İSTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ

BİLGİİŞLEM ORTAMI SUNAN BULUT İÇİN GÜVENLİK DÜZENEKLERİ

DOKTORA TEZİ

Mehmet Tahir SANDIKKAYA

Bilgisayar Bilimleri Anabilim Dalı

Bilgisayar Bilimleri Programı

ARALIK 2015

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ

BİLGİİŞLEM ORTAMI SUNAN BULUT İÇİN GÜVENLİK DÜZENEKLERİ

DOKTORA TEZİ

Mehmet Tahir SANDIKKAYA

(704052007)

Bilgisayar Bilimleri Anabilim Dalı

Bilgisayar Bilimleri Programı

Tez Danışmanı: Prof. Dr. Ali Emre HARMANCI

ARALIK 2015

İTÜ Bilişim Enstitüsü'nün 704052007 numaralı Doktora Öğrencisi Mehmet Tahir SANDIKKAYA, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "BİLGİİŞLEM ORTAMI SUNAN BULUT İÇİN GÜVENLİK DÜZENEKLERİ" başlıklı tezini aşağıdaki imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Prof. Dr. Ali Emre HARMANCI** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. Sema Fatma OKTUĞ** İstanbul Teknik Üniversitesi	.....
	Prof. Dr. Mehmet Bülent ÖRENCİK Beykent Üniversitesi	.....
	Prof. Dr. Mehmet Ufuk ÇAĞLAYAN Boğaziçi Üniversitesi	.....
	Doç. Dr. Deniz Turgay ALTILAR İstanbul Teknik Üniversitesi	.....

Teslim Tarihi : **7 Ekim 2015**
Savunma Tarihi : **23 Aralık 2015**

Öğretenlere ve öğreneceklere,

ÖNSÖZ

Bir eser ortaya koyarken öncelikli görevimin o eserin ortaya çıkmasında katkısı olan kişilere teşekkür etmek olduğunu düşünüyorum. Ancak, katkısı olanları düşünmeye başlayınca bu iş içinden çıkılmaz oluyor. Her düşüncenin birkaç öğreteni, birkaç hazırlayanı, birkaç esin vereni, birkaç tartışanı var. Bunların ötesinde bu tezin hazırlanması için gereken koşulların oluşmasında hakkı zor ödenecek emeği geçenler var. Elbette kimileri diğerlerinden daha çok katkı sunmuş olabilir. Yine de, bir eserin bütünlüğü içinde her katkı övgüye değerdir. Kimseyi adıyla anmama gerek kalmadan yaşamımdaki herkesin bu eser üzerinden kendine hak ettiğince bir pay biçebileceğini söylemekle yetinmeliyim.

Aileme, öğretmenlerime, dostlarıma teşekkür ederim.

Bunun ötesinde bu eserin bilimsel yetkinliğinin yanında Türkçe olmasına ve Türkçenin doğru kullanılmasına özendiğimizi söylemek zorundayım. Türkçede bilimsel dilin birliği için bugüne dek Türkiye Bilimler Akademisi'nin terim sözlüklerine girmiş olanlar varsa bunlar yeğlenmiş, sözlüklerde bulunmayan terimler için akademide sık kullanım gözetilmiştir. Bundan başka, ilk kez bu tezde kullanıldığını düşündüğümüz terimler de vardır. Her durumda, anılan sözlüklerde yer almayan ve kullanımının şüphe uyandırabileceği düşünülen her söz dipnot ile açıklanmıştır.

Aralık 2015 Mehmet Tahir SANDIKKAYA

İÇİNDEKİLER

Sayfa

KISALTMALAR

ABD	**:**	Amerika Birleşik Devletleri
ARPA	**:**	Advanced Research Projects Agency
	:	ABD Savunma Bakanlığı İleri Araştırma Projeleri Ajansı
AVISPA	**:**	Automated Validation of Internet Security Protocols and Applications
	:	İnternet Güvenlik Protokol ve Programlarının Bilgisayarla Onaylanması
CTSS	**:**	Compatible Time-Sharing System
	:	Uyumlu Zaman Paylaşımlı Dizge
DTLS	**:**	Datagram Transport Layer Security
	:	Veri Bloğu Taşıma Katmanı Güvenliği
EC2	**:**	Elastic Computing Cloud
	:	Esnek Bilgiişlem Bulutu
G/Ç	**:**	Giriş/Çıkış
HTTP	**:**	Hyper Text Transport Protocol
	:	Hiper Metin Aktarım Protokolü
IaaS	**:**	Infrastructure-as-a-Service
	:	Sanal Bilgisayar Sunan Bulut
IETF	**:**	Internet Engineering Task Force
	:	İnternet Mühendisliği Çalışma Kolu
IP	**:**	Internet Protocol
	:	İnternet Protokolü
JMX	**:**	Java Management Extensions
	:	Java Yönetim Eklentileri
MIT	**:**	Massachusetts Institute of Technology
	:	Massachusetts Teknoloji Enstitüsü
Multics	**:**	Multiplexed Information and Computing Service
	:	Çoklanmış Bilgi ve Bilgiişlem Hizmeti
NIST	**:**	National Institute of Standards and Technology
	:	ABD Ticaret Bakanlığı Ulusal Standartlar ve Teknoloji Enstitüsü
PaaS	**:**	Platform-as-a-Service
	:	Bilgiişlem Ortamı Sunan Bulut
Project MAC	**:**	Project on Mathematics and Computation
	:	Matematik ve Bilgiişlem Üzerine Proje
RAM	**:**	Random-access Memory
	:	Rasgele Erişilir Bellek
s.	**:**	sayfa(lar)
SaaS	**:**	Software-as-a-Service
	:	Hazır Yazılım Sunan Bulut

TCP	**:**	Transmission Control Protocol
	:	İletim Denetimi Protokolü
TLS	**:**	Transport Layer Security
	:	Taşıma Katmanı Güvenliği
t.y.	**:**	tarih yok
UCSD	**:**	University of California, San Diego
	:	Kaliforniya Üniversitesi, San Diego
UDP	**:**	User Datagram Protocol
	:	Kullanıcı Veri Bloğu Protokolü
v.d.	**:**	ve diğerleri

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

BİLGİİŞLEM ORTAMI SUNAN BULUT İÇİN GÜVENLİK DÜZENEKLERİ

ÖZET

Tez çalışmalarına bilgiişlem ortamı sunan bulutlara uygun güvenlik düzenekleri oluşturmak için başlanmıştır. Düzeneklerin oluşturulması sırasında tezin bölümleri bir bütünlük içinde bilgiişlem ortamı sunan bulutların farklı güvenlik sorunlarını belirleyerek ve irdeleyerek tezde sayılan tüm sorunlara çözüm önerileri getirir. Öncül çalışmalar hem bulutun en önemli eksiğinin hem müşterileri bulutu kullanmaktan alıkoyanın hem de akademik olarak en çok ilgi çeken konunun güvenlik olduğunu ortaya koymaktadır.

NIST'in yaygın onay gören tanımına göre: "bulut bilgiişlem, paylaşılan bir havuz içindeki yapılandırılabilir, en az yönetimsel çaba ya da etkileşim ile, hızla, hizmete sokulabilen ve hizmetten çıkarılabilen bilgiişlem kaynaklarına (ağ, sunucu, depolama, yazılımlar ve hizmetler) yaygın ve kullanışlı yollarla, ağ üzerinden, istendiği an erişim sağlayan modeldir."

Bulut hizmet modelleri içinden biri tezin odaklandığı bilgiişlem ortamı sunan buluttur. Aynı kaynağın tanımına göre: "Bu hizmet modelinde müşterinin işletim sistemine ve üzerindeki programların pek çoğuna erişimi yoktur. Müşteri hazırladığı ya da edindiği programları sağlayıcının sunduğu bilgiişlem ortamlarında çalıştırır. Bir gereklilik olmamakla birlikte, seçenek olarak, sağlayıcı müşterilerine uygulama geliştirmelerini kolaylaştıracak araçlar (metin düzenleyiciler, derleyiciler, kitaplıklar, vb.) sunabilir."

Tez yedi bölümden oluşmaktadır. Tezin birinci bölümü bulutun tanımını ve tanıtımını yapar.

İkinci bölüm bulutun tarihsel evrimini ve gelişimini açıkladıktan sonra güvenlik tartışmasının kapsamını belirler ve tezin sınırlarını çizer.

Üçüncü bölüm günümüzde sık kullanılan bilgiişlem ortamı sunan bulutların güvenlik gereksinimlerinin giderilmesine yönelik bir girişimdir. Günümüzde kurulu bilgiişlem ortamı sunan bulutlar işlem gücünü çoğunlukla izlekler yoluyla sunar. Bu tür bilgiişlem ortamı sunan bulutlarda işlem yapanların verilerinin birbirine karışmaması için izleklerin yalıtımı güvenliğin temelini oluşturur. İzleklerin daha iyi yalıtılması yolları üçüncü bölümde araştırılmış, yalıtımın sınırları gösterilmiştir.

Dördüncü bölümde izleklerden daha güvenli ve kolay biçimde işlem gücünü sunmada süreçlerin kullanılması önerilerek pek çok düzeneğin bir arada çalışabileceği bir tasarım yapılır. Sorunlar ve çözümler sıralanır. Tasarımın açıkta kalan noktaları vardır.

Bunlardan biri süreçlerin bağlamlarıyla birlikte bulutta nasıl yönetileceği, diğeri de bulutta gerçekleşen olayların tarafsız ve yadsınamaz biçimde nasıl günlüklere aktarılacağıdır. Bu konular için yenilikçi yaklaşımlarla tasarım genişletilir ve her biri izleyen bölümlerde ayrıca açıklanır.

Beşinci bölümde süreçlerin bulutta güvenli biçimde çalıştırılmaları için nasıl bir yol izlenmesi gerektiği açıklanır. Süreçlerin bulutta çalıştırılmaları, yalıtımları, erişim kurallarının ve buluttaki kaynakların kullanımının yönetimi, bu sırada alınması gereken kriptolojik önlemleri bir arada içeren süreç kozaları bu bölümün konusudur.

Altıncı bölümde bulutu kullanan taraf ile bulutta hizmet sağlayan tarafın eşit sayılarak bulutta gerçekleşen olayların tarafsız ve yadsınamaz biçimde günlük kayıtlarına aktarıldığı, biçimsel olarak doğrulanmış bir düzenek tanıtılır. Bu düzenek kurulu bulutlarda önemli bir sorun oluşturan tek tarafın denetimindeki günlüklere bir seçenek oluşturur ve kayıt bulunmasına karşın denetimi tek tarafta olduğu için çözülemeyen uyuşmazlıkların ortadan kalkmasında yararı olacağı düşünülmektedir.

Gerek var olan bilgiişlem ortamı sunan bulutlar için ortaya konan güvenlik düzenekleri ile gerekse pek çok güvenlik düzeneğini bir araya getirerek bilgiişlem ortamı sunan bulutlara yeni ve daha güvenli bir yaklaşım getirmeyi öneren tasarımla, sunulan düzeneklerde bilgiişlem ortamı sunan bulutun güvenliği belirlenen çerçeve içinde tüm yönleriyle tartışılmış ve sayılan güvenlik zayıflıkları için önlemler getirilmiştir. Tezin başlıca katkıları izleyen paragraflarda sıralanmıştır.

Üçüncü bölümde ele alınan sorunlar ve tasarlanan düzeneklerle var olan ve sık kullanılan bilgiişlem ortamı sunan bulutların güvenliğine katkı sunulmuştur. Bu bölümde sunulan iki düzenekten biri yardımıyla bulutu kullanan müşterilerin ağ ve dosya erişimlerinin çalışma zamanında denetlenmesi sağlanır. Diğer düzenek, sıradışı işlemler yürüten müşterilerin istatistiksel yöntemlerle belirlenmesine ve engellenmesine; böylece buluttaki bilgiişlem kaynaklarının yok yere tüketilmesinin önüne geçilmesine yöneliktir.

Dördüncü bölümün temel katkısı, bilgiişlem ortamı sunan bulutun tüm güvenlik sorunlarını bütüncül bir yaklaşımla çözmesidir. Sonuçta, var olan pek çok düzenek listelenmiş, bir arada nasıl çalışacakları belirlenmiş ve daha yetenekli düzeneklerin oluşturulması için bir temel hazırlanmıştır. Bu bölümde asıllama, yetkilendirme, gizlilik, denetlenebilirlik, özlük bilgilerinin korunması, veri yitimi, hizmet yitimi, programların işleyişinin gizliliği, programların yalıtımı, standartlaşma konularındaki sorunlar belirlenir. Bu sorunların pek çoğu bu bölüm içinde anılan düzeneklerle çözüme kavuşturulur. Özlük bilgileri vekil sertifikalar aracılığıyla korunur. Haberleşme gizliliği ve asıllama taşıma katmanı güvenliği (TLS) ile kotarılır. Veri ve hizmet yitimini engellemek için Bizans yetersayı kümeleri önerilir. Bunlardan başka, yetkilendirme, gizlilik, program yalıtımı sorunlarını bir arada çözen yaklaşımın hazırlıkları yapılır.

Beşinci bölümde buluttaki konaklar üzerinde kolaylıkla yer değiştirebilen, veri ve ayarlarıyla birlikte kozalanmış, bununla birlikte güvenlik ödün verilmeden gerektiğinde izinler ve ayarlar doğrultusunda farklı kullanıcılar adına da çalışabilen programların nasıl bir düzenekle kullanılabileceği gösterilmiştir. Bu düzeneğin en önemli katkısı, bilgiişlem ortamı sunan bulutlarda uygulanandan farklı olarak izlekler yerine

yeni bir yaklaşım önermesi, süreçlerin kullanılmasına olanak vermesidir. Böylelikle müşterinin etkileşimli programlarını bulutta yürütmesi, eşzamanlı algoritmalarını aynı sürecin içinde çalıştırması, toplu işlerini art arda çalıştırması kolaylaşacaktır. Bunların ötesinde, sözü edildiği gibi, bilgiişlem ortamı sunan bulutların yetkilendirme, gizleme, program yalıtımı sorunları bir arada çözülür. Program işleyişinin gizliliğini sağlamak üzere geliştirilecek çözümler için güvenilir bir dayanak oluşturulur.

Denetlenebilirlik sorunu altıncı bölümde çözülmüştür. Bu bölümde ele alınan, birbirine tam güveni olmayan iki tarafın gerçekleşen bir olayın kaydını güvenilir biçimde tutmaları sorunudur. Bu genel sorun, tarafların birbirine güveninin kesin olmadığı bulut gibi ortamlarda günlük kayıtlarının nasıl tutulacağı sorununu da kapsar. Bu sorundan yola çıkılarak bu genel sorunu çözen bir düzeneğin tasarımı ve doğrulanması yapılmıştır. Bu düzenek ile bulutta gerçekleşen olayların, taraflardan hiçbirinin yadsıyamayacağı kanıtlarıyla birlikte tüm taraflarda kaydı oluşur. Bu kayıtlar hem tüm tarafların hem de –içeriğini görmemekle birlikte– bağımsız üçüncü tarafların denetimi altındadır. Bu düzeneğin en önemli katkısı, güven ilişkilerinin sorgulanabilir olduğu bulutta, günlük kayıtlarını güvenildiği şüpheli bir tarafın denetimine bırakmak yerine tarafların gerçekleştirdikleri eylemleri kanıtlanabilir ve yadsınamaz günlük kayıtlarıyla saptanabilir biçime getirmesidir.

Üçüncü ve dördüncü bölümlerde sunulan katkılar uluslararası konferanslarda birer bildiri olarak sunulmuş, beşinci ve altıncı bölümlerde yapılan katkılar da uluslararası hakemli dergilerde birer makale olarak basılmıştır.

Tezde, işlem gücünün süreçler yoluyla kullanıldığı bilgiişlem ortamı sunan bulutlara ağırlık verilmekle birlikte, bulut kavramı ile birlikte ortaya çıkan buluta özgü güvenlik tehditleri belirlenmiş; bu tehditlere karşı yenilikçi önlemler geliştirilmiş; belirlenen tüm tehditleri bütünlük içinde önlemeye yönelik, olurluğu yüksek, kullanışlı *bilgiişlem ortamı sunan bulut için güvenlik düzenekleri* tasarlanmıştır.

Tez, hem var olan ve kullanılan bilgiişlem ortamı sunan bulutlara yaptığı katkılarla hem de bilgiişlem ortamı sunan bulutların kullanımı üzerine getirdiği yeni öneriler ve bilimsel literatüre kattığı yenilikçi güvenlik düzeneği tasarımlarıyla bilgiişlem ortamı sunan bulutların güvenliği konusunda kapsamlı bir çalışmadır.

SECURITY MECHANISMS
FOR
PLATFORM AS A SERVICE CLOUD

SUMMARY

This doctoral study is aimed to design secure mechanisms for Platform-as-a-Service clouds. During the design of the security mechanisms, the thesis brings integrated solutions to the many aspects of the security of the Platform-as-a-Service clouds. Previous studies are shown that security is the most important weakness of clouds. Moreover, security is counted as the primary concern of prospective customers as well as draws vast amount of attention from academic world.

According to the common definition of NIST, "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

One of the service models according to the given definition is Platform-as-a-Service, which is the focus of the thesis. The same source defines Platform-as-a-Service as "The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment."

The thesis consists of seven chapters. The first chapter defines and introduces the cloud.

The second chapter explains the evolution, development and history of the cloud. Moreover, this chapter discusses the coverage of security in the cloud, its contributions to security, search for the origins of the security problems, defines security terms, makes a brief introduction to proposed solutions to security problems. Finally, it defines the outline of the thesis.

The third chapter is an attempt to fix the common security problems of concurrent Platform-as-a-Service clouds. Currently deployed Platform-as-a-Service clouds provide computational power by threads. Thread isolation is the main concern of security in such of clouds. The reason behind is to not to interfere the data of the tenants of the cloud. Better ways of isolating threads are sought and limits of isolation are shown in the third chapter.

The main proposition of the fourth chapter is utilization of processes instead of threads. Processes are generally much more convenient and easier to use. A unified design to integrates many security solutions is presented in the chapter. Possible problems and solutions to these problems are enlisted. However, the design has two flaws. First, it is not clear how to run processes in the cloud within their context. Second, an undeniable and fair logging mechanism does not exist to record events occurred in the cloud. The design is enhanced with novel mechanisms to fix these issues. Each of these is explained separately in their own chapter.

In the fifth chapter, methods of securely executing the processes in the cloud are explained. Execution and isolation of processes in the cloud as well as management of access control and governance of permissions to the cloud resources are solutions that are integrated in process containers. Cryptologic mechanisms to adopt these solutions are also covered in this chapter.

In the sixth chapter, a mechanism where the service providing party and the service customer in the cloud are treated fairly is introduced. In this mechanism occurred events are neutrally and undeniably recorded to logs. This is an alternative to problematic conventional logging mechanisms in which only one of the parties is controlling the records. The proposed mechanism may ease to solve the potential conflicts in between the parties.

Not only security measures are taken for concurrent Platform-as-a-Service clouds, but also novel Platform-as-a-Service paradigms have been proposed. Proposed security mechanisms discuss every aspect of the cloud security within the given context of the thesis and solutions are proposed for every enlisted security problem. The main contributions of the thesis are stated in the following paragraphs.

The main contribution of the third chapter is to the concurrent and frequently used Platform-as-a-Service clouds. First of the two mechanisms in this chapter enables controlling the customer programs' access to the network and to the file system on the cloud provider host. The second mechanism statistically detects the customer who execute anomalous sequences of codes. Upon detection, those code sequences could be prevented from running; therefore, cloud resources are not consumed in vain.

The main contribution of the fourth chapter is the integrated approach to solve all security problems of the Platform-as-a-Service clouds. As a results, it enlists many mechanisms at once, defines how to orchestrate them, and builds the foundation for more advanced mechanisms. This chapter successfully defines the problems of: authentication, authorization, secrecy, auditability, privacy, data loss, service loss, secure program execution, program isolation, and standardization. Many of the problems among these are successfully covered within the chapter. Privacy is handled with proxy certificate authorities in a public key infrastructure. Authentication and communication secrecy is obtained via transport layer security (TLS). Data and service loss is prevented with the help of Byzantine quorum systems. Moreover, preparations for the new mechanism to collectively solve authorization, secrecy, program isolation problems are made.

A mechanism is presented to execute programs as processes in the Platform-as-a-Service clouds in the fifth chapter. In this mechanism: the programs can easily migrate

on the hosts within the cloud, they are encapsulated together with their data and settings, they can be executed by different users without compromising security. The most important contribution of this mechanism is, it proposes a new paradigm in contradiction to the concurrent Platform-as-a-Service cloud applications and features processes instead of threads. As a result, customers can run their interactive programs in the cloud, they can submit batch tasks to the cloud or parallel jobs can be executed more conveniently. Furthermore, this approach successfully solves the aforementioned authorization, secrecy and program isolation problems of Platform-as-a-Service clouds as well as prepares an extendable scope for secure program execution solutions.

Auditability problem is solved in the sixth chapter. Two parties who do not completely trust each other but willing to keep reliable records of occurred events is the main topic of this chapter. This generic problem includes the problem of how a customer and a cloud provider keep log records in the cloud environment where they are not certainly trust each other. Originating from this specific problem, the generic problem is solved and verified in this chapter. The proposed mechanism generates undeniable copies of log records of occurred events for each party. The correctness of these log record can be verified by any of the party or even by a third party –however, the contents of the records are hidden from the third party. The most important contribution of this mechanism is, it renders the actions detectable with the help of the provable undeniable logs instead of leaving the records to the control of one of the parties who is doubtfully trusted.

The contributions presented in the third and fourth chapters are published in international conference proceedings. The contributions presented in the fifth and sixth chapters are published in international journals.

As a result of the thesis, a paradigm shift is proposed from concurrent thread based Platform-as-a-Service clouds to process based ones. Both methods of providing computational power are compared throughout the thesis and benefits of using process based approach is presented. In the thesis, *security mechanisms for Platform-as-a-Service cloud* that are aimed to feasibly and conveniently solve all defined problems in an integrated manner are designed.

The thesis is comprehensive in the subject of Platform-as-a-Service with its contributions to concurrent Platform-as-a-Service clouds as well as proposing novel paradigms for using Platform-as-a-Service clouds and designing innovative security mechanisms for them.

1. GİRİŞ

Bu bölümde tez çalışmalarının konusunu oluşturan bulutun tanımı verilmiş, bulutlar sınıflandırılmış, tez çalışmalarını başlatan bilimsel ortam ve tezin içeriği belirtilmiştir.

1.1 Bulutun Tanımı

Bulut, yeni bir teknoloji değil, var olan teknolojinin yeni bir yolla kullanılması [1], bilgiişlemin bir hizmet olarak sunulmasıdır [2].

Bulut, pek çok farklı tanımı olan, bununla birlikte pazarlanırken zaman zaman olduğundan farklı biçimde tanıtılan ve adlandırılan bir kavramdır. Bu nedenle birçok farklı tanımın kullanılmasıyla anlatımın zenginleşeceği öngörülmüştür. Bilişim endüstrisinin önde gelen danışmanlık şirketlerinden Gartner, bulutu şöyle tanımlar: *"bulut bilgiişlem, bilişim yeteneklerinin İnternet teknolojileri kullanılarak ölçeklenebilir ve esnek hizmetler olarak sunulduğu bir bilgiişlem tarzıdır"* [3]. Doğal olarak, endüstrinin bakış açısı kullanımda karşılaşılanı dolaysız yansıtmaktadır. Bulutun tarihi açısından erken dönemde yapılan akademik tanımlardan biri Vaquero v.d. tarafından 2008'de yazılmıştır: *"Bulutlar, kolay kullanılabilen ve erişilebilen (donanım, geliştirme ortamı ve hizmetler gibi) sanallaştırılmış geniş kaynaklardır. Bu kaynaklar istek üzerine yüke (ölçeğe) göre koşullanır, böylece en uygun kaynak yararlanımı sağlanır"* [4]. Daha kapsamlı bir tanım izleyen yıl Buyya v.d. tarafından dile getirilmiştir: *"Bulut, kendi aralarında bağlantılı ve sanallaştırılmış bilgisayarlardan oluşan, paralel ve dağıtık bir bilgiişlem dizgesidir. Bu dizge istek üzerine hizmete sokulup çıkarılabilir ve müşteri ile sağlayıcı arasında pazarlığı yapılacak hizmet sözleşmesi*[1] *uyarınca bir ya da daha çok bilgisayardan oluşuyormuş gibi sunulabilir"* [5]. 2010 yılında yaygın onay gören tanımlardan biri ise NIST tarafından yayınlanan bir raporda geçer: *"bulut bilgiişlem, paylaşılan bir havuz içindeki yapılandırılabilir, en az*

[1] İngilizcesi: "service level agreement", kısaltılarak "SLA" biçiminde de kullanılır.

yönetimsel çaba ya da etkileşim ile, hızla, hizmete sokulabilen ve hizmetten çıkarılabilen bilgiişlem kaynaklarına (ağ, sunucu, depolama, yazılımlar ve hizmetler) yaygın ve kullanışlı yollarla, ağ üzerinden, istendiği an erişim sağlayan modeldir" [6]. Burada sayılan ve sayılmayan birçok tanımı bir araya getirerek bunların ortak noktalarını belirlemeye yönelik bir çalışma yapan Voorsluys v.d. şu dört özelliği saptayan bir başka tanımlama çalışması yapmıştır [7]:

- Müşterinin kullandığı kadar ödediği hizmetler
- Esnek sığa ve sonsuz hesaplama kaynağı olduğu algısı
- Müşteri istediğinde hizmet sunabilen arayüz
- Soyutlanmış veya sanallaştırılmış hesaplama kaynakları

Burada sunulan tüm tanımlarda *esneklik*, *ölçeklenebilirlik*, *değişken sığa*, *kolay erişilebilen* sözleri ile sezdirilen ancak açıkça belirtilmeyen bir başka özellik de bulutta sunulan hizmetlerin müşterinin konumundan bağımsız olmasıdır. Konumdan bağımsızlık burada sayılan özelliklerin sonucunda ortaya çıkar; örneğin kolay erişilebilen, kendiliğinden kaynak ataması yapabilen dağıtık bir dizgeyi kullanan müşteri dizgeye her zamankinden başka bir noktadan bağlandığında bilgiişlem kaynakları aynı başarımla kullanılabilir.

1.2 Bulutların Sınıflandırılması

En genel anlatımıyla, bulut, kullanılan bilgiişlem işlevini bilgiişlem altyapısından soyutlama ilkesine dayanır. Sözü edilen bilgiişlem altyapısı, kullanılacak işleve göre değişik düzeylerde soyutlanır. Örneğin, bir işletim sistemi onu taşıyacak fiziksel bilgisayardan ya da bir yazılım onu barındıracak işletim sisteminden soyutlanabilir. Bu soyutlamanın türüne göre bulut için en sık kullanılan sınıflandırma olan hizmet modelleri belirlenir: sanal bilgisayar sunan bulut, bilgiişlem ortamı sunan bulut ve hazır yazılım sunan bulut [6].

Sanal bilgisayar sunan bulut (IaaS) Bu hizmet modelinde müşterinin fiziksel altyapı ile ilgili bir denetimi yoktur. Ancak işletim sisteminin kendisi ve bunun üzerindeki tüm programlar müşterinin denetimindedir. Bununla birlikte, seçilmiş bazı ağ ayarları, örneğin güvenlik duvarı ayarlarının bir bölümü, müşterinin denetimine açılabilir.

Bilgiişlem ortamı sunan bulut (PaaS) Bu hizmet modelinde müşterinin işletim sistemine ve üzerindeki programların pek çoğuna erişimi yoktur. Müşteri hazırladığı ya da edindiği programları sağlayıcının sunduğu bilgiişlem ortamlarında çalıştırır. Bir gereklilik olmamakla birlikte, seçenek olarak, sağlayıcı müşterilerine uygulama geliştirmelerini kolaylaştıracak araçlar (metin düzenleyiciler, derleyiciler, kitaplıklar, vb.) sunabilir.

Hazır yazılım sunan bulut (SaaS) Bu hizmet modelinde müşteri sağlayıcının hazırlamış ve sunmakta olduğu programı kullanır. Müşterinin programı yapılandırma yetkisi yoktur, ancak sağlayıcının izin verdiği ölçüde kişiselleştirebilir.

NIST raporunda yapılan bu sınıflandırma oldukça başarılı olmuş ve yayınlanmasının ardından pek çok bulut hizmeti sonuna "aaS"[2] getirilerek adlandırılmıştır.

Diğer bir sınıflandırma da bulutun kime hizmet verdiği üzerine kuruludur [6]. Bu sınıflandırmaya göre bulut dörde ayrılır: özel bulut, topluluk bulutu, açık bulut, melez bulut. Bu sınıflandırmada bulutun parçaları fiziksel olarak birden fazla konumda bulunabilir ya da bazı parçaları yüklenici şirketlere bırakılmış olabilir. Sınıflandırmada bunlar önemsenmez, hizmetin kime verildiği gözetilir.

Özel bulutlarda hizmet tek bir kuruma verilir. Topluluk bulutu benzer amaçlar güden bir topluluğun hizmetindedir. Açık bulut herkesin kullanımına açılmıştır. Sağlayıcı tüm müşterilere hizmet vermeye çalışır. Bu günümüzde en yaygın olan bulut türüdür. Ticari bulutların hemen hepsi bu sınıftadır. Melez bulut ise bu sınıflandırmada sayılanlardan birkaç örneğin bir arada çalışmasıyla ortaya çıkan buluttur.

[2]Her üç hizmet modelinin de İngilizcesinde kullanılan "aaS" (açık biçimiyle "as-a-Service") sözü "hizmet olarak" biçiminde Türkçeleştirilebilir. Hizmet modellerinin İngilizce uzun yazımları için xiii. sayfadaki kısaltmalara bakınız.

1.3 Bulut Olma Koşulları

Bulut için verilen tanımların bugüne dek bilişim teknolojisi adına yapılan her şeyi kapsadığı eleştirileri yapılmıştır [8]. Bu eleştirilerden sıyrılmanın yolu bir dizgenin bulut sayılması için taşıması gereken koşulları [6] açıkça belirtmektir:

İstendiğinde hizmet sunabilme Müşteri, yardıma gerek duymadan, istediği an buluttan hizmet alabilmelidir.

Yoğun ağ kullanımı Bulut kaynaklarına ağ[ın herhangi bir noktası] üzerinden çeşitli ve alışıldık donanımlar aracılığıyla erişilebilmelidir.

Kaynak paylaşımı Sağlayıcının sanal ya da fiziksel bilgiişlem kaynakları pek çok müşteri arasında paylaşılmalıdır. Birçok müşteri aynı fiziksel kaynağa atanabilmeli, atamalar müşterilerin isteğine göre ya da bulut yönetimince düzenlenebilmelidir.

Yüksek esneklik Kaynaklar, istekleri karşılamak üzere, hızla, kendiliklerinden hizmete girmeli ve hizmetten çıkabilmelidir. Müşteri için bulutun bilgiişlem kaynağı sınırsız izlenimi vermelidir.

Hizmetin ölçülmesi Bulutta sunulan hizmet hem sağlayıcı hem de müşteri için rahatlıkla izlenebilir, ölçülebilir, raporlanabilir, denetlenebilir olmalıdır.

1.4 Güdülenme

Önceki bölümde anlatılan koşullarla anlamlandırılan bulut kavramında, genel güvenlik sorunlarının yanında altyapının soyutlanmasından ve tanımı gereği bulutun taşıması gereken özelliklerden kaynaklı yeni güvenlik sorunları ortaya çıkar [9].

Sözü edilen hizmet sınıflarından, sanal bilgisayar sunan bulut için güvenlik gerekleri büyük ölçüde sanal bilgisayarların güvenliğine ve bilginin sağlayıcıya devredilmesinden kaynaklı sorunlara dönüştürülebilir.

Hazır yazılım sunan bulut için güvenlik gerekleri de İnternet güvenliği sorunlarına benzer. Erişim *de facto* standart olan İnternet ağı ile sağlanacağından müşteriye ve onun

kullanıcılarına[3] sunulan arayüz alışıldık bir ağ yöresidir. Hazır yazılım sunan bulut böylece, iyi ve kötü yönleriyle İnternet'in bilinen güvenlik özelliklerini taşır. Bunun yanında, müşteri verisinin sağlayıcının bilgisayarlarında bulunması ve işlenmesi hazır yazılım sunan bulutlardaki güvenlik sorunlarının temelini oluşturur.

Alışıldıktan farklı güvenlik sorunları ve gerekleri olan bulut hizmet sınıfı, bilgiişlem ortamı sunan bulutlardır. Bu hizmet sınıfındaki bulutlarda sadece verinin bulunduğu ve işlendiği yer nedeniyle değil, verinin işlenmesi sırasında ortaya çıkacak durumların karmaşıklığı nedeniyle de güvenlik açıkları belirir. Çok sayıdaki güvenlik açığının varlığı araştırıcılar için geniş bir çalışma alanı oluşturur. Bu nedenle bu konuya yoğunlaşmak uygun bulunmuştur.

1.5 Tez İçeriği

Bulutun tarihçesi ve güvenlik sorunları 2. bölümde anlatılmıştır.

Bilgiişlem ortamı sunan bulutlarda, bilgiişlem ortamı sunmanın birden çok yolu vardır. Önemli görülen iki yoldan biri bilgiişlem ortamının izlekler biçiminde sunulması, ikincisi ise süreçler biçiminde sunulmasıdır. Bu yöntemlerin her ikisi de tezde ele alınmıştır. 3. bölümde belirtilecek nedenlerle bilgiişlem ortamı sunan bulutlarda süreçlerin kullanılmasının daha uygun olduğu belirlenmiştir.

Bilgiişlem ortamının izlek kullanılarak sunulduğu durumda kullanılabilecek güvenlik düzenekleri 3. bölümde ele alınmıştır.

İzlekler kullanılarak sunulan bir bilgiişlem ortamının kullanımının kısıtları göz önüne alınarak, genel kullanıma uygun ve bulut kavramını daha iyi karşılayan süreç tabanlı bir bilgiişlem ortamı sunan bulutun kullanımına yönelik güvenlik düzeneklerinin tasarımları 4. bölümde yapılmıştır. Bilgiişlem hizmeti sunan süreç tabanlı bir bulutun neden olacağı güvenlik sorunları tanımlanmış ve bunlara çözümler önerilmiştir. Çözüm önerilerinin bazıları uygulamada sık kullanılan, iyi bilinen kavramlardır. Önerilen diğer çözümlerse buluta özgü sorunları ortadan kaldırmak üzere getirilen yenilikçi

[3]Tez boyunca bulutun üç temel paydaşından sıkça söz edilmiştir. Bulut altyapısını kuran ve bunu para karşılığı ödünç veren "sağlayıcı", bulut altyapısını parayla bir süreliğine ödünç alıp burada programlarını çalıştıran "müşteri" ve bulut üzerinde yürütülen bu programları kullanan ancak sağlayıcı ile olasılıkla doğrudan parasal ilişkisi bulunmayan "kullanıcı".

yaklaşımlardır. İzleyen iki ayrı bölümde önerilen yenilikçi yaklaşımlar ayrıntılarıyla incelenmiş, iyi bilinen kavramlara ise daha fazla değinilmemiştir.

Genelleştirilmiş bir tasarımla süreç tabanlı bilgiişlem ortamı sunan bulutlarda süreçlerin yalıtılması 5. bölümde ayrıntılarıyla gösterilmiştir. Bu bölümde bir sürecin diğer süreçlerden ve bulut sağlayıcısının denetimindeki konaklardan nasıl yalıtılacağı, programın işletildiği sırada gerekli olan verilere nasıl erişeceği, programın ayarlarının nasıl yapılacağı, giriş ve çıkışları hangi yolla yapacağı ve bu sırada erişim izinlerinin nasıl denetleneceği açıklanmıştır.

Bulutta taraflar arasında karşılıklı güveni sağlamak için önemli bir yöntem, gerçekleşen olayların kayıtlarının yadsınamaz bir biçimde tutulması ve güvenli bir ortamda saklanmasıdır. Bu konuda geliştirilen güvenli günlük tutma düzeneği 6. bölümde yer almıştır.

Tez, yorum ve tartışmaları içeren 7. bölüm ile sonlandırılmıştır.

1.6 Tezin Sunduğu Katkılar

Bilgiişlem ortamı sunan bulut için tez içinde anılan güvenlik düzenekleri var olan bulutların güvenlik sorunlarını çözmeye yönelik çözümler sunmanın dışında yeni tasarlanacak bilgiişlem ortamı sunan bulutların ne yönde değişikliklerle daha güvenli olacağını da irdeler. 1.5 bölümünde sayılan içerikle oluşturulan tez bilimsel literatüre de dört yayınla katkı sunmuştur [9–12].

Üçüncü bölümde var olan bilgiişlem ortamı sunan bulutların sorunları ele alınır. Önde gelen sorunlardan biri olan müşteri programlarının birbirinden etkilenmeden çalışmasına yönelik katkılar sunulur.

Dördüncü bölümde baştan tasarlanacak bir bilgiişlem ortamı sunan bulutta çözüm olarak yer alacak güvenlik düzenekleri araştırılır. Bilgiişlem ortamı sunan bulutlara özgü güvenlik sorunları saptandıktan sonra var olan ve bilinen pek çok güvenlik düzeneği uyumlu biçimde bir araya getirilerek bilgiişlem ortamı sunan bulutta kullanılır. Var olan yöntemlerle çözüm bulunamayan zorlu güvenlik sorunlarının çözümü

için kapsamlı düzeneklerin uygulanabileceği ve bilinen yöntemlerle nasıl bütünleştirilebileceği gösterilir.

Beşinci bölümde, dördüncü bölümde öngörülen bilgiişlem ortamı sunan bulutla uyum içinde çalışacak standart bir program ve veri kozalama yapısı sunulur. Böylece bilgiişlem ortamı sunan bulutta yer alan programın farklı kullanıcılar adına, farklı ayarlarla, farklı verilerle, gerektiğinde farklı konaklara göçerek, güvenliği zedelemeden nasıl çalışacağı gösterilir.

Altıncı bölümde bulutlardan esinlenilerek çözülmüş genel bir sorundan söz edilir. Karşılıklı tam güven duymayan iki tarafın bir hizmet alımı senaryosunda ortaklaşa bir belge ile hizmet alımının kaydını oluşturması sorununu çözen bir düzenek oluşturulmuştur. Genelleştirilmiş bu sorun bilgiişlem ortamı sunan bulutlarda sağlayıcı ve müşteri arasında güvenli günlük kayıtlarının tutulması sorununu da çözer. Bu çözümde günlük kayıtlarının varlığı ya da kaynağı yadsınamaz. Günlük kayıtlarının bir saldırı anında silinmesine karşı farklı yerlerde bulunan farklı üçüncü taraflarda kopya tutulması sağlanarak önlem alınmıştır. Günlük kayıtlarının tutulması ya da oluşturulması sırasında taraflardan herhangi birini kayıracak bir durum oluşturulamaz. Kayıtların doğruluğu herkesçe denetlenebilir.

Tez, hem var olan ve kullanılan bilgiişlem ortamı sunan bulutlara yaptığı katkılarla hem de bilgiişlem ortamı sunan bulutların kullanımı üzerine getirdiği yeni önerilerle; bunların yanında özgün güvenlik düzeneği tasarımlarıyla bilgiişlem ortamı sunan bulutların güvenliği konusunda kapsamlı bir çalışmadır.

2. BULUTUN EVRİMİ VE GÜVENLİĞİ

Bu bölümde bulutun ortaya çıkışı ve tarihçesi anlatılmıştır. Bunun ardından bulutun güvenliği birkaç alt başlıkta tartışılmıştır. Bunların başlıcaları şöyledir: bulutun güvenliğe katkıları, bulut güvenliğine konu olan taraflar, güvenlik sorunlarının kaynakları, bulut güvenliğine duyulan ilgi, bulut güvenliğindeki sorunlar ve bunlara karşı önerilen çözümler.

2.1 Bulutun Tarihçesi ve Evrimi

Bulut düşüncesi ilk kez 1965 yılında Multics projesi kapsamında ortaya atılmıştır [13]. Başarıya ulaşmayan projenin amacı "sürekli çalışan, telefon ya da elektrik şebekelerine benzer, insan denetiminde ya da kendi kendine çalışabilen, uzaktan yönetilebilen bir bilgiişlem aracı" geliştirmektir. Projenin adında[1] ve amacında belirtildiği gibi proje kapsamında geliştirilmesi öngörülen dizgenin buluta benzediği açıktır. Öncelikle, bilgiişlemin bir hizmet olarak sunulması görüşü ilk kez dile getirilmiş ve gerçekleşmesi için çaba harcanmıştır. Günümüzde kullanılan çok kullanıcılı işletim sistemlerinin pek çok ilkesi ilk kez bu proje kapsamında tartışılmış; örneğin kullanıcılar arasında veri yalıtımının nasıl gerçekleştirileceğinin temelleri ya da izinlerin nasıl kullanılacağı bu projede ortaya çıkmıştır.

Multics'in geliştirildiği ortam incelenirse 1963'de MIT'de ARPA desteğiyle başlatılan ve daha sonra tek başına enstitünün en büyük bilgisayar araştırma laboratuvarına dönüşen Project MAC göze çarpar [14, 15]. Projede Marvin Minsky, John McCarthy, Joseph Carl Robnett Licklider, Robert Fano, Fernando José Corbató çalışmaktadır. İlk kez 1957'de önerilen [16–18] zaman paylaşımlı dizgelerin Corbató tarafından 1961'de geliştirilen bir örneği olan CTSS [19] bu projede kullanılır. Proje kapsamında zaman paylaşımlı, uzaktan erişilebilen, çok kullanıcılı, kaynak paylaşımlı bir dizge geliştirilir.

[1] xiii. sayfadaki kısaltmalara bakınız.

Licklider'ın 1963'de yazmış olduğu "Galaksilerarası Bilgisayar Ağı" temalı öngörüleri günümüzdeki bulutun gerçeğe yakın kullanım senaryolarını içerir [20].

Bir dizge olarak bulutun ilk örneği olarak Multics gösterilse de, günümüzdeki bulutta sıklıkla kullanılan pek çok teknoloji Multics'ten çok daha sonra geliştirilmiştir. Sanal bilgisayarları ortaya atanlar 1973'te Madnick ve Donovan olur [21]. Multics ile bilgiişlemin bir bütün olarak soyutlanmasından, sanal bilgisayarlarla da donanımın soyutlanmasından söz edilmiştir. Verinin soyutlanmasını ilk düşünenlerse 1974'te Barbara Liskov ve Stephen Zilles olur [22]. Böylece bilgiişlemin değişik bileşenlerinin soyutlanmasına ilişkin tüm örnekler tamamlanır.

Tanımı verilen güncel buluta az ya da çok benzeyen pek çok dizge aradan geçen yıllar boyunca tasarlanmış, kurulmuş ve kullanılmıştır. Yazılım ve donanım alanındaki dikkate değer ilerleme ile bilgisayar teknolojisinin ucuzlaması güncel bulut dizgelerinin ortaya çıkması için verimli bir ortam hazırlamıştır. Bulutun altyapısında önemli yer tutanlardan biri de sürekli gelişen ağ teknolojileri olmuştur. Özellikle doksanlı yıllarda İnternet'in kazandığı yaygınlık, bununla birlikte aynı yıllarda ortaya çıkan telsiz telefon teknolojilerinin yaygınlaşması ve bu iki teknolojinin iki binli yıllarda birlikte kullanılması ağ teknolojilerinin geliştirilmesi için dikkate değer bir ekonomik sinerji yaratmıştır.

Bulut günümüzdeki biçimiyle oluşana dek pek çok benzer teknoloji ortaya çıkmış ve benzer amaçlarla kullanılmıştır. Görece en yakın teknolojiler öbekte bilgiişlem[2] [23], örgüde bilgiişlem[3] [24] ve dağıtık bilgiişlem[4] [13] teknolojileridir. Öbekte bilgiişlem yapılırken tek tip ve sıkı bağlı işlemciler bir araya getirilerek süper bilgisayarlar oluşturulur ve tek bir bilgisayarın altından kalkamayacağı zorlukta hesaplar için kullanılır. Tek tip işlemci kullanmanın getireceği zorlukları aşmak için gevşek bağlı ve farklı işlemcilerle kurulabilen örgüde bilgiişlem kullanılır. Bu yöntemde farklı işlemcilerin bir arada çalışması için yazılım ara katmanları kullanılması gerekir. Bunun karşılığında yazılım karmaşıklığı artar. Dağıtık bilgiişlem yapılırken bir veri iletişim ağı üzerinde küçük sığalı bilgisayarlar paylaşımlı olarak tek bir görev üzerinde çalışırlar.

[2] İngilizcesi: "cluster computing".
[3] İngilizcesi: "grid computing".
[4] İngilizcesi: "distributed computing".

Her üç dizgenin de bugünkü bulut kadar esnek olması öngörülmemiştir. Kullanıcı arayüzleri hızlı tepki vermeye yönelik değildir, hatta insan denetiminde olabilir. Eğer ölçülüyorsa sunulan hizmetin nasıl ölçüldüğü iyi tanımlanmamıştır. Ağ kullanımı ilk iki dizgede de sık görülür, üçüncünün tanımında vardır. Kaynakların başka kullanıcılarla paylaşılması ilk iki dizgede de var olmakla birlikte, zorunlu değildir. Müşterinin ya da kullanıcının konumundan bağımsızlık her üç dizgede de belirgin değildir.

Bulutla ilişkili bir başka teknoloji, ilk kullanımı daha eski olmakla birlikte, doksanların ikinci yarısında sıklıkla uyarlanan hizmet odaklı mimari[5] adlı yazılım tasarım kalıbıdır [25]. Bu tasarım İnternet'i daha kullanışlı yapma çabaları sırasında sık kullanılmıştır. Yazılımsal bir arayüzün bir istek ve ona karşılık sunulan bir hizmetle tanımlandığı bu tasarım kalıbı bulutta kullanıcıya sunulan arayüzü andırır.

Bulut terimi günümüzdeki anlamıyla ilk kez 2006 yılında Ağustos ayında duyulmuştur. Ay başında Google şirketinden Eric Schmidt bir toplantıda kabataslak bir bulut tanımı yapmıştır [26], ardından ay sonunda Amazon şirketi sanal bilgisayar sunan bulut olan EC2 hizmetini duyurmuştur [27].

Terimin anlamını kazanması ve açık seçik sınırlarla tanımının yapılması zaman almıştır. Kavramı karşılayan ilk tanım ancak 2008'de Gartner şirketi tarafından yapılmıştır [28]. Belirsiz hizmetlerin bulut olarak adlandırıldığı birkaç yılın ardından daha önceleri farklı adlarla pazarlanan bazı hizmetlerin de bu tanımdan sonra bulut kapsamında değerlendirilmesi gerektiği düşünülmüştür. Böylece bulut hizmeti olarak tarihlendirilen ilk örnekler daha eskilere dayandırılmıştır. Örnek olarak, 1999'da pazara giren Salesforce.com şirketi erken dönem hazır yazılım sunan bulut örneklerinden sayılmaya başlanmıştır.

Terimin kullanımı 2010 yılından 2012 yılına dek bir modaya dönüşmüştür. Örneğin, bu süre içinde İnternet'te "bulut bilgiişlem" sözü, "ağ güvenliği" sözünün beş katı, "örgüde bilgiişlem" sözünün on beş katı sıklıkta aranmıştır[6] [29]. Bu yıllarda "cloud" sözü dikkat çekmek amacıyla kullanılmış, buluta imrenen kişi ya da kurum-

[5] İngilizcesi: "service-oriented architecture".

[6] Gerçekte, bu karşılaştırma İngilizce "cloud computing", "network security" ve "grid computing" söz öbekleri gözetilerek yapılmıştır.

lar tarafından tanımın gereklerini karşılamayan pek çok dizge bulut adıyla pazarlanmıştır [30, 31].

Gartner danışmanlık şirketi 2012 yılında yaptığı kestirimlerde [32], hızla büyüyen sektörün küresel pazar payının 2012 yılında 109 milyar $, 2016'da 206 milyar $ olmasını beklerken 2015 yılında açıkladığı raporda [33] daha önce örneği görülmemiş bir büyümenin yaşandığını belirtmiş ve önceki kestirimlere %30 ekleme yapılması gerektiğini bildirmiştir. Avrupa Birliği 2012 yılında yayınladığı bir basın bülteninde [34] bulutun yıllık getirisinin 160 milyar € olacağını öngörmüştür.

2008 yılından sonra bulutla ilgili akademik yayınlar görülür. 2010 yılından itibaren önemli bir literatür oluşmaya başlar. İlk yayınlardan başlayarak bulut konusundaki akademik ilgi güvenlik konusundaki sorunlara odaklanmaktadır.

2.2 Bulut Güvenliği

Bilgiişlemin buluta taşınması ile güvenliğin nasıl değişeceği çok yönlü bir tartışmadır. Herhangi bir alanda yüklenicilerin varlığıyla ortaya çıkan katkılara ve sakıncalara benzer durumlar bulutun kullanımında da geçerlidir. Sonuçta, bulut, konusunda uzman yüklenici bir kurum tarafından bir bilgiişlem kaynağının hizmet olarak sağlanmasıdır. Ya da, diğer bakış açısıyla, işletim ve yatırım giderlerini düşürmek amacıyla, bir müşterinin bilgiişlem, veri depolama ve güvenlik sorumluluklarını bir yükleniciye bırakmasıdır [35]. Bırakılan sorumluluklar içinden bilgiişlem ve veri depolama ölçülebilir olanlardır, ancak güvenlik doğrudan ölçülemez [36]. Bu nedenle güvenlik ancak hizmetin kalitesi ile ilişkilendirilmelidir.

2.2.1 Bulutun güvenliğe katkıları

Yüklenici kurumun başarısı, uzmanlığı, yeterliği, deneyimi ve bilgisi yüklenilen işin, yani bilgiişlem hizmetinin, niteliğini belirler. Bilgiişlemin güvenli yürütülmesi de bu nitelikler arasındadır. Bulutun sağlayacağı güvenlik hizmeti, ideal durumda, sağlayıcının uzmanlığına bağlı olmak üzere, büyük olasılıkla, müşterilerin kendi başlarına elde edecekleri güvenlik düzeyinden üstün olur.

Buradan yola çıkılırsa, bulutun güvenliğe sağlayacağı en büyük katkı güvenlik konusunda uzmanlıktır. Tek yönetimsel alanda yapılacak güvenlik ayarları, denetim altında tutulan ve sürekli izlenen ağ erişimi, zamanında ve etkileri anlaşılarak yapılan yazılım ve donanım güncellemeleri bulut kullanıcılarını etkilerini kolayca kestiremeyecekleri pek çok bilinçsiz seçimden korur. Ayrıca, güvenlik sorunlarına hızla tepki verilebilmesi ve sürekli denetim, gecikmeden yararlanmayı oldukça iyi bilen bilgisayar korsanlarının işini zorlaştıracaktır.

2.2.2 Bulut güvenliğinden etkilenen taraflar

Bulutla birlikte ortaya çıkan güvenlik sorunları da yine yüklenici varlığından kaynaklanır. Müşteri verilerinin ve kullanılan kaynakların müşteri denetiminde olmaması nedeniyle müşterinin hem kendi özlük bilgilerine hem de kullandığı kaynakların güvenilirliğine etki etmesi zorlaşır. Bu durumda müşteriyi koruyan tek şey sağlayıcıya duyduğu güven olur. Müşterinin verilerini kendi denetiminde olmayan bir alana, dahası güdüsü kâr sağlamak olan bir şirkete bırakması yoğun olarak eleştirilmektedir [37]. Tüm bu eleştirilere karşın, kendi başına verinin denetimini sağlayacak bilgiden yoksun ve ucuz bilgiişlem kaynağı arayışındaki küçük ölçekli müşteriler düşünüldüğünde, ekonomik yönelimin yine de buluta doğru olacağı kestirilmiştir [1].

Bulut güvenliğine sağlayıcı açısından bakılırsa, karşılaşılacak güvenlik sorunları başkadır. Müşterilerden birinin ya da müşterinin programı aracılığıyla bir kullanıcının bulutu kullanarak gerçekleştireceği yasadışı eylemler sağlayıcı için bir sakınca doğurur[7]. Dahası, bir sızmanın ardından denetimi bir bilgisayar korsanının eline geçmiş çok büyük kaynaklar yasadışı eylemler için kullanılabilir.

Son olarak, sağlayıcının sunduğu bulutu birden çok müşterinin paylaştığı düşünülürse, bir kader birliği söz konusudur. Bunun sonucunda bulutun müşterisi olan bir şirkette yaşanan sıradışı bir olayın, örneğin yasal soruşturma sırasında mahkemenin tedbir amaçlı donanıma el koymasının, diğer müşterileri etkilemesi kaçınılmazdır.

[7]Bulutun üç temel paydaşını bir kez daha anımsatmak yerinde olacaktır: Bulut altyapısını kuran ve bunu para karşılığı ödünç veren "sağlayıcı", bulut altyapısını parayla bir süreliğine ödünç alıp burada programlarını çalıştıran "müşteri" ve bulut üzerinde yürütülen bu programları kullanan ancak sağlayıcı ile olasılıkla doğrudan parasal ilişkisi bulunmayan "kullanıcı".

Yukarıda değinilen senaryolar düşünüldüğünde bulutta tüm tarafların güvenlik riski aldığı görülür. Sağlayıcı, müşterilerden ve kullanıcılardan gelecek saldırıların tehdidi altındadır. Müşteriler, sağlayıcıdan, diğer müşterilerden ve kullanıcılardan gelecek saldırıların tehdidi altındadır. Böylece bulutun güvenlik sorunlarını çözmek için sağlayıcının müşteriye karşı; müşterinin de sağlayıcıya karşı uygun yollarla korunması gerektiği ortaya çıkar.

2.2.3 Bulut güvenliği teriminin kapsamı

Bulut güvenliği, konunun tüm paydaşları dikkate alındığında, taraflar arasındaki sözleşmeleri de içeren yasal sorunları, standartları ve uyumluluk konularını kapsar. Bu konular teknik bağlamdaki tartışmaların içinde zaman zaman seziliyor olsalar da konunun ağırlık verilen yanını oluşturmazlar. Tez boyunca bulut güvenliği sözü teknik bağlamda ele alınarak bu sözle beş temel ilke; gizlilik, bütünlük, erişilebilirlik, izlenebilirlik ve özlük bilgilerinin gerektiğince gizli tutulması anlatılmıştır.

2.2.4 Bir buluttaki güvenlik sorunlarının kaynakları

Genelgeçer güvenlik sorunları (Örneğin kullanıcı eğitimi, şifre yönetimi, haberleşme güvenliği, ağ güvenliği, yazılım hataları gibi.) bir yana bırakılırsa buluta özgü güvenlik sorunlarının kaynaklarının belirlenmesi çözümün ilk adımı olmalıdır. Dikkatle incelenirse, bulutun güvenlik zayıflıklarının bulut tanımından kaynaklandığı görülür.

Bulut tanımında sözü edilen hizmet alımı, veri ve işlem gücünün bir yükleniciye bırakılacağını anlatır. Bu, veri ve altyapının devrinden kaynaklanan sorunları ortaya çıkarır.

Kaynak paylaşımı, kaynakları paylaşan müşterilerin aynı fiziksel kaynağın üzerinde bir arada bulunarak birbirlerini etkileyeceklerini söylemektedir. Çok sayıda program bir arada bulunurken bunların bir bölümü ister istemez farklı amaçları nedeniyle uyuşmayacaktır.

Bilgiişlem kaynağının bir hizmet olarak bir yükleniciden satın alınması, işlem gücünün ve veri miktarının yerelde alışılmış boyutların çok üzerinde olacağını gös-

terir. Olasılıkla, bulutu kullanan müşteriler kendi başlarına üstesinden gelemeyecekleri görevleri yükleniciye bırakmak eğilimindedir. Bu nedenle sıradan güvenlik önlemleri, örneğin bir müşterinin verilerinin tamamının tek seferde şifrelenmesi ya da şifresinin çözülmesi ve böylece bulutta saklanması uygulamada kullanışsızdır.

Esnek sözüyle tanımlanan bulutun statik fiziksel altyapı ile karşılanması zordur. Esneklik gereksinimini karşılamak üzere yükleniciler sanallaştırılmış yapılar kurmak eğilimindedir. Bu teknolojiyle bulutun değişik katmanlarında karşılaşılır. Sanallaştırma kullanmanın sonucunda, sanal bilgisayarların[8], işletim sistemlerinin, süreç sanal makinalarının[9] güvenliği bulutun güvenliğini etkilemeye başlar.

2.2.5 Bulut güvenliğine gösterilen ilgi

Bulutun içselleştirilmesi konusunda yapılan ilk çalışmalardan bazıları International Data Corporation adlı araştırma şirketi tarafından önde gelen şirketlerin bilgiişlem bölümü yöneticileri arasında yürütülen anket çalışmalarıdır. Dikkat çeken iki anket bir yıl arayla 2008 ve 2009 yıllarında yürütülmüştür [38, 39]. "Bulutun en önemli sorununu bildiriniz" dendiğinde 2008 yılında % 74,6 ile en çok "güvenlik" yanıtı verilmiştir. Bir sonraki sene "güvenlik" yine en önemli sorun sayılır ancak bu kez oran % 87,5'e yükselmiştir. Bunu destekleyen yönde, "şirketlerinin hassas verilerinin bulutta asla yer almayacağını" söyleyenlerin olduğu akademik çalışmalarda da bildirilmiştir [40].

Bulut üzerine yapılan akademik çalışmaların güvenlik üzerine yoğunlaştığı daha önce 2.1 bölümünde belirtilmişti. Bunun yansıması Google Scholar [41] veritabanında kolaylıkla görülür. Bu veritabanı akademik yayınlarla birlikte akademik olmayan yayınları da içine alan geniş kapsamlı bir veritabanıdır. Akademik olmayan yayınları içerse de, bulut araştırmaları konusundaki değişimlerin izlenmesi açısından yararı olacağı düşünülerek bu veritabanındaki değerlerin gösterilmesi uygun bulunmuştur. Google

[8] İngilizce "system virtual machine" teriminin karşılığı olarak "dizge sanal makinası" yerine tezde "sanal bilgisayar" terimi yeğlenmiştir. Bu terimle fiziksel bir bilgisayarın mimarisine olduğu gibi öykünen ve üzerinde bir işletim sisteminin çalışacağı bir yazılımdan söz edilir.

[9] İngilizce "process virtual machine" teriminin karşılığı olarak tezde "süreç sanal makinası" terimi kullanılmıştır. Bu terimle tek bir süreci ilgili olduğu programlama dilinin kuralları içinde işletim sisteminden bağımsız bir ortamda çalıştırmaya yarayan yazılım anlatılmaktadır.

Scholar tarafından taranan bulut bilgiişlem[10] konulu yayınların sayısı 2010 yılından 2014'e dek sırasıyla 13 600, 24 700, 37 700, 38 500, 38 300 adettir. Bu değerlere bakılarak konu üzerine araştırmaların ilgi yitirilmeden sürdüğü söylenebilir. Daha dikkat çekici olan aynı yıllarda bu çalışmalar içinden güvenlikle ilgili olanların, yani aynı anda hem bulut bilgiişlem hem güvenlik[11] konulu olanların sayısıdır. Bu yayınların sayısı 7 250, 12 100, 19 100, 22 400, 24 000 olarak sıralanır. Buluta olan ilginin azalmadığı, bununla birlikte, bulut güvenliğine olan ilginin giderek arttığı görülmektedir. Öyle ki, son zamanlardaki yayınların neredeyse üçte ikisi güvenlik ile ilgilidir. Buradan anlaşılan, araştırıcıların bulutun çok geniş uygulama olanaklarını araştırmadan önce var olan önemli güvenlik sorunlarının üstesinden gelmek istedikleridir [36].

Bir başka örnek 2013 yılında yapılan bir taramadır [36]. Bu taramada 2008'den 2012'ye dek bulut üzerine yapılan kapsamlı bilimsel yayınlar, diğer taramalar da dahil olmak üzere, incelenmiştir. Bu taramada adı geçen 504 yayından 322'si güvenlik araştırmalarıdır. Burada da oran yaklaşık üçte ikidir. Bir başka dikkat çekici nokta, bu taramada incelenen 504 çalışma içinde ancak tek bir yayının [42] tezin konusu olan bilgiişlem ortamı sunan bulutların güvenliğinden söz ediyor olmasıdır.

Güvenlik konusunda pek çok yayın varken bilgiişlem ortamı sunan bulutların güvenliğinden söz eden tek bir yayının taramada yer alması ilginç bir durumdur. Bunun nedenlerinden biri daha önce 1.3 ve 1.4 bölümlerinde söz edilen bulutun geniş kapsamı olabilir. Olasılıkla, kapsanan alt konulardan birinde uzman olan kişiler o konunun buluttaki izdüşümünde çalışmaya devam etmişlerdir. Böylece yeni oluşan *bilgiişlem ortamı sunan bulut* alanı görece bakir kalmıştır.

2.2.6 Tezde kullanılan güvenlik terimleri

Bulut güvenliğini derinlemesine tartışmaya başlamadan önce bir kavram karışıklığı oluşmaması için sık kullanılan terimlerin tezde hangi anlamda kullanılacaklarını açıklamak uygun görülmüştür.

[10] İngilizce "cloud computing" konulu yayınlar aranmıştır.
[11] İngilizce "security" konulu yayınlar aranmıştır.

Güvenlik zayıflığı Bir dizgenin olası güvenlik sorunlarına yol açacak özellikleri içermesi

Tehdit Bir etkenin bir dizgeye zarar verecek biçimde güdülenmesi ve davranması

Önlem Bir tehdidin olası bir saldırıya dönüşmesini engellemek için yapılan hazırlık

Risk Bir kişi ya da kurumun bilişim yapısının aynı anda tehdit altında olması ve güvenlik zayıflığının bulunması

Saldırı Bir dizgeye etkin yollarla zarar vermeyi denemek

Savunma Bir dizgeyi bir saldırının etkilerinden saldırı sırasında korumayı denemek

Tezin devamında terimlerin birbiriyle ilişkilendirilmesi için indisler kullanılmıştır. Bu bağlamda güvenlik zayıflıklarının ayrılması için tek indis yeterli olur. Z_1 ile birinci zayıflık anlatılır. Tehditler iki indis ile belirtilir. $T_{1,1}$ ile gösterilen tehdit, birinci zayıflıktan kaynaklanan birinci tehdidi belirtir. $Ö_{x,y,1}$ ile gösterilen önlem $T_{x,y}$ tehdidine karşı alınan birinci önlemi belirtir.

2.2.7 Buluttaki temel güvenlik sorunları ve bunlara önerilen çözümler

Buluttaki güvenlik sorunlarını değerlendirirken boş bir tartışmadan kaçınmak için müşteri ile sağlayıcı arasındaki güven ilişkisinden söz etmek gerekir. Güven, güvenlik için kötüdür [43]. Bu ilkeden yola çıkılırsa, taraflar arasında güvenin hiç olmadığı ya da sınırsız olduğu durumda bulut güvenliği tartışmasını yürütmenin anlamsız olacağı görülür. Çünkü, bu iki durumdan birinde müşteri buluttan hizmet almak istemez, diğerinde ise kendisininmiş gibi benimser. Tezde tartışılan ve gündelik hayatta da gözlenen durumda ise taraflar arasında bir hizmet sözleşmesi oluşturacak bir güvenin olduğu varsayılmıştır. Ancak bu güven sınırsız değildir. Hem müşteri hem sağlayıcı karşı tarafın uygun davranacağını öngörmekle birlikte karşı taraftan gelecek bilinçli veya bilinçsiz saldırılara hazırlıklı olmaya çalışmaktadır.

Bulutta karşılaşılan temel sorunları belirlemek için güvenlik zayıflıklarının listelenmesi ile işe başlanmıştır. Daha sonra bunlara dayanılarak tehditlerin belirlenmesine çalışılmıştır.

Z_1 Müşteri verilerinin sağlayıcı alanına taşınması

Z_2 Müşteri programlarının sağlayıcı alanına taşınması

Z_3 Sağlayıcı alanında kaynakların paylaşılması

Z_4 Bulutta gerçekleşen olayların sorumlularını belirlemenin zor olması

Burada sayılan güvenlik zayıflıkları bulutun diğer özellikleriyle birleştikçe tehditleri oluşturur. Literatürde çok sayıda tehdit sayılmıştır. Bu tehditlerin listelendiği kapsamlı taramalar bulunmaktadır [1, 9–12, 35, 42, 44–60]. Burada tehditler ayrıntılarıyla anlatılmamış, tez kapsamında önemli görülenler gruplanarak verilmiştir.

$T_{1,1}$ Müşteri verilerinin yitirilmesi Müşteri verilerinin sağlayıcı bilgisayarlarına taşınmasıyla müşteri denetiminde olmayan verilerin yitirilmesi tehdidi ortaya çıkar. Tehdidin kaynağı kullanıcılar ya da bilinçli veya bilinçsizce sağlayıcının kendisi olabilir.

$T_{1,2}$ Müşteri verilerinin sızması Müşteri denetiminde olmayan veriler üçüncü kişiler ya da sağlayıcı tarafından okunabilir. İlke olarak, verinin yetkisi olmayan biri tarafından okunması ile sızması arasında fark yoktur.

$T_{2,1}$ Müşteri programının sağlayıcıya zarar vermesi Müşteri kendi oluşturduğu programları bulutta çalıştırma yetkisini elinde bulunduracağından sağlayıcı bu programların bulut altyapısına zarar vermeyeceğinden emin olmak ister.

$T_{2,2}$ Müşteri programının sağlayıcıdan zarar görmesi Müşteri, programları uygun biçimde çalıştığı sürece yapmakta olduğu işin sekteye uğramamasını bekler. Sağlayıcının müşteri programlarına yapacağı bozucu etkilerin engellenmesi gerekir.

$T_{3,1}$ Müşterinin diğer müşterilerden zarar görmesi Kaynak paylaşımından söz edilmesiyle birlikte bir müşterinin (Z_1 ve Z_2 ile belirtilen) sağlayıcı denetiminde bulunan veri ve programlarının sağlayıcı dışında diğer müşteriler ve onların programlarının kullanıcıları tarafından da bilinçli ya da bilinçsizce zarara uğratılması riski doğar.

$T_{4,1}$ Bulutta gerçekleşen olayların hatalı kaydedilmesi Bulutta gerçekleşen olayların hatasız ve yadsınamaz biçimde kaydedilmemesi durumunda gerektiğinde etkinliklerin sorumluları kesin olarak belirlenemeyeceğinden tüm paydaşlar için doğrudan ya da dolaylı tehditler söz konusu olur.

Önceden sayılan tehditlerin her biri çok sayıda saldırıya kaynaklık eder. Bu saldırıların listelenmesi tez kapsamında olanaklı ve gerekli değildir. Günümüze kadar yapılan ve tehditleri listeleyen çalışmalarda saldırıların akademik ilgi gören bölümü de bulunmaktadır [1, 9–12, 35, 42, 44–60]. Bunlardan başka, henüz sözü edilmemiş saldırıların da ortaya çıkacağı öngörülmelidir. Örneğin, güvenlik uzmanlarınca belirlenen saldırıların günü gününe listelendiği küresel çapta veritabanları bulunmaktadır [61]. Bu veritabanları incelenerek bu tehditler kapsamında olan yeni saldırılar var olanların arasına eklenir. Elbette, bunun ardından saldırılara karşılık gelen çeşitli savunmalar da ortaya çıkacaktır. Bu veritabanlarının tez kapsamında sunulması gerekli görülmemiştir.

Yapılacak saldırıların etkinliğini azaltmak ya da saldırı olanaklarını daraltmak için önlemler alınmalıdır. Sayılan tehditlere karşılık gelen önlemler göz önüne alınarak oluşturulmuş, böylelikle saldırılardan olabildiğince arındırılmış bir *bilgiişlem ortamı sunan bulut için güvenlik düzenekleri* tasarlamak bu tezin konusudur. Aşağıda sayılan tehditlere karşı alınacak önlemler listelenmiştir:

$Ö_{1,1,1}$ Bizans yetersayı kümeleri Müşteri verilerini kendi denetiminde olmayan bir alana aktarmakla birlikte onları yitirmek tehlikesiyle karşılaşır. Bu tehlikeden sıyrılmanın yollarından biri, verilerin birden çok kopyasının Bizans yetersayı kümeleri[12] içinde oluşturularak farklı ortamlarda tutulmasıdır. Böylece aynı anda tüm kopyaların yitirilmesi olasılığı azaltılmış olur.

$Ö_{1,2,1}$ Şifreleme Gizli kalması gereken verilerin sadece ilgili kullanıcıların bildiği anahtarlarla şifrelenmesiyle veri sızması engellenir. Bir veriyi hangi kullanıcının okuması ve hangi kullanıcının değiştirmesi gerekeceği sorusunun yanıtını vermek veri buluttayken zordur. Çünkü verinin kendisi, veriyi kullananlar, verinin yeri,

[12] İngilizcede "Byzantine quorum" olarak kullanılan terimin ikinci sözcüğü Latincedir. "Quorum" Latincede "bir kurulun karar alabilmesi için gereken en az sayı" anlamındadır.

hatta veriye erişim hakları zamanla değişebilir. Şifreleme yapısının tüm değişikliklere ayak uyduracak esneklikte tasarlanmış olması gerekir.

$Ö_{2,1,1}$ Güvenli hesaplama katmanı Müşteri programı kendisine sağlanan altyapıyı sağlayıcı ile yaptığı hizmet sözleşmesi koşulları çerçevesinde kullanmalıdır. Bunun sağlanması için müşteri programının davranışlarını sınırlayan bir yazılım katmanı önerilmiştir. Programın başka yollarla bulut kaynaklarına erişimi engellenmişken güvenli hesaplama katmanı sözleşme koşulları ile belirlenen kaynaklardan fazlasını programa sağlamayacak olursa olası kuraldışı kullanımlar engellenmiş olur.

$Ö_{2,2,1}$ Bizans yetersayı kümeleri Sağlayıcının müşteri programına yapacağı bilinçli ya da bilinçsiz herhangi bir bozucu etkiyi önlemenin yolu ancak programın çıktısını güvenilir bir başkasıyla kıyaslamaktır. Bu durumda müşteri önemli olduğuna inandığı hesaplar için birden fazla sağlayıcı ile hesaplama yaparak çıktıları karşılaştırır.

$Ö_{3,1,1}$ Güvenli hesaplama katmanı Müşteri programlarının altyapıdan yalıtılmış olması, birbirlerinin programlarına erişemeyecekleri anlamına gelir.

$Ö_{3,1,2}$ Yetkilendirme Müşterilerin veri ve programlarına belirledikleri erişim haklarına göre yetkilendirme yapılarak sadece izin verilen kullanıcıların izin verilen içeriğe erişmesi sağlanmalıdır.

$Ö_{4,1,1}$ Güvenli günlük tutma düzeneği Bulutta gerçekleşen olayların bağımsız ve hatasız kaydedilmesi sağlanırsa hem hizmet dökümüne hem yasal sorumluluğa ilişkin tarafların yadsıyamayacağı ve güvenilir bir dayanak oluşur.

Burada sayılan tehditlerin bir bölümüne karşı alınacak başkaca önlemlerin olduğu tartışması yürütülebilir. Tezin ilerleyen bölümlerinde her bir önlemin neden seçildiği olası diğer önlemlerle kıyaslanarak açıklanmıştır.

2.2.8 Buluttaki yerleşik güvenlik sorunları

Hemen her bilgisayar dizgesinde var olan ve buluta da taşınan önemli güvenlik zayıflıkları vardır. Bu konular güvenli bir bulutun varlığını tehlikeye sokacak önemli

zayıflıklardır. Bu zayıflıklardan kaynaklanan çok sayıda tehdit ve bunlara karşılık gelen pek çok önlem sayılabileceğinden burada sadece zayıflıklar sıralanmıştır. Tezin devamında tehditler ve önlemler yeri geldikçe buradaki zayıflıklarla ilişkilendirilerek incelenmiştir.

Z_5 Standart arayüzlerin eksikliği

Z_6 Özlük bilgilerinin gizlenememesi

Z_7 Yararlanırlığın sağlanamaması

Z_8 Hata hoşgörüsünün olmayışı

2.3 Vargı

Bu bölümün başında bulutun öncülü olan tarihsel teknolojiler ve bugüne kadarki evrimi gösterilmiş, ardından günümüzdeki anlamıyla ortaya çıkışı anlatılmıştır. Bölümün devamında bulut güvenliğine yönelik tez boyunca kullanılan tanımlar ve açıklamalar yapılmış, terimler tanıtılmıştır. Bulutlara özgü güvenlik zayıflıkları ve bunlara karşı alınabilecek önlemler belirtilmiştir.

Bu bölümde yapılan saptamalara göre 2.2.7 bölümünde belirtilen önlemleri içeren güvenlik düzenekleriyle donatılmış ve 2.2.8 bölümünde belirtilen zayıflıkların oluşturacağı tehditlere karşı alınacak önlemlerle desteklenen bir tasarımla ortaya konacak bir bilgiişlem ortamı sunan bulutun güvenli olması beklenir.

3. İZLEK TABANLI BİLGİİŞLEM ORTAMI SUNAN BULUTTA İZLEKLERİN YALITIMI

Bu bölümde bulutun sunduğu işlemci gücünden yararlanmanın farklı yolları tartışılmış, bilgiişlem ortamı sunan bulutların işlemci gücünün izleklerle kullanıldığı durumlarda müşteri verilerinin bir başka müşterinin kullandığı izleğin denetimine girmemesi için yalıtımının nasıl gerçekleneceği üzerine yöntemler açıklanmış ve bu alandaki geçmiş çalışmalar belirtilmiştir.

3.1 Bilgiişlem Ortamı Sunan Bulutlarda Yalıtımın Gereği

Bilgiişlem yapılan bir bulutta işlemci gücü her ne biçimde sağlanırsa sağlansın, verilerin yalıtılması gerekir. Bunun nedeni, aynı fiziksel altyapının aynı anda ya da yakın zamanlarda birden çok farklı kişiye bilgiişlem hizmeti sunmasıdır. Bilgiişlem hizmeti aynı altyapı üzerinde sunulurken farklı kişilerin verilerinin hatayla ya da kötü amaçlarla bir diğerininkine karışması olasılığı vardır. Bu daha önce 2.2.7 bölümünde *müşterinin diğer müşterilerden zarar görmesi ($T_{3,1}$)* olarak adlandırılan tehdittir.

Müşteri, güvenliğin ilk adımı olarak, bulutta diğer müşterilerden gelecek tehditlerden korunabilirse, daha güçlü saldırganlara karşı savunulması düşünülür. Örneğin, sağlayıcıdan kaynaklanacak *müşteri verilerinin sızması ($T_{1,2}$)* tehdidine karşı bir önlem alınmasına çalışılabilir.

Bu açıdan bakıldığında, müşterinin işlem gücünden yararlanırken kullandığı izlek henüz tam anlamıyla yalıtılmamışken daha ileri düzeyde koruma tasarlamanın bir anlamı olmaz. Örneğin, müşterinin kullandığı veriler, ağ bağlantıları ya da veritabanları uygun biçimde şifrelenmiş olsa bile; uygun biçimde yalıtılmamış bir izlek yoluyla işlem anında müşterinin bu kaynaklara erişmek üzere bellekte sakladığı işaretçilere erişilebilir.

3.2 Bulutun Sunduğu İşlemci Gücünden Yararlanma Yolları

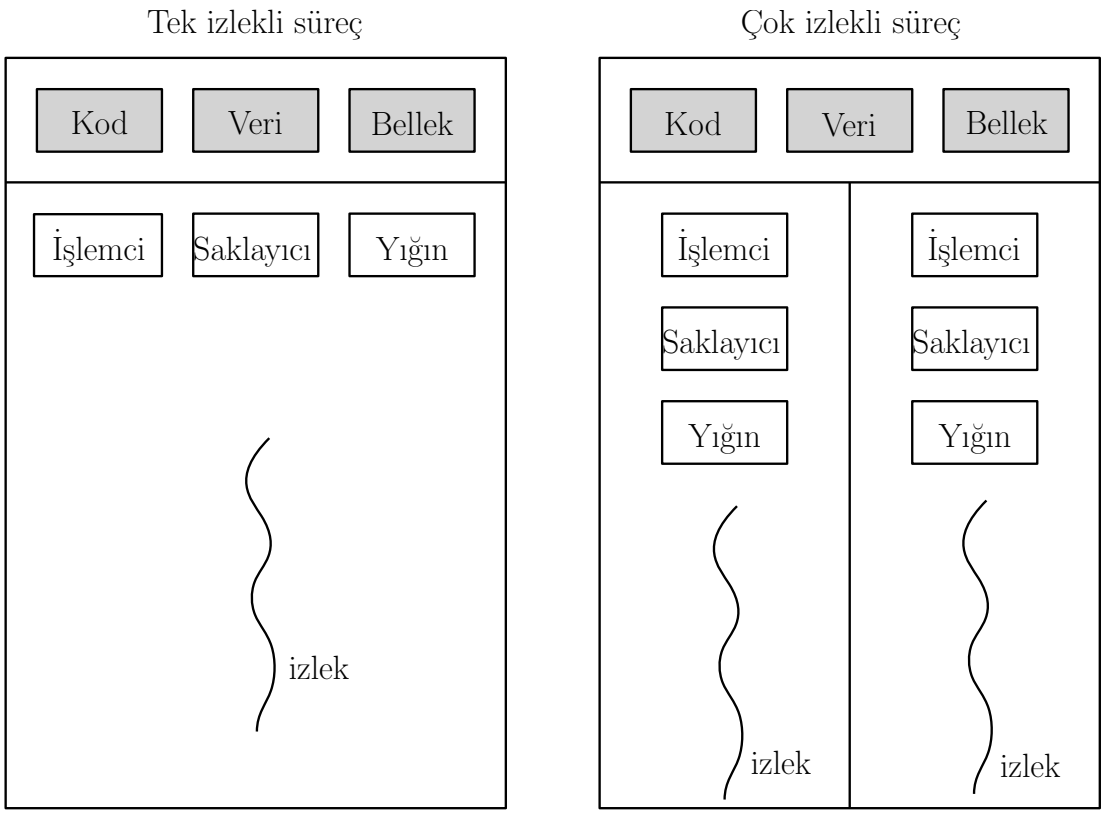
Bilgiişlem ortamı sunan bulutun sağladığı hizmetlerin başta geleni işlemci gücüdür. İşlemci gücünün kullanılmasının uygulamada iki yolu vardır[1]. Müşteri, işletim sistemi aracılığı ile oluşturulan bağımsız bir süreç olarak ya da belli bir sürece bağlı izlekler aracılığı ile işlemci gücünü kullanabilir. İşlemci gücünün süreç ya da izlek olarak kullanılmasının yararları ve olumsuzlukları irdelenerek uygun yöntem belirlenmelidir.

3.2.1 İzlekler ve süreçler

İzlekler ardışıl yürütülen program parçalarıdır, tek başlarına var olmazlar, bir sürecin kapsamı içinde canlandırılırlar. Süreçler ise görevleri yalıtılmış bağlamlar içinde tamamlamaya yönelik programlardır. Çağdaş işletim sistemlerinde her bir süreç bir ya da birden fazla izlekten oluşur. Bir görevi tamamlamak için bir izleğin ardışıl yürütülmesi yeterliyse ilgili süreç tek bir izlek kullanır. Daha karmaşık görevler eşzamanlı çalışan izlekler gerektirir; bu durumda birbirleriyle haberleşerek görevi ortaklaşa yürüten izleklerle oluşturulmuş bir süreç söz konusu olur. Tek izlek içeren ve çok izlek içeren iki ayrı süreci gösteren bir çizim Şekil 3.1'de gösterilmiştir. Şekilde izlekler tarafından paylaşılan kaynaklar gri zemin rengi ile belirtilmiştir.

İzleklerin olmadığı işletim sistemlerinde eşzamanlı görevlerin tamamlanması için süreçler arası haberleşme yapılması gerekir. Süreçler, işletim sistemleri tarafından bağlamları yalıtılmış tasarlanmışlardır. O nedenle ortaklaşa bir görev yürütmek üzere haberleşmeleri işletim sistemine ağır yük getirir [63]. Bunun aksine, izlekler, aynı sürecin içinde yer alırlar; sürecin tüm verisine, sürecin tüm bellek uzayına, hatta birbirlerinin yığınlarına erişebilirler [64]. Kısaca, izleklerin ortaya çıkışındaki amaç, yalıtılmış bağlamları olan süreçler arasında haberleşme yapmak yerine eşzamanlı yürütülebilen kodları bir araya toplayarak aynı bağlamın içinde kalmalarını sağlamaktır [63].

[1] Taşıyıcı (İngilizcesi: "container") içine yerleştirilmiş uygulamaların bulutta kullanımı 2012'den bu yana yaygınlaşmıştır [62]. Ancak, bulut hizmet modeli içinde taşıyıcı kullanımı sanal bilgisayar sunan bulutlar kapsamında değerlendirilmelidir.



Şekil 3.1 : Tek izlekli ve çok izlekli süreçler.

3.2.2 İzlekleri yalıtmanın gereği

Sadece bu yapısal inceleme bile tanım gereği izleklerin yalıtıma uygun olmadıklarını göstermektedir. Zaten, izlekler arasında bir yalıtımın var olamayacağı ve bunun gereksiz olduğu öteden beri söylenmektedir [64]. Ancak, bu uygunsuzluğa karşın, bilgiişlem ortamı sunan bulutlarda işlemci gücünün izlekler yoluyla sağlanmasını akla yakın gösterecek kullanımlar da vardır. Günümüzde sıkça kullanılan bazı programlar işlemci gücünü yapıları gereği ancak izlekler yoluyla sağlar. Bu tasarım elbette değiştirilebilir ancak oldukça yaygın kullanılan programlama standartlarının bir anda değiştirilmesi hem kullanıcıların hem de buna alışmış sağlayıcıların işine gelmeyecektir. Bu durumlarda tanım gereği yalıtılmamak üzere tasarlanmış olmalarına karşın, izleklerin yalıtılmasının sağlanması istenmektedir.

3.3 İzleklerin Yalıtımının Gerçeklenmesi

Bu bölümde, var olan ve tezde tasarlanan düzenekleri kapsayacak biçimde basitten karmaşığa doğru izlek tabanlı bir bilgiişlem ortamı sunan bulutta bir izleğin yalıtımının uygulamada nasıl yapıldığı gösterilmiştir.

3.3.1 Gerçekleme ortamı

İzleklerin yalıtılmasını gerektirecek kullanımlara en güzel örnek ağ hizmetleridir. Bu programlar İnternet yoluyla bir HTTP isteği alır ve bu isteği işledikten sonra buna karşılık bir HTTP yanıtı verir. Ağ hizmeti olarak tasarlanmış bir program, belirlenmiş olan standarda uymak zorunda olduğundan bu döngünün dışında bir yol izleyemez. Her bir istek bir diğerinin zamanlamasından etkilenmeden yürütülür. Bu yapının uygunluğu nedeniyle, ağ hizmeti standardına uygun programlarda her bir HTTP isteği hemen her zaman bir izlekle ilişkilendirilmiştir. Bu izlek HTTP isteği ile verilen görevi yerine getirdikten sonra yanıtı oluşturarak sonlanır.

Anlatılan yapının arkasına kurulan bir bulutun *bilgiişlem ortamı sunan bulut* olduğu varsayılırsa, işlemci gücü de ancak izlekler yardımıyla kullanılabilir. Bu durumda 2.2.7 bölümünde sözü edilen güvenli hesaplama katmanı önlemi akla uygun bir yol olur. Bu konunun nasıl gerçekleştirildiği 3.3.4 ve 3.3.5 bölümlerinde anlatılmıştır. İşlemci gücünden izlekler kullanılarak yararlanılan, kullanıma açılmış bir bulut olan Imona Cloud [65] yaptığımız çalışmada [10] bu konunun uygulama ortamı olarak kullanılmış ve bu bölümde sunulan düzenekler Imona Cloud üzerinde gerçeklenmiştir.

3.3.2 Yalıtımda kullanılan ilkelerin programlama dilleriyle ilişkisi

Tezde izlek yalıtımı için kullanılan ilkeler çoğu zaman bunların gerçeklenmesinde kullanılacak programlama dillerinden bağımsızdır. Ancak bilgiişlem ortamı sunan bulutlarda genellikle altyapıdaki farklılıklardan etkilenmemek için işletim sistemlerinden bağımsız, taşınabilir kod[2] olarak derlenip süreç sanal makinalarında çalıştırılan Java

[2]İngilizcede bu terim "portable code" ya da "p-code" olarak kullanılmaktadır.

programlama dili [66] yeğlenmektedir. Bu yaklaşım, 2.2.8 bölümünde belirtilen *Standart arayüzlerin eksikliği (Z_5)* zayıflığının getireceği tehditlere karşı da uygun bir önlemdir. Bu nedenle izlek yalıtımı anlatılırken Java programlama dili temel alınmıştır. Anlatılan ilkeler, örneğin, UCSD Pascal [67] dilinde ya da .NET [68] tabanlı programlama dillerinde kolaylıkla uygulanabilir. Hatta, bu ilkeler taşınabilir koda dönüştürülmeden doğrudan makina diline derlenen programlama dillerinde de –kimi uyarlamalarla– kullanılabilir.

3.3.3 Yalıtımı gereken kaynak tipleri

Bilgiişlem kaynakları yalıtılırken dört temel kaynağın yalıtılması gerekir. Bunlar, dosya, ağ, bellek ve işlemcidir. Bu dört kaynaktan ilk ikisi için Java dilinde soyutlama yapılmıştır, yani bunlara has yazılım sınıfları bulunur. Bu kaynaklara erişim programlanırken bu sınıfların kullanılması yeterli olur. Yazılım ile soyutlanmış kaynaklara erişimin yalıtılmasında en uygun yol bu yazılım sınıflarına erişimin denetlenmesiyle çözülür. Diğer iki kaynağa erişim için ise belli bir soyutlama yoktur. Erişim doğrudandır. Erişimin yazılımda soyutlanmadığı belleğin yalıtımı her ne kadar tamı tamına yapılamaz [64] ise de, yalıtıma yardımcı yöntemler tasarlanmış [10] ve 3.3.5 bölümünde anlatılmıştır.

3.3.4 Dosya ve ağ erişiminin yalıtılması

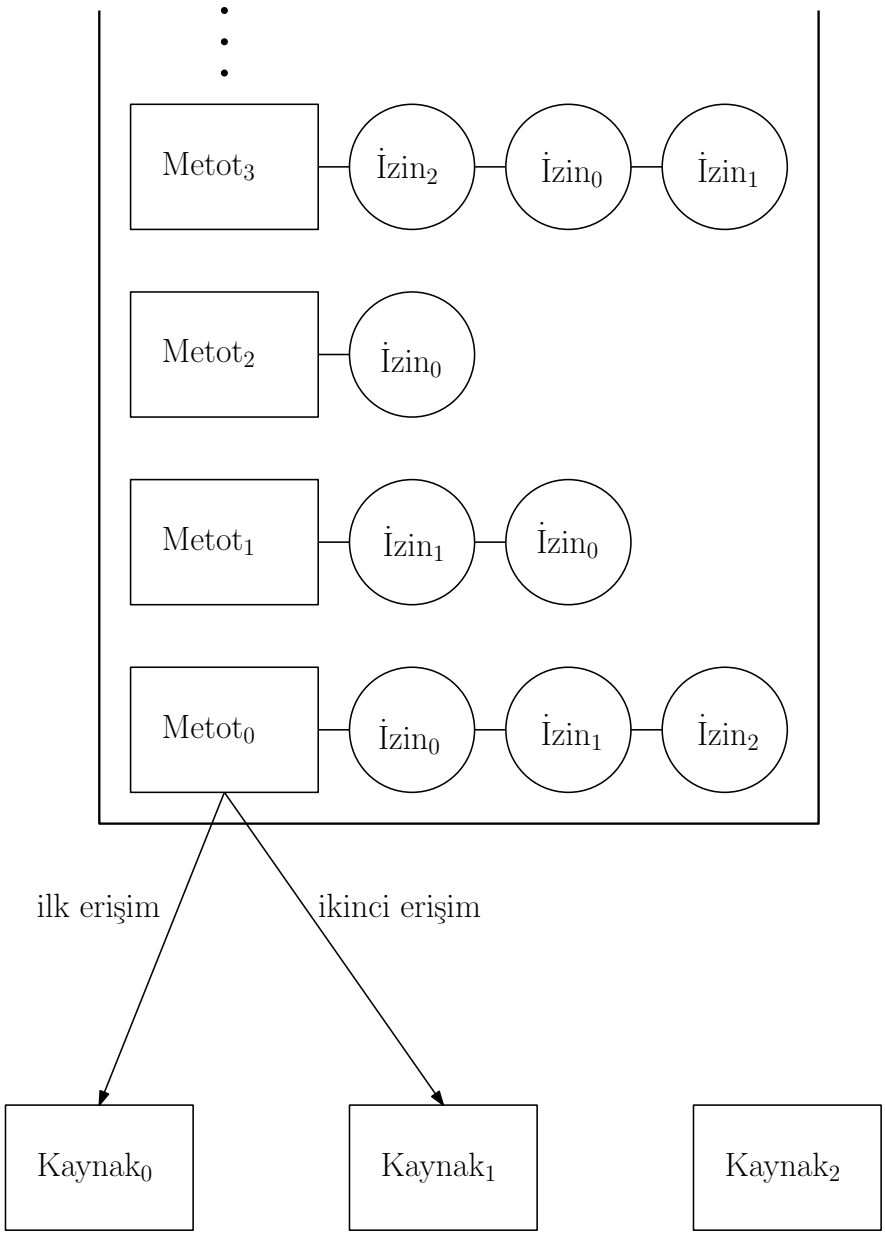
Dosya ve ağ kaynaklarına erişim yalıtılırken çalışma anında değişmeyen koşullar belirlenmiş olursa Java'nın izin düzeneğini [69] kullanmak yeterlidir. Çalışma anında değişen koşullara uyum gösteren izinler için ise buna ek olarak yeni bir düzenek kurulmalıdır. Bu düzenekler aşağıda sırasıyla tanıtılmıştır.

3.3.4.1 Java'nın var olan izin düzeneğinin kullanımı

Dosya ve ağa erişimin yalıtılması için Java'nın var olan izin düzeneğinden [69] yararlanılabilir. Bu düzeneğin etkin kılınması için programın sınıfları çalışma sırasında yüklenirken ilgili koruma uzayı[3] içinde tanımlanır. Bu nedenle dilin sınıf yükleyicilerinin[4]

[3] Java programlama dilinde "ProtectionDomain" sınıfı.
[4] Java programlama dilinde "ClassLoader" sınıfı.

yeniden tanımlanması ve koruma uzayının belirlenmesi gerekmektedir. Bundan sonra programın güvenlik yöneticisi[5] izinleri denetleyerek erişimi yönetir.



Şekil 3.2 : Kaynaklara erişimin yalıtılması.

Erişim yönetiminin nasıl gerçekleştiği Şekil 3.2'de gösterilmiştir. Herhangi bir anda izin gerektiren bir kaynağa erişmek isteyen bir Java metodunun bulunması durumunda, o metodun gerekli izni bulundurup bulundurmadığı denetlenir. İzin olmaması durumunda, kuraldışı bir durum[6] oluşacaktır. İzin olması durumunda, çağrı yığınında ilgili metodu çağıran bir başka metodun olup olmadığına bakılır; varsa izin denetimi o metot

[5]Java programlama dilinde "SecurityManager" sınıfı.
[6]Java programlama dilinde "Exception" sınıfı.

üzerinde de işletilir. Böylece çağrı yığını boyunca tüm metotların gerekli izni bulundurduğu durumlar dışında bir kaynağa erişimi engellenmiş olur.

Şekildeki örnekte Metot_3 Metot_2'yi, Metot_2 Metot_1'i, Metot_1 Metot_0'ı çağırmıştır. Metot_0 sırasıyla Kaynak_0'a ve Kaynak_1'e erişmeyi denemiştir. Kaynak_0'a erişim denendiğinde Metot_0'ın erişmek için gerekli izin olan İzin_0'ı bulundurduğu görülür. Bu durumda Metot_0'ın kaynağa erişmeyi deneyen son metot olup olmadığı önem kazanır. Örneğimizde, Metot_0'ı çağıran bir metotlar silsilesi vardır. Bu durumda, bu metotların Metot_3'e dek her biri, İzin_0'ı taşıyıp taşımadığına bakılarak denetlenir. Çağrı yığınının sonuna ulaşıldığında kuraldışı bir durum oluşmamışsa erişime izin verilir. Aynı biçimde Kaynak_1'e erişilmesi denendiğinde Metot_2 gerekli izni taşımadığı için kuraldışı bir durum oluşacak ve erişime izin verilmeyecektir.

3.3.4.2 Çalışma sırasında değişebilen izinlerin gerçeklenmesi

Söz konusu izinlerin metotlarla ilişkilendirilmesi ya süreç sanal makinasının çalışması anında bir kerelik ve sonradan değiştirilemez biçimde yapılır ya da metotları taşıyan sınıfların yüklenmesi sırasında gerçekleştirilir. 3.3.4.1 bölümünde anlatılan ilk yöntem izleklerin iyeliğinin sürekli değişebildiği bulut ortamına uygun olmadığı için bu bölümde açıklanan ikinci yöntem kullanılmalıdır. İzinler sınıfların yüklenmesi sırasında belirleneceğine göre, erişim güvenliğinin kritik olduğu anlardan biri sınıfların yüklendiği andır. Ancak Java programlama dilinde sınıf yükleyicilerinin nasıl davranacağına, hangi sınıfların izinler kapsamında yüklenebileceğine ve izinlerin yüklenen sınıflarla nasıl ilişkilendirileceğine yönelik hazır bir denetim düzeneği yoktur.

Java'nın izin düzeneği devreye alınmamışken, bu düzeneğe ek olarak bunun bir benzeri kurularak hangi sınıfların yüklenebileceği, yüklenen sınıfların hangi metotlarının hangi kaynakların kullanımına izin vereceği belirlenir. Bu izinler de, yürütme sırasında yine çağrı yığını üzerindeki metotlar taranarak denetlenir ve gerektiğinde engellemeler yapılır. Böyle bir izin düzeneği gerçeklenmiştir ve uygulamada kullanılmaktadır [10]. Burada anlatılan izin düzeneğinde, sınıf yükleyici her bir izleğin çalışması sırasında kullanılacak sınıfları bağımsız olarak yeniden yükleyecek biçimde tasarlanarak sözü geçen ağ ve dosya kaynaklarına erişimi yalıtmayı başarır.

Gerçeklenen izin düzeneğinin bu yolla oluşturulmasının nedenleri izleklere ve işlemci gücünü izlekler aracılığıyla kullandıran bilgiişlem ortamı sunan bulutlara özgü şu özelliklerdir: Sağlayıcıya ait konakta sağlayıcıya ait bir süreç olarak çalışan bir program bulunur. Müşterilerin ya da müşterilerin kullanıcılarının işlemleri bu sürecin içinde kısa süreli izlekler olarak bir süre yürütülmekte ve sonlanmaktadır. Genellikle, görev yürütülmediği anlarda izlekler bir havuzda boşta bekletilir. Herhangi bir istek yapıldığındaysa boştaki rasgele bir izlek seçilerek görev tamamlanana kadar çalıştırılır. İzlek tarafından bakılacak olursa, verilen görev sağlayıcının, müşterinin ya da bir kullanıcının olabilir; hatta aynı izlek farklı müşteriler ya da kullanıcılar adına görevler yürütebilir. İzleğin farklı müşteriler ya da kullanıcılar adına görevler yürütmesi sırasında kendine daha önce verilmiş bir görevde kullandığı yazılım nesnelerini tekrar kullanması durumunda, bu nesneler ile önceki görevlere ilişkin bilgilerin taşınması söz konusu olur. Bu durumda izlek yalıtımından söz edilemez. Ancak, izleklerin yürüttükleri her bir bağımsız görev boyunca ayrı nesneler kullanarak yalıtılabilmeleri durumunda bunun güvenliğe yararı olur. Bu tür bir yalıtımın da sürekli nesne silmek ve yeniden oluşturmayı gerektireceği için başarım açısından götürüsünün çok olacağı öngörülür. Sadece izleklerin yalıtılması yetmeyeceği gibi, izlekler bir de zaman ekseninde yalıtılmak zorundadır. Bu yaklaşımın yerine, kullanılan kaynaklara, kullanılan yazılım sınıflarına ve çağrı yapan metotlara bakılarak bir karar düzeneği oluşturulur. Ağ hizmeti standardına uyarak izlekler aracılığıyla işlemci gücünü kullandıran bulut bir istek aldığında istek incelenerek hangi kullanıcının istekte bulunduğu anlaşılabilir. Kullanıcının belirlenmesinden sonra, bulutun kullanıcıları yönetmekle sorumlu bölümü her kullanıcının ayrı bir metottan çağrı yapmasını sağlayacak biçimde programlandığında, kurulan izin düzeneğinin kullanıcılar temelinde izinleri yönetebilmesinin önü açılmış olur. Bundan sonra gerekli düzenleme sağlayıcı ve müşteri arasındaki anlaşmaya göre kaynakların, yazılım sınıflarının hangi haklarla ve nasıl kullanılacağının belirlenmesinden oluşur.

Şekil 3.3'te izinlerin nasıl kullanılacağı gösterilmiştir. Gösterilen, yazılımda kullanım kolaylığı getirmesi için bir arada tasarlanmış olan bileşik bir izindir. İzinde belirtilen birinci parametre bir paketin ya da sınıfın adıdır ve bu paket ya da sınıfın yüklenmesine izin verildiğini belirtir. İkinci parametre ise yüklenen bu paket ya da sınıfa hangi metodun çağrı yapabileceğini belirtir. Şekilde birinci satırda görülen izin ile

```
JointPermission("java.lang.*", "*");
JointPermission("java.io.File", "java.*", "Foo.bar");
JointPermission("java.net.Socket", "tr.edu.itu.IletisimYoneticisi.sadeceYerel");
```

Şekil 3.3 : Çalışma sırasında değişebilen izinlerin Java dilinde kullanımı.

Java dilinin temel sınıflarını barındıran "java.lang" paketi içindeki tüm sınıfların yüklenebileceği ve hiçbir ayrıklık olmaksızın her metoda bu sınıflara çağrı yapma izni verileceği gösterilmiştir. İkinci satırdaki izin ile dosya kaynaklarını soyutlayan yazılım sınıfı olan "java.io.File" sınıfına "java" paketi içindeki bütün sınıfların bütün metotlarının ve buna ek olarak "Foo" sınıfının "bar" metodunun erişebileceği belirtilmiştir. "Foo" sınıfının "bar" metodu yukarıda anlatıldığı gibi bir kullanıcının erişimi ile özdeşleştirilirse; o kullanıcıya dosya erişim hakkı tanındığı düşünülür. Son satırda gösterilen örnekte "java.net.Socket" sınıfının yüklenmesine izin verildikten sonra sadece "tr.edu.itu" paketi içindeki "IletisimYoneticisi" sınıfının "sadeceYerel" metodu tarafından erişilebileceği belirtilmiştir.

3.3.5 İşlemci gücünün sınırlanması

Belleğin ve işlemcinin yalıtılamaması güvenlik açısından dikkate değer bir zayıflıktır. Ancak daha önceden de belirtildiği gibi [64], yalıtım tam anlamıyla başarılamaz. Belleğin yalıtımı üzerine uygulamada geçerli kullanışlı bir yol yoktur. İşlemci kullanımının sınırlanması üzerine tarafımızdan yapılan çalışma [10] ise aşağıda açıklanmıştır.

İzleklerin işlemci kullanım süreleri Java süreç sanal makinası [70] tarafından yaklaşık olarak belirlenebilir. Bu değerler bir denetçi izlek tarafından izlenerek bir istatistik oluşturulursa, buna bakılarak olması gerekenin dışında davranan izlekler sezilebilir. Ancak, bu yaklaşım aşağıdaki varsayımları içermektedir:

1. Kötücül davranan izlek işlemci kullanım süresindeki sıradışılıklarla kendi belli etmektedir.
2. İzleklerin işlemci kullanım sürelerinin bağıl istatistiği yeterli ve anlamlı bilgi vermektedir.
3. Kötücül olmayan izlekler istatistiksel olarak işlemciyi benzer biçimde kullanmaktadır.

Öte yandan bu varsayımlar altında kötücül olan izleklerin denetçi izlek tarafından nasıl sonlandırılacağı da ayrı bir sorun oluşturur. Bu sorun getirilen çözüm bölümün sonunda açıklanmıştır.

Java'nın sunduğu JMX paketi içinde yer alan sınıfların metotlarına yürütme anında yapılan çağrılarla izleklerin işlemci kullanım bilgileri toplanır. Sanal makinanın sağladığı bilgi o anda tüm işlemciler üzerinde çalışan tüm izleklerin bilgisidir. Bu bilginin elekten geçirilip sadece ilgili izleklere ilişkin verilerin istatistiklere alınması gerekir. Ayrıca, izleklerin işlemci kullanım süreleri verilirken, ne zaman işe başladıkları, ne kadar zamandır canlı oldukları, çalışmaya başlamadan önce bir izlek havuzunda bekletilip bekletilmedikleri, o anda üzerinde çalıştıkları görevden önce bir başka görevleri olup olmadığı, hangi işlemcide ne kadar zaman harcadıkları gibi soruların yanıtları doğrudan alınamaz. Bu bilgilerin elde edilmesi için ayrıca kod yazarak hesaplama yapmak gerekir. Sayılan fazladan veri toplama ve hesaplama işleri yapılacak olursa asıl hesaplama işlerinin aldığı süreye kıyasla önemli bir süre almaya ve ölçüm sonuçlarını bozmaya başlar. Bu sakıncalar göz önüne alınarak istatistiğin doğruluğundan olabildiğince az ödün verilerek sanal makinanın en az biçimde yükleneceği veri toplama ve istatistik oluşturma yöntemleri uygulanmıştır. Bunu sağlamak için toplanan verilerin tamamı uzun süre bellekte saklanmamış, sadece o ana kadar toplanan örnek sayısı ve bunların ortalaması bellekte tutulmuş, eskiyen ölçümler kısa sürede bellekten silinmiştir. Her izlek sadece kendine ilişkin ortalama işlemci kullanım süresini saklamış, bunun dışında işlemci üzerindeki tüm izlekler için ortak bir değer daha saklanmıştır. Gerekli yoğunluk değerleri bunlar oranlanarak işlemciye fazla yük bindirmeden hesaplanmıştır.

Elde edilen istatistiklere dayanılarak sıradışılığın belirlenmesi için olağan durumun öğrenilmesi gerekir. Bu da uygulamada zorluklar getirecek bir öğrenme süreci gerektirir. Böyle bir süreçten geçmek yerine çalışmada yalın bir yol seçilmiş ve izleklerin çalışma süre ve yoğunluk değerleri göz önüne alınıp belirlenen düzeylerin aşılması sıradışılık ölçütü olarak kabul edilmiştir. Bir izleğin çalıştığı süre belirlenen bir limiti aşarsa, işlemciyi belirlenenden daha yoğun kullanırsa veya belirlenen süre boyunca belirlenen yoğunluğun üzerinde kullanırsa o izleğin sonlandırılması için girişimde bulunulur.

Sıradışılık belirlendikten sonra kötücül olduğu düşünülen izleğin sonlandırılması gerekir. Bunu yapmak üzere denetçi izlek kötücül olduğu düşünülen izleğe durması yönünde bir uyarı gönderir. Ancak izleklerin durdurulmalarını dayatan düzenekler yoktur. Bu nedenle, kötücül izlek bu uyarıya uyup uymamayı kendisi seçecektir. Bu belirsiz durumu görece onarmak için denetçi izlek uygulamada şu yolu dayatır [10]: kötücül olduğu düşünülen izleğin kullanabileceği kaynaklara da aynı biçimde uyarı gönderilir. Bu durumda kötücül izlek kullanmak üzere uyarılmış süreç sanal makinası kaynaklarından birine yöneldiğinde izinleri yetersiz olarak algılanacak ve bir kuraldışı durum oluşturularak çalışması sonlandırılacaktır. Bu yöntem izlenirse, kötücül izleğin denetçi izleğin uyarısına karşın çalışmayı sürdürmesinin tek yolu süreç sanal makinası yoluyla sunulan hiçbir kaynağa erişmemesi olacaktır; ki bu da kötücül izleği çalışır durumda kalsa bile sağlayıcının hiçbir hesaplama kaynağına yönelemez edilgen bir durumda bırakır, yani görece zararsız kılar.

3.4 Benzetim

Bilgiişlem ortamı sunan bulutlarda önerilen izlek yalıtımı yöntemlerinin uygulamaya konmadan önce başarıma etkilerinin görülmesi için kullanımdaki buluta [65] eşdeğer bir ortam oluşturularak benzetimler yapılmıştır. Bu benzetimde elde edilen sonuçlar aşağıda sunulmuştur.

3.4.1 Benzetim ortamı

Benzetim, trafiği oluşturmakla görevli olan kullanıcı tarafında yaşanacak donanım zayıflıklarının ölçümlere etkisinin en aza indirilmesi için, kullanıcı tarafı güçlü, sağlayıcı tarafı daha zayıf donanımla kurgulanmıştır. Sağlayıcı, 4 GB bellekli, Intel® Core i5-4200U 1.6 GHz işlemcili bir bilgisayar üzerinde 64 bitlik Microsoft® Windows® 8.1 işletim sistemi kullanmaktadır. Güvenliği sağlayacak düzenekler Java 1.7.60 sürümü bir süreç sanal makinası ve bununla birlikte çalışan Apache Tomcat 7.0.42.A sürümü uygulama sürücüsü üzerinde çalışmaktadır. Kullanıcıları modelleyen diğer bilgisayar Apache JMeter 2.11 trafik üretecini 8 GB bellekli, Intel® Core i7-4700EC 2 GHz işlemcili bir donanım üzerinde 64 bit Linux® Mint® 16 işletim sistemi ile

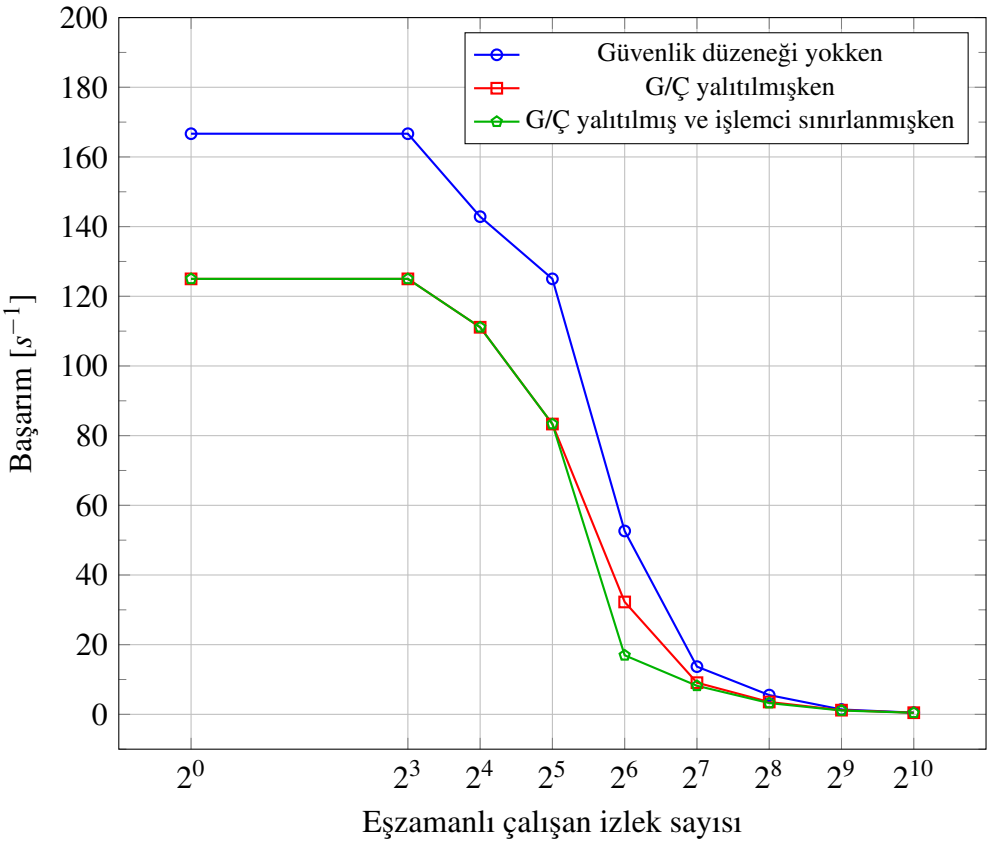
yürütmektedir. Bu iki bilgisayar 802.11g standardında telsiz bir ağ üzerinden alışıldık TCP/IP protokol yığını ile haberleşmektedir.

3.4.2 Ölçeklenebilirlik

Benzetime başlamadan önce olası bellek taşmalarını belirlemek ya da ağa bağlı gecikmeleri ölçmek için öncül sınamalar yapılmıştır. Bellek taşmalarını gözlemlemek için tek bir izleğin yaklaşık beş dakika boyunca sürekli rasgele kaynak erişimi isteğinde bulunması sağlanmıştır. Bu süre boyunca hem sağlayıcıyı hem de kullanıcıyı modelleyen bilgisayarlardaki değerler başından sonuna dek kararlı bir aralıkta ölçülmüştür. Bu durum, kayda değer bir bellek taşması olmadığı biçiminde yorumlanmıştır. Ağda ölçüm sonucunu etkileyecek değerde bir gecikme olup olmadığının anlaşılması için eşzamanlı erişim isteği yapan izleklerin sayısı ve izleklerin yaptıkları erişim isteği türleri rasgeleyken her bir isteğin yanıtlanması için geçen süre hem sağlayıcı tarafında hem de kullanıcı tarafında ölçülmüştür. Daha sonra her iki taraftaki ölçümler birleştirilerek farklar kıyaslanmıştır. Dağılım, açıkça 0 ms farka yaslanmış olduğu için ağ kaynaklı gecikmelerin önemsenmeyecek kadar küçük olduğu, ölçülen gecikmelerin büyük oranda sağlayıcı tarafında hizmet sunmak üzere yürütülen programlarda harcanan zamandan kaynaklandığı yorumu yapılmıştır.

Benzetimin ilk aşamasında daha sonraki aşamalarda önem taşıyacak olan eşzamanlı çalışacak izlek sayısı belirlenmiştir. Bunun için eşzamanlı izlek sayısı her adımda iki katına çıkarılarak sağlayıcıyı modelleyen bilgisayarın tepkisi ölçülür. Bilgisayarın hem uygun sürede tepki verdiği hem de sığasının tamamını kullandığı eşzamanlı çalışan izleklerin sayısı olarak belirlenen 50 değeri bu aşamadan sonra izleyen benzetimlerde kullanılmak üzere seçilmiştir.

Benzetim yapılırken her adımda üç durum ele alınmıştır. Birincisi, kontrol için kullanılır: herhangi bir güvenlik düzeneğinin çalışmadığı ve kaynaklara erişime izin verilen durum. İkincisi, sadece dosya ve ağ erişiminin belirlenen izinlere bakılarak kısıtlandığı, işlemci üzerinde denetim yapılmayan durum. Üçüncüsü, dosya ve ağ erişimi kısıtlanmakla beraber işlemci kullanım istatistiklerinin işlendiği durum. Dikkat edilirse üçüncü durumda sadece istatistiklerin işlendiği belirtilmiş ancak izleklerin çalışmasının sınırlanacağından söz edilmemiştir. Bunun nedeni, benzetimde başarımın

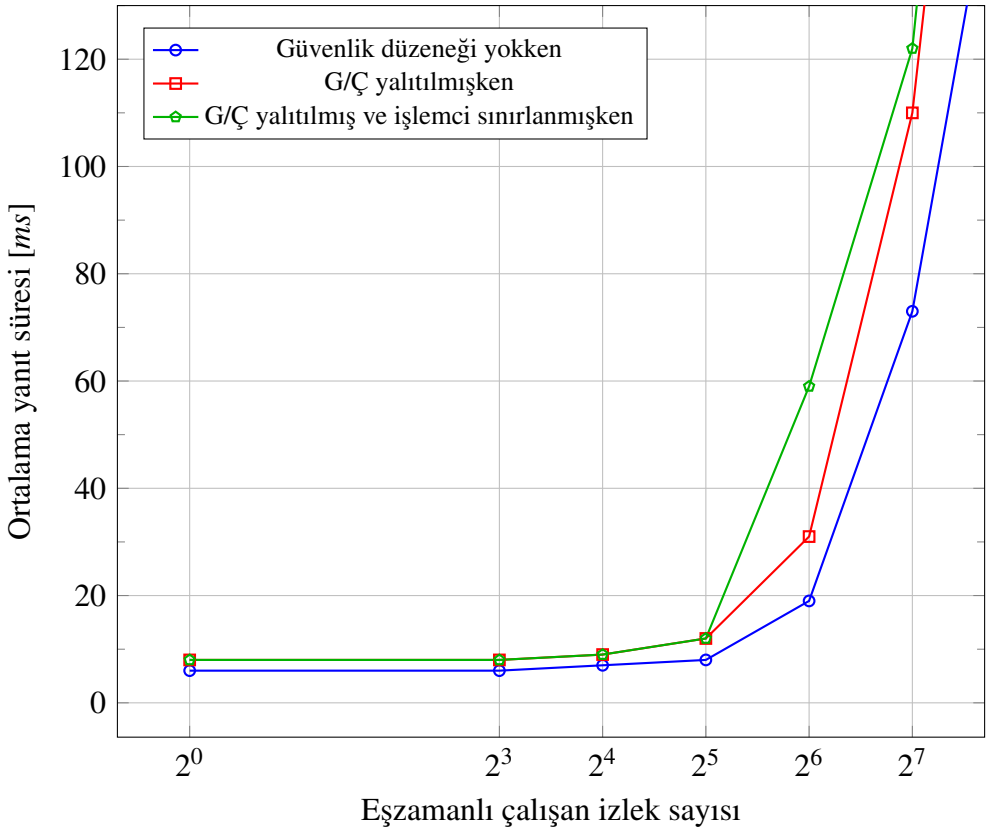


Şekil 3.4 : Başarımın eşzamanlı çalışan izlek sayısına oranı.

ölçülüyor olmasıdır. İzleklerin durdurulması işlemcinin üzerindeki toplam yükü azaltacağından başarıma olumlu etki eder. Benzetimde, kurulan izin düzeneklerinin olası en kötü durumları aranmaktadır. Bu nedenle istatistikler çalıştırılmasına karşın ölçümlerde izlekler sonlandırılmamıştır. Şekil 3.4, 3.5 ve 3.6'da sunulan grafiklerde işaretlenmiş her bir nokta en az yüz bin tekil ölçümün aritmetik ortalamasıdır.

Ölçeklemenin başarıma göre ölçüldüğü Şekil 3.4'te üç eğrinin de aynı noktalarda bel verdiği görülür. Bu, başarımdaki değişikliklerin kurulan güvenlik düzeneğinden çok eşzamanlı çalışan izlek sayısına bağlı olduğunu gösterir. Bu durumda eşzamanlı izlek sayısının üst sınırı belirlenirse, 3.3.4.2 ve 3.3.5 bölümlerinde sunulan düzenekler ölçeklenebilir. Ölçeklemenin ortalama yanıt süresine göre ölçüldüğü Şekil 3.5 ve Şekil 3.6'da bu olay daha belirgindir. Eşzamanlı çalışan izlek sayısı arttıkça güvenlik düzeneklerinin getirdiği gecikmenin sabit kaldığı Şekil 3.6'da açıkça görülmektedir.

Ölçekleme benzetimiyle ilgili ilgi çekici bir durum Şekil 3.6'da görülmektedir. Burada izleklerin etkinliğini sınırlamaya yönelik düzeneğin çalışmasına izin verilerek etkisi gözlenmiştir. Beklendiği gibi, eşzamanlı çalışan izleklerin sayısı azken güvenlik

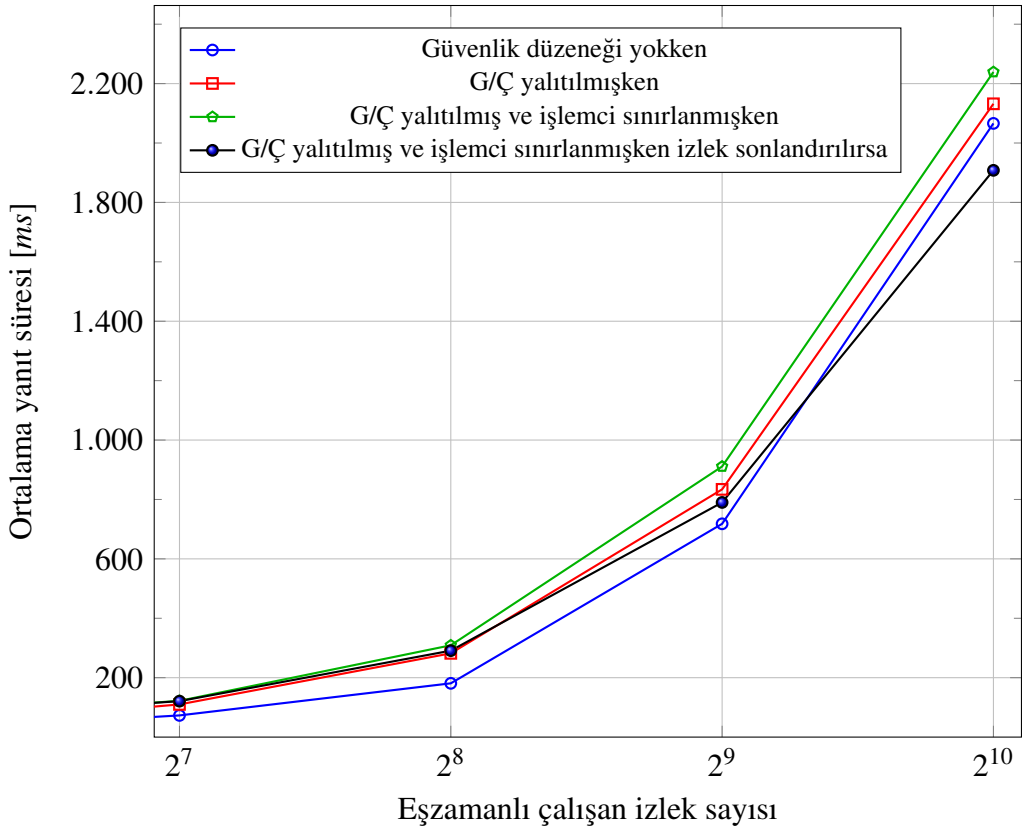


Şekil 3.5 : Ortalama yanıt süresinin eşzamanlı çalışan izlek sayısına oranı.

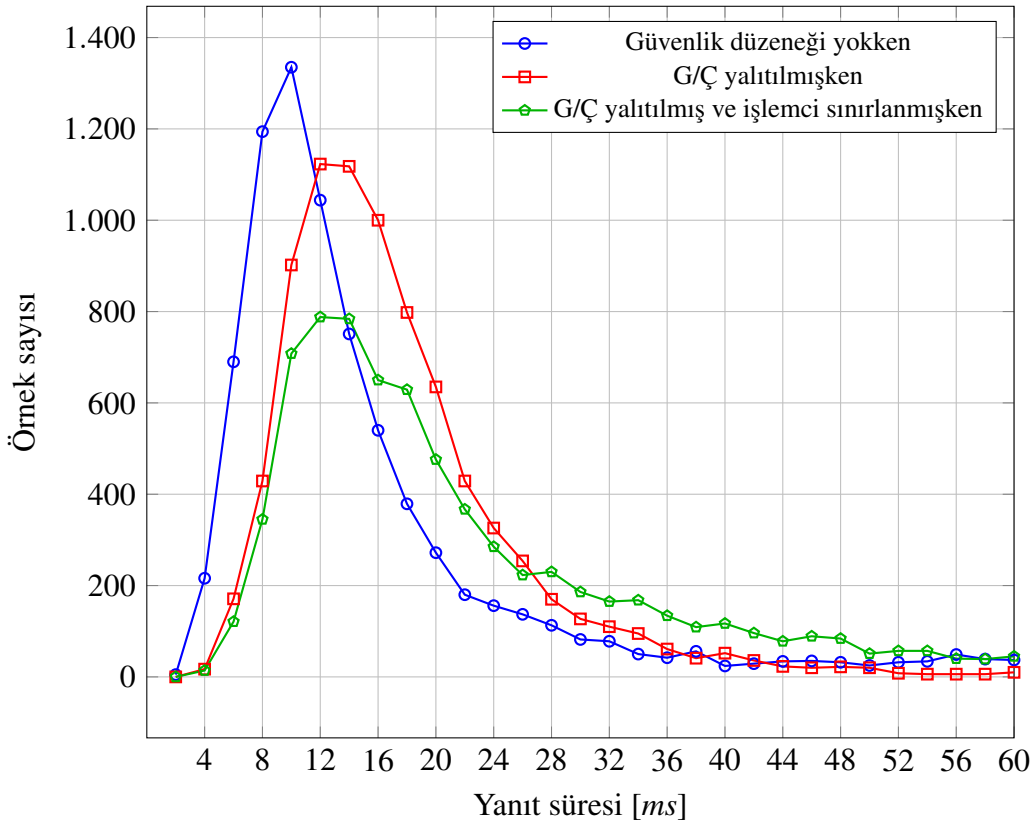
düzeneği bir yük oluşturmaktadır. Ancak, eşzamanlı çalışan izleklerin sayısı arttıkça izin verilen toplam süreyi aşan izlekler zararlı izlekler olarak algılanarak sonlandırılmakta ve toplam yük azalmaktadır.

3.4.3 Gecikme

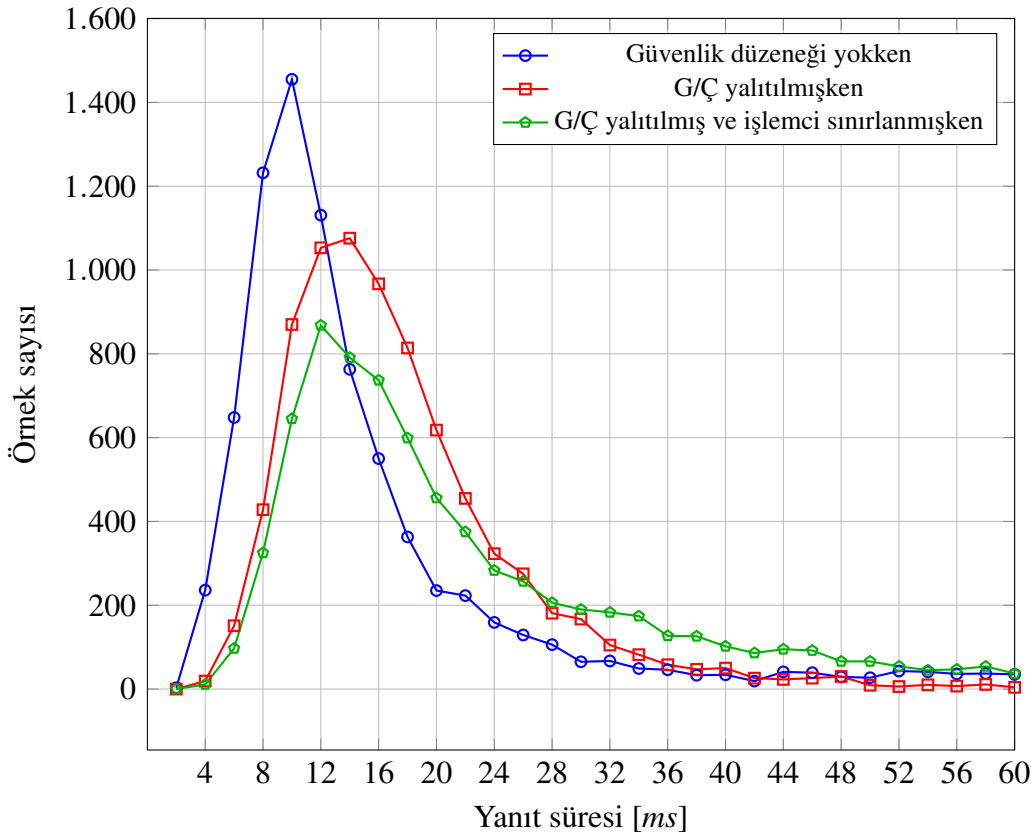
Benzetimin ikinci aşamasında kullanıcıların sağlayıcı tarafında farklı tipten kaynaklara erişim isteğinde bulundukları varsayılır. Sağlayıcı bu isteklere uygun yanıtlar verir. Örnek olarak, bir güvenlik düzeneği yoksa kullanıcı dosyalara ve ağa erişebilir; bir güvenlik düzeneği varsa erişim belirlenen izinlere göre yönetilir. Yirmi binin üzerinde rasgele istek elliden fazla eşzamanlı izlek tarafından gönderilerek benzetim gerçekleştirilmiştir. Elde edilen ölçümlere göre disk erişimi sırasında ortalama yanıt süresinin dağılımı Şekil 3.7'de, ağ erişimi sırasında ortalama yanıt süresinin dağılımı Şekil 3.8'de ve rasgele erişim sırasında ortalama yanıt süresinin dağılımı Şekil 3.9'da sunulmuştur.



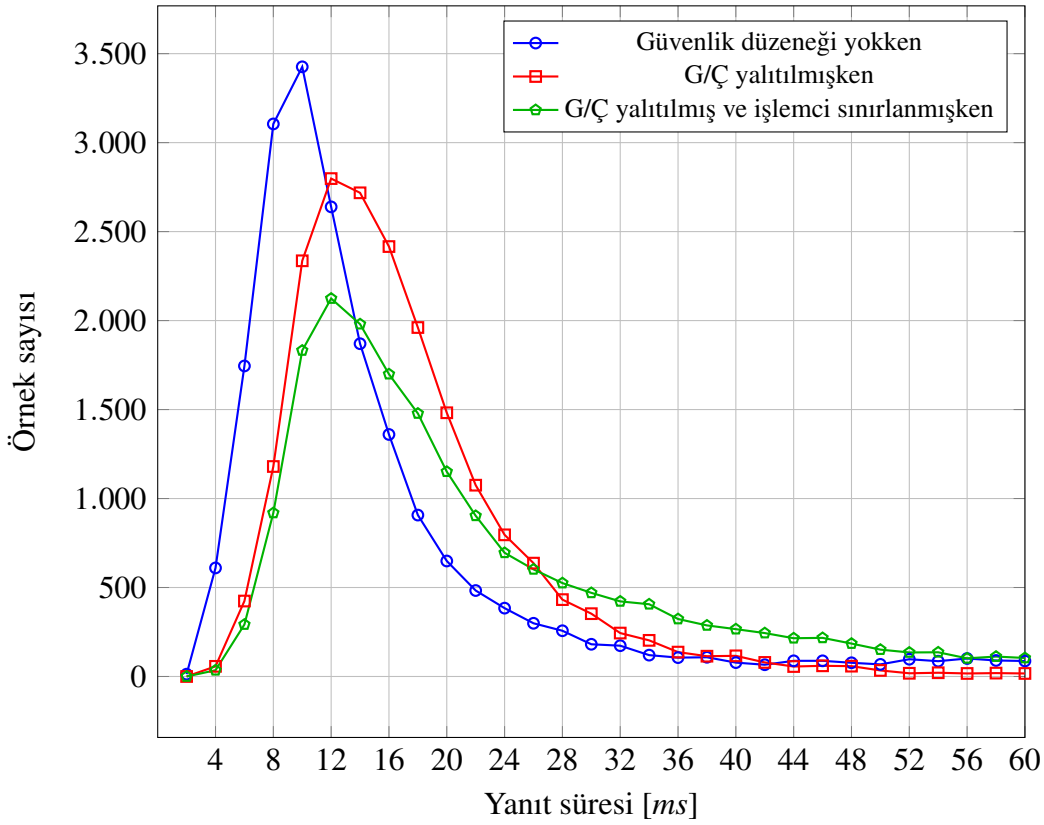
Şekil 3.6 : Ortalama yanıt süresinin eşzamanlı çalışan izlek sayısına oranı.



Şekil 3.7 : Diske erişim isteği yapılırken oluşan gecikmelerin dağılımı.



Şekil 3.8 : Ağa erişim isteği yapılırken oluşan gecikmelerin dağılımı.



Şekil 3.9 : Diske ve ağa rasgele erişim istekleri yapılırken oluşan gecikmelerin dağılımı.

Sunulan grafikler ordinattaki değerler önemsenmeyecek olursa birbirini andırır. Bu da, düzeneklerin gecikmeye etkisinin ardında yatan etkinlikten bağımsız olduğunu gösterir. Düzeneğin olmadığı durumlarda dağılım sıfıra yaklaşmış ve sivrilmişken, düzenek karmaşıklaştıkça dağılımın tepe noktasının törpülenerek sıfırdan uzaklaştığı görülmektedir. Her durumda tepe noktalarının yüksekliği arasındaki bağıl fark %20 kadardır ve tepe noktaları arasındaki zaman gecikmesi 2 ms kadar ölçülür. Bu süre, istek/yanıt yaklaşımıyla çalışan ve bir ağ hizmetinin arkasına kurulu çalışan bir bulut için önemsiz sayılabilecek bir süredir. Bu sürenin hemen hemen sabit kalacağını gösteren önceki ölçümlerle birlikte, 3.3.4.2 ve 3.3.5 bölümlerinde sunulan düzeneklerin uygulamada kullanılabilir olduğu sonucuna varılır.

3.5 Çalışma Sırasında Değişebilen İzin Düzeneğinin ve İşlemci Gücünü Sınırlandırma Düzeneğinin Getireceği Fazladan Yük

Çalışma sırasında değişebilen izinlerin kullanılabilmesi için kurulan düzeneğin yükü ve işlemci gücünün sınırlanması için kurulan düzeneğin yükü sunuldukları sırayla, art arda incelenmiştir.

Çalışma sırasında değişebilen izinlerin kullanılması için gereken yük Java'nın var olan izin düzeneğinden yer ve zaman açısından farklı değildir. Yer açısından getirilen yük ancak izinlerin saklanması için gereken alandır. İzinlerin yorumlanması için en kötü durumda gereken zaman ise, çağrı yığınındaki metotların sayısı ile, her bir metoda ilişkin izinlerin sayısı ile ve yorumlanacak bu izinlerin kapsamı ile ilgilidir. Teorik olarak, çağrı yığını Java süreç sanal makinasının sınırladığı en yüksek metot sayısı kadar metot içeriyor olabilir. Her bir metoda bellek elverdiğince izin iliştirilmiş olabilir. İzinlerin sınıflara ve metotlara bakılarak yorumlanması da metin tabanlı bir örüntü eşleme işidir.

İşlemci gücünün sınırlanması için fazladan harcanan yer; izlek başına ortalama işlemci kullanımı hesabının başlayacağı zamanı tutan bir değişken ve bunun yanında Java süreç sanal makinasının tamamı için ortalama işlemci kullanımını adım adım hesaplamakta kullanılan iki değişkendir. Fazladan harcanan zaman ise, Java süreç sanal makinasının içerdiği tüm izlekler içinden ağ hizmetleriyle ilgili izlekleri ayırdıktan sonra, bu izleklerin sayısı kadar sorgu ile işlemci zamanlarını öğrenip bir ortalama almak için

geçen zamandır. Burada göz önüne alınması gereken konu, bu hesabın sık yapılması işlemci zamanının kendisini etkileyeceği için bir kilit düzeneği ile belli bir süreden sık yapılmasının engellenmiş olduğu ve o süre boyunca yapılan ilk ölçümün doğru sayıldığıdır. Böylece zaman olarak getirilen yük düzeneğin çalışmasını etkilemeyecek düzeyde tutulur.

3.6 Çözülemeyen sorunlar

Sözü edilen dört kaynaktan ağ ve dosya erişimi engellenebilmektedir. İşlemci kullanımı ise belli koşullar altında sınırlanabilmektedir. Ancak bellek kullanımını yalıtmanın ya da sınırlandırmanın uygulamada geçerli kullanışlı bir yolu bulunamamıştır.

Bellek alanını alt alanlara bölerek veya güvenilir ya da güvenilmez alanlar olarak ayırarak buna göre izleklerin bu alanlara erişimine izin vermek üzere yapılan çalışmalar vardır [71, 72]. Ancak bu çalışmalar süreç sanal makinası üzerinde değişikliklerin yapılmasını gerektirmektedir. Süreç sanal makinası değişmediği sürece tüm bellek alanı aynı uygulamanın tüm izleklerinin ortak erişimine açık tek bir bölümdür.

Bilgiişlem ortamı sunan bulutlarda 3.3.2 bölümünde söylendiği gibi genellikle ortak bir altyapının kullanılmasına çalışılır. Bu 2.2.8 bölümünde sözü edilen *Standart arayüzlerin eksikliği (Z_5)* zayıflığına karşı bir önlemdir. Bu önlemin alınmasının temel nedeni farklı kullanıcıların ve onların kullanabileceği programlama ortamları arasındaki uyumluluğu artırmak, böylece daha geniş bir kitleye hizmet sunabilmektir. Ayrıca bu yolla bulut altyapısının bakımı ve yönetimi büyük oranda kolaylaşacaktır. Bir bulut sağlayıcısının standart programlama ortamlarının, kitaplıklarının ve arayüzlerinin dışına çıkmasını pek ummamak gerekir. Bu yorumlar ışığında, sanal makina değişmediği sürece izlek tabanlı bilgiişlem ortamı sunan bulutlarda belleğin yalıtımından söz etmek olanaklı değildir.

İkinci bir sorun, izleklerin sonlandırılmasıdır. Herhangi bir izlek kendi kendini sonlandırmazsa dışarıdan bir etki ile sonlandırılması uygun değildir. Bunun nedeni, izleğin sonlandırıldığı anda elinde tuttuğu kaynaklara ne olacağıdır. Bir izleğin elinde tuttuğu bellek, dolayısıyla o bellek yoluyla adreslenmiş ilgili tüm kaynaklar, diğer izleklere açık biçimde bulunur. İzleğin sonlanmasıyla bu bellek alanı ve ilgili tüm

kaynaklar açıkta kalır ve saldırıya açık hedeflere dönüşür. Olması gereken, izleğin sonlanırken elinde tuttuğu bütün kaynakları süreç sanal makinasına geri vermesi, tüm belleğin silinmesi ve izleğin o andan sonra sonlanmasıdır. Bunu da bir izleğin ancak kendisi yapabilir.

Bir başka sorun statik kaynakları gösteren bellek adreslerinin paylaşılmasıdır. Örneğin, aynı statik kaynağı iki ayrı izlek kullanırken izleklerden birinin kaynağı bekletmesi diğerinin de beklemesine neden olur. Bu beklemeler akıllıca tasarımlarla zamanı kullanan haberleşme kanallarına dönüştürülerek bilgi sızdırılmasına neden olabilmektedir.

3.7 İzlek Yalıtımı Üzerinde Yapılmış Diğer Çalışmalar

Bölüm boyunca sözü edilen çalışmaların dışında izleklerin yalıtılmasına yönelik başka girişimlerde de bulunulmuştur. Bu çalışmalar çoğunlukla endüstride karşılaşılan sorunların çözümüne yöneliktir. Konunun bütünlüğü açısından bunların da tanıtılmasında yarar görülmektedir.

3.7.1 Enterprise Java Bean

Enterprise Java Bean [73] ya da kısaca EJB adıyla bilinen Java kozaları sunucu tarafında yabancı kaynaklardan gelen kodların çalıştırılmasında kullanılması için Java dili içinde geliştirilen standart bir yöntemdir. Bu kozalar içinde statik değişkenlere ve sınıf yükleyicilere erişim, izlek ve dosya işlemleri yasaklanmıştır. EJB kozaları bu yasaklarla oldukça kısıtlı bir programlama olanağı sunar. Bu yolla işlemci gücünün kullanılabilmesi için kullanıcının uygulamasını EJB standardına uygun yazması ve derlemesi gerekmektedir.

3.7.2 Servlet

Servlet [74] Java dilinde HTTP isteklerini izleklere bağlamak için geliştirilmiş bir standarttır. Bu standartta bir HTTP isteği ve yanıtı ile bir izlek ilişkilendirilmekle birlikte bu izleğin yalıtımı üzerine pek bilgi verilmemiştir. Geliştiricileri özgür bırakan bu yaklaşım güvenlik açısından dikkatle ele alınmalıdır. Google App Engine [75], AWS Elastic Beanstalk [76], Heroku [77], Apache Stratos [78] ve bu bölümün konusunu

oluşturan çalışmamızda [10] kullanılan bulut [65] bu standardı kullanır. Güvenlik ve yalıtım yaklaşımları Servlet standardı üzerine kurulur.

3.7.3 Multitasking Virtual Machine

Multitasking Virtual Machine [79] ya da kısaca MVM adıyla duyurulan özelleştirilmiş süreç sanal makinası izlekler arasındaki yalıtım sorununu çözmek üzere önerilmişse de sorunların hepsini çözmek olanaklı olmamıştır. Sunulan çözümler de daha önce anlatılan izlek tanımı ile çelişmekte ve uygulamada verimsiz olmaktadır. Bir anlamda, önerilen her çözüm izleği sürece daha çok benzetmektedir.

MVM'nin bir Java standardı olarak geliştirilmesi girişimi de başlatılmıştır [80], ancak 2010 yılının başında Oracle şirketinin Java'nın marka haklarını satın almasıyla Barcelona projesi olarak adlandırılan bu girişim kısa süre sonra sonlandırılmıştır.

MVM'nin bulutlarda kullanılması düşünülmüşse de, MVM, bulutun ortaya çıkışından önce gelişen ve sonlanan dolayısıyla buluttan bağımsız bir girişimdir. Göz önünde tutulması gereken bir başka konu, özelleştirilmiş bir süreç sanal makinasının bulut ortamında kullanılmaya elverişli olmadığıdır. 3.3.2 ve 3.6 bölümlerinde belirtildiği gibi bulut dizgelerinde altyapı farklılıklarından olabildiğince kaçınılır. Bulutta konaklar arasında taşınmaya hazır ve her konakta aynı özellikte çalışma ortamı bulmayı bekleyen uygulamalar bulunmaktadır.

3.7.4 KaffeOS

KaffeOS Java süreç sanal makinasının üzerine alışılagelmiş işletim sisteminin bir örneğini kurmaya çalışan MVM'ye göre daha yeni bir yaklaşımdır [72]. Bu yolla sanal makinanın belleği içinde güvenilir ve güvenilmez alanlar belirlenebilir ve izlekler yalıtılabilir. Özelleştirilmiş süreç sanal makinalarının buluta uygun olmayışı konusundaki eleştiriler bu yaklaşım için de geçerlidir.

3.7.5 I-JVM

I-JVM [81], MVM'nin başarımdaki eksikliklerini kapatmak üzere her bir izleği yalıtmaktan vazgeçer. Bir sınıf yükleyicinin yüklediği izlekler kendi aralarında yalıtılmaz. Ancak farklı sınıf yükleyiciler tarafından yüklenmiş izlek toplulukları diğer sınıf yükleyicilerin yükledikleri izlek topluluklarından yalıtılır. I-JVM bu yönüyle 3.3.4.2 bölümünde anlatılan çalışmamızı [10] andırır. I-JVM'de de statik değişkenlerin tam anlamıyla yalıtıldığı söylenemez. Ancak, özelleştirilmiş bir süreç sanal makinası kullanımı yine kaçınılmazdır.

3.8 Vargı

Bu bölümde izlek tabanlı bilgiişlem ortamı sunan bulutlarda yalıtımın sağlanması için yöntemler önerilmiştir. Önerilen yöntemlerle izlekler kullanılırken müşterilerin dosya ve ağ erişimlerinin yalıtılabildiği, işlemci kullanımının sınırlandırılabildiği görülmüştür. Belleğin izlekler arasında paylaşılmamasını sağlayacak uygun bedelli bir çözüm bulmak söz konusu olmamıştır.

Belleğin yalıtımını yaptığı savında bulunan başka çalışmalar vardır. Ancak bu çalışmalarda kullanılan yöntemler de izlekleri hantallaştırmakta, giderek süreçlere benzetmektedir. Bu yolla, izlekler ortaya çıkışlarındaki işletim sistemine yük olmadan kolay bellek paylaşma aracı olma amaçlarından uzaklaşırlar.

Çalışmakta olan müşteri izleklerinin bir diğer müşterinin izleklerinden tam anlamıyla yalıtılamadığı durumda müşterilerin daha ileri düzeydeki varlıklarının, örneğin veritabanı erişimlerinin ya da dosyalarının şifrelenerek korunması boşa bir çabadır. Çünkü, yalıtılmamış bir izlek üzerinden bu varlıklara erişmenin yolu bulunabilir.

Hem kullanımı hem de yönetimi kolay olan süreçlerin yalıtımı işletim sistemi tarafından tanım gereği sağlanır. Güvenlik bakımından önemli sorunlar oluşturan yalıtım risklerini ortadan kaldırmak için, tezin ilerleyen bölümlerinde, süreç tabanlı bir yaklaşım etrafına kurulmuş bir bilgiişlem ortamı sunan bulut tasarımı içinde güvenlik düzeneklerinin geliştirilmesi çalışmaları sürdürülmüştür.

4. GÜVENLİ BİR BİLGİİŞLEM ORTAMI SUNAN BULUT İÇİN GÜVENLİK DÜZENEKLERİNİN TASARIMI

Bu bölümde tez çalışmalarının temelini oluşturan bilgiişlem ortamı sunan bulutlara ilişkin güvenlik sorunları belirlenmiş ve bu sorunların çözülmesi için geliştirilen önlemler anlatılmıştır. Bölüm boyunca işlemci gücünün süreçler yoluyla kullanıldığı genelgeçer bir bilgiişlem ortamı sunan bulut tanımı yapılarak bu bulutun güvenlik zayıflıkları tartışılmış ve bunlara karşılık kurulan düzenekler irdelenmiştir. Açıklanan düzenekler içinden özel çalışma gerektirenler izleyen bölümlerde ayrıntılarıyla ele alınmıştır.

4.1 Süreç Tabanlı Bilgiişlem Ortamı Sunan Bulutların Üstünlükleri

İlerleyen paragraflarda bilgiişlem ortamı sunan bulutları süreç tabanlı kullanmanın yararları ve bunun işlem gücünü izlekler aracılığıyla sunan kullanımdaki bulutlara [65, 75–78] göre üstünlükleri anlatılmıştır.

Bilgiişlem ortamı sunan bulutlardan bir müşterinin beklentisi, verilerinin bulut altyapısı üzerinde saklanması ve bu verilerle ilişkilendirdiği bir programın verileri kullanarak yine bulut altyapısı üzerinde bu programla tanımlanmış görevi yerine getirmesidir. Aynı müşteri bu hizmeti yerel bilgisayarındaki işletim sisteminde karşılamak isterse bir süreç çalıştıracaktır. Bir süreç bir görevi yerine getirmek üzere verileri program uyarınca kendi bağlamı içinde yürütür. Bu açıklamalara bakılarak kavramsal olarak işletim sistemi süreçleri ile bilgiişlem ortamı sunan bulutların birbiriyle uyumlu olduğu görülür. Buna karşın bilgiişlem ortamı sunan bulutlarda izlek tabanlı yaklaşımları yeğlemek uyumsuzluklara yol açacağından güvenlik ve kullanışlılık sorunlarını beraberinde getirir. İzleyen paragraflarda bilgiişlem ortamı sunan bulutlarda işlem gücünün izlekler yoluyla ve süreçler yoluyla kullanımı bu açıdan kıyaslanmıştır. Bu

kıyaslamada da veriyle ilişkilendirilmiş bir programı çalıştırmak konusunda izleklerin uyumsuzluğu ve süreçlerin buna doğal yatkınlığı sezilir.

Yapıları gereği, izlekler arasında bellek tam anlamıyla yalıtılamaz. Bu, en önemli güvenlik açığıdır. Bu konunun ayrıntıları 3. bölümde, özellikle 3.6 bölümünde bulunmaktadır. Süreçler kullanıldığında her program kendi bağlamında çalışacağı için müşteriler arasında bellek paylaşımı olmaz.

İşletim sistemleri uzun yıllar boyunca süreçlere yönelik geliştirilmiş olduğu için süreçlerin yönetimi görece kolaydır. İşletim sistemi kaynaklarının paylaştırılması, atanması ve kaynaklara erişim izinlerinin yönetilmesi izlekler için büyük güçlüklerle başarılır ya da başarılamazken; süreçlerde bunlar kolaylıkla yapılır.

İzlekler yapıları ve amaçları gereği güvenlik gözetilmeden tasarlanmıştır [63]. Oysa süreç benzeri ilk yapıların kurulduğu ilkel işletim sistemlerinden Multics [13] dahi güvenliği en başta gelen gereksinimlerden saymıştır.

Eşzamanlı çalışması düşünülen görevlerde de süreçlerin kullanılması büyük yarar sağlar. Bulutta izleklerin kullanımı eşzamanlı hesaplama gerektiren görevlerde eşgüdümü zorlaştırır. Bir yöntem, işe başlayan izleğin eşzamanlı görev yürütecek diğer izlekleri yaratması ve onların sonlanmasını beklemesidir. Ancak zaten kendisi sınırlandırılmış olan bir izleğin yeni izlekler yaratmasına izin verilmesi pek yeğlenen bir yol değildir. Bir başka yöntem, izleklerin ayrı ayrı koşullandırılarak buluta gönderilmesi ve aynı anda bulutta işlemeye başlamasıdır. Ancak bu yöntemde de bulutun aynı kullanıcının izlekleriyle farklı kullanıcıların izlekleri arasındaki farkı gözetmesinin bir yolu olmadığı için izlekler arası eşgüdümün sağlanması oldukça zor olur. Oysa, süreçler yoluyla işlemci gücü kullanılırsa, bir kullanıcının eşzamanlı hesaplama gerektiren bir görevi bir sürecin içinde tanımlı bulunan izlekler kullanılarak kolaylıkla ve belleğin paylaşımından ve süregiden işlemin sonlandırılamamasından kaynaklı güvenlik sorunları olmaksızın hesaplanabilir.

Kullanışlılığa bakıldığında, izleklerle işlem gücünü sunan bulutlarda müşterinin sağlayıcıdan bir istekte bulunduğu, isteğe karşılık olarak bir hizmetin yerine getirildiği ve bunun yanıtının müşteriye gönderildiği görülür. Bu döngü ne kadar hızlandırılsa da, etkileşimli bir kullanımdan söz edilemez. Ne var ki, sağlayıcının herhangi bir

biçimde döngüyü başlatma olasılığı yoktur. Oysa, süreçler kullanıldığında bir sürecin giriş, çıkış ve hata akımları ayrı ayrı kanallarla müşterinin uçbirimine bağlanabilir. Böylece etkileşimli kullanım sağlanır. Aynı akımlar bir başka sürece ya da müşterinin yine bulutta sakladığı bir dosyasına bağlanarak birbirini izleyen görevler de kolaylıkla yürütülür.

Açıklanan nedenlerle özel bir durum olarak 3. bölümde izleklerin yalıtımı anlatılmışsa da, tezde özellikle süreçleri kullanan bilgiişlem ortamı sunan bulutlara odaklanılması temel ilke olarak belirlenmiştir.

4.2 Belirlenen Güvenlik Sorunlarının Bilgiişlem Ortamı Sunan Bulutlardaki Görünümü

Bilgiişlem ortamı sunan bulutların güvenlik sorunlarının kaynakları çeşitlidir. Bazı güvenlik zayıflıkları haberleşme gereklerinden ortaya çıkar, bazıları her dizgede var olabilecek güvenlik zayıflıklarının yeni görüntüleridir, bazıları da bilgiişlem ortamı sunan bulutun oluşturulması için gereken yapı kurulurken ortaya çıkar. Bu zayıflıkların yol açtığı bilgiişlem ortamı sunan bulutlara özgü tehditler aşağıda listelenmiştir.

4.2.1 Müşteri verilerinin yitirilmesi ($T_{1,1}$)

Bu tehdit daha önce 2.2.7. bölümde *"Müşteri verilerinin sağlayıcı bilgisayarlarına taşınmasıyla müşteri denetiminde olmayan verilerin yitirilmesi tehdidi ortaya çıkar. Tehdidin kaynağı kullanıcılar ya da bilinçli veya bilinçsizce sağlayıcının kendisi olabilir."* açıklamasıyla verilmişti. Müşteri, verilerinin denetimini gönüllü olarak sağlayıcıya bırakmaktadır. Bu nedenle verinin kullanıcı saldırılarına karşı korunması sorumluluğunun da sağlayıcıda olduğu varsayılmalıdır. Bu sadeleştirmenin ardından tehdit *müşteri verilerinin sağlayıcı tarafından yitirilmesi* olarak düşünülür.

4.2.2 Müşteri verilerinin sızması ($T_{1,2}$)

Bu tehdit daha önce 2.2.7. bölümde *"Müşteri denetiminde olmayan veriler üçüncü kişiler ya da sağlayıcı tarafından okunabilir. İlke olarak, verinin yetkisi olmayan biri tarafından okunması ile sızması arasında fark yoktur."* açıklamasıyla verilmişti. Bu

tehdit bilgiişlem ortamı sunan bulutlar için düşünüldüğünde müşteri programlarının ve bunlara ilişkin verilerin sağlayıcı tarafından okunmasının sakıncalı olduğunu vurgular. Müşteri oluşturmuş olduğu programın işleyişini sağlayıcıdan gizlemek isteyebilir, hatta yasal olarak bu işleyiş bir patentin konusu olabilir ve gizlenmesi gerekebilir. Müşteri programlarının yürütme sırasında kullandıkları ve yürütmenin ardından oluşturdukları veriler kişiye özel ya da ticari sırlar içerebilir ve gizli tutulmaları istenebilir. Sayılan nedenlerle, program işleyişlerinin ya da verilerin gizli tutulmaması bilgiişlem ortamı sunan bulutlarda bir tehdittir.

4.2.3 Müşteri programının sağlayıcıya zarar vermesi ($T_{2,1}$)

Bu tehdit daha önce 2.2.7. bölümde *"Müşteri kendi oluşturduğu programları bulutta çalıştırma yetkisini elinde bulunduracağından sağlayıcı bu programların bulut altyapısına zarar vermeyeceğinden emin olmak ister."* açıklamasıyla verilmişti. Bilgiişlem ortamı sunan bulutların çalışma ilkesi gereği, sağlayıcı, müşterinin oluşturduğu ya da edindiği programları işleyişlerini gözetmeden çalıştırmak durumundadır. Sağlayıcı, işleyişlerinden habersiz olduğu programların bulut altyapısına zarar verecek nitelikte olup olmadıklarını kestiremeyeceği için bir tehdit söz konusudur.

4.2.4 Müşteri programının sağlayıcıdan zarar görmesi ($T_{2,2}$)

Bu tehdit daha önce 2.2.7. bölümde *"Müşteri, programları uygun biçimde çalıştığı sürece yapmakta olduğu işin sekteye uğramamasını bekler. Sağlayıcının müşteri programlarına yapacağı bozucu etkilerin engellenmesi gerekir."* açıklamasıyla verilmişti. Bilgiişlem ortamı sunan bulutların çalışma ilkesi gereği, sağlayıcı, müşterinin oluşturduğu ya da edindiği programları bir hata ya da zorunluluk olmadığı sürece kesintiye uğratmadan çalıştırmak durumundadır. Programın yürütülmesine sağlayıcı tarafından bilinçli veya bilinçsiz yapılacak bozucu bir etki kullanıcının çalışmasına bir tehdit oluşturur.

4.2.5 Müşterinin diğer müşterilerden zarar görmesi ($T_{3,1}$)

Bu tehdit daha önce 2.2.7. bölümde *"Kaynak paylaşımından söz edilmesiyle birlikte bir müşterinin (Z_1 ve Z_2 ile belirtilen) sağlayıcı denetiminde bulunan veri ve programlarının sağlayıcı dışında diğer müşteriler ve onların programlarının kullanıcıları tarafından da bilinçli ya da bilinçsizce zarara uğratılması riski doğar."* açıklamasıyla verilmişti. Bu tehdit Z_1 ve Z_2 zayıflıklarından kaynaklanan tüm tehditlerin sonucunda ortaya çıkabileceği için, alınacak önlemin yukarıda sayılan diğer dört tehdidin önlenmesine de dolaylı yoldan katkı sunacağı düşünülür. Bilgiişlem ortamı sunan bulutlar bilgiişlem kaynaklarının müşteriler arasında paylaşıldığı altyapılardır. Bilgiişlem kaynaklarının ortaklaşa kullanımı müşterilerin bu kaynaklar üzerinden birbirlerinin veri ve programlarına erişmesi olasılığını ortaya çıkarır. Bu erişimin engellenmemesi durumunda aynı bulut altyapısını kullanan müşterilerin birbirlerinin veri ve programlarına erişmesi ile kişisel ya da ticari sırların açığa çıkması ya da verilerin tümüyle bozulması tehdidi oluşur.

4.2.6 Bulutta gerçekleşen olayların hatalı kaydedilmesi ($T_{4,1}$)

Bu tehdit daha önce 2.2.7. bölümde *"Bulutta gerçekleşen olayların hatasız ve yadsınamaz biçimde kaydedilmemesi durumunda gerektiğinde etkinliklerin sorumluları kesin olarak belirlenemeyeceğinden tüm paydaşlar için doğrudan ya da dolaylı tehditler söz konusu olur."* açıklamasıyla verilmişti. Bulutta gerçekleşen olayların hatasız kaydedilmesi sağlayıcı, müşteri ve kullanıcı arasındaki karşılıklı yükümlülüklerin yerine getirilip getirilmediğinin izlenmesi için bir gerekliliktir. Olayları kaydedecek tarafın güvenilirliği de tartışmanın önemli bir bölümünü oluşturur. Bundan başka, gerekli durumlarda yasal soruşturmanın tutulan kayıtlara göre yürütülmesi ve hataların ya da kusur sayılabilecek olayların sorumlularının bu yolla bulunması da olasıdır. Sayılan nedenlerle bulutta gerçekleşen olayların kaydedilmesinde hata veya eksiklerin olması bulutta yer alan tüm taraflar için doğrudan ya da dolaylı tehditler içermektedir.

4.3 Bilgiişlem Ortamı Sunan Bulutlarda Karşılaşılan Güvenlik Sorunlarına Çözümler

Yukarıda sayılan tehditlere karşı önlemler anlatılırken konuyu bir bütünlük içinde izlemeyi kolaylaştırmak için birbiriyle ilişkili önlemler bir arada anlatılmıştır. Böylece yukarıda verilen tehditlerin sırasından farklı yeni bir sıralama ile önlemler anlatılır ve bazı tehditler için alınacak önlemlerin aynı olmaları nedeniyle anlatılanlar birleştirilir. Önlemler ile tehditler arasındaki ilişkinin izlenmesi için 2.2.6 bölümünde belirtildiği biçimde indisler kullanılmıştır.

4.3.1 Bizans yetersayı kümeleri ($Ö_{1,1,1}$ ve $Ö_{2,2,1}$)

Bizans yetersayı kümeleri [82, 83] hem *müşteri verilerinin yitirilmesi ($T_{I,1}$)* hem de *müşteri programının sağlayıcıdan zarar görmesi ($T_{2,2}$)* tehditlerine karşı kullanılacak bir önlemdir. Bu yaklaşımda aynı verinin ya da programın kopyalarının herhangi bir alt kümesi bir küme oluşturur. Küme içindeki kopyaların bazılarının farklılıklar göstermesi durumunda hatanın veya kötü niyetle bozmanın olduğu varsayılarak yetersayının belirlediği durum doğru kabul edilir.

Bizans yetersayı kümelerinin müşteri verilerinin yitirilmesi tehdidine karşı kullanıldığı durumlarda yararı, yalın biçimde, verinin birden çok kopyasının bulunması ve olası hata durumlarında güvenilir bir kopyanın belirlenebilmesidir.

Müşteri programının sağlayıcıdan zarar görmesi tehdidinin önlenmesi sırasında ise daha farklı bir kullanım gerekir. Bu durumda kümeyi oluşturan konakların program yürütme yeteneğinin bulunduğu varsayılır. Bir programın doğru biçimde yürütüldüğünden emin olmak için aynı programın kopyaları farklı sağlayıcıların konaklarına gönderilir ve aynı ayarlarla çalıştırılır. Yürütme sonunda programların aynı biçimde sonuçlanmaları beklenir. Farklı çıktılar alınacak olursa, yetersayının verdiği sonuç doğru kabul edilir.

Bizans yetersayı kümelerinin verilerin yitirilmesine karşı önlem olarak kullanımının veri saklama açısından getireceği yük olası diğer önlemlerle kıyaslandığında önemli bir fark göstermez. Verilerin yitirilmeye karşı yedeklenmesinde sık kullanılan yaklaşım 3-2-1 kuralıdır [84]. Bu kurala göre bir verinin en az üç kopyası en az iki farklı

saklama ortamında ve en az biri farklı konumda olacak biçimde saklanmalıdır. Bizans yetersayı kümeleri yaklaşımında da verilerin en az üç kopyasının bulunması, oranlar içinde yetersayının belirlenmesi için genellikle gereklidir. Bulutta saklama ortamları konusunda farklılıklar müşteriden soyutlanmıştır, ancak en az iki farklı sağlayıcıya gönderilmiş kopyaların iki farklı ortamda bulunduğu düşünülebilir. Bunlardan biri yitirilse dahi ötekinin sağlam kalacağı böylelikle varsayılır. Konumun farklılığı ise bulutun doğasında vardır. Bizans yetersayı kümeleri, bulutlarda bu biçimleriyle alışıldık veri yedekleme yaklaşımlarının yerini almaya adaydır.

Bizans yetersayı kümeleri programların güvenilir biçimde yürütüldüğünden emin olmak üzere kullanıldıklarında müşteriye getirecekleri fazladan parasal yük kopya sayısı ile doğru orantılı artar. Müşteri bilgiişlem hizmetini kendisi karşılamak istemediği için buluttan bilgiişlem hizmeti almaya yönelmişken yapılan hesaplamayı doğrulamak için birkaç kat fazla hizmet alımı yapmasını beklemek gerçekçi bir yaklaşım değildir. Anlatılan nedenle, Bizans yetersayı kümelerinin programlar için kullanımını bilgiişlem ortamı sunan bulutlarda kullanıcıya bırakmak gerekir.

Bunun dışında, Bizans yetersayı kümeleri üzerinde aynı programın birden çok kopyası yürütülecek olursa ortaya çıkacak sonuçların anlaşılması da ayrı bir sorun olur. Programların sonuçlarının incelenerek hangisinin doğru olduğuna bulutta karar verilmesi durumunda bu kararın doğruluğundan da emin olunması gerekir. Bu durumda, kopyaların sonuçları ortaya çıkınca ortak doğru sonuca karar verilmesi için her bir programın hesapladığı sonucu müşteriye göndermesi gerekir. Oysa, karar verme işinin kendisi de hesaplama açısından müşterinin altından kalkamayacağı büyüklükte bir yük olabilir. Bizans yetersayı kümeleri eskiden beri üzerinde çalışmalar yapılan bir konudur. Hesaplamanın yapılması ve özlük bilgilerine zarar verilmeden sonuca karar verilmesi sorunu bulutlar ortaya çıkmadan önce, dağıtık bilgiişlem dizgeleri için çözülmüştür [85]. Verimli çalışan protokollerin bulutlara nasıl uygulanacağına ilişkin çalışmalar sürdürülmektedir [86, 87].

Bizans yetersayı kümelerinin kullanımı *yararlanırlığın sağlanamaması (Z_7)* ve *hata hoşgörüsünün olmayışı (Z_8)* zayıflıklarının oluşturacakları tehditlere karşı da yerinde bir önlem olacaktır. Yararlanırlığın sağlanamaması sonucu oluşacak veriye veya programa erişememek tehditleriyle karşılaşan müşteri veya kullanıcı Bizans yetersayı

kümesi içinde çoğullanmış kopyalardan birine erişerek devre dışı kalmış sağlayıcıya karşın işini sürdürür. Öte yandan, bulut altyapısı hata hoşgörüsü gözetilmeden kurulmuş olsa dahi, Bizans yetersayı kümesi oluşturulduğunda farklılık gösteren hatalı veriler kolaylıkla belirlenebilir ve onarılabilir. Böylece, kopyaların bir bölümünün hatalı ya da hasarlı oluşundan etkilenmeden müşteri ve kullanıcıların çalışmalarını sürdürmesi sağlanır.

4.3.2 Erişim Denetimi

Erişim denetimi müşterilerin diğer müşterilerden ve kullanıcılarından zarar görmesini engellemek üzere alınan önlemlerden biridir. Genellikle üç aşamadan oluşur. Bunlar asıllama, yetkilendirme, denetlenebilirlik olarak sıralanır. Asıllamada, bir öznenin veri ya da programa erişmek üzere geçerli bir belgeyi bulundurup bulundurmadığı belirlenir. Yetkilendirmede, asıllamada belirlenen belgeye dayanılarak ilgili verilere erişim sağlanır. Denetlenebilirlikte ise, asıllama ve yetkilendirmede gerçekleşenlere ilişkin kayıtlar oluşturulur.

4.3.2.1 Özlük bilgilerinin bir bölümüyle asıllama

Özlük bilgilerinin korunabileceği en önemli aşama bir öznenin kendini bir dizgeye tanıttığı asıllama aşamasıdır. Bu aşamada tanıtım için sunulan belgenin kimlik olması gerektiği yanılsamasına sık düşülür. Oysa tanıtımda gerekli olan *yetkinlik* belgesinin kimlik bilgilerinin tümünü içermesine gerek yoktur, hatta bu özellikle istenmeyebilir. Örnek olarak, bir üniversitenin giriş kapısındaki güvenlik görevlisinin bilmesi gereken sadece içeri girenlerin üniversitenin kayıtlı öğrencileri olup olmadıklarıdır. Güvenlik görevlisinin kapıdan giriş yapan herkesin tüm kimlik bilgilerini görmesi gerekmediği gibi, bu, üniversiteye girenlerin kişisel saldırılara uğramasına olanak verebilir.

Aynı biçimde, veriye ya da programa erişmek için sağlayıcıya özlük bilgilerinin tamamını vermek gerekli değildir, erişim için yetkinliği belgelendirmek yeterli olmalıdır. Bu yaklaşım kullanıcının tüm özlük bilgilerinin gereksiz yere açığa çıkmasını büyük ölçüde engeller. Veri ve programlara erişim için gereken yetkinliğin ne olacağına müşteri ile sağlayıcı arasında yapılacak hizmet sözleşmesi kapsamında karar verileceği varsayılmıştır.

Kullanıcıların tüm kimlik bilgileri yerine kimlik bilgilerinin bir bölümünü ya da kimlik bilgilerinden türetilebilecek bazı özellikleri taşıyan belgelerin kullanıcı istekleri doğrultusunda hazırlayan vekil sertifika yetkelerinin oluşturulması geçmiş çalışmamızda önerilmiştir [9]. Bu yetkelerden alınacak vekil sertifikalar bulutta asıllama sırasında kimlik yerine kullanılabilir, böylelikle yetkinlik belgelendirilir.

Vekil sertifika yetkelerinin [88] çalışma ilkesini açıklamak için bir açık anahtar altyapısı [89] içinde gerçek kimliklerin nasıl belgelendiğini göstermek gerekir. Gerçek ve tüzel kişilerin gizli ve açık anahtarlarının bulunduğu düşünülen bir ortamda bir yetke kimlikleri doğrulamak üzere yetkilendirilir. Bu kurum çoğunlukla devlete bağlı bir birim ya da toplumca saygın bir sivil toplum kuruluşudur. Yetke öncelikle kendi kimliğini kendi adına imzalayarak bir sertifika oluşturur. Bu sertifika yetkenin kimliğini belgelendirdiği gibi, diğer tüm sertifikaların da kökü olacaktır. Bu nedenle bu kuruma *kök sertifika yetkesi* de denir. Daha sonra bu yetke kendisine başvuruda bulunan kişilerin kimliklerini denetleyerek onlar için de birer sertifika oluşturmak üzere bir sunucu kurar. Yetke tarafından kişiler için oluşturulan sertifikalar yetke tarafından imzalanmıştır. Dolayısıyla, yetkenin sertifikasında belirtilen açık anahtarla bu imzaların doğruluğu denetlenebilir. Böylece, bir kimliğin belgelenmesi için kişinin kendi sertifikası ve sertifikayı imzalayan yetkenin sertifikası bir arada gerekli olur. Yetke, görevi tek başına yürütmekte iş yükü açısından zorlanabileceği için, başka yetkelere sertifika oluşturma izni verebilir. Bu yetkeler de daha başka yetkelere izin verebilir. Böylece sertifika sağlamakla görevli yetkelerden oluşan bir ağaç yapısı kurulur. Bu yapının içinde bir kimliği belgelendirmek isteyen kişi, kendi sertifikasını ve sırasıyla bu sertifikadan kök sertifikaya kadarki imza zincirini oluşturan tüm yetkelerin sertifikalarını doğrulatmak durumundadır.

Vekil sertifika yetkeleri ise bu açık anahtar altyapısına eklendiklerinde ancak sertifika sağlama ağacının yaprakları olabilir. Bir başka deyişle, kişiler ellerindeki bir sertifikayla vekil sertifika oluşturan bir yetkeye başvuruda bulunacaklardır. Vekil sertifika yetkesi kimliğin doğruluğunu denetlemekle ilgilenmez. Var olan sertifikaya dayanarak, kişinin isteği üzerine sertifikanın belli alanlarını gösteren, geri kalanını gizleyen yeni sertifikalar oluşturur. Böylece, kişiler isterlerse vekil sertifika yetkelerine başvuruda bulunarak kimliklerine belirleyen bazı özellikleri içeren ancak adlarını sak-

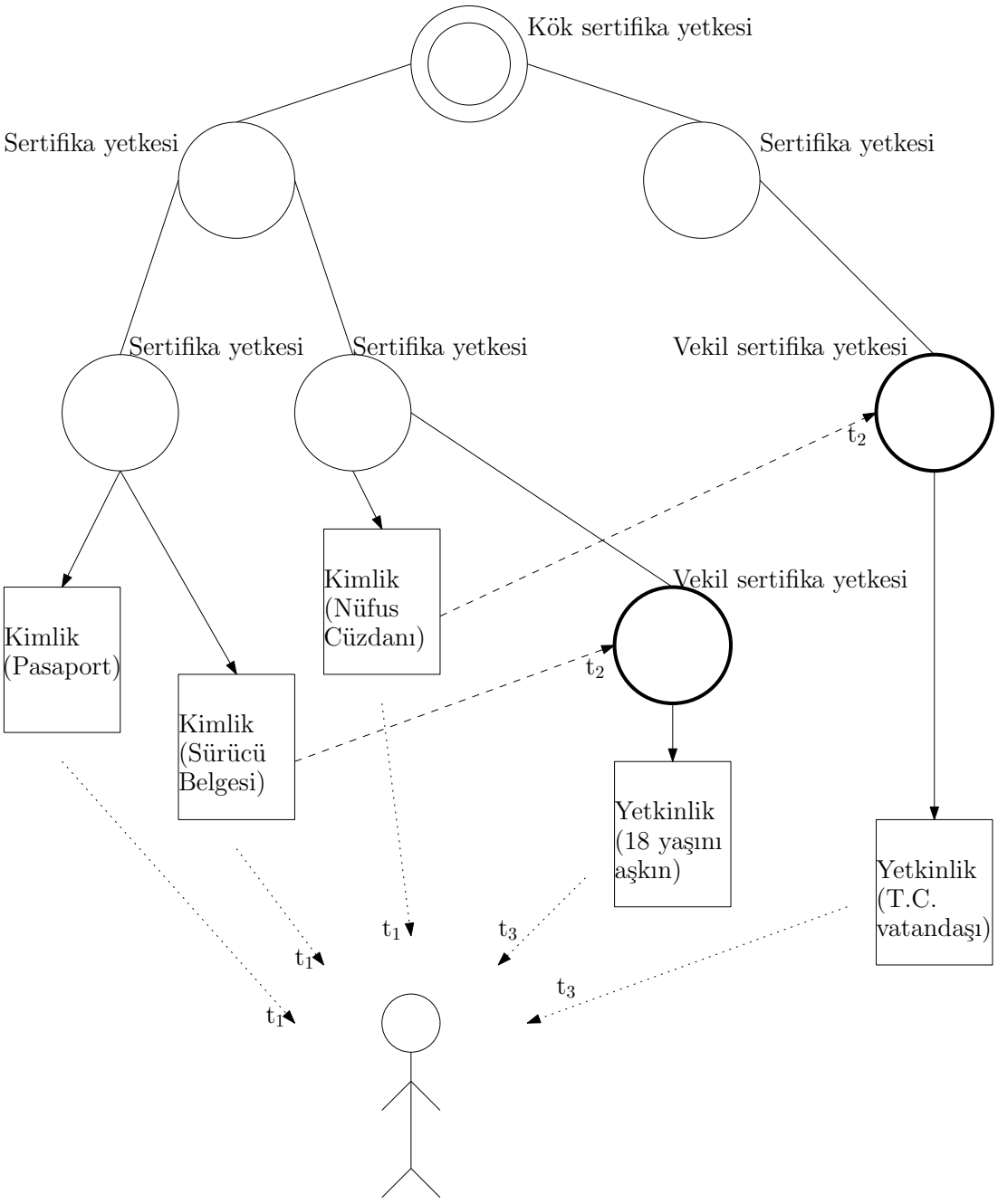
layan sertifikalar edinebilir. Örnek olarak, kişinin belli bir yaştan büyük olduğunu gösteren bunun dışında hiçbir ayrıntı vermeyen bir sertifika oluşturulabilir. Vekil sertifika yetkesi asıl sertifikaların ve bunlara dayanarak oluşturduğu vekil sertifikaların kaydını tutma sorumluluğunu üstlenir. Böylece kişilerin özlük bilgilerini koruma olanağı elde edilirken yasal gerekliliklerde asıl kimliklerin elde edilmesi de sağlanır.

Şekil 4.1'de açık anahtar altyapısı içinde yer alan sertifika yetkeleri, kök sertifika yetkesi ve vekil sertifika yetkelerinin oluşturduğu bir ağaç yapısı betimlenmiştir. Şekilde gösterilen örnekte, kullanıcı, ilk aşamada iki farklı sertifika yetkesinden kimliğini açıkça gösteren üç ayrı sertifika almıştır. Ardından, edindiği sertifikaların ikisi ile iki ayrı vekil sertifika yetkesine başvuruda bulunmuştur. Kullanıcı, son aşamada, yetkelerin birinden yasal erginliği konusunda bir vekil sertifika edinirken diğerinden Türkiye Cumhuriyeti vatandaşlığı konusunda bir vekil sertifika edinmiştir.

Sağlayıcının asıllama sırasında kimlik yerine bu vekil sertifikalara dayanarak karar vermesinin erişim denetiminin denetlenebilirlik aşamasında oluşan kayıtların geçerliliği üzerine olumsuz etkileri olacağı eleştirisi yapılabilir. Ancak, vekil sertifika yetkelerinin vekil sertifikalarını gerçek kimliklere dayanarak verdiği göz önüne alınırsa vekil sertifikaların da kimlikler kadar geçerli olduğu anlaşılır. Yasal soruşturmalar gibi zorunlu durumlarda vekil sertifika yetkelerinin kayıtları incelenerek her zaman gerçek kimliklere ulaşılması olanaklıdır.

Kimliklerin ve yetkinlik belgelerinin geçerliliğinin eşitliğinin açık kanıtı, her ikisinin de elektronik sertifika olarak saklandığı durumlarda bir sertifika zincirinin son halkası olmasıdır. Vekil sertifika yetkesi yetkili bir sertifika sağlayıcısı olarak düşünüldüğünde, zincir üzerinde yer alan herhangi bir sertifika sağlayıcıdan özünde farklı olmayacaktır. Fazladan yapılması gereken, vekil kimlik yetkesinin vekil sertifikalar ile asıl sertifikalar arasındaki ilişkiyi saklama sorumluluğunu üstlenmesidir. Bu durum Şekil 4.2'de görselleştirilmiştir.

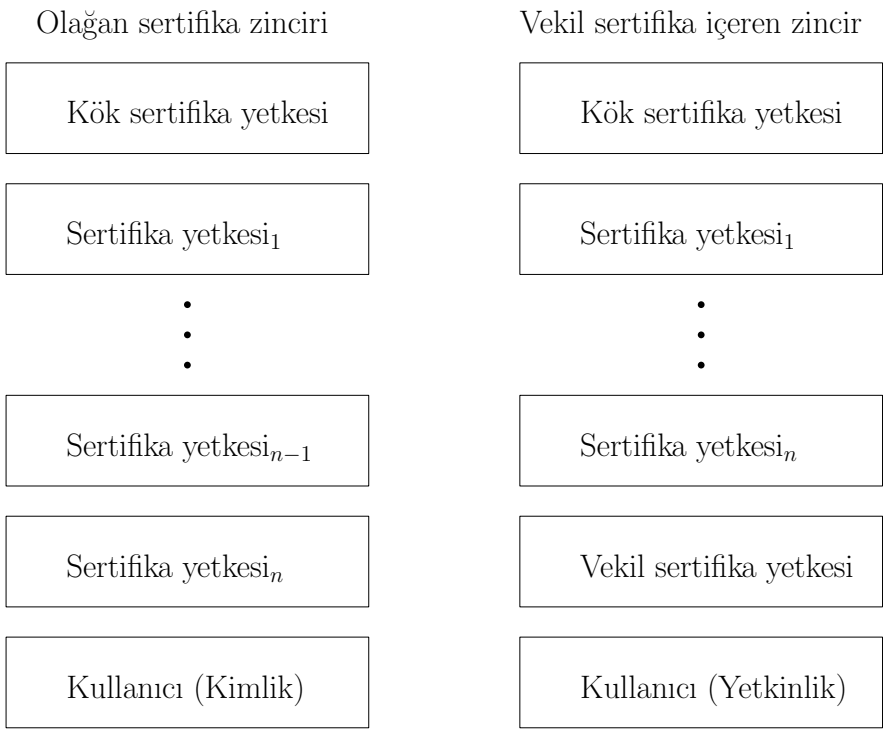
Asıllama aşamasında özlük bilgilerinin açığa çıkmasının engellenmesiyle *özlük bilgilerinin gizlenememesi (Z_6)* zayıflığının yaratacağı pek çok tehdide karşı yerinde bir önlem alınmış olur.



Şekil 4.1 : Açık anahtar altyapısı içinde vekil sertifikaların elde edilmesi.

4.3.2.2 Asıllama ve haberleşme güvenliği

Haberleşme güvenliğini sağlamanın iyi bilinen ve onay görmüş bir yolu TLS [90] kullanmaktır. Bu yaklaşımda, belirlenen oturumlar boyunca uçtan uca güvenli haberleşme kanalları oluşturmak, bu kanallar içinden akan bilginin güvende kalmasını sağlamak olanaklıdır. Oturumlar oluşturulurken taraflardan birinin veya her ikisinin sertifikaları kullanılabilir. Karşılıklı asıllama yapılırsa her iki tarafın da birbirinin kimliğinden ya da yetkinliğinden emin olması sağlanır. TLS ve vekil sertifika yetkelerini ortaklaşa



Şekil 4.2 : Olağan sertifika zinciri ile vekil sertifika yetkesi içeren bir sertifika zincirinin kıyası.

kullanmanın yararı burada açıkça ortaya çıkar; aynı yapıdaki asıl ve vekil sertifikalar, açık anahtar altyapısı içinde TLS yaklaşımıyla uyumlu çalışır.

TLS ile güvenli bir kanal oluşturulurken öncelikle taraflardan biri istemci sıfatıyla sunucu olan diğerine başvuruda bulunur. Kendisinin hangi kriptolojik algoritmalarını kullanabildiğini ve hangi güvenlik düzeyini yeğlediğini belirtir. Bunun üzerine her iki taraf için de uygun olan güvenlik düzeyinin belirlenmesi için bir algoritma yürütülür. Bu algoritmanın tamamlanıp güvenlik düzeyi üzerinde anlaşılmasının ardından ya sadece sunucunun sertifikasına dayanılarak ya da seçime bağlı olarak hem sunucu hem de istemcinin sertifikalarına dayanılarak bakışımsız şifreleme kullanılarak bir anahtar değişimi algoritması yürütülür. Anahtar değişimi algoritması ile oluşturulan kanalda sadece o oturum için geçerli bir bakışımlı anahtar her iki tarafta da elde edilir. Oturumun devamında elde edilen bakışımlı anahtar kullanılarak güvenli haberleşme sağlanır.

Tarihsel olarak 1995 yılında Netscape tarafından pazara sunulan SSL [91], TLS protokolünün öncülüdür. TLS protokolünün de yıllar içinde farklılıklar göstererek gelişen sürümleri olmuştur. Bu evrim yıllar içinde ortaya çıkan çeşitli tehditler, saldırılar ve gelişmelerin sonunda oluşmuştur. Güncel bir TLS protokolünün güncel kriptolojik al-

goritmalarla birlikte kullanılması sadece bulut bağlamında değil, güvenlik açısından tüm güvenli haberleşme çözümlerinde salık verilir.

Bilgiişlem ortamı sunan bulutlar için, müşterinin sağlayıcının kimliğini asıllayarak, sağlayıcının da müşterinin kimliğini asıllayarak güvenli kanallar oluşturdukları oturumların kurulması uygun bir çözümdür. Böylece sağlayıcı ile müşteri arasında kimliklerin açıkça görülebildiği doğrudan ilişkiler kurulmuş olur. Ancak, kullanıcıların müşteri programlarına erişimi sırasında kimlik bilgilerini açıkça vermeleri gerekmeyebilir. Bu durumda kullanıcılar bir vekil sertifika yetkesi üzerinden aldıkları vekil sertifika ile kimlik bilgilerini göstermeden, sadece programa erişmeye olan yetkinliklerini belgelendirerek TLS ile asıllamayı tamamlar.

4.3.2.3 Yetkilendirme ($\ddot{O}_{3,1,2}$)

Yukarıda anlatılan yollarla kimliği ya da yetkinliği belirlenmiş bir kullanıcı ya da müşterinin, bulut sağlayıcısının sunduğu kaynaklara erişimi sağlandığında erişim yetkilerinin belirlenmesinin nasıl yapılacağı ve nasıl sınırlandırılacağı yetkilendirme konusu içinde ele alınır. Müşteri programlarının ve verilerinin bulutta nasıl saklanacağına ayrıca bunlara erişimin nasıl uygulanacağına ilişkin tartışma 5. bölümde ayrıntısıyla sürdürülmüştür. Bu nedenle burada kısaca yetkilendirmenin ilişkili olduğu bileşenler sayılmıştır.

Sağlayıcı, müşteri program ve verilerini müşterinin belirlediği sınırlar çerçevesinde kullanıcılara sunar. Müşteri, bulut altyapısına taşıdığı veri ve programlar üzerinde erişim kurallarını tanımlar. Erişim kurallarını olduğu gibi uygulamak öncelikle sağlayıcının sorumluluğundadır. Sağlayıcı bunu sağlamak üzere müşterinin erişim kurallarını izlemesi ve uygulaması için her konakta bir kural uygunluk noktası oluşturur. Kullanıcıların bulut üzerindeki herhangi bir kaynağa uzaktan erişimi her zaman kural uygunluk noktaları aracılığı ile kurulur. Böylece tüm uzaktan erişimlerin sağlayıcının uyguladığı müşterinin erişim kuralları çerçevesinde olması sağlanır.

4.3.2.4 Denetlenebilirlik (Güvenli günlük tutma düzeneği, $\ddot{O}_{4,1,1}$)

Bulutta gerçekleşen olayların gündelik yaşamda noter örneğinde olduğu gibi yadsınamaz ve silinemez bir biçimde kaydedilmesi pek çok olası uyuşmazlığı ortadan kaldırır.

Bulutta tutulan veri ve programlara kimin, ne zaman, ne biçimde eriştiğinin bilinmesi ve buna ilişkin kayıtların taraflardan hiçbirinin yadsıyamayacağı biçimde bulunması bulutla ilişkili tüm özneleri (sağlayıcıyı, müşterileri ve kullanıcıları) dürüst davranmaya zorlar.

Ancak, kayıtların sadece sağlayıcının konaklarında saklanması müşteriler ve kullanıcılar açısından yeterli güveni vermez. Günlükler için tüm tarafların etkinliklerini kaydeden ortak bir saklama alanının bulunması, her tarafın kendi kayıtlarını ayrıca saklaması, üçüncü tarafların kayıtlara tanıklık edebilmesi buluttaki güven ilişkilerini daha iyi yansıtır.

Bu yaklaşıma uygun, bulutla uyumlu bir güvenli günlük tutma düzeneği 6. bölümde ayrıntısıyla ele alınmıştır.

4.3.2.5 Şifreleme ($\ddot{O}_{1,2,1}$)

Yetkilendirmeyle bir arada ve sıkı ilişki içinde çalışması öngörüldüğü için erişim denetimi başlığının kapsamında anlatılan ve erişim denetiminin geleneksel aşamalarının yanına eklenecek bir önlem de şifrelemedir.

Müşterinin belirleyeceği erişim kuralları doğrultusunda müşteri veri ve programlarına hakkı olmayan kişilerin bilinçli ya da bilinçsiz erişimlerini engellemek için şifreleme kullanılır. Veri ve programlar sadece ilgili kişilerin anahtarlarına uygun şifrelenmeli ve erişim kurallarının değişmesiyle birlikte güncellenerek değişikliklere uydurulmalıdır. Böylece yetkilendirme konusunda sağlayıcının sorumluluğu hâlâ geçerli olsa da, bunun yanına kriptolojik bir koruma düzeneği daha eklenmiş olur.

Şifreleme ve yetkilendirmeyi bir arada yürütmek üzere tasarlanan bir düzenek ayrıntısıyla 5. bölümde açıklanmıştır.

4.3.3 Güvenli hesaplama katmanı ($\ddot{O}_{2,1,1}$ ve $\ddot{O}_{3,1,1}$)

Müşterinin, sağlayıcı ile yapmış olduğu hizmet sözleşmesi uyarınca davranması beklenir. Müşteri programlarının hizmet sözleşmesi ile çerçevelenen koşullar içinde kalması için sağlayıcı tarafından ABD Savunma Bakanlığı'nın bir raporunda [92] kap-

samlı olarak tanıtılan güvenli hesaplama katmanı kullanılmıştır. Güvenli hesaplama katmanı aynı zamanda programların birbirlerine etkilerini engeller, böylece müşteriler aynı fiziksel altyapı üzerinde birbirlerine zarar vermeden bir arada bulunabilir.

Güvenli hesaplama katmanı oluşturmak için yaygın kullanılan yapılardan biri olan Java süreç sanal makinası [70] seçilmiştir. Müşteri programları önceden belirlenmiş kurallar çerçevesindeki süreçler olarak sağlayıcı denetimindeki konaklarda çalışacaktır. Konaklardaki kural uygunluk noktaları müşteri programlarının varsa belirlenen standartlara uygunluğunu belirler. Programın hizmet sözleşmesinde belirtilen kaynak kullanım kurallarına uygunluğu da kural uygunluk noktasında incelenir. Güvenli hesaplama katmanı kaynak kullanım kurallarına göre koşullandırıldıktan sonra programın yürütülmesine başlanır.

Kaynak kullanım kurallarının hizmet sözleşmesi uyarınca düzenlenmesi de çoğu zaman erişim kuralları ile örtüşür. Yetkilendirme sırasında bir müşterinin bir kullanıcıya verdiği dosya erişim hakkının karşılığı gerçekte sağlayıcının bir konağının diski üzerinde bir dosyanın erişime açılması anlamını taşır. Bu nedenle kural uygunluk noktası uygun dönüşümleri yapacak yetenekte olduğu sürece hem erişim kurallarını hem de işletim sistemi düzeyindeki kaynak kullanım kurallarını bir arada yürütür.

Bulut, yapılan tanıma göre, müşteri programlarının gerektiğinde konaklar üzerinde yer değiştirebileceği, konumdan bağımsız bir hesaplama altyapısıdır. Buna göre, bir müşteri programının her konakta aynı biçimde çalışabilmesi için ortak bir altyapıda yürütülmesi ve ortak bir yapıda oluşturulması gerekir.

Java süreç sanal makinasının güvenli hesaplama katmanı olarak seçilmesi ile sağlayıcılar arasındaki olası uyumluluk sorunlarının en aza indirilmesi hedeflenmiştir. Böylelikle hem ortak bir altyapı sağlanır hem de *standart arayüzlerin eksikliği (Z_5)* zayıflığının oluşturacağı tehditlere karşı bir önlem alınmış olur.

Güvenli hesaplama katmanı üzerinde yürütülecek müşteri programlarının nasıl oluşturulacağı, nasıl çalışacağı, bu sırada müşteri verileri ile programlarının nasıl korunacağı şifrelemenin ve yetkilendirmenin ayrıntılarıyla birlikte izleyen 5. bölümde ele alınmıştır.

4.4 Vargı

Bilgiişlem ortamı sunan bulutlarda süreç kullanmanın güvenlik ve kullanışlılık bakımından izlek kullanmaya göre üstünlükleri bölümün başında belirtilmiştir. Bulutlarda karşılaşılan tehditlerin bilgiişlem ortamı sunan bulutlardaki izdüşümleri açıklanmış bunlara uygun önlemlerin neler olacağı belirlenmiştir.

Süreç tabanlı bir bilgiişlem ortamı sunan bulut tasarımının güvenli olması için, bulutta ortaya çıkan tehditlere karşılık gelen önlemler ve bunun yanında tasarlanan bulutun temel güvenlik gereksinimlerini karşılayacak diğer önlemler ayrı ayrı açıklanmıştır. Böylece güvenli bilgiişlem ortamı sunan bulut tasarımının çerçevesi çizilmiştir.

Kapsamlı açıklama gerektirdiği düşünülen süreç yalıtımına ilişkin açıklamalar 5. bölümde, güvenli günlük tutma düzeneğini anlatan açıklamalar 6. bölümde yapılmıştır.

5. SÜREÇ TABANLI BİLGİİŞLEM ORTAMI SUNAN BULUTTA HİZMET YALITIMI

Tasarlanan bilgiişlem ortamı sunan bulutta süreçlerin yalıtımının sağlanması güvenlik gereklerinin önde gelenlerindendir. Yalıtımın sağlanması için 4. bölümde sözü edilen önlemler arasından *yetkilendirme ($Ö_{3,1,2}$)*, *şifreleme ($Ö_{1,2,1}$)* ve *güvenli hesaplama katmanı ($Ö_{2,1,1}$ ve $Ö_{3,1,1}$)* önlemleri bir arada kullanılarak süreç tabanlı bilgiişlem ortamı sunan bulutlarda programların daha güvenli işletilmesi için elde edilen yeni bütünleşik düzenek bu bölümün konusunu oluşturur. Bu bölümün içeriği yöntemin ilk kez duyurulduğu yayınımızla [11] örtüşmektedir.

5.1 Konaklar Üzerindeki Koruma Önlemleri

Müşteri veri ve programları buluta taşındıklarında sağlayıcı denetimindeki konaklarda bulunurlar. Toplamda verimliliği artırmayı isteyen sağlayıcı fiziksel olarak aynı konağın üzerinde birden fazla müşterinin veri ve programının aynı anda tutarak bu hedefine ulaşır. Bu koşullar altında, sağlayıcı denetimindeki bir konakta bulunan müşteri veri ve programlarının korunması gerekir. Koruma çok yönlü olmalıdır. Müşteri programları, müşterinin kendi kullanıcılarından, diğer müşterilerin programlarından ve onların kullanıcılarından, son olarak da sağlayıcının etkilerinden korunmalıdır. Bunun yanında sağlayıcının konaklarının da müşteri programlarının olası kötücül etkilerinden korunması gerekir.

5.1.1 Bir programı bulutta yalıtmanın gerekleri

Programlar yerelde çalıştırılırken işletim sistemleri süreç kavramını kullanır. Bunun yararı, programların (I) farklı bağlamlarda (II) farklı verilerle (III) farklı ayarlarla (IV) farklı kullanıcılar için çalıştırılabilmesidir. Bağlamların ayrılmasını, bir başka söyleyişle bir programın kullandığı belleğin diğer programların etkilerinden yalıtılmasını

işletim sistemi üstlenir. Programların farklı verilerle ve farklı ayarlarla başlatılması ise kullanıcının ya da önceden kurulmuş görevlerin denetiminde işletim sisteminin sorumluluğudur. Bir programın farklı kullanıcıların denetiminde çalıştırılması arasındaki farkları gözetecek olan da yine işletim sistemidir. Yerelde işletim sistemine yüklenmiş olan bu sorumlulukların programın bulutta çalıştırılması gerektiğinde nasıl karşılanacağı belirsizdir. Sayılan dört sorumluluğun bulutta nasıl ele alınacağı izleyen bölümde tartışılmıştır.

5.1.1.1 Bulutta süreç bağlamlarının yalıtımı

Bir programın bağlamını belirlemek görevi, her durumda programın o anda üzerinde çalıştığı işletim sisteminin denetimindedir. Bunun nedeni, belleğin fiziksel bir kaynak olarak doğrudan doğruya işletim sistemi denetiminde olmasıdır. Örneğin, sanal adresleme yapılıyorsa ilgili tabloların denetimi ister istemez işletim sisteminde bulunmalıdır. Bu nedenlerle bu sorumluluğun işletim sisteminden alınması söz konusu edilemez. Bulut ortamında yapılması gereken, işletim sisteminin kaynak kullanımını denetim altına almasına olanak verecek bir yordam geliştirip bunu sağlayıcının konakları üzerinde gerektiğince kullanabilmektir. Bunu yapmak üzere konak işletim sisteminde her programı çalıştırmaya elverişli süreç sanal makinaları hazırlanır. Bu sanal makinalar uygun ayarlarla başlatılan işletim sistemi süreçleridir. Böylece müşterinin uygun programlama dilinde yazdığı programlar kullanılan dilin standartları ve belirlenen kaynak kullanım koşulları altında işletim sistemi süreçlerine çevrilir. Bu yapılar güvenli hesaplama katmanı adı altında oluşturulur. Tasarlanan çözüm, uygulamada yaygın bir kullanım alanı olduğu için Java programlama dilini kullanmış ve güvenli hesaplama katmanını Java süreç sanal makinası kullanarak oluşturmuştur. Ancak başka dillerin kullanılması da bu bölümde yazılan ilkelerin kullanımına engel olmaz.

Bu yaklaşımın bölümün konusunu oluşturan önlemlerin dışında bir başka zayıflığı da karşıladığı sezilir. Bulutta müşteri programlarının doğal olarak farklı olabilecekleri ve zaman zaman değişik konaklara taşınabileceği göz önüne alınırsa hem müşteri programlarının hem de bunları çalıştıracak işletim sistemlerinin standartlaştırılması gerektiği ortaya çıkar. Uygulamada, pazardaki tüm sağlayıcıların her biri farklı fiziksel özelliklerdeki konaklarında aynı işletim sistemini bulundurmaları beklenemez. Müşterilerin de farklı işletim sistemleri için ayrı ayrı programlar oluşturmaları veya tek bir

sağlayıcıya bağlanıp kalmaları istenemez. Bu durumda karşılaşılan *standart arayüzlerin eksikliği (Z_5)* zayıflığıdır ve yaratacağı tehditlere karşılık standart arayüzler sunan bir katman olarak Java süreç sanal makinası önlemi getirilmiştir.

5.1.1.2 Bulutta sürecin farklı verilerle çalışması

Bulutta bir programın farklı verilerle başlatılması için verilere erişiminin olması gerekir. Program çalışması sırasında bir ağ bağlantısıyla ilgili veriye erişmek üzere tasarlanmışsa, örneğin bir veritabanına erişecekse, bu programın bir ayarı olarak değerlendirilmelidir. Bu konu 5.1.1.4 bölümünde tartışılacaktır. Ancak, program verileri giriş değerleri olarak alacak ve işlemeye başlayacaksa bu verilerin de programla birlikte bulunması yeğlenir. Bu durumda verilerle programın konum olarak birbirine yakın olması hem veri ve programın taşınmasının getireceği fazladan ağ yükünü hem de ayrı konumlardaki veri ve programları eşleştirmek üzere harcanacak yönetim çabasını ortadan kaldırır. Veri ve programı bir araya getirerek her ikisine bulutta tek bir kavramsal nesne olarak davranmak kullanışlılığı artırır. Veri ve programı bir arada içereceği düşünülerek tasarlanan bu yapı *süreç kozası* olarak adlandırılmıştır. Böyle bir kozanın oluşturulması durumunda programa giriş olarak verilecek farklı veri kümeleri, bunların kullanıcılarının erişim hakları ve çıkışın nasıl oluşacağıyla ilgili ayarların da oluşturulması gerekecektir. Bu konular 5.2 ve 5.3 bölümlerinde anlatılmıştır.

5.1.1.3 Bulutta süreçlerin kullanıcılarla ilişkilendirilmesi

Bulutta kullanıcı yönetimi yerel işletim sistemlerinde olduğundan daha zordur. Buna yol açan dört belirgin neden vardır. (I) Yerel işletim sistemlerinde kullanıcı sayısı sınırlıdır. Bulutta, sınır belirgin değildir. (II) Her kullanıcı kendi programlarını kendi oturumları içinde yürütürler. Sağlayıcıya ilişkin bir konak müşteri ya da onun kullanıcı adına bir programı yürütürken onlar adına açılmış bir oturum yerine tüm müşteriler ve kullanıcılar için oluşturulmuş ortak bir oturum kullanır. (III) Yerel işletim sistemlerine kullanıcıları önceden tanıtılır. Bulutta hizmet verecek konaklar her ne kadar her kullanıcının yerini alabilen ortak bir oturum kullansalar da, gerçekte hizmet verilen daha önce hiç karşılaşılmamış bir kullanıcı olabilir. (IV) Bunun sonucu olarak, konağın karşılaştığı program yapısını önceden kestirmek üzere denetim yapamayacağı yeni bir program, veriler önceden denetleyemeyeceği o anda karşılaştığı veriler, ayarlar izin

verip veremeyeceğinin o anda sorgulanması gereken ayarlardır. Kullanıcı programının konaklar üzerinde göçebileceği ya da birkaç konakta eşzamanlı çalışabileceği de göz önüne alınırsa, hem konağın güvenli program yürütmesine hem de kullanıcının özlük bilgilerinin korunmasına ve programın tutarlı çalışmasına olanak verecek bir yapının tasarımı gereklidir. Müşterilerin ve kullanıcılarının alacağı hizmetin sınırları sözleşme koşulları ile belirlenmelidir ve bu koşullar altında müşteri programları yürütülmelidir.

5.1.1.4 Bulutta süreçlerin etkisinde olduğu ayarlar

Buluttaki programın çalışması sırasında etkisinde kalacağı ayarların üç kaynağı olabilir. Bunlardan biri müşteri ile sağlayıcı arasında yapılan hizmet sözleşmesidir. Sözleşmede müşteri programın çalışması için gereken koşulları belirtir, sağlayıcı da müşterinin olağandışı bellek tüketmesi gibi altyapısına zarar vereceğini düşündüğü davranışları kısıtlar. Olasılıkla bunlara dayanılarak bir ücret belirlenir ya da önceden hazırlanmış standart sözleşmelerden biri kullanılır. Programın etkisinde kalacağı ikinci ayar, programın çalışması için gerekli olan ayarlardır. Bir veritabanı erişimi, bir sunucu erişimi, bir dosya erişimi gerekli ise bu ayarlar programı hazırlayan kişi tarafından belirlenmelidir. Böylece sözleşmeye uygunsa bu erişimlere izin verilir. Üçüncü ayar kaynağı programı kullanandır. Program, kullanıcısının isteklerine ve o anki gereksinimlerine göre farklı ayarlarla başlatılabilir. Bu düzeydeki ayarlar önceden konağa bildirilemez, ancak yürütme komutuyla birlikte belirtilir.

Bir müşterinin programı bulutta çalıştırıldığında bunu tetikleyecek olan çoğunlukla müşterinin kendisi ya da müşterinin yetki vermiş olduğu bir kullanıcıdır. Programı çalıştırmak üzere uzaktan erişim yapılarak konakla bağlantı kurulur ve konağa programı ne biçimde çalıştırması gerektiği söylenir. Konakta bağlantıları kuran ve pek çok güvenlik doğrulamasını yürüterek kuralların uygunluğunu denetleyen bir yazılım bileşeni bulunur. Bu bileşen *kural uygunluk noktası* olarak adlandırılır. Denetlenen kurallarda, yani erişim ve kaynak kullanım ayarlarında herhangi bir uygunsuzluk olmaması durumunda kural uygunluk noktası işletim sisteminde bağlamı ayırmak üzere kendisi de işletim sisteminin bir programı olan güvenli hesaplama katmanından bir örneği canlandırarak belirlenen ayarlarla çalıştırır. Bu ayarlar güvenli hesaplama katmanının davranışını hizmet sözleşmesine, müşteri programının gereklerine ve müşteri programını o sırada kullanan kişiye göre sınırlandırır. Çalışmaya başlayan güvenli

hesaplama katmanı ayarlar yoluyla kendisine bildirilen müşteri programını çalıştırarak program mantığının akışını –ayarlara uygunluğunu denetlemek koşuluyla– ona bırakır.

Müşteri programlarının ne şekilde çalıştırılacağını ya da çalışma anındaki kısıtları belirten ayarların programın her çalıştırılışında yürütmeyi gerçekleştiren bilgisayara aktarılması gerekmez. Bunun yapılması öncelikle yönetimsel açıdan zordur. Ayrıca, bulut tanımıyla hesaplamada konumdan bağımsızlık sağlandığı ve gerektiğinde programların farklı konumlardaki bilgisayarlara göçebildiği düşünülürse, programın her yer değiştirmesinden sonra bu ayarların yeni konağa bildirilmesi gerekir. Bu sorunlardan sıyrılmak için *iliştirilmiş kullanım ilkeleri*nden[1] [93] esin alınarak ayarların programa iliştirilmesi düşünülmüştür.

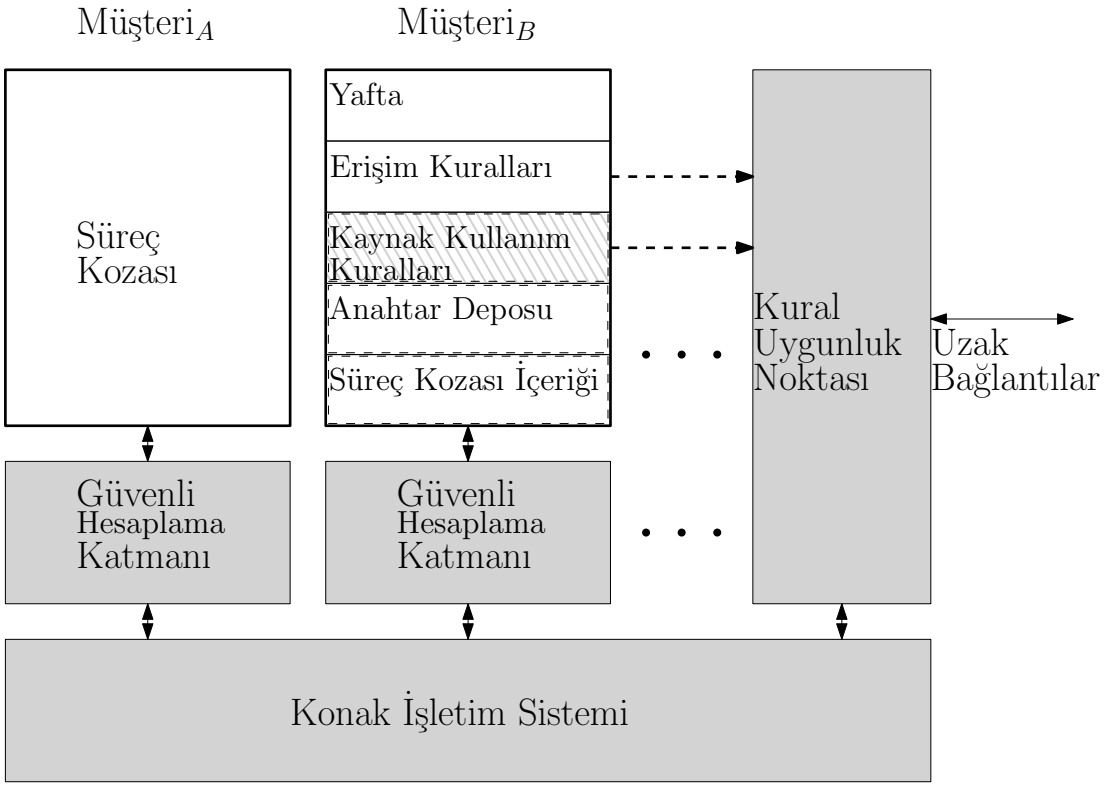
5.1.2 Müşteri veri ve programlarının konaklardaki yerleşimi

Konaklarda yer alacak bileşenler müşteri veri ve programlarını içeren süreç kozalarıyla birlikte Şekil 5.1'de gösterilmiştir. Şekilde gri renkteki kutular sağlayıcı denetimindeki bileşenleri, beyaz renkteki kutular müşteri denetimindeki bileşenleri göstermektedir. Süreç kozasının üçüncü alanı sağlayıcı tarafından imzalandığı için taralıdır. Süreç kozasının son üç alanının şifrelenmiş olduğu bu alanlar kesik çizgili ikinci bir zarf içine alınarak betimlenmiştir. Sağlayıcı denetimindeki konaklarda bulunması öngörülen bileşenlerin tanıtımı izleyen bölümlerde yapılmıştır.

5.1.2.1 Güvenli hesaplama katmanı

Güvenli hesaplama katmanı konak işletim sistemi üzerinde yer alan ve müşteri programlarını yürütmeye yarayan, bir Java süreç sanal makinası ve bu süreç sanal makinayı yönetmeye yönelik araçlardan oluşan bir yazılım bileşenidir. Güvenlik açısından bakıldığında dikkat çeken özelliği, müşteri programlarının işletim sistemi kaynaklarını kullanmasına yarayan ortamı oluşturmasıdır. Her bir kaynak güvenli hesaplama katmanı içinde bir yazılım sınıfı olarak soyutlanır ve müşteri programlarının işletim sistemi kaynaklarına sadece bu sınıflar yoluyla erişmesine izin verilir.

[1] İngilizcesi: "sticky policies". Özlük bilgilerini korumak için verileri imlemek üzere tasarlanmıştır.



Şekil 5.1 : Sağlayıcının bir konağında iki süreç kozasının yerleşimi.

Güvenli hesaplama katmanı kullanmanın iki yararı olur. Birincisi, müşteri programları için standart arayüz oluşturmasıdır. Daha önemli olan ikincisi, müşteri programlarının işletim sistemi kaynaklarını kullanımlarını yönetmesidir. Kaynak kullanımı şöyle yönetilir: Programların kaynak kullanım kuralları, bulundukları süreç kozasına iliştirilmiş durumdadır. Programın her yürütülüşünden önce iliştirilmiş kaynak kullanım kuralları, kural uygunluk noktası tarafından okunur ve güvenli hesaplama katmanı bu kurallara göre koşullanır. Böylece, kaynak kullanımı için kısıtlar belirlenir ve bunlar daha program çalışmaya başlamamışken uygulamaya sokulur. Program yürütüldüğü sürece bu kısıtlar etkin kalır.

5.1.2.2 Kural uygunluk noktası

Kural uygunluk noktası süreç kozalarına uzaktan erişimi sağlayan haberleşme noktasıdır. Ayrıca, erişim kurallarının ve kaynak kullanım kurallarının sağlayıcı denetiminde uygulanmasında görev alır.

Bu görevlerden ilki yürütülürken süreç kozaları üzerindeki erişim kuralları okunup bunlara uygun olarak uzaktan erişime izin verilir. Bir kullanıcıdan gelen erişim isteği

üzerine kural uygunluk noktası ilgili süreç kozası konakta bulunuyorsa üzerine iliştirilmiş erişim kurallarını okuyarak kullanıcının erişim hakkı olup olmadığına bakar. Erişim hakkı varsa, bunun kaydı 6. bölümde anlatılan düzenekle günlüğe işlenerek kullanıcının süreç kozasına erişimine izin verilir. Bir kullanıcının süreç kozasına erişmesine izin verilse dahi, koza içindeki veri ve programlar büyük olasılıkla şifrelenmiştir. Bu konu 5.2 bölümünde süreç kozasının yapısı anlatılırken ve 5.3 bölümünde şifreleme ile yetkilendirmenin nasıl uygulanacağı açıklanırken ayrıntısıyla ele alınmıştır.

Kural uygunluk noktasının ikinci görevi, bir kullanıcının bir programı çalıştırmak üzere erişimine izin verilmesiyle ortaya çıkar. Bu durumda kural uygunluk noktası süreç kozası üzerine iliştirilmiş ilgili kullanıcıya yönelik kaynak kullanım kurallarını okur. Buna dayanarak güvenli hesaplama katmanını koşullar. Kullanıcıdan gelen yürütme komutunu ayarlarıyla birlikte işletmek üzere güvenli hesaplama katmanına ileterek programı yürütülmeye hazırlar.

Süreç kozalarına iliştirilmiş erişim kurallarının müşterinin oluşturduğu biçimde değişmeden kaldığını ve kaynak kullanım kuralları konusunda hem müşteri hem sağlayıcı tarafından sakınca bulunmadığını doğrulamak üzere kriptolojik önlemler alınmıştır, bu konu 5.2.2 bölümünde açıklanmıştır.

5.2 Süreç Kozasının Yapısı

Süreç kozası müşteri tarafından hazırlanır ve kaynak kullanım kuralları sağlayıcı tarafından onaylandıktan sonra sağlayıcının konaklarından biri üzerinde tutulur. Süreç kozası sadece edilgen kriptolojik koruma yöntemlerinin uygulandığı bir yapıdır. Etkin yazılımsal düzeneklerin bulunmayışının nedeni, süreç kozasını dışarıdan gelecek olası etkilerden korumaktır.

Süreç kozası beş alandan oluşur. İlk iki alan düz metin içerir ancak bu alanlar değişikliklerden korunması amacıyla imzalanmıştır. Üçüncü alan kaynak kullanım kurallarını içerir ve sağlayıcının kullanımına yöneliktir. Bu alan sağlayıcının açık anahtarı ile şifrelenmiş, ayrıca hem müşteri hem de sağlayıcı tarafından imzalanmıştır. Dördüncü alan, ikinci alanın içerdiği erişim kurallarının kriptolojik yollarla uygulanması için gereken ara anahtarların bulunduğu, ayrıldığı alt kesimlerin her kullanıcı ya da kul-

lanıcı kümesi için farklı şifrelendiği bir alandır. Beşinci alanda ara anahtarlarla şifrelenmiş biçimde süreç kozasının içeriği olan veri ve programlar bulunur. Bu alanın içindeki kesimlerde yetkilendirme ayarları farklı olabilir.

5.2.1 Somut bir örnek

Süreç kozası anlatılırken somut bir örnekle devam etmenin konunun kavranmasına yararı olacağı düşüncesiyle aşağıdaki senaryo süreç kozasının alanları tanıtılırken kullanılmıştır.

Örnek olarak bir üniversitenin harfli notların verilmesinde her zaman aynı ölçütleri uygulayabilmek üzere bir yazılım geliştirmeye yönelmiş olduğu varsayılmıştır. Ölçütlerin uygulandığından kuşku duyulmaması için üniversite tarafından genel bir not hesaplama programı yazılmış ve okutmanların bunu kullanması zorunlu tutulmuştur. Program giriş olarak okutmanlardan sınav sonuçlarını, ders yardımcılarından da ödev sonuçlarını alır. Çıkış olarak dönem notları üretir. Öğrenciler her aşamada notlarını okuyabilir.

Bu örnekte, süreç kozasının içeriğinde üç veri kesimi, iki program kesimi olması gerekecektir. Okutmanın sınav notlarına *oku/yaz* erişimi, ders yardımcısının ödev notlarına *oku/yaz* erişimi, öğrencilerin tüm notlara *oku* erişimi olması gerekir. Okutmanın sınav ve ödev notları girildikten sonra programı çalıştırma hakkı olması gerekir. Ayrıca programın da dönem notlarını oluşturacak erişim hakları bulunmalıdır. Sağlayıcıda bulunan verilere erişimi sağlamak üzere konakta çalışıp kullanıcılarla haberleşecek bir programın bulunması kullanışlı olur. Bu programa tüm kullanıcıların erişimi olmalıdır. Bu programın veri erişimi işlevinin dışında kimi kullanıcı arayüzü işlevlerini yerine getirdiği, öğrencilerin tüm sınıfın notlara erişmesini engelleyerek sadece kişisel notlarını görmelerine izin verdiği varsayılabilir.

Sunulan tasarımda süreç kozasının içeriği için bir sınırlama yoktur. Pek çok program ve veri kesimi bir arada, farklı erişim hakları ile bulunabilir. Bu yolla üniversitenin tüm sınıflarını kapsayacak biçimde süreç kozası genişletilebilir. Ancak yönetimin daha kolay olacağı düşünülürse her ders için ya da her sınıf için ayrı kozalar tanımlanabilir.

Bu karar yönetimseldir ve kozanın sahibine bırakılmıştır. Örnekte sadece bir dersin bir sınıfı söz konusu edilmiştir.

5.2.2 Süreç kozasının alanları

Süreç kozasının birinci alanı kozayı bulutta tanımlamak üzere kullanılan bir yaftadır ve kozanın sahibi olan müşteri tarafından imzalanmış özgün bir karakter dizisi içerir. Birinci alan, örneğe uyarlanmış biçimde, Çizelge 5.1'de gösterilmiştir. Örnekte yaftalama yapılırken alışıldık Java adlandırma kurallarına uyulmuştur. İmzalama işlemi σ ile gösterilmektedir. "İTÜ", süreç kozasının sahibi olan müşteridir.

Çizelge 5.1 : Süreç kozasının birinci alanı kozayı yaftalar.

$$\left\{ \text{ tr.edu.itu.bulutprogramlari.notlandirma.1\_7\_73 } \right\} \sigma_{\text{İTÜ}}$$

İkinci alan erişim kurallarını, bir başka deyişle uzaktan erişim yapan hangi kullanıcıların hangi içeriği okuyabileceğini, değiştirebileceğini veya çalıştırabileceğini gösteren bir erişim hakları tablosunu içerir. Bu tabloda gösterilen kullanıcıların doğrudan doğruya kimlikleri bildirilmeyebilir; bunun yerine 4.3.2.1 bölümünde anlatılan vekil sertifikalar kullanılabilir ya da 5.3.3 bölümünde anlatıldığı biçimde sadece özelliklere dayanılarak erişim hakkı verilmiş olabilir. Bu alan müşteri tarafından imzalanır. Bu alanı gösteren bir örnek Çizelge 5.2'de verilmiştir. Örnekte süreç kozasının içeriğinin farklı erişim kurallarıyla ayrıldığı beş kesim için erişim hakları tanımlanmıştır: kesim_0 herkes için gerekli olan veri erişimi yapmak üzere "eriş.jar" adlı bir programı barındırır. Sadece Okutman'ın çalıştırmaya hakkı olan "notlandır.jar" adlı dönem sonu notlarını hesaplamaya yarayan program kesim_1'de yer alır. kesim_2'de öğrencilerin sınav notlarını içeren "sınav.txt" dosyası, kesim_3'te ödev notlarını içeren "ödev.txt" dosyası ve kesim_4'te dönem sonu notlarının yazılacağı "dönem.txt" dosyası bulunur.

Üçüncü alan süreç kozasının konaktaki kaynak kullanım kurallarını düzenler. İkinci alanda tanımlanmış olan erişim haklarının bulut kaynakları üzerindeki izdüşümleri belirlenerek her bir kullanıcı için ayrı ayrı buraya yazılmalıdır. Böylece olası tüm erişimler için sağlayıcının altyapısını kullanmak üzere kullanıcı temelinde izin istenmiş olacaktır. Kitaplıklara erişim izni gibi her kullanıcı için geçerli izinler buna eklenebilir. Ayrıca, erişim kuralları içinde görünmeyen ancak koza içindeki programların çalışması

Çizelge 5.2 : Süreç kozasının ikinci alanı erişim kurallarını içerir.

	$kesim_0$ eriş.jar	$kesim_1$ notlandır.jar	$kesim_2$ sınav.txt	$kesim_3$ ödev.txt	$kesim_4$ dönem.txt
Okutman	Ç	Ç	O/Y	O	O
Yardımcı	Ç	-	-	O/Y	-
[Öğrenciler]	Ç	-	O	O	O

$\sigma_{\text{İTÜ}}$

için gereken başka izinler olabilir. Örneğin, bir veritabanına erişim için gereken ağ erişim izinleri belirtilmelidir. Konunun bütünlüğünü bozmamak için, söz konusu veritabanının bir başka süreç tarafından bir başka hizmet olarak sunulduğu varsayılabilir. Bunun dışında, hizmet sözleşmesinde belirlenen koşullar uyarınca programların çalışmasına sınırlar getirecek kurallar da alanın içeriğine eklenir. Dikkat edilmesi gereken, bu kuralların açığa çıkmasının hizmet sözleşmesinin de bir bölümünün açığa çıkması anlamına geldiğidir. Bu nedenle hizmet sözleşmesinin taşıdığı bilginin gizli tutulması için üçüncü alan şifrelenir. Kuralları okuyacak olan taraf sadece konak olduğu için şifreleme sağlayıcının açık anahtarı ile yapılır. Şifrelemenin ardından, alanın bütünlüğünün doğrulanabilmesi için müşteri alanı imzalar. Böylece üçüncü alan müşteri tarafından oluşturulmuş olur. Ancak oluşan içerik iki tarafın da onayını gerektirmektedir. Süreç kozasının sağlayıcıya taşınmasıyla konakta alanın bütünlüğü denetlenir ve içeriği okunur. Belirtilmiş olan kaynak kullanım kuralları değerlendirilir. Değerlendirme olumlu ise sağlayıcının da imzalaması ile bir uzlaşma olduğu kriptolojik yollarla onaylanır. Böylece süreç kozasının buluta yerleşmesinden başlanarak altyapıyı kullanmakta bir sakınca olmadığı belirlenmiş olur.

Çizelge 5.3'te üçüncü alanı gösteren bir örnek verilmiştir. Şifreleme ε ile gösterilmiştir. Örnekte Yardımcı'nın ve öğrencilerin veri erişim hakları kaynak kullanım kurallarına kolaylıkla dönüştürülmüştür. Ancak Okutman'ın "notlandır.jar" programı ile dönem notlarını oluşturabilmesi için çıktı dosyasına yazma hakkı da edinmesi gerekmektedir. Notlandırma programı bulutta çalışırken kullanıcısıyla haberleşmek için 80. iskeleyi kullanacaktır ve ancak üniversitenin "160.75" ile başlayan ağ adreslerinden gelen bağlantılarla iletişim kuracaktır. Programın bulutta çalışırken tüketebileceği işlemci zamanı ve bu sırada tutabileceği bellek miktarı sağlayıcı ile üniversite arasındaki hizmet sözleşmesi ile sınırlandırılmıştır. Benzer biçimde veri erişimi için kul-

lanılan “eriş.jar” programı da 80. iskeleyi kullanarak herhangi bir ağ adresindeki kullanıcılarla haberleşecektir.

Çizelge 5.3 : Süreç kozasının üçüncü alanı kaynak kullanım kurallarını içerir.

Okutman :	~/eriş.jar	çalıştır		$\varepsilon_{\text{Sağlayıcı}}$
	~/eriş.jar	bağlan,	80	$\sigma_{\text{İTÜ}}$
	~/eriş.jar	dinle,	*	$\sigma_{\text{Sağlayıcı}}$
	~/eriş.jar	enÇokBellek,	64M	
	~/eriş.jar	enUzunSüre,	10’	
	~/notlandır.jar	çalıştır		
	~/notlandır.jar	bağlan,	80	
	~/notlandır.jar	dinle,	160.75.*	
	~/notlandır.jar	enÇokBellek,	512M	
	~/notlandır.jar	enUzunSüre,	3’	
	~/veri/sınav.txt	oku/yaz		
	~/veri/ödev.txt	oku		
	~/veri/dönem.txt	oku/yaz		
Yardımcı :	~/eriş.jar	çalıştır		
	~/eriş.jar	bağlan,	80	
	~/eriş.jar	dinle,	*	
	~/eriş.jar	enÇokBellek,	64M	
	~/eriş.jar	enUzunSüre,	10’	
	~/veri/ödev.txt	oku/yaz		
[Öğrenciler] :	~/eriş.jar	çalıştır		
	~/eriş.jar	bağlan,	80	
	~/eriş.jar	dinle,	*	
	~/eriş.jar	enÇokBellek,	64M	
	~/eriş.jar	enUzunSüre,	10’	
	~/veri/sınav.txt	oku		
	~/veri/ödev.txt	oku		
	~/veri/dönem.txt	oku		

σ ile imzalama, ε ile şifreleme gösterilmiştir.

Programların konaklarda süreçler olarak yürütüleceği göz önüne alınırsa her bir sürecin bir giriş, bir çıkış, bir de hata akımı olacaktır. Bu akımlar süreçlerin başlatılmasından sonra sözü geçen iskeleler aracılığıyla kullanıcıya bağlanır. Böylece programlar çalıştıkları süre boyunca doğrudan kullanıcının denetiminde kalır.

Dördüncü alan beşinci alana yerleştirilecek içeriği şifrelemekte kullanılacak ara anahtarların saklanması için kullanılan bir anahtar deposudur. Farklı kesimlere farklı biçimlerde erişim için farklı ara anahtarlar bulunur. Bu ara anahtarlar farklı kullanıcılar için ayrı ayrı şifrelenerek bu alanda depolanır. Bu alanın yapısı ve ayrıntısı 5.3.3 bölümünde verilmiştir. Alanın içeriğini örneklendiren Çizelge 5.4 de yine 5.3.3 bölümünde sunulmuştur.

Beşinci alan bir disk alanını andıracak biçimde kesimlere ayrılmıştır. Her kesime erişim ayrıca belirlenir. Bu kesimlerdeki erişimin ikinci alanda belirtilen erişim kuralları ile uyumlu olması beklenir. Bunu uygulamak öncelikle konağın ödevidir. Yine de, konağın kötücül olabileceği, saldırgan tarafından ele geçirilebileceği veya atlatılabileceği düşünülerek müşteri tarafından da erişim kurallarının uygulanmasına yönelik önlemler alınır. Bu biçimde korunan kesimlerdeki veri ya da programlar gizli tutulur, herhangi bir bozma girişimi ise sezilir. Bu önlemlerin tasarımı ayrı bir bölüm olarak 5.3'te sunulmuştur. Bu alanın içeriğini gösteren Çizelge 5.5 açıklamasıyla birlikte 5.3.3 bölümü içinde verilmiştir.

5.3 Kriptolojik Düzeneklerin Tasarımı

Müşterinin belirlediği erişim kurallarını kriptolojik yollarla uygulanmasını sağlayacak bir düzeneğin tasarlanması denetimin sadece sağlayıcıya bırakılmasından daha güven verici olur. Bu düzeneğin tasarımı, ilerleyen bölümlerde erişim kurallarından başlanarak, basitten karmaşığa doğru adım adım açıklanacaktır.

5.3.1 Erişim türleri

Süreç kozasının beşinci alanı içinde iki farklı türde içerik bulunabilir: program ya da veri. Buna bakılarak üç farklı erişim türü belirlenir. İçerik programsa erişen kullanıcının *çalıştırma* hakkı vardır ya da yoktur. İçerik veriyse erişen kullanıcının ya *okuma* hakkı vardır ya *hem okuma hem yazma* hakkı vardır ya da hiç erişim hakkı yoktur.

Önerilen düzenekte bir programa yazma hakkı verilmemiştir. Verilmesi, programda bir değişiklik olacağı, yani programın bir başka programa dönüşmesi demektir. Bu

yaklaşım güncelleme ya da yeni sürümleri yama olarak yayma için uygundur. Ancak bu bilgiişlem ortamı sunan bulutun işlevi dışında sayılmıştır ve güvenlik düşüncesiyle bu yetki sadece müşterinin kendisine bırakılmıştır.

Aynı biçimde, programların ve verilerin mantıksal olarak nasıl sınıflanacağını ve kaç koza içinde bulutta nasıl yer alacağını müşteriye bırakmak gerekir. Tasarım müşteri programlarını ayrı ayrı kozalarla buluta yerleştirmeye olanak tanıdığı gibi, aynı müşterinin buluttaki tüm programlarını ve verilerini tek bir koza içindeki farklı kesimlerde bulundurmasına da olanak verir. Kullanışlılığın en uygun olduğu durumu seçmek programın yapısına ve kullanım alanına göre müşterinin elindedir.

5.3.2 Erişim haklarını şifreleme ve imzalama ile ilişkilendirmek

Veri ya da program oluşundan bağımsız olarak bir bit dizisinin okunmasını kriptolojik olarak engellemenin yolu şifrelemedir. Ancak, bir bit dizisinin değiştirilmesini engellemenin bilinen bir yolu yoktur. Sadece imzaların kullanımı ile değişiklikler sezilir. Bu da, özellikle değişikliği yapan kişi belirlenebiliyorsa, saldırganları caydırmanın bir yoludur.

Tezde önerilen süreç kozasında bir önceki 5.3.1 bölümünde sözü edilen erişim hakları şifreleme ya da imzalama ile ilişkilendirilmiştir. Şifreleme *oku* ve *çalıştır* hakları ile kriptolojik olarak uygulanır. Bir bit dizisinin değiştirilemezliği de *oku/yaz* haklarının kriptolojik izdüşümü olan imzalama ile ilişkilendirilir.

İşletim sistemleri açısından bakıldığında *çalıştır* hakkı ile *oku* hakkı arasında ayrıklıklar olsa da kriptolojik olarak her ikisi için de verinin okunması gerekir ve süreç kozasının tasarımında bu ikisinin birleştirilmesinde bir sakınca yoktur.

5.3.3 İçeriğin kullanıcılarla ilişkilendirilmesi

Erişim haklarının hangi kriptolojik yöntemlerle ilişkilendirileceği belirlendikten sonra bu yöntemlere uygun anahtarların oluşturulması tekdüze bir iştir. Her kesim için kesimi gizlemekte kullanılacak bir bakışımlı anahtar ve imzalamakta kullanılacak bir

bakışımsız anahtar çifti gerekir. Oluşturulan bu ara anahtarlarla her kesimde yer alan içerik önce şifrelenir, daha sonra da imzalanır.

Ara anahtarların ilgili kullanıcılara gönderilmesi pek çok nedenden ötürü uygun olmaz. Yönetimi zordur, ilgili kullanıcılar o sırada henüz bilinmiyor olabilir, kullanıcılar anahtarları saklama sorumluluğunu üstlenmek istemeyebilir ya da güvenilir bir haberleşme kanalı bulunamayabilir. Sonuç olarak, ara anahtarların da içerikle birlikte süreç kozası üzerinde bulunması uygulamada kolaylık sağlar. Ara anahtarlar süreç kozasının dördüncü alanında, ilgili kullanıcıların erişebileceği biçimde şifrelenmiş biçimde saklanır. Bu yaklaşım da iliştirilmiş kullanım ilkelerini [93] anımsatır.

Ara anahtarların kimlere dağıtılacağını belirlemek müşterinin sorumluluğundadır. Bu, ikinci alandaki erişim kurallarına bakılarak kolaylıkla yapılır. Bir kesim için *oku* veya *çalıştır* hakkı olan kişiler o kesimi okumak için şifrelemekte kullanılan ara anahtara ve kesimin en son yazılışından sonra değiştirilmediğini onaylamak için o kesimi imzalamakta kullanılan gizli anahtarın eşi olan açık anahtara gerek duyar. Aynı kesimde *oku/yaz* hakkı olan kişiler bu anahtarların yanında o kesimi imzalamakta kullanılan gizli anahtara da gerek duyar. Böylece içerikte yapılan bir değişikliğin geçerli görünmesi için imzayı da güncelleyebilir.

Ara anahtarlar gerçek kişiler için saklanırken, o kişilerin kişisel açık anahtarları ile şifrelenerek dördüncü alanda saklanması yeterlidir. Böylece kişiler süreç kozasının bu alanındaki kendilerine ayrılmış bölümün şifresini gizli anahtarları ile çözerek ara anahtarlara erişir. Ancak, ara anahtarların bazı özellikleri taşıyan kişiler adına şifrelenerek saklanması isteniyorsa, bu farklı kriptolojik yöntemler gerektirir.

Çözüm olarak daha önce 4.3.2.1 bölümünde de sözü edilen vekil sertifikalar kullanılabilir. Bu yolla bir kişinin belli özellikleri gözetilerek aynı veya farklı özlük bilgileri ile yeni bir sertifika edinmesi sağlanabilir. Ancak bu vekil sertifikanın müşterinin kozada kullanacağı özelliklere göre önceden vekil sertifika yetkesinden edinilip hazır edilmesi ve ara anahtarlar bu sertifikaya göre şifreleneceği için kozanın ömrü boyunca değiştirilmemesi gerekir.

Daha esnek bir çözüm, özellik tabanlı şifrelemedir[2]. Bu yöntem son yıllarda ortaya çıkmış, ilk uygulanabilir örneği 2007 yılında oluşturulmuştur [94]. Özellik tabanlı şifreleme ile açık metin önceden belirlenmiş özelliklere dayanarak şifrelenir. Örnek olarak, bir metin sadece "recep" adlı kullanıcı *veya* yönetim düzeyi "başkan" olmayan kişiler tarafından okunabilecek biçimde şifrelenebilir. Yazıda kullanılan örnekte, üniversite, "2015 mezunları", "mühendislikteki kadınlar", "bilgisayar mühendisliği öğrencileri" gibi pek çok farklı özelliğe göre kullanıcıları arasından özelliklerine dayalı seçimler yaparak ara anahtarları uygun biçimde şifreler.

Süreç kozasının sahibi olan müşterinin tüm ara anahtarlara erişmesi gerekir. Böylece daha sonra erişim kurallarında değişiklik yapılsa da anahtar dağıtımını yönetir.

Sağlayıcı, kullanıcıların erişimi sırasında ve erişimlerinin ardından içeriğin bütünlüğünü denetlemek üzere tüm içerikler için oluşturulan imzalama açık anahtarlarına erişmelidir. İçerik şifrelenmiş olduğu için, bu, sağlayıcıya içerik hakkında hiçbir bilgi vermez. Ancak içeriğin sonradan değiştirilip değiştirilmediğini anlamasına yarar. Bu yolla güven duyulan bir sağlayıcının içeriği bozmaya çalışan bir kullanıcıyı fark ettiği sırada engelleyeceği varsayılır. Bu engelleme yapılamasa bile 6. bölümde sözü edilen güvenli günlük tutma düzeneği kullanımda ise içeriğe son erişen kullanıcı belirlenecektir.

Yapılan açıklamalar ışığında anahtar deposu olarak tasarlanan dördüncü alan Çizelge 5.4'te görüldüğü gibi oluşturulur. Ara anahtarların indisleri beşinci alandaki kesimlerin indisleriyle örtüşür. Müşteri tüm ara anahtarların bir kopyasını kendisi için saklamıştır; gerektiğinde erişim kurallarını yeniden düzenler. Okutman tüm imzalama açık anahtarlarına ve gizleme anahtarlarına erişebilir. Böylece tüm kesimleri okuyabilir ve bütünlüklerini denetleyebilir. Ayrıca kesim_2'nin ve kesim_4'ün imzalama gizli anahtarlarına da erişebilir. Böylece bu kesimlere yazma hakkına da kavuşur. kesim_4'e yazma hakkı, erişim kuralları gereği değil, kullanacağı programın gerektirmesi nedeniyledir. Yardımcı, kesim_3 için tüm anahtarlara, kesim_0 için gizleme anahtarına ve imzalama açık anahtarına erişebilir. Üniversitenin "öğrencilik" özelliğini taşıyan kullanıcıları tüm verileri okuyabilir ve bütünlüklerini denetleyebilir. Bunu yapmak üzere imzalama açık anahtarlarına ve gizleme anahtarlarına erişebilir. Bunu sağlayan özellik

[2]İngilizcesi: "attribute based encryption", kısaltılarak "ABE" biçiminde de kullanılır.

Çizelge 5.4 : Süreç kozasının dördüncü alanı ara anahtarları içerir.

İTÜ: $\left\{\begin{array}{lcl} \text{kesim}_0\ (\text{eriş.jar}) & : & \text{GizlemeAnahtarı}_0 \\ & & \text{İmzaAçıkAnahtarı}_0 \\ & & \text{İmzaGizliAnahtarı}_0 \\ \text{kesim}_1\ (\text{notlandır.jar}) & : & \text{GizlemeAnahtarı}_1 \\ & & \text{İmzaAçıkAnahtarı}_1 \\ & & \text{İmzaGizliAnahtarı}_1 \\ \text{kesim}_2\ (\text{sınav.txt}) & : & \text{GizlemeAnahtarı}_2 \\ & & \text{İmzaAçıkAnahtarı}_2 \\ & & \text{İmzaGizliAnahtarı}_2 \\ \text{kesim}_3\ (\text{ödev.txt}) & : & \text{GizlemeAnahtarı}_3 \\ & & \text{İmzaAçıkAnahtarı}_3 \\ & & \text{İmzaGizliAnahtarı}_3 \\ \text{kesim}_4\ (\text{dönem.txt}) & : & \text{GizlemeAnahtarı}_4 \\ & & \text{İmzaAçıkAnahtarı}_4 \\ & & \text{İmzaGizliAnahtarı}_4 \end{array}\right\} \varepsilon_{\text{İTÜ}}$

Sağlayıcı: $\left\{\begin{array}{lcl} \text{kesim}_0\ (\text{eriş.jar}) & : & \text{İmzaAçıkAnahtarı}_0 \\ \text{kesim}_1\ (\text{notlandır.jar}) & : & \text{İmzaAçıkAnahtarı}_1 \\ \text{kesim}_2\ (\text{sınav.txt}) & : & \text{İmzaAçıkAnahtarı}_2 \\ \text{kesim}_3\ (\text{ödev.txt}) & : & \text{İmzaAçıkAnahtarı}_3 \\ \text{kesim}_4\ (\text{dönem.txt}) & : & \text{İmzaAçıkAnahtarı}_4 \end{array}\right\} \varepsilon_{\text{Sağlayıcı}}$

Okutman: $\left\{\begin{array}{lcl} \text{kesim}_0 & : & \text{GizlemeAnahtarı}_0 \\ (\text{eriş.jar}) & & \text{İmzaAçıkAnahtarı}_0 \\ \text{kesim}_1 & : & \text{GizlemeAnahtarı}_1 \\ (\text{notlandır.jar}) & & \text{İmzaAçıkAnahtarı}_1 \\ \text{kesim}_2 & : & \text{GizlemeAnahtarı}_2 \\ (\text{sınav.txt}) & & \text{İmzaAçıkAnahtarı}_2 \\ & & \text{İmzaGizliAnahtarı}_2 \\ \text{kesim}_3 & : & \text{GizlemeAnahtarı}_3 \\ (\text{ödev.txt}) & & \text{İmzaAçıkAnahtarı}_3 \\ \text{kesim}_4 & : & \text{GizlemeAnahtarı}_4 \\ (\text{dönem.txt}) & & \text{İmzaAçıkAnahtarı}_4 \\ & & \text{İmzaGizliAnahtarı}_4 \end{array}\right\} \varepsilon_{\text{Okutman}}$

Yardımcı: $\left\{\begin{array}{lcl} \text{kesim}_0 & : & \text{GizlemeAnahtarı}_0 \\ (\text{eriş.jar}) & & \text{İmzaAçıkAnahtarı}_0 \\ \text{kesim}_3 & : & \text{GizlemeAnahtarı}_3 \\ (\text{ödev.txt}) & & \text{İmzaAçıkAnahtarı}_3 \\ & & \text{İmzaGizliAnahtarı}_3 \end{array}\right\} \varepsilon_{\text{Yardımcı}}$

[Öğrenciler]: $\left\{\begin{array}{lcl} \text{kesim}_0 & : & \text{GizlemeAnahtarı}_0 \\ (\text{eriş.jar}) & & \text{İmzaAçıkAnahtarı}_0 \\ \text{kesim}_2 & : & \text{GizlemeAnahtarı}_2 \\ (\text{sınav.txt}) & & \text{İmzaAçıkAnahtarı}_2 \\ \text{kesim}_3 & : & \text{GizlemeAnahtarı}_3 \\ (\text{ödev.txt}) & & \text{İmzaAçıkAnahtarı}_3 \\ \text{kesim}_4 & : & \text{GizlemeAnahtarı}_4 \\ (\text{dönem.txt}) & & \text{İmzaAçıkAnahtarı}_4 \end{array}\right\} \varepsilon_{\text{[Öğrenciler]}}$

tabanlı şifrelemenin kullanılması olmuştur. Sağlayıcı sadece imzalama açık anahtarlarına erişir ve tüm kesimlerin bütünlüğünü denetler; bu yolla uygunsuz değişiklikleri gözler.

Erişim kuralları anlatılan kriptolojik yöntemlerle uygulandıktan sonra beşinci alanda içeriği saklamak oldukça tekdüzedir. Yetkilendirmenin farklılaştığı her kesim ilgili gizleme ara anahtarıyla şifrelendikten sonra ilgili imzalama gizli anahtarıyla imzalanır. Örneğe uygun biçimde kurulan beşinci alan Çizelge 5.5'te gösterilmiştir.

Çizelge 5.5 : Süreç kozasının beşinci alanındaki şifrelenmiş ve imzalanmış içerik.

$$\left\{ \text{~/eriş.jar} \right\} \varepsilon_{\text{GizlemeAnahtarı}_0}\ \sigma_{\text{İmzalamaGizliAnahtarı}_0}$$

$$\left\{ \text{~/notlandır.jar} \right\} \varepsilon_{\text{GizlemeAnahtarı}_1}\ \sigma_{\text{İmzalamaGizliAnahtarı}_1}$$

$$\left\{ \text{~/veri/sınav.txt} \right\} \varepsilon_{\text{GizlemeAnahtarı}_2}\ \sigma_{\text{İmzalamaGizliAnahtarı}_2}$$

$$\left\{ \text{~/veri/ödev.txt} \right\} \varepsilon_{\text{GizlemeAnahtarı}_3}\ \sigma_{\text{İmzalamaGizliAnahtarı}_3}$$

$$\left\{ \text{~/veri/dönem.txt} \right\} \varepsilon_{\text{GizlemeAnahtarı}_4}\ \sigma_{\text{İmzalamaGizliAnahtarı}_4}$$

5.4 Süreç Kozasının Getireceği Fazladan Yük

Sunulan düzeneğin yer olarak getireceği fazladan yük irdelenirken süreç kozasının olmadığı, veri ve programların sağlayıcı konaklarına doğrudan doğruya yerleştirildiği bir durumla süreç kozasının var olduğu durumu kıyaslamak yerinde olur. Süreç kozasının olmadığı durumda, konakta, sadece veri ve programların kapladığı yer tüketilir. Süreç kozası kullanıldığında kozanın ilk dört alanı içeriğin dışındaki alanlardır ve fazladan kullanılmışlardır. Beşinci alanda bulunan içerik ise bakışımlı şifrelenmiştir. Bakışımlı şifrelemede şifreleme bloklarını tamamlamaya yönelik eklemeler dışında veri boyunu büyüten bir ekleme yoktur. Bu nedenle içeriği bulunduran beşinci alanda yer açısından bir artış olacağını söylenemez.

Düzeneğin zaman olarak getireceği fazladan yük incelenirken birkaç adımlı bir yapı ortaya çıkar. İçeriğin şifresinin çözülmesi ya da şifrelenmesi için bakışımlı şifreleme gerekmektedir. Ancak bu şifreleme için gerekli anahtar dördüncü alanda bulunur ve bir bakışımsız anahtarla şifrelenmiştir. Düzeneğin kullanılması için bu şifrenin de

çözülmesi gerekir. Bundan başka, imzaların denetimi ya da oluşturulması için öz alma ve yine ardından bir bakışımsız şifreleme ya da şifre çözme gerekir. Bu durumda, bir okuma ya da çalıştırma işlemi veri üzerinde yapılacak bir bakışımlı şifre çözme, bir öz alma ve kısa veri parçaları üzerindeki iki bakışımsız şifre çözme işlemi zamanı kadar sürer. Benzer biçimde, bir veri yazma işlemi de veri üzerinde yapılacak bir bakışımlı şifreleme, bir öz alma ve kısa veri parçaları üzerindeki iki bakışımsız şifreleme işlemi zamanı kadar sürer. Bunların dışında, içeriğe erişim sonlanırken, denetleme amacıyla, beşinci alandaki ilgili kesimin imzasını konak tarafından denetlenerek erişimi yapanın veriyi bozup bozmadığına bakılır. Bu da veri üzerinde bir öz alma zamanı ve kısa bir veri parçası üzerinde bir bakışımsız şifre çözme zamanı kadar sürer.

5.5 Veri ve Programların Yürütme Sırasında Sağlayıcıdan Gizlenmesi

Süreç kozası ile şifreleme uygun biçimde kullanılarak sağlayıcı denetimindeki konaklarda bulunan müşteri veri ve programları diğer müşterilerden ve kullanıcılardan gizlenebilir. Ancak şifrenin çözülmesi gerektiğinde, bu işlem geçici süre için de olsa yine sağlayıcı denetimindeki konakta yapılmaktadır.

Sağlayıcıya duyulan güven tartışmalı ise bu yaklaşım da tartışmalıdır. Burada dikkat çeken, müşterinin bir düzeye kadar sağlayıcıya güvenmesi, varlıklarını bulut altyapısına taşımasıdır. Sağlayıcı da müşterilere hizmet vermektedir fakat aynı zamanda olabildiğince çok bilgi toplamayı denemektedir. Bu, *dürüst ancak meraklı*[3] saldırgan modeli olarak adlandırılır. Bu modelde sağlayıcı ekonomik getirinin sürmesini istediği için görünürde dürüst davranmaktadır ancak hizmet verdiği süre içinde olabildiğince çok gizli bilgiye erişmeye çalışır.

Müşterinin haklı olarak gizli tutmak istediği kişisel bilgileri ya da ticari sırları olabilir. Bundan başka, müşteri yürütülen programların işleyişini de saklamak isteyebilir. Programların işleyişi yasal olarak saklanması gereken, patente konu edilmiş bir yöntem olabilir.

Müşteriyi bu koşullar altında dahi koruyacak kriptolojik yöntemler vardır. Ancak, bu yöntemler yaygın kullanıma sokulacak denli gelişkin durumda ve yüksek başarımlı

[3] İngilizcesi: “honest-but-curious”.

değildir. Tasarlanan süreç kozasının uygun bir bedelle gerçeklenebilir olması için bu yöntemlere süreç kozasının tasarımında özellikle yer verilmemiştir. Yine de konunun eksik kalmaması için yöntemlerin tanıtımı aşağıda yapılmıştır.

Bazı benzeryapılı[4] aritmetik işlemlerin bazı bakışımsız şifreleme dizgelerine uygulanabildiği uzun zamandır bilinmektedir [95]. Ancak, her dizge şifrelenmiş uzayda tek bir aritmetik benzeryapılı işlemin yapılmasına olanak verir: ya toplama işlemi yapılan dizgeler[5] vardır ya da çarpma işlemi yapılan dizgeler [6]. Bu matematiksel kısıt, kullanışlı programlar oluşturulması olanağını oldukça zayıflatır.

Tam benzeryapılı şifreleme[7] 2009 yılında önerilmiş bir yöntemdir [98]. Bu yöntemle şifrelenmiş uzayda hem benzeryapılı toplama hem benzeryapılı çarpma aynı anda yapılabilir. Böylece teoride bir Turing makinasının çalıştırabileceği tüm programlar, veriler şifrelenmiş uzayı terk etmeden çalıştırılır. Ancak tam benzeryapılı şifrelemeyi uygulamanın önünde hâlâ önemli engeller vardır. Birincisi, hem hesaplama hem de depolama yükü akıl almaz boyuttadır. Kriptoloji alanında süregiden sayısız çalışma verimliliği sürekli artırıyor olsa da, iyileştirilmiş gerçeklemeler dahi olağan algoritmalardan sekiz dokuz mertebe kadar yavaş çalışmaktadır [99]. İkincisi, yüksek bir dilde yazılmış kaynak kodları tam benzeryapılı şifrelenmiş verileri işleyecek programlara dönüştürecek genelgeçer derleyiciler ya da yorumlayıcılar ortada yoktur. Bu yolda yapılmış bir çalışmada [100] iyi bilinen iki algoritmanın dönüşümü gerçekleştirilmiştir. Birbirini izleyen diğer iki çalışmada [101, 102] ise dönüştürülmüş programın bittiğinin anlaşılması sorunu çözülmüş; C [103] dilinde toplama, kaydırma, atama, dallanma, VE ve dışlayıcı VEYA komutlarının çalıştırılması başarılmıştır. Ancak bu komut kümesi ya da bazı algoritmaların hazır olması bir yazılımcının bulutta programlamaya girişmesi için yeterli değildir. Üçüncüsü, veriler bu yolla saklanabilse de işlemci ve bellek sağlayıcının denetiminde olduğu sürece algoritmalar hâlâ *dürüst ancak meraklı* sağlayıcı tarafından izlenebilir.

Sağlayıcının bellek erişimini izleyememesi için uzun zamandır bilinen bir yöntem *ilgisiz bellek*[8] [104] kullanmaktır. Bu yöntemde, basitçe, rasgele okumalar ve yazım-

[4] İngilizcesi: "homomorphic".
[5] Örnek olarak: Paillier [96].
[6] Örnek olarak: ElGamal [97].
[7] İngilizcesi: "fully homomorphic encryption", kısaltılarak "FHE" biçiminde de kullanılır.
[8] İngilizcesi: "oblivious RAM".

lar yapılarak ya da bellekte verilerin yeri değiştirilerek bellek erişiminin nasıl gerçekleştiğinin ayrımsanması engellenir. Rasgele okuma ve yazmalar ile yer değiştirmelerin sıklığı ilgisiz bellek kullanmanın sağladığı güvenlikle getirdiği fazladan yük arasında bir ödünleşmedir. Kurcalamaya dayanıklı donanımlarla oluşturulmuş bir güvenli hesaplama katmanının bu yöntemi kullandığı ve yürütülen programın da tam benzeryapılı şifreleme ile gizlenmiş verilere eriştiği varsayılırsa, sadece kullanılan güvenli hesaplama katmanı donanımı güvenli varsayılarak bulutta yürütülen programın işleyişini bellek erişimine bakarak kestirme olanağı da engellenebilir. Böyle bir donanımın mimarisi yakın zamanda tanımlanmıştır [105].

Ayrımsanamaz perdeleme[9] 2013 yılında önerilmiş bir yöntemdir [106]. Henüz gerçeklemesi yapılmamıştır. Bu yöntemde programın özünü oluşturan kriptolojik bir parça saldırgandan gizlenmek amacıyla çıkarılır. Buna programda bir *delik*[10] oluşturulması denir. Ancak bu deliğin yeri kriptolojik olarak gizlendiği için her iki programı karşılaştırması için kendisine verilen saldırganın delinmemiş "asıl" program ile delikli "nöbetçi" programı birbirinden ayırması olasılığı yoktur. Teoride, müşteri gizlemek istediği algoritmayı buluta yüklediği programın delikli alanının içine yerleştirirse *dürüst ancak meraklı* saldırgan olarak modellenmiş sağlayıcıdan gizleyebilir. Bu yöntemin gerçeklenmesi başarıldığında getireceği işlemci ve depolama yükü de göz önüne alınarak gelecekteki bulut dizgelerinde kullanılması olasıdır.

5.6 Vargı

Bu bölümde, kurulacak kullanışlı süreç kozası yapısı ile müşterinin belirleyeceği erişim kurallarının sağlayıcıya aktarılacağı, üstelik uygun kriptolojik yöntemlerle sağlayıcıdan bağımsız olarak kriptolojik yollarla uygulanacağı açıklanmıştır. Bu yöntemde erişim kurallarının sağlayıcı tarafından denetimi sadece programlama ile değil, kriptolojik araçlarla da gerçekleştirilmiştir. Denetimler sırasında sıradışı bir davranışla karşılaşılırsa eylemi gerçekleştiren kullanıcı 6. bölümde ele alınacak olan güvenli günlük düzeneği ile yadsınamaz biçimde belirlenir.

[9] İngilizcesi: "indistinguishability obfuscation", kısaltıldığında ilk harfi küçük ikinci harfi büyük ve kaligrafik yazılarak *iO* biçiminde kullanılır.

[10] İngilizcesi: "puncture".

Tasarlanan süreç kozalarının konaklarda çalışması için güvenli hesaplama katmanı sunulmuştur. Güvenli hesaplama katmanının temelde iki işlevi vardır. Birincisi, aynı sanal bağlamı tüm konaklarda sunarak bulutta sunulan bilgiişlem ortamının standartlaşmasını sağlar. İkinci ve önemli olan işlevi ise, süreç kozalarına sunduğu sanal bağlam ile her bir kozanın yalıtımını sağlamasıdır. Güvenli hesaplama katmanı yapısında bir süreç sanal makinası barındırdığı için işletim sistemi tarafından ayrı bir bellek uzayında tutulur. Güvenli hesaplama katmanı üzerinde çalışan programların eriştikleri kaynakların yalıtımı da kaynak kullanım kuralları ile sınırlandırılırsa müşteri programları çalışma mantıkları açısından özgür, ancak kullandıkları kaynaklar açısından sağlayıcı denetiminde olur.

Kural uygunluk noktası süreç kozaları üzerine iliştirilmiş erişim kurallarına göre kullanıcıların kozalara erişimine izin veren ya da vermeyen bir uzaktan haberleşme noktasıdır. Aynı zamanda süreç kozaları üzerinden kaynak kullanım kurallarını okuyarak bunları güvenli hesaplama katmanına bildirir ve müşteri programlarının uygun ayarlarla başlatılmasını sağlar.

Sayılan bileşenlerin tasarımıyla bilgiişlem ortamı sunan bulutta müşteri programlarının süreç olarak çalışması sırasındaki güvenliği ve yalıtımı incelenmiş, erişim haklarının nasıl uygulanacağı belirlenmiş, sağlayıcıdan bağımsız olarak gizliliğin elde edileceği ve bütünlüğün denetleneceği, sağlayıcının kaynaklarını koruyan, kullanışlı ve özelleştirilebilir bir yapı tasarlanmıştır.

Bulutun çok tartışılan sorunlarından olan “hesaplama sırasında verinin ve işleyişin açığa çıkması” konusu 5.5 bölümünde tartışılmış ve bugünkü kriptolojik araçlarla çözülemeyişinin nedenlerine değinilmiştir. Müşteri verilerini ve programlarını bulutta çalıştırıldıkları sırada da gizleyen yeni yöntemler de aynı bölümde tanıtılmıştır. Önümüzdeki yıllarda, verimlerinin artması durumunda bu yöntemlerin bulutta yaygın biçimde kullanılmaları olasıdır. Süreç kozaları sözü edilen yeni yöntemlerle uyum düşünülerek tasarlanmıştır.

6. GÜVENLİ GÜNLÜK TUTMA DÜZENEĞİ

Bulut, duyurulduğu günden [26] bu yana günlük hayata daha çok katıldıkça sağlayıcı ile kullanıcılar arasında gerçekleşen ileti alışverişinin ayrıntılarıyla bilinmesi gerekliliği de giderek önem kazanmıştır. 4. ve 5. bölümlerde güvenli bir bilgiişlem ortamı sunan bulut tasarımı yapılırken, değiş tokuşu yapılan iletilerin, bir başka deyişle iletiler sonucu oluşan olayların, güvenli bir günlük düzeneği kapsamında kaydedilmesi gerektiğinden söz edilmişti. Bu bölümün konusu bu amaca uygun bir güvenli günlük düzeneğinin tasarımıdır.

Önerilen günlük tutma düzeneği karşılıklı güven ilişkilerinin tartışmalı olduğu tüm hizmet alım senaryolarında kullanılmaya uygundur. Güven ilişkilerinin belirsiz olduğu bilgiişlem ortamı sunan ve sanal bilgisayar sunan bulutlar da bu genellemenin kapsamında kalır.

6.1 Var Olan Günlük Tutma Düzeneklerinin Yaklaşımı

Gerçekleşen olayların kaydını tutmak bilinen ve uzun zamandır kullanımda olan güvenlik önlemlerinden biridir [92, 107]. Ancak var olan günlük düzenekleri bulutun gerekleri ile tamı tamına bağdaşmaz. Eksikleri belirlemek ve bunları giderecek bir düzenek kurmak gerekir.

Geçmişten beri *ideal günlük tutma* olarak düşünülen, olayların gerçekleştikleri anda tek bir yazım yapılabilen[1] ortamlara aktarılmasıdır [108–110]. Çoğu zaman bu ideal düşüncenin yerini alan; olabilecek en kısa sürede güvenilir olduğu umulan haberleşme kanallarından saldırıya uğramadığı umulan bir merkezi bilgisayara (günlük sunucusuna) gönderilen kayıtlardır [108, 110, 111].

[1] İngilizcesi: “write-once”.

Bugüne dek bu konuda yapılan çalışmalar genelde üç ayrı alanda ve birbirinden bağımsız olarak yapılmıştır. Birinci alandaki çalışmalar, kayıtların günlük sunucusuna gönderilmeden önce olayların gerçekleştiği yerde ve kayıtlar gönderildikten sonra günlük sunucusunda nasıl güvenli saklanacağı üzerinedir. İkinci alandaki çalışmalar kayıtların günlük sunucusuna gönderilmesini bir standarda bağlamak üzerinedir. Üçüncü alandaki çalışmalar günlük sunucusunun mimarisi üzerinedir.

Kayıtları güvenli saklamak üzerine yapılan çalışmalar Bellare ve Yee'nin günlük kayıtlarının geleceğe dönük bütünlüğünü[2] ileti asıllama kodları[3] ile sağlamayı deneyen ilk önerisi ile başlar [108]. Schneier ve Kelsey'nin çalışması öncekini andırmakla birlikte bu yaklaşımın dağıtık ağ ortamlarına nasıl uyarlanacağını gösterir [110]. Holt'un bakışımsız şifrelemeyi öneren çalışması bakışımlı şifreleme kullanan yaklaşımları izleyen doğal bir evrimdir [111]. Holt'un çalışmasında, günlük kayıtlarının herkesçe doğrulanabilirliğinin sağlanması karşılığında hesaplama bedeli yükselmiştir. Ma ve Tsudik hesap yükünü azaltacak ilerlemeler önerirler [112]. Son olarak Yavuz v.d. bu yükü çok hafifletmeyi başarmışlar ve diğer çalışmalarla karşılaştırmalı bir tablo sunmuşlardır [109].

Günlük kayıtlarının taşınmasının standardını oluşturma çalışmaları ise endüstride *de facto* varlığını sürdüren Syslog etrafında gelişmiştir. Syslog, ilk olarak Sendmail programı içinde kullanılmak amacıyla Eric Allman tarafından geliştirilmiştir [113] ve daha sonra IETF tarafından standartlaştırılmıştır [114]. Syslog, ilkel biçimiyle, taşınan kayıtlar için herhangi bir koruma içermez. Hatta, taşıma sırasında UDP [115] kullanıldığından, kayıtların günlük sunucusuna ulaşacağı bile kesin değildir. Standart 2009 yılında yenilendiğinde taşıma sırasında kullanılacak protokol serbest bırakılmıştır [116]. Bir başka standart ile yeni standardın UDP ile nasıl kullanılacağı belirlenir [117]. İlkel Syslog standardının sağlıklı haberleşmesi için oluşturulan standardın [118] eşdeğeri sayılabilecek bir başka standart yeni Syslog standardının TCP protokolüne [119] uyarlanması ile oluşturulan standartla [120] elde edilir. Sağlıklı haberleşme sorunu böylelikle çözülmüş olur. Syslog güvenlik üzerinde çok fazla durmasa da, Syslog eklentileri güvenliği dikkate almışlardır. Kayıtların değiştirilmeden taşınması gereksinimini sağlamak üzere imzalanmış Syslog iletileri stan-

[2] İngilizcesi: "forward integrity".

[3] İngilizcesi: "message authentication code", kısaltılarak "MAC" olarak da kullanılır.

dartlaştırılmıştır [121]. Ancak bu kayıtların taşınırken gizliliğinin sağlanmadığını ve taşıma sonrasında imzalarla ilişkilerinin kopuk olduğunu unutmamak gerekir. Son olarak, taşıma sırasında gizliliği sağlamak üzere Syslog'u sırasıyla TLS [90] ve DTLS [122] üzerinde kullanmayı öneren standartlar önerilmiştir [123, 124].

Günlük sunucularının mimarileri üzerine çalışmalar genelde kurcalamaya dayanıklı donanımlar ya da güvenli hesaplama katmanı ile ilişkilidir. Chong ve diğerleri [125] kurcalamaya dayanıklı donanımlara Schneier ve Kelsey'nin çalışmasını [110] aktarmıştır. Accorsi güvenli hesaplama katmanını açık anahtar altyapısıyla birlikte kullanarak günlük sunucusuna ulaşan kayıtların güvenli saklanmasını sağlar [126]. Bu sırada kayıtların hem nasıl ulaştırılması gerektiğini belirtir hem de saklanacakları uygun biçemi tanımlar. Bunlardan farklı olarak Ray v.d. günlük sunucusunu buluta taşıyarak kayıtları saklama işini bir bulut hizmeti olarak sunar [127]. Bu yaklaşımda herhangi bir güvenli hesaplama katmanı ya da özel donanımdan söz edilmemişse de kriptolojik geleceğe dönük bütünlüğü koruma yöntemleri kullanılmıştır. Her kayıt şifrelenir ve her kayıt için geleceğe dönük ileti asıllama kodu oluşturulur. Ayrıca günlük dosyalarının tamamı için ayrı bir ileti asıllama kodu tutulur. Bu yaklaşımda günlük tutma görevini buluta bırakmak düşük bütçeli şirketler için uygun bir çözüm olarak görülse de günlük sunucusunun güvenliği ile ilgili tartışmaları bir adım ileri taşımaktan öte bir yenilik getirmemektedir. Saldırgan için hedef artık sunucu değil, bulut olacaktır. Bu, doğrudan doğruya 2.2.7 bölümünde *müşteri verilerinin sağlayıcı alanına taşınması (Z_1)* olarak sözü edilen zayıflıktır; zayıflık açıklanırken müşteri verileri olarak anılan ve sağlayıcıdan sakınılan değerler bu yaklaşımda günlük kayıtlarının kendisi olmaktadır. Zawoad v.d. Bellare ve Yee'nin özgün düşüncesini yakın zamanda bulutta yasal kanıt toplama bağlamında yeniden kurgulamışlardır [128]. Ne yazık ki, önceki yaklaşımın uzun zamandır bilinen hatalarını da tekrar etmişlerdir.

6.2 Tezde Güvenli Günlük Tutma Konusuna Yapılan Katkılar

Aşağıda var olan günlük tutma yaklaşımlarının sorunlarına değinilmiş ve bunların çözümleri belirlenmiş, bunun ardından bulutta karşılaşılan sorunları çözmeye yönelik

bir günlük tutma düzeneğinin taşıması gereken özellikler açıklanarak, önceki çalışmalarla kıyaslanmıştır.

6.2.1 Var olan yaklaşımların kısıtları ve günlük tutmada sık karşılaşılan sorunlar

Daha önce 6.1 bölümünde belirtilen *ideal günlük tutma*, olayların gerçekleştikleri anda tek bir yazım yapılabilen ortamlara aktarılmasıdır. Fakat bu tanıma uyularak kurgulanan düzenekler belirsizlikler taşır. "Gerçekleştiği an" sözü sorgulanmalıdır. Ayrıca, ortamların veri saklamadaki sürekliliği belirlenmelidir. Bunlardan başka, günlük kayıtlarını saklamak için kullanılacak tek bir yazım yapılabilen ortamın varlığıyla dolaylı olarak ortaya çıkan, ortamı elinde bulunduran kişiye duyulan güven ve ortamın can alıcı arıza noktası olması konuları etraflıca tartışılmalıdır.

Gerçekleştiği an kayıt elde edilmesi zor bir koşuldur. Genellikle günlük kayıtları gecikmesi olan ve kopmaların zaman zaman yaşandığı kanallardan taşınır. Bu, kayıtların saklanamaması olasılığını doğurur. Oysa, ideal bir düzenekte gerçekleşen olaylar ve kayıtların saklanması atomik olmalıdır [109]. Bunu uygulamada sağlamak olanaklı olmadığından kayıtları olay gerçekleşmeden önce saklamak uygun bir çözüm olur. Gerçekleşmemiş bir olayın kaydedilmesi de, daha sonra doğrulanmasını gerektirir. Böylece, günlük kayıtlarının sonuna onların gerçekleşip gerçekleşmediğini onaylayan açıklamalar eklenir. Bu çözüm bir sonraki 6.2.2 bölümünde 1. özellik olarak ele alınmıştır.

Günlük kayıtlarının taşınma ve saklanma sırasında değiştirilmeleri olasılığına karşı kriptolojik önlemler alınması gerekir. Kayıtların imzalanmasıyla hem değiştirilemezlik hem de kaydın varlığının yadsınamazlığı[4] sağlanır. Bu çözüm 6.2.2 bölümünde 2. özellik olarak tanımlanmıştır.

Kayıtların gizliliği pek çok düzenekte [108, 109, 111, 112, 114, 117, 118, 120, 121] eksik ya da isteğe bağlıdır. Uygun şifreleme yöntemleri ile gizliliğin sağlanması çözümü bir sonraki bölümde 3. özellik olarak tanımlanmıştır. Bu çözüm aynı zamanda kaydın kaynağının yadsınamazlığını[4] da sağlar.

[4]İngilizcesi "undeniability" ya da "non-repudiability" olarak kullanılan bu terim genelde iki yönlüdür. Bunlardan biri söz konusu verinin varlığının yadsınamazlığı, diğeri söz konusu verinin kaynağının yadsınamazlığıdır. Terim ayrı ayrı kullanıldığında "undeniability/non-repudiability of receipt" ve "undeniability/non-repudiability of origin" biçimlerini alır.

Bazı düzenekler günlük kayıtlarının silinmediğini anlamak üzere geleceğe dönük bütünlüğü kullanarak kayıtların sıralı bir silsile oluşturup oluşturmamasını denetler [108, 110, 111, 128]. Bu durumda aradaki bir kaydın sezilmeden silinemeyeceği açıktır. Ancak silsilenin sonunda yer alan kayıtlar sezilmeden silinebilir. Gerçekte saldırganın istediği de tam olarak budur; saldırıya başladığı andan sonraki kayıtları silmek [108, 127]. Güncel düzeneklerin çoğu [109, 112, 126, 127] bu tehdide karşı önlem almışlardır. Ancak, kayıtların tamamının silinmesinden pek söz edilmez. Elbette, günlük sunucusunu ele geçirmiş bir saldırgan için sona eklenmiş kayıtları silmekle kayıtların tamamını silmek arasında kayda değer bir fark yoktur. Bu düzeneklerde ya kayıtların hep birlikte silineceğini ya da hep birlikte güvende kalacağını düşünmek gerekir. Bunun yerine, her bir kaydın bir banka dekontunda olduğu gibi kendi başına anlamlı olduğu bir düzenek kurulur. Bu çözüm kayıtların yönetimini kolaylaştırarak silinmelerine karşı ele alınacak çözümleri kolaylaştırır. Bu, 4. özellik olarak adlandırılmıştır.

Kayıtların kendi başlarına anlamlı olmaları silinmelerine engel oluşturmaz. Silinmeyi engellemenin yolu kayıtların saklandığı ortamın can alıcı arıza noktası oluşturmamasıdır. Genelde hata hoşgörüsü olan dizgelerin tasarımında görülen toparlanma kavramı ile bu soruna çözüm getirilir. Günlük kayıtlarının birden çok kopyası farklı yerlere dağıtılır. Böylece, kopyalardan bazıları yitirilse dahi kalanlar düzeneğin çalışmasına yeterli olacaktır. Bu yaklaşımda önemli noktalardan biri, her bir kopyanın bütünden ayrılabilmesi ve kendi başına anlam taşımasıdır. Bu durumda yukarıdaki paragrafta sunulan özellik ile bu çözüm bir araya geldiğinde kayıtların silinmesinin önüne geçilir. Bu çözüm 6.2.2 bölümünde 5. özellik olarak sunulmuştur.

Kayıtların merkezi bir günlük sunucusunda tutulması saldırganlar için doğal bir hedef oluşturur [108, 127]. Şimdiye dek önerilmiş tüm günlük tutma düzenekleri taraflardan birini güvenilir olarak belirleyerek günlük tutma sorumluluğunu ona bırakır. Ancak, bulutta güvenilir olarak hangi tarafın seçileceği belirsizdir; kimse bu sorumluluğu hakkıyla üstlenemez. Gerçekte, güven ilişkilerini belirlemek, devam ettirmek ve yönetmek zordur. En iyisi, herhangi bir tarafın bir diğerine güvenmesini gerektirmeyen bir düzeneğin varlığıdır. Bu çözüm ilerleyen bölümde 6. özellik olarak adlandırılacaktır.

Var olan düzeneklerin aksine, tasarlanan düzenekte günlük kayıtları tüm taraflara dağıtılır. Kullanıcı tarafından sağlayıcıdaki bir hizmetin kullanılması için elverişli bir öneri yapılmışsa, bu uygun biçimde kaydedilerek bir anlaşmaya dönüştürülür ve anlaşmanın sonunda hizmetin sunulacağı güvence[5] altına alınarak taraflara günlük kayıtları dağıtılır. Bu çözüm 6.2.2 bölümünde 7. özellik olarak anılmıştır.

Kayıtlar taşınırken oluşan hatalara karşı önlem almamış düzenekler vardır. Geçici hatalarda kaydın yitirilmesi söz konusudur, geçici hatalara karşı iletilerin tekrarlanması durumunda da kaydın yinelenmesi olasıdır ya da bu bir saldırı olarak algılanır [114]. Tasarlanan düzenekte geçici haberleşme hataları düzenekte kurulu bulunan bildiri tahtası[6] üzerinden protokolün durumu izlenerek kolayca düzeltilir. Bu çözüm 8. özellik olarak anılmışır.

6.2.2 Tasarlanan günlük tutma düzeneğinin taşıdığı özellikler ve bu özelliklerin var olan yaklaşımlarla kıyaslanması

Bilgiişlem işinin bir yükleniciye bırakılmasındaki en önemli konu, değişen güven ilişkileridir. Hizmet sunulurken güven ilişkilerinin hatalı yorumu, özellikle bilgiişlem ortamı sunan bulutlarda ve sanal bilgisayar sunan bulutlarda, günlük kayıtlarının denetiminin yanlı biçimde sağlayıcıda kalmasına neden olur. Hazır yazılım sunan bulutlarda günlükler sunulan yazılımın bir parçası olacağı için tanım gereği sağlayıcı denetiminde olacaklardır. Bilinen kadarıyla, tezde anlatılan düzenek birbirine tam anlamıyla güvenmeyen iki tarafın ilişkisinde gerçekleşenleri bir günlük düzeneği ile kayıt altına alan ilk tasarımdır.

Günlük tutmak konusundaki var olan yaklaşımların sorunları ve bunların yanına bulut için kullanışlı başka özellikler de eklemek üzere düşünülen çözümler 6.2.1 bölümünde belirlenmiştir. Önceki çalışmalar üzerinde yapılan irdelemeler sonucu tartışılan çözümler göz önüne alınarak 6.3 ve 6.4 bölümlerinde açıklanmış olan tasarımda aşağıda sayılan sekiz temel özelliğin bulunmasına karar verilmiştir. Bu özelliklerin tamamı önerilen günlük tutma düzeneği anlatıldıktan sonra 6.7.1 bölümünde akıl yürütmeyle tanıtlanacaktır.

[5] İngilizcesi: “commitment”.
[6] İngilizcesi: “online bulletin board”.

Aşağıda sıralanan özellikler 6.1 bölümünde sayılan var olan düzeneklerle kıyaslanmak üzere hazırladığımız Çizelge 6.1'de sunulmuştur.

Kimi zaman var olan düzeneklerin aşağıda sayılan özelliklerden bazılarını karşılamadığı ya da hiçbir özelliği karşılamadığı görülmüştür. Buna karşın tarihsel olarak tablonun bütünlüğünü bozmamak için bu düzenekler tablodan çıkarılmamış, yorumu tartışmalı olabilecek kıyaslamalarda da en yakın seçenek yeğlenerek tablo tamamlanmıştır.

1. **Gerçekleşen olaylar ile bunların kayıtları sıkı ilişkilidir** Düzenekte, olaylar gerçekleşmeden önce kayıtlar saklanır. Kullanıcı sağlayıcıdan hizmet almak için bir istekte bulunur. İstek uygun görülürse sağlayıcı, kullanıcının kaynağa erişmesine izin vereceğine ilişkin bir güvence verir. Bu güvence kriptolojik bir yapıdadır ve ancak kaydın tutulmasının ardından etkin olur. Ancak, erişim izni verilse bile, erişim gerçekleşmeyebilir. Gerçekte hizmet alımının nasıl sonuçlandığı daha sonra kayıtlara eklenir.

2. **Günlük kayıtları değiştirilemez ve varlıkları yadsınamaz** Günlük kayıtları değiştirilmeye karşı imzalanmıştır.

3. **Günlük kayıtlarının içeriği gizlidir** Günlük kayıtları şifrelenmiştir ve ancak ilgili taraflarca erişilebilir.

4. **Her bir kayıt doğruluğunun kanıtlanması için tek başına yeterlidir** Her bir kaydın doğruluğu tek başına denetlenebilir. Bunun dışında, her bir hizmet alımını başlangıcından bitişine dek açıklayan üç kayıt sıralı olmak zorundadır. Bu sıra içinde söz konusu kayıtların içeriği sunulan hizmetin o ana kadar geldiği durumla ilgili anlamlı bilgi vermektedir.

5. **Günlük kayıtları silinemez** Her bir günlük kaydı tüm bulutta yürütülen hizmetin tüm taraflarına dağıtılır. Bundan başka, her kayıt bir bildiri tahtasında herkesin kopyalamasına uygun biçimde duyurulur. Böylece her bir kaydın çok sayıda ve dileyen herkeste kopyası bulunur.

6. **Kayıtlar tek bir tarafın denetimi altında değildir** Bulutta sunulan hizmetle ilgili her taraf kendi günlük kaydını elinde bulundurur. Ayrıca, bildiri tahtasında

yayınlanan kayıtların istenirse bir bölümü, istenirse tamamı herhangi biri tarafından kopyalanarak yedeklenebilir.

7. **Elverişli önerilere karşılık hizmet güvencesi veren anlaşmalar sunulur** Tasarlanan düzenekte tarafların ilişkisi bir iyimser adil değiş tokuş[7] olarak modellenir. Bu değiş tokuşta iyimser taraf olan kullanıcı, protokolü kendi zararına sonuçlanması riskini göze alarak başlatır. Protokolün adil olmasını sağlayan ise üçüncü taraf olan bildiri tahtasının gerektiğinde elindeki ipucu[8] ile uyuşmazlıkların çözümlenmesidir[9] [129]. Tasarlanan düzenekte değiş tokuş kullanıcının sunulan hizmeti kullanmayı önermesiyle başlar. Bu önerinin uygun bulunduğunu belirten bir anlaşmanın kayıt altına alınması ve kullanıcıya sunulmasıyla sürer. Öneride belirtilen hizmetin kullanılmasıyla değiş tokuş tamamlanır.

8. **Haberleşme kanalındaki geçici kesintiler düzeneği etkilemez** Geçici haberleşme hataları bildiri tahtasının izlenmesiyle sezilir. Böylece protokolün tutarlılığı bozulmadan iletilerin yinelenmesiyle hatalar düzeltilir.

6.3 Günlük Tutma Düzeneğinin Genel Yapısı

Günlük düzeneğinin genel yapısı Şekil 6.1'de gösterilmiştir. Temelde üç taraftan söz edilebilir. Bulutta bilgiişlem hizmetini başlatan müşteri ya da müşterinin programını kullanan kullanıcı, buluttaki bilgiişlem ortamını sunan sağlayıcı ve günlük kayıtlarının herkese duyurulduğu, aynı zamanda saklandığı bildiri tahtası. Konak, sağlayıcının altyapısı içinde, müşteri programlarının çalışmasına olanak veren bir birimdir. Kullanıcı, müşterinin sunduğu programı sağlayıcının konaklarından biri üzerinde kullanır. Kullanıcı bulutta gerçekleşecek olayın (bilgiişlem hizmetinin) başlatıcısı durumundadır. Konak da bu olayın gerçekleşeceği yerdir. Bildiri tahtası tek bir yazım yapılabilen, herkesin erişimine açık bir depolama alanıdır. Günlük kayıtlarının saklandığı temel depolama alanını oluşturur. Ayrıca, diğer taraflar arasında hizmete erişim konusunda bir uyuşmazlık çıkacak olursa üçüncü taraf olarak hakemlik yapar.

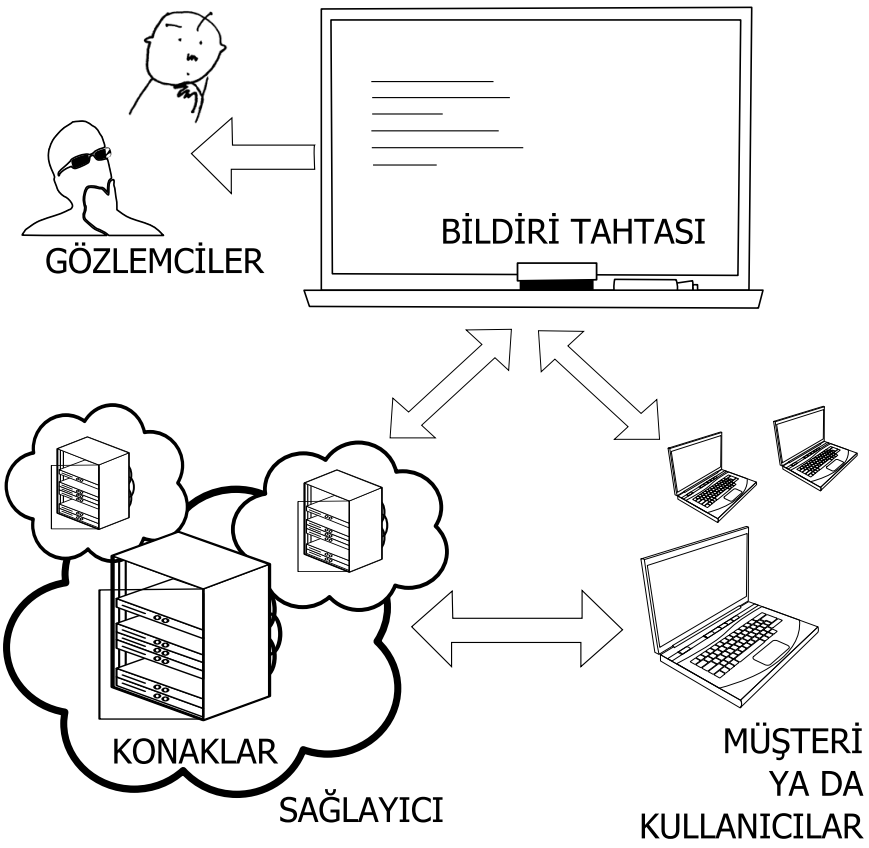
[7] İngilizcesi: "optimistic fair exchange".
[8] İngilizcesi: "evidence".
[9] İngilizcesi: "conflict resolution".

Çizelge 6.1 : Tasarlanan düzeneğin temel özelliklerinin var olan yaklaşımlarla kıyaslanması.

Özellik	Uyumlu	Uyumsuz
Olaylar ve kayıtlar sıkı ilişkilidir	Tasarlanan düzenek	[108–112, 114, 117, 118, 120, 121, 123–128]
Kayıtlar değiştirilemez ve varlıkları yadsınamaz	Tasarlanan düzenek, [108–112, 125, 128][b], [121, 123, 124][c], [126, 127]	[114, 117, 118, 120]
Kayıtların içeriği gizlidir[a]	Tasarlanan düzenek, [110, 111, 128][b], [123, 124][c], [125–127]	[108, 109, 112, 114, 117, 118, 120, 121]
Her kayıt tek başına kanıttır	Tasarlanan düzenek, [109][b], [121, 123, 124][c]	[108, 110–112], [114, 117, 118, 120][d], [125–128]
Kayıtlar silinemez	Tasarlanan düzenek, [109, 112, 125–127][e]	[108, 110, 111, 128][f], [114, 117, 118, 120, 121], [123, 124][c]
Kayıtlar tek tarafın denetiminde değildir	Tasarlanan düzenek	[108–112, 114, 117, 118, 120, 121, 123–126, 128], [127][g]
Adil değiş tokuş yapılır	Tasarlanan düzenek	[108–112, 114, 117, 118, 120, 121, 123–128]
Haberleşme hataları düzeneği etkilemez	Tasarlanan düzenek, [123, 124, 127]	[108–112][d], [114, 117, 118, 120, 121, 125, 126, 128]

[a] Tasarlanan düzenekte kaynağın yadsınamazlığı şifreleme ile sağlanmıştır.
[b] Sadece kayıtlar saklanırken.
[c] Sadece kayıtlar taşınırken.
[d] Bu düzeneğin bu özellik için kıyaslanması yerinde değildir.
[e] Kayıtların topluca silinmesine karşı önlem alınmamıştır.
[f] Silsilenin sonundaki kayıtlar sezilmeden silinebilir.
[g] Sağlayıcı güvenilir günlük sunucusu rolündedir.

Bu üç tarafın dışında herhangi bir özne gözlemci olarak düzeneğe katılabilir. Gözlemci kayıtların tutarlılığını kriptolojik imzaların tutarlılığını gözeterek her an doğrulayabilir. Bunun dışında, gözlemciler bildiri tahtasının içeriğinin bir bölümünü veya tamamını kopyalayabilir. Bağımsız gözlemcilerin düzeneğe katılımı ve içerik yedek-



Şekil 6.1 : Günlük tutma düzeneğinin bulutta işleyişi.

lemesi düzeneğin güvenliğini, hata hoşgörüsünü ve toparlanmayı artırır. Her bir kaydın kendi doğruluğunu kanıtlamaya elvermesi kopyaların gündelik hayattaki banka dekontları gibi kendi başına dağıtılıp saklanabilmesini sağlar. Böylece, kayıt bir kere oluşturulup bildiri tahtasında yayınlandıktan sonra taraflar kayıtları değiştirmekten veya silmekten caydırılır. Kayıtları değiştirmeye ya da silmeye kalkan bildiri tahtası, hatta kaydı ilk oluşturan taraf dahi olsa, bu değişiklik ya da silme girişimi daha önceki kaydın bir kopyası alınmışsa bu kopya ile kolaylıkla sezilir; böylece girişimde bulunanı zorda bırakır. Gözlemciler düzenekteki tüm tarafları ve düzeneğin içindeki taraflardan herhangi biri diğerlerini sürekli denetler durumdadır. Bu biçimiyle düzenekte pek çok adil değiş tokuş protokollerinden farklı olarak üçüncü taraf (hakem) olan bildiri tahtasının güvenilir olduğu varsayımını yapmaya da gerek yoktur.

Günlüğe yapılan her giri bildiri tahtasına gönderilen üç kayıttan oluşur. Bunlar sırasıyla *İSTEK*, *YANIT* ve *ONAY* kayıtlarıdır. Her günlük girisi, sunulan hizmetle sıkı ilişkilidir. Hizmet, *İSTEK* kaydı ile istenir, istek uygun bulunursa *YANIT* kaydı ile hizmetin sunulduğu bildirilir, *ONAY* kaydı ile hizmetin alındığına onay ve-

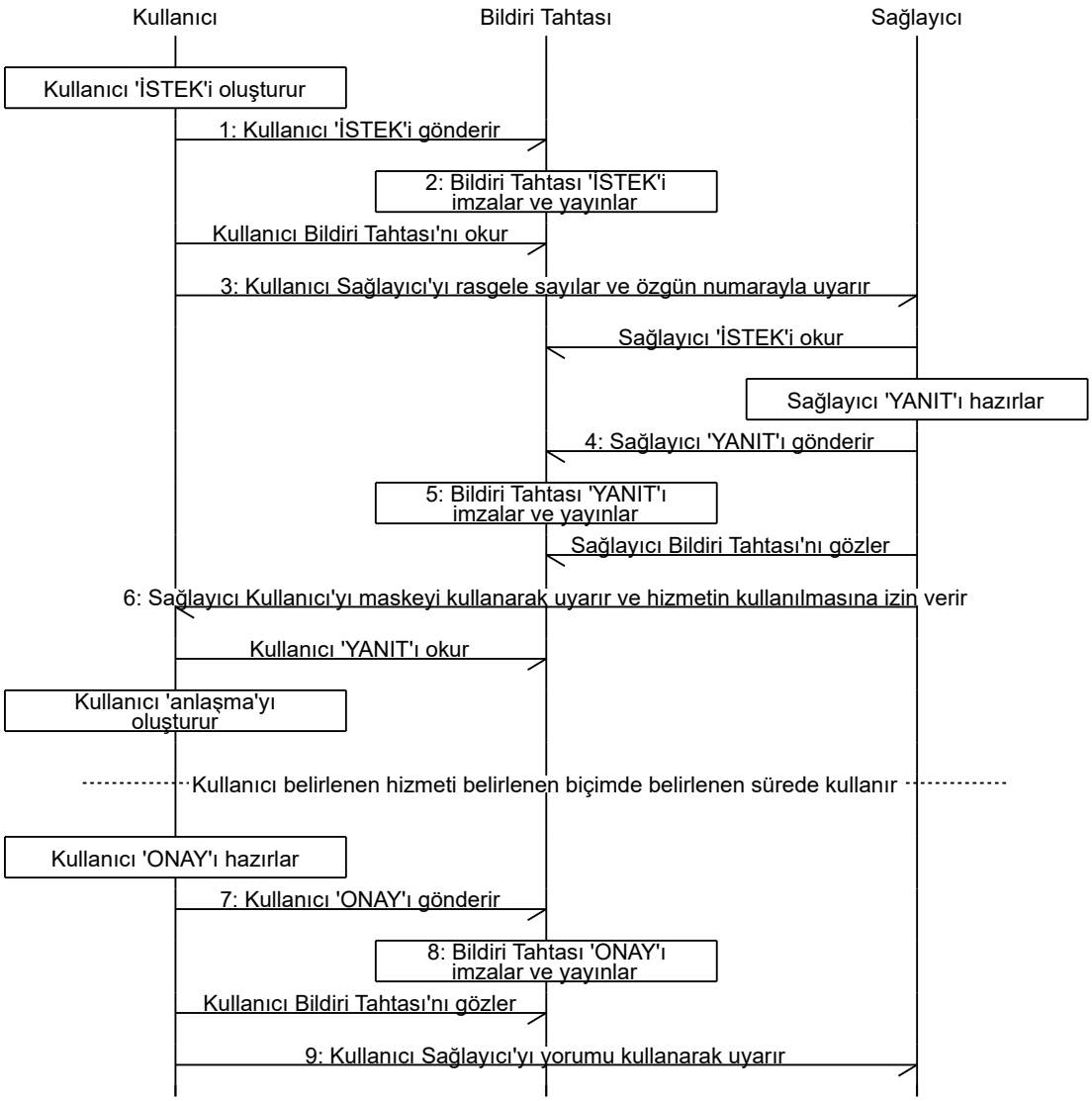
rilir. Kullanıcı tarafından olağan biçimde sonlandırılan başarılı bir akışı gösteren bir çizim Şekil 6.2'de gösterilmiştir. Şekilde numaralandırılmış olan etkinlikler Çizelge 6.2'deki sıra ile uyumludur. Sözü geçen üç kayıt içinden *İSTEK*, müşteri ya da kullanıcı tarafından oluşturulup bildiri tahtasına gönderilir. Sağlayıcı bildiri tahtasını okuyarak *İSTEK*i değerlendirir ve *YANIT*ı yine bildiri tahtasına gönderir. Verilen yanıt uyarınca müşteri ya da kullanıcı sunulan hizmeti belli bir süre kullanır. Hizmet alımını sonlandıran *ONAY* olur. *ONAY*, hizmetin sonlanışına göre, sağlayıcı, müşteri ya da kullanıcı tarafından oluşturulabilir. Örnek olarak, olağan bir sonlanışta bir kullanıcı *başarılı* yorumuyla *ONAY* kaydı oluşturur. Bir hata ile karşılaşıldığında kullanıcı *hata/uygunsuzluk* yorumunu kullanır. Öte yandan, kullanıcı *İSTEK* göndermesine karşın sunulan bilgiişlem ortamını kullanmıyorsa, belli bir süre bekledikten sonra, sağlayıcı, *zamanaşımı* yorumuyla *ONAY* kaydı oluşturur ya da kaynaklara erişim yasaklanmışsa sağlayıcı *hizmetYok* yorumunu kullanır.

6.4 Güvenli Günlük Tutma Protokolünün Adımları

Çizelge 6.2'de sağlayıcı konakları üzerindeki bir süreç kozasına bir kullanıcının erişmek üzere istekte bulunduğu, isteğin uygun bulunduğu, bulutta yürütmenin tamamlandığı ve kullanıcının da bunu onayladığı varsayılmıştır. Protokol çalışmaya başlamadan önce protokole katılan taraflar saatlerini aynı zamana ayarlamışlardır. Kısaca, çizelge her şeyin yolunda gittiği başarılı bir akışı göstermektedir.

Sağlayıcı, yapılan isteğin ardından kullanıcının ilgilendiği kozayı bir dizin hizmeti yardımıyla konakları arasından bulmuş ve bundan sonra kullanıcıyı ilgili konağa yönlendirmiştir. 4. bölümde anlatılan açık anahtar altyapısı yoluyla kişilerin bakışımsız şifreleme kullanmaları önünde bir engel yoktur. Bu sırada kullanıcı özlük bilgilerini sağlayıcı ile paylaşıyor ya da bir vekil sertifika kullanarak gizliyor olabilir. 5. bölümde anlatılan ayrıntılar uyarınca koza içindeki verilere erişim ve programların yürütülmesi olanaklıdır.

Açık bir haberleşme kanalından gönderilen bir ileti $\rightarrow$ ile gösterilir. $\mathbb{K}$, $\mathbb{S}$ ve $\mathbb{B}$ simgeleri sırasıyla kullanıcıyı, sağlayıcıyı ve bildiri tahtasını belirtir. Bildiri tahtasının $*$'a göndermiş olduğu iletiler yayınlanmaktadır. K, S ve B ilgili kullanıcıların gizli



Şekil 6.2 : Bulutta gerçekleşen bir olayın günlük girisini oluşturan akış.

anahtarları; K', S' ve B' ise açık anahtarlarıdır. D_s ile '$_s$' sırasını göstermek üzere bir zaman damgası anlatılır. $R_{\mathbb{XY}}$ rasgele bir sayı, aynı zamanda indislerle belirtilen taraflar arasında paylaşılan rasgele bir giri asıllama belirtecidir. $\ddot{O}$, erişim için yapılan öneriyi gösterir. A, öneri üzerine yapılan anlaşmayı belirtir. Y, gerçekleşen hizmet üzerine yapılan yorumdur. M, maske anlamında kullanılmıştır ve bununla gösterilen değer rasgele ikililerden oluşmaktadır. $\oplus$ ikili tabanda dışlayıcı VEYA işlemi yapar. $(\dot{I})_A$ ile $\dot{I}$ iletisinin A anahtarı ile şifrelendiği anlatılır. $\{\dot{I}\}_A$ ise $\dot{I}$ iletisinin A anahtarı ile imzalandığını gösterir.

Protokol üç aşamalıdır. Her aşama üç adımda yürütülür. İlk aşama kullanıcının bir $\mathcal{ISTEK}$te bulunması ve bu isteğin bildiri tahtasında yayınlanmasını içerir. İkinci aşamada sağlayıcının bu isteğe $\mathcal{YANIT}$ı bildiri tahtasında yayınlanır. Üçüncü aşamada

Çizelge 6.2 : Güvenli günlük tutma protokolü.

1:	$\mathbb{K} \rightarrow \mathbb{B}$	$\mathit{İSTEK} = (\mathbb{K}, (\ddot{O}, R_{\mathbb{KS}}, D_1)_K)_{S'}, (\mathbb{K}, (R_{\mathbb{KSB}}, D_1)_K)_{B'}$
2:	$\mathbb{B} \rightarrow *$	$N, \mathit{İSTEK}, D_2, \{N, \mathit{İSTEK}, D_2\}_B$
3:	$\mathbb{K} \rightarrow \mathbb{S}$	$((N, R_{\mathbb{KSB}}, R_{\mathbb{KS}})_K)_{S'}$
4:	$\mathbb{S} \rightarrow \mathbb{B}$	$\mathcal{YANIT} = ((A \oplus M, R_{\mathbb{KS}}, D_3)_S)_{K'}, ((N, M, R_{\mathbb{KSB}}, D_3)_S)_{B'}$
5:	$\mathbb{B} \rightarrow *$	$N, \mathcal{YANIT}, D_4, \{N, \mathcal{YANIT}, D_4\}_B$
6:	$\mathbb{S} \rightarrow \mathbb{K}$	$((N, M)_S)_{K'}$
7:	$\mathbb{K} \rightarrow \mathbb{B}$	$\mathcal{ONAY} = ((Y, R_{\mathbb{KS}}, D_5)_K)_{S'}, ((N, R_{\mathbb{KSB}}, D_5)_K)_{B'}$
8:	$\mathbb{B} \rightarrow *$	$N, \mathcal{ONAY}, D_6, \{N, \mathcal{ONAY}, D_6\}_B$
9:	$\mathbb{K} \rightarrow \mathbb{S}$	$((N, Y)_K)_{S'}$

gerçekleşen hizmet alımıyla ilgili bir yorum içeren $\mathcal{ONAY}$ iletisi taraflardan biri tarafından oluşturulur ve bildiri tahtasında yayınlanır. Çizelge 6.2'de gösterilen başarılı akışta $\mathcal{ONAY}$ iletisini kullanıcı oluşturmuştur.

Birinci adımda kullanıcı $\mathit{İSTEK}$ iletisini oluşturur ve bildiri tahtasına gönderir. Bu ileti iki parçalıdır. Kriptolojik olarak ilk parça sağlayıcının okuyabileceği, ikinci parça bildiri tahtasının okuyabileceği biçimdedir. Kullanıcı yapmak istediği erişim hakkındaki önerisini bu iletinin ilk parçasındaki *öneri* alanında belirtir. Bu alanda kullanıcının kimliği ya da yetkinliği, sağlayıcının kimliği, erişilecek kozayı tanımlayan yafta, erişim tipi ve gerekiyorsa erişim süresi, erişim koşulları gibi diğer belirleyici ayrıntılar yer alır. $R_{\mathbb{KS}}$ kullanıcı ile sağlayıcı arasında paylaşılan rasgele bir sayıdır. Bu, erişim için günlük girisi oluşturulurken kayıtların birbiriyle ilişkisini belirlemek için kullanılır. D_1 iletinin oluşturulduğu anı belirleyen zaman damgasıdır. Günlük girisini oluşturan ve bildiri tahtasında yayınlanan tüm iletiler bir zaman damgası içerir ve gerekli durumlarda kayıtların tutarlılığı[10] bu damgalar yardımıyla incelenir. İletinin ikinci parçasında öneri yer almaz. Rasgele değer olarak da ilk parçadan farklı olan $R_{\mathbb{KSB}}$ değeri bulunur. Bu değer sadece gerektiğinde ilk parçadakine benzer biçimde günlük kayıtlarının birbiriyle ilişkisini kanıtlamak üzere kullanılan, ancak üç taraf arasında paylaşılan bir değerdir.

İkinci adıma bildiri tahtası $\mathit{İSTEK}$ iletisinin ikinci parçasını inceleyerek başlar. D_1 damgası güncel zamanla tutarlıysa bildiri tahtası bu isteği geçerli sayarak özgün bir nu-

[10]Kayıtların tutarlılığı birbirini izleyen iki kaydın taraflarca önceden belirlenmiş zamanaşımı süresini aşmadan ve zamanın doğal akışını bozmadan ilgili tarafa ulaşmasıdır.

mara verir. Bildiri tahtası isteği imzalayarak yayınlar. Bu sırada D_2 zaman damgasını da ekler. Kullanıcı bu andan sonra bildiri tahtasını izleyerek yapmış olduğu isteğe ilişkin özgün numarayı öğrenir.

Üçüncü adımda, kullanıcı öğrenmiş olduğu özgün numara ile birlikte kayıtların birbiriyle ilişkisini sağlayacak rasgele sayıları sağlayıcıya gönderir. Bu ileti bildiri tahtasında yayınlanmayacağı için zaman damgası yoktur. Bu iletiyle birlikte sağlayıcı bir istek yapıldığının farkına varır.

Dördüncü adımda sağlayıcı $\mathcal{YANIT}$ iletisini oluşturur. Ancak iletiyi oluşturmadan önce kendisine ulaşan iletilerin tutarlılığını ve doğruluğunu denetlemelidir. Sağlayıcı özgün numaraya ilişkin isteği bildiri tahtasından okumakla işe başlar. D_1 ve D_2 zaman damgalarının tutarlılığını inceledikten sonra üçüncü adımda aldığı $R_{\mathbb{KS}}$ değerinin $\mathcal{İSTEK}$ içinde de bulunduğunu gözeterek bildiri tahtasındaki $\mathcal{İSTEK}$ ile kendisine üçüncü adımda gönderilen iletilerin ilişkili olduğunu görür.

Bu denetimlerin ardından oluşturulan $\mathcal{YANIT}$ iletisi, $\mathcal{İSTEK}$i andırır. İki parçalıdır. Birinci parça kullanıcının okuyabileceği biçimde, ikinci parça bildiri tahtasının okuyabileceği biçimde şifrelenmiştir. İlk parça erişim için gerekli olan anlaşmayı maskelenmiş olarak, $A \oplus M$ biçiminde içerir. $R_{\mathbb{KS}}$ değeri kayıtları ilişkilendirmekte kullanılacaktır. D_3 iletinin oluşturulduğu zamanı belirtir. İkinci parça özgün numarayı, maskeyi, üç taraf arasında paylaşılan rasgele sayıyı ve zaman damgasını içermektedir. Maskenin bildiri tahtasına gönderilmesi olası uyuşmazlıkların çözümünde ipucu olarak kullanılması içindir.

Maskelenmiş gönderilen anlaşma, anlaşmanın protokolün ilerleyen adımlarında kullanıcıya iletileceğine ilişkin bir güvence verir. Anlaşma ve maske dışlayıcı VEYA ile gizlendiklerinden sonraki adımlarda maskenin kullanıcıya iletilmesi ile kullanıcı anlaşmayı oluşturabilir. Maske değeri rasgele ikililerden oluşmuştur. Bu bakımdan, iyimser adil değiş tokuş sırasında üçüncü taraf olarak belirlenen bildiri tahtasına ipucu olarak maskenin verilmesi değiş tokuşta kullanılan değer hakkında üçüncü tarafa hiçbir bilgi sızdırmaz.

Anlaşma, buluttaki bir hizmete, kaynağa, veriye ya da programa erişim için gereken izindir. Anlaşmanın önemli bir özelliği, öneriye olan kriptolojik bağımlılığıdır. Bu

biçimiyle anlaşma sağlayıcının kullanıcıya sunduğu ikinci bir güvenceyi oluşturur. Bu, sağlayıcının öneride belirtilen koşullarla hizmeti sunacağı güvencesidir. A değerinin kriptolojik olarak yadsınamaz biçimde $\ddot{O}$ içeriğinin onaylandığı anlamına gelmesi her bir kaydın kendi kendine yeter ve tek başına anlamlı olmasını sağlayacaktır. Bunun oluşması için basitçe önerinin sağlayıcı tarafından imzalanması yeterlidir. Ancak bu durumda oluşacak anlaşma değeri aynı öneri için her zaman aynıdır ve kestirilebilir. Oysa, güvenliği artırmak için her bir günlük girisinde yeni ve kestirilemez bir A değeri üretilmelidir. Bu nedenle anlaşma değeri oluşturulurken bir rasgeleliğin katılması gerekir. Bu rasgelelik maske değeri olarak seçilmiş, böylece maske değeri önce rasgelelik kaynağı, daha sonra da ortaya çıkan anlaşma değerini maskelemek üzere iki ayrı işlevle kullanılmıştır. Böylece, anlaşma değeri $A = \{\ddot{O}, M\}_S$ ile belirlenir. $\ddot{O}$, $A \oplus M$, M değerlerinin bilinmesiyle anlaşmanın öneriye bağlılığının denetlenebileceği açıktır. Ayrıca, A değerinden yola çıkılarak $\ddot{O}$ değerinin elde edilemeyeceği gözden kaçmamalıdır.

Beşinci adımda, bildiri tahtası $\mathcal{YANIT}$ iletisini imzalayarak ve D_4 zaman damgasını ekleyerek yayınlar. Bunu yapmadan önce dördüncü adımda kendisine ulaşan iletinin ikinci parçasını okur. Buradaki N değerine bakarak ilgili $\mathcal{İSTEK}$ iletisinin ikinci parçasına ulaşır. Üç zaman damgasının sıralı ve tutarlı olduğunu denetler. Kullanıcı ve sağlayıcı tarafından aynı rasgele sayının gönderildiğini gözetir.

Altıncı adımda, bildiri tahtasında $\mathcal{YANIT}$ iletisinin yayınlandığını gören sağlayıcı, kullanıcıya maske değerini ve bu maskenin hangi erişimle ilgili olduğunu belirtmek üzere özgün numarayı içeren bir ileti gönderir. Böylece kullanıcı maskelenmiş $A \oplus M$ değerinden A değerini hesaplayarak erişim yapar. A değerinin $\mathcal{YANIT}$ içinde doğrudan sunulmayarak maskelenmesi ve maskenin ancak altıncı adımda kullanıcıya verilmesi erişimi kayıt altına alındığından emin olunana dek geciktirmek içindir. Dördüncü adımda hazırlanan $\mathcal{YANIT}$ iletisi ile geçerli bir erişim sağlanacağı hakkında güvence verilirken, beşinci adımda bu bildiri tahtasında yayınlanarak kayıt altına alınmış, son olarak sağlayıcı yadsınamazlık için bildiri tahtasında yayınlananların bir kopyasını alıp altıncı adımda maskeyi kullanıcıya göndermiştir.

Altıncı adımdan sonra kullanıcı bulutta sunulan hizmeti belirlenen süre boyunca kullanır.

Yedinci adımda, hizmetin sonlanmasından sonra, kullanıcı *ONAY* iletisi ile hizmet hakkındaki yorumunu belirtir. Bu ileti de iki parçalıdır; ilk parça sağlayıcı, ikinci parça bildiri tahtası içindir. İlk parça içinde hizmetin nasıl sonlandığını belirtmek üzere yorum alanı bulunur. *ONAY*ın diğer kayıtlarla ilişkisini sağlamak üzere rasgele sayılar, özgün numara ve zaman damgası kullanılmıştır. Çizelge 6.2'de gösterilen başarılı akışta *ONAY* iletisini kullanıcı oluşturmuşsa da *ONAY* iletisinin sağlayıcı tarafından oluşturulduğu bir akış yedinci adımdan başlanarak Çizelge 6.3'de gösterilmiştir.

Sekizinci adımda *ONAY* iletisi bir zaman damgası eklendikten sonra imzalanır ve bildiri tahtasında yayınlanır.

Dokuzuncu adım işleyişi hızlandırmak için kullanılacak isteğe bağlı bir adımdır ve *ONAY* bekleyen tarafın bildiri tahtasını sürekli denetlemesi gereğini ortadan kaldırır. Kullanıcı, yorumunu sağlayıcıya ayrıca göndererek uzun süren hizmetlerde erişimin sonlandığını bildirebilir.

ONAY iletisi içindeki *yorum* alanının dört farklı tipte olacağı öngörülmüştür. Bunların ikisi kullanıcı ya da müşteri tarafından gönderilir. Diğer ikisi ise sağlayıcı tarafından gönderilir. Olası durumlar aşağıda sıralanmıştır:

başarılı Kullanıcı/müşteri hizmeti başarıyla kullandı.

hata/uygunsuzluk Üzerinde anlaşma sağlanan hizmet kullanıcı/müşteri tarafından kullanılamadı. Bunun nedeni bir hata olabileceği gibi sağlayıcının bir başka işle uğraşı ya da uygunsuz davranışı olabilir.

zamanaşımı Kullanıcı/müşteri bir istekte bulundu ve sağlayıcı anlaşmayı imzalayarak erişim izni verdi. Ancak kullanıcı/müşteri daha sonra belirlenen süre içinde sunulan kaynaklara erişmedi.

hizmetYok Kullanıcı/müşteri isteğinin tanımlanmış erişim kurallarına uymaması veya önceden tasarlanmış bakım gibi nedenlerle hizmetin sunulamaması durumunda sağlayıcı geçersiz bir anlaşma gönderdikten hemen sonra yorum olarak *ONAY* iletisi içinde hizmetin sunulamadığı yorumunu yapar.

Çizelge 6.3 : $\mathcal{ONAY}$ iletisinin sağlayıcı tarafından oluşturulması.

$\overline{7}$: $\mathbb{S} \rightarrow \mathbb{B}$ $\quad \overline{\mathcal{ONAY}} = ((\overline{Y}, R_{\mathbb{KS}}, \overline{D_5})_S)_{K'}, ((N, R_{\mathbb{KSB}}, \overline{D_5})_S)_{B'}$

$\overline{8}$: $\mathbb{B} \rightarrow *$ $\quad N, \overline{\mathcal{ONAY}}, \overline{D_6}, \{N, \overline{\mathcal{ONAY}}, \overline{D_6}\}_B$

$\overline{9}$: $\mathbb{S} \rightarrow \mathbb{K}$ $\quad ((N, \overline{Y})_S)_{K'}$

6.5 Saldırgan Modeli

Sağlayıcı ve müşteri arasında bir hizmet sözleşmesi olduğu varsayımı yapılmaktadır. Buna bağlı olarak sağlayıcının denetimindeki konaklarda müşteri programları çalışmakta ve bu programlar müşteri ve müşterinin kullanıcıları tarafından kullanılmaktadır. Müşteri sağlayıcıya bir yüklenici olarak güvenmekle birlikte aldığı hizmeti denetlemek istemektedir. Diğer yandan, sağlayıcı müşterinin ve onun kullanıcılarının altyapıyı ne biçimde kullandıklarını kayıt altına alıp ileriye dönük olarak denetlemek istemektedir. Duyulan karşılıklı güven, içerdiği usa uygun şüphe ile birlikte, herhangi bir tarafı kayırmayan yadsınamaz bir günlük tutma düzeneği gerektirir. Bu düzeneğin sağlıklı çalışması için ortamda en az bir güvenilen bildiri tahtası olmalıdır.

Bildiri tahtası çoğu zaman gönderilen iletinin yazılı kaldığı silinemez bir haberleşme ortamı olarak kullanılmaktadır. İşlevi kendisine gönderilen iletileri imzalayarak yayınlamaktır. İyimser adil değiş tokuş içindeki üçüncü taraf olması nedeniyle zaman zaman kendisine uyuşmazlıkları çözmesi için başvurulur.

Protokol tasarlanırken kullanılan tüm haberleşme kanalları Dolev-Yao modelinde düşünülmüştür [130]. Bu modelde saldırgan kanaldaki tüm haberleşmeyi dinleyebilir, kesebilir, silebilir, yeni iletiler oluşturabilir ya da bir iletiyi istediği gibi değiştirebilir. Bunun dışında, saldırganın protokoldeki tarafları bir başka tarafmış gibi kandırabileceği düşünülmüştür. Ancak, saldırgan bir tarafın yerine geçmekle onun bilgilerini edinemez.

Kanal saldırıya açık olmakla birlikte kanalda taşınan iletiler iletişim kuranlardan başkasının okuyamayacağı biçimde şifrelenmiştir. Böylece saldırganın iletilerin içeriğine erişmesi engellenmiştir. Saldırgan iletiyi kanalda taşınırken değiştirse bile alıcı taraf şifreyi çözerken bunu geçici bir gürültü olarak algılayacaktır. Benzer biçimde,

süregiden bir bozma girişimi de, saldırganın karşı taraf yerine geçtiği ve gönderilen iletilere yanıt vermediği bir durumu andırır.

6.6 Tasarlanan Düzeneğe Yönelik Saldırı Girişimleri

6.3 bölümünde gösterilen yapıda kurulan ve 6.4 bölümünde açıklanan protokole göre işleyen düzeneğe yönelik dikkat çeken saldırı senaryoları ve bu saldırıların yapılan tasarımla nasıl engellendiği bu bölümde açıklanmıştır.

1. Saldırgan bildiri tahtasındaki kayıtları kullanmayı deneyebilir. Bildiri tahtası herkesin erişimine açık olduğu için saldırganın da burada yazılanlara erişimi vardır. Ancak yayınlanan günlük kayıtları şifrelenmiş olduğu için içerikleri okunamaz. Saldırganın kayıtlarla yapabileceği tek şey, denetçilerin yapabileceği gibi, imzalara bakarak tutarlılıklarını doğrulamaktır.

2. Bildiri tahtası uygunsuz davranışla herhangi bir yarar sağlayamaz. Buna karşın uygunsuz davranmayı seçse bile, değiştirilmiş bir iletiyi yayınlamasının hemen ardından diğer taraflarca bunun farkına varılacaktır. Bu andan sonra, işlevde hiçbir bozukluk olmadan taraflar dürüst hizmet veren bir başka bildiri tahtası kullanmayı seçerler. Taraflar önceki kayıtların kopyalarını ellerinde bulundurdukları için protokolü baştan başlatmaları dahi gerekmemektedir, sadece dürüst olan bildiri tahtası üzerinde taraflarca anlaşmaya varıldıktan sonra bozuk olan kaydın yinelenmesi gerekir.

3. Bildiri tahtasına gönderilen iletilerin ilk parçalarının şifreleri bildiri tahtası tarafından açılamaz. Bu nedenle, Dolev-Yao modelinde saldırganın ilk parçalarda yapacağı değişiklikleri bildiri tahtası algılayamaz. Bu değişiklikler ancak iletiler yayınlandıktan sonra gönderen tarafından, ya da bu sırada farkına varılmazsa ilerleyen adımlarda alıcı tarafından, algılanır. Bu durumda göndericinin iletiyi değişmeden ulaştırana dek tekrar göndermesi gerekecektir. Görünürde bu saldırının kanaldaki bir gürültüden farkı yoktur.

4. İletilerin sürekli değiştirilmesi bir hizmet engelleme saldırısı oluşturur. Yine de hatalı bir kayıt oluşturulamaz. Süregiden kanal bozmalara karşı Turing testleri [131]

ya da soğan ağı[11] [132] gibi bilinen yöntemler vardır, bu nedenle sürekli bozmalara karşı kapsamlı çalışmalara tezde değinilmemiştir.

5. Bildiri tahtasından yapılacak okumalar saldırgan tarafından bozulabilir. Okumalar eşzamanlı yapılmak zorunda olmadığı için protokolün işleyişi bozulmaz, doğru bir okuma yapılana dek protokolün akışının bekletilmesinde sakınca yoktur. Hatta, doğru okuma yapılması, teoride, çevrimdışı güvenli bir kanaldan sağlanabilir [133]. Doğru okuma yapıldığı imzalar doğrulanarak anlaşılır.

6. Bildiri tahtası ile kurulacak iletişimde (TLS [90] ya da TOR [132] gibi) güvenli bir kanalın kullanımı yararlıdır ancak bir koşul değildir. 2., 3. ve 5. senaryolarda sözü edilen tekrarlar protokolün işleyişini etkilemez. Yine de odaklanmış bir saldırı sırasında güvenli bir kanal yinelenen iletileri engelleyerek başarımı büyük oranda artıracaktır. Güvenli kanal olarak soğan ağı seçilecek olursa hizmet engelleme saldırıları oluşmadan önlenir. Ancak soğan ağının haberleşme başarımına getireceği fazladan yük hesaba katılmalıdır.

Aşağıdaki iki senaryoda kullanılan iletilerin sıraları Çizelge 6.2 ya da Şekil 6.2'de sunulmuştur.

7. Bildiri tahtasına 2. senaryoda belirtilen nedenlerle 2., 5. ve 8. iletiyi yayınlaması konusunda güvenilir. Böylece 1., 4. ve 7. iletilerin kanalda yitmesi durumunun kolaylıkla farkına varılarak gerektiğinde bu iletiler yinelenir. 3. ileti yitirilirse $\mathcal{YANIT}$ yayınlanmayacaktır ve kullanıcı 6. iletiyi alamayacaktır. Bir süre sonra bunu gören kullanıcı 3. iletiyi yineler. Bu durumun protokolde bir gecikme dışında bir etkisi olmaz.

8. Ayrıca incelenmesi gereken bir durum 6. iletinin yitmesidir. Hizmet sunulmasına karşın kullanılmazsa sağlayıcı tarafından algılanır ve iletinin yinelenmesiyle kolayca çözülür. Ancak, en kötü durum 4. iletinin iletilmesinden sonra 6. ileti gönderilmeden sağlayıcının devre dışı kalmasıdır. Böylece $\mathcal{YANIT}$ bir kayıt olarak bildiri tahtasına işlenmiş olur ancak kullanıcı hizmeti kullanmak üzere maskeyi elde edemez. Bu davranış düzeneğin birinci özelliği olarak sayılan *gerçekleşen olaylar ile*

[11] İngilizcesi: "onion network".

bunların kayıtları sıkı ilişkilidir ilkesine aykırıdır. Bu aykırılığı düzeltmek üzere uyuşmazlığın çözümü için bildiri tahtasına başvurulur. *M* değeri böylece bildiri tahtasından elde edilir. Elde edilemediği durumda ONAY iletisindeki *yorum* alanı kullanılır. Kullanıcı bu alana *hata/uygunsuzluk* açıklaması içinde maske değerini hangi nedenle alamadığını yazar. Bunun ayrıntıları 6.7.1 bölümünde 7. özelliğin tanıtında verilmiştir.

6.7 Tasarlanan Düzeneğin Güvenlik Açısından Yorumu

Tasarlanan düzeneğin güvenlik çözümlemesi iki adımda yapılacaktır. İlk adımda güvenlik çözümlemesi ileri sürülen özelliklerin sağlanıp sağlanmadığını akıl yürütmeyle doğrulamak üzerinedir. İkinci adımda düzeneğin çekirdeğini oluşturan Çizelge 6.2'deki protokol biçimselleştirilmiş ve model yoklama[12] yöntemiyle doğrulanmıştır.

[12]İngilizcesi: "model checking".

6.7.1 Tasarlanan düzeneğin özelliklerinin doğrulanması

Tasarlanan güvenli günlük tutma düzeneği için 6.2.2 bölümünde belirlenmiş olan sekiz özelliğin karşılandığı aşağıda akıl yürütme ile gösterilmiştir.

Aşağıda sıralanan akıl yürütmelerin bazılarında protokolün sürekliliği önemlidir. Temelde, iyimser adil değiş tokuşlar protokolün sonlanması varsayımına dayanırlar [134]. Bu bakımdan, sunulan yaklaşımın temel kolaylığı, bildiri tahtasının belleği olan bir haberleşme kanalı olarak düşünülebilmesi, protokol kesintiye uğrasa dahi tarafların daha sonra kaldıkları yerden protokolü sürdürebilmeleridir. Böylece protokolün *eninde sonunda* biteceği kesinleşir. 6.6 bölümündeki 4–6 senaryolarında açıklandığı gibi haberleşme kanalındaki süregiden saldırılar ve bozmalarda bile iletişimin farklı yolları bulunabilir. Hiç değilse, teoride, güvenilir çevrimdışı bir kanal oluşturulabilir [133].

1. **Gerçekleşen olaylar ile bunların kayıtları sıkı ilişkilidir** Olaylar ile kayıtların *sıkı ilişkili* olması, olay *gerçekleştiği anda* kayıt tutulamasa bile, kurulan düzenekle, kayıtların olayın gerçekleşmesinden önce tutulması ve olayın gerçekleşip gerçekleşmediğinin daha sonra doğrulanmasıdır. Bu karışık açıklamanın zamansal mantık[13] [135, 136] kullanılarak biçimsel biçimde yazılmasının doğrulanması sırasında kolaylık sağlayacağı düşünülmüştür.

   $$\Box(uygun(\mathbb{S}.\ddot{O}) \rightarrow (gönder(A \oplus M) \wedge (\neg gönder(M)\ \mathscr{U}\ yayın(\mathcal{YANIT})))) \quad (6.1)$$

   Gösterilen denklem 6.1'de alışıldık zamansal mantık işleçleri dışında, üç yapıtaşı edim[14] vardır. $uygun(\cdot)$, sağlayıcının öneriyi uygun bulduğunu ve protokole devam etmek istediğini bildirir. $gönder(\cdot)$, sağlayıcının kullanıcıya herhangi bir yolla argüman olarak belirtilen değeri gönderdiğini bildirir. $yayın(\cdot)$, argüman olarak belirtilen değerin bildiri tahtası tarafından alındığını, saklandığını ve imzalanarak yayınlandığını belirtir. Böylece diğer taraflar argüman olarak verilen bu değere erişebileceklerdir.

   Gerçekleşen olaylarla günlük kayıtları arasında uygulamada atomik ilişkiler kurma olanağı olmadığı için kayıtların olayların gerçekleşmesinden önce tutulmasına karar

[13] İngilizcesi: "temporal logic".
[14] İngilizcesi: "action primitive".

verilmiştir. Kayıtların olaylardan önce tutulabilmesi için olayların gerçekleşmesine ilişkin güvenceler içermesi gerekir. Bu güvencelerin sağlanıp sağlanmadığı olayın ilerleyişine göre kayda geçirilir.

Protokolde sağlayıcının hizmet sözleşmesine uygun bir öneriyi olumlu karşılaması beklenmektedir. Bunun ardından sağlayıcı kullanıcıya iki güvence sunar. Bunlardan biri hizmeti kullanmasının onaylandığını gösteren anlaşmadır. Bu hizmetin kullanıcıya sunulacağını yadsınamaz biçimde gösteren bir belge olarak saklanacak bir güvencedir. İkinci güvence ise, ilk güvencenin ancak kaydın tutulmasından sonra verileceğine ilişkin güvencedir ve bu ikinci güvence kayıt tutulmadan önce verilir.

Anlaşma, kullanıcıya, öncelikle maskelenmiş olarak gönderilir. Bu, anlaşmanın açık biçimde gönderileceğinin güvencesi olmakla birlikte kullanıcının kaydın bildiri tahtasına geçirilmesinden önce hizmete erişmesini engeller. Sağlayıcı bildiri tahtasına kaydın geçirildiğini gördükten sonra maskeyi kullanıcıya göndererek erişime izin verir. Anlaşma, öneri ve maske arasındaki ilişki $A = \{\ddot{O}, M\}_S$ biçiminde tanımlanmış olduğu için kullanıcı anlaşmayı elde ederek bunun öneriye uygunluğunu denetler. Bu, aynı zamanda öneride belirlenen biçimde erişimin yadsınamaz biçimde sağlanacağına bir kanıt oluşturur. Kullanıcı bundan sonra A değerini kullanarak hizmet erişimini gerçekleştirmeyi dener. Hizmet alımının sonucu *ONAY* kaydı içinde belirtilir.

Dikkat edilirse anlaşmaya bakılarak herhangi bir yolla önerinin ya da maskenin hesaplanamayacağı görülür. Böylece A değerinin erişim sırasında kullanımıyla yapılan önerinin ayrıntılarının açığa çıkması da engellenir.

2. **Günlük kayıtları değiştirilemez ve varlıkları yadsınamaz** Bildiri tahtası her kayda ikinci bir zaman damgası ekledikten sonra kaydın varlığının yadsınamazlığını ve değiştirilemezliğini sağlamak üzere imzalar. İkinci zaman damgası kaydın oluşturulduğu an ile yayınlandığı an arasında bir tutarlılığın gözetilmesini sağlar.

3. **Günlük kayıtlarının içeriği gizlidir** Hizmetin durumunu gösteren içerik ancak *İSTEK* içinde *öneri* alanında ya da *ONAY* içinde *yorum* alanında bulunur. Kayıtlarda sadece bu alanlar değil, tüm alanlar bakışımsız şifreleme kullanılarak kaydı oluşturan tarafından sadece ilgili kişinin şifresini çözüp içeriğini okuyabileceği

biçimde şifrelenmiştir. Böylece gizlilik sağlanmakla kalınmamıştır, kaynağın yadsınamazlığı da sağlanmıştır.

4. **Her bir kayıt doğruluğunun kanıtlanması için tek başına yeterlidir** Diğer pek çok çalışmadan [108–112, 127] farklı olarak, kayıtlar bir silsile olarak düşünülmemiştir, her bir kayıt ayrıca imzalanmıştır. Böylece, bir banka dekontunda olduğu gibi, tek bir kaydın varlığı, o kaydın gösterdiği olayın gerçekleştiğini kanıtlamaya yeter. Bir girinin (birbiriyle ilişkili üç kayıt: $\mathcal{İSTEK}$, $\mathcal{YANIT}$ ve $\mathcal{ONAY}$) varlığı ile bir hizmetin istenmesi, sunulması ve tamamlanmasına kadar geçen süredeki tüm döküm elde edilir. Günlük girisini oluşturan üç kayıt rasgele sayıların yinelenmesi ve kriptolojik güvencelerin kullanımıyla birbirine bağlanmış ve sıralanmıştır. $\mathcal{İSTEK}$ kaydının varlığı hizmete erişimin ayrıntılarını içerir. $\mathcal{YANIT}$ kaydının varlığı kullanıcının hizmete erişimine izin verildiğinin kanıtıdır. $\mathcal{ONAY}$ kaydının varlığı ise hizmetin ne biçimde kullanıldığını bildiren kanıtı oluşturur.

5. **Günlük kayıtları silinemez** Günlük kayıtlarının silinemezliği kayıtların protokole katılan taraflara gönderilmesiyle ve herkesin kopyalayabileceği biçimde bildiri tahtasında yayınlanmasıyla sağlanır. Bu sözel açıklama, $Taraf = \mathbb{K} \cup \mathbb{S}$ olmak üzere denklem 6.2'de biçimselleştirilmiştir.

$$\begin{aligned} &\Box(yayın(\mathcal{ONAY}) \rightarrow \\ &\forall Taraf\ [(Taraf.\mathcal{İSTEK} \wedge Taraf.\mathcal{YANIT}) \wedge \Diamond Taraf.\mathcal{ONAY}] \end{aligned} \tag{6.2}$$

Bildiri tahtasındaki herkese açık verinin hata hoşgörüsü içermesinin yanında kayıtların değiştirilemez, yadsınamaz ve tek başına doğrulanabilir olması özelliklerinin birleşmesinin sonucunda kayıtlar silinemez. Var olmadığı savlanan bir kaydın varlığı herhangi birinin elindeki bir asıl kayıtla kolaylıkla gösterilir. Bu, kaydı silen kişiyi zor duruma düşüreceği için kayıtları silmeyi (ya da değiştirmeyi) tasarlayan kötücül kişiler için caydırıcıdır. Asıl kopyalar protokol sırasında ilgili taraflara ulaştırılır ve bildiri tahtasında yayınlanır. Yayınlanan kayıtlar herkesçe kopyalanabilir. Yerinde bir saldırı için tüm kopyaların silinmesi gerekir. Kötücül kişi, herkese açık alandaki bir verinin kaç kişi tarafından kaç kopya saklandığını bilemeyeceği için geniş çaplı bir saldırı düzenlemesi olasılığı da yoktur. Özetle, en az bir kopyanın yedeklenmesi durumunda günlük kayıtlarının silinmesi olanaksıza yakındır. Düzenekte her an en az bir kopya bulunduğundan emin olunması için bazı tarafsız sivil toplum kuruluşlarının bildiri tahtasının yedeğini alması salık verilir.

6. **Kayıtlar tek bir tarafın denetimi altında değildir** Gerçekleşen olaylara ilişkin günlük kayıtları protokol sırasında ilgili taraflara ulaştırılır. Kayıtlar ayrıca bildiri tahtasında da yayınlanır. Yayınlanan kayıtları bağımsız denetçiler (Örneğin, noter rolünü üstlenecek bir sivil toplum kuruluşu ya da devlet kurumu.) kopyalayarak yedekleyebilir. Böylece, sorumluluk hiçbir tarafa bırakılmadan paylaşılır. Kayıtların yedeklenmesi ayrıca hata hoşgörüsünü de artırır. Bu yolla *hata hoşgörüsünün olmayışı (Z_8)* zayıflığının getireceği tehditlere karşı da bir önlem alınmış olur.

7. **Elverişli önerilere karşılık hizmet güvencesi veren anlaşmalar sunulur** Protokolde kullanıcının elverişli önerisine karşılık, sağlayıcı kullanıcının hizmeti kullanmasını sağlayacak bir anlaşma sunar. Bu anlaşma, ancak görüşme kayıt altına alındıktan sonra sunulur. Bu koşulu biçimsel dilde göstermek üzere zamansal mantıkta izleyen önermeler yazılmıştır:

$$P : \Box(\mathbb{K}.\ddot{O} \rightarrow \Diamond\mathbb{S}.\ddot{O}) \tag{6.3}$$

$$R : \Box((uygun(\mathbb{S}.\ddot{O}) \wedge \mathbb{S}.M \wedge \mathbb{S}.A) \rightarrow \Diamond(\mathbb{K}.(A \oplus M) \wedge \mathbb{B}.M)) \tag{6.4}$$

$$S : \Box(yayın(\mathcal{YANIT}) \rightarrow \Diamond\mathbb{K}.M) \tag{6.5}$$

Burada anılan P, R ve S önermeleri protokol uyarınca verildikleri sırada gerçekleşmek zorundadır. Bu değiş tokuşun sonunda, denklem 6.4'te gösterildiği gibi A değeri ve denklem 6.5'te gösterildiği gibi M değeri kullanıcının eline geçer. Denklem 6.5'te belirtilen $yayın(\cdot)$ yapıtaşı edimi $\mathcal{YANIT}$ın kayıt altına alındığını göstermektedir.

Protokolün her aşamasının ardından bildiri tahtasına bir kayıt eklenir. Yayınlanan bu kayıtların hem kaynağı hem de varlıkları yadsınamaz. Bu bölümün başında açıklandığı gibi, bu kayıtların eninde sonunda alıcısına ulaşacağı varsayıldığı için, protokole katılan tarafların protokolü sürdürmesi ve protokolü eninde sonunda sonlandırması beklenir. Protokolün sonlanması için yeter koşul, $\mathcal{ONAY}$ kaydının bildiri tahtasında yayınlanmasıdır. Yayınlanan bir kayıt eninde sonunda taraflarca okunacaktır.

Kullanıcının hizmet sözleşmesi çerçevesinde bir kaynağa erişmek üzere bir öneri ile istekte bulunduğu düşünülürse, olumlu bir anlaşma ile yanıt alacaktır. Bu anlaşma maskelenmiş olarak bir güvence durumunda gönderilir. Kullanıcının güvenceyi al-

masıyla birlikte protokolün sonlanması için dört olasılık kalır. Bu olasılıklar aşağıda gösterilmiştir.

$\mathcal{YANIT}$ kaydı içinde gönderilen güvence $\{\ddot{O},M\}_S \oplus M$ biçiminde, anlaşmanın maskelenmesiyle oluşturulur. Anlaşmanın kendisi, $A = \{\ddot{O},M\}_S$, öneriye bağlı bir imzadan oluşmuştur. Böylece M ve $\ddot{O}$ değerleri ile birlikte anlaşma yadsınamaz. Anlaşmanın protokolün dördüncü adımında maskelenmesi, kullanıcının günlük kaydının yayınlanmasından önce erişimini engellemek içindir. Maskenin kendisi, değiş tokuş sırasında ortaya çıkabilecek uyuşmazlıkların çözümü için yine $\mathcal{YANIT}$ kaydı içinde, değiş tokuşta üçüncü taraf rolünü üstlenen bildiri tahtasına ayrıca gönderilir.

(a) Sağlayıcı dürüst davranırsa ve herhangi bir hata oluşmazsa protokolün altıncı adımında maskeyi elde eden kullanıcı anlaşmayı oluşturarak bunu hizmete erişmekte kullanır. Bu, 6.2.2 bölümünde 7. özellik olarak yapılan iyimser adil değiş tokuş tanımına uygundur.

Sağlayıcı altıncı iletiyi göndermezse ya da 6.6 bölümündeki 8. senaryoda anlatıldığı gibi haberleşmede bir hata oluşursa ya da maskeleme kaldırıldıktan sonra elde edilen değer öneri ile belirtilen kaynak erişimini karşılayan geçerli bir imza değilse uyuşmazlık ortaya çıkar. Bu uyuşmazlığın çözümü için kullanıcı bildiri tahtasına başvuruda bulunur. Bu başvurunun bağımsız bir kanalda yürütüldüğü varsayılmıştır. Uyuşmazlık aşağıda gösterilen üç ayrı olasılıkla çözülür.

(b) Kullanıcı, maskenin bildiri tahtasında bulunan kopyasını kullanarak anlaşmayı elde eder ve hizmeti olağan biçimde kullanır. Bu, 6.2.2 bölümünde verilen tanıma uygundur.

(c) Kötücül sağlayıcı $\mathcal{YANIT}$ kaydı içinde bildiri tahtasına sahte maske değeri göndermiştir. Bu, kullanıcı bir uyuşmazlık çözümünü deneyene dek ortaya çıkmaz. Uyuşmazlık çözümünün sonunda da kullanıcı geçerli bir anlaşma oluşturamaz. Ancak bu durum hem bildiri tahtasının tanıklığında gerçekleşir hem de bildiri tahtasına gönderilen sahte maske değerinin kaynağı ve varlığı yadsınamaz biçimde kayıtlıdır. Bu durumda kullanıcı kolaylıkla yasal yollara başvurarak hizmet alamadığını kanıtlar. Bu sonuç da verilen tanıma uygundur.

(d) Kötücül bildiri tahtası uyuşmazlık çözümü sırasında *YANIT* kaydı içindeki geçerli maske değeri yerine sahte bir maske değerini kullanıcıya gönderir ya da uyuşmazlık çözümü protokolünü hiç yürütmez. Bir önceki duruma benzer biçimde, *YANIT* kaydı kaynağı ve varlığı yadsınamaz biçimde kullanıcının elinde olduğu sürece, kullanıcı yasal yollarla hizmet alamadığını ve bunun sorumlusunu kanıtlayabilir.

8. **Haberleşme kanalındaki geçici kesintiler düzeneği etkilemez** Bölüm 6.6 içindeki 7. ve 8. senaryolarda yitik iletilerin protokole etkileri tartışılmış ve sonuçları anlatılmıştır. Sözü edilmeyen 9. iletinin yitmesi durumu vardır, ancak bu ileti protokole işleyişi hızlandırmak üzere seçime bağlı olarak eklenmiştir. Yitmesi ancak işleyişi yavaşlatır.

6.7.2 Güvenli günlük tutma protokolünün biçimsel olarak doğrulanması

AVISPA [137] güvenlik protokollerinin doğrulanması alanında kullanılan bir araçtır. Role dayalı ortak bir protokol tanımlama dili[15] geliştirerek dört farklı biçimsel doğrulama yöntemini bir arada kullanma olanağı sunar. Bu yöntemler OFMC[16] [138], CL-AtSe[17] [139], SATMC[18] [140] ve TA4SP[19] [141] olarak sıralanır. Bunlar arasından SATMC ve TA4SP dışlayıcı VEYA işlemini desteklemedikleri için tasarlanan protokolün doğrulanmasında kullanılamaz.

AVISPA ancak *gizliliği* ya da *asıllamayı* doğrulayabilecek yetenektedir. Bunların ötesindeki yapılar doğrulanmak için karmaşıktır; göz ardı edilmeleri ya da doğrulanmak üzere bunlardan birine uyarlanmaları gerekmektedir. Doğrulama işlemleri için AVISPA HLPSL eğitim kılavuzunda [142] belirtildiğine göre zamansal mantık deyişleri üzerinde uzlaşmalara bakılmaktadır. Asıllama, iki tarafın diğer taraflardan gizli tutulan bir değerde uzlaşmaları durumunda oluşur. Bu tanım, Lowe'nin uzlaşma[20] tanımına da uygundur [143].

[15] "High Level Protocol Specification Language" adlı dil kısaca "HLPSL" olarak da anılır.
[16] On-the-fly Model-Checker
[17] Constraint-Logic-based Attack Searcher
[18] SAT-based Model-Checker
[19] Tree Automata based on Automatic Approximations for the Analysis of Security Protocols
[20] İngilizcesi: "agreement".

Tasarlanan protokolün bir asıllama gereksinimi yoktur. Ancak, asıllamanın ötesinde güvenlik gereksinimleri vardır ve bazıları yukarıda belirtilen uzlaşma tanımına uygun olarak çözülecektir. Benzetimin tamamlanması için gereken durum geçişleri uzlaşmanın gerçekleştiğinin kanıtı olarak kullanılacaktır.

Protokol bağlantısız tasarlanmıştır. Sonuç olarak, herhangi bir oturum yoktur ve asıllama gereksizdir. Protokolde yer alan her bir ileti gizliliği sağlayacak biçimde şifrelenmiştir. Ancak, bir günlük girisini oluşturan üç kaydın birbiriyle ilişkilendirilmesi için kullanılan rasgele sayıların kullanımı iki farklı kaynaktan gelen rasgele sayıların uzlaşmasını gerektirir. Bu, asıllamayı andırır. Bunun asıllamadan farkı, asıllamada uzlaşma için kullanılan sayılardan birinin yerelde bulunması, diğerinin asıllanan taraftan edinilmesidir. Protokolde üç taraf vardır ve uzlaşması gereken sayılar ilişkide olunan diğer iki taraftan gelmektedir. Böylece her üç tarafın gerçekleşen aynı olay üzerine kayıt tutmakta olduğundan emin olunur. Bu durumdaki iki değer $R_{\mathbb{KSB}}$ ve $R_{\mathbb{KS}}$ değerleridir. $R_{\mathbb{KSB}}$ değeri olağan akışta gereksizdir. Sadece uyuşmazlık çözümünde bildiri tahtasının bilgisine gerek duyulduğunda kayıtların tutarlılığını izlemek üzere kullanılır. $R_{\mathbb{KS}}$ değeri kullanıcı ve sağlayıcı arasında aynı giriyi izlemeyi kolaylaştırması için bulunur. Rasgele sayılar kullanıcı tarafından oluşturulur ve sağlayıcı ile bildiri tahtasına iki farklı yoldan ulaştırılır. Güvenlik aksamadıkça iki ayrı kanaldan ulaşan değerlerin uzlaşması beklenir. Benzetim, bu uzlaşma sağlanmadıkça durum geçişi olmayacak biçimde oluşturulmuştur.

Kayıtların gizliliği AVISPA ile doğrudan doğruya gösterilebilir. Benzetimde $\ddot{O}$ ve Y alanlarının gizliliği doğrulanır. Öneri ve yorum alanları kullanıcı ve sağlayıcı dışında biri tarafından görülmemelidir. Daha önce 6.4 bölümünde değinildiği gibi A tek başına veya M değeriyle birlikte $\ddot{O}$ hakkında bir bilgi vermez. Ancak A değerinin gizliliğini doğrulamak bir başka nedenle gereklidir.

Hizmete erişimi sağlayan iki önemli değer A ve M değerleridir. Bu iki değerin gizli tutulması hakkı olmayan birinin hizmete erişimini engeller. O nedenle A değeri sadece kullanıcı ve sağlayıcı tarafından bilinmelidir. M değeri bildiri tahtası ile de paylaşılacaktır ancak bildiri tahtasının bu değeri bilmesi ile A değerini oluşturması olanaksız olduğundan bu yolla hiçbir bilgi sızdırılamaz.

Benzetimin başarıyla tamamlanması anahtarların uygun biçimde ve uygun sırada kullanıldıklarının kanıtıdır. Ayrıca, durum geçişlerinin istenen biçimde gerçekleştiğinin ve beklenen değerlerin görüldüğünün de göstergesidir. Bu yaklaşım, önceden de belirtildiği gibi, rasgele sayıların uzlaştığının kanıtıdır.

Doğrulama için benzetim şu biçimde kurulmuştur: Kullanıcı (Şekil A.2), sağlayıcı (Şekil A.3) ve bildiri (tahtası) (Şekil A.4) ilgili durum geçişlerini de içeren roller olarak oluşturulmuştur. Bir ileti almak eğer belirtilen değerler uzlaşıyorsa olanaklıdır ve bu, durum değişikliğini tetikler. Durum değişiklikleri sırasında değişkenlerin değerleri değiştirilebilir ve başka rollere iletiler gönderilebilir.

Her üç rolde de değişken olarak ortak değerler görülür: Üç taraf (kullanıcı **k**, sağlayıcı **s** ve bildiri tahtası **b**), bir öz alma işlevi (**h**), tarafların açık anahtarları (**ak**, **as** ve **ab**) ve altı tek yönlü Dolev-Yao haberleşme kanalı (**ks**, **kb**, **sk**, **bk**, **bs** ve **sb**). Benzetimde doğrulama iki ayrı bağlamda (Şekil A.5) yapılmıştır. Burada sözü edilen değişkenler bağlamlar içinde tanımlandıktan sonra HLPSL gelenekleri uyarınca diğer rollerde referans olarak kullanılır ve bu sırada ilk harfleri büyük yazılarak anılır.

Üç rolü bir araya getiren oturumların nasıl tanımlanacağı Şekil A.6'da gösterilmiştir. Her oturum ilgili roller arasında gerçekleştirilen bir protokol işleyişini karşılar. İki oturumun eşzamanlı çalışması aynı anda iki olaya ilişkin günlük kayıtlarının tutulması anlamına geldiği gibi, bunun yapılması sırasında protokolün işleyişinde ortaya çıkabilecek tekrar saldırıları gibi olası aksaklıkları bulmaya da yardımcı olur. Bunu gösteren bağlam Şekil A.5a ile gösterilen bağlamdır. Bu bağlamda tanımlanmış olan saldırgan haberleşme kanallarını izleyebilir, bozabilir ve haberleşme kanalları üzerinden dilediği tarafa dilediği iletiyi gönderebilir.

Şekil A.5b'de tanımlanan bağlam ise üç oturum içermektedir. Saldırgan bu bağlamda önceki bağlamda sıralanan yeteneklerin ötesinde 6.5 bölümünde belirtildiği gibi sırayla üç role bürünmüş ve her bir rolde bulunmasının olası etkileri böylece denenmiştir.

Benzetimde kullanılan güvenlik hedeflerinin neler olduğu Şekil A.7'de belirtilmiştir. Hedefler, rasgele sayıların, önerinin, anlaşmanın, maskenin ve yorumun gizliliğidir.

Benzetimin çalışması sonunda, her iki yöntem de benzetim yapılan koşullar altında her iki bağlamda da verilen hedefler doğrultusunda protokolün güvenli olduğu sonucuna varmıştır. Elde edilen sonuçlar dürüst taraflardan oluşan iki oturumdan oluşan bağlam için Şekil A.8'de verilmiştir. Saldırganın üç farklı role bürünerek eşzamanlı üç ayrı oturumda protokolü çalıştırdığı bağlamın sonuçları Şekil A.9'da verilmiştir. Şekil A.9a'da görülen şaşırtıcı "gidilmiş düğümler" [21] ve "derinlik" [22] değerleri AVISPA'nın bilinen bir hatası [144] nedeniyle oluşmuştur ve sonucu değil sadece o alanların değerini etkilemektedir. OFMC yöntemi yaklaşık yirminci saniyede ağaç üzerindeki arama yordamını değiştirdiği için o andan sonra düğümleri ve derinliği saymayı bırakır. Burada sunulan arama gerçekte üç günden biraz daha uzun sürmüştür.

Karmaşık bir protokolün biçimsel doğrulanması sırasında karşılaşılan en önemli sınırlayıcı etken, kullanılan dile uygun soyutlamaları yapmaktaki zorluklardır. Bu yapılsa bile protokolün adımları arttıkça durum uzayındaki durum sayısının bir patlamayı andırır hızdaki yükselişi [23] uygulamadaki sınırı belirler [145, 146]. Benzetim sırasında protokol olduğu gibi HLPSL diline dönüştürüldüğünde, AVISPA gibi ardışıl işleyen bir programı çalıştırmak üzere Red Hat® Enterprise Linux® 6 Academic Edition işletim sistemi çalıştıran, 2.8 GHz frekanslı altı çekirdekli iki Intel® Xeon® X5660 merkezi işlem birimi olan 24 GB bellekli, günümüz için oldukça güçlü donanımlı bir bilgisayar bile, kullanılan doğrulama yöntemlerine göre farklılıklar göstermekle birlikte, protokolün ilk birkaç adımından sonra yetersiz bellek hatası vererek sonlanmaktadır. Bu durumun aşılması için önerilen protokol benzetim sırasında sınırlandırılmış ve 6.6 bölümünde 6. senaryoda sözü edilen güvenli haberleşme kanalı bildiri tahtasından özgün numaralar okunurken kullanılmıştır. Ayrıca, bildiri tahtasına sorgu gönderecek tarafların sayısı sınırlandırılmış, böylece durumların sayısının gereksiz yere artmasının önüne geçilmiştir. Saldırgan haberleşme kanalına her zaman ve herkes adına ileti ekleyebileceği için bununla benzetimde saldırganın davranışı kısıtlanmamıştır. Bu nedenle, bu değişikliğin olası bir saldırının benzetimde açığa çıkmasını engellemesi söz konusu değildir. Tasarlanan protokol ile benzetimde kullanılan protokolün farklarını gri renkle vurgulayan bir akış Şekil A.1'de sunulmuştur.

[21] İngilizcesi: "visited nodes".
[22] İngilizcesi: "depth".
[23] İngilizcesi: "state space explosion".

6.8 Güvenli Günlük Tutma Düzeneğinin Getireceği Fazladan Yük

Düzenek her günlük girisi için üç kayıt kadar saklama alanı gerektirir. Ayrıca bu kayıtlar hem düzeneğin güvenlik gerekleri nedeniyle hem de hata hoşgörüsünü ve yararlanırlığı artırmak için güvenli günlük tutma protokolüne katılan taraflara ve denetime katılacak gözlemci taraflara dağıtılır. Öte yandan, tarafların birbirine güvendiği bir senaryoda sadece öneri ve yorum alanının taraflardan birinin –olasılıkla sağlayıcının– bilgisayarında tutulması yeterli olur. Karşılaştırmak gerekirse, yer açısından getirilen fazladan yük, sadece öneri ve yorumu saklamak yerine üç kaydın üç ya da daha fazla kopyasını saklamak kadardır.

Güvenli günlük tutma protokolünün sunulduğu biçiminin hesaplama açısından getirdiği yükü hafifletmek olanaklıdır. Örneğin, bakışımlı şifreleme olanakları ya da oturum kullanan yaklaşımlar araştırılabilir. Protokolün Çizelge 6.2'de sunulduğu biçimiyle gerçeklenmesi durumunda işlemci için en yorucu adımların kriptolojik işlemler olacağı deneyimlere dayanarak söylenebilir. Protokolün zaman açısından getireceği fazladan yük incelenirken her günlük girisi için tarafların yapması gereken temel kriptolojik işlemlerin sayısına bakılmıştır.

Protokolün birinci adımından önce, $\mathit{İSTEK}$ hazırlanırken kullanıcı 2 rasgele sayı oluşturur, ardından 4 bakışımsız şifreleme yapar.

Birinci adımın hemen ardından bildiri tahtası iletinin tazeliğini incelemek üzere $\mathit{İSTEK}$in ikinci yarısının şifresini çözer. Bu sırada 2 bakışımsız şifre çözme işlemi gerçekleştirir.

İkinci adımda bildiri tahtası 1 öz alma, 1 bakışımsız şifreleme yaparak imza oluşturur.

İkinci adımdan sonra kullanıcı bildiri tahtasını okurken kaydın bütünlüğünü doğrulamak isteyecektir. Hatasız bir akış olduğu varsayılırsa kullanıcı 1 öz alma, 1 bakışımsız şifre çözme yapar.

Üçüncü adımda kullanıcı 2 bakışımsız şifreleme yaparken iletinin alıcısı olan sağlayıcı 2 bakışımsız şifre çözme yapar.

Üçüncü adımdan sonra sağlayıcı, bildiri tahtasındaki kaydı okur. Kaydın bütünlüğünü denetlemesi hatasız bir akışta 1 öz alma, 1 bakışımsız şifre çözme ile yapılır. Sağlayıcının kaydın içeriğini okuması için 2 bakışımsız şifre çözme daha yapması gerekir. Sağlayıcının *anlaşma*yı oluşturması için 1 rasgele sayı üretmesi, 1 kere öz alması ve 1 bakışımsız şifreleme yapması gerekir. *YANIT*ın oluşturulması için 4 bakışımsız şifreleme gerekir.

Dördüncü adımın hemen ardından bildiri tahtası iletinin tazeliğini ve *İSTEK* ile uyumunu incelemek üzere her iki kaydın ikinci yarısının şifresini çözer. Bu sırada 4 bakışımsız şifre çözme işlemi gerçekleştirir.

Beşinci adımda bildiri tahtası 1 öz alma, 1 bakışımsız şifreleme yaparak imza oluşturur.

Beşinci adımdan sonra sağlayıcı bildiri tahtasında kaydın yayınlandığından emin olmak ister. Akışta hata olmazsa bütünlüğünü doğrulamak için 1 öz alma, 1 bakışımsız şifre çözme yapar.

Altıncı adımda sağlayıcı 2 bakışımsız şifreleme yaparak iletiyi gönderir. Kullanıcı 2 bakışımsız şifre çözme yaparak iletinin içeriğini okur.

Altıncı adımın ardından kullanıcı bildiri tahtasına erişerek *YANIT*ı okur. Bütünlüğü doğrulamak 1 öz alma, 1 bakışımsız şifre çözme gerektirir. İçeriği okumak 2 şifre çözme gerektirir. İçerik okunduktan sonra anlaşmanın geçerliliğini anlamak üzere maske kullanılarak bütünlük denetimi yapılır. Bu denetimde de 1 kere öz alınır, 1 kere bakışımsız şifre çözülür.

Hizmet alındıktan sonra kullanıcı 4 bakışımsız şifreleme ile *ONAY* iletisini hazırlar.

Yedinci adımın hemen ardından bildiri tahtası iletinin tazeliğini ve *YANIT* ile uyumunu incelemek üzere her iki kaydın ikinci yarısının şifresini çözer. Bu sırada 4 bakışımsız şifre çözme işlemi gerçekleştirir.

Sekizinci adımda bildiri tahtası 1 öz alma, 1 bakışımsız şifreleme yaparak imza oluşturur.

Sekizinci adımdan sonra kullanıcı bildiri tahtasını gözleyerek kaydın yayınlandığını doğrulamak isteyecektir. Hatasız bir akış olduğu varsayılırsa kullanıcı 1 öz alma, 1 bakışımsız şifre çözme yapar.

Dokuzuncu adımda kullanıcı 2 bakışımsız şifreleme yaparak iletiyi gönderir. Sağlayıcı 2 bakışımsız şifre çözme yaparak iletinin içeriğini okur.

Dokuzuncu iletinin ardından bildiri tahtasındaki kaydı okuyan sağlayıcı kaydın bütünlüğünü doğrulamak için 1 öz alma, 1 bakışımsız şifre çözme yapar. İçeriği okumak için 2 bakışımsız şifre çözme gerekir.

Bu biçimde tamamlanan hatasız ve olağan akışı bozulmamış bir protokolde üç kayıttan oluşan her bir günlük girisi için toplamda: kullanıcının 2 rasgele sayı üretmesi, 4 öz alması, 8 bakışımsız şifre çözmesi, 12 bakışımsız şifreleme yapması; bildiri tahtasının 3 öz alması, 10 bakışımsız şifre çözmesi, 3 bakışımsız şifreleme yapması; sağlayıcının 1 rasgele sayı üretmesi, 4 öz alması, 11 bakışımsız şifre çözmesi, 7 bakışımsız şifreleme yapması gerekmektedir.

6.9 Vargı

Bu bölümde tasarlanan güvenli günlük düzeneği tanıtılmıştır. Düzenekte bağımsız bir bildiri tahtası günlük kayıtlarını saklamak üzere kullanılmaktadır. Bildiri tahtası silinemez, bozulmaya dayanıklı, tek bir yazım yapılabilen bir saklama alanı özelliklerindedir.

Düzenek kayıtların gizlilik içinde taşınmasını ve saklanmasını sağlar. Ayrıca, imzalar kullanılarak kayıtların yadsınamazlığı elde edilir ve herkesçe doğrulanabilir olma özelliği getirilir. Düzeneğin önemli bir yeniliği, kayıtları olaylar gerçekleşmeden önce kayıt altına almasıdır. Bunu, *ideal günlük tutma*daki haberleşme kanalının kopmayacağı ve kayıtların ulaşacağı günlük sunucusunda güvenle saklanacağı varsayımlarının yerine koyar. Kaydın ardından olayın gerçekleşmemesi olasılığına karşı da yorumların yazılacağı bir sonlandırma iletisi kullanılır. Protokolün iyimser adil değiş tokuş kullanması, birbirine tam güveni olmayan iki tarafın günlüklerin denetimini herhangi birine bırakmaya gerek duymadan düzeneği kullanmalarına olanak verir.

Düzeneğin var olan benzerlerinden farkları açıklanmış, getirdiği yenilikler karşılaştırılarak sıralanmıştır. Tehditlerin nasıl oluşabileceği ve düzeneğin bunları nasıl engellediği açıklanmıştır. Haberleşmedeki geçici kesintilerin ya da saldırıların ardından düzeneğin nasıl toparlanacağı belirtilmiştir. Sürekli saldırı ya da kesintiler kapsam dışı bırakılmışsa da yol gösterici çalışmalar belirtilmiştir.

Güvenli günlük protokolü, AVISPA ile biçimsel olarak çözümlenmiş, herhangi bir saldırı olasılığına rastlanmamış ve böylece doğrulanmıştır.

7. SONUÇ

Bu bölümde tez çalışması süresince tasarlanan düzenekler ve bu bağlamda bilime yapılan katkılar ortaya koyulmuştur. Ayrıca, sunulan düzeneklerin bir arada kullanımı üzerine bir tartışma ile tez sonlandırılmıştır.

7.1 Teze Genel Bakış

Tezin başlangıcında bulutun tanıtımı ve gerekli tanımlar yapılmıştır.

2. bölümde bulut kavramının ortaya çıkışı tarihçesi ve evrimiyle birlikte ele alınmış daha sonra bulutun güvenliği genel çerçevesi içinde anlatılmıştır. Bilgiişlem ortamı sunan bulutların tezin asıl alanı olduğu, tezin çıktısı olarak *bilgiişlem ortamı sunan bulut için güvenlik düzenekleri* tasarlanacağı belirtilmiştir.

3. bölümden başlanarak tezin odaklanmış olduğu bilgiişlem ortamı sunan bulutlar üzerinde durulmuştur. Bu bölümde özellikle var olan bulutlarla ilgilenilmiştir. İşlem gücünün izlekler aracılığıyla kullanıldığı bir bilgiişlem ortamı sunan bulutta müşteri programlarının nasıl yalıtılacağı üzerine tasarlanan düzenekler tanıtılmıştır. Tasarlanan düzeneklerin bir düzeye kadar başarılı olmalarına karşın izleklerin yapısı gereği çözümü güç sorunlarla karşılaşıldığı için ileriki aşamalarında izlekler yerine süreçlerin kullanıldığı bilgiişlem ortamı sunan bulutların önerilmesine karar verilmiştir.

4. bölümde işlem gücünün süreçler aracılığıyla kullanıldığı bilgiişlem ortamı sunan bulutun güvenliğini sağlamak üzere kullanılması gereken güvenlik düzeneklerinin tasarımı yapılmıştır. Önceki bölümlerde belirlenmiş olan güvenlik tehditlerine karşı alınacak önlemler tek tek sıralanmış ve bütünleştirilmiştir. Tasarım yenilikçi çözümleri de gerektirmiştir. Bu çözümlere tasarımda değinilmekle birlikte, öne sürülen düşünceler yeni oldukları için izleyen iki ayrı bölümde ayrıca açıklanmışlardır.

5. bölümde süreç tabanlı bilgiişlem ortamı sunan bulutlarda veri ve program yalıtımının sağlanmasına yönelik bir düzenek tasarlanmıştır. Bu tasarım birkaç farklı yaklaşımı bütünleştirir, böylece bilgiişlem ortamı sunan bulutlarda güvenlik için gereken koşulları bir arada sağlar. Bu tasarımda sağlananlar şöyle sıralanabilir: Müşteri veri ve programlarına yapılan kullanıcı erişimleri sağlayıcı tarafından denetlenebilmektedir. Ayrıca, erişim kurallarına, kullanıcıların erişimleri sırasında uyulması kriptolojik yollarla sağlanmıştır. Sağlayıcı müşteri programlarının kullanacakları kaynakları hizmet sözleşmesi kapsamında her kullanıcı için ayrı ayrı sınırlayabilmektedir. Bundan başka müşteri programları ve sağlayıcı konakları arasında bir standart arayüz oluşturulmuştur. Böylece müşteri programları sağlayıcı konakları üzerinde konumdan bağımsız çalışabilir, gerekirse yer değiştirebilirler.

6. bölümde bulutta gerçekleşen olayların kaydını yadsınamaz biçimde günlüğe işleyen bir düzeneğin tasarımı verilmiştir. Güvenli bir bilgiişlem ortamı sunan bulut tasarımı yapılmış olsa dahi zaman zaman taraflar arasında ortaya çıkacak uyuşmazlıkların çözümünde yol gösterici kanıtlar olan günlük kayıtları bu düzenek ile elde edilir. Bulut ortamında kimin denetiminde olduğu iyi tanımlanmamış olan günlük kayıtlarının birbirine tam güven duymayan iki tarafın ortaklaşa oluşturacakları, yadsınamaz ve silinemez günlük kayıtlarıyla nasıl aşılacağı bu bölümün asıl konusudur. Sunulan tasarım benzer güven sorunlarının ortaya çıktığı başka alanlarda da kullanılabilecek genelleştirilmiş bir çözümdür.

7.2 Tezin Gelişimi ve Bu Süreçte Literatüre Yapılan Katkılar

Tez önerisinin kabul edilmesi bulut güvenliğinin dikkat çekmeye başladığı yıllara rastlamaktadır. Böylece bulut güvenliği konusunda yapılan çalışmaların en başından beri yakından izlenmesi olanağı olmuştur. Tez çalışmasının başlamasıyla sürdürülen kapsamlı bir araştırma ile güvenlik sorunları ve kaynakları belirlenmiş, çözüm için tutulacak olası yollar tartışılmıştır. Yapılan çalışmalarda izlek tabanlı bilgiişlem ortamı sunan bulut için yapılan ilk kodlamalar ve denemelerin ardından izlekler yerine süreçlerin kullanılması yeğlenmiş, araştırmalara bu yönde devam edilmiştir.

Bu yönde sürdürülen araştırmaların ilk üretimi 2012 yılında "IEEE 31st Symposium on Reliable Distributed Systems" konferansında sunulan bildiri olmuştur [9]. Bu bildiride tezde sunulan süreç tabanlı bilgiişlem ortamı sunan bulut için güvenlik düzeneklerinden ilk kez söz edilir. Ancak bu yazıda düzeneklerin ayrıntıları tam değildir. Bu yayının içeriği tezin dördüncü bölümünün ana hatlarını oluşturur.

Buna paralel olarak "Yazılım Geliştirme Platformu Sunan Bir Bulut Sisteminin Operasyonel Eniyilemesinin Gerçeklenmesi"[1] adlı San-Tez projesi yürütülmüştür. Proje kapsamında kullanımda olan izlek tabanlı bir bilgiişlem ortamı sunan bulutun [65] güvenlik gerekleri araştırılmış ve sağlanmıştır. Bu çalışmanın sonucu olan güvenlik düzenekleri kullanımdaki bulut üzerinde çalışır duruma getirilerek başarımı ölçülmüştür. Bu konuda hazırlanan yayınımız 2014 yılında "IEEE Global Communications Conference (Globecom)" konferansında bildiri olarak sunulmuştur [10]. Bu yayının içeriği tezin üçüncü bölümü içinde anlatılmaktadır.

Süregiden çalışmalar dördüncü bölümde önerilen yaratıcı çözümlerin yayına hazırlanması ile sürdürülmüştür. Bilgiişlem ortamı sunan bulutlar için güvenli bir paradigma öneren, kullanıcı erişim kurallarını, müşterinin bulut kaynaklarına erişimini düzenleyen ve programların yalıtılmış çalışmasını sağlayan bir çalışma 2015 yılında "Proceedings of the Romanian Academy Series A: Mathematics, Physics, Technical Sciences, Information Science" dergisinde yayınlanmıştır [11]. Bu yayının içeriği tezin beşinci bölümü ile örtüşmektedir.

Son olarak, tezin altıncı bölümünde sunulan güvenli günlük tutma düzeneği yine 2015 yılında "IET Information Security" dergisinde yayınlanmıştır [12]. Bu düzenek birbirine tam güven duymayan iki tarafın bir hizmet alımı konusunda anlaşarak iyimser adil değiş tokuş protokolü çerçevesinde bu hizmet alımını yadsınamaz biçimde kayıt altına almaları üzerine kuruludur. Düzenek içinde tasarlanan protokolün biçimsel doğrulaması yapılmış ve herhangi bir güvenlik açığına rastlanmamıştır.

[1] T.C. Bilim, Sanayi ve Teknoloji Bakanlığı 0095.STZ.2013---1

7.3 Tartışma

Tez çalışmalarının başından başlayarak belirlenen zayıflıkların çözümünde izleklere yönelmenin yeterince yararlı olmayacağı düşünülmüştür. Bu düşüncenin yerinde olduğu daha sonra yürütülen San-Tez projesinde de belleğin yalıtımının başarılamamasıyla görülmüştür. Literatürdeki yaklaşımlar da görüşümüzü desteklemektedir. Sorunu çözme girişimleri izlekleri adım adım süreçlere benzetmekte, bir yerden sonra başarıma olumsuz etkileri olduğu görülerek geri adım atılmaktadır. İzlekler için belleği başarıyla yalıtan yaklaşım ise ancak süreç sanal makinasının üzerine bir sanal işletim sistemi kurmakla yapılabilmiştir. Bunların ötesinde, bulut için 1.1 bölümünde verilen tanımlar incelenirse süreç kullanmanın tanımı ağ hizmetleri arkasında izlek kullanmaktan daha iyi karşıladığı görülür.

Tezde öngörülen, süreçlerin kullanıldığı bilgiişlem ortamı sunan bulut yaklaşımının sonucunda hazırlanan güvenlik düzenekleriyle, müşteri uygulamalarının bulutta çalıştırılmaları sırasında gereksinim duyacakları program, veri ve ayarlar bir araya getirilerek süreç kozası adı verilen kavramsal bir bütün oluşturulur. Süreç kozasının bulutta kullanışlı biçimde çalıştırılması için gereken altyapının da tasarlanmasıyla yalıtılmış ve güvenli kozalar içinde üzerinde bulundukları konakların işletim sistemlerinin süreçleri olarak çalıştırılabilen programlar elde edilmiştir.

Kullanışlılık açısından bakılırsa, pazarda bulunan bilgiişlem ortamı sunan bulutlarda izlek tabanlı yaklaşım nedeniyle güçlükle yapılabilen ya da yapılamayan birçok iş süreç tabanlı yaklaşım sayesinde doğal olarak yapılabilir. Örneğin, programcı eşzamanlı işlem yürüten bir program yazabilir; kullanıcısına programın çalışma anında giriş, çıkış ve hata akımlarının denetimini verebilir, hatta bu akımlar üzerinden tasarlanan bir arayüz ile buluttaki programını etkileşimli kullanabilir. Bu yaklaşım süreçle doğrudan ve kalıcı iletişim kurmakta ve denetimi programı o anda kullanana bırakmaktadır, bu biçimiyle var olan bilgiişlem ortamı sunan bulutlardaki istemci/sunucu modelinin, diğer deyişle yapılan bir isteğe bir yanıtla karşılık verilmesi yaklaşımının oldukça uzağındadır. Süreçlerin tasarımda temel alınmış olması örgüde bilgiişlem yapılırken karşılaşılan toplu görevlerin de bu bulut ile başarılacağını akla getirir.

Bilgiişlem ortamı sunan bulutun güvenliğini sağlamak için pek çok düzenek bir arada kullanılmıştır. Gerektiğinde özlük bilgilerinin saklanacağı bir açık anahtar altyapısının var olduğu varsayılarak güvenli haberleşme kanallarının kurulabildiği, asıllama ve şifrelemenin yapılabildiği bir ortamda 5. bölümde ayrıntılarıyla anlatılan düzenekler kurulursa olası kötüye kullanımların sorumlularını belirlemek için yadsınamaz güvenli günlük tutma düzeneği de gerekecektir. Bu düzenek 6. bölümde sunularak bilgiişlem ortamı sunan bulutun güvenli olması için gerekenler eksik bırakılmamıştır. Güvenli günlük tutma düzeneği taraflar arasında oluşabilecek pek çok anlaşmazlığı karara bağlayacak kanıtları oluşturacak niteliktedir.

Bunun ötesinde, bulutta sonuçların hatalı oluşması veya bilgilerin yitirilmesi riskine karşı Bizans yetersayı kümeleri çözümü önerilmiştir. Ancak, 4. bölümde açıklandığı gibi, bu çözümün getirdiği fazladan parasal yük düşünülerek kullanımı müşterinin seçimine bırakılmıştır.

Şifrelenmiş veri üzerinde hesaplama yapılarak sağlayıcının müşteri verisini programın yürütülmesi sırasında da görmesini engelleyen kriptolojik yöntemler literatürde olmalarına karşın uygulamada kullanışlı olacak düzeyde başarımlı değildir. Benzer biçimde, yürütme sırasında algoritmayı sağlayıcıdan gizleyen kriptolojik yapılar da ancak teorik kalmaktadır. Veriyi ve algoritmayı gizleyen yapıların verimliliklerinin zamanla artmasıyla tasarlanan buluta eklenmesi beklenebilir. Tasarım bunun gerçekleşebileceği göz önüne alınarak bu konuda esnek bırakılmıştır. Müşteri, bu özelliklerde hazırladığı bir programı yanında şifrelenmiş veriyle birlikte süreç kozası içine yerleştirebilir. Bu özelliklerdeki program ve verinin hazırlanması işi ve bulutta gerçekleşecek hesaplamanın ardından şifrenin çözülmesi doğası gereği hiçbir zaman buluta taşınmaz. Bu nedenle, bu kriptolojik yöntemler için algoritma ve veri dönüştürücülerin tasarımı yapılacaksa dahi, bu, hiçbir zaman bulut tasarımı içinde yer almamalıdır.

Tasarım sırasında kullanılan kriptolojik araçların başarıma ve verimliliğe etkisi göz önüne alınmıştır ve uygulamada kullanışlı, olurluğu yüksek; bir başka deyişle uygun bir bedelle gerçeklenebilir düzeneklerin bir araya getirilmesine çalışılmıştır. Bu yönüyle, sunulan düzeneklerin işlerlik kazanması beklenir. İlk kez önerilen yenilikçi düzenekler çalışmalar sırasında kavramların yapılabilirliğinin gösterilmesi için model olarak kodlanmış durumdadır. Bilinen düzeneklerin yenilikçi düzeneklerle bir araya

getirilmesiyle süreçleri kullanan güvenli bir bilgiişlem ortamı sunan bulutun ortaya çıkması olasıdır. Bu yolla, 2.2.5 bölümünde söylendiği gibi bulut müşterilerinin en büyük çekincesi olan güvenlik konusuna bir çözüm getirilebilir.

7.4 Vargı

Tezde, işlem gücünün süreçler yoluyla kullanıldığı bilgiişlem ortamı sunan bulutlara ağırlık verilmekle birlikte, bulut kavramı ile birlikte ortaya çıkan buluta özgü güvenlik tehditleri belirlenmiş; bu tehditlere karşı yenilikçi önlemler geliştirilmiş; belirlenen tüm tehditleri bütünlük içinde önlemeye yönelik, olurluğu yüksek, kullanışlı *bilgiişlem ortamı sunan bulut için güvenlik düzenekleri* tasarlanmıştır.

KAYNAKLAR

[1] **Catteddu, D. ve Hogben, G.** (2009). *Cloud Computing: Benefits, risks and recommendations for information security*. (Teknik rapor). The European Network and Information Security Agency. Erişim adresi http://www.enisa.europa.eu/activities/risk-management/files/deliverables/cloud-computing-risk-assessment/at_download/fullReport.

[2] **Jaeger, T. ve Schiffman, J.** (2010). Outlook: Cloudy with a Chance of Security Challenges and Improvements. *IEEE Security & Privacy*, *8*(1), s. 77–80. doi:10.1109/MSP.2010.45.

[3] **Cloud Computing** (t.y.). *Gartner IT Glossary*. Erişim tarihi 30.01.2015, http://www.gartner.com/it-glossary/cloud-computing/.

[4] **Vaquero, L. M., Rodero-Merino, L., Caceres, J. ve Lindner, M.** (2008). A Break in the Clouds: Towards a Cloud Definition. *ACM SIGCOMM Computer Communication Review*, *39*(1), s. 50–55. doi:10.1145/1496091.1496100.

[5] **Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J. ve Brandic, I.** (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, *25*(6), s. 599–616. doi:10.1016/j.future.2008.12.001.

[6] **Mell, P. ve Grance, T.** (2011). *The NIST Definition of Cloud Computing*. (Teknik Rapor Special Publication 800-145). Gaithersburg, Maryland, ABD: U.S. Department of Commerce National Institute of Standards and Technology. Erişim adresi http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[7] **Voorsluys, W., Broberg, J. ve Buyya, R.** (2011). Introducing to cloud computing. İçinde *Cloud computing, Principles and Paradigms*, (Bölüm 1, s. 3–41). New Jersey: John Wiley and Sons, Inc.

[8] **Fowler, G. A. ve Worthen, B.** (2009, 26 Mart). The Internet Industry Is on a Cloud – Whatever That May Mean — Forget ASP and Web 2.0: Tech Companies Push Cirrus, Stratus, Other Cumulo-Nebulous Lingo. *The Wall Street Journal*, (s. A1, A10).

[9] **Sandıkkaya, M. T. ve Harmancı, A. E.** (2012). Security Problems of Platform-as-a-Service (PaaS) Clouds and Practical Solutions to the Problems. İçinde *2012 IEEE 31st Symposium on Reliable Distributed Systems (SRDS)*, (s. 463–468). IEEE Computer Society. doi:10.1109/SRDS.2012.84.

[10] **Sandıkkaya, M. T., Ödevci, B. ve Ovatman, T.** (2014). Practical Runtime Security Mechanisms for an aPaaS Cloud. İçinde *IEEE Global Communications Conference (Globecom 2014)*, (s. 53–58). IEEE Communications Society. doi:10.1109/GLOCOMW.2014.7063385.

[11] **Sandıkkaya, M. T. ve Harmancı, A. E.** (2015). A Security Paradigm for PaaS Clouds. *Proceedings of the Romanian Academy Series A: Mathematics, Physics, Technical Sciences, Information Science*, *16*(Special Issue), s. 345–356.

[12] **Sandıkkaya, M. T., Ovatman, T. ve Harmancı, A. E.** (2015). Design and formal verification of a cloud compliant secure logging mechanism. *IET Information Security*. Tamamlanmış çevrimiçi yayın. doi:10.1049/iet-ifs.2014.0625.

[13] **Corbató, F. J. ve Vyssotsky, V. A.** (1965). Introduction and Overview of the Multics System. İçinde *Proceedings of the AFIPS '65 November 30–December 1, 1965, Fall Joint Computer Conference, Part I*, (s. 185–196). New York, New York, ABD: ACM. doi:10.1145/1463891.1463912.

[14] **Project MAC** (t.y.). *Wikipedia*. Erişim tarihi 10.02.2015, https://en.wikipedia.org/wiki/Project_MAC.

[15] **MIT Computer Science and Artificial Intelligence Laboratory** (t.y.). *About CSAIL*. Erişim tarihi 10.02.2015, http://www.csail.mit.edu/about.

[16] **Time-Sharing** (t.y.). *Wikipedia*. Erişim tarihi 10.02.2015, https://en.wikipedia.org/wiki/Time-sharing.

[17] **Bemer, R. W.** (t.y.). *Origins of Time-Sharing*. Erişim tarihi 01.07.2015, http://www.bobbemer.com/TIMESHAR.HTM.

[18] **Bemer, R. W.** (1957). How to consider a computer. *Automatic Control Magazine*, *1957*(March), s. 66–69.

[19] **Corbató, F. J., Merwin-Daggett, M. ve Daley, R. C.** (1962). An Experimental Time-sharing System. İçinde *Proceedings of the AIEE-IRE '62, May 1-3, 1962, Spring Joint Computer Conference*, (s. 335–344). New York, New York, ABD: ACM. doi:10.1145/1460833.1460871.

[20] **Licklider, J. C. R.** (1963). *MEMORANDUM FOR: Members and Affiliates of the Intergalactic Computer Network*. Erişim tarihi 05.08.2013, http://worrydream.com/refs/Licklider-IntergalacticNetwork.pdf.

[21] **Madnick, S. E. ve Donovan, J. J.** (1973). Application and Analysis of the Virtual Machine Approach to Information System Security and Isolation. İçinde *Proceedings of the Workshop on Virtual Computer Systems*, (s. 210–224). New York, New York, ABD: ACM. doi:10.1145/800122.803961.

[22] **Liskov, B. ve Zilles, S.** (1974). Programming with Abstract Data Types. İçinde *Proceedings of the ACM SIGPLAN Symposium on Very High Level Languages*, (s. 50–59). New York, New York, ABD: ACM. doi:10.1145/800233.807045.

[23] **Becker, D. J., Sterling, T., Savarese, D., Dorband, J. E., Ranawake, U. A. ve Packer, C. V.** (1995). BEOWULF: A parallel workstation for scientific computation. İçinde P. Banerjee (Editör), *Proceedings of the 1995 International Conference on Parallel Processing*, (Cilt I Architecture, s. 11–14). CRC Press.

[24] **Foster, I.** (2002). *What is the Grid? A Three Point Checklist*. (Teknik rapor). Argonne National Laboratory & University of Chicago. Erişim adresi http://dlib.cs.odu.edu/WhatIsTheGrid.pdf.

[25] **MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F., Metz, R. ve Hamilton, B. A.** (2006). *OASIS Reference Model for Service Oriented Architecture 1.0*. (Teknik rapor). OASIS. Erişim adresi http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf.

[26] **Schmidt, E.** (2006). *Conversation with Eric Schmidt hosted by Danny Sullivan at Search Engine Strategies Conference*. Erişim tarihi 09.08.2015, http://www.google.com/press/podium/ses2006.html.

[27] **Amazon Web Services, Inc.** (2006). *Announcing Amazon Elastic Compute Cloud (Amazon EC2)*. Erişim tarihi 24.08.2015, https://aws.amazon.com/about-aws/whats-new/2006/08/24/announcing-amazon-elastic-compute-cloud-amazon-ec2---beta/.

[28] **Plummer, D. C., Bittman, T. J., Austin, T., Cearley, D. W. ve Smith, D. M.** (2008). *Cloud Computing: Defining and Describing an Emerging Phenomenon*. (Teknik Rapor G00156220). Gartner, Inc. Erişim adresi https://www.gartner.com/doc/697413.

[29] *Google Trends - Web Search interest: "cloud computing", "network security", "grid computing" - Worldwide, 2004 - present*. (t.y.). Erişim tarihi 24.02.2015, http://www.google.com/trends/explore#q=%22cloud%20computing%22%2C%20%22network%20security%22%2C%20%22grid%20computing%22.

[30] **Babcock, C.** (2010). *Management Strategies for the Cloud Revolution: How Cloud Computing Is Transforming Business and Why You Can't Afford to Be Left Behind* (1. Baskı). McGraw-Hill Education.

[31] **Adams, S.** (2012, 21 Ekim). *Dilbert*. Erişim adresi http://dilbert.com/strip/2012-10-21.

[32] **Anderson, E., Eschinger, C.,Wurster, L. F., De Silva, F., Contu, R., Liu, V. K., ... Pang, C.** (2012). *Forecast Overview: Public Cloud Services, Worldwide, 2011-2016, 2Q12 Update*. (Teknik Rapor G00234817). Gartner, Inc. Erişim adresi http://www.gartner.com/newsroom/id/2163616.

[33] **Plummer, D. C., Fiering, L., Dulaney, K., McGuire, M., Da Rold, C., Sarner, A., ... Welch, K.** (2014). *Top 10 Strategic Predictions for 2015 and Beyond: Digital Business Is Driving 'Big Change'*. (Teknik Rapor G00269904). Gartner, Inc. Erişim adresi https://www.gartner.com/doc/2864817.

[34] **Avrupa Komisyonu** (2012). *Unleashing the Potential of Cloud Computing in Europe - What is it and what does it mean for me?* Erişim tarihi 27.09.2015, http://europa.eu/rapid/press-release_MEMO-12-713_en.htm.

[35] **Xiao, Z. ve Xiao, Y.** (2013). Security and Privacy in Cloud Computing. *IEEE Communications Surveys Tutorials*, *15*(2), s. 843–859. doi:10.1109/SURV.2012.060912.00182.

[36] **Fernandes, D. A., Soares, L. F., Gomes, J. V., Freire, M. M. ve Inácio, P. R.** (2014). Security issues in cloud environments: a survey. *International Journal of Information Security*, *13*(2), s. 113–170. doi:10.1007/s10207-013-0208-7.

[37] **Johnson, B.** (2008, 29 Eylül). *Cloud computing is a trap, warns GNU founder Richard Stallman*. Erişim tarihi 29.09.2014, http://www.theguardian.com/technology/2008/sep/29/cloud.computing.richard.stallman.

[38] **Gens, F.** (2008). *IT Cloud Services User Survey, pt.2: Top Benefits & Challenges*. (Teknik rapor). International Data Corporation. Erişim adresi http://blogs.idc.com/ie/?p=210.

[39] **Gens, F.** (2009). *New IDC IT Cloud Services Survey: Top Benefits and Challenges*. (Teknik rapor). International Data Corporation. Erişim adresi http://blogs.idc.com/ie/?p=730.

[40] **Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., ... Zaharia, M.** (2009). *Above the Clouds: A Berkeley View of Cloud Computing*. (Teknik Rapor UCB/EECS-2009-28). University of California at Berkeley. Erişim adresi http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html.

[41] *Google Scholar*. (t.y.). Erişim tarihi 27.02.2015, http://scholar.google.com/.

[42] **Rodero-Merino, L., Vaquero, L. M., Caron, E., Muresan, A. ve Desprez, F.** (2012). Building safe PaaS clouds: A survey on security in multitenant software platforms. *Computers & Security*, *31*(1), s. 96–108. doi:10.1016/j.cose.2011.10.006.

[43] **Gollmann, D.** (2006). Why Trust is Bad for Security. *Electronic Notes in Theoretical Computer Science*, *157*(3), s. 3–9. doi:10.1016/j.entcs.2005.09.044.

[44] **Zhou, M., Zhang, R., Xie, W., Qian, W. ve Zhou, A.** (2010). Security and Privacy in Cloud Computing: A Survey. İçinde *Sixth International Conference on Semantics Knowledge and Grid (SKG)*, (s. 105–112). IEEE Computer Society. doi:10.1109/SKG.2010.19.

[45] **Vaquero, L. M., Rodero-Merino, L. ve Morán, D.** (2011). Locking the sky: a survey on IaaS cloud security. *Computing*, *91*(1), s. 93–118. doi:10.1007/s00607-010-0140-x.

[46] **Subashini, S. ve Kavitha, V.** (2011). A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications*, *34*(1), s. 1–11. doi:10.1016/j.jnca.2010.07.006.

[47] **Ahuja, S. P. ve Komathukattil, D.** (2012). A Survey of the State of Cloud Security. *Network and Communication Technologies*, *1*(2), s. 66–75. doi:10.5539/nct.v1n2p66.

[48] **Aguiar, E., Zhang, Y. ve Blanton, M.** (2014). An Overview of Issues and Recent Developments in Cloud Computing and Storage Security. İçinde K. J. Han, B.-Y. Choi ve S. Song (Editörler), *High Performance Cloud Auditing and Applications*, (s. 3–33). Springer New York. doi:10.1007/978-1-4614-3296-8_1.

[49] **Pearce, M., Zeadally, S. ve Hunt, R.** (2013). Virtualization: Issues, Security Threats, and Solutions. *ACM Computing Surveys*, *45*(2), s. 17:1–17:39. doi:10.1145/2431211.2431216.

[50] **Pearson, S.** (2013). Privacy, Security and Trust in Cloud Computing. İçinde S. Pearson ve G. Yee (Editörler), *Privacy and Security for Cloud Computing*, (s. 3–42). Springer London. doi:10.1007/978-1-4471-4189-1_1.

[51] **Grobauer, B., Walloschek, T. ve Stocker, E.** (2011). Understanding Cloud Computing Vulnerabilities. *IEEE Security Privacy*, *9*(2), s. 50–57. doi:10.1109/MSP.2010.115.

[52] **Chen, Y., Paxson, V. ve Katz, R. H.** (2010). *What's New About Cloud Computing Security?* (Teknik Rapor UCB/EECS-2010-5). EECS Department, University of California, Berkeley. Erişim adresi http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-5.html.

[53] **Dahbur, K., Mohammad, B. ve Tarakji, A. B.** (2011). A Survey of Risks, Threats and Vulnerabilities in Cloud Computing. İçinde *Proceedings of the 2011 International Conference on Intelligent Semantic Web-Services and Applications (ISWSA '11)*, (s. 12:1–12:6). New York, New York, ABD: ACM. doi:10.1145/1980822.1980834.

[54] **Mathisen, E.** (2011). Security challenges and solutions in cloud computing. İçinde *Proceedings of the 5th IEEE International Conference on Digital Ecosystems and Technologies Conference (DEST)*, (s. 208–212). doi:10.1109/DEST.2011.5936627.

[55] **Takabi, H., Joshi, J. B. D. ve Ahn, G.-J.** (2010). Security and Privacy Challenges in Cloud Computing Environments. *IEEE Security and Privacy*, *8*(6), s. 24–31. doi:10.1109/MSP.2010.186.

[56] **Dupré, L. ve Haeberlen, T.** (2012). *Cloud Computing: Benefits, risks and recommendations for information security*. (Teknik Rapor Rev. B). The European Network and Information Security Agency. Erişim adresi https://resilience.enisa.europa.eu/cloud-security-and-resilience/publications/cloud-computing-benefits-risks-and-recommendations-for-information-security.

[57] **Reavis, J.** (2009). *Security Guidance for Critical Areas of Focus in Cloud Computing*. (Teknik Rapor V1.0). Cloud Security Alliance. Erişim adresi http://www.cloudsecurityalliance.org/guidance/csaguide.v1.0.pdf.

[58] **Brunette, G. ve Mogull, R.** (2009). *Security Guidance for Critical Areas of Focus in Cloud Computing*. (Teknik Rapor V2.1). Cloud Security Alliance. Erişim adresi http://www.cloudsecurityalliance.org/guidance/csaguide.v2.1.pdf.

[59] **Simmonds, P., Rezek, C. ve Reed, A.** (2011). *Security Guidance for Critical Areas of Focus in Cloud Computing*. (Teknik Rapor V3.0). Cloud Security Alliance. Erişim adresi https://downloads.cloudsecurityalliance.org/initiatives/guidance/csaguide.v3.0.pdf.

[60] **Los, R., Shackleford, D. ve Sullivan, B.** (2013). *The Notorious Nine: Cloud Computing Top Threats in 2013*. (Teknik rapor). Cloud Security Alliance. Erişim adresi https://downloads.cloudsecurityalliance.org/initiatives/top_threats/The_Notorious_Nine_Cloud_Computing_Top_Threats_in_2013.pdf.

[61] **Open Security Foundation** (t.y.). *OSVDB: Open Sourced Vulnerability Database*. Erişim tarihi 09.03.2015, http://osvdb.org/.

[62] **He, S., Guo, L., Guo, Y., Wu, C., Ghanem, M. ve Han, R.** (2012). Elastic Application Container: A Lightweight Approach for Cloud Resource Provisioning. İçinde *IEEE 26th International Conference on Advanced Information Networking and Applications (AINA)*, (s. 15–22). doi:10.1109/AINA.2012.74.

[63] **O'Sullivan, B.** (1996). *The history of threads*. Erişim tarihi 13.08.2015, http://www.faqs.org/faqs/os-research/part1/section-10.html.

[64] **Tanenbaum, A. S. ve Bos, H.** (2014). *Modern Operating Systems* (4. Baskı). New York, New York, ABD: Pearson.

[65] *Imona Cloud*. (t.y.). Erişim tarihi 17.12.2014, https://www.imona.com/.

[66] **Gosling, J., Joy, B., Steele, G., Bracha, G. ve Buckley, A.** (2010). *The Java® Language Specification Java SE 8 Edition*. (Teknik rapor). Redwood City, Kaliforniya, ABD: Oracle Corporation. Erişim adresi http://docs.oracle.com/javase/specs/jls/se8/jls8.pdf.

[67] **Bowles, K. L.** (1978). A (Nearly) Machine Independent Software System for Micro and Mini Computers. *SIGMINI Newsletters*, *4*(1), s. 3–7. doi:10.1145/1041256.1041257.

[68] **Ecma** (2012). *Common Language Infrastructure (CLI) 6th edition*. (Standart ECMA 335 ve ISO/IEC 23271:2012(E)). Cenevre, İsviçre: Ecma International - European association for standardizing information and communication systems.

[69] **Gong, L.** (2002). *Java SE Platform Security Architecture Specification v1.2*. (Teknik rapor). Redwood City, Kaliforniya, ABD: Oracle Corporation. Erişim adresi http://docs.oracle.com/javase/8/docs/technotes/guides/security/spec/security-spec.doc.html.

[70] **Lindholm, T., Yellin, F., Bracha, G. ve Buckley, A.** (2010). *The Java® Virtual Machine Specification Java SE 8 Edition*. (Teknik rapor). Redwood City, Kaliforniya, ABD: Oracle Corporation. Erişim adresi http://docs.oracle.com/javase/specs/jvms/se8/jvms8.pdf.

[71] **Sun, K., Li, Y., Hogstrom, M. ve Chen, Y.** (2006). Sizing Multi-space in Heap for Application Isolation. İçinde *Companion to the 21st ACM SIGPLAN Symposium on Object-oriented Programming Systems, Languages, and Applications (OOPSLA '06)*, (s. 647–648). New York, New York, ABD: ACM. doi:10.1145/1176617.1176654.

[72] **Back, G. ve Hsieh, W. C.** (2005). The KaffeOS Java Runtime System. *ACM Transactions on Programming Languages and Systems*, *27*(4), s. 583–630. doi:10.1145/1075382.1075383.

[73] **Demichiel, L. ve Keith, M.** (2007). *JSR 220: Enterprise JavaBeans 3.0*. Erişim tarihi 01.04.2015, https://jcp.org/en/jsr/detail?id=220.

[74] **Mordani, R.** (2007). *JSR 154: Java Servlet 2.4 Specification*. Erişim tarihi 01.04.2015, https://jcp.org/en/jsr/detail?id=154.

[75] *App Engine - Platform as a Service — Google Cloud Platform*. (t.y.). Erişim tarihi 18.03.2015, https://cloud.google.com/appengine/.

[76] *AWS Elastic Beanstalk - Application Management - Platform as a Service*. (t.y.). Erişim tarihi 10.05.2015, https://aws.amazon.com/elasticbeanstalk/.

[77] *Heroku | Cloud Application Platform*. (t.y.). Erişim tarihi 10.05.2015, https://www.heroku.com/.

[78] *Apache Stratos*. (t.y.). Erişim tarihi 10.05.2015, https://stratos.apache.org/.

[79] **Czajkowski, G. ve Daynés, L.** (2001). Multitasking Without Comprimise: A Virtual Machine Evolution. İçinde *Proceedings of the 16th ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications (OOPSLA '01)*, (s. 125–138). New York, New York, ABD: ACM. doi:10.1145/504282.504292.

[80] **Palacz, K.** (2006). *JSR 121: Application Isolation API Specification*. Erişim tarihi 01.04.2015, https://jcp.org/en/jsr/detail?id=121.

[81] **Geoffray, N., Thomas, G., Muller, G., Parrend, P., Frénot, S. ve Folliot, B.** (2009). I-JVM: a Java Virtual Machine for component isolation in OSGi. İçinde *IEEE/IFIP International Conference on Dependable Systems Networks (DSN '09)*, (s. 544–553). doi:10.1109/DSN.2009.5270296.

[82] **Lamport, L., Shostak, R. ve Pease, M.** (1982). The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, *4*(3), s. 382–401. doi:10.1145/357172.357176.

[83] **Lamport, L.** (1998). The Part-time Parliament. *ACM Transactions on Computer Systems (TOCS)*, *16*(2), s. 133–169. doi:10.1145/279227.279229.

[84] **Leopando, J.** (2013). *World Backup Day: The 3-2-1 Rule*. Erişim tarihi 02.04.2015, http://blog.trendmicro.com/trendlabs-security-intelligence/world-backup-day-the-3-2-1-rule/.

[85] **Yin, J., Martin, J.-P., Venkataramani, A., Alvisi, L. ve Dahlin, M.** (2003). Separating Agreement from Execution for Byzantine Fault Tolerant Services. *ACM SIGOPS Operating Systems Review*, *37*(5), s. 253–267. doi:10.1145/1165389.945470.

[86] **Lim, J., Suh, T., Gil, J. ve Yu, H.** (2014). Scalable and leaderless Byzantine consensus in cloud computing environments. *Information Systems Frontiers*, *16*(1), s. 19–34. doi:10.1007/s10796-013-9460-7.

[87] **Veronese, G., Correia, M., Bessani, A., Lung, L. C. ve Verissimo, P.** (2013). Efficient Byzantine Fault-Tolerance. *IEEE Transactions on Computers*, *62*(1), s. 16–30. doi:10.1109/TC.2011.221.

[88] **Tuecke, S., Welch, V., Engert, D., Pearlman, L. ve Thompson, M.** (2004). *Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile*. (Request for Comments 3820). Fremont, Kaliforniya, ABD: Internet Engineering Task Force. Erişim adresi http://www.ietf.org/rfc/rfc3820.txt.

[89] **Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R. ve Polk, T.** (2008). *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. (Request for Comments 5280). Fremont, Kaliforniya, ABD: Internet Engineering Task Force. Erişim adresi http://www.ietf.org/rfc/rfc5280.txt.

[90] **Dierks, T. ve Rescorla, E.** (2008). *The transport layer security (TLS) protocol version 1.2*. (Request for Comments 5246). Fremont, Kaliforniya, ABD: Internet Engineering Task Force. Erişim adresi http://www.ietf.org/rfc/rfc5246.txt.

[91] **Hickman, K. E. B.** (1995). *The SSL Protocol*. (Internet draft). Mountain View, Kaliforniya, ABD: Netscape Communications Corporation. Erişim adresi https://tools.ietf.org/html/draft-hickman-netscape-ssl-00.

[92] **Brand, S. L.** (1985). *Department of Defense Trusted Computer System Evaluation Criteria*. (Department of Defense Standard DoD 5200.28-STD). U.S. Department of Defense.

[93] **Mont, M. C., Pearson, S. ve Bramhall, P.** (2003). Towards accountable management of identity and privacy: sticky policies and enforceable tracing services. İçinde *Proceedings of 14th International Workshop on Database and Expert Systems Applications*, (s. 377–382). doi:10.1109/DEXA.2003.1232051.

[94] **Bethencourt, J., Sahai, A. ve Waters, B.** (2007). Ciphertext-Policy Attribute-Based Encryption. İçinde *IEEE Symposium on Security and Privacy (SP '07)*, (s. 321–334). IEEE Computer Society. doi:10.1109/SP.2007.11.

[95] **Menezes, A. J., Van Oorschot, P. C. ve Vanstone, S. A.** (1996). *Handbook of Applied Cryptography* (1. Baskı). Boca Raton, Florida, ABD: CRC press.

[96] **Paillier, P.** (1999). Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. İçinde J. Stern (Editör), *Advances in Cryptology — EUROCRYPT '99*, (Cilt 1592, s. 223–238). Springer Berlin Heidelberg. doi:10.1007/3-540-48910-X_16.

[97] **ElGamal, T.** (1985). A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. İçinde G. Blakley ve D. Chaum (Editörler), *Advances in Cryptology*, (Cilt 196, s. 10–18). Springer Berlin Heidelberg. doi:10.1007/3-540-39568-7_2.

[98] **Gentry, C.** (2009). Fully Homomorphic Encryption Using Ideal Lattices. İçinde *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing (STOC '09)*, (s. 169–178). New York, New York, ABD: ACM. doi:10.1145/1536414.1536440.

[99] **Gentry, C., Halevi, S. ve Smart, N. P.** (2012). Homomorphic Evaluation of the AES Circuit. İçinde R. Safavi-Naini ve R. Canetti (Editörler), *Advances in Cryptology — CRYPTO 2012*, (Cilt 7417, s. 850–867). Springer Berlin Heidelberg. doi:10.1007/978-3-642-32009-5_49.

[100] **Fletcher, C. W., van Dijk, M. ve Devadas, S.** (2012). Towards an Interpreter for Efficient Encrypted Computation. İçinde *Proceedings of the 2012 ACM Workshop on Cloud Computing Security Workshop (CCSW '12)*, (s. 83–94). New York, New York, ABD: ACM. doi:10.1145/2381913.2381928.

[101] **Brenner, M., Wiebelitz, J., von Voigt, G. ve Smith, M.** (2011). Secret program execution in the cloud applying homomorphic encryption. İçinde *Proceedings of the 5th IEEE International Conference on Digital Ecosystems and Technologies Conference (DEST)*, (s. 114–119). doi:10.1109/DEST.2011.5936608.

[102] **Zhuravlev, D., Samoilovych, I., Orlovskyi, R., Bondarenko, I. ve Lavrenyuk, Y.** (2014). Encrypted Program Execution. İçinde *IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, (s. 817–822). doi:10.1109/TrustCom.2014.108.

[103] **ISO/IEC** (2011). *Information technology – Programming languages – C*. (Standart ISO/IEC 9899:2011). Cenevre, İsviçre: ISO (the International Organization for Standardization) ve IEC (the International Electrotechnical Commission).

[104] **Goldreich, O.** (1987). Towards a Theory of Software Protection and Simulation by Oblivious RAMs. İçinde *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing (STOC '87)*, (s. 182–194). New York, New York, ABD: ACM. doi:10.1145/28395.28416.

[105] **Fletcher, C. W., van Dijk, M. ve Devadas, S.** (2012). A Secure Processor Architecture for Encrypted Computation on Untrusted Programs. İçinde *Proceedings of the Seventh ACM Workshop on Scalable Trusted Computing (STC '12)*, (s. 3–8). New York, New York, ABD: ACM. doi:10.1145/2382536.2382540.

[106] **Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A. ve Waters, B.** (2013). Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits. İçinde *IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, (s. 40–49). doi:10.1109/FOCS.2013.13.

[107] **Swanson, M. ve Guttman, B.** (1996). *Generally Accepted Principles and Practices for Securing Information Technology Systems*. (NIST Special Publication 800-14). Gaithersburg, Maryland, ABD: U.S. Department of Commerce, National Institute of Standards and Technology, Technology Administration.

[108] **Bellare, M. ve Yee, B. S.** (1997). *Forward Integrity For Secure Audit Logs*. (Teknik rapor). Computer Science and Engineering Department, University of California at San Diego.

[109] **Yavuz, A. A., Ning, P. ve Reiter, M. K.** (2012). Efficient, Compromise Resilient and Append-Only Cryptographic Schemes for Secure Audit Logging. İçinde A. D. Keromytis (Editör), *16th International Conference on Financial Cryptography and Data Security*, (Cilt 7397, s. 148–163). Springer Berlin Heidelberg. doi:10.1007/978-3-642-32946-3_12.

[110] **Schneier, B. ve Kelsey, J.** (1999). Secure Audit Logs to Support Computer Forensics. *ACM Transactions on Information and System Security (TISSEC)*, *2*(2), s. 159–176. doi:10.1145/317087.317089.

[111] **Holt, J. E.** (2006). Logcrypt: Forward Security and Public Verification for Secure Audit Logs. İçinde *Proceedings of the 2006 Australasian Workshops on Grid Computing and e-Research – Volume 54 (ACSW Frontiers '06)*, (s. 203–211). Australian Computer Society, Inc.

[112] **Ma, D. ve Tsudik, G.** (2009). A New Approach to Secure Logging. *ACM Transactions on Storage (TOS)*, *5*(1), s. 2:1–2:21. doi:10.1145/1502777.1502779.

[113] **Syslog** (t.y.). *Wikipedia*. Erişim tarihi 02.04.2015, https://en.wikipedia.org/wiki/Syslog.

[114] **Lonvick, C.** (2001). *The BSD syslog protocol*. (Request for Comments 3164). Fremont, Kaliforniya, ABD: Internet Engineering Task Force. Erişim adresi http://www.ietf.org/rfc/rfc3164.txt.

[115] **Postel, J.** (1980). *User Datagram Protocol*. (Request for Comments 768). Fremont, Kaliforniya, ABD: Internet Engineering Task Force. Erişim adresi http://www.ietf.org/rfc/rfc768.txt.

[116] **Gerhards, R.** (2009). *The Syslog Protocol*. (Request for Comments 5424). Fremont, Kaliforniya, ABD: Internet Engineering Task Force. Erişim adresi http://www.ietf.org/rfc/rfc5424.txt.

[117] **Okmianski, A.** (2009). *Transmission of Syslog Messages over UDP*. (Request for Comments 5426). Fremont, Kaliforniya, ABD: Internet Engineering Task Force. Erişim adresi http://www.ietf.org/rfc/rfc5426.txt.

[118] **New, D. ve Rose, M. T.** (2001). *Reliable delivery for syslog*. (Request for Comments 3195). Fremont, Kaliforniya, ABD: Internet Engineering Task Force. Erişim adresi http://www.ietf.org/rfc/rfc3195.txt.

[119] **Postel, J.** (1981). *Transmission control protocol*. (Request for Comments 793). Fremont, Kaliforniya, ABD: Internet Engineering Task Force. Erişim adresi http://www.ietf.org/rfc/rfc793.txt.

[120] **Gerhards, R. ve Lonvick, C.** (2012). *Transmission of Syslog Messages over TCP*. (Request for Comments 6587). Fremont, Kaliforniya, ABD: Internet Engineering Task Force. Erişim adresi http://www.ietf.org/rfc/rfc6587.txt.

[121] **Kelsey, J., Callas, J. ve Clemm, A.** (2010). *Signed syslog messages*. (Request for Comments 5848). Fremont, Kaliforniya, ABD: Internet Engineering Task Force. Erişim adresi http://www.ietf.org/rfc/rfc5848.txt.

[122] **Rescorla, E. ve Modadugu, N.** (2006). *Datagram Transport Layer Security*. (Request for Comments 4347). Fremont, Kaliforniya, ABD: Internet Engineering Task Force. Erişim adresi http://www.ietf.org/rfc/rfc4347.txt.

[123] **Miao, F., Ma, Y. ve Salowey, J.** (2009). *Transport layer security (TLS) transport mapping for Syslog*. (Request for Comments 5425). Fremont, Kaliforniya, ABD: Internet Engineering Task Force. Erişim adresi http://www.ietf.org/rfc/rfc5425.txt.

[124] **Salowey, J., Petch, T., Gerhards, R. ve Feng, H.** (2010). *Datagram Transport Layer Security (DTLS) Transport Mapping for Syslog*. (Request for Comments 6012). Fremont, Kaliforniya, ABD: Internet Engineering Task Force. Erişim adresi http://www.ietf.org/rfc/rfc6012.txt.

[125] **Chong, C. N., Peng, Z. ve Hartel, P. H.** (2003). Secure Audit Logging with Tamper-Resistant Hardware. İçinde D. Gritzalis, S. De Capitani di Vimercati, P. Samarati ve S. Katsikas (Editörler), *Security and Privacy in the Age of Uncertainty IFIP TC11 18th International Conference on Information Security (SEC 2003), May 26-28, 2003, Athens, Greece*, (Cilt 122, s. 73–84). Springer US. doi:10.1007/978-0-387-35691-4_7.

[126] **Accorsi, R.** (2011). BBox: A Distributed Secure Log Architecture. İçinde J. Camenisch ve C. Lambrinoudakis (Editörler), *8th European Workshop on Public Key Infrastructures, Services and Applications*, (Cilt 6711, s. 109–124). Springer Berlin Heidelberg. doi:10.1007/978-3-642-22633-5_8.

[127] **Ray, I., Belyaev, K., Strizhov, M., Mulamba, D. ve Rajaram, M.** (2013). Secure Logging as a Service —Delegating Log Management to the Cloud. *IEEE Systems Journal*, *7*(2), s. 323–334. doi:10.1109/JSYST.2012.2221958.

[128] **Zawoad, S., Dutta, A. K. ve Hasan, R.** (2013). SecLaaS: Secure Logging-as-a-service for Cloud Forensics. İçinde *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security (ASIA CCS '13)*, (s. 219–230). ACM. doi:10.1145/2484313.2484342.

[129] **Asokan, N., Schunter, M. ve Waidner, M.** (1997). Optimistic Protocols for Fair Exchange. İçinde *Proceedings of the 4th ACM Conference on Computer and Communications Security (CCS '97)*, (s. 7–17). ACM. doi:10.1145/266420.266426.

[130] **Dolev, D. ve Yao, A. C.** (1983). On the security of public key protocols. *IEEE Transactions on Information Theory*, *29*(2), s. 198–208. doi:10.1109/TIT.1983.1056650.

[131] **von Ahn, L., Blum, M., Hopper, N. J. ve Langford, J.** (2003). CAPTCHA: Using Hard AI Problems for Security. İçinde E. Biham (Editör), *Advances in Cryptology – EUROCRYPT 2003*, (Cilt 2656, s. 294–311). Springer Berlin Heidelberg. doi:10.1007/3-540-39200-9_18.

[132] **Dingledine, R., Mathewson, N. ve Syverson, P.** (2004). Tor: The Second-Generation Onion Router. İçinde *13th USENIX Security Symposium*, (s. 303–320).

[133] **Drielsma, P. H. ve Mödersheim, S.** (2005). The ASW Protocol Revisited: A Unified View. *Electronic Notes in Theoretical Computer Science*, *125*(1), s. 145–161. doi:10.1016/j.entcs.2004.05.024.

[134] **Asokan, N., Shoup, V. ve Waidner, M.** (1998). Asynchronous Protocols for Optimistic Fair Exchange. İçinde *Proceedings of 1998 IEEE Symposium on Security and Privacy*, (s. 86–99). IEEE. doi:10.1109/SECPRI.1998.674826.

[135] **Pnueli, A.** (1977). The temporal logic of programs. İçinde *18th Annual Symposium on Foundations of Computer Science, 1977.*, (s. 46–57). doi:10.1109/SFCS.1977.32.

[136] **Clarke, E. M., Grumberg, O. ve Peled, D. A.** (1999). *Model checking*. Cambridge, Massachusetts, ABD: MIT Press.

[137] **Armando, A., Basin, D., Boichut, Y., Chevalier, Y., Compagna, L., Cuellar, J., ... Vigneron, L.** (2005). The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. İçinde K. Etessami ve S. K. Rajamani (Editörler), *17th International Conference on Computer Aided Verification*, (Cilt 3576, s. 281–285). Springer Berlin Heidelberg. doi:10.1007/11513988_27.

[138] **Basin, D., Mödersheim, S. ve Viganò, L.** (2005). OFMC: A symbolic model checker for security protocols. *International Journal of Information Security*, *4*(3), s. 181–208. doi:10.1007/s10207-004-0055-7.

[139] **Chevalier, Y. ve Vigneron, L.** (2002). Automated Unbounded Verification of Security Protocols. İçinde E. Brinksma ve K. G. Larsen (Editörler), *14th International Conference on Computer Aided Verification*, (Cilt 2404, s. 324–337). Springer Berlin Heidelberg. doi:10.1007/3-540-45657-0_24.

[140] **Armando, A. ve Compagna, L.** (2004). SATMC: A SAT-Based Model Checker for Security Protocols. İçinde J. J. Alferes ve J. Leite (Editörler), *9th European Conference on Logics in Artificial Intelligence*, (Cilt 3229, s. 730–733). Springer Berlin Heidelberg. doi:10.1007/978-3-540-30227-8_68.

[141] **Boichut, Y., Héam, P.-C., Kouchnarenko, O. ve Oehl, F.** (2004). Improvements on the Genet and Klay technique to automatically verify security protocols. İçinde *Proceedings of International Workshop on Automated Verification of Infinite-State Systems (AVIS'2004)*, (s. 1–11).

[142] **The AVISPA team** (2006). HLPSL Tutorial A Beginner's Guide to Modelling and Analysing Internet Security Protocols (1.1 baskı) [Bilgisayar yazılımı kılavuzu]. http://www.avispa-project.org/package/tutorial.pdf.

[143] **Lowe, G.** (1997). A hierarchy of authentication specifications. İçinde *Proceedings of 10th Computer Security Foundations Workshop*, (s. 31–43). IEEE. doi:10.1109/CSFW.1997.596782.

[144] **Moedersheim, S. A.** (2006). *[Avispa-users] Help please*. Erişim tarihi 17.05.2015, http://www.avispa-project.org/avispa-users-old/2006-May/000236.html.

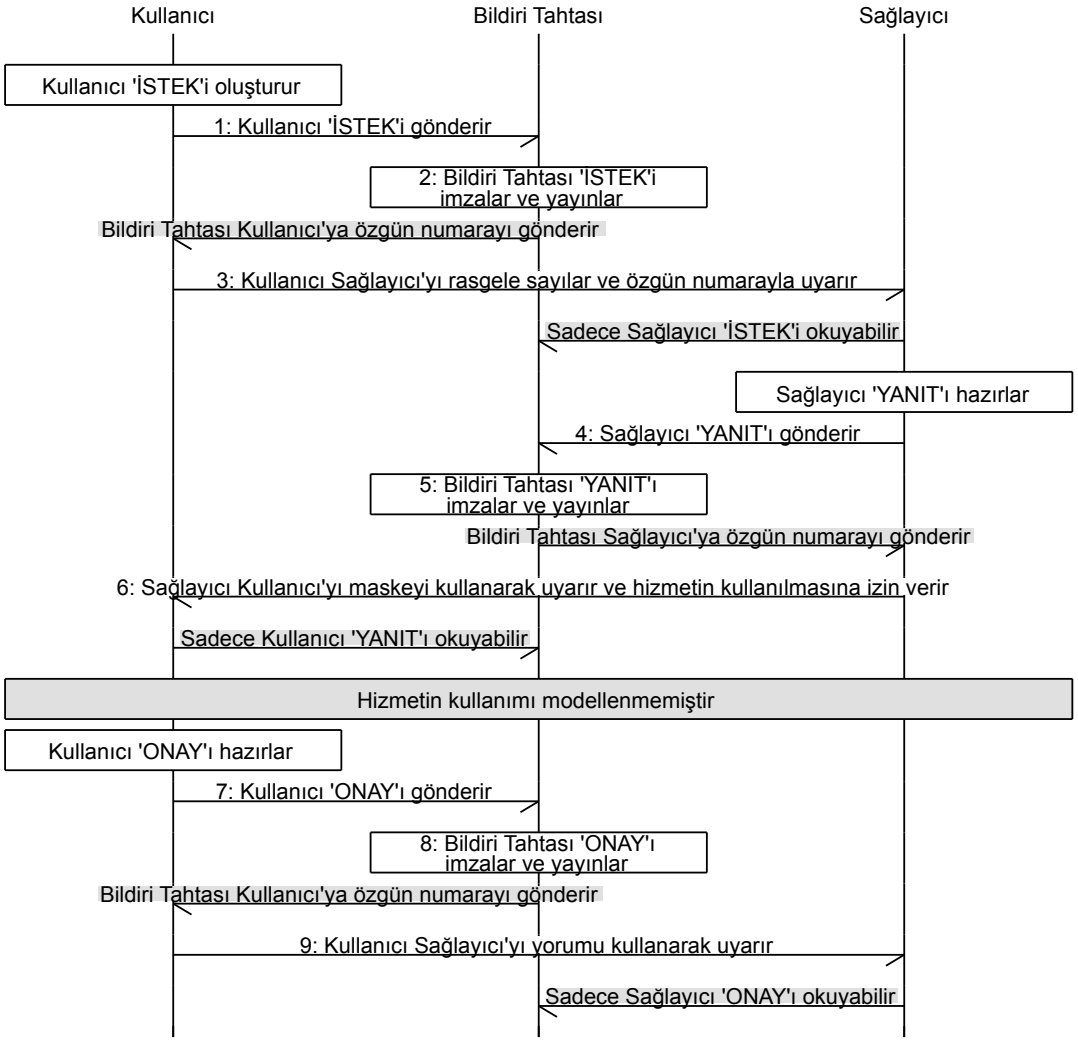
[145] **Lin, F. J., Chu, P. M. ve Liu, M. T.** (1987). Protocol Verification Using Reachability Analysis: The State Space Explosion Problem and Relief Strategies. *ACM SIGCOMM Computer Communication Review*, *17*(5), s. 126–135. doi:10.1145/55483.55496.

[146] **Pelánek, R.** (2009). Fighting State Space Explosion: Review and Evaluation. İçinde D. Cofer ve A. Fantechi (Editörler), *13th International Workshop on Formal Methods for Industrial Critical Systems*, (Cilt 5596, s. 37–52). Springer Berlin Heidelberg. doi:10.1007/978-3-642-03240-0_7.

EKLER

EK A : Güvenli günlük tutma düzeneğinin biçimsel doğrulanması için hazırlanan modele ilişkin akış, kodlar ve sonuçlar

EK A



Şekil A.1 : Benzetimde kullanılan protokolün önerilen protokolden farklarını gösteren akış.

```
role kullanici(

  K,S,B       : agent,
  H           : hash_func,
  Ak,As,Ab    : public_key,
  Ks,Kb,Sk,Bk : channel(dy)) played_by K def=

  local
    State,Rksb,Rks,N      : nat,
    Y,M,O,D1,D3,D4,D5,D6  : text

  init
      State  := 0

  transition

% İSTEK'in hazırlanıp gönderildiği durum geçişi
  0.  State   = 0
  =|> State' := 1
  /\  D1' := new()
  /\  O' := new()
  /\  Rksb' := new()
  /\  Rks' := new()
  /\  secret(O', o, {K,S})
  /\  secret(Rks', rks, {K,S})
  /\  secret(Rksb', rksb, {K,S,B})
  /\  Kb({K.{O'.Rks'.D1'}_inv(Ak)}_As.{{Rksb'.D1'}_inv(Ak)}_Ab)
% Bir değişkenle ilk kez karşılaşıldığında bu tırnak imi (') ile belirtilir. Değiş-
% ken sabitse tırnak imi kullanılmaz. Rasgele sayılara bu durum geçişinde ilk kez
% değer verildiği için tırnak imi kullanıldı. Bundan sonra kullanılmayacak.

% İSTEK'in okunduğu ve Sağlayıcı'nın uyarıldığı durum geçişi
  1.  State   =  1
  /\  Bk({{N'}_inv(Ab)}_Ak)
  =|> State' :=  2
  /\  Ks({{N'.Rksb.Rks}_inv(Ak)}_As)

% Maske'nin alındığı ve Bildiri Tahtası'nda YANIT'ın arandığı durum geçişi
  2.  State   =  2
  /\  Sk({{N.M'}_inv(As)}_Ak)
  =|> State' :=  3
  /\  Kb(N)

% YANIT'ın alındığı ve ONAY'ın hazırlanıp gönderildiği durum geçişi
  3.  State   =  3
  /\  Bk(N.{{xor({H(O.M)}_inv(As),M).Rks.D3'}_inv(As)}_Ak.
      {{N.Rksb.D3'}_inv(As)}_Ab.D4'.
      {H(N.{{xor({H(O.M)}_inv(As),M).Rks.D3'}_inv(As)}_Ak.
      {{N.Rksb.D3'}_inv(As)}_Ab.D4')}_inv(Ab))
  =|> State' :=  4
  /\  Y' := new()
  /\  D5' := new()
  /\  secret(Y', y, {K,S})
  /\  Kb({{Y'.Rks.D5'}_inv(Ak)}_As.{{N.Rksb.D5'}_inv(Ak)}_Ab)
% Anlaşmanın hesabı "A := xor(xor({H(O.M)}_inv(As),M),M)" biçiminde yapılabilir.

% ONAY'ın okunup Sağlayıcı'nın uyarıldığı durum geçişi
  4.  State   =  4
  /\  Bk({{N}_inv(Ab)}_Ak)
  =|> State' :=  5
  /\  Ks({{N.Y}_inv(Ak)}_As)

end role
```

Şekil A.2 : Kullanıcının HLPSL rolü.

```
role saglayici(

  K,S,B       : agent,
  H           : hash_func,
  Ak,As,Ab    : public_key,
  Sk,Ks,Sb,Bs : channel(dy)) played_by S def=

  local
    State,Rksb,Rks,N         : nat,
    Y,M,O,D1,D2,D3,D4,D5,D6  : text,
    A                        : {hash_func}_inv(public_key)

  init
      State  := 0

  transition

% Sağlayıcı'nın uyarıyı aldıktan sonra Bildiri Tahtası'nı yokladığı durum geçişi
  0.  State   =  0
  /\  Ks({{N'.Rksb'.Rks'}_inv(Ak)}_As)
  =|> State' :=  1
  /\  Sb(N')
% İki rasgele sayı da ilk kez burada elde edildiği için ilk değerlerini burada
% alıyor. Bu nedenle tırnakla imleniyor.

% İSTEK'in alındığı ve YANIT'ın hazırlanıp gönderildiği durum geçişi
  1.  State   =  1
  /\  Bs(N.{K.{O'.Rks.D1'}_inv(Ak)}_As.{{Rksb.D1'}_inv(Ak)}_Ab.D2'.
      {H(N.{K.{O'.Rks.D1'}_inv(Ak)}_As.{{Rksb.D1'}_inv(Ak)}_Ab.D2')}_inv(Ab))
  =|> State' :=  2
  /\  M' := new()
  /\  A' := {H(O'.M')}_inv(As)
  /\  secret(A', a, {K,S})
  /\  secret(M', m, {K,S,B})
  /\  D3' := new()
  /\  Sb({{xor(A',M').Rks.D3'}_inv(As)}_Ak.{{N.Rksb.D3'}_inv(As)}_Ab)
% Bu durum geçişinde rasgele sayıların uzlaşması denetlenmektedir. O durumundan 1
% durumuna geçişte değerleri belirlenen rasgele sayılarla uyuşan rasgele sayılar bu
% durum geçişinin koşulu olarak Bildiri Tahtası'ndan okunmak zorundadır. Uzlaşma
% olmazsa durum geçişi gerçekleşmeyecek ve benzetim sonlanmayacaktır. İSTEK'in ikinci
% yarısındaki uzlaşma bağlamın tüm bilgisine sahip olan benzetim aracı tarafından
% denetlenebilir. Bu şifrenin alıcı tarafta çözüldüğü anlamında gelmez.

% YANIT'ın okunup maske'nin gönderildiği durum geçişi
  2.  State   =  2
  /\  Sb({{N}_inv(Ab)}_As)
  =|> State' :=  3
  /\  Sk({{N.M}_inv(As)}_Ak)

% Yorum'un alınıp Bildiri Tahtası'nın yoklandığı durum geçişi
  3.  State   =  3
  /\  Ks({{N.Y'}_inv(Ak)}_As)
  =|> State' :=  4
  /\  Sb(N)

% ONAY'ın alındığı durum geçişi
  4.  State   =  4
  /\  Bs(N.{{Y.Rks.D5'}_inv(Ak)}_As.{{N.Rksb.D5'}_inv(Ak)}_Ab.D6'.
      {H(N.{{Y.Rks.D5'}_inv(Ak)}_As.{{N.Rksb.D5'}_inv(Ak)}_Ab.D6')}_inv(Ab))
  =|> State' :=  5

end role
```

Şekil A.3 : Sağlayıcının HLPSL rolü.

```
role bildiri(

  K,S,B       : agent,
  H           : hash_func,
  Ak,As,Ab    : public_key,
  Bk,Bs,Kb,Sb : channel(dy)) played_by B def=

  local
    State,Rks,Rksb,N         : nat,
    M,O,Y,D1,D2,D3,D4,D5,D6  : text,
    Sign                     : {hash_func}_inv(public_key)

  init
      State  := 0

  transition

% İSTEK'in yayınlandığı ve ilk kez oluşturulan özgün numaranın döndüğü durum geçişi
  0.  State   =  0
  /\  Kb({K.{O'.Rks'.D1'}_inv(Ak)}_As.{{Rksb'.D1'}_inv(Ak)}_Ab)
  =|> State' :=  1
  /\  N' := new()
  /\  D2' := new()
  /\  Sign' := {H(N'.{K.{O'.Rks'.D1'}_inv(Ak)}_As.
      {{Rksb'.D1'}_inv(Ak)}_Ab.D2')}_inv(Ab)
  /\  Bk({{N'}_inv(Ab)}_Ak)
% Burada bildiri tahtası N' değerini doğrudan doğruya güvenli bir kanaldan bildirerek
% durum sayılarının artışını engellemektedir. Bu yapılmadan da kullanıcının özgün
% numarayı öğrenmesi olanaklıdır ancak saldırganın olası davranışlarının modellenmesi
% durum uzayını gereğinden fazla genişletir.

% Sağlayıcı'nın İSTEK'i okuduğu durum geçişi
  1.  State  =  1
  /\  Sb(N)
  =|> State'  :=  2
  /\ Bs(N.{K.{O.Rks.D1}_inv(Ak)}_As.{{Rksb.D1}_inv(Ak)}_Ab.D2.Sign)
% Tam bir benzetim için bildiri tahtasının her isteğe yanıt vermesi gerekir. Burada
% durum sayısını düşük tutmak için sadece sağlayıcının yanıtı istediği
% varsayılmıştır. Benzer varsayımlar sonraki durum geçişlerinde de geçerlidir.

% YANIT'ın yayınlandığı ve özgün numaranın döndüğü durun geçişi
  2.  State  =  2
  /\  Sb({{xor({H(O.M')}_inv(As),M').Rks.D3'}_inv(As)}_Ak.{{N.Rksb.D3'}_inv(As)}_Ab)
  =|> State'  :=  3
  /\  D4' := new()
  /\  Sign' := {H(N.{{xor({H(O.M')}_inv(As),M').Rks.D3'}_inv(As)}_Ak.
      {{N.Rksb.D3'}_inv(As)}_Ab.D4')}_inv(Ab) /\ Bs({{N}_inv(Ab)}_As)

% Kullanıcı'nın YANIT'ı okuduğu durum geçişi
  3.  State  =  3
  /\  Kb(N)
  =|> State'  :=  4
  /\  Bk(N.{{xor({H(O.M)}_inv(As),M).Rks.D3}_inv(As)}_Ak.
      {{N.Rksb.D3}_inv(As)}_Ab.D4.Sign)

% ONAY'ın yayınlandığı ve özgün numaranın döndüğü durun geçişi
  4.  State  =  4
  /\  Kb({{Y'.Rks.D5'}_inv(Ak)}_As.{{N.Rksb.D5'}_inv(Ak)}_Ab)
  =|> State'  :=  5
  /\  D6' := new()
  /\  Sign' := {H(N.{{Y'.Rks.D5'}_inv(Ak)}_As.{{N.Rksb.D5'}_inv(Ak)}_Ab.D6')}_inv(Ab)
  /\  Bk({{N}_inv(Ab)}_Ak)

% Sağlayıcı'nın ONAY'ı okuduğu durum geçişi
  5.  State  =  5
  /\  Sb(N)
  =|> State'  :=  6
  /\  Bs(N.{{Y.Rks.D5}_inv(Ak)}_As.{{N.Rksb.D5}_inv(Ak)}_Ab.D6.Sign)

end role
```

Şekil A.4 : Bildiri Tahtasının HLPSL rolü.

```
role environment() def=

  const
    k,s,b,i           :  agent,
    ak,as,ab,ai       :  public_key,
    h                 :  hash_func,
    ks,kb,sk,bk,bs,sb :  channel(dy),
    y,a,m,o,rks,rksb  :  protocol_id

  intruder_knowledge  =  {k,s,b,i,h,ak,as,ab,ai,inv(ai)

% Bağlam modelinde tüm değişkenlerin son değeri tanımlanır. Bu rol hiç eferans
% içermez. Bu nedenle, geleneğe göre değişken adlarında hiç büyük harf kullanılmaz.

% Bağlamda özel bir etmen olan saldırgan (intruder) 'i' değişkeniyle gösterilir. Bu
% etmen önceden "intruder_knowledge" kümesi içinde tanımlanan bilgilere ulaşabilir.
% Ayrıca, bağlam içinde çalışan tüm oturumlarda erişebildiği tüm kanallarda ya da
% öykündüğü tüm kullanıcılardan toplayabildiği tüm bilgileri bir araya getirerek
% saldırı tasarlayabilir.

% protocol_id ile gösterilenler tarafların (asıllama amacıyla) üzerinde uzlaşmayı ya
% da (gizli tutmak üzere) saklamayı seçebilecekleri sabit değerlerdir.

  composition
  session(k,s,b,h,ak,as,ab,ks,kb,sk,bk,bs,sb)
  /\  session(k,s,b,h,ak,as,ab,ks,kb,sk,bk,bs,sb)

end role
```

(a) : Dürüst taraflar içeren iki oturumlu bağlam.

```
role environment() def=

  const
    k,s,b,i           :  agent,
    ak,as,ab,ai       :  public_key,
    h                 :  hash_func,
    ks,kb,sk,bk,bs,sb :  channel(dy),
    y,a,m,o,rks,rksb  :  protocol_id

  intruder_knowledge  =  {k,s,b,i,h,ak,as,ab,ai,inv(ai)

  composition
  session(i,s,b,h,ai,as,ab,ks,kb,sk,bk,bs,sb)
  /\  session(k,i,b,h,ak,ai,ab,ks,kb,sk,bk,bs,sb)
  /\  session(k,s,i,h,ak,as,ai,ks,kb,sk,bk,bs,sb)

end role
```

(b) : Saldırganlar içeren üç oturumlu bağlam.

Şekil A.5 : Bağlamları tanımlayan HLPSL kodları.

```
role session(

  K,S,B             : agent,
  H                 : hash_func,
  Ak,As,Ab          : public_key,
  Ks,Kb,Sk,Bk,Bs,Sb : channel(dy)) def=

  composition
  kullanici(K,S,B,H,Ak,As,Ab,Ks,Kb,Sk,Bk)
  /\  bildiri(K,S,B,H,Ak,As,Ab,Bk,Bs,Kb,Sb)
  /\  saglayici(K,S,B,H,Ak,As,Ab,Sk,Ks,Sb,Bs)

end role
```

Şekil A.6 : Protokolün bir oturumunu tanımlayan HLPSL kodu.

```
goal
  secrecy_of  o
  secrecy_of  rks
  secrecy_of  rksb
  secrecy_of  a
  secrecy_of  m
  secrecy_of  y
end goal
```

Şekil A.7 : AVISPA ile doğrulanması istenen hedefleri belirleyen HLPSL kodu.

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /usr/share/avispa-1.1/testsuite/results/ikiDurust.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 21.64s
  visitedNodes: 7027 nodes
  depth: 32 plies
```

(a) : OFMC yöntemiyle elde edilen sonuç.

```
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL
  /usr/share/avispa-1.1/testsuite/results/ikiDurust.if

GOAL
  As Specified

BACKEND
  CL-AtSe

STATISTICS

  Analysed   : 1131512 states
  Reachable  : 401128 states
  Translation: 0.15 seconds
  Computation: 107.40 seconds
```

(b) : CL-AtSe yöntemiyle elde edilen sonuç.

Şekil A.8 : Dürüst taraflar içeren iki oturumlu benzetimin sonuçları.

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /usr/share/avispa-1.1/testsuite/results/ucSaldirgan.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 22.88s
  visitedNodes: 0 nodes
  depth: 1000000 plies
```

(a) : OFMC yöntemiyle elde edilen sonuç.

```
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL
  /usr/share/avispa-1.1/testsuite/results/ucSaldirgan.if

GOAL
  As Specified

BACKEND
  CL-AtSe

STATISTICS

  Analysed   : 206970 states
  Reachable  : 152981 states
  Translation: 0.11 seconds
  Computation: 64.29 seconds
```

(b) : CL-AtSe yöntemiyle elde edilen sonuç.

Şekil A.9 : Saldırganın taraflara öykündüğü üç oturumlu benzetimin sonuçları.

ÖZGEÇMİŞ

Ad-Soyad : Mehmet Tahir SANDIKKAYA

Doğum Yeri ve Tarihi : Ankara, 25.02.1979

Adres : İTÜ Bilgisayar ve Bilişim Fakültesi, TR34469, Reşitpaşa, Sarıyer, İstanbul.

E-Posta : sandikkaya@itu.edu.tr

ÖĞRENİM DURUMU:

- **Lisans** : İstanbul Teknik Üniversitesi, Elektrik-Elektronik Fakültesi, Elektrik Mühendisliği Programı, 2002.
- **Yüksek Lisans** : İstanbul Teknik Üniversitesi, Bilişim Enstitüsü, Bilgisayar Bilimleri Programı, 2006.

MESLEKİ DENEYİM VE ÖDÜLLER:

- **Araştırma Görevlisi** : İstanbul Teknik Üniversitesi, Bilişim Enstitüsü, Aralık 2002–Şubat 2008.
- **Araştırmacı** : Katholieke Hogeschool Sint-Lieven ve Katholieke Universiteit Leuven, Şubat 2008–Ocak 2010.
- **Araştırma Görevlisi** : İstanbul Teknik Üniversitesi, Elektrik-Elektronik Fakültesi, Bilgisayar Mühendisliği Bölümü, Kasım 2010–Haziran 2011.
- **Araştırma Görevlisi** : İstanbul Teknik Üniversitesi, Bilgisayar ve Bilişim Fakültesi, Bilgisayar Mühendisliği Bölümü, Haziran 2011–.

YAYIN VE PATENT LİSTESİ:

- **Sandıkkaya, M. T.** ve Örencik, B. (2006). Agent-Based Offline Electronic Voting. İçinde *30th Annual International Computer Software and Applications*

Conference (COMPSAC 2006), (Cilt 2, s. 333–340). IEEE Computer Society. doi:10.1109/COMPSAC.2006.107.

▪ Kantarcı, B., **Sandıkkaya, M. T.**, Gençata, A. ve Oktuğ, S. F. (2007). Prudent Creditization Polling (PCP): A Novel Adaptive Polling Service for an EPON. İçinde I. Tomkos, F. Neri, J. Solé-Pareta, X. Masip-Bruin ve S. Sánchez-López (Editörler), *Optical Network Design and Modeling, 11th International IFIP TC6 Conference (ONDM 2007)*, (Cilt 4534, s. 388–397). Springer Berlin Heidelberg. doi:10.1007/978-3-540-72731-6_42.

▪ Naessens, V., **Sandıkkaya, M. T.**, Lapon, J., Verslype, K., Verhaeghe, P., Nigusse, G. ve De Decker, B. (2009). Privacy Policies, Tools and Mechanisms of the Future. İçinde J. Camenisch ve D. Kesdoğan (Editörler), *iNetSec 2009 — Open Research Problems in Network Security - IFIP WG 11.4 International Workshop, Revised Selected Papers*, (Cilt 309, s. 125–138). Springer Berlin Heidelberg. doi:10.1007/978-3-642-05437-2_12.

▪ Layouni, M., Verslype, K., **Sandıkkaya, M. T.**, De Decker, B. ve Vangheluwe, H. (2009). Privacy-Preserving Telemonitoring for eHealth. İçinde E. Gudes ve J. Vaidya (Editörler), *Data and Applications Security XXIII, 23rd Annual IFIP WG 11.3 Working Conference Proceedings*, (Cilt 5645, s. 95–110). Springer Berlin Heidelberg. doi:10.1007/978-3-642-03007-9_7.

▪ **Sandıkkaya, M. T.**, De Decker, B. ve Naessens, V. (2010). Privacy in Commercial Medical Storage Systems. İçinde M. Szomszor ve P. Kostkova (Editörler), *Electronic Healthcare - Third International Conference (eHealth 2010) Revised Selected Papers*, (Cilt 69, s. 247–258). Springer Berlin Heidelberg. doi:10.1007/978-3-642-23635-8_32.

DOKTORA TEZİNDEN TÜRETİLEN YAYINLAR, SUNUMLAR VE PATENTLER:

▪ **Sandıkkaya, M. T.** ve Harmancı, A. E. (2012). Security Problems of Platform-as-a-Service (PaaS) Clouds and Practical Solutions to the Problems. İçinde *IEEE 31st Symposium on Reliable Distributed Systems (SRDS 2012)*, (s. 463–468). IEEE Computer Society. doi:10.1109/SRDS.2012.84.

▪ **Sandıkkaya, M. T.**, Ödevci, B. ve Ovatman, T. (2014). Practical Runtime Security Mechanisms for an aPaaS Cloud. İçinde *IEEE Global Communications Conference (Globecom 2014)*, (s. 53–58). IEEE Communications Society. doi:10.1109/GLOCOMW.2014.7063385.

▪ **Sandıkkaya, M. T.** ve Harmancı, A. E. (2015). A Security Paradigm for PaaS Clouds. *Proceedings of the Romanian Academy Series A: Mathematics, Physics, Technical Sciences, Information Science*, *16*(Special Issue), s. 345–356.

▪ **Sandıkkaya, M. T.**, Ovatman, T. ve Harmancı, A. E. (2015). Design and formal verification of a cloud compliant secure logging mechanism. *IET Information Security*. Tamamlanmış çevrimiçi yayın. doi:10.1049/iet-ifs.2014.0625.