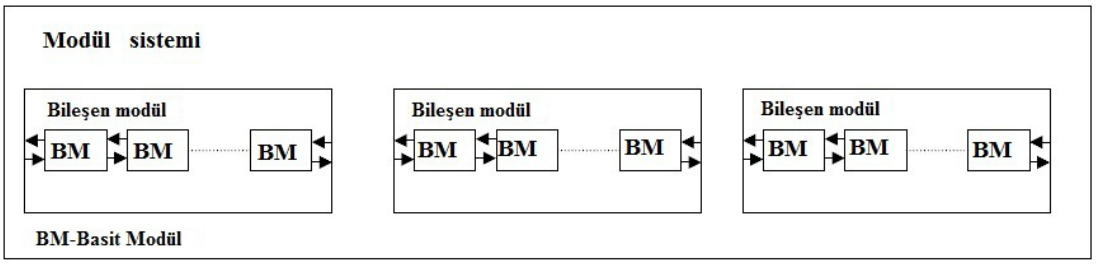
Yukarıdaki bahsedildiğine dayanarak, NS2'nun performansı pek çok iyi değildir. OPNET'in performası ise OMNeT++ gibi çok iyidir yalnız çok pahalıdır. Dolayısıyla KAA'lar için OMNet++ diğer benzetim ortamlarınıdan daha iyidir.

4.2. OMNeT++

OMNeT++ “Objective Modular Network Test-bed in C++” [86]. OMNeT++ açık kaynak kodlu, ayrık olay (discrete event) simülatörü, nesne yönelimli (object-oriented) modüler, bileşen bazlı (component-based) modüler, güçlü bir grafik kullanıcı arayüzüne sahip (GUI) ve gömülü çekirdekli simülasyon (embeddable simulation kernel) ile açık mimarisi(open-architecture) benzetim ortamıdır. NS2 gibi ISO/OSI modeli hedefler [87]. Binlerce düğümü işleyebilir, grafiksel ağ editörünü bellirler, veri akışını ve ağı görselleştirir. Yüksek performans için simülatör c++ diliyle yazılmış ve NED “NEtwork Description Language” diliyle desteklenmiştir. OMNeT++'in ana amacı bileşen mimarisi sağlamaktır, ki bunun araçlığıyla simülasyon çok esnek oluşur. Bileşenler c++ diliyle programlanır sonra, NED daha büyük bileşenlerle toplanır. Bilimsel amaçta kullanmak için ücretsizdir. Yanı sıra ticari sürümü de bulunmaktadır [87].

Ana uygulama alanı haberleşme ağlarının simülasyonudur, ancak kapsamlı ve esnek mimarisinden dolayı, diğer alanlarda başarılı bir şekilde kullanılmıştır. OMNeT++ modeli hiyerarşik biçimde iç içe modüllerden oluşur. Tüm simülasyon modelini kapsayan ve "ağlar" olarak alınan sistem modeli, en üst seviye modelidir. Sistem kendi alt modüllerine de sahip olabilen alt modüller içerir. Bundan dolayı modüller birbirinin içine her seviyede girebilecek şekilde tanımlanabilir. Bunun sonucunda kompleks sistem modelleri birkaç basit modülün (simple module) bir kombinasyonu olarak tanımlanabilir. Alt modüller içeren modüllere bileşik modeller denir. Basit modüller modüllerdeki algoritmaları içerir ve modül hiyerarşisinin en alt seviyesini oluşturur. Kullanıcı basit modülleri uygulamak için, C++ ve OMNeT++ simülasyon sınıf kütüphanesini kullanır. Modüller karmaşık bir veri yapısı gibi olan mesajları aktararak haberleşirler. Modüller, gitmesini istedikleri yere doğrudan ve bir dizi geçit ve bağlantılar yoluyla diğer modüllere mesaj gönderebilirler. Mesajlar bir ağ simülasyonu içerisindeki çerçeveleri veya paketleri temsil edebilir [88]. Bir modül diğer modüllerden veya aynı modülden kendisine gönderdiği gelen mesajları aldığında bölgesel simülasyon zamanı artar, Bu da simülasyon zamanlayıcısını temsileder. Modül kendi kendisine göndern mesajlar (self-messages), daha sonraki bir zamanda modül tarafından yürütülecek olan olayları planlamak için kullanılır. Modüllerin yapısı ve ara yüzü bir (NED) kullanılarak tanımlanır. NED basit modüllerin temel davranışlarını uygularlar. Simülasyon uygulamaları başlatma dosyaları vasıtasıyla kolaylıkla düzenlenir. NED üretilen olayları izler ve mesajların doğru modüle ve doğru zamanda gönderilmesini sağlarlar [88].

Her giriş ve çıkış arayüzüne geçit (gate) adı verilen araçlığıyla modüller arasında iletiler gerçekleşir. İletiler çıkış-geçitiyle(output-gat) gönderilir ve giriş -geçitiyle (in-gate) alınır. Topolojiye veya sisteme dayanarak bağlantılar alt-modüller arasında veya alt-modüller ile bileşimler arasında yaratılır. Modül sistem mimarisi şekil 4.1. gibi gösterilebilinir [87].



Şekil 4.1. Modül mimarisi [87].

4.3. Performans Kriterleri

Algılayıcı ağları, birkaç özelliğe sahiptir ki bu ağları saldırılara karşı açık kılar. Anahtar dağıtım yöntemini değerlendirmek için aşağıdaki açıklanan beş kriter değerini simülasyonda inceleyeceğiz:

- Güvenli bağlanabilirlik: Güvenli bağlanabilirlik ağdaki tüm bağlantılara göre güvenli bağlantıların oranını göstermektedir. Bir anahtar dağıtımı yöntemi için yüksek bir güvenli bağlanabilirliği elde etmek için, ya gizli anahtar paylaşan düğüm çiftlerine sahip olmaktır ya da etkili ve gizli bir yol anahtar belirleme yöntemi sunmaktır.

- Hafıza yükü: Algılayıcı düğümünün hafızası bir kısmı işletim sistem tarafından kullanılır, geri kalan kısmı ise anahtar dağıtımı yöntemi tarafından kullanılır. Anahtar depolanması hafızayı en iyi biçimde kullanması gerekir, bu da yöntemden yönteme değişir.

- İletişim yükü: Algılayıcı düğümleri arasında paketlerde bulunan anahtar gönderme ve alma nedeniyle bazı ağlarda tıkanıklık veya çarpışmaya neden olur. O yüzden paket sayısını bir seviyede tutmalıyız ki düğümler arasında en iyi şekilde anahtar paylaşımı gerçekleşsin, aynı zamanda ağda çarpışma veya tıkanıklık olmasın.

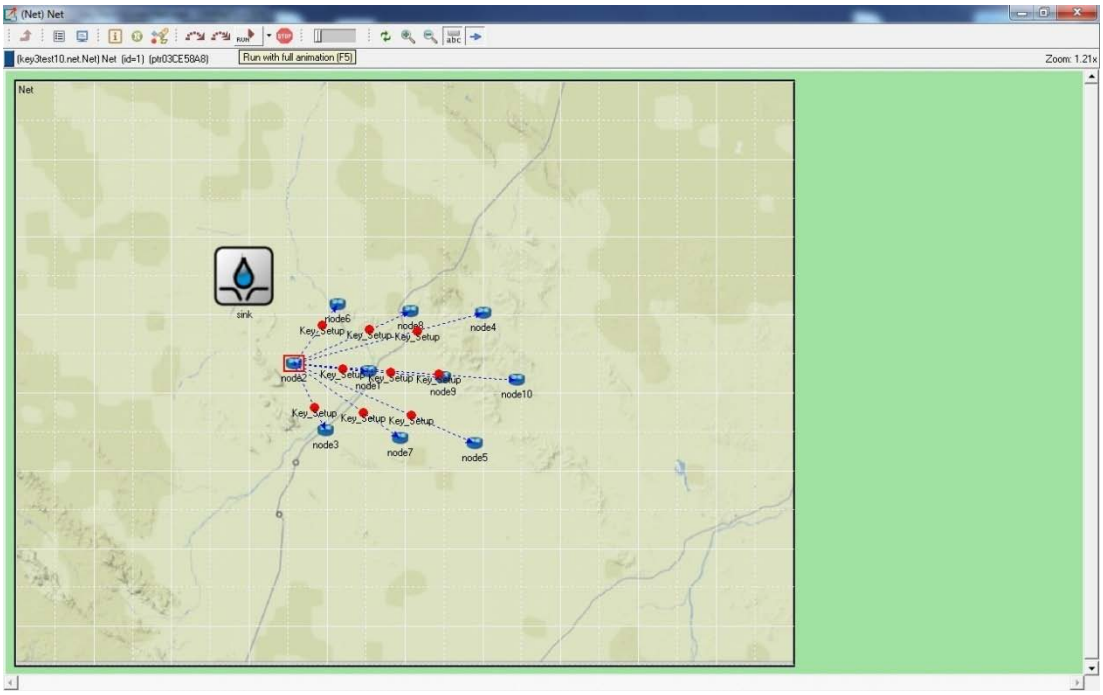
- Süres: Ağda, birinci anahtarın dağıtımından son anahtarın dağıtımına kadargeçen süre önemlidir. Anahtar dağıtım süresinin kısa olması güvenlik ataklarını azaltacaktır.

- Gizliliği ihlal edilmiş düğüm sayısı: Ağın tehlikeye düşmesi, gizliliği ihlal edilmiş düğüm sayısına bağlıdır. Eğer bir saldırgan bir düğümü ele geçirirse, düğüm hafızasında bulunan anahtar listesi ve dolayısı ile bu düğümle iletişime geçen diğer düğümleri elde edebilir.

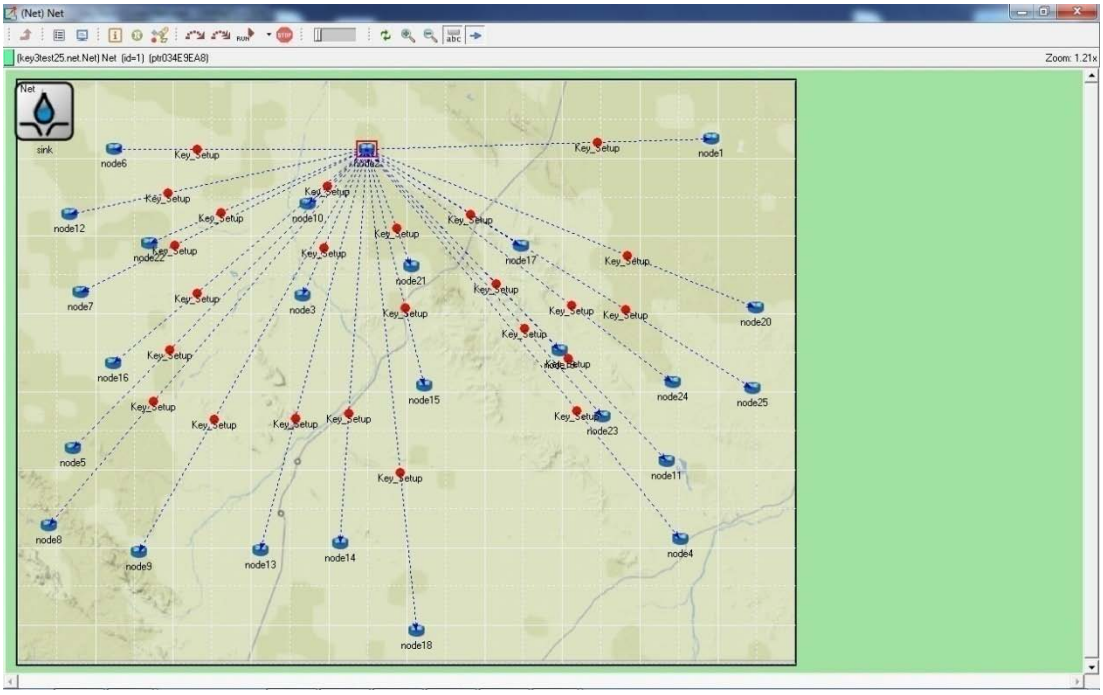
Uygulanan farklı anahtar dağıtım yöntemine göre, yukarıdaki bahsedilen kriterler orta bir ölçüde tutmalıdır ki aynı zamanda hem güvenliği sağlamalı hemde algılayıcı kaynaklarını tüketmemelidir.

4.4. Senaryolar ve Sonuçları

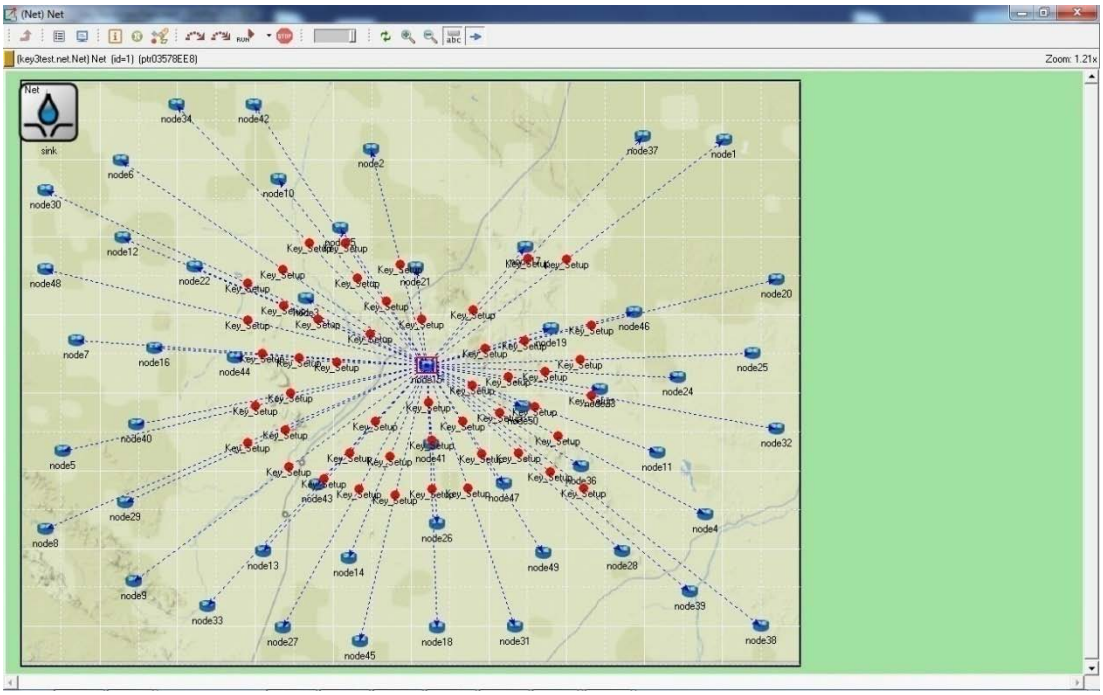
Yukardaki sözü edilen dört kriteri analize ve sonra protokolleri değerlendirmek için üç farklı senaryoda incelenmektedir. Bu kriterleri etkileyen en önemli faktör, ağdaki düğüm sayısıdır. Bu nedenle dört protokolün (tek anahtar, ikili anahtar, temel olasılıksal yöntem, Q-bileşim rastgele anahtar ön dağıtım yöntemi) her biri için üç farklı düğüm sayısından oluşan ağın benzetimini yapmaktayız. Birinci ağ sadece 10 düğümden (Şekl 4.2.), ikinci ağ 25 düğümden (Şekl 4.3.) ve üçüncü ağ ise 50 düğümden (Şekl 4.4.) oluşmak üzere ve düğümler $1000 \times 750m^2$lik bir alana dağılmıştır. Bu ağların her biri diğer iki ağdan bağımsız bir şekilde uygulanmıştır. İncelenen her yöntemi değerlendirmek için güvenli bağlanabilirlik, hafıza yükü, iletişim yükü, zaman ve gizliliği ihlal edilmiş düğüm sayısı açısından analiz yapılmıştır.



Şekil 4.2. 10 düğümden oluşan ağ.

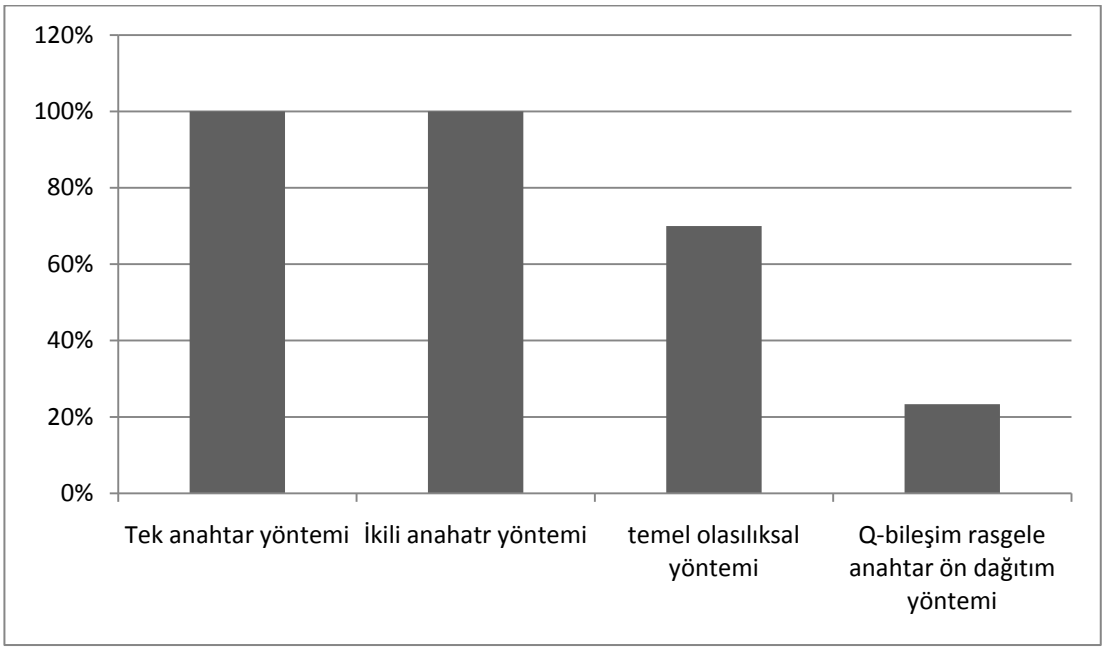


Şekil 4.3. 25 düğümden oluşan ağ.



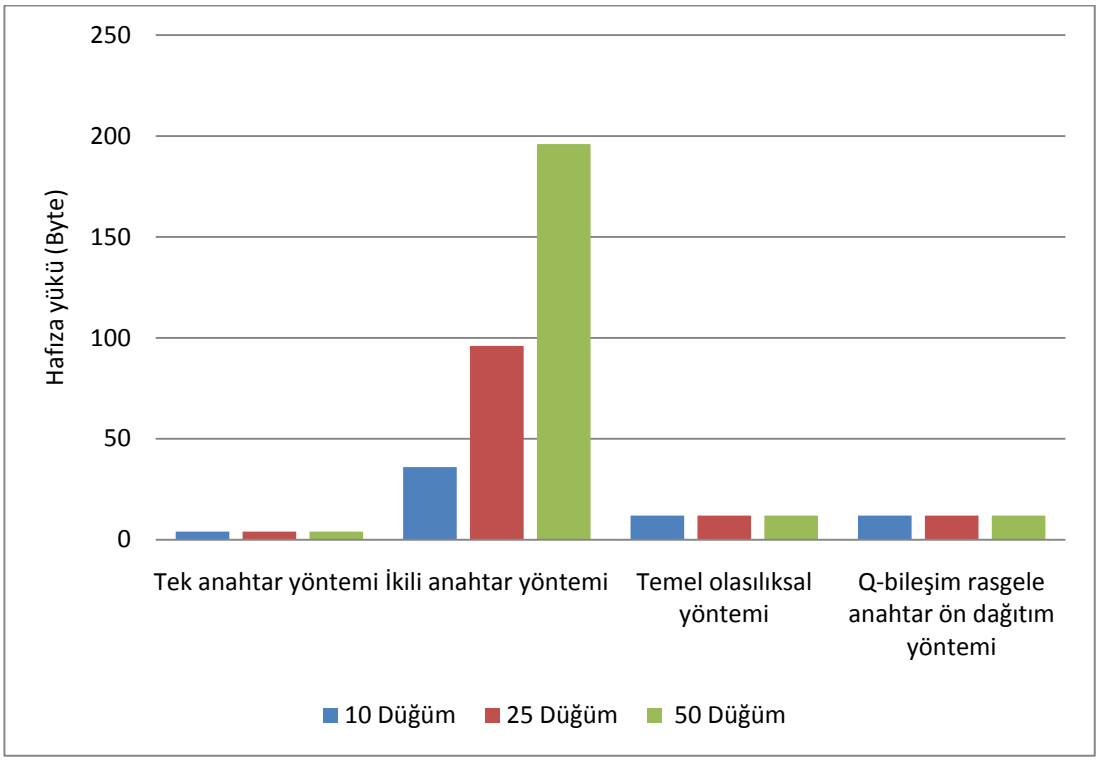
Şekil 4.4. 50 düğümden oluşan ağ.

Yol anahtarı oluşturma aşamasını göz önüne almadan 50 düğümden oluşan bir ağ için güvenli bağlanabilirlik oranını ölçüyoruz, sonuçları da Şekil 4.5.te görmekteyiz. Şekil 4.5.te görüldüğü gibi tekli anahtar yönteminde sadece bir anahtar bulunduğundan dolayı, her düğüm ağdaki diğer düğümlerle bağlanabilir ve %100 güvenli bağlanabilirlik oluşturur. İkili anahtar yönteninde, her düğüm ağdaki diğer düğümlerin anahtarlarını taşıdığı için bağlanabilirlik %100dür. Temel olasılıksal yöntemde her düğüm *p* olasılığına gore diğer düğümlere bağlanır. Simülasyon sonuçu ağın güvenli bağlanabilirliği %70 göstermektedir. Buna benzer şekilde q-bileşim rastgele anahtar ön dağıtım yöntemi kendi komşu düğümlerine bağlanma p(q) olasılığına bağlıdır. Simülasyonda q=3 seçilmiştir ve ağın güvenli bağlanabilirliği %23.33 göstermektedir.



Şekil 4.5. Ağın güvenli bağlanabilirlik oranı.

Şekil 4.6. her anahtar dağıtım yöntemi için farklı ağ boyutlarında hafıza yükünü sunmaktadır. Tek anahtar yönteminde her algılayıcı düğümü sadece 4 byte kullanmaktadır o da her anahtar boyutudur (Longe Bayt). İkili anahtar yönteminde ise her algılayıcı düğümü için farklı bir anahtar kullandığımız için ve her algılayıcı bu anahtarı ağdaki diğer algılayıcılarla paylaştığı için bu denkleme göre hesaplayabiliriz $(n-1) \times 4$, dolayısıyla hafıza yükü ağdaki düğüm sayısına bağlıdır (başka bir deyişle ağdaki düğüm sayısı arttıkça hafıza yükü artmaktadır) bunuda farklı ağ boyutlarıda görebiliriz nitekim 10 düğümden oluşan ağda her düğüm hafızası=40 ve 25 düğümden oluşan ağda =100 ve 50 düğümden oluşan ağda=200. Temel olasılıksal ve Q-bileşim rastgele anahtar Ön dağıtım yöntemlerin de aynı hafıza yüküne sahiptiler, çünkü 3 anahtardan oluşan bir anahtar halkası kullandık sonuc 12 byte olmuştur (Çizelge 4.1.)



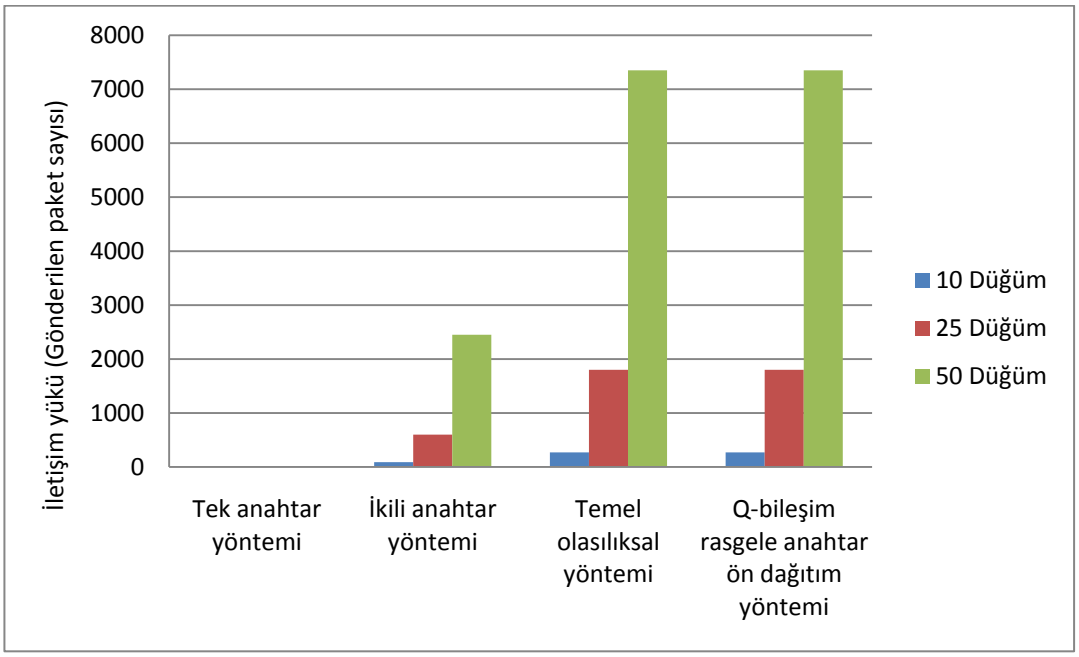
Şekil 4.6. Hafıza yükü.

Çizelge 4.1. Hafıza yükü

	10 Düğüm	25 Düğüm	50 Düğüm
Tek anahtar yöntemi	4	4	4
İkili anahtar yöntemi	36	96	196
Temel olasılıksal yöntem	12	12	12
Q-bileşim rastgele anahtar ön dağıtım yöntemi	12	12	12

Şekil 4.7. ve Çizelge 4.2. her anahtar dağıtım yöntemi için iletişim yükü sonuçlarını açıklamaktadır. Tek anahtar yönteminde tek bir anahtar tüm düğümlere yerleştirmeden önce atadığımız için düğümler arasında iletişime gerek duyulmaz, onun için iletişim yükü sıfırdır. Temel olasılıksal ve q-bileşim rastgele anahtar ön dağıtım yöntemlerini ikili anahtar yöntemiyle karşılaştıdığımız zaman her üç ağda da temel olasılıksal yöntem ve q-bileşim rastgele anahtar ön dağıtım yöntemleri daha yüksektir. Örneğin, son iki yöntemde 25 düğümden oluşan ağda iletişim yükü =

1800’e buna karşı ikili anahtar yönteminde = 270’e. Çünkü temel olasılıksal ve q-bileşim rastgele anahtar ön dağıtım yöntemlerinde kimlik doğrulaması yapmak için her iki yöntemde aynı anahtar boyutunu kullanmaktadır.

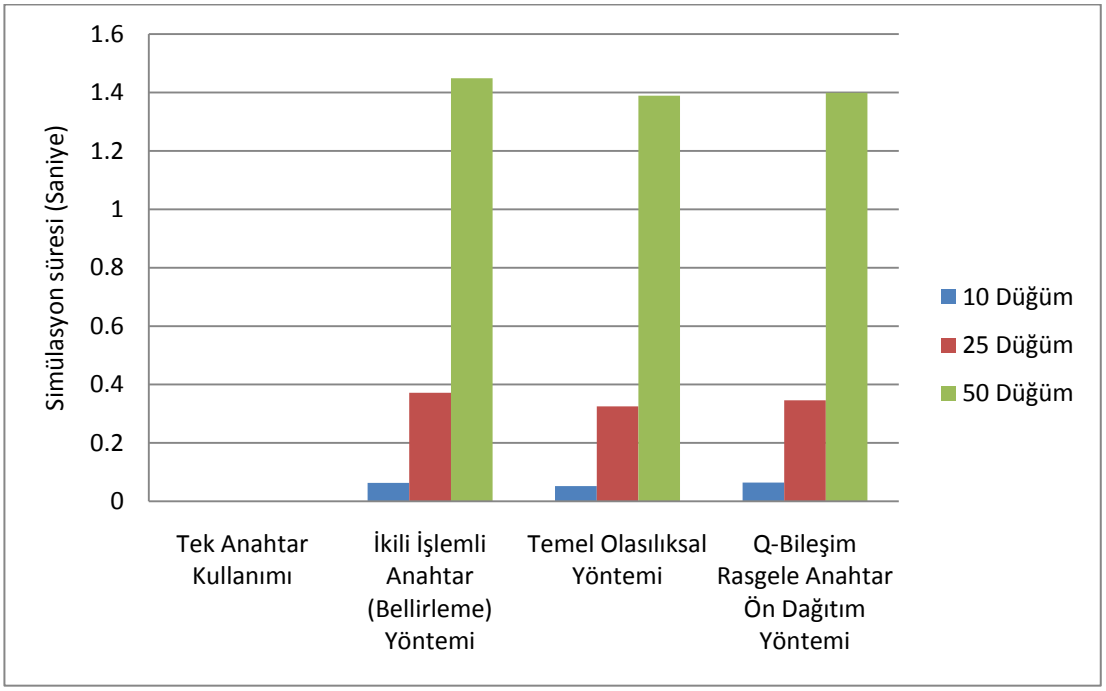


Şekil 4.7. İletişim yükü.

Çizelge 4.2. İletişim yükü

	10 Düğüm	25 Düğüm	50 Düğüm
Tek anahtar yöntemi	0	0	0
İkili Anahtar	90	600	2450
Temel Olasılıksal	270	1800	7350
Q-Bileşim Rastgele Anahtar Ön Dağıtım	270	1800	7350

Şekil 4.8. ve Çizelge 4.3.te olduğu gibi ortak anahtar keşif süresini hesaplarken, tek anahtar yönteminde düğümleri alana dağıtmadan önce anahtarı atadığımız için hiç süre almamıştır sure 0’dır, ikili anahtar, temel olasılıksal ve q-bileşim rastgele anahtar ön dağıtım yöntemlerinde her üç ağda da yaklaşık hepsi aynı süre almıştır .

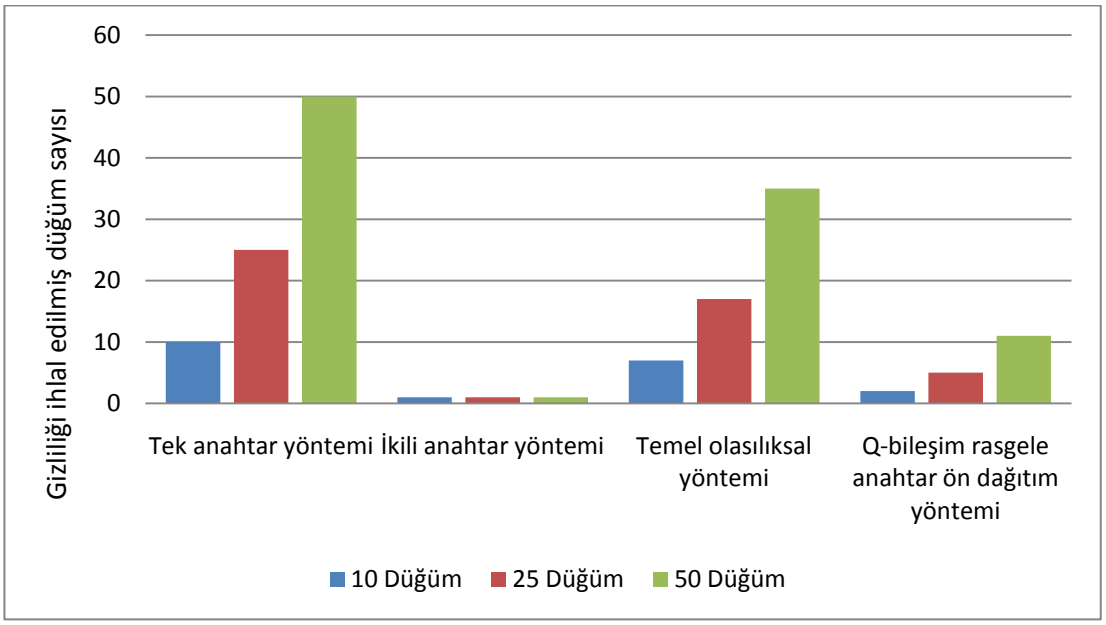


Şekil 4.8. Ortak anahtar keşif süresi.

Çizelge 4.3. Ortak anahtar keşif süresi.

	10 Düğüm	25 Düğüm	50 Düğüm
Tek Anahtar	0	0	0
İkili İşlemli Anahtar (Bellirleme)	0.063	0.372	1.449
Temel Olasılıksal	0.052	0.325	1.389
Q-Bileşim Rastgele Anahtar Ön Dağıtım	0.064	0.346	1.399

Çizelge 4.4'e göre ve gizliliği ihlal edilmiş düğüm sayısıyla ilgili en verimli ve en dayanıklı yöntem ikili anahtar yöntemidir. Tek anahtar yöntemi ise düşman tarafından bir ağı ele geçirmek için bir düğümün gizliliğini ihlal etmek yeterlidir. Temel olasılıksal ve q-bileşim rastgele anahtar ön dağıtım yöntemlerinde, ağıdaki toplam gizliliği ihlal edilmiş düğüm sayısını hesaplamak için altaki formülü kullandık, $\frac{p*n}{s}$, burada, p: anahtar olasılığıdır, n:ağdaki toplam düğüm sayısıdır ve s: ağdaki her iki düğüm arasındaki istenilen ortak anahtar sayısıdır.



Şekil 4.9. Gizliliği ihlal edilmiş düğüm sayısı.

Çizelge 4.4. Gizliliği ihlal edilmiş düğüm sayısı.

	10 Düğüm	25 Düğüm	50 Düğüm
Tek anahtar yöntemi	10	25	50
İkili anahtar yöntemi	1	1	1
Temel olasılıksal yöntem	7	17	35
Q-bileşim rastgele anahtar ön dağıtım yöntemi	2	5	11

5. SONUÇLAR

Algılayıcı ağların gerçek potansiyelini elde etmek için gerekli olan kritik etmen güvenliktir. Bu tezde KAA larda iletişim güveniğini sağlamasının ilk ve en önemli olan anahtar dağıtım yöntemleri incelenmiştir. Performans değerlendirmesinde güvenilir iletişim, hafıza yükü, iletişim yükü, anahtar dağıtım süresi ve düğüm yakalanmasına karşı direnç metrikleri kullanarak tek anahtar, ikili anahtar, temel olasılıksal yöntem ve q- bileşim rastgele anahtar öndağıtım yöntemleri incelenmiştir. Bu metrikleri ölçmek ve değerlendirmek için OMNeT++ "Objective Modular Network Test-bed in C++" açık kaynak (public-source), ayrık olay (discrete event) simülatörü, nesne yönelimli (object-oriented) modüler, bileşen bazlı (component-based) modüler, güçlü bir grafik kullanıcı arayüzüne sahip (GUI) benzetme ortamı kullanılmıştır. Yapılan performans değerlendirmesi sonucunda ağın güvenli bağlanabilirliğinde tek anahtar yöntemi en iyi performansı göstermiştir, ayrıca tek anahtar yöntemi en küçük hafıza gerektirmiştir ve iletişim yükünde en az gönderilen paket sayısına sahiptir . Aynı şekilde süre ölçümünde de yine tek anahtar yöntemi en iyi performansı göstermektedir. Buna karşın, güvenlik açısından tek anahtar yöntemi tüm düğümlerin aynı anahtarı kullanmaları nedeniyle en zayıf yöntem olarak tespit edilmiştir. Düğüm ele geçirme ataklarına karşı, ikili anahtar yöntemi kullanıldığında, en az sayıda düğüm etkilenmektedir. Yapılan performans ve güvenlik analizi sonuçlarında temel olasılıksal yöntem ve Q- bileşim rastgele anahtar ön dağıtım yöntemi hem güvenlik hem de iletişim ve hafıza yükleri açısında KAA'lar için uygun anahtar dağıtım yöntemleri olarak tespit edilmiştir.

KAYNAKLAR

1. Alemdar, A., Ibnkahla, M., "Wireless sensor networks: applications and challenges", ***in: Proceedings of the Ninth International Symposium on Signal Processing and Its Applications (ISSPA 2007), IEEE Computer Society***, 1–6 (2007).

2. Du, X., Chen, H., "Security in wireless sensor networks", ***IEEE Wireless Communications,*** 51(4): 60- 67 (2008).

3. Yick**,** J., Mukherjee, B., and Ghosal**,** D., "Wireless sensor network survey", ***Computer Networks,*** 52(12): 2292-2330 (2008).

4. Kazem, S., Daniel, M., And Taieb Z., ''Wireless Sensor Networks: Technology, Protocols, and Applications'', Mobile Networks and Applications, ***John Wiley & Sons, Inc.,*** New Jersey, 10-12 (2007).

5. İnternet : Mobility Technologies, Wayne, PA, http://www.mobilitytechnologies.com/ntdc/

6. İnternet : Cornell University, Ithaca ''Cougar: The Sensor Network Is the Database,'' http://www.cs.cornell.edu/database/cougar/ (2003)

7. Cowen, S., Briest, S. And Dombrowski, J. ''Underwater Docking of Autonomous Undersea Vehicles Using Optical Terminal Guidance'' ***Annual Joint Conference of the IEEE Ocean Engineering Society and the Marine Technology Society***, vol.2: 1143 – 1147, (1997).

8. Conner, W., S., Heidemann, J., Krishnamurthy, L., Wang, X. And Yarvis, M., ''Workplace Applications of Sensor Networks'', **USC/ISI Technical Report -591,** ***Artech House***, 232-248(2004).

9. Cox, J., ‘‘Wireless Sensor Networks Grabbing Greater Attention’’ ***NetworkWorld***, 864-878 (2004).

10. Hatler, M., ‘‘Wireless Sensor Networks: Mass Market Opportunities’’, ***OneWorld Inc***., 27-48(2004).

11. Estrin, D., Govindan, R., Heidemann, J. And Kumar, S., ‘‘Next century challenges: scalable coordination in sensor networks’’, ***MobiCom '99 Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking***, Washington, USA, 263-271(1999).

12. Xue, Y., “Key Management Schemes for Distributed Sensor Networks”, Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, ***Faculty of Graduate Studies The University of Western Ontario***, London, 19-20 (2008).

13. Chong, C., Kumar, S. P., ‘‘Sensor Networks: Evolution, Opportunities, and Challenges,’’ ***Proceedings of the IEEE***, Vol. 91, No. 8, 1247 - 1256 (2003).

14. Clouqueur, T., Phipatanasuphorn, V., Saluja K. ‘‘Sensor Deployment Strategy for Target Detection,’’ ***Proceedings of the 1st Workshop on Sensor Networks and Applications (WSNA’02)***, Atlanta, GA, 42-48 (2002).

15. G. Hoblos, M. Staroswiecki, A. Aitouche, ‘‘Optimal design of fault tolerant sensor networks’’, ***IEEE International Conference on Control Applications***, Anchorage, AK, (September 2000).

16. Shen, C., Srisathapornphat, C., Jaikaeo, C., ‘‘Sensor information networking architecture and applications’’, ***IEEE Personal Communications***, **Volume:** 8(4), 52 - 59 (2001).

17. Konstantinidis, A., “Multiobjective Deployment and Power Assignment in Wireless Sensor Networks using Metaheuristics”, A thesis submitted for the degree of Doctor of Philosophy, ***School of Computer Science and Electronic Engineering University of Essex***, 67-68(2009).

18. Hoblos, G., Staroswiecki, M., Aitouche, A., “Optimal design of fault tolerant sensor networks”, ***IEEE International Conference on Control Applications***, 467 - 472(2000).

19. Shih, E., Cho, S., Ickes, N., Min, R., Sinha, A., Wang, A. And Chandrakasan, A., “Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks”, ***MobiCom '01 Proceedings of the 7th annual international conference on Mobile computing and networking***, 272-287 (2001).

20. Lester, J., “System Architecture for Wireless Sensor Networks”, ***A dissertation submitted in partial satisfaction of the requirements for the degree of Doctor of Philosophy In Computer Science, UNIVERSITY OF CALIFORNIA,*** Berkeley, 28-29 (2003).

21. Liu, X. and Mohapatra, P., “On the Deployment of Wireless Data Back-haul Networks”, ***IEEE Transactions on Wireless Communications***, vol. 6(4), 1426-1435 (2007).

22. Rabaey, J., Ammer, J., da Silva, J., L., Patel, D., “Pico- Radio: ad-hoc wireless networking of ubiquitous lowenergy sensor/monitor nodes”, ***Proceedings of the IEEE Computer Society Annual Workshop on VLSI (WVLSI’00)***, 9-12(2000).

23. Rulikowski, P., Martinez, R. And Barrett, J., “Sensor Network Hardware Infrastructure for Smart Spaces”, ***1st International Workshop on Managin Ubiquitous Communications and Services (MUCS).*** 79-88（2003）.

24. Kahn, J., Katz, R., Pister, J., “Next century challenges: mobile networking for smart dust”, ***Proceedings of the ACM MobiCom’99***, 271-278 (1999).

25. Vardhan, S., Wilczynski, M., Pottie, G. And Kaiser, J., “Wireless integrated network sensors (WINS): distributed in situ sensing for mission and flight systems”, ***IEEE Aerospace Conference***, vol.7, 459 - 463 (2000).

26. Rabaey, J., Ammer, M., da Silva, J., “PicoRadio supports ad hoc ultra-low power wireless networking”, ***IEEE Computer Magazine***, **Volume:** 33(7) 42 - 48 (2000).

27. Perrig, A., Szewczyk, R., Wen, V., Culler, D., Tygar, J., “SPINS: security protocols for sensor networks”, ***Wireless Networks,*** Vol. 8, no. 5, 521-534. (2002).

28. Shih, E., Cho, Ickes, S., Min, R., Sinha, A., Wang, A., “Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks”, ***Proceedings of ACM MobiCom’01***, DOI=10.1145/381677.381703, 272-287 (2001).

29. Li L., Halpern, Y., “Minimum-energy mobile wireless networks revisited”, ***IEEE International Conference on Communications ICC’2001***, vol.1:278-283 (2001)

30. Yoneki, E. and Bacon, J., “A survey of Wireless Sensor Network technologies: research trends and middleware’s role”, ***Technical Report-University of Cambridge/Computer Laboratory***, ***University of Cambridge*** 10-11 (2005).

31. Li, Y., My, T., Wu, T., “Wireless Sensor Networks and Applications”, ***springer series on signals and communication technology, pub. springer***, 113-140*(* 2008).

32. Intanagonwiwat, C., Govindan, R. And Estrin, D., “Directed diffusion: a scalable and robust communication paradigm for sensor networks”, ***Proceedings of the ACM Mobi- Com’00***, DOI=10.1145/345910.345920 (2000).

33. Meguerdichian, S., Koushanfar, F., Qu, G. And Potkonjak, M., “Exposure in wireless ad-hoc sensor networks”, ***MobiCom '01 Proceedings of the 7th annual international conference on Mobile computing and networking*** 139-150 (2001).

34. Woo, A., Culler, D., “A transmission control scheme for media access in sensor networks, ***MobiCom '01 Proceedings of the 7th annual international conference on Mobile computing and networking***, DOI=10.1145/381677.381699, 221-235 (2001).

35. Min, R., Furrer, T. And Chandrakasan, A., “Dynamic voltage scaling techniques for distributed microsensor networks”, ***VLSI, 2000. Proceedings. IEEE Computer Society Workshop on***, 11: 43 - 46 (2002).

36. Akyildiz, F., Su, W., Sankarasubramaniam, Y., and Cayirci, E., “A survey on sensor networks”. ***IEEE Communications Magazine***, 40(8):102–114, 2002.

37. Avancha, S., “A Holistic Approach to Secure Sensor Networks”, PhD thesis, ***Faculty of the Graduate School of the University of Maryland***, ***University of Maryland***, Baltimore County, 31-32 (2005).

38. Karlof, C., Sastry, N., and Wagner, D., “Tinysec: a link layer security architecture for wireless sensor networks”. In ***Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)***, 162-175 (2004).

39. Wood, D. and Stankovic, J., “Denial of service in sensor networks. Computer”, ***IEEE Computer Society Press Los Alamitos***, 35(10):54-62, (2002).

40. Cardenas, A., Radosavac, S., and Baras, J., “Detection and prevention of mac layer misbehavior for ad hoc networks”. ***In The Second ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)***, 17-22 (2004).

41. Kyasanur, P. And Vaidya, N., “Detection and handling of mac layer misbehavior in wireless networks”. ***In International Conference on Dependable Systems and Networks (DSN)***, 173 - 182 *(*2003).

42. M. Raya, J. Hubaux, and And Aad, I. “Domino: A system to detect greedy behavior in IEEE 802.11 hotspots”. ***In International Conference On Mobile Systems, Applications And Services (MobiSys),*** *5(12):* 1691-1705(2004).

43. Karlof, C., Wagner, D., “Secure routing in wireless sensor networks: Attacks and countermeasures”. ***In Proceedings of the 1st IEEE International Workshop on Sensor Network Protocols and Applications***,1(2-3): 113-127 (2003).

44. Capkun, S. And Hubaux, J., “Secure positioning in sensor networks”, ***EPFL/IC/200444***, ***Swiss Federal Institute of Technology Lausanne (EPFL)***, 6-8 (2004).

45. Lazos, L. And Pooverdran, R.., “Serloc: Secure range-independent localization for wireless sensor networks”. ***In Workshop on Information Systems and Economics (WiSe)***, 21-30 (2004).

46. Wagner, D., “Resilient aggregation in sensor networks”. ***In The Second ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)***, 78-87 (2004).

47. Blackert, W., J., Gregg, D., M., Castner, A., K., Kvle, E., M., Horn, R., L., and Jokerst, R., M., “Analyzing interaction between distributed denial of service attacks and mitigation technologies”, ***In Proceedings of DARPA Information Survivability Conference and Exposition***, 1: 26-36 (2003).

48. Zia, T., Zomaya, A., “Security Issues in Wireless Sensor Networks”, ***ICSNC '06 Proceedings of the International Conference on Systems and Networks Communication***, 40-44 (2006).

49. Wang, T., and Schulzrinne, H., “An IP traceback mechanism for reflective DoS attacks”. **In *Canadian Conference on Electrical and Computer Engineering***, 2: 901-904 (2004).

50. Culpepper, J., and Tsent, H., “Sinkhole intrusion indicators in DSR MANETs”, **In *Proceedings of First International Conference on Broadband Networks***, 681-688 (2004).

51. Douceur, J., “The sybil attack”. ***In The First International Workshop on PeertoPeer Systems***, 251-25 *(*2002).

52. Newsome, J., Shi, E., Song, D., and Perrig, A., “The Sybil attack in sensor networks: analysis and defenses”. **In *Proceedings of the Third International Symposium on Information Processing in Sensor Networks ACM***, 259-268 (2004).

53. Chris Tseng, H., Jack Culpepper, B., “Sinkhole intrusion in mobile ad hoc networks: The problem and some detection indicators” ***Computer and Security***, 24(7): 561-570 (2005).

54. Hu, Y., C., Perrig, A., and Johnson, D., B., “Packet leashes: a defense against wormhole attacks in wireless networks” **In *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM)***, 3: 1976-1986 (2003).

55. Yong Wang, Attebury, G., Ramamurthy, B., “A survey of security issues in wireless sensor networks”, ***Communications Surveys & Tutorials***, 8(2): 2-23 (2006).

56. Zhou, L., and Haas, Z., J., “Securing ad hoc networks”. ***IEEE Network***, 13(6):24-30, 1999.

57. Hubaux, J., Buttyan, L., and Capkun, S., “The quest for security in mobile ad hoc networks”. In ***Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC 2001),*** 146-155 (2001).

58. Simplício, A., Barreto, P., Margi, C., Carvalho, T., “A survey on key management mechanismsnext term for previous termdistributed Wireless Sensor Networks”, ***Computer Networks***, 54(15): 2591-2612 (2010).

59. Zapata, M. G., “Secure ad-hoc on-demand distance vector (AODV) routing” ***A CM SIGMOBILE Mobile Computing and Communications Review***, 6(3): 106 107, 2002.

60. Perrig, A., Stankovic, J., and Wagner, D., “Security in wireless sensor networks”, ***Communications of the ACM***, 47(6): 53-57 (2004).

61. Menezes, A.,J., van Oorschot, P.,C., Vanstone, S.,A., “Handbook of Applied Cryptography, 978-0849385230” 1st ed, Boca Raton, FL, ***CRC-Press***, New York, NY, USA, 385-387 (1996).

62. Merwe, J., Dawoud, D., McDonald, S., "A Survey on Peer-to-Peer Key Management for Mobile Ad Hoc Networks", ***ACM Computing Surveys (CSUR,)***, 39 (1): 1-45 (2007).

63. Du, W., Deng, J., Han, Y.,S., and Varshney, P.,K., (2003) “A pairwise key predistribution scheme for wireless sensor networks”, ***In Proceedings of the ACM CCS,*** 8(2): 228-258, DOI=10.1145/1065545.1065548 (2003).

64. Du, W., Deng, J., Han, Y., Chen, S., and Varshney, P., K., (2004) “A key management scheme for wireless sensor networks using deployment knowledge” ***Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies,*** 1: 586-597 (2004).

65. Neuman, B., C., and Tso, T., “Kerberos: an authentication service for computer networks”, ***IEEE Communications Magazine,*** 32(9): 33-38 (1994).

66. Diffie, W., and Hellman, M., (1976) “New directions in cryptography”, ***IEEE Transactions on Information Theory***, 22(6): 644-654 (1976).

67. Rivest, L., Shamir, A., and Adleman, M., “A method for obtaining digital signatures and public key cryptosystems”. ***Communications of the ACM,*** 21 (2): 120-126 (1978).

68. Chan, H., Perrig A., and Song, D., “key distribution techniques for sensor networks”, ***Carnegie Mellon University, Kluwer Academic Publishers,*** 277-303 (2004).

69. Zhang, J., Varadharajan, V., “Wireless sensor network key management survey and taxonomy”, ***Journal of Network and Computer Applications***, 33(2): 63-75 (2010).

70. Malan, M., Welsh, M., D., Smith, M., “A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography”, ***in: Proceedings of 1st IEEE International Conference Communications and Networks (SECON)***, 71-80 (2004).

71. Gura, N., Patel, A., Wander, A., Eberle, H., And Shantz, C., “Comparing elliptic curve cryptography and RSA on 8-bit CPUs”, ***in: Proceedings of the 6th International Workshop on Cryptographic Hardware and Embedded Systems***, 119-132 (2004).

72. Wander, A., S., Gura, N., Gupta, V., Shantz, C., “Energy analysis of public-key cryptography for wireless sensor networks”, ***in: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications (PERCOM)***, 327-332 (2005).

73. Du, X., Guizani, M., Xiao, Y., Ci, S., Chen, H., “A routing-driven elliptic curve cryptography based key management scheme for heterogeneous sensor networks”, ***in: IEEE Transactions on Wireless Communications,*** 8(3): 1223-1229 (2009).

74. Eschenauer, L., Gligor, V.,D., “A key management scheme for distributed sensor networks”, ***in: Proceedings of the 9th ACM Conference on Computer and Communication Security***. 41-47, November 2002.

75. Liu, D., Ning, P., “Establishing pairwise keys in distributed sensor networks”, ***Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS ’03,)*** 52–61 (2003).

76. Chan, H., Perrig, A., Song, D., “Random key predistribution schemes for sensor networks”, ***in: Proceedings of the 2003 IEEE Symposium on Security and Privacy***, ,197– 213 (2003).

77. Ning, P., Li, R., Liu, D., “Establishing pairwise keys in distributed sensor networks”, ACM Transactions on Information and System Security, 8 (1): 41-77 (2005).

78. Anjum, F., “Location dependent key management using random keypredistribution in sensor networks ***Communication Systems Software and Middleware, 2007. COMSWARE 2007. 2nd International Conference***. 21-30 (2007).

79. Younis, M., Ghumman, K., Eltoweissy, M., “Location-aware combinatorial key management scheme for clustered sensor networks”, IEEE Transactions on Parallel and Distributed Systems, 17 (8): 865–882 (2006).

80. Eltoweissy, M., Moharrum, M. And Mukkamala, R., “Dynamic key management in sensor networks”, ***IEEE Communications Magazine,*** 44 (4): 122–130 (2006).

81. Zhu, S., Setia, S., Jajodia, S., “LEAP: efficient security mechanisms for large-scale distributed sensor networks”, ***in: Proceedings of The 10th ACM Conference on Computer and Communications Security (CCS ’03),*** 62-72 (2003).

82. Du, X., Xiao, Y., Guizani, M., Chen, H., "An Effective Key Management Scheme for Heterogeneous Sensor Networks", ***Ad Hoc Networks, Elsevier***, 5(1): 24–34 (2007) .

83. Dulman, S., Havinga, P., "A Simulation Template for Wireless Sensor Networks", ***IEEE/ACM Transactions of Networking***, 12(4), (2004).

84. Naicken, S., Livingston, B., Basu, A., Rodhetbhai, S., Wakeman, I., And Chalmers, D., "The state of peer-to-peer simulators and simulations", *SIGCOMM Comput,* 37(2) : 95-98 (2007).

85. Xian, X., , Shi, W., , Huang, H., "Comparison of OMNET++ and other simulator for WSN simulation", ***Industrial Electronics and Applications, 2008. ICIEA 2008. 3rd IEEE Conference***, 1439 – 1443 (2008).

86. İnternet : Objective Modular Network Test-bed in C++ "What is OMNeT++?" http://www.omnetpp.org.

87. Vargo, A., Hornig R., "An Overview Of The Omnet++ Simulation Environment", ISBN 978-963-9799-20-2, (2008).

88. Wang, S., Liu, K.Z., Hu, F.P., "Simulation of Wireless Sensor Networks Localization with OMNeT", ***Mobile Technology, Applications and Systems, 2005 2nd International Conference,*** 1-6 (2005).

89. Drytkiewicz, W., Sroka, S., Handziski, V., Köpke, A. And Karl, H., "A Mobility Framework for OMNeT++", January 22, 2003.

EKLER

EK-1. Tek anahtar yönteminde 50 düğümden oluşan ağ için düğüm dağıtımı

```
package key1test.net;

import key1test.col.Sink;
import key1test.col.Node;


// WSN Deployment
network Net
{
    parameters:


@display("bgb=1000,750;bgi=background/terrain;bgg=100,2,grey95;bgs=0
.75,m");
    submodules:
        node1: Node;
        node2: Node;
        node3: Node;
        node4: Node;
        node5: Node;
        node6: Node;
        node7: Node;
        node8: Node;
        node9: Node;
        node10: Node;
        node11: Node;
        node12: Node;
        node13: Node;
        node14: Node;
        node15: Node;
        node16: Node;
        node17: Node;
        node18: Node;
        node19: Node;
        node20: Node;
        node21: Node;
        node22: Node;
        node23: Node;
        node24: Node;
        node25: Node;
        node26: Node;
        node27: Node;
        node28: Node;
        node29: Node;
        node30: Node;
        node31: Node;
        node32: Node;
        node33: Node;
        node34: Node;
        node35: Node {
```

EK-1 (Devam). Tek anahtar yönteminde 50 düğümden oluşan ağ için düğüm dağıtımı

```
        @display("p=409.33334,188");
    }
    node36: Node;
    node37: Node;

    node39: Node;
    node40: Node;
    node41: Node;
    node42: Node;
    node43: Node;
    node44: Node;
    node45: Node;
    node46: Node;
    node47: Node;
    node48: Node;
    node49: Node;
    node50: Node;
    node38: Node {
        @display("p=950,700");
    }
    sink: Sink {
        @display("p=34.666668,40");
    }

}
```

EK-2. Tek anahtar yönteminde bir düğümün tanımlaması (.ned) dosyası

```
package key1test.col;

simple Node
{
       parameters:
           int  key =3;
        @display("i=abstract/router_vs");
    gates:
        input in @directIn;

}
```

EK-3. Tek anahtar yönteminde düğümün basit modülü

```
#ifndef __KEY1TEST_NODE_H_
#define __KEY1TEST_NODE_H_

#include <omnetpp.h>
#include "time.h"

class Node : public cSimpleModule
{

      char *pp[13];
      cMessage *smsg;
      cModule *des;
      int key;
  protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
    // encryption  Function Prototype
    void encmsg(char str[]);
    // Decryption  Function Prototype
    void decmsg(char str[]);
  public:
    //SimTime tm;
    //simtime_t tm1;
    clock_t start, end;
    double cpu_time_used;

};

#endif
```

EK-4. Tek anahtar yönteminde düğümün C++ dosyası

```
#include "node.h"
#include <iostream>
#include <string>
#include <cstring>
#include "time.h"
Define_Module(Node);

void Node::initialize()
{
      pp[1]="node38";
      pp[2]="node39";
      pp[3]="node28";
      pp[4]="node49";
      pp[5]="node47";
      pp[6]="node41";
      pp[7]="node15";
      pp[8]="node3";
      pp[9]="node22";
      pp[10]="node12";
      pp[11]="node6";
      pp[12]="sink";
      key=par("key");
      smsg = new cMessage();
      scheduleAt(simTime(),smsg);

}

void Node::handleMessage(cMessage *msg)
{
      // Start Node

   if((this==simulation.getModuleByPath(pp[1])) && (msg-
>isSelfMessage()))
   {
         cModule *st;
         st=simulation.getModuleByPath(pp[1]);
         start= clock();
       end  = clock();
         cpu_time_used = ((double) (end - start))/CLOCKS_PER_SEC; ;

ev<<"************************************************************
***************"<<'\n';
       ev<<"Time = "<<cpu_time_used<<'\n';
       ev<<"Memory Usage= "<<sizeof(key)<<" Byte"<<'\n';
       ev<<"Vulnerability if a Node (or any number of Nodes
Captured) =  "<<50<<'\n';
       ev<<"Connection Required To Distribute The Key =  "<<0<<'\n';
       ev<<"Connectivity for each node = "<<100<<"%"<<'\n';

ev<<"************************************************************
***************"<<'\n';
```

EK-4 (Devam). Tek anahtar yönteminde düğümün C++ dosyası

```
            cancelAndDelete(msg);
             //Embedding the Message
             cMessage *msg=new cMessage("Encripted_message");
             msg->addPar("count");
             msg->par("count").setLongValue(11);
             msg->addPar("text");
             char str[]="computer";
             encmsg(str);
             ev<<"Message ="<<str;
             msg->par("text").setStringValue(str);
             //Getting the Next Node in path
             des = simulation.getModuleByPath(pp[2]);
             //Sending the Message
             sendDirect(msg,0,0,des->gate("in"));
}
else
{
      if(not(msg->isSelfMessage()))
      {
            long temp=msg->par("count").longValue();
            temp--;
          // if it is the End Node
          if(temp==0)
          {
            const char *str=msg->par("text").stringValue();
                char dstr[30];
                strcpy(dstr,str);
                decmsg(dstr);
                  ev<<dstr;
           }

      else
      { // if it is an intermediate Node
            /*
             *
           long temp=msg->par("count").longValue();
               temp--;
               cMessage *nmsg=new cMessage("Encripted_message");
               nmsg->addPar("count");
              nmsg->par("count").setLongValue(temp);
            *
            */

              cMessage *copy=msg->dup();
              cancelAndDelete(msg);
              copy->par("count").setLongValue(temp);
              //Getting the Next Node in path
                des = simulation.getModuleByPath(pp[12-temp+1]);
                //Sending the Message
                sendDirect(copy,0,0,des->gate("in"));
      }
```

EK-4 (Devam). Tek anahtar yönteminde düğümün C++ dosyası

```
     }
   }
}
// Function to Encrypt the Message
void Node::encmsg(char str[])
{
      unsigned int  i;
      for( i=0;i< strlen(str);i++)
      {
            str[i]=  str[i]+key % 26;
      }
}
// Function to uncipher the Encrypted Message
void Node::decmsg( char  str[])
{

      unsigned int  i;
      for( i=0;i< strlen(str);i++)
      {
            str[i]= str[i]-key % 26;
      }
}
```

EK-5. Tek anahtar yönteminde ağda baz istasyonu tanımlaması

```
package key1test.col;


simple Sink
{
     int  key =3;
    @display("i=block/sink_l");
        gates:
        input in @directIn;
```

EK-6. Tek anahtar yönteminde baz istasyonu için basit modül

```
#ifndef __KEY1TEST_SINK_H_
#define __KEY1TEST_SINK_H_

#include <omnetpp.h>

class Sink : public cSimpleModule
{
      cMessage *smsg;
      int key;
  protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);

    // Decryption  Function Prototype
    void decmsg(char  str[]);
};

#endif
```

EK-7. Tek anahtar yönteminde baz istasyonu için C++ dosyası

```
#include "sink.h"
#include "string.h"

// Sink Member Functions
Define_Module(Sink);

void Sink::initialize()
{
    key=par("key");
}

void Sink::handleMessage(cMessage *msg)
{

    const char *str=msg->par("text").stringValue();
    char dstr[30];
    strcpy(dstr,str);
    decmsg(dstr);
      ev<<dstr;

}
// Function to uncipher the Encrypted Message
void Sink::decmsg( char  str[])
{;
      unsigned int  i;
      for( i=0;i< strlen(str);i++)
      {
            str[i]= str[i]-key % 26;
      }
}
```

EK-8. İkili anahtar yönteminde 50 düğümden oluşan ağ için düğüm dağıtımı

```
package key2test.net;

import key2test.col.Node;
import key2test.col.Sink;

// WSN Deployment
network Net
{
    parameters:


@display("bgb=1000,750;bgi=background/terrain;bgg=100,2,grey95;bgs=0
.75,m");

    submodules:
        node1: Node
        {
            id=1;
            key=14;
        }
        node2: Node
        {
            id=2;
            key=5;
        }

        node3: Node
        {
            id=3;
            key=13;
        }
        node4: Node
        {
            id=4;
            key=76;
        }
        node5: Node
        {
            id=5;
            key=62;
        }
        node6: Node
        {
            id=6;
            key=19;
        }
        node7: Node
        {
            id=7;
            key=20;
        }
```

EK-8 (Devam). İkili anahtar yönteminde 50 düğümden oluşan ağ için düğüm dağıtımı

```
node8: Node
{
    id=8;
    key=8;
}
node9: Node
{
    id=9;
    key=50;
}
node10: Node
{
    id=10;
    key=7;
}

node11: Node
{
    id=11;
    key=25;
}
node12: Node
{
    id=12;
    key=55;
}
node13: Node
{
    id=13;
    key=32;
}
node14: Node
{
    id=14;
    key=1;
}
node15: Node
{
    id=15;
    key=36;
}
node16: Node
{
    id=16;
    key=22;
}
node17: Node
{
    id=17;
    key=42;
```

EK-8 (Devam). İkili anahtar yönteminde 50 düğümden oluşan ağ için düğüm dağıtımı

```
}
node18: Node
{
    id=18;

    key=16;
}
node19: Node
{
    id=19;
    key=71;
}
node20: Node
{
    id=20;
    key=10;
}
node21: Node
{
    id=21;
    key=18;
}
node22: Node
{
    id=22;
    key=64;
}
node23: Node
{
    id=32;
    key=11;
}
node24: Node
{
    id=24;
    key=25;
}
node25: Node
{
    id=25;
    key=47;
}
node26: Node
{
    id=26;
    key=23;
}
node27: Node
{
    id=27;
    key=55;
```

EK-8 (Devam). İkili anahtar yönteminde 50 düğümden oluşan ağ için düğüm dağıtımı

```
}
node28: Node
{

    id=28;
    key=72;
}
node29: Node

{
    id=29;
    key=9;
}
node30: Node
{
    id=30;
    key=4;
}
node31: Node
{
    id=31;
    key=49;
}
node32: Node
{
    id=32;
    key=87;
}
node33: Node
{
    id=33;
    key=17;
}
node34: Node
{
    id=34;
    key=29;
}
node35: Node {
    @display("p=409.33334,188");
    id=35;
    key=73;
}
node36: Node
{
    id=36;
    key=59;
}
node37: Node
{
    id=37;
```

EK-8 (Devam). İkili anahtar yönteminde 50 düğümden oluşan ağ için düğüm dağıtımı

```
    key=27;
}

node39: Node
{
    id=39;
    key=15;
}
node40: Node
{
    id=40;

    key=6;
}
node41: Node
{
    id=41;
    key=51;
}
node42: Node
{
    id=42;
    key=44;
}
node43: Node
{
    id=43;
    key=24;
}
node44: Node
{
    id=44;
    key=53;
}
node45: Node
{
    id=45;
    key=21;
}
node46: Node
{
    id=46;
    key=2;
}
node47: Node
{
    id=47;
    key=78;
}
node48: Node
{
```

EK-8 (Devam). İkili anahtar yönteminde 50 düğümden oluşan ağ için düğüm dağıtımı

```
            id=48;
            key=65;
        }
        node49: Node
        {
            id=49;
            key=81;
        }
        node50: Node
        {
            id=50;
            key=77;
        }
        node38: Node {

            @display("p=950,700");
            id=38;
            key=3;
        }
         sink: Sink {
            @display("p=34.666668,40");
        }
}
```

EK-9. İkili anahtar yönteminde bir düğümün tanımlaması (.ned) dosyası

```
package key2test.col;

simple Node
{
     int  key;
     int id;
        @display("i=abstract/router_vs");
    gates:
        input in @directIn;
}
```

EK-10. İkili anahtar yönteminde düğümün basit modülü

```
#ifndef __KEY2TEST_NODE_H_
#define __KEY2TEST_NODE_H_

#include <omnetpp.h>
#include "time.h"
/**
 * TODO - Generated class
 */
class Node : public cSimpleModule
{
      cMessage *smsg;
      cModule *des;
      //Particular Node Key
      int key;
      long id;
      // Internal Node Memory Which Contains Keys
      int key_mem[50];
      char *p[100];
      char *pp[13];
  protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
    // encryption  Function Prototype
    void encmsg(char str[],int key);
    // Decryption  Function Prototype
    void decmsg(char str[],int key);
  public:
    clock_t start, end;
    double cpu_time_used;
    int ccn;

};

#endif
```

EK-11. İkili anahtar yönteminde düğümün C++ dosyası

```
#include "node.h"
#include "string.h"
#include "time.h"
Define_Module(Node);

void Node::initialize()
{
      id=par("id");
      key=par("key");

      cMessage *msg=new cMessage();

      p[1]="node1";p[2]="node2";p[3]="node3";p[4]="node4";p[5]="node
5";
      p[6]="node6";p[7]="node7";p[8]="node8";p[9]="node9";p[10]="nod
e10";
      p[11]="node11";p[12]="node12";p[13]="node13";p[14]="node14";p[
15]="node15";
      p[16]="node16";p[17]="node17";p[18]="node18";p[19]="node19";p[
20]="node20";
      p[21]="node21";p[22]="node22";p[23]="node23";p[24]="node24";p[
25]="node25";
      p[26]="node26";p[27]="node27";p[28]="node28";p[29]="node29";p[
30]="node30";
      p[31]="node31";p[32]="node32";p[33]="node33";p[34]="node34";p[
35]="node35";
      p[36]="node36";p[37]="node37";p[38]="node38";p[39]="node39";p[
40]="node40";
      p[41]="node41";p[42]="node42";p[43]="node43";p[44]="node44";p[
45]="node45";
      p[46]="node46";p[47]="node47";p[48]="node48";p[49]="node49";p[
50]="node50";
      p[51]="sink";pp[1]="node38";pp[2]="node39";pp[3]="node28";pp[4
]="node49";
      pp[5]="node47";pp[6]="node41";pp[7]="node15";pp[8]="node3";pp[
9]="node22";
      pp[10]="node12";pp[11]="node6";pp[12]="sink";

      msg->addPar("isp");
      msg->par("isp").setBoolValue(false);

      scheduleAt(simTime()+0.5,msg);
      ccn=0;
}

void Node::handleMessage(cMessage *msg)
{

    if(msg->isSelfMessage())
    {
      if(not(msg->par("isp").boolValue()))
```

EK-11 (Devam). İkili anahtar yönteminde düğümün C++ dosyası

```
      {   // if it Key Distribution Self-Message
            //Key Distribution
            if(this==simulation.getModuleByPath(p[1]))
            {
                  start =  clock();
            }
            for(int i=1;i<=50;i++)
            {
                  //If it is Not The Current Node
                  if(i !=id)
                  {
                        //Embedding the Key
                        cMessage *kmsg=new
cMessage("Key_Distribution");
                     kmsg->addPar("keyfrm");
                     kmsg->par("keyfrm").setLongValue(key);
                     kmsg->addPar("nid");
                     kmsg->par("nid").setLongValue(id);
                     kmsg->addPar("iskeymsg");
                     kmsg->par("iskeymsg").setBoolValue(true);
                       des = simulation.getModuleByPath(p[i]);

if(this==simulation.getModuleByPath(p[1])&&(i==38))
                                 {
                                       kmsg->addPar("start");
                                       kmsg-
>par("start").setDoubleValue(start);
                                 }
                     sendDirect(kmsg,0,0,des->gate("in"));
                     ccn=ccn+1;
               }
          }
            //Embedding The Key Message to be Sent to Sink
            cMessage *kmsg=new cMessage("Key_Distribution");
          kmsg->addPar("keyfrm");
          kmsg->par("keyfrm").setLongValue(key);
          kmsg->addPar("nid");
          kmsg->par("nid").setLongValue(id);
          kmsg->addPar("iskeymsg");
          kmsg->par("iskeymsg").setBoolValue(true);
          des = simulation.getModuleByPath(p[51]);

          sendDirect(kmsg,0,0,des->gate("in"));
          key_mem[id]=key;

            if(this==simulation.getModuleByPath(p[38]))

            {
                  //If it Sending Message Start Node
                  cMessage *msg=new cMessage();
                  msg->addPar("isp");
```

EK-11 (Devam). İkili anahtar yönteminde düğümün C++ dosyası

```
                  //To indicate it's Not Key Distribution Self-
message
                msg->par("isp").setBoolValue(true);
                scheduleAt(simTime()+0.5,msg);

            }

      }
      else
      {
            cModule *st;
            st=simulation.getModuleByPath(p[1]);
            end  = clock();
            cpu_time_used = ((double) (end - start))/CLOCKS_PER_SEC;
;

ev<<"**************************************************************
**************************************************************"<<
'\n';
            ev<<"Time = "<<cpu_time_used<<'\n';
          ev<<"Memory Usage for single node= "<<sizeof(key_mem)<<"
Byte"<<'\n';
          ev<<"Total Memory Usage in the Network=
"<<(sizeof(key_mem))*50<<" Byte"<<'\n';
          ev<<"Vulnerability if a Node (or any number of Nodes
Captured) =  "<<50<<" All Nodes It Communicate With them
Directly"<<'\n';
          ev<<"Total Connection Required To Distribute The Key =
"<<50*ccn<<'\n';
          ev<<"Connectivity for each node = "<<100<<"%"<<'\n';

ev<<"**************************************************************
**************************************************************"<<
'\n';
            //if it Self-message to start sending the message
            cancelAndDelete(msg);
            //Embedding the Message
          cMessage *msg=new cMessage("Encripted_message");
            msg->addPar("count");
            msg->par("count").setLongValue(11);
            msg->addPar("text");
            char str[]="computer";
            encmsg(str,key_mem[id]);
            ev<<"Message = "<<str;
            msg->par("text").setStringValue(str);
            msg->addPar("iskeymsg");
            msg->par("iskeymsg").setBoolValue(false);
            msg->addPar("sid");
            msg->par("sid").setLongValue(id);
            //Getting the Next Node in path
            des = simulation.getModuleByPath(pp[2]);
            //Sending the Message
```

EK-11 (Devam). İkili anahtar yönteminde düğümün C++ dosyası

```
            sendDirect(msg,0,0,des->gate("in"));
      }

    }//if(msg->isSelfMessage())
    else //if it not self-Message
    {
      if(not(msg->isSelfMessage()))
      {
            if(msg->par("iskeymsg").boolValue())
            {
                        if((msg-
>getSenderModule()==simulation.getModuleByPath(p[1]))&&(this==simula
tion.getModuleByPath(p[38])))
                        {
                           start=msg->par("start");
                        }
                  //Entering The Received Key in the Corresponding
                  //Location of sender node in the internal Memory
                  //of Current Node
                  long index=msg->par("nid").longValue();
                    key_mem[index]=msg->par("keyfrm").longValue();
            }
            else
            { //if it is normal message
                  long temp=msg->par("count").longValue();
                      temp--;
                      if(temp==0) // if it's the End Node
                        {
                           const char *str=msg-
>par("text").stringValue();
                                 char dstr[30];
                                 strcpy(dstr,str);
                                 long sid=msg->par("sid");
                                 decmsg(dstr,key_mem[sid]);
                                   ev<<dstr;
                        }
                       else // if it is an intermediate Node
                      {
                         cMessage *copy=msg->dup();
                             cancelAndDelete(msg);
                             copy->par("count").setLongValue(temp);
                           //Getting the Next Node in path
                    des = simulation.getModuleByPath(pp[12-temp+1]);
                    //Sending the Message
                              sendDirect(copy,0,0,des->gate("in"));
                      }
            }
      }
    }
}

// Function to Encrypt the Message
```

EK-11 (Devam). İkili anahtar yönteminde düğümün C++ dosyası

```
void Node::encmsg(char str[],int key)
{
      unsigned int  i;
      for( i=0;i< strlen(str);i++)
      {

            str[i]=  str[i]+ key % 26;
      }
}

// Function to uncipher the Encrypted Message
void Node::decmsg( char  str[],int key)
{
      unsigned int  i;
      for( i=0;i< strlen(str);i++)
      {
            str[i]= str[i]-key % 26;
      }
}
```

EK-12. İkili anahtar yönteminde ağda baz istasyonu tanımlaması

```
package key2test.col;

simple Sink
{
     @display("i=block/sink_l");
        gates:
        input in @directIn;
}
```

EK-13. Tek anahtar yönteminde baz istasyonu için basit modülü

```
#ifndef __KEY2TEST_SINK_H_
#define __KEY2TEST_SINK_H_

#include <omnetpp.h>

class Sink : public cSimpleModule
{
      long key_mem[50];
  protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
    // Decryption  Function Prototype
    void decmsg(char str[],int key);
};

#endif
```

EK-14. Tek anahtar yönteminde baz istasyonu için C++ dosyası

```
#include "Sink.h"

Define_Module(Sink);

// Sink Member Functions
void Sink::initialize()
{
}

void Sink::handleMessage(cMessage *msg)
{

              if(msg->par("iskeymsg").boolValue())
              {
                  long index=msg->par("nid").longValue();
                  key_mem[index]=msg->par("keyfrm").longValue();

              }
              else
              { // if it'se Normal Message
                    const char *str=msg->par("text").stringValue();
                    char dstr[30];
                    strcpy(dstr,str);
                    long sid=msg->par("sid");
                    decmsg(dstr,key_mem[sid]);
                    ev<<dstr;
              }
}
// Function to uncipher the Encrypted Message
void Sink::decmsg( char  str[],int key)
{
      unsigned int  i;
      for( i=0;i< strlen(str);i++)
      {
            str[i]= str[i]-key % 26;
      }
}
```

EK-15. Temel olasılıksak yönteminde 50 düğümden oluşan ağ için düğüm dağıtımı

```
package key3test.net;

import key3test.col.Node;
import key3test.col.Sink;

// WSN Deployment
network Net
{


@display("bgb=1000,750;bgi=background/terrain;bgg=100,2,grey95;bgs=0
.75,m");
    submodules:
       node1: Node;
        node2: Node;
        node3: Node;
        node4: Node;
        node5: Node;
        node6: Node;
        node7: Node;
        node8: Node;
        node9: Node;
        node10: Node;
        node11: Node;
        node12: Node;
        node13: Node;
        node14: Node;
        node15: Node;
        node16: Node;
        node17: Node;
        node18: Node;
        node19: Node;
        node20: Node;
        node21: Node;
        node22: Node;
        node23: Node;
        node24: Node;
        node25: Node;
        node26: Node;
        node27: Node;
        node28: Node;
        node29: Node;
        node30: Node;
        node31: Node;
        node32: Node;
        node33: Node;
        node34: Node;
        node35: Node {
```

EK-15 (Devam). Temel olasılıksak yönteminde 50 düğümden oluşan ağ için düğüm dağıtımı

```
            @display("p=409.33334,188");
        }
        node36: Node;
        node37: Node;
        node39: Node;
        node40: Node;
        node41: Node;
        node42: Node;
        node43: Node;
        node44: Node;
        node45: Node;
        node46: Node;
        node47: Node;
        node48: Node;
        node49: Node;
        node50: Node;
        node38: Node {
            @display("p=950,700");
        }
                 sink: Sink {
            @display("p=34.666668,40");
        }

}
```

EK-16. Temel olasılıksal yönteminde bir düğümün tanımlaması (.ned) dosyası

```
package key3test.col;

simple Node
{
        @display("i=abstract/router_vs");
    gates:
        input in @directIn;
}
```

EK-17. Temel olasılıksal yönteminde düğümün basit modülü

```
#ifndef __KEY3TEST_NODE_H_
#define __KEY3TEST_NODE_H_

#include <omnetpp.h>
#include "time.h"

class Node : public cSimpleModule
{
      //Key Ring of Length 3
      int key[3];
      //ID For Each Corresponding Key in the Key Ring
      int kid[3];
      //Particular Node ID
      int id;
      char *p[100];
      char *pp[13];
      int cancom[50];
      int index;
      cModule *des;
      float prob;

  protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
    // encryption  Function Prototype
    void encmsg(char str[],int key);
    // Decryption  Function Prototype
    void decmsg(char str[],int key);
  public:
    clock_t start, end;
    double cpu_time_used;
    int ccn;
};

#endif
```

EK-18. Temel olasılıksal yönteminde düğümün C++ dosyası

```
#include "node.h"
#include "time.h"
Define_Module(Node);

void Node::initialize()
{

      int size=100;
      //Pool of size 100
      int pool[size];
      int temp;
      prob=0.9;
      long rem=((1-prob)*(size/10));

    for(int i=1;i<=size;i++)
    {
      pool[i]=i+1;

    }

    for(int j=1;j<=3;j++)
    {
      temp=intrand(rem)+1;
        /*  if(temp==0)
             {
                 temp=intrand(3);
             }
        */
          key[j]=pool[temp];
          kid[j]=temp;
    }
    //Random Selection of keys

    for(int k=1;k<=3;k++)
    {
      for(int h=1;h<=3;h++)
      {
            if(k!=h)
            {
                  if(key[k]==key[h])
                  {
                        temp=intrand(rem)+1;
                        key[h]=pool[temp];
                        kid[h]=temp;

                  }
            }
```

EK-18 (Devam). Temel olasılıksal yönteminde düğümün C++ dosyası

```
      }
    }

p[1]="node1";p[2]="node2";p[3]="node3";p[4]="node4";p[5]="node5";

p[6]="node6";p[7]="node7";p[8]="node8";p[9]="node9";p[10]="node10";

p[11]="node11";p[12]="node12";p[13]="node13";p[14]="node14";p[15]="n
ode15";

p[16]="node16";p[17]="node17";p[18]="node18";p[19]="node19";p[20]="n
ode20";

p[21]="node21";p[22]="node22";p[23]="node23";p[24]="node24";p[25]="n
ode25";

p[26]="node26";p[27]="node27";p[28]="node28";p[29]="node29";p[30]="n
ode30";

p[31]="node31";p[32]="node32";p[33]="node33";p[34]="node34";p[35]="n
ode35";

p[36]="node36";p[37]="node37";p[38]="node38";p[39]="node39";p[40]="n
ode40";

p[41]="node41";p[42]="node42";p[43]="node43";p[44]="node44";p[45]="n
ode45";

p[46]="node46";p[47]="node47";p[48]="node48";p[49]="node49";p[50]="n
ode50";

p[51]="sink";pp[1]="node38";pp[2]="node39";pp[3]="node28";pp[4]="nod
e49";
      pp[5]="node47";pp[6]="node41";pp[7]="node15";pp[8]="node3";pp[
9]="node22";
      pp[10]="node12";pp[11]="node6";pp[12]="sink";
    index=1;
    //Getting the Particular Node ID
    id=getId();
    cMessage *msg=new cMessage();
    msg->addPar("isp");
    msg->par("isp").setBoolValue(false);
    ccn=0;
    scheduleAt(simTime()+0.5,msg);

}

void Node::handleMessage(cMessage *msg)
{
      //ev<<id<<"   "<<getId();
      /*
```

EK-18 (Devam). Temel olasılıksal yönteminde düğümün C++ dosyası

```
      ev<<"node="<<id<<"  ";
      for(int i=1;i<4;i++)
      {
            ev<<key[i]<<"  ";

      }
      */
            if(msg->isSelfMessage())
          {
            if(not(msg->par("isp").boolValue()))
            {   // if it Key Distribution Self-Message
                  //Key Distribution
                  if(this == simulation.getModuleByPath(p[1]))
                  {
                  start=clock();
                  }
                  for(int i=1;i<=50;i++)
                  {
                        if(this!=simulation.getModuleByPath(p[i]))
                        {
                              //Embedding the Key
                              cMessage *kmsg=new
cMessage("Key_Setup");
                             kmsg->addPar("kid1");
                             kmsg->par("kid1").setLongValue(kid[1]);
                             kmsg->addPar("kid2");
                             kmsg->par("kid2").setLongValue(kid[2]);
                             kmsg->addPar("kid3");
                             kmsg->par("kid3").setLongValue(kid[3]);
                             kmsg->addPar("nid");
                             kmsg->par("nid").setLongValue(id);
                             kmsg->addPar("iskeymsg");
                             kmsg-
>par("iskeymsg").setBoolValue(true);
                                des =
simulation.getModuleByPath(p[i]);

if(this==simulation.getModuleByPath(p[1])&&(i==38))
                                {
                                      kmsg->addPar("start");
                                      kmsg-
>par("start").setDoubleValue(start);
                                }

                                sendDirect(kmsg,0,0,des->gate("in"));
                                ccn=ccn+1;
                        }
                  }
                  /*
                   * cMessage *kmsg=new
cMessage("Key_Distribution");
                   kmsg->addPar("keyfrm");
```

EK-18 (Devam). Temel olasılıksal yönteminde düğümün C++ dosyası

```
                kmsg->par("keyfrm").setLongValue(key);
                kmsg->addPar("nid");
                kmsg->par("nid").setLongValue(id);
                kmsg->addPar("iskeymsg");
                kmsg->par("iskeymsg").setBoolValue(true);
                des = simulation.getModuleByPath(p[51]);
                sendDirect(kmsg,0,0,des->gate("in"));
                key_mem[id]=key;
                *
                */

               if(this==simulation.getModuleByPath(p[38]))
               {   //If it Sending Message Start Node
                     cMessage *msg=new cMessage();
                     msg->addPar("isp");
                     msg->par("isp").setBoolValue(true);
                     scheduleAt(simTime()+0.5,msg);
               }
           }
           else
           {    // //if it Self-message to start sending the
message//isp=true

                      end  = clock();
                        cpu_time_used = ((double) (end -
start))/CLOCKS_PER_SEC; ;

ev<<"**************************************************************
*****"<<'\n';
                      ev<<"Time = "<<cpu_time_used*3<<'\n';
                      ev<<"Memory Usage for single node=
"<<sizeof(key) <<" Byte"<<'\n';
                      ev<<"Total Memory Usage in  the network=
"<<(sizeof(key))*50 <<" Byte"<<'\n';
                      ev<<"Vulnerability if a Node Captured =
"<<int(0.7*50/1)<<'\n';
                      ev<<"Vulnerability if 3 Node Captured =
"<<3*3*0.7*50/1<<'\n';
                      ev<<"Total Connection Required To Distribute
The Key =  "<<50*ccn<<'\n';
                      ev<<"Connectivity for each node =
"<<100*0.7<<"%"<<'\n';

ev<<"**************************************************************
*****"<<'\n';
                    cancelAndDelete(msg);
                       cMessage *msg=new
cMessage("Encripted_message");
                       msg->addPar("count");
                       msg->par("count").setLongValue(11);
                       msg->addPar("text");
                       char str[]="computer";
```

EK-18 (Devam). Temel olasılıksal yönteminde düğümün C++ dosyası

```
                        encmsg(str,key[2]);
                        ev<<"Message ="<<str;
                        msg->addPar("kid");
                        msg->par("kid").setLongValue(kid[2]);
                        msg->par("text").setStringValue(str);
                        msg->addPar("iskeymsg");
                        //To indicate it's Not Key Distribution
Self-message
                        msg->par("iskeymsg").setBoolValue(false);
                        msg->addPar("sid");
                        msg->par("sid").setLongValue(id);
                        //Getting the Next Node in path
                        des = simulation.getModuleByPath(pp[2]);
                        //Sending the Message
                        sendDirect(msg,0,0,des->gate("in"));
            }
          }//if(msg->isSelfMessage())

            else //If Not Self Message
            {
                  if(msg->par("iskeymsg").boolValue())// If received
key message
                  {
                        if((msg-
>getSenderModule()==simulation.getModuleByPath(p[1]))&&(this==simula
tion.getModuleByPath(p[38])))
                        {
                           start=msg->par("start");
                     }
                       // For Checking if there are keys in common
                       int temp[3];
                       temp[1]=msg->par("kid1").longValue();
                       temp[2]=msg->par("kid2").longValue();
                       temp[3]=msg->par("kid3").longValue();
                       bool can=false;
                       for(int i=1;i<=3;i++)
                       {
                             for(int j=1;j<=3;j++)
                             {
                                   if(kid[i]==temp[j])
                                   {
                                         // To Indicate that the
current Node Can Communicate
                                         // With the Received
Message ID
                                         can=true;
                                   }
                             }
                       }
                       if(can)
                       {
```

EK-18 (Devam). Temel olasılıksal yönteminde düğümün C++ dosyası

```
                              // List of what nodes it can
Communicate With
                              cancom[index]=msg-
>par("nid").longValue();

                              index+=1;
                        }
                        /*
                         *
                         for(int y=1;y<=index-1;y++)
                        {
                              ev<<cancom[y]<<"   ";
                        }
                         */
                  }
                  else // intermediate or end node
                  {
                        long iter=msg->par("count").longValue();
                  iter--;
                  /*
                      if(iter==0)
                      {
                        const char *str=msg-
>par("text").stringValue();
                        char dstr[30];
                            strcpy(dstr,str);
                            long sid=msg->par("sid");
                              decmsg(dstr,key_mem[sid]);
                              ev<<dstr;

                        }
                        */
                    //  else
                    //  {
                            int cd=msg->par("sid");
                            bool cdc=false;
                            for(int b=1;b<=index-1;b++)
                            {
                              if(cancom[b]==cd)
                              {
                                    cdc=true;
                              }
                            }
                            // To Indicate That there is key in
common
                            if(cdc)
                              ev<<"Key in Common";
                            else
                              ev<<"No Key in Common";
                        cMessage *copy=msg->dup();
                              cancelAndDelete(msg);
                              copy->par("sid").setLongValue(id);
```

EK-18 (Devam). Temel olasılıksal yönteminde düğümün C++ dosyası

```
                              copy->par("count").setLongValue(iter);
                              des = simulation.getModuleByPath(pp[12-
iter+1]);
                              sendDirect(copy,0,0,des->gate("in"));


                        // }

                  }
            }
}
// Function to Encrypt the Message
void Node::encmsg(char str[],int key)
{
      unsigned int  i;
      for( i=0;i< strlen(str);i++)
      {
            str[i]=  str[i]+ key % 26;
      }
}

// Function to uncipher the Encrypted Message
void Node::decmsg( char  str[],int key)
{
      unsigned int  i;
      for( i=0;i< strlen(str);i++)
      {
            str[i]= str[i]-key % 26;
      }
}
```

EK-19. Temel olasılıksal yönteminde ağda baz istasyonu tanımlaması

```
package key3test.col;

simple Sink
{
        @display("i=block/sink_l");
        gates:
        input in @directIn;
}
```

EK-20. Temel olasılıksal yönteminde baz istasyonu için basit modül

```
#ifndef __KEY3TEST_SINK_H_
#define __KEY3TEST_SINK_H_

#include <omnetpp.h>

class Sink : public cSimpleModule
{
      int keys[100];
  protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
    // Decryption  Function Prototype
    void decmsg(char str[],int key);
};

#endif
```

EK-21. Temel olasılıksal yönteminde baz istasyonu için C++ dosyası

```
#include "sink.h"

Define_Module(Sink);

// Sink Member Functions
void Sink::initialize()
{
   for(int i=1;i<=100;i++)
   {
         keys[i]=i+1;
   }
}

void Sink::handleMessage(cMessage *msg)
{
const char *str=msg->par("text").stringValue();

char dstr[30];
strcpy(dstr,str);
decmsg(dstr,keys[msg->par("kid").longValue()]);
ev<<dstr;

}

// Function to uncipher the Encrypted Message
void Sink::decmsg( char  str[],int key)
{
      unsigned int  i;
      for( i=0;i< strlen(str);i++)
      {
            str[i]= str[i]-key % 26;
      }
}
```

EK-22. Q-bileşim rastgele anahtar ön dağıtım yönteminde 50 düğümden oluşan ağ için düğüm dağıtımı

```
package key4test.net;

import key4test.col.Node;
import key4test.col.Sink;

// WSN Deployment
network Net
{


@display("bgb=1000,750;bgi=background/terrain;bgg=100,2,grey95;bgs=0
.75,m");
    submodules:
       node1: Node;
        node2: Node;
        node3: Node;
        node4: Node;
        node5: Node;
        node6: Node;
        node7: Node;
        node8: Node;
        node9: Node;
        node10: Node;
        node11: Node;
        node12: Node;
        node13: Node;
        node14: Node;
        node15: Node;
        node16: Node;
        node17: Node;
        node18: Node;
        node19: Node;
        node20: Node;
        node21: Node;
        node22: Node;
        node23: Node;
        node24: Node;
        node25: Node;
        node26: Node;
        node27: Node;
        node28: Node;
        node29: Node;
        node30: Node;
        node31: Node;
        node32: Node;
        node33: Node;
        node34: Node;
        node35: Node {
            @display("p=409.33334,188");
        }
```

EK-22 (Devam). Q-bileşim rastgele anahtar ön dağıtım yönteminde 50 düğümden oluşan ağ için düğüm dağıtımı

```
        node36: Node;
        node37: Node;
        node39: Node;
        node40: Node;
        node41: Node;
        node42: Node;
        node43: Node;
        node44: Node;
        node45: Node;
        node46: Node;
        node47: Node;
        node48: Node;
        node49: Node;
        node50: Node;
        node38: Node {
            @display("p=950,700");
        }
                 sink: Sink {
            @display("p=34.666668,40");
        }

}
```

EK-23. Q-bileşim rastgele anahtar ön dağıtım yönteminde bir düğümün tanımlaması (.ned) dosyası

```
package key4test.col;

simple Node
{
        @display("i=abstract/router_vs");
    gates:
        input in @directIn;
}
```

EK-24. Q-bileşim rastgele anahtar ön dağıtım yönteminde düğümün basit modül

```
#ifndef __KEY3TEST_NODE_H_
#define __KEY3TEST_NODE_H_

#include <omnetpp.h>
#include "time.h"


class Node : public cSimpleModule
{
      //Key Ring of Length 3
      int key[5];
      //ID For Each Corresponding Key in the Key Ring
      int kid[5];
      //Particular Node ID
      int id;
      char *p[100];
      char *pp[13];
      int cancom[50];
      int index;
      cModule *des;
      float prob;
      //Nomber of Shared keys To be Connected
      int q;

  protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
    // encryption  Function Prototype
    void encmsg(char str[],int key);
    // Decryption  Function Prototype
    void decmsg(char str[],int key);
  public:
    clock_t start, end;
    double cpu_time_used;
    int ccn;
};

#endif
```

EK-25. Q-bileşim rastgele anahtar ön dağıtım yönteminde düğümün C++ dosyası

```
#include "node.h"
#include "time.h"
Define_Module(Node);

void Node::initialize()
{

      int size=100;
      //Pool of size 100
      int pool[size];
      int temp;
      prob=0.7;
      q=3;

      long rem=((1-prob)*(size/10));

    for(int i=1;i<=size;i++)
    {
      pool[i]=i+1;

    }

    for(int j=1;j<=5;j++)
    {
      temp=intrand(rem)+1;
        /*  if(temp==0)
             {
                 temp=intrand(3);
             }
        */
          key[j]=pool[temp];
          kid[j]=temp;
    }
    //Random Selection of keys

    for(int k=1;k<=5;k++)
    {
      for(int h=1;h<=5;h++)
      {
            if(k!=h)
            {
                  if(key[k]==key[h])
                  {
                        temp=intrand(rem)+1;
                        key[h]=pool[temp];
                        kid[h]=temp;

                  }
            }
```

EK-25 (Devam). Q-bileşim rastgele anahtar ön dağıtım yönteminde düğümün C++ dosyası

```
      }
    }


p[1]="node1";p[2]="node2";p[3]="node3";p[4]="node4";p[5]="node5";

p[6]="node6";p[7]="node7";p[8]="node8";p[9]="node9";p[10]="node10";

p[11]="node11";p[12]="node12";p[13]="node13";p[14]="node14";p[15]="n
ode15";

p[16]="node16";p[17]="node17";p[18]="node18";p[19]="node19";p[20]="n
ode20";

p[21]="node21";p[22]="node22";p[23]="node23";p[24]="node24";p[25]="n
ode25";

p[26]="node26";p[27]="node27";p[28]="node28";p[29]="node29";p[30]="n
ode30";

p[31]="node31";p[32]="node32";p[33]="node33";p[34]="node34";p[35]="n
ode35";

p[36]="node36";p[37]="node37";p[38]="node38";p[39]="node39";p[40]="n
ode40";

p[41]="node41";p[42]="node42";p[43]="node43";p[44]="node44";p[45]="n
ode45";

p[46]="node46";p[47]="node47";p[48]="node48";p[49]="node49";p[50]="n
ode50";

p[51]="sink";pp[1]="node38";pp[2]="node39";pp[3]="node28";pp[4]="nod
e49";
      pp[5]="node47";pp[6]="node41";pp[7]="node15";pp[8]="node3";pp[
9]="node22";
      pp[10]="node12";pp[11]="node6";pp[12]="sink";
    index=1;
    //Getting the Particular Node ID
    id=getId();
    cMessage *msg=new cMessage();
    msg->addPar("isp");
    msg->par("isp").setBoolValue(false);
    ccn=0;
    scheduleAt(simTime()+0.5,msg);

}
```

EK-25 (Devam). Q-bileşim rastgele anahtar ön dağıtım yönteminde düğümün C++ dosyası

```
void Node::handleMessage(cMessage *msg)
{
      //ev<<id<<"   "<<getId();
      /*
      ev<<"node="<<id<<"  ";

      for(int i=1;i<4;i++)
      {
            ev<<key[i]<<"  ";

      }
      */
            if(msg->isSelfMessage())
           {
            if(not(msg->par("isp").boolValue()))
            {   // if it Key Distribution Self-Message
                  //Key Distribution
                  if(this==simulation.getModuleByPath(p[1]))
                  {
                  start=clock();
                  }
                  for(int i=1;i<=50;i++)
                  {
                        if(this!=simulation.getModuleByPath(p[i]))
                        {
                              //Embedding the Key
                              cMessage *kmsg=new
cMessage("Key_Setup");
                           kmsg->addPar("kid1");
                           kmsg->par("kid1").setLongValue(kid[1]);
                           kmsg->addPar("kid2");
                           kmsg->par("kid2").setLongValue(kid[2]);
                           kmsg->addPar("kid3");
                           kmsg->par("kid3").setLongValue(kid[3]);
                           kmsg->addPar("kid4");
                             kmsg-
>par("kid4").setLongValue(kid[4]);
                           kmsg->addPar("kid5");
                             kmsg-
>par("kid5").setLongValue(kid[5]);
                           kmsg->addPar("nid");
                           kmsg->par("nid").setLongValue(id);
                           kmsg->addPar("iskeymsg");
                           kmsg-
>par("iskeymsg").setBoolValue(true);
                             des =
simulation.getModuleByPath(p[i]);
                             ccn=ccn+1;

if(this==simulation.getModuleByPath(p[1])&&(i==38))
```

EK-25 (Devam). Q-bileşim rastgele anahtar ön dağıtım yönteminde düğümün C++ dosyası

```
                                {
                                      kmsg->addPar("start");
                                      kmsg-
>par("start").setDoubleValue(start);
                                }

                                sendDirect(kmsg,0,0,des->gate("in"));
                          }
                    }
                      /*
                       * cMessage *kmsg=new
cMessage("Key_Distribution");
                     kmsg->addPar("keyfrm");
                     kmsg->par("keyfrm").setLongValue(key);
                     kmsg->addPar("nid");
                     kmsg->par("nid").setLongValue(id);
                     kmsg->addPar("iskeymsg");
                     kmsg->par("iskeymsg").setBoolValue(true);
                     des = simulation.getModuleByPath(p[51]);
                     sendDirect(kmsg,0,0,des->gate("in"));
                     key_mem[id]=key;
                     *
                     */

                    if(this==simulation.getModuleByPath(p[38]))
                    {   //If it Sending Message Start Node
                          cMessage *msg=new cMessage();
                          msg->addPar("isp");
                          msg->par("isp").setBoolValue(true);
                          scheduleAt(simTime()+0.5,msg);
                    }
               }
               else
               {    // //if it Self-message to start sending the
message//isp=true

                           end  = clock();
                         cpu_time_used = ((double) (end -
start))/CLOCKS_PER_SEC; ;

ev<<"**********************************************************
*****"<<'\n';
                         ev<<"Time = "<<cpu_time_used*3<<'\n';
                         ev<<"Memory Usage for single node=
"<<sizeof(key)-8 <<" Byte"<<'\n';
                         ev<<"Total Memory Usage in the network =
"<<(sizeof(key)-8)*50 <<" Byte"<<'\n';
                         ev<<"Vulnerability if a Node Captured =
"<<int(0.7*50/3)<<'\n';
```

EK-25 (Devam). Q-bileşim rastgele anahtar ön dağıtım yönteminde düğümün C++ dosyası

```
                        ev<<"Vulnerability if 3 Node Captured =
"<<3*3*0.7*50/3<<'\n';
                        ev<<"Total Connection Required To Distribute
The Key =  "<<50*ccn<<'\n';
                        ev<<"Connectivity for each node =
"<<0.7*100/3<<"%"<<'\n';

ev<<"**************************************************************
*****"<<'\n';
                    cancelAndDelete(msg);

                          cMessage *msg=new
cMessage("Encripted_message");
                          msg->addPar("count");
                          msg->par("count").setLongValue(11);
                          msg->addPar("text");
                          char str[]="computer";
                          encmsg(str,key[2]);
                          ev<<str;
                          msg->addPar("kid");
                          msg->par("kid").setLongValue(kid[2]);
                          msg->par("text").setStringValue(str);
                          msg->addPar("iskeymsg");
                          //To indicate it's Not Key Distribution
Self-message
                          msg->par("iskeymsg").setBoolValue(false);
                          msg->addPar("sid");
                          msg->par("sid").setLongValue(id);
                          //Getting the Next Node in path
                          des = simulation.getModuleByPath(pp[2]);
                          //Sending the Message
                          sendDirect(msg,0,0,des->gate("in"));
            }
          }//if(msg->isSelfMessage())

            else //If Not Self Message
            {
                  if(msg->par("iskeymsg").boolValue())// If received
key message
                  {   // For Checking if there are keys in common
                        if((msg-
>getSenderModule()==simulation.getModuleByPath(p[1]))&&(this==simula
tion.getModuleByPath(p[38])))
                        {
                              start=msg->par("start");
                        }
                        int temp[10];
                        temp[1]=msg->par("kid1").longValue();
                        temp[2]=msg->par("kid2").longValue();
                        temp[3]=msg->par("kid3").longValue();
```

EK-25 (Devam). Q-bileşim rastgele anahtar ön dağıtım yönteminde düğümün C++ dosyası

```
                        temp[4]=msg->par("kid4").longValue();
                        temp[5]=msg->par("kid5").longValue();
                        int at=0;
                        // bool can=false;
                        for(int i=1;i<=5;i++)
                        {
                              for(int j=1;j<=5;j++)
                              {
                                    if(kid[i]==temp[j])
                                    {
                                          // To Indicate that the
current Node Can Communicate

                                          // With the Received
Message ID if the Share at Least
                                          // q Keys
                                          at+=1;
                                    }
                              }
                        }
                        if(at>=q)
                        {
                              // List of what nodes it can
Communicate With
                              cancom[index]=msg-
>par("nid").longValue();
                              index+=1;
                        }
                        /*
                         *
                         for(int y=1;y<=index-1;y++)
                        {
                              ev<<cancom[y]<<"   ";
                        }
                         */
                  }
                  else // intermediate or end node
                  {
                        long iter=msg->par("count").longValue();
                  iter--;
                  /*
                      if(iter==0)
                      {
                        const char *str=msg-
>par("text").stringValue();
                        char dstr[30];
                            strcpy(dstr,str);
                            long sid=msg->par("sid");
                              decmsg(dstr,key_mem[sid]);
                              ev<<dstr;
```

EK-25 (Devam). Q-bileşim rastgele anahtar ön dağıtım yönteminde düğümün C++ dosyası

```
                        }
                        */
                    //  else
                    //  {
                            int cd=msg->par("sid");
                            bool cdc=false;
                            for(int b=1;b<=index-1;b++)
                            {
                              if(cancom[b]==cd)
                              {
                                    cdc=true;
                              }
                            }

                            // To Indicate That there is key in
common
                            if(cdc)
                              ev<<"Key in Common";
                            else
                              ev<<"No Key in Common";
                          cMessage *copy=msg->dup();
                              cancelAndDelete(msg);
                              copy->par("sid").setLongValue(id);
                            copy->par("count").setLongValue(iter);
                            des = simulation.getModuleByPath(pp[12-
iter+1]);
                            sendDirect(copy,0,0,des->gate("in"));

                     // }

                  }
            }
}
// Function to Encrypt the Message
void Node::encmsg(char str[],int key)
{
      unsigned int  i;
      for( i=0;i< strlen(str);i++)
      {
            str[i]=  str[i]+ key % 26;
      }
}

// Function to uncipher the Encrypted Message
void Node::decmsg( char  str[],int key)
{
      unsigned int  i;
      for( i=0;i< strlen(str);i++)
      {
```

EK-25 (Devam). Q-bileşim rastgele anahtar ön dağıtım yönteminde düğümün C++ dosyası

```
        str[i]= str[i]-key % 26;
    }
```

EK-26. Q-bileşim Rastgele Anahtar Ön dağıtım Yönteminde Ağda Baz İstasyonu Tanımlaması

```
package key4test.col;

simple Sink
{
        @display("i=block/sink_l");
        gates:
        input in @directIn;
}
```

EK-27. Q-bileşim rastgele anahtar ön dağıtım yönteminde baz istasyonu için basit modülü

```
#ifndef __KEY3TEST_SINK_H_
#define __KEY3TEST_SINK_H_

#include <omnetpp.h>

class Sink : public cSimpleModule
{
      int keys[100];
  protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
    // Decryption  Function Prototype
    void decmsg(char str[],int key);
};

#endif
```

EK-28. Q-bileşim rastgele anahtar ön dağıtım yönteminde baz istasyonu için C++ dosyası

```
#include "sink.h"

Define_Module(Sink);

// Sink Member Functions
void Sink::initialize()
{
   for(int i=1;i<=100;i++)
   {
         keys[i]=i+1;
   }
}

void Sink::handleMessage(cMessage *msg)
{
const char *str=msg->par("text").stringValue();

char dstr[30];
strcpy(dstr,str);
decmsg(dstr,keys[msg->par("kid").longValue()]);
ev<<dstr;

}

// Function to uncipher the Encrypted Message
void Sink::decmsg( char  str[],int key)
{
      unsigned int  i;
      for( i=0;i< strlen(str);i++)
      {
            str[i]= str[i]-key % 26;
      }
}
```

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : Önder, Khalil
Uyruğu : Irak/ Kerkük
Doğum tarihi ve yeri : 08.02.1980 Bağdat
Medeni hali : Bekar
Telefon : 0 534 528 16 91
e-mail : onderameen@gmail.com

Eğitim

Derece	**Eğitim Birimi**	**Mezuniyet tarihi**
Lisans	Bağdat Üniversitesi/ Bilgisayar Bilimleri	2003
Lise	Merkeziye Lisesi	1998

İş Deneyimi

Yıl	**Yer**	**Görev**
2005-2007	Bağdat Üniversitesi	Araştırma Görevlisi

Yabancı Dil

Arapça
İngilizce

Hobiler

Müzik, Tenis