Düzce Üniversitesi Bilim ve Teknoloji Dergisi

Araştırma Makalesi

Yayın Şifreleme Şemaları Üzerinde Bir Karşılaştırma: Bir Yeni Yayın Şifreleme Şeması

Hüseyin BODUR [a,*], Resul KARA [b]

[a] *Bilgisayar Mühendisliği Bölümü, Mühendislik Fakültesi, Düzce Üniversitesi, Düzce, TÜRKİYE*

[b] *Bilgisayar Mühendisliği Bölümü, Mühendislik Fakültesi, Düzce Üniversitesi, Düzce, TÜRKİYE*

** Sorumlu yazarın e-posta adresi: huseyinbodur@duzce.edu.tr*

ÖZET

Bir yayın haberleşme yönteminde bir kaynaktan çoklu kullanıcılara mesaj iletimi için genellikle içerisinde şifreleme yöntemlerinin kullanıldığı şemalardan yararlanılır. Bu şema yapıları anahtar sunucu işlemleri açısından merkezi ve de-merkezi olmak üzere ikiye ayrılır. Bu çalışmada günümüzde yaygın olarak kullanılan iki merkezi yöntem olan Mantıksal Anahtar Hiyerarşisi (MAH) ve Tek Yönlü Fonksiyon Ağacı (TFA) şemalarına değinilmiştir. Yayın haberleşme için bir şema önerilmiş ve kullanıcı ekleme/çıkarma işlemlerinde anahtar iletim sayıları-boyutları ve kullanıcılarda bulunan anahtar sayıları-boyutları açılarından mevcut şemalar ile karşılaştırılmıştır.

Anahtar Kelimeler: *Yayın Haberleşme, Mantıksal Anahtar Hiyerarşisi, Tek Yönlü Fonksiyon Ağacı.*

A Comparison on Broadcast Encryption Schemes: A New Broadcast Encryption Scheme

ABSTRACT

In a broadcast communication method,the schemes in which encryption methods are used are often used to transmit messages from a source to multiple users. These schemes are divided into central and de-central in terms of key server operations. In this work, two central methods which are widely used nowadays are mentioned: Logical Key Hierarchy (LKH) and One Way Function Tree (OFT) schemes. A scheme for broadcast communication is proposed and compared with the existing schemes in terms of numbers-sizes of key transmissions in user joining/removing and numbers-sizes of keys in the user.

Keywords: *Broadcast Communication, Logical Key Hierarchy, One Way Function Tree*

Geliş: 07/01/2019, Düzeltme: 23/01/2019, Kabul: 25/01/2019

I. GİRİŞ

Günümüzde yayın haberleşme olarak adlandırılan ve bir kaynaktan çoklu kullanıcılara mesaj iletimi amacını taşıyan birçok şema yapısı bulunmaktadır. Literatürde bu şema yapıları üzerinde birçok defa çalışılmıştır. Özellikle kullanıcı ekleme/çıkarma işlemlerinde güvenilir, anahtar güncelleme açısından verimli yöntemler geliştirilmiştir. Bu şema yapıları kendi aralarında merkezi ve de-merkezi olmak üzere ikiye ayrılmaktadır. Merkezi ve de-merkezi yapıların aralarındaki en önemli fark, merkezi şema yapılarında Anahtar Sunucu (AS) isimli bir yapının bulunması ve kullanıcı ekleme/çıkarma işlemlerinin ardından ilgili kullanıcılara güncel anahtar değerlerinin iletiminde AS'ın görev almasıdır. De-merkezi yapılarda ise bir AS bulunmamakta, işlemler dağıtık kullanıcılar tarafından gerçekleştirilmektedir. Bu çalışmada literatürde bulunan ve günümüzde birçok alanda sıklıkla kullanılan MAH ve TFA şema yapılarından bahsedilmiş, yeni bir merkezi şema yapısı önerilmiştir.

Literatürde MAH ve TFA yapılarını geliştirmek adına pek çok çalışma yapılmıştır. Bu amaçla yapılan çalışmalardan birinde Shanu ve Chandrasekaran MAH yapısındaki en önemli işlemin kullanıcı ekleme/çıkarma işlemlerinin ardından ileri ve geri gizliliğin sağlanması için yapılan yeniden anahtarlama olduğunu belirtmiş, yeniden anahtarlama işlem maliyetini azaltmak için bir dağıtım fonksiyonundan yararlanmışlardır [1]. Bir diğer çalışmada Prathap ve Vasudevan, çeşitli şema yapılarını incelemişlerdir. Kullanıcı ekleme/çıkarma gibi işlemler üzerine bu şema yapılarının avantajlı yönlerini bir araya getirerek yeni hibrit bir şema yapısı önermişlerdir [2]. Bir diğer çalışmada Sakamoto ve arkadaşları, MAH şemasının kullanıcılardan yayın merkezine olan yol uzunluğunu azaltmak için Huffman algoritmasını kullandıkları bir şema yapısı önermişlerdir [3].
Bir diğer çalışmada Gu ve arkadaşları Anahtar Ağacı Yeniden Kullanımı (AAYK) isimli bir etkili anahtar yönetim şeması önermiştir. AAYK kullanıcıların yayın sistemi içerisinde bulunan çoklu programlara aynı anahtar değeriyle kaydolmasına izin veren bir anahtar yönetim yaklaşımıdır. MAH tabanlı olmasına rağmen, MAH yapısına göre daha düşük yeniden anahtarlama maliyetine sahiptir [4].
Bir diğer çalışmada Song ve arkadaşları dinamik gruplara şifreli bulut verisi paylaşımı yapılabilmesi için açık anahtar altyapısına dayanan bir yeni grup anahtar yönetim algoritması önermiştir. Bulut sunucu üzerinde kötü niyetli kullanıcıların saldırılarına maruz kalınsa dahi açık anahtarlı şifrelemenin avantajlarından yararlanan önerilen şema ile veri güvenliği sağlanmaktadır [5].
Bir diğer çalışmada Alyani ve arkadaşları Diffie-Hellman anahtar değişiminin MAH yapısı üzerine nasıl uygulanacağı konusunda bilgi vermiş ve MAH yapısı üzerinde değişiklik yaparak anahtar yönetim şemasını geliştirmeye çalışmışlardır. Bu değişiklik ağacın altkümelerindeki kullanıcı sayısını arttırarak performansını iyileştirmeye dayalıdır [6].

Bir diğer çalışmada Liu ve arkadaşları kullanıcı ekleme/çıkarma işlemlerinin ardından gerçekleştirilen anahtar güncelleme maliyetini azaltmak için bir sezgisel arama algoritmasına dayanan bir yeni ağaç yapısı önermişlerdir. Çalışmada ayrıca MAH ağacının her seviyesinde farklı sayıda düğüm bulunmaktadır [7]. Bir diğer çalışmada Sakamoto, bir anahtar ağacına eklenen veya çıkartılan ortalama kullanıcı sayısının bilindiği takdirde, anahtar güncelleme maliyetinin azaltılabileceğini savunan bir çalışma önermiştir [8]. Bir diğer çalışmada ise, MAH yapısının bir bulut sunucu üzerinde bulunan Nosql bir veritabanına uygulanması sırasında karşılaşılan sorunların çözümü için Diffie-Hellman anahtar değişiminden yararlanılmıştır [9].

Bir diğer çalışmada TFA şemasının güvenlik zayıflıkları incelenmiştir. TTFA (Tekrarlanan TFA) ve DTFA (Düğüm TFA) adında iki geliştirilmiş TFA şeması önerilmiştir. TFA ile karşılaştırıldığında

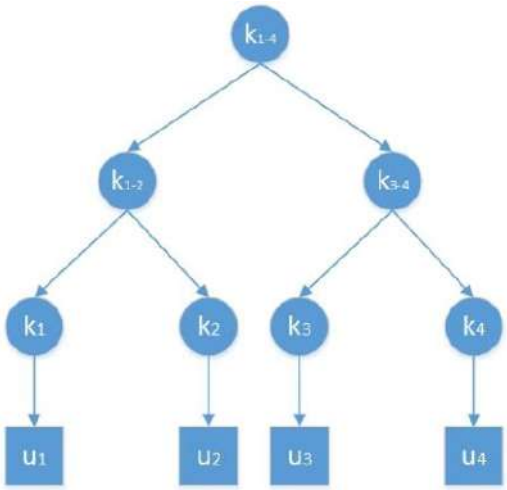
TTFA ve DTFA şemalarının grup yönetiminde ekstra iletişim maliyetine ihtiyaç duymadığı tespit edilmiştir [10] Bir diğer çalışmada TFA şemasının gizli anlaşma atakları karşısındaki zayıflığına değinilmiştir. TFA şeması üzerine bir yöntem eklenerek yeni bir şema yapısı önerilmiştir. Bu yöntem ile şema üzerinde rasgele sayıda kullanıcı ekleme/çıkarma işlemlerinin ardından minimum yayın boyutu ile gizli anlaşma ataklarının engellenmesi amaçlanmıştır [11]. Bir diğer çalışmada mikro ve makro ödeme sistemleri hakkında bilgi verilmiştir. Eliptik eğri şifreleme yöntemine dayanan yeni bir mikro ödeme şeması önerilmiştir. Önerilen şema TFA yapısından türetilmiş olan tek yönlü fonksiyon zinciri (TFZ) tabanlıdır [12].

Bu çalışmada MAH, TFA şemalarında kısaca bahsedilmiş, önerilen yöntem ile kullanıcı ekleme/ çıkarma işlemleri üzerinde anahtar iletim sayıları-boyutları açılarından ve kullanıcılarda saklanması gereken anahtar sayıları-boyutları açısından karşılaştırılmıştır.

II. KULLANILAN YÖNTEMLER

A. MANTIKSAL ANAHTAR HİYERARŞİSİ (MAH)

1997 yılında Chung Kei Wong ve ekibi tarafından geliştirilen MAH şemasının [13] amacı yetkili kullanıcıların eklendiği bir anahtar ağacı oluşturmaktır. Merkezi şema yapıları içerisinde yer alan MAH, en yaygın olarak kullanılan simetrik anahtar altyapısına sahip şema yapılarından bir tanesidir. Şekil 1'de görüldüğü üzere şema yapısında yayın merkezi kök düğümde, kullanıcılar ise yapraklarda bulunur.



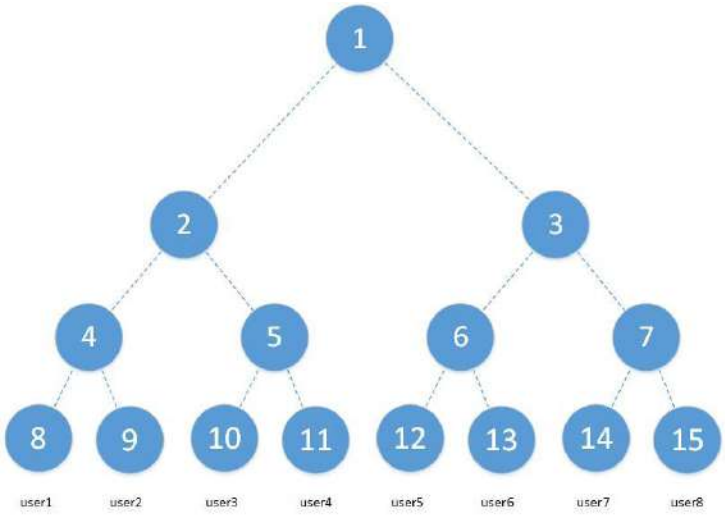
Şekil 1. *Örnek Bir Yayın Şifreleme Şeması.*

Yayın merkezi çeşitli şifreleme algoritmalarından birini kullanarak tek seferde kullanıcıların tümüne yayın mesajları gönderebilir. Bir AS, şema üzerinde kullanıcı ekleme/çıkarma gibi işlemlerden sorumludur. Yayın mesajları kök düğümde bulunan yayın merkezinden, yapraklarda bulunan kullanıcılara doğru tek yönlüdür. Her kullanıcının kendisine ait simetrik bir anahtarı bulunur.
Ağaç üzerinde ayrıca her ara düğümlerin ve kök düğümün de simetrik anahtarı vardır. Her kullanıcıya, kök düğümden kendisine kadar olan yolda bulunan anahtar değerleri AS tarafından iletilmelidir. Eğer

ağaç dengeli ve dolu ise, her kullanıcıda kendisinden kök düğüme kadar olan yolda toplamda $1 + log_d\, n$ adet anahtar bulunur. n ağaçta bulunan kullanıcı sayısını, d ise kullanıcının bulunduğu altkümenin derecesini temsil etmektedir. MAH ağaç yapısı dinamiktir. Bir kullanıcı herhangi bir zamanda MAH ağacına katılmak ya da ağaçtan ayrılmak için talepte bulunabilir. Ağaç yapısına eklenen ve çıkarılan her kullanıcının ardından ileri ve geri gizlilik sağlanmalıdır. İleri gizlilik, yayın şemasından ayrılan bir kullanıcının, yayın merkezi tarafından yayınlanan mesajları çözmesini engellemektir. Geri gizlilik ise, yayın şemasına eklenen bir kullanıcının, geçmişte yayınlanmış olan mesajları çözmesini engellemektir. İleri ve geri gizliliğin sağlanması için, şema yapısına bir kullanıcı eklendiğinde veya çıkartıldığında, kullanıcının bulunduğu konumdan yayın merkezine doğru yol üzerinde bulunan tüm düğüm anahtarlarının güncellenmesi gerekir. Ardından güncellenen kök düğüm ve ara düğüm anahtarları, bu anahtar değerlerine ihtiyaç duyan kullanıcılara AS tarafından dağıtılır.

B. TEK YÖNLÜ FONKSİYON AĞACI (TFA)

İlk olarak 1999 yılında Balenson ve arkadaşları tarafından taslak olarak önerilmiştir [14]. 2003 yılında ise Sherman ve McGrew tarafından daha detaylı olarak analiz edilmiştir [15]. Simetrik anahtar altyapısına sahiptir. Ağaç üzerindeki simetrik kullanıcı anahtarları aşağıdan yukarıya doğru bir tek yönlü fonksiyon *g()*, bir birleştirme fonksiyonu *f()* ve bir anahtar oluşturma fonksiyonu *key()* kullanılarak hesaplanır. Kullanıcılar yapraklarda bulunur. Her kullanıcı düğümü *(x)* ile ilişkilendirilmiş üç kriptografik anahtar bulunur. n_x düğüm sırrıdır. n'_x ise kör (blinded) düğüm sırrıdır. Bir tek yönlü fonksiyon *g()* kullanılarak hesaplanır: $n'_x = g(n_x)$. Tek yönlü fonksiyonların özelliğinin sonucu olarak, elde edilen değerin tersinin alınması yani n'_x'den n_x'in elde edilmesi mümkün değildir. *f()* fonksiyonu ise karıştırma (ör. XOR) fonksiyonudur. MAH şemasında anahtar değerleri kökten yapraklarda bulunan kullanıcılara yukarıdan aşağıya doğru dağıtılırken, TFA şemasında yapraklarda bulunan kullanıcılardan köke doğru aşağıdan yukarıya bir yol izler. TFA şeması, MAH gibi merkezi şema yapıları içerisinde yer alır. Şekil 2'de örnek bir TFA ağaç yapısını görülmektedir.



Şekil 2. *Örnek Bir TFA Ağaç Yapısı.*

Şekil 2'deki yapıda kullanıcılar yalnızca yapraklarda (8-15) bulunur.

Her düğüm anahtarı k, düğüm sırrının *key()* fonksiyonuna girmesiyle elde edilir.

$k = key(n_x)$

Her düğüm sırrı n_x, sol ve sağ çocuklarının kör düğüm sırlarının birleştirilmesinden elde edilir. Ara düğüm hesaplaması aşağıdaki şekildedir.

$$n_x = f\left(g\left(n_{x_{left}}\right), g\left(n_{x_{right}}\right)\right) = f(n^{l}_{x\_left}, n^{l}_{x\_right})$$

$n_4 = f(g(n_8), g(n_9))$ $\quad$ $n_5 = f(g(n_{10}), g(n_{11}))$

$n_6 = f(g(n_{12}), g(n_{13}))$ $\quad$ $n_7 = f(g(n_{14}), g(n_{15}))$

$n_2 = f(g(n_4), g(n_5))$ $\quad$ $n_3 = f(g(n_6), g(n_7))$

$n_1 = f(g(n_2), g(n_3))$

Yapraklarda bulunan her kullanıcının, düğüm sırlarını hesaplayabilmesi için kendisinden kök düğüme kadarki yolda kardeş kör düğüm sırlarını bilmesi gerekir. Örneğin; 8 numaralı düğüme $g(n_9)$, $g(n_5)$ ve $g(n_3)$ değerleri iletilmelidir. Aksi halde bu düğüm grup anahtarını hesaplayamaz.

III. UYGULANAN ÇALIŞMA

Önerilen şema yapısında literatürdeki çalışmalarda olduğu gibi bir ikili ağaç yapısı üzerinde yayın merkezi kök düğümde, kullanıcılar ise yapraklarda bulunur. MAH ve TFA yapılarından farklı olarak önerilen şemada hem simetrik hem de asimetrik şifreleme yöntemleri birlikte kullanılır. Ağaca eklenen her kullanıcının asimetrik bir şifreleme yöntemi kullanılarak oluşturulmuş biri açık (pu_{user_X}), diğeri gizli anahtar (pr_{user_X}) olmak üzere birbirleriyle ilişkili iki anahtarı bulunur. Ortak gizli anahtar değerinin hesaplanması için her kullanıcıda ayrıca bir simetrik anahtar değeri bulunmalıdır. Her kullanıcı bu değeri kendi gizli anahtarını kullanarak hesaplar.
Bunun için gizli anahtar değerini kendi pozisyon değeriyle tuzlayarak bir *hash()* fonksiyonuna sokar.

$n_{user_X} = hash(pr_{user_X} + position)$

Kullanıcı, pr_{user_X}ve n_{user_X} değerlerini kendisinde saklarken, pu_{user_X} değerini önceden belirlenen açık anahtar kütüphanesine yükler.
Yayın merkezinde bulunan ortak gizli anahtarların hesaplanması için TFA yapısına benzer bir şekilde kullanıcılardan kök düğüme doğru anahtar hesaplaması yapılması gerekir.
Her ara düğüm anahtarı n_x, sol çocuğunun gizli anahtar değerinin sol yarısı ile sağ çocuğunun gizli anahtar değerinin sağ yarısının birleştirilerek özetinin alınmasıyla elde edilir.

$n_x = hash(n_{x_{left}} + n_{x_{right}})$

Şekil 2'deki şema yapısına benzer olarak, tam ve dolu bir ağaç yapısında kullanıcılardan kök düğüme sırasıyla ara düğüm anahtarlarının ve ortak gizli anahtarın hesaplanması aşağıdaki hesaplamayla elde edilir.

$n_x = hash(n_{x_{left}} + n_{x_{right}})$
$n_4 = hash\ (n_{8_{left}} + n_{9_{right}})$
$n_5 = hash\ (n_{10_{left}} + n_{11_{right}})$
$n_6 = hash\ (n_{12_{left}} + n_{13_{right}})$
$n_7 = hash\ (n_{14_{left}} + n_{15_{right}})$
$n_2 = hash\ (n_{4_{left}} + n_{5_{right}})$
$n_3 = hash\ (n_{6_{left}} + n_{7_{right}})$
$n_1 = hash(n_{2_{left}} + n_{3_{right}})$

n_1 ortak gizli anahtar değeri TFA şemasındakine benzer bir şekilde yaprak düğümlerden kök düğüme gizli anahtar değerleri kullanılarak hesaplanır. TFA şemasından farklı olarak aşağıdan yukarıya anahtar değerinin yalnızca yarısı iletilir. Bu ise ortak anahtar üretiminde toplam hesaplama maliyetini azaltır.

Kök düğüm ortak gizli anahtar değeriyle tüm kullanıcılara bir veri göndermek istediğinde öncelikle hash fonksiyonu ile verinin özetini çıkartır. Özet değerini kendi gizli anahtar değeri pr_{root} ile imzalar ve mesajın sonuna ekleyerek tüm kullanıcılara gönderir.

Kullanıcılar kendilerine gelen veri içindeki imzayı ayırır. Bu değeri açık anahtar kütüphanesindeki pu_{root} ile çözer. Ardından verinin özet değerini çıkartarak bu iki değeri karşılaştırır. Eğer değerler birbirlerine eşitse veri yayın merkezi tarafından gönderilmiştir. Bu sayede herhangi bir kötü niyetli kullanıcı kök düğüm gibi davranıp mesaj iletim işlemi gerçekleştiremez. Eğer yayın merkezi bir kullanıcıya özel bir yayın yapmak isterse anahtar kütüphanesinden o kullanıcın açık anahtar değerini kullanabilir.

Her kullanıcıya kendisinden kök düğüme kadarki yolda bulunan kardeş anahtar değerleri verilmelidir. Yayın işleminde ortak gizli anahtar değeri herhangi bir gizli anahtarlı şifreleme yöntemi içerisinde kullanılarak iletim gerçekleştirilebilir.

IV. DEĞERLENDİRME

Bu çalışmada, literatürde bulunan MAH ve TFA yapıları ile önermiş olduğumuz yöntemin kodları oluşturularak, dört kriter açısından karşılaştırılmıştır. Bu kriterler sırasıyla: anahtar iletim sayıları, anahtar iletim boyutları, kullanıcıda bulunan anahtar sayıları ve kullanıcıda bulunan anahtar boyutları şeklindedir.

Kullanıcı ekleme/çıkarma işlemi için toplamda 2^{21} kullanıcılı bir örnek gerçekleştirilmiştir. Her 2^n kullanıcının eklenmesinin/çıkartılmasının ardından, anahtar iletim sayıları ile bayt cinsinden anahtar iletim boyutları elde edilmiş ve Tablo 1 ve Tablo 2'de gösterilmiştir. Yine şema üzerindeki her 2^n kullanıcı için kullanıcılarda bulunan anahtar sayıları ve boyutları Tablo 3'de gösterilmiştir.
K.S. kısaca "Kullanıcı Sayısı", Ö.Ş. ise kısaca "Önerilen Şema" anlamına gelmektedir.

Tablo 1. *Kullanıcı Ekleme – Anahtar İletim Sayıları ve Boyutları*

	Anahtar İletim Sayısı			**Anahtar İletim Boyutu (Bayt)**		
K.S.	**MAH**	**TFA**	**Ö.Ş.**	**MAH**	**TFA**	**Ö.Ş.**
2^0	2	2	5	48	44	298
2^1	5	5	12	120	108	606
2^2	13	13	26	288	256	1036
2^3	33	33	56	696	612	1730
2^4	81	81	122	1680	1464	2992
2^5	193	193	268	4008	3468	5470
2^6	449	449	590	9408	8096	10540
2^7	1025	1025	1296	21720	18612	21114
2^8	2305	2305	2834	49392	42184	43336
2^9	5121	5121	6164	110856	94428	90134
2^{10}	11265	11265	13334	246048	209136	188644
2^{11}	24577	24577	28696	540984	459012	395698
2^{12}	53249	53249	61466	1179984	999704	830080
2^{13}	114689	114689	131100	2556264	2162988	1739598
2^{14}	245761	245761	278558	5505408	4653376	3640348
2^{15}	524289	524289	589856	11796888	9961812	7605482
2^{16}	1114113	1114113	1245218	25166256	21234024	15863224
2^{17}	2359297	2359297	2621476	53477832	45089148	33033862
2^{18}	4980737	4980737	5505062	113246688	95420816	68685652
2^{19}	10485761	10485761	11534376	239075832	201327012	142610466
2^{20}	22020097	22020097	24117290	503317008	423625144	295702768
2^{21}	46137345	46137345	50331692	1056965160	889192908	612372926

Tablo 2. *Kullanıcı Çıkarma – Anahtar İletim Sayıları ve Boyutları*

	Anahtar İletim Sayısı			**Anahtar İletim Boyutu (Bayt)**		
K.S.	**MAH**	**TFA**	**Ö.Ş.**	**MAH**	**TFA**	**Ö.Ş.**
2^{21}	24117248	24117248	26214402	578813976	490733588	413139150
2^{20}	35651584	35651584	38797316	855638064	725614632	614465948
2^{19}	41156608	41156608	44826630	987758664	837812284	712508010
2^{18}	43778048	43778048	47710216	1050673248	891289680	760218424
2^{17}	45023232	45023232	49086474	1080557688	916717668	783418374
2^{16}	45613056	45613056	49741836	1094713488	928776312	794690772
2^{15}	45891584	45891584	50053134	1101398184	934477964	800163234
2^{14}	46022656	46022656	50200592	1104543936	937164960	802817648
2^{13}	46084096	46084096	50270226	1106018520	938426548	804103998
2^{12}	46112768	46112768	50302996	1106706672	939016392	804726796
2^{11}	46126080	46126080	50318358	1107026184	939290844	805028058
2^{10}	46132224	46132224	50325528	1107173664	939417840	805173672
2^9	46135040	46135040	50328858	1107241272	939476228	805244022
2^8	46136320	46136320	50330396	1107272016	939502872	805278020
2^7	46136896	46136896	50331102	1107285864	939514924	805294482
2^6	46137152	46137152	50331424	1107292032	939520320	805302496
2^5	46137264	46137264	50331570	1107294744	939522708	805306446
2^4	46137312	46137312	50331636	1107295920	939523752	805308444
2^3	46137332	46137332	50331666	1107296424	939524204	805309506

2^2	46137340	46137340	50331680	1107296640	939524400	805310120
2^1	46137343	46137343	50331687	1107296736	939524488	805310520
2^0	46137345	46137345	50331692	1107296784	939524532	805310818

Ağaca her 2^n kullanıcı ekleme/çıkarma işlemlerinin ardından anahtar iletim sayısı açısından MAH ve TFA şemaları benzer sonuç vermektedir. İletim sayıları aynı olsa da, yöntemlerin anahtar güncelleme işlemleri birbirinden farklı olduğundan anahtar iletim boyutları birbirinden farklıdır. Önermiş olduğumuz yöntem kullanıcı ekleme/çıkarma işlemlerinde anahtar iletim sayısı açısından diğer yöntemlerle karşılaştırıldığında kötü sonuç vermektedir. Fakat iletimi yapılan anahtar boyutu açısından incelendiğinde, anahtar güncelleme işlemi, aşağıdan yukarıya doğru bir simetrik anahtarın sol yâda sağ yarısının iletimi şeklinde, daha küçük anahtar boyutu kullanılarak yapıldığından diğer yöntemlere göre daha iyi sonuç vermektedir.

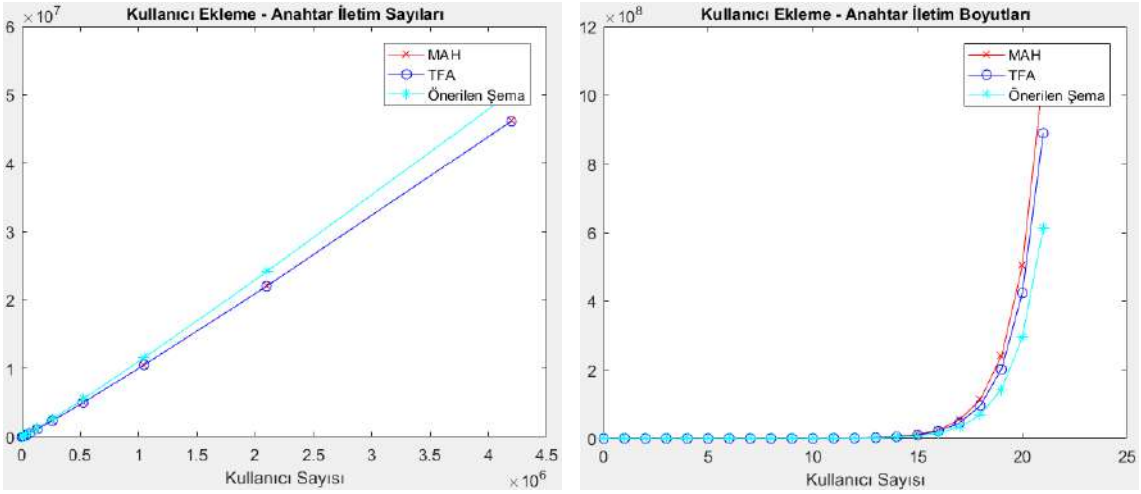
Tablo 3. *Kullanıcıda Bulunan Anahtar Sayıları ve Boyutları*

	Kullanıcılarda Bulunan Anahtar Sayıları			**Kullanıcılarda Bulunan Anahtar Boyutları (Bayt)**		
K.S.	**MAH**	**TFA**	**Ö.Ş.**	**MAH**	**TFA**	**Ö.Ş.**
2^0	1	1	3	24	24	216
2^1	2	2	4	48	44	226
2^2	3	3	5	72	64	236
2^3	4	4	6	96	84	246
2^4	5	5	7	120	104	256
2^5	6	6	8	144	124	266
2^6	7	7	9	168	144	276
2^7	8	8	10	192	164	286
2^8	9	9	11	216	184	296
2^9	10	10	12	240	204	306
2^{10}	11	11	13	264	224	316
2^{11}	12	12	14	288	244	326
2^{12}	13	13	15	312	264	336
2^{13}	14	14	16	336	284	346
2^{14}	15	15	17	360	304	356
2^{15}	16	16	18	384	324	366
2^{16}	17	17	19	408	344	376
2^{17}	18	18	20	432	364	386
2^{18}	19	19	21	456	384	396
2^{19}	20	20	22	480	404	406
2^{20}	21	21	23	504	424	416
2^{21}	22	22	24	528	444	426

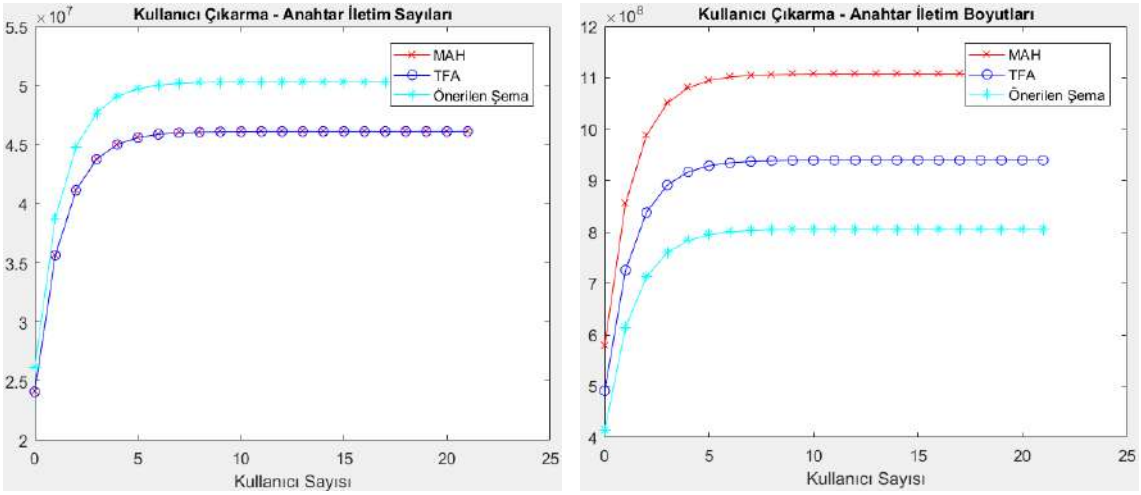
Ağaçta bulunan her 2^n kullanıcı için, kullanıcı ekleme/çıkarma işlemlerinin ardından yapılan anahtar güncelleme işleminin başarılı bir şekilde gerçekleştirilmesi için her kullanıcıda belirli sayıda anahtarın bulunması gerekir. Bu sayı MAH ve TFA şemalarında aynı iken bizim yöntemimizde bu yöntemlere kıyasla 2 adet daha fazladır. Bunun nedeni MAH ve TFA şemalarının içerisinde simetrik bir şifreleme yönteminin kullanılıyor olması, bizim yöntemimizde ise hem simetrik hem de asimetrik şifreleme yöntemlerinden faydalanıyor olmasıdır. Kullanıcılarda bulunan anahtarların boyutları incelendiğinde

yöntemimiz diğer yöntemlere kıyasla belirli bir kullanıcı sayısına kadar daha kötü sonuç vermekte, 2^{20} ve üzeri kullanıcının ardından iyi sonuç vermeye başlamaktadır.

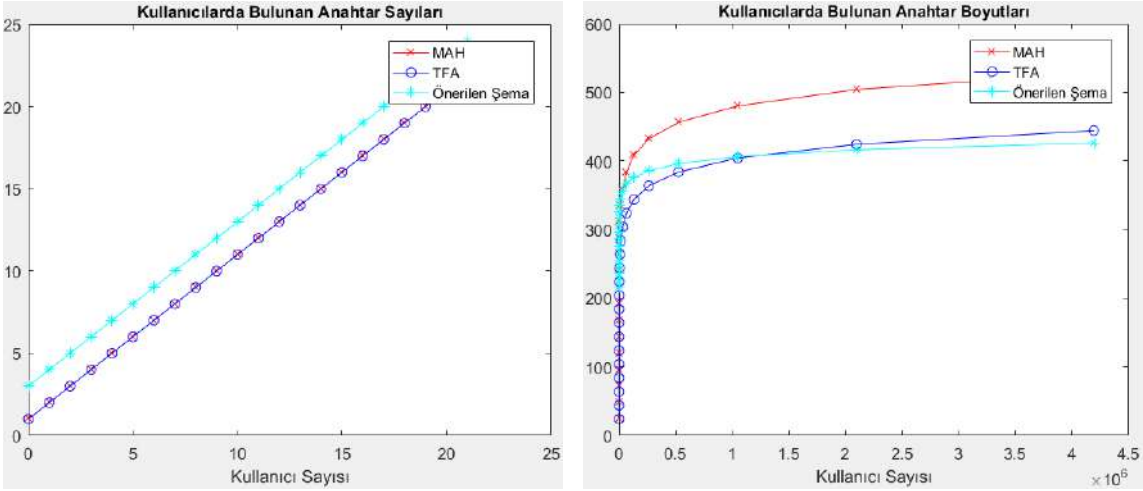
Tablo 1, Tablo 2 ve Tablo 3'de gösterilen veriler grafiksel olarak ifade edildiğinde Şekil 3, Şekil 4 ve Şekil 5 elde edilir.



Şekil 3. *Kullanıcı Ekleme - Anahtar İletim Sayıları ve Boyutları*



Şekil 4. *Kullanıcı Çıkarma - Anahtar İletim Sayıları ve Boyutları*



Şekil 5. *Kullanıcılarda Bulunan Anahtar Sayıları ve Boyutları*

Elde edilen sonuçlara göre önerilen yöntem özellikle anahtar boyutları bakımından diğer şemalara göre iyi sonuçlar vermektedir. Bu ise hesaplama ve iletim maliyetinin azalmasını sağlamaktadır.

IV. SONUÇ

Çalışmada yayın şifreleme alanında literatürde bulunan MAH ve TFA yapılarına değinilmiştir. Bu yöntemler Java dilinde yazılmış, sonuçlar Matlab üzerinden karşılaştırılmıştır. Sonuçlar i7-7700HQ CPU, 2.80GHz bir işlemci ve 16 GB RAM'e sahip bir bilgisayar kullanılarak elde edilmiştir. Yeni bir şema yapısı önerilmiştir. Kullanıcı ekleme/çıkarma işlemleri için anahtar iletim sayıları ve boyutları ile kullanıcılarda bulunan anahtar sayıları ve boyutları gibi dört kriter açısından karşılaştırılmıştır.
Önerilen şemanın anahtar iletim sayıları ve kullanıcılara bulunan anahtar sayıları açılarından kötü sonuçlar verdiği, anahtar iletim boyutları ve kullanıcılarda bulunan anahtar boyutları açılarından ise iyi sonuçlar verdiği gözlemlenmiştir. Elde edilen sonuçlar tablo yapısı ve grafiksel olarak ifade edilmiştir. İlerleyen çalışmalarda kriter sayısı arttırılarak daha detaylı bir analiz işlemi gerçekleştirilecektir.

V. KAYNAKLAR

[1] Shanu PK. ve Chandrasekaran K., "Distribution function based efficient secure group communication using key tree," Recent Trends in Information Technology (ICRTIT), ss. 1-6, 2016.

[2] Prathap M. Joe ve Vasudevan V., "Analysis of the various key management algorithms and new proposal in the secure multicast communications," arXiv preprint arXiv:0906.3956, 2009.

[3] Sakamoto T., Tsuji T. ve Kaji Y., "Group key rekeying using the LKH technique and the huffman algorithm," Information Theory and Its Applications (ISITA), ss. 1-6, 2008.

[4] Gu Q., Peng L. ve Wang-Chien L., "KTR: An efficient key management scheme for secure data access control in wireless broadcast services," IEEE Transactions on Dependable and Secure Computing, 6.3, ss. 188-201, 2009.

[5] Song W., Zou H., Liu H. ve Chen J., "A practical group key management algorithm for cloud data sharing with dynamic group," China Communications, 13.6, ss. 205-216, 2016.

[6] Alyani N., Seman K., Nawawi NM. ve Sayuti MNSM., "The Improvement of Key Management Based On Logical Key Hierarchy by Implementing Diffie Hellman Algorithm," J. Emerging Trends in Computing and Information Sciences, 3.3, 2012.

[7] Liu H., Li J., Hao X. ve Zou G., "A novel LKH key tree structure based on heuristic search algorithm," Communication Problem-Solving (ICCP), ss. 35-38, 2014.

[8] Sakamoto N., "An efficient structure for LKH key tree on secure multicast communications,"In Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD) ss. 1-7, 2014.

[9] Bodur H. ve Kara R., "Implementing Diffie-Hellman key exchange method on logical key hierarchy for secure broadcast transmission," Computational Intelligence and Communication Networks (CICN), ss. 144-147, 2017.

[10] Sun Y., Chen M., Bacchus A. ve Lin X., "Towards collusion-attack-resilient group key management using one-way function tree," Computer Networks, 104, ss. 16-26, 2016.

[11] Xu X., Wang L., Youssef A. ve Zhu B., "Preventing collusion attacks on the one-way function tree (OFT) scheme," Applied Cryptography and Network Security, ss. 177-193, 2007.

[12] Hwang MS. ve Sung PC., "A study of micro-payment based on one-way hash chain," IJ Network Security, 2.2, ss. 81-90, 2006.

[13] Wong CK., Gouda M. ve Lam SS., "Secure group communications using key graphs," IEEE/ACM transactions on networking, 8.1, 28, ss. 16-30, 1998.

[14] Balenson D., McGrew D. ve Sherman A., "Key management for large dynamic groups: One-way function trees and amortized initialization," 1999.

[15] Sherman AT. ve McGrew DA., "Key establishment in large dynamic groups using one-way function trees," IEEE transactions on Software Engineering, 29.5, ss. 444-458, 2003.