Formal Security of Zero Trust Architectures

Alexandre Poirrier

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2024IPPAX050

Thèse de doctorat

ÉCOLE POLYTECHNIQUE

IP PARIS

Formal Security of Zero Trust Architectures

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à l'École polytechnique

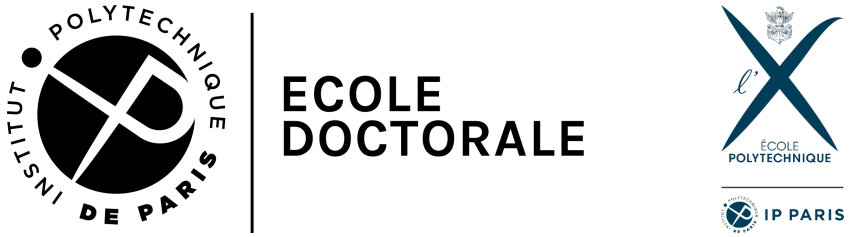
École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (EDIPP)
Spécialité de doctorat : Réseaux, information et communications

Thèse présentée et soutenue à Palaiseau, le 30/09/2024, par

ALEXANDRE POIRRIER

Composition du Jury :

HERVÉ DEBAR Professeur, Télécom SudParis, France	Président
MIGUEL RIO Professeur, University College London, UK	Rapporteur
ROMAIN LABORDE Professeur, Institut de Recherche en Informatique de Toulouse, France	Rapporteur
JÉRÉMY BUISSON Maître de conférences, École de l'Air et de l'Espace, France	Examinateur
THOMAS HEIDE CLAUSEN Professeur, École polytechnique, France	Directeur de thèse
Pierre Dauchy Ingénieur en Chef de l'Armement, Conseil Général de l'Armement, France	Invité



Institut Polytechnique de Paris - Ecole polytechnique

Thèse de doctorat

Formal Security of Zero Trust Architectures

Candidate: Alexandre Poirrier

Supervisors: Thomas Heide Clausen
Professor, Ecole polytechnique, France
Laurent Cailleux
Direction Générale de l'Armement, France

2024

Abstract

The security architecture of Information Technology (IT) systems has traditionally been based on the *perimeter security* model, in which resources are grouped into perimeters isolated through network mechanisms, and devices are authenticated to access perimeters. Once within a perimeter, devices are *implicitly trusted*, and enjoy unrestricted access to resources within that perimeter. However, history has shown that such trust is misplaced, as numerous security threats and successful attacks against perimeter-based architectures have been documented. Furthermore, the emergence of new IT usages, such as cloud services, work-from-home, and service providers and subsidiaries relationships, has challenged the relevance of considering a monolithic, trusted network for accessing resources. These considerations have led to the emergence of a novel security paradigm, called by *zero trust*. Founded on the principle 'never trust, always verify', this approach transforms the notion of perimeter and establishes a set of security principles that prioritize context-aware and dynamic authorization. Nevertheless, implementing zero trust poses significant challenges, due to a lack of clear guidelines for defining zero trust.

In this context, this thesis investigates whether, and how, it is possible to develop a practical and formal framework for reasoning about the security of IT architectures. First, a thorough survey of zero trust is conducted, and a taxonomy of existing zero trust technologies and architectures is developed, enabling a comprehensive understanding of zero trust. This leads to the development of an evaluation framework, that is used to identify gaps within zero trust research. This thesis provides contributions aiming to address some of these gaps, for enhancing the state of zero trust technology development. These improvements are integrated into a proof-of-concept zero trust architecture implemented for this thesis, illustrating a method for extending an existing zero trust architecture. Finally, the thesis takes a step back, and evaluates the extent to which the zero trust framework addresses real-world problems, demonstrating that the zero trust framework alone is not sufficient for protecting sensitive data and services.

Keywords Architecture, Cryptography, Cybersecurity, Network Security, Zero Trust

Résumé

Depuis les années 1980, les technologies d'information se démocratisent, en se propageant au sein des entreprises et des foyers. Cette popularisation des technologies d'information entraîne une transition numérique, par laquelle de nombreux secteurs d'activité utilisent des outils numériques – afin d'accroître leurs capacités de production et de communication, leur agilité, et leur productivité.

Toutefois, cette généralisation des systèmes d'information à tout secteur d'activité engendre une dépendance à ces systèmes, comme l'illustrent les nombreuses 'cyber-attaques' des années 2010 et 2020 – ces attaques pouvant gravement nuire aux entreprises et organismes victimes, autant sur le plan financier que sur sa réputation. Et ce, même si l'activité principale de la victime n'est pas en lien avec les technologies d'information, comme par exemple un hôpital ou une banque. De plus, l'essor de l'usage des technologies informatiques implique une forte croissance des données informatisées, y compris des données personnelles et privées (tels que des dossiers médicaux ou des coordonnées bancaires) ou des données sensibles (secrets industriels ou contrats). L'interconnection des systèmes d'information via Internet rendent ces données potentiellement accessibles à des acteurs malveillants partout dans le monde.

Ainsi, entreprises et organismes sont de plus en plus vulnérables aux défaillances et attaques informatiques. Il est donc nécessaire de protéger l'intégrité et la confidentialité des données, ainsi que la disponibilité des services fournis par les systèmes d'information.

La sécurité des systèmes d'information repose traditionnellement sur le modèle de *sécurité du périmètre*, qui compartimentalise le réseau en différents périmètres isolés qui regroupent les ressources. Un système accède à un périmètre en s'authentifiant, et une fois au sein d'un périmètre, il est implicitement considéré comme étant *de confiance*, jouissant d'un accès non restreint à toutes les ressources de ce périmètre. Cependant, de nombreuses attaques envers les architectures périmétriques ont montré les limites de cette confiance. En effet, bien que le périmètre soit protégé par des systèmes de protection ayant pour but d'empêcher un acteur extérieur de pénétrer au sein du périmètre, certains de ces systèmes peuvent avoir des failles. Par ailleurs, la menace peut provenir de l'intérieur du périmètre – par exemple, un employé malveillant. Or, de par la définition du paradigme de protection périmétrique, une fois qu'un agent malveillant pénètre au sein du réseau, il peut se 'déplacer latéralement' et accéder sans contrainte aux autres ressources dans ce réseau, y compris des ressources dont il n'a pas forcément besoin de connaître ou d'utiliser.

De plus, la transformation des systèmes d'information, par exemple l'émergence des services en nuage (*cloud services*), le télétravail ou l'usage de prestataires, a remis en cause la vision d'un réseau monolithique de confiance.

Ainsi, un nouveau paradigme de sécurité a émergé, appelé *zéro confiance*. Fondé sur le principe "ne jamais faire confiance, toujours vérifier", ce paradigme transforme la notion de périmètre, en établissant un ensemble de principes de sécurité reposant sur une autorisation contextuelle et dynamique. Cependant, mettre en œuvre une architecture zéro confiance pose de nombreux défis, en particulier car il n'existe pas de définition claire du paradigme zéro confiance.

Dans ce contexte, cette thèse explore comment développer un cadre pratique et formel pour raisonner sur la sécurité des systèmes d'information. Tout d'abord, une étude approfondie du modèle zéro confiance est conduite en étudiant l'Histoire de la sécurité informatique, puis une taxonomie des technologies et architectures zéro confiance existantes est établie, ce qui permet une compréhension exhaustive du paradigme zéro confiance. Cela mène à la création d'une méthode d'évaluation des architectures, qui permet également d'identifier des lacunes dans la recherche sur le zéro confiance.

Cette thèse propose plusieurs contributions pour combler ces lacunes, afin d'améliorer l'état de l'art pour le zéro confiance. En premier lieu, une étude sur la sécurité centrée sur la donnée est effectuée, qui permet de protéger la donnée directement, indépendamment de son système de stockage. Ensuite, une étude de mécanismes d'authentification continue est proposée dans le contexte de messagerie instantanée sécurisée. Ces améliorations sont intégrées au sein d'un démonstrateur d'architecture zéro confiance, illustrant comment une architecture zéro confiance peut être complémentée par de nouvelles technologies pour améliorer sa maturité.

Enfin, cette thèse prend du recul et évalue comment le paradigme zéro confiance répond aux problèmes de sécurité auxquels font face les entreprises, démontrant qu'il n'est pas suffisant seul pour protéger les données et services sensibles. En particulier, le problème de comment partager des données et services au sein d'une fédération de domaines, sans faire confiance implicitement aux autres domaines, est étudié. Enfin, le problème de comment préserver la vie privée des utilisateurs au sein d'une architecture zéro confiance (qui surveille continuellement les utilisateurs) est abordé dans le cadre du transfert IP.

Mots-clé Architecture, Cryptographie, Cybersécurité, Sécurité des Réseaux, Zéro Confiance

Remerciements

Au terme de ces trois années de thèse, je tiens à exprimer ma gratitude pour toutes les personnes qui m'ont soutenu et aidé pendant mes études.

Tout d'abord, je remercie très chaleureusement Thomas Clausen pour avoir été mon directeur de thèse. Merci pour tes conseils avisés, qui m'ont poussé dès mes années d'études à l'École polytechnique à me lancer dans cette aventure qu'est le doctorat, et qui me portent encore aujourd'hui pour la suite de ma carrière. Ta bienveillance, ton soutien, et nos nombreuses discussions scientifiques enrichissantes m'ont permis de donner le meilleur de moi-même et de progresser avec confiance tout au long de cette thèse.

Mes remerciements vont ensuite naturellement à Laurent Cailleux, qui de même m'a encadré avec brio malgré la distance. Tes nombreuses idées et tes cas concrets issus des besoins de la DGA ont donné du sens à mes travaux de recherche. Je suis reconnaissant d'avoir pu travailler ensemble, et je suis ravi de pouvoir poursuivre l'aventure au sein de la DGA ensemble.

Je remercie ensuite l'ensemble des membres de mon jury de thèse d'avoir accepté d'évaluer mes travaux, en particulier Romain Laborde et Miguel Rio pour la lecture attentive qu'ils ont faite de mon manuscrit, et leurs nombreux commentaires très pertinents qui m'ont permis d'approfondir la réflexion sur le sujet.

Je remercie également Pierre Dauchy, François Durand et Chantal Keryjaouen qui ont permis la mise en oeuvre et le financement de cette thèse.

Je tiens à remercier chaudement les membres de l'équipe, Juan-Antonio Cordero-Fuertes, pour nos nombreuses discussions au Magnan qui m'ont de nombreuses fois donné de bonnes idées ; Kevin Jiokeng, pour ses théories pour optimiser le travail des doctorants et ses expériences sur le wifi sensing très divertissantes ; Maxence Elfatihi pour nos parties de ping-pong et de baby-foot. Merci également aux doctorants qui m'ont précédé, Thomas Feltin, Zhiyuan Yao et Yoann Desmouceaux, pour vos divers conseils et astuces.

Merci également aux différents collègues avec qui j'ai pu donner cours, Antoine Lavignotte et Ben Smith, avec qui j'ai pu avoir de super expériences. Merci également à Karima Rimbourg pour m'avoir intégré à l'équipe de l'EXED.

Je remercie également mes amis, qui ont dû subir – ou apprécier – des descriptions plus ou moins détaillées de mes sujets de recherche. Les amis de l'X, les camarades de l'escrime, en particulier Léon avec qui on partageait les galères de la thèse, Daniel grâce auquel je me suis dit que finalement la thèse ce n'est pas si galère, Guillaume pour nos soirées dîner-jeu de société-problème de RER B, et Pierre-Louis qui essayait de me distraire avec les échecs et l'escalade. Merci aussi aux coucous, Athé pour ses conseils en pâtisserie, et qui m'a montré qu'il ne faut pas faire d'escalade avant la soutenance de thèse ; Alexandre qui

m'a convaincu que passer une journée entière en tournoi d'échecs était une activité lucrative ; Arthur qui est toujours là quand il faut se retrouver dans Paris ; et Froux avec qui les discussions sont toujours enrichissantes.

Enfin, je remercie tout particulièrement ma maman, pour ta présence tout au long de ces études, du lycée à aujourd'hui. Je n'aurais jamais pu en arriver jusque là sans ton soutien inconditionnel, ta confiance en moi et ton amour.

Merci à Cécile d'avoir été là du début à la fin, de m'avoir aidé à déchiffrer les pattes de mouche de mon directeur de thèse, d'avoir supporté mes répétitions de présentation pas toujours compréhensibles, de m'avoir soutenu les mauvais jours en me préparant de bons petits plats, et d'avoir partagé les bons moments.

Contents

Part I

Introduction

Chapter 1

Introduction

An early survey on Internet security [1] systematically analyses security incidents between 1989 and 1995. In that survey, computer security is defined as a state of *trust*, in which protected processes, files, and data in transit are safeguarded against *unauthorized access* or *unauthorized use*, thus preventing attackers from achieving their objectives.

Similarly, [2] defines computer security as the techniques controlling who may modify the computer or information contained in it. More specifically, security is the set of techniques which prevent unauthorized information release, unauthorized information modification, and unauthorized denial of use.

Therefore, the protection of processes, files, and data has been – and largely is – based on the *perimeter security* model, with *network compartmentalization* and *defense-in-depth* serving as fundamental principles. This approach used to be endorsed by various national security agencies, *e.g.*, the National Institute of Standards and Technologies (NIST) [3], the Department of Homeland Security [4], and the French Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI)[1]. To implement compartmentalization, system access privileges are granted depending on the topological location of systems, grouped into *perimeters*, which are then isolated through network mechanisms, *e.g.*, firewalls and VLANs [5], [6]. Therefore, once a device is authenticated within the network and assigned to a specific perimeter, it is deemed *trusted*, enjoying unrestricted access to resources within that perimeter, and thus has the ability to transmit and exfiltrate data.

However, history has demonstrated that such trust is misplaced, as more sophisticated attacks can target vulnerabilities in perimeters [7]. Furthermore, insider threats, *e.g.*, a malicious user, a compromised user, or an unauthorized individual using the device of an authorized user, pose a significant risk to perimeter security [70]. Additionally, the emergence of cloud services and 'work from home' have challenged the relevance of perimeter security [8], [9]. Indeed, the shift to cloud services [10] has transformed the way 'systems' are designed: a 'system' may no longer be a monolithic entity hosted on a server in

[1] Agence Nationale de la Sécurité des Systèmes d'Information, "Recommandations pour la protection des systèmes d'information essentiels," ANSSI, Tech. Rep., 2020. [Online]. Available: https://cyber.gouv.fr/publications/recommandations-pour-la-protection-des-systemes-dinformation-essentiels.

a data-centre, but can instead combine multiple microservices — each hosted in disparate locations across the cloud. Moreover, the rise of remote work and Bring-Your-Own-Device policies have employees connect to organization resources from unmanaged systems and untrusted networks. Likewise, users are no longer confined to a specific corporate network, as they may have multiple, distinct means of accessing resources, *e.g.*, employees from subsidiaries or service providers needing access to corporate resources. Thus, establishing a single and static network perimeter is no longer sufficient.

The shortcomings of perimeter security have led to the emergence of a novel security paradigm: *zero trust*. This approach abandons the notion of a perimeter, by establishing a set of security principles that prioritize context-aware and dynamic authorization schemes. Unlike entry-point authentication, zero trust does not rely solely on verifying identities at network boundaries; instead, it grants access to resources based on continuous evaluation of user behavior and contextual information.

Nevertheless, implementing a zero trust architecture poses significant challenges. First, the lack of clear guidelines and metrics for defining what constitutes a zero trust architecture makes it difficult to design migration plans and clear objectives [71]. A second challenge is the integration of zero trust technologies with existing legacy systems, and making them interoperable [72]. Additionally, introducing zero trust components into an existing architecture augments its attack surface, potentially creating new vulnerabilities [157].

The rise of insider threats and of cloud services prompted for a change of security paradigm. Similarly, *privacy*, defined as 'the ability for individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others' [11], is a growing concern. This concern was born from privacy breaches, *e.g.*, the possibility of retrieving trips made by celebrities from an anonymized data set composed of taxi locations [12], leading to technical solutions for ensuring privacy, *e.g.*, differential privacy for ensuring anonymity in data sets [13], as well as regulations, *e.g.*, the European Union General Data Protection Regulation (GDPR)[2] that requires organizations to get consent from subjects for processing their data. Moreover, organizations resources may not be completely isolated, with organizations sharing their data and services with external entities, *e.g.*, with partners, cloud service providers, or subsidiaries. Therefore, considering a security framework for IT security protecting solely the confidentiality, integrity, availability, and authenticity of resources may not be enough.

> *In this context, the question explored in this thesis is whether, and how, it is possible to develop a practical and formal framework for reasoning about the security of IT architectures, thereby enabling architects to evaluate the security guarantees provided by their designs. This is accomplished by studying the fundamental components of zero trust, and by exploring how architectures can be enhanced with technologies to better align with zero trust principles, and even to offer a security level beyond zero trust principles.*

[2] E. Commission, *General data protection regulation*, 2016. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj.

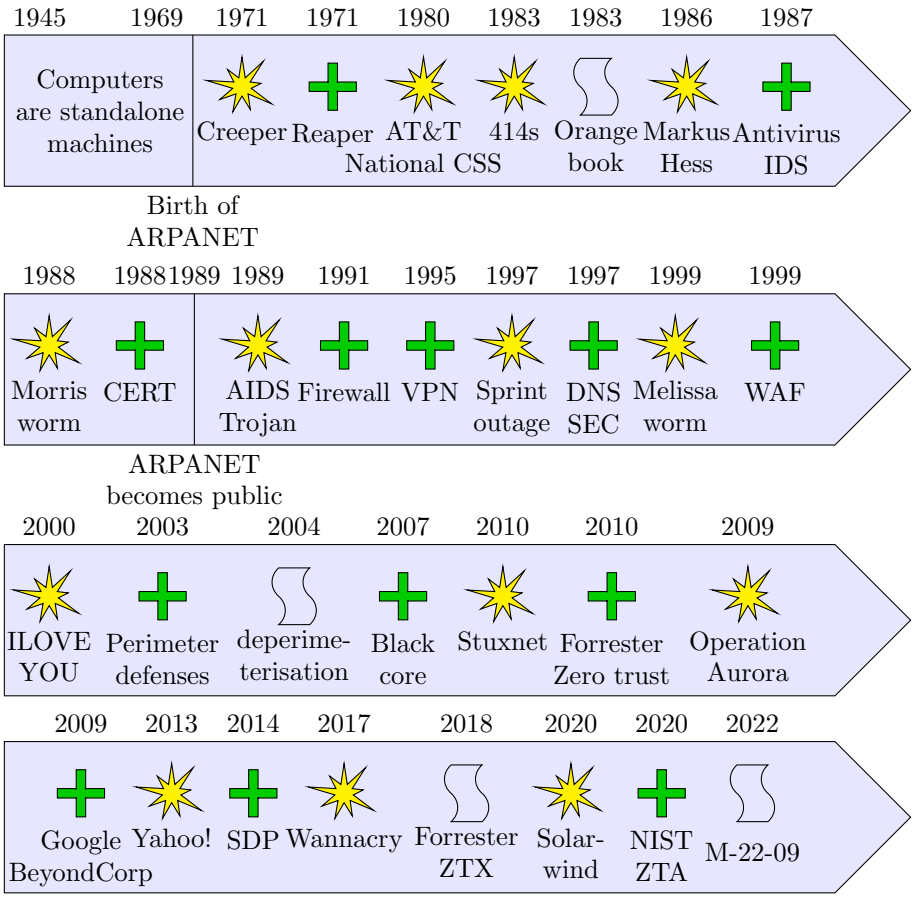


Figure 1.1: History of Internet security.

The remainder of this introductory chapter is structured as follows. Section 1.1 provides historical context on the evolution of IT security technologies, from early cybersecurity concerns to the widespread adoption of perimeter security, and ultimately to the emergence of zero trust as a response to evolving networks and threats. Then, section 1.2 presents multiple approaches for defining zero trust, which highlights the challenges involved in establishing a uniform definition of zero trust. Section 1.3 argues for the necessity of adopting zero trust technologies to secure organizations, discusses the need for a further security model for IT infrastructures, and highlights the need for a formal framework to systematically evaluate the security guarantees offered by IT architectures. Finally, section 1.4 summarizes the contributions of this thesis.

1.1 Background

At the dawn of digital computing, between the Second World War and the 1960s, computer systems were standalone entities. Security was enforced by

restricting physical access to the 'machine room'[3].

This section presents a brief history of cybersecurity from there, to highlight the context that has led to the development of the zero trust paradigm: how it is inspired, and how it differs, from previous cybersecurity paradigms. Major events and security innovations in that department are depicted in figure 1.1.

1.1.1 Birth of Cybersecurity

The first idea of defensive security emerged in 1967, when IBM invited students to try one of their computers. These students explored accessible parts of the system, then learnt the language used to program the system, enabling them to gain access to restricted parts within the system. This first instance of 'ethical hacking' resulted in the development of defensive measures[4], *e.g.*, memory protection, privileged instructions, or logging significant events [14].

Next, with the birth of the Advanced Research Projects Agency NETwork (ARPANET) in 1969, computers became communicating entities, accessible at distance. In 1971, the first computer worm, a standalone computer program that replicates itself to spread to other computers [15], was born. Named 'The Creeper', it would move from computer to computer, displaying a message on each computer that it infected [16]. Shortly after, 'Reaper' was created – the first antivirus, which propagated in the network with the task of removing Creeper occurrences.

Thus, the topic of cybersecurity gained significance during the 1970s. The US Department of Defense (DoD), with companies such as MITRE, provided initial definitions for secure computer systems [17], and academic teams focused their effort on cybersecurity research [18].

1.1.2 Cybersecurity as a Growing Concern

Numerous impactful attacks marked the beginning of the 1980s, such as the breach of security at National CSS in 1980[5], Ian Murphy (also known as 'Captain Zap') breaking into AT&T's computers, for which he became the first to receive a felony conviction for a cybercrime, or the 414s attack of the Los Alamos National Laboratory in 1983 [19]. Those attacks triggered in the U.S. a national level response, to protect sensitive information, and in 1983, the DoD published the *Trusted Computer System Evaluation Criteria*, referred to as the '*Orange Book*', providing guidance on how to assess the trustworthiness of software for processing classified information, and laying out a set of guidelines for security solutions manufacturers.

Moreover, the National Policy on Telecommunications and Automated Information Systems Security, issued as National Security Decision Directive 145 (NSDD-145) [20], was released in 1984. Its novelty was the creation of the notion of 'sensitive' data, as an intermediate level between the traditional 'classified' and 'routine' levels for classifying and securing information [21].

[3] B. J. Copeland, *Colossus: The secrets of Bletchley Park's code-breaking computers*. Oxford University Press, 2010, ISBN: 9780199578146.

[4] K. Chadd, *The history of cybersecurity*, blog post, 2020. [Online]. Available: https://blog.avast.com/history-of-cybersecurity-avast.

[5] V. McLellan, "Case of the purloined password," *The New York Times*, 1981. [Online]. Available: https://www.nytimes.com/1981/07/26/business/case-of-the-purloined-password.html.

To support NSDD-145, the *Orange Book* was updated in 1985. However, evaluation criteria for software in the *Orange Book* were unclear, leading to vendors and evaluators disagreeing on products classifications. Thus, few venders developed security solutions following Orange Book recommendations [22].

Following the Pentagon attack by Markus Hess in 1986[6], the Computer Security Act of 1987 updated NSDD-145, by putting the National Bureau of Standards (NBS) in charge of civilian computer security, and by tasking the National Security Agency (NSA) with advising the federal government on the protection of sensitive information.

In parallel, numerous computer viruses spread worldwide, using floppy disks as the main infection vector. The Brain virus was the first virus to infect computers outside a test laboratory, but often did little damage [23]. The Lehigh virus, discovered in 1987, would infect floppy disks by hiding itself into an operating system executable. When the executable was executed, the virus would load itself into memory, and intercept systems interrupts. If during such an interrupt, another disk was accessed, and was not already infected, the virus would copy itself to this disk. After having been copied four times, the virus would completely erase the original disk [24], [25].

To counter the threat of viruses, the first commercial antivirus programs were published in 1987, *e.g.*, Anti4us by Erwin Lantig and FlushShot by Ross Greenberg[7]. Similarly, the first model for Intrusion Detection System (IDS) was developed in 1987 by [26], and the first IDS, called Network Security Monitor, was designed and prototyped in 1989 by [27].

In 1988, Robert T. Morris launched a worm on ARPANET. That worm would exploit a vulnerability in the email application – to send and execute files on remote machines – and a buffer overflow vulnerability in the `finger` application, to replicate itself on remote machines[8]. While the worm's only action was to replicate itself, without code for causing damage, it could infect a same machine many times, until the machine became inoperable, and this incident led to the creation of the first Computer Emergency Response Team (CERT) by the Defense Advanced Research Project Agency (DARPA) [28].

1.1.3 Rise of the Internet

In 1989, ARPANET became public, and more commonly known as 'the Internet'. Internet security and legislation became a worldwide concern, leading to the creation of agencies and laws, such as the United Kingdom Computer Misuse Act in 1990. In 1996, the Clington-Cohen Act in the U.S. promoted high security for any federal agency, and not just specifically for classified networks, thus acknowledging that classified information can reach non-classified networks.

The number of viruses, and the diversity of threats grows. The first ransomware, the AIDS Trojan developed by Joseph Popp, was released in 1989[9].

[6] J. Goodchild, "10 infamous hacks and hackers," *CIO*, 2012. [Online]. Available: https://www.cio.com/article/220322/10-infamous-hacks-and-hackers.html.

[7] A. Terekhov, *History of the antivirus*. [Online]. Available: https://www.hotspotshield.com/blog/history-of-the-antivirus/.

[8] B. Page, *A report on the internet worm*, Nov. 1988. [Online]. Available: https://www.ee.torontomu.ca/~elf/hack/iworm.html.

[9] S. M. Kelly, "The bizarre story of the inventor of a ransomware," *CNN Business*, 2021.

In the mid 1990s, the number of viruses and malwares exploded, with viruses using stealth capabilities, polymorphism, and micro viruses to evade detection by antivirus[10].

Antivirus technology also improved, and became more and more popular, with free antivirus software made available in 2001, *e.g.*, by Avast, or the open source antivirus ClamAV[11]. Because antivirus software takes space on computers, and can reduce performance, the first cloud-based antivirus solutions were proposed in 2007 by Panda Security and McAfee [29].

1.1.4 Towards Perimetric Security

ARPANET was designed as a research project, and initially used by researchers and military personnel. Thus, people using ARPANET were trusted not to – at least, intentionally – break network functionalities.

This paradigm was inherited by the Internet, as it was publicly used, and thus configuration and other messages would be transmitted unencrypted, with their content trusted. This was – and still in 2024 largely is – the case for, *e.g.*, the Domain Name System (DNS) messages, used when translating a domain name into an IP address.

However, several events have shown that *trust is misplaced*.

In 1997 an Internet service provider, MAI Network Services, inadvertently caused a major Internet outage due to a router misconfiguration[12]. Misconfigured routers at MAI Network Services announced bad routing information to an Internet backbone operator, which then ended up redirecting all Internet traffic to MAI Network Services. This caused the network of MAI Network Services to be overwhelmed, and to consequently shut down, which created a major outage in the U.S., showing that *trust in router information is misplaced*.

In 1999, David Lee Smith released the Melissa worm, which propagated through email, attached to Word documents. It spread widely, with more than 300 corporations and government agencies having to completely shut down, and with an estimated 80 million dollars of loss, due to profits lost and the recovery of affected systems[13]. Similarly, the ILOVEYOU virus caused, in 2000, around 10 billion dollars worth of damage, with a very fast worldwide propagation[14]. Those examples showed that, as for the floppy disks of the decade prior, *trust in shared files is misplaced*.

This led to new directives, with the creation of frameworks and security programs, prompting for stronger cybersecurity [30], and the creation, in 2000, of the document *Defending America's Cyberspace: National Plan for Information Systems Protection*, signed by the United States President [31]. This

[Online]. Available: https://edition.cnn.com/2021/05/16/tech/ransomware-joseph-popp/index.html.

[10] K. Chadd, *The history of cybersecurity*, blog post, 2020. [Online]. Available: https://blog.avast.com/history-of-cybersecurity-avast.

[11] A. Terekhov, *History of the antivirus*. [Online]. Available: https://www.hotspotshield.com/blog/history-of-the-antivirus/.

[12] C. N. staff, "Router glitch cuts net access," *CNET*, 1997. [Online]. Available: https://www.cnet.com/tech/mobile/router-glitch-cuts-net-access/.

[13] F. News, *The Melissa virus*, 2019. [Online]. Available: https://www.fbi.gov/news/stories/melissa-virus-20th-anniversary-032519.

[14] R. S. Mueller III, "Protecting the U.S. economy in a global age," *FBI Speeches*, 2003. [Online]. Available: https://archives.fbi.gov/archives/news/speeches/protecting-the-u.s.-economy-in-a-global-age.

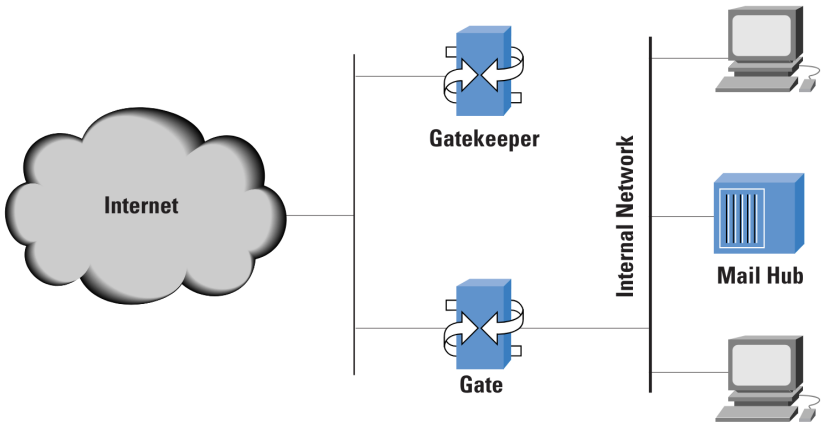


Figure 1.2: DEC SEAL, first commercial firewall (from [33]).

document presents the Federal Intrusion Detection Network (FIDNet), an intrusion detection system linked to the US Administration, to detect threats. Privacy concerns, however, prevented deployment of the solutions proposed in this document.

In parallel, because protecting machines with antiviruses was a constant war between security developers and attackers, perimetric security was developed. Their goal: to stop attacks before they reach target computers. Network-based defense solutions were outlined in the DoD *Information Operations Roadmap* from 2003 [32].

An IT system using a perimetric approach has up to hundreds of *firewalls* in their infrastructure [34]. The first firewall was designed by a NASA researcher in 1988, and the first commercial firewall, called 'Screening External Access Link' (DEC SEAL)[15], depicted in figure 1.2, was produced in 1991 by Digital Equipment Corporation [33].

In perimeter-based defense, firewalls implement two strategies: segmentation, and defense-in-depth [5]. Segmentation consists of dividing the network into several subnetworks, each with their own set of policies and security protocols, to prevent lateral movement of intruders and of other entities between subnetworks [5]. Defense-in-depth consists of using several layers of security control to prevent unauthorized access [6]. By combining segmentation and of defense-in-depth, multiple zones of trust, with different security levels, are created [73].

The usage of segmentation and defense-in-depth was recommended in 2006 by the DoD [35], was standardized in 2009 by the National Institute of Standards and Technologies (NIST) [3], and was still recommended by the Department of Homeland Security (DHS) in 2016 [4].

Moreover, end-to-end protections were also developed, *e.g.*, DNSSEC in 1997, designed to protect DNS messages [36].

By the year 2009, at least 14 mitigation techniques, *e.g.*, firewalls, intrusion prevention systems, network access control, encrypted virtual private networks,

[15]Some documentation also referred to it as Securing External Access Link

and web application firewalls, were used to protect networks and infrastructures[16].

Perimeter-based security was at the time found to be overall efficient against attacks such as malware, pishing, denial of service, and zero-day attacks [37].

1.1.5 The Era of Major Breaches

The years 2010s brought more and more sophisticated attacks: multi-vectors attacks, social engineering, advanced persistent threats, etc [7], [70]. Attacks targeted military and governmental agencies, as well as civilian companies – and as a consequence, regulations were designed to limit the impact of cyber threats, *e.g.*, the European General Data Protection Regulation (GDPR) of 2016. In 2010, the U.S. Cyber Command was created to answer to military cyber threats.

In 2010, the Stuxnet attack targeted Iranian nuclear plants, resulting in the destruction of several centrifuges. This, despite the nuclear plant being an air gapped environment, *i.e.*, that the network had no network interface connected to outside networks. In 2013, improper input validation enabled attackers to compromise 3 billion Yahoo! accounts, and in 2017, the Wannacry ransomware infected hundreds of thousands of personal computers [19]. All this occurred, *despite* perimetric security and firewalls being the largest market in the IT security sector, with the firewall market being estimated at 9.27 billions dollars[17]. Thus, those examples show that *trust in perimetric security is misplaced*.

Moreover, the evolution of practices, such as cloud computing, and work-from-home, has increasingly made it harder to precisely define perimeters, and to enforce them [8], [9], [38], [39], [74], [75], [40].

1.1.6 Towards Zero Trust

Trust in perimeter security is misplaced, largely because of insider threats and lateral movements attacks. Moreover, the dynamic and hybrid nature of networks of any given organization makes it hard to precisely define perimeters, and thus to rely on perimeter security. These considerations have prompted the need for a paradigm change for cybersecurity, which involves both architectural and philosophical transformations.

This paradigm change was postulated in 2004 by the Jericho Forum [76] with the concept of 'deperimeterization'. An early example of actual architecture is the Defense Information Systems Agency (DISA) 'Black Core' [41], published in 2007 and used by the DoD.

This concept of deperimeterization was extended in 2010 by Forrester, introducing the term 'zero trust': a network architecture which makes the notion of perimeter evolve, and in which no part of the network is inherently trusted [77]. Or, to put it differently: having gotten access to a given part of a network, *i.e.*, being 'inside a specific perimeter', does not imply any automatic trust from any other system.

[16] "Techradar^{tm} for security & risk professionals: Network threat mitigation, q3 2009," Forrester, Tech. Rep., 2009.

[17] Gartner, *Magic quadrant for enterprise network firewalls*, Jul. 2017. [Online]. Available: https://www.gartner.com/en/documents/3757665.

After a series of governmental cyber-attacks 'Operation Aurora' in 2009[18], Google created a zero trust transition project named BeyondCorp [78]–[80].

In the 2010s, US federal agencies followed the defense-in-depth recommendations from NIST and DHS. In December 2020, attackers targeted a software vendor, SolarWinds, and managed to insert a backdoor, named 'Sunburst', into Orion, its network monitoring tool. This backdoor was delivered to ten of thousands of Orion customers, including several US federal agencies and Fortune 500 companies[19]. This attack demonstrated that *trust in the supply chain is misplaced*, and it changed the perspective of the US government in favor of a zero trust approach to security [71], [81], [82]. The transition to zero trust was made compulsory by the White House in 2022, by way of the American President's Executive Order 14028[20] and the M-22-09 memorandum[21], both requiring US federal agencies to meet zero trust standards by the end of year 2024.

Outside of government, zero trust is also recommended as a target architecture [42], [138], for example in the medical field [43], or to help with regulations compliance such as GDPR [44].

Following the need for governmental agencies to adopt a zero trust approach to security, several standards have been proposed by NIST [83], by governmental agencies such as the Cybersecurity and Infrastructure Security Agency (CISA) [81], or the DoD [71], [82], by manufacturers such as Forrester [84], Microsoft [85], VMWare [86], Google [87], or Cloudflare [88], and by academic literature [73], [89]–[91], [128].

1.2 Methods for Defining Zero Trust

Zero trust is widely studied – and debated – in both academic literature and industry publications. As described in section 1.1, numerous security techniques have been developed to ensure computer security, leading to the emergence of the zero trust paradigm.

According to [130], the academic community has focused on formally defining zero trust architectures, while industry-oriented publications have emphasized presenting the benefits of zero trust and implementing migration strategies. Moreover, there are few studies on the economic benefits of zero trust, or on user experience.

The zero trust paradigm is characterized by several *tenets*, which are formally presented in [128], [132]. These tenets include identifying resources, granting access to resources per connection, based on dynamic policies, and always authenticating subjects, without relying solely on network location. Additionally, telemetry must be continuously collected to improve the security

[18] Google, *A new approach to china*, blog post, 2010. [Online]. Available: https://googleblog.blogspot.com/2010/01/new-approach-to-china.html.

[19] J. Rundle, "Solarwinds, microsoft hacks prompt focus on zero-trust security," *The Wall Street Journal*, 2021. [Online]. Available: https://www.wsj.com/articles/solarwinds-microsoft-hacks-prompt-focus-on-zero-trust-security-11619429402.

[20] J. Biden, *Improving the nation's cybersecurity*, Executive order 14028, 2021. [Online]. Available: https://www.federalregister.gov/documents/2021/05/17/2021-10460/improving-the-nations-cybersecurity.

[21] S. D. Young, *Moving the U.S. government toward zero trust cybersecurity principles*, Memorandum M-22-09, 2022. [Online]. Available: https://www.whitehouse.gov/wp-content/uploads/2022/01/M-22-09.pdf.

Table 1.1: Comparison of Zero Trust Surveys

Reference	[129]	[139]	[130]	[128]	[131]	[132]
Definition of zero trust including all tenets.	✘	✘	✘	✔	✘	✔
Categorization and review of main zero trust technologies.	✔	✘	✘	✔	✔	✔
Taxonomy of industrial and academic zero trust architectures.	✘	✘	✔	✘	✔	✘
Analysis of zero trust as a migration process.	✘	✔	✘	✘	✘	✔
Analysis of zero trust challenges.	✘	✔	✔	✘	✔	✔
Evaluation of zero trust benefits, drawbacks, and limits.	✘	✔	✔	✘	✘	✘

✔: discussed; ✘: not or partially mentioned.

posture of the organization. Those tenets are directly applied to data objects in [92].

To implement these zero trust tenets, various techniques and approaches are employed in architectures, through technologies. Thus, according to [131], a zero trust architecture is composed of three core technologies: identity authentication, access control, and trust assessment. Four fundamental technologies are required for successful implementation, according to [128]: authentication and access control, segmentation, encryption, and security automation and orchestration. The relationship between context-aware access control and zero trust is studied in [158]. For [132], automation plays is crucial role in zero trust architectures, with artificial intelligence helping to automate tasks. Blockchain-based IDS have also been explored as a means to potentially increase zero trust capabilities [133].

Migration to zero trust requires the integration of these technologies into existing architectures. This migration provides several benefits, including those presented in [129], but also presents challenges, which are discussed in [139].

The works mentioned above provide slightly different notions for zero trust, emphasizing distinct principles and technologies, as summarized in table 1.1.

1.3 Thesis Statement

This chapter so far has introduced the historical context that led to the emergence of the zero trust paradigm. Section 1.1 has shown how the transformation of organizations and networks, as well as the evolution of threats and cyber-attacks, have collectively contributed to the development and spread of zero trust. Nevertheless, section 1.2 revealed the diversity of zero trust definitions,

emphasizing the challenge of establishing a unique definition for zero trust, both as an abstract notion and as an architecture. Furthermore, the need for additional security models, *e.g.*, encompassing privacy, has arisen.

Therefore, this thesis seeks to address the need for a framework that provides comprehensive definitions and evaluations of IT security, in particular to zero trust architectures. More specifically, this manuscript contributes to the understanding of zero trust and IT security by:

- presenting a comprehensive survey of zero trust definitions and a taxonomy of existing zero trust architectures, leading to the creation of an evaluation framework for assessing the maturity levels of architectures with respect to zero trust;
- building a proof-of-concept zero trust architecture based on off-the-shelf products, serving as a foundation for further development;
- proposing a methodology for enhancing data-centric security within existing systems;
- investigating continuous authentication, a crucial capability for zero trust architectures, within the context of secure messaging protocols;
- exploring the notion that underlying infrastructure, *e.g.*, networks, can also be considered untrusted, thus providing organizations with enhanced privacy and security;
- developing a method for enabling multiple zero trust architectures to share resources while maintaining equivalent levels of security.

In sum: establishing a comprehensive definition of zero trust enables to reason about the security guarantees provided by zero trust architectures. It makes it possible for architects to position their architectures relative to their compliance with zero trust principles, thereby identifying specific goals that these architectures must meet. Moreover, the framework built for evaluating the maturity of zero trust architectures highlights gaps within the zero trust frameworks, enabling a continuous improvement of IT security, as is studied throughout this manuscript. This approach enables organizations to gain a deeper understanding of zero trust, while also offering additional flexibility and capabilities to enhance the overall security of organizations.

1.4 Thesis Contributions

This section concludes the introductory part of the manuscript by summarizing the thesis contribution in section 1.4.1, and by presenting a list of publications in section 1.4.2.

1.4.1 Thesis Summary and Outline

This thesis explores architectures for protecting organizations from cyber threats. It investigates in particular the concept of zero trust, of how this paradigm can be integrated into existing architectures to enhance overall security, and of how it can be extended to provide additional security capabilities

for answering real-world scenarios. The thesis comprises five parts and nine chapters, structured as follows.

Part I provides an introductory discussion. In chapter 1 (submitted as part of [430]), the historical context that led to the emergence of the zero trust paradigm is presented. The chapter delves discusses the challenges of defining zero trust and its potential benefits in augmenting the security of organizations.

Part II provides an in-depth analysis of the concept of zero trust. Chapter 2 (also submitted as part of [430]) presents a comprehensive survey of existing zero trust definitions and architectures from academia, governments, and industry publications. It proposes a comprehensive definition of zero trust, by analyzing its underlying principles, the migrations strategies for incorporating zero trust capabilities into existing architectures, and the multiple technologies that can be used to provide these capabilities. Moreover, this chapter provides an in-depth analysis of existing zero trust architectures, positioning them relatively to their adherence to zero trust principles.

With the requirements for building a zero trust architecture in mind, chapter 3 (submitted as part of [431]) demonstrates how a zero trust architecture can be build, by combining and modifying open-source products. The framework built in chapter 2 is used to position the proof-of-concept relative to zero trust principles.

The analysis of existing zero trust architectures and of research on zero trust architectures presented in part II highlights gaps in zero trust research. Part III aims to fill some of these gaps, by leveraging technologies and by providing methods for improving the zero trust maturity level of architectures. Some of these improvements are implemented within the proof-of-concept presented in chapter 3, illustrating how technologies can be integrated and interoperate with existing architectures.

More precisely, chapter 4 (submitted as [432]) presents a method for adding data-centric security to an existing architecture, by leveraging Attribute-Based Encryption (ABE). ABE combines data confidentiality with attribute-based access control, offering additional flexibility to organizations. The proposed approach enables the storage of data protected by distinct access control policies on a single server. Moreover, confidential data can be stored on untrusted servers, *e.g.*, offered by a cloud provider, and documents can be encrypted with parts having different access control policies.

Chapter 5 (presented in the Real World Crypto symposium[22] and published in [159]) studies in more details a core property of zero trust, continuous authentication, in the context of secure messaging protocols. The existing continuous authentication procedure is extended to enhance post-compromise security, allowing for the recovery of confidentiality and authenticity even if a peer in the communication has been compromised.

Part IV studies real-world scenarios that require zero trust architectures to be extended for providing sufficient security.

[22] https://rwc.iacr.org/2022/program.php

In chapter 6 (presented at the C&ESAR conference[23], published in [93], and with an extended version being submitted as [431]), a method for federating multiple zero trust architectures, while maintaining equivalent levels of security, is proposed. The main challenge is that in a federation, the verification of identity and monitoring of requesters are performed by their origin domain. Therefore, if a requester requires access to a federated resource, the domain protecting the resource must explicitly verify the identity and context of the requester, which is normally performed by the domain of the requester. However, in a zero trust environment, the domain of the resource cannot implicitly trust information provided by the domain of the requester. To address this challenge, this chapter proposes a method that enables the domain of the resource to verify this information, by leveraging remote attestation, without implicitly trusting the domain of the requester nor requiring intrusive methods that may not be applicable to real world scenarios.

Chapter 7 (under submission) investigates how the underlying infrastructure, *e.g.*, networks, can be considered untrusted. It proposes several methods for enabling datagrams forwarding by routers without allowing routers to learn their destination. The proposed solution provides privacy to the organization, with anonymity of communications.

Finally, part V concludes this manuscript. Appendix A presents an additional contribution performed during this PhD, presenting how to optimally distribute resources and requests within a data centre with network load-balancing. A summary in French of this manuscript is provided in appendix B.

1.4.2 List of Publications

The following publications were published or submitted during the course of this PhD.

Conference Publications

- B. Dowling, F. Günther, and A. Poirrier, "Continuous authentication in secure messaging," in *Computer Security – ESORICS 2022*, Springer Nature Switzerland, 2022, pp. 361–381. DOI: 10.1007/978-3-031-17146-8_18 (chapter 5).
- A. Poirrier, L. Cailleux, and T. H. Clausen, "An interoperable zero trust federated architecture for tactical systems," in *MILCOM 2023 - 2023 IEEE Military Communications Conference (MILCOM)*, IEEE, Oct. 2023. DOI: 10.1109/milcom58377.2023.10356247 (chapter 6).

Submitted articles

- A. Poirrier, L. Cailleux, and T. H. Clausen, "Is trust misplaced?" Submitted to *Proceedings of the IEEE*, 2024 (chapter 2).
- A. Poirrier and T. H. Clausen, "Privacy-preserving forwarding," *Under submission*, 2024 (chapter 7).

[23] https://2023.cesar-conference.org/

- A. Poirrier, L. Cailleux, and T. H. Clausen, "Building a zero trust federation," Submitted to *IEEE Journal on Selected Areas in Communications*, 2024 (chapter 6).
- A. Poirrier, Laurent, and T. H. Clausen, "Data-centric security protections in zero trust architectures," Submitted to *MILCOM 2024 - 2024 IEEE Military Communications Conference (MILCOM)*, 2024 (chapter 4).
- A. Poirrier and T. H. Clausen, "Top-of-rack-assisted load-aware and server-agnostic load-balancing," Submitted to *2024 IEEE 32nd International Conference on Network Protocols (ICNP)*, 2024 (appendix A).

Part II

Zero Trust Architectures

Chapter 2

What Is Zero Trust?

Information Technologies (IT) security has been, and largely is, based on compartmentalization. To implement compartmentalization, system access privileges are granted depending on the topological location of systems, grouped into *perimeters*, with network mechanisms (firewalls, VLANs, ...) enforcing isolation between perimeters, thus implicitly trusting systems based on their location. However, as presented in section 1.1, history has shown that such trust is misplaced. This has led to the emergence of an alternative paradigm, called *zero trust*.

Statement of Purpose

This chapter proposes a taxonomy of zero trust, by analyzing zero trust definitions, existing zero trust architectures, and zero trust evaluations. Definitions, migration strategies, core principles and capabilities of zero trust architectures are compared, and core technologies for building zero trust are reviewed. This taxonomy establishes a postulated 'ideal' zero trust architecture, that existing architectures aim at approaching through zero trust migration. This chapter also provides a method for positioning zero trust architectures relatively to this 'ideal' concept of zero trust, by estimating their zero trust maturity. Finally, this chapter analyzes the benefits, and drawbacks, of zero trust, focusing on the security properties granted by zero trust, as well as the vulnerabilities introduced.

Chapter Outline

The remainder of this chapter is organized as follows. Section 2.1 explains zero trust, by presenting a taxonomy of zero trust definitions and models. Several core principles derive from the main zero trust goal, which are presented in section 2.2. To follow zero trust principles, existing architectures migrate to zero trust by implementing zero trust capabilities. The migration process, and zero trust maturity models, which describe to what extent an architecture follows zero trust principles, are presented in section 2.3. Zero trust capabilities and technologies, which enable architectures to reach higher levels of zero trust maturity, are presented in section 2.4. Then, section 2.5 establishes a taxonomy of existing zero trust architectures, by describing how technologies are assembled to form complete architectures. That section positions each surveyed archi-

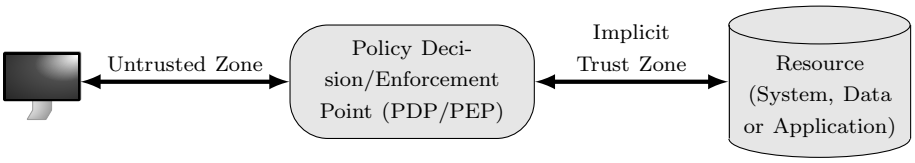


Figure 2.1: Zero trust access (from [83]).

tecture relatively to the ideal concept of zero trust. Section 2.6 assesses the benefits of zero trust, while presenting the challenges posed by zero trust migration and the limits of zero trust architectures. Finally, section 2.7 concludes this chapter.

2.1 Zero Trust Definition

Introduced in 2010 by Forrester [77], zero trust aims to provide solutions to the pitfalls of the perimetric security approach: that it is impossible to trust network portions or packets, that verifications are seldom performed within a perimeter, and that malicious insiders thus often are in position of trust that can be abused.

This zero trust model for computer security aims at producing architectures following the motto 'never trust, always verify'. It means that, to reach the goal of having no unauthorized access nor unauthorized use of a resource, every access is to be explicitly verified, without implicit trust.

One of the first architecture considered to be following the zero trust principle is the 2007 Black Core architecture [41]. It protects every resource with a physical gateway, which intercepts traffic for access control verification, and continuously ensures the correctness, availability, and secrecy of the information provided.

In 2014, the Cloud Security Alliance (CSA) published an architecture inspired from Black Core, with software-defined gateways replacing physical gateways for protecting resources [94]. Dynamic, systematic, and continuous verification before granting access ensures the architecture follows the zero trust goal.

After the 2009 cyber-attacks 'Operation Aurora'[1], Google launched a migration to a zero trust architecture called BeyondCorp [78]. In this architecture, every connection goes through an access proxy [79], which explicitly verifies the connection request, and grants or denies access to resources, thus following the aforementioned zero trust motto. Nevertheless, transition to a full zero trust architecture is complex, as, according to [140], by 2023 that migration had yet to complete.

In 2020, NIST [83] published a standard for zero trust architectures, wherein zero trust provides an answer to an access problem: a *subject* needs access to an enterprise *resource*. To follow the zero trust motto, every access request is explicitly verified by a Policy Decision Point (PDP), and the access decision

[1] Google, *A new approach to china*, blog post, 2010. [Online]. Available: https://googleblog.blogspot.com/2010/01/new-approach-to-china.html.

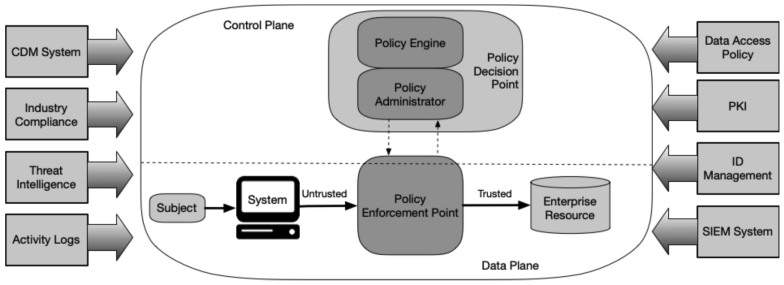


Figure 2.2: Core zero trust logical components (from [83]).

is enforced by the corresponding Policy Enforcement Point (PEP), as depicted on figure 2.1.

More precisely, NIST [83] defines an abstract architecture for zero trust, depicted in figure 2.2. The PDP is separated into a Policy Engine (PE), which takes access decisions, and a Policy Administrator (PA), which applies the decision of the PE on the infrastructure. The PE implements a trust algorithm, which takes into account information on the access request, on the subject performing the request, and on the resource, the access policy, and contextual information, to take the access decision. Those inputs come from the logical components presented on either side in the figure.

Following the NIST zero trust framework, other American agencies, such as CISA [81], or the DoD [82], published their guidelines for migrating to zero trust architectures. Those documents present techniques for increasing the awareness and visibility of assets and resources, for explicitly verifying connections.

Synthesis

From those architectures and standards, a zero trust architecture is defined as an architecture that follows the zero trust motto 'never trust, always verify'. The goal is to build architectures ensuring that every access to a resource is explicitly verified, without implicitly trusting the entity making the request or its device – and this, even if it is connected to a known network, or if it was granted access in the past. To do so, an architecture implements isolation mechanisms that prevent unauthorized access to resources. Authorizations are granted following a dynamic trust algorithm, that evaluates if the entity requesting access and its device are trustworthy, based on awareness and visibility of assets and entities within the infrastructure.

2.1.1 Threat Model

Zero trust architectures aim at preventing unauthorized access to and unauthorized use of resources. The MITRE ATT&CK Enterprise Framework [45] models threats and identifies vulnerabilities in architectures, by creating heatmaps and risk registers, categorizing cyberattacks and evaluating the security of endpoints [95]. That framework defines several phases for attacks: reconnaissance, resource development, initial access, execution, persistence, privilege escalation, defense evasion, credential access, discovery, lateral movement, collection, command and control, exfiltration, and impact. Attacks are composed of sev-

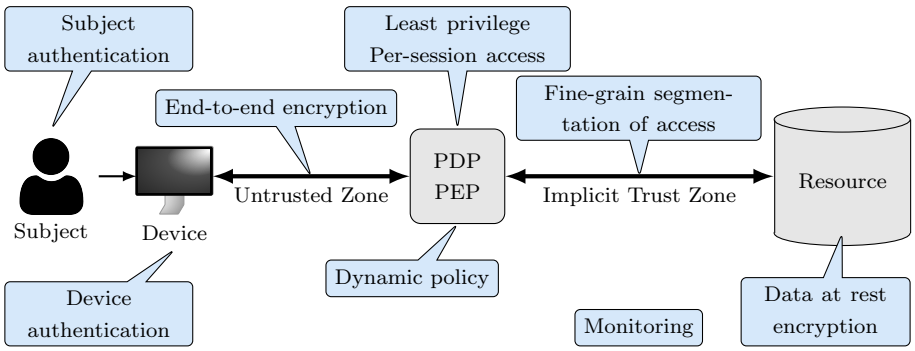


Figure 2.3: Zero trust principles.

eral phases. Therefore, the goal of zero trust is to prevent attacks from being successful, *i.e.*, preventing at least one phase in attacks [96]. Thus, an example zero trust architecture, which prevents reconnaissance, initial access and lateral movement, is presented in [96].

The zero trust motto, 'never trust, always verify', means that zero trust aims at protecting resources against both external and internal threats. The internal network is not trusted any more than external networks, and connected devices may not be owned or managed by the enterprise. Resources are not inherently trusted, and may not be limited to enterprise-owned infrastructure.

2.2 Zero Trust Core Principles

As discussed in section 2.1, zero trust models and architectures explicitly verify every access to a resource, without relying on implicit trust. This explicit verification is made possible by following a set of core principles, for which a taxonomy is presented in this section. Every zero trust model and architecture in the literature do not follow the exact same core principles, thus a comparison is provided in this section, summarized in table 2.1.

In its original zero trust model, Forrester proposes three core principles for zero trust architectures: all resources are accessed securely regardless of location, access is strictly enforced following least privilege strategy, and all traffic is inspected and logged [77].

In the Google BeyondCorp architecture [78], every access to an enterprise resource needs to be fully authenticated, authorized and encrypted, with a fine-grained access, by taking into account the requestor device state and user credentials.

Microsoft [85] presents three core principles for zero trust: 'verify explicitly', meaning that security decisions need to be performed using every available data, 'use least privilege access', meaning that access is limited in time and width, and 'assume breach', meaning that lateral movement needs to be hindered. The NSA [70] also considers those as the three core principles of zero trust.

NIST [83] defines the ideal zero trust architecture as an architecture following seven basic principles: (i) all data sources and computing services are

Table 2.1: Comparison of Zero Trust Core Principles.

Reference	(1)	(2)	(3)	(4)	(5)	(6)
Forrester [77]	+	++		++		++
BeyondCorp [78]	++		++	++	++	
Microsoft [85]	++	++	++	++	++	+
NIST [83]	++	++	++	++	++	++
DoD [82]	++		++	++	++	++
MIT report [71]	++	++	++	++	++	++
Gilman and Barth [73]	+	++	++	++	+	++
He et al. [131]	++	++	++	++	+	+
Yan et al. [129]	+	++		+	+	++
Alevizos et al. [133]	++	++	+	++	++	++
Cao et al. [132]	++	++	++	+	+	++

Zero trust principles:

(1) Subject and device authentication.

(2) Least-privilege, per-session authorization.

(3) Dynamic access policy.

(4) Fine-grain authorization and segmentation of access.

(5) Encryption.

(6) Monitoring.

++: core principle; +: present but not a core principle.

considered resources; (ii) all communication is secured regardless of network location; (iii) access to individual enterprise resources is granted on a per-session basis; (iv) access to resources is determined by dynamic policy; (v) the enterprise monitors and measures the integrity and security posture of all assets; (vi) all resource authentication and authorization are dynamic and strictly enforced before access is allowed; and (vii) the enterprise collects as much information as possible about the current state of assets, network infrastructure and communications, to improve its security posture. This definition from NIST is the base for numerous zero trust definition, such as definitions from surveys [128], [130], or from CISA [97].

The DoD [82] defines five tenets for zero trust: assume a hostile environment; presume breach; never trust, always verify; scrutinize explicitly; and apply unified analytics. Those tenets are translated into seven principles: (i) assume no implicit or explicit trusted zones in networks; (ii) identity-based authentication and authorization are strictly enforced for all connections and access to infrastructure, data, and services; (iii) machine-to-machine authentication and authorization are strictly enforced for communication between servers and the applications; (iv) risk profiles, generated in near-real-time from monitoring, and assessment of both user and devices behaviors, are used for

authorizing users and devices access to resources; (v) all sensitive data is encrypted, both in transit and at rest; (vi) all events are to be continuously monitored, collected, stored, and analyzed, to assess compliance with security policies; and (vii) policy management and distribution is centralized.

A report from the MIT for the US government [71] presents five core principles for defining zero trust: universal authentication, access segmentation, minimal trust authorization, encryption everywhere, and continuous monitoring and adjustment.

According to [73], zero trust is defined by five principles: using identity as the basis of access control, using the least privilege principle for resource allocation, real-time calculation of access control strategy, only allowing controlled and secure access to resources, and continuous evaluation of trust level from multiple data sources.

Zero trust adheres to four basic principles according to [131]: authenticate users, authenticate devices, restrict access and permissions, and adaptivity.

According to [129], zero trust architectures are based on three principles: no trusted domains, strictly enforced access control using minimal privilege, and checking and recording of all network traffic.

As presented by [133], zero trust has five principles: access segmentation, universal authentication, encrypt as much as possible, least-privilege principle, and continuous monitoring and adjusting.

Finally, according to [132], zero trust is defined following three principles: resource-centric and context-aware access control, authentication and authorization of users and devices based on least-privilege dynamic policies, and security improvement by continuous monitoring of the integrity and security of owned and associated assets.

2.2.1 Synthesis

Despite the different terms and weights attributed to zero trust principles by the existing zero trust models and architectures discussed, a common core can be identified, depicted in figure 2.3:

1. **Subject and device authentication**: Human and non-human entities, as well as their device, interacting with the organization infrastructure needs to be continuously authenticated;

2. **Least-privilege, per-session authorization**: authorization is granted with the least amount of privilege for a limited time;

3. **Dynamic access policy**: authorization takes into account the environment and current context;

4. **Fine-grain authorization and segmentation of access**: access to resources is evaluated, and authorized, for smallest possible pieces;

5. **Encryption**: data is encrypted in transit, and at rest;

6. **Monitoring**: the infrastructure, entities, and resources are constantly monitored for improving security.

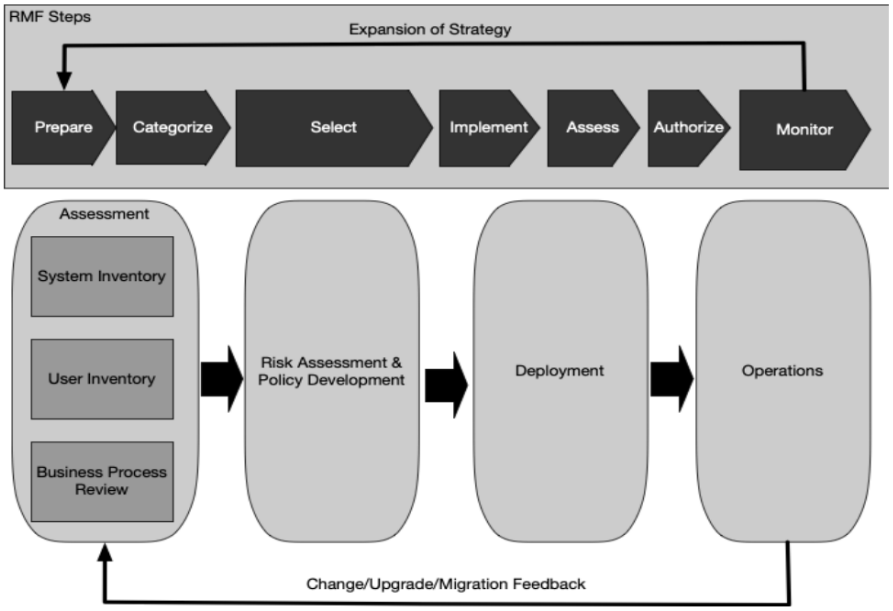


Figure 2.4: Zero trust transition cycles (from [83]).

2.3 Migrating to a Zero Trust Architecture

Zero trust is an **architecture** – but it is also a **process**: one of transforming existing IT architectures into zero trust architectures, that is, into architectures that follow the core principles of zero trust.

The extent to which those core principles are followed in a given architecture can be measured, to determine the *zero trust maturity* of an architecture. The goal of measuring the zero trust maturity of an architecture is twofold: first, it measures progress, and highlights remaining efforts that need to be performed, until the objective of an 'ideal' zero trust architecture is reached. Second, it supports the migration, by providing milestones and goals for following zero trust core principles.

This section presents a taxonomy of zero trust migration models.

2.3.1 Migration Process

Migration to a zero trust architecture is performed in four steps[2]: (i) a preparation phase, evaluating the existing architecture; (ii) a planning phase, for deciding which changes need to be incorporated into the existing architecture; (iii) an implementation phase, for incorporating those changes; and (iv) a testing phase, to validate these changes. According to a dozen of zero trust solutions vendors[3], those steps require: developing inventories (of data, of applications, and of assets), auditing and logging existing traffic and architecture, as well as

[2]T. Morrow and M. Nicolai, *The zero trust journey: 4 phases of implementation*, blog post, Jun. 2022. [Online]. Available: http://insights.sei.cmu.edu/blog/the-zero-trust-journey-4-phases-of-implementation.

[3]M. Nicolai, N. Richmond, and T. Morrow, "Industry best practices for zero trust architecture," 2022. [Online]. Available: https://apps.dtic.mil/sti/pdfs/AD1187390.pdf.

every change made towards zero trust, evaluating risk, leveraging governance, and using automation and orchestration.

An alternative view of this migration process decomposes it into six phases: strategizing zero trust, assessing the context, architecting zero trust, transforming to zero trust, monitoring and maintaining, and optimizing zero trust security [72]. Changes are thoroughly documented, taking into account risks, laws, and regulations [139].

As an illustration, [78] presents how the transition by Google to BeyondCorp has been performed. Several cycles, each involving a different part of the organization, were needed, with each cycle being composed of several phases: an observation phase to assess the resources and workflows in use; a hybrid phase, with the zero trust technology deployed, but with legacy traffic still allowed yet discouraged; and finally an enforcement phase, in which access is always performed using the added zero trust technology. According to [140], this migration process, which started in 2014, was still ongoing in 2023, because of processes difficult to integrate in the zero trust architecture, thus needing a flexibility for those legacy services.

A migration process, working in cycles, has been standardized by NIST [83]. Developments are performed in cycles, leading to hybrid zero trust and perimeter-based architectures. Each cycle is composed of four steps, depicted in figure 2.4:

1. **Assessment**: inventory of assets, subjects, data flows, and workflows;

2. **Risk assessment and policy development**: identification of key processes, evaluation of risks associated with business processes, formulation of policies for zero trust candidate technologies, and identification of a candidate zero trust solution;

3. **Deployment**: implementation and deployment of the selected zero trust solution, while still authorizing legacy traffic, and monitoring to assess the success of the transition;

4. **Operation**: the solution deployment is complete, and is continuously monitored.

2.3.2 Maturity Models

Assessing the zero trust progress of an architecture is performed with *maturity models*. A maturity model presents the milestones that need to be integrated into the architecture in order to approach an 'ideal' zero trust.

Each migration cycle moves the architecture towards 'ideal' zero trust. These advancements can be performed on several axes, called *pillars* [134]. Each pillar can progress at its own pace, until cross-pillar coordination is required [97]. Advancing a pillar improves the zero trust maturity level of the architecture for that pillar. A comparison of pillars, considered by the zero trust models in the literature, is provided in this section, summarized in table 2.2.

NIST [83] provides three axis on which improvements towards zero trust can be made: enhanced identity governance, micro-segmentation, and network infrastructure and software defined perimeters.

Table 2.2: Comparison of Zero Trust Pillars.

Zero Trust Pillar	[83]	[97]	[85]	[82]	[84]
Identity	✔	✔	✔	✔	✔
Device	✔	✔	✔	✔	✔
Infrastructure, Network, Environment	✔	✔	✔	✔	✔
Workloads, Applications		✔	✔	✔	✔
Data		✔	✔	✔	
Visibility and Analytics	✔	✔	✔	✔	✔
Automation and Orchestration		✔		✔	✔
Policy	✔		✔		
Governance	✔	✔			

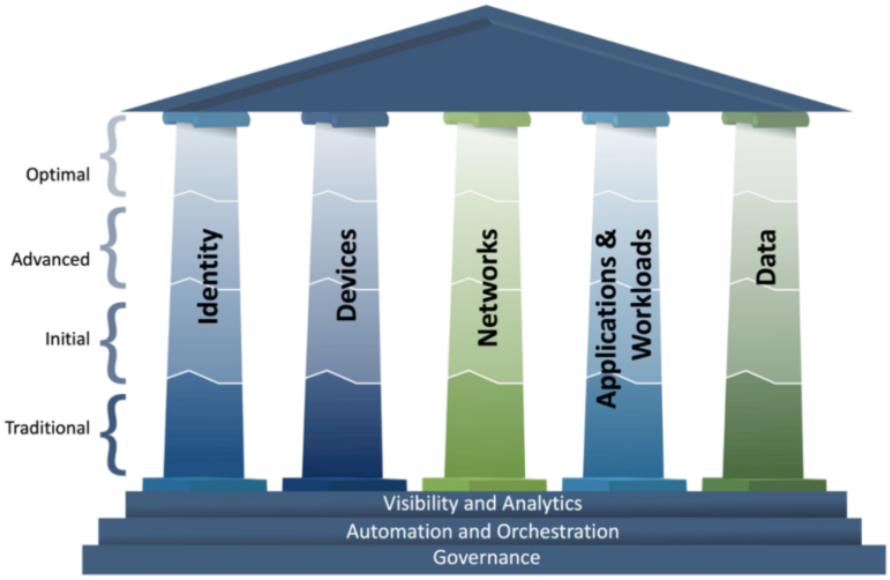


Figure 2.5: CISA zero trust maturity model (from [97]).

CISA presents a zero trust maturity model [97], depicted in figure 2.5. When first released in 2021, it was composed of three maturity stages: 'traditional', 'advanced', and 'optimal' [81]. An updated model from 2022 [97] presents four maturity stages, with the 'initial' stage added between 'traditional' and 'advanced'. Those maturity stages evaluate the maturity of five pillars: identity, device, networks, applications and workloads, and data. These pillars are supported by visibility and analytics, automation and orchestration, and governance.

In the same spirit, Microsoft [85] considers three stages of development: 'getting started', which is a first stage towards zero trust; 'advanced', where significant progress has been made; and 'optimal', which is the most mature stage. Pillars considered by Microsoft are identities, endpoints, networks, applications, data, infrastructure, policy optimization, policy enforcement, and threat protection. The NSA [70] also presents a maturity model with four

stages, one without zero trust: 'preparation', and three stages with zero trust: 'basic', 'intermediate', and 'advanced'.

Both the DoD [82] and Forrester [84] consider the same zero trust pillars: users, devices, networks and environment, applications and workloads, data, visibility and analytics, and automation and orchestration. They both consider data as the center pillar, but while for the DoD other pillars are at the same level, Forrester considers visibility and analytics, and automation and orchestration, to be transverse capabilities supporting the pillars: users, devices, networks, and workloads.

2.3.3 Synthesis

To summarize, zero trust is a migration process supporting the metamorphosis of existing architectures towards a zero trust architecture. This process is incremental, decomposed into several stages, each stage involving the assessment of the existing architecture, the design of a plan of action tailored to the organization needs, and the implementation and deployment of zero trust changes, while continuously monitoring the impact of the migration.

Attaining the zero trust core principles presented in section 2.2 is achieved through the development of zero trust pillars: **identity; device; infrastructure, network, and environment; workloads and applications; and data**, each pillar having a traditional, basic, advanced, or optimal zero trust maturity. **Visibility and analytics, automation and orchestration, policy, and governance** support those pillars, as transverse pillars. An architecture with basic zero trust maturity has limited visibility, limited segmentation, manual and static configurations, and manual incident response; advanced zero trust maturity means cross-pillar coordination, real-time analytics for dynamic policies, automatic incident response, unified identity, segmentation of access, and least-privilege policies; optimal zero trust maturity means fully automated and dynamic policies, fully automated incident response and mitigation, and centralized visibility.

2.4 Zero Trust Capabilities and Technologies

Zero trust maturity models provide guidelines for making architectures meet zero trust principles. Improving the zero trust maturity of an architecture is performed through the integration of one or several technologies [83].

Technologies enable zero trust *capabilities*. A capability is the ability to achieve a desired effect under specified (performance) standards and conditions, to perform a set of activities [82].

For example, as NIST describes [83], PKI and ID management technologies enable the continuous authentication capability, required for the identity and device pillars. They enable the architecture to follow the 'authentication' core principle of zero trust architectures.

CISA [97] provides a list of capabilities for every zero trust pillar, depicted in figure 2.6. For example, the identity pillar requires multifactor authentication (MFA), identity federation, continuous validation and real-time machine learning analysis for having optimal maturity. The capabilities proposed by Microsoft [85] are similar.

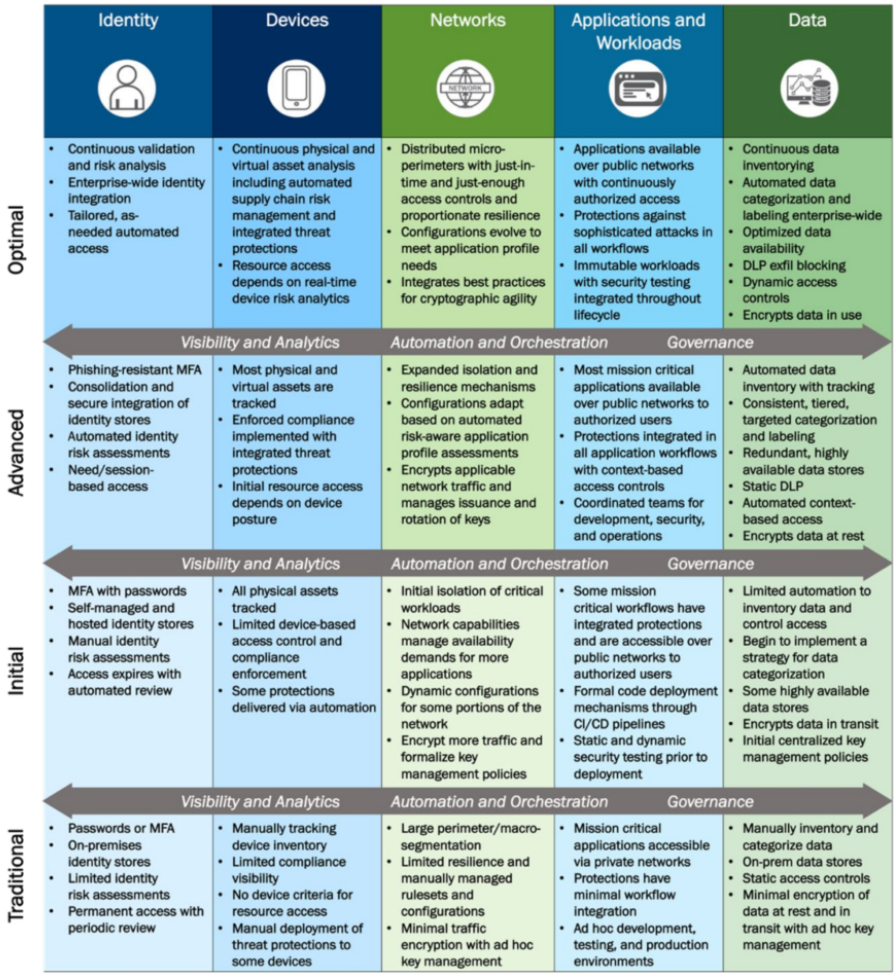


Figure 2.6: CISA Technologies for each pillar and maturity level (from [97]).

The DoD [82] also associates capabilities to zero trust pillars, illustrated in figure 2.7, and distinguishes categories for capabilities: continuous authentication, conditional access authorization, enabling infrastructure, securing application and workload, securing data, automation, analytics and orchestration.

Capabilities are categorized in [71] into the following categories: network traffic filtering, network access control, local system access control, application segmentation and execution control, operational and forensic analysis, network encryption, and trust and policy engine. Forrester presents several capabilities that zero trust architectures need to have: network segmentation, parallelization of switching cores, and central management [98]. Moreover, capabilities in network analysis and visibility are required, in conjunction with security information management [77].

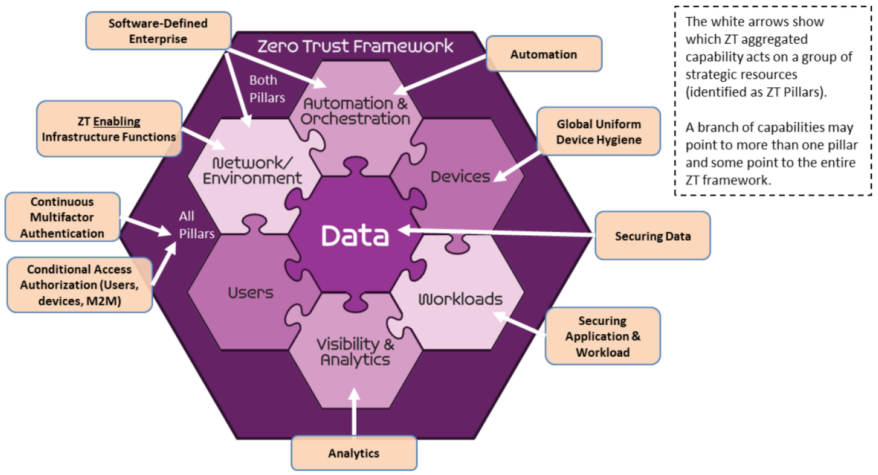


Figure 2.7: DoD Pillars and Capabilities (from [82]).

Table 2.3: Zero Trust Capabilities.

Pillar	Capabilities
Identity	User authentication Application and workload authentication Device authentication Identity management
Device	Device validation
Network	Micro-segmentation End-to-end encryption
Workloads, Applications	Isolation
Data	Data access segmentation Data encryption at rest
Policy	Access control and authorization
Visibility and Analytics	Monitoring and mitigation Threat intelligence
Automation and Orchestration	Security orchestration

Synthesis

Every zero trust capability is enabled by one or several technologies. Those links are summarized in table 2.3.

The remainder of this section presents, for every zero trust capability, a taxonomy of technologies enabling zero trust capabilities.

2.4.1 Zero Trust Capabilities: Authentication and Identity Management

One of the core pillars of zero trust architectures is authentication: every entity, device, and resource requires authentication. Entities may either be human, and are then called *users*, or non-human entities, *e.g.*, an application. They interact with other components in the architecture using devices.

Identity refers to an attribute, or a set of attributes, that uniquely describe a subject – user, entity, or device – within a given context [97]. Authentication is the process of verifying, and establishing confidence in, identities [160]. It is distinct from authorization, which grants or denies access to resources, relying on authentication for making access decisions.

User Authentication

User authentication needs to answer several challenges. First, users need to be uniquely and unambiguously authenticated [82], [83], [97]. According to [97], their identity needs to be unified and centrally managed across all services in the organization, for designing coherent access policies. Moreover, authentication needs to be continuously assessed and validated [82], [85], [97].

Authentication methods can be grouped into several categories [128]:

- Traditional methods, such as passwords, or physical biometrics (fingerprints, face recognition, iris scans, ...).
- Multifactor authentication (MFA), combining several authentication factors (knowledge, ownership, or inherence).
- Context-aware user authentication, using contextual information (*e.g.*, location, time, or behavior) to assign a confidence score on the user identity.
- Continuous authentication, continuously re-authenticating users throughout a session.

Passwords alone do not offer enough guarantees for authentication, because to guarantee security, users would need to choose passwords with enough entropy, different for every accessed service, and regularly changed. However, such passwords are hard to remember, resulting in some users choosing easy passwords, which can be stolen or guessed [161]. Moreover, sophisticated side-channel attacks break even strong passwords [162]. Similarly, physical biometrics alone can be bypassed [163].

With Multi-Factor Authentication (MFA), users are identified using several factors, amongst knowledge (such as a password), ownership (of a special hardware device, of a cryptographic key, or of a smartphone for example), and inherence (biometric information) [164].

Projects such as OATH[4], FIDO[5], and U2F[6] provide standardized and interoperable MFA solutions.

MFA schemes that do not require passwords are called 'passwordless authentication schemes'. They are either based on biometric information [165], or on ownership of cryptographic material, such as a private cryptographic key, used for example in WebAuthn[7] for web-based technologies, or in Secure SHell (SSH) public key authentication [166]. The FIDO2 specification[8] provides one standard for passwordless authentication.

Continuous authentication enables users to be regularly authenticated throughout a session. However, it is not a universal authentication method as it authenticates a user only for a specific device or scenario. Therefore, several methods need to be combined for creating viable zero trust authentication [167]. A first method for implementing continuous authentication is based on biometric information, in which users re-authenticate more easily using biometric information – *e.g.*, using a fingerprint instead of typing again a password – once the previous authentication has expired [168]–[170]. Alternatively, continuous authentication can be performed without action from the user, using as basis the context, or user unique behavior [171], such as the tilt produced by a phone when a user walks [172], [173], or their typing, or screen touching, patterns [174].

Adaptive authentication takes monitoring results into account to make authentication risk-based [175].

Applications and Workloads Authentication

A first authentication method for workloads is *workload tagging*: an already authenticated application, or a device, adds an *identification tag* to the workload, acting as an identity token [176]. This creates a hierarchical authentication: if the application or device tagging the workload is already authenticated, and is trustworthy, then the authentication of the workload is trustworthy. Hierarchical authentication enables the architecture to rely on an existing authentication system (for applications or devices), without needing to create another authentication system dedicated to workloads. Zero trust architectures such as Cilium[9] or Trireme[10] use this technique.

There are multiple approaches for authenticating applications. Similar to users, ownership is a factor of authentication for applications. For example, HTTPS web servers rely on ownership of a private cryptographic key, related to a public certificate, for proving their identity [177]. An equivalent to inher-

[4]OpenAuthentication, *OATH reference architecture, release 2.0*, 2007. [Online]. Available: https://openauthentication.org/specifications-technical-resources/.

[5]S. Srinivas and J. Kemp, "FIDO UAF architectural overview," FIDO Alliance, Tech. Rep., 2013. [Online]. Available: https://fidoalliance.org/specs/fido-uaf-v1.2-id-20180220/FIDO-UAF-COMPLETE-v1.2-id-20180220.pdf.

[6]S. Srinivas, D. Balfanz, E. Tiffany, *et al.*, "Universal 2nd factor (U2F) overview," FIDO Alliance, Tech. Rep., 2015.

[7]J. Bradley, C. Brand, T. Cappalli, *et al.*, "Web authentication: An API for accessing public key credentials level 3," Web Authentication Working Group, Tech. Rep., 2023. [Online]. Available: https://w3c.github.io/webauthn/.

[8]F. Alliance, *FIDO2*. [Online]. Available: https://fidoalliance.org/fido2/.

[9]Isovalent, *Cilium*, 2018. [Online]. Available: https://cilium.io/.

[10]Aporeto, *Trireme*, Github, 2018. [Online]. Available: https://github.com/aporeto-inc/trireme-lib.

ence for applications is implicit authentication, which is based on application fingerprinting, for example by monitoring their system calls [178], or by sending a challenge to verify the application behavior [179]. Continuous authentication builds upon implicit authentication to continuously authenticate applications, which enables to re-authenticate applications regularly, potentially limiting the time of undetected compromission of an application.

Device Authentication

Similarly to entity authentication, devices on which entities interact need to be continuously and universally authenticated, in order to follow zero trust principles [83], [97].

Device authentication methods are split into three categories [128]:

- certificate-based: the identity of a device is a cryptographic certificate, the device proving its identify by proving ownership of the associated secret key;
- supplier-based: the device supplier installs a specific module which embeds the identity of the device;
- fingerprint-based: equivalent to human biometry, based on unique physical characteristics of devices.

Multifactor authentication for devices combine several of those factors for authentication [180].

Certificates are installed by human operators, who are the root of trust for authentication [181].

Device manufacturers propose to include specific hardware for uniquely identifying devices[11] [182], [183]. *Onboarding* those devices into the architecture, *i.e.*, using the identity provided by the manufacturer to authenticate the device into the architecture, is ideally an automated and scalable process [184]. Therefore, in 2024, a standard for device onboarding was being drafted by NIST, to provide standard interfaces between manufacturers and client architectures [185]. Similarly, the FIDO Alliance has proposed a standardization for onboarding[12], enabling architectures to link secrets installed in devices by manufacturers to identity providers present in the architecture.

There are several ways to derive fingerprints to identify devices. The first one is to use Physical Unclonable Functions (PUF) [186]–[191]. They are functions based on unique physical characteristics of the channel between the authenticating device and the authenticator, which cannot be reproduced by entities other than the device [192], [193]. Implicit authentication, based on context [194]–[197], networks [198], or other sources [199]–[201], is another way to authenticate devices uniquely.

[11] INTEL®, *Intel® secure device onboard*, 2019. [Online]. Available: www.intel.com/securedeviceonboard.

[12] G. Cooper, B. Behm, A. Chakraborty, *et al.*, "FIDO device onboard specification 1.1," FIDO Alliance, Tech. Rep., 2021. [Online]. Available: https://fidoalliance.org/specs/FDO/FIDO-Device-Onboard-PS-v1.1-20220419/FIDO-Device-Onboard-PS-v1.1-20220419.pdf.

Internet of Things (IoT) networks are composed of devices with limited computing resources, called 'things', and typically have numerous devices. Because things have limited computing capabilities, computing may need to be performed by the authentication system [99]. To prevent the authentication (and authorization) system to be a single point of failure, due to the high number of devices that require authentication, it is possible to distribute it to several sub-systems. However, according to [97], authentication requires to be centrally managed, for enforcing coherent access policies. One possibility to enable this is the use of blockchain, using a distributed authentication and authorization system [202], [100]. However, blockchain requires the use of asymmetric cryptography, which may consume too many computing resources in IoT devices.

To reduce the cost of cryptographic operations in IoT devices, lightweight cryptography [203], [204] has been developed, as a special kind of cryptographic procedures dedicated to computationally limited devices.

Identity Management

Entities and devices may have different digital identities, depending on the context in which they are authenticated. For those identities to be stored, monitored, and managed, an Identity Management system is used [205]. An *Identity Management* (IdM) system is a framework that manages a collection of identities, their authentication, their use, and the information related to the identity [206]. A general IdM system involves three parties: a service provider, an entity accessing a service, and an identity provider [207].

There are three types of IdM models: isolated, centralized, and federated [208]. In an isolated IdM, the service provider is also an identity provider: identity storage and user operations are performed by the same entity, *e.g.*, an active directory [209]. In a centralized IdM, the identity provider manages identity storage and authentication, and all service providers use the same identity provider. Entities authenticate to several services via the same identity provider, by using the Single Sign-On (SSO) technology [210]. A federated IdM is built upon is a set of standards that enables service providers to recognize identities from different identity providers.

Synthesis

Authentication is a main pillar of zero trust, whose main technologies are summarized in table 2.4. 'Strong' authentication is enabled by combining several technologies to establish confidence in identities. Higher levels of confidence are reached by combining several factors of authentication, by taking into account the context, and by authenticating entities, devices, and resources continuously. Authentication interoperates with other zero trust technologies through identity managers.

2.4.2 Zero Trust Capability: Device Validation

Entities use devices to access resources. To follow the zero trust principles, to grant or deny access to resources, the access policy needs to take into account the security posture of the device, and its compliance with organization policies.

Table 2.4: Zero Trust Identity Technologies.

Capability	Technologies
User Authentication	Multi-factor authentication Context-aware user authentication Continuous authentication
Applications and Workloads Authentication	Workload tagging Certificate-based authentication Application fingerprinting Continuous authentication
Device Authentication	Certificate-based authentication Supplier-based authentication Fingerprint-based authentication
Identity Management	Identity Managers

Verifying the security posture, and the compliance with organization policies, of devices is called *device validation*, and is performed using *Mobile Device Management* (MDM) [211].

MDM ensures that managed equipment are configured following the company standards (*e.g.*, application whitelisting, or antivirus configurations), verifies their correct usage, and performs automatic updates. It also enables organizations to perform an inventory of their managed devices. The Open Mobile Alliance has specified a platform independent device management protocol called OMA Device Management[13].

The Bring Your Own Device (BYOD) paradigm requires the architecture to be able to secure resources even in the presence of unmanaged devices [212]. This is addressed by technologies such as Mobile Application Management (MAM), which supervises organization *applications* instead of devices, Unified Endpoint Management (UEM), which also monitors unmanaged devices, or a mix of MDM, MAM, and UEM, called Enterprise Mobility Management[14]. Dedicated monitoring solutions for IoT devices are also available [213], [214], for monitoring and ensuring the compliance of IoT devices while preserving their computing capabilities.

2.4.3 Zero Trust Capability: Micro-Segmentation

In perimeter security, firewalls and virtual LANs are the basis for segmentation. Micro-segmentation goes a step further, by placing each device, or even each application, within its own segment [71], [95]. A resource, or group of resource, in a segment is protected from other components in the architecture by a gateway security component, *e.g.*, a next-generation firewall, a network gateway, or a software agent [83]. Those gateways dynamically grant access to individual entities [128].

[13] O. M. Alliance, *OMA specifications*. [Online]. Available: https://technical.openmobilealliance.org/index.html.

[14] L. Mearian, "What's the difference between MDM, MAM, EMM and UEM?," *Computer World*, 2017. [Online]. Available: https://www.computerworld.com/article/3206325/whats-the-difference-between-mdm-mam-emm-and-uem.html.

Micro-segmentation has four architectural models[15]: native, third party, overlay, and hybrid. Native micro-segmentation uses the inherent capabilities of the platform, infrastructure, or operating system, *e.g.*, network security groups or virtual private clouds. It leverages those native security features for enforcing granular segmentation across the architecture, without requiring additional technologies to be integrated into the architecture. Native micro-segmentation is suited for cloud-based architectures[16], and for Infrastructure-as-a-Service [86]. If the platform has no inherent segmentation capabilities, third party firewalls are available for creating third party micro-segmentation. Overlay micro-segmentation creates an overlay network on top of an existing network, using Software-Defined Networks (SDN). SDN separates network control from the forwarding process, enabling controllability of the network, which can be used for micro-segmentation. This is the basis for the zero trust architecture 'Software-Defined Perimeters' (SDP) [89], with commercial implementations such as Istio[17], vArmour[18], and Cilium[19]. Finally, hybrid micro-segmentation combines native, third-party, and overlay techniques.

2.4.4 Zero Trust Capability: Encryption

Preservation of *confidentiality* (that data remains secret), of *integrity* (that data remains unchanged), and of *authentication* (knowing who has created the data), is a core principle of zero trust. Focusing on the protection of data directly, rather than securing servers hosting data and networks on which data transit, is a security approach called *'data-centric' security* [215]. According to [97], a 'data-centric' approach for cybersecurity is necessary for zero trust. Thus, in [82], data is the central pillar, as depicted in figure 2.7. *Encryption* is a technology protecting the confidentiality and integrity both of data at rest and of data in transit, as well as being a building block for authentication and isolation techniques.

Symmetric key encryption is an encryption mechanism that uses a secret key, used both for encrypting and for decrypting data. It preserves the confidentiality, and optionally integrity, of data. However, the authentication of data is not possible, as any entity having the secret key can encrypt data.

The Advanced Standard Encryption (AES) [216] is an example of symmetric key encryption scheme, used both for encrypting data at rest, *e.g.*, on cloud servers [217], and for encrypting data in transit, *e.g.*, in the Transport Layer Protocol (TLS) 1.3 [177]. In 2024, 23 years after its standardization, AES was still assumed to preserve the confidentiality of encrypted data, with the best known attack [46] being only four times faster than exhaustive search [47].

[15] G. Young, "Technology insight for microsegmentation," Gartner, Tech. Rep., Mar. 2017. [Online]. Available: https://www.gartner.com/en/documents/3640817.

[16] R. Mansdoerfer, G. Moore, S. Wray, *et al.*, "Implement network segmentation patterns on azure," Microsoft Azure, Tech. Rep., 2022. [Online]. Available: https://learn.microsoft.com/en-us/azure/architecture/framework/security/design-network-segmentation.

[17] Google, IBM, and Lyft, *The Istio service mesh*, 2018. [Online]. Available: https://istio.io/.

[18] vArmour, "Pathway to multi-cloud security architecture," vArmour, Tech. Rep., Nov. 2015. [Online]. Available: https://info.varmour.com/rs/650-OZW-112/images/vArmour_Customers_PathwayToMultiCloudSecurity_Whitepaper_Nov2015.pdf.

[19] Isovalent, *Cilium*, 2018. [Online]. Available: https://cilium.io/.

However, in a cloud deployment, it may be relevant for the cloud provider to be able to perform some operations on tenant data, *e.g.*, categorization of data, enabling data owners to decrypt only the data they need [9]. Therefore, Searchable Symmetric Encryption (SSE) schemes have been developed. With SSE, data owner upload encrypted data to a storage provider, along with encrypted tags describing the data. When the data owner wants to recover data with some tag, it queries the storage provider with a search token, enabling the storage provider to retrieve the encrypted data, without learning information on the data or the query [218]–[220].

To provide authentication, *asymmetric key cryptography* is used. In asymmetric key cryptography, keys come in pairs of: a public key, and a private key. The public key is tied to the identity of an entity in a document called a *certificate*. An entity proves their identity by proving ownership of the private key, using a *digital signature* [221].

A certificate is a document, which binds a public key to the identity of an entity. However, it alone does not prove the trustworthiness of the certificate. Thus, Public Key Infrastructures (PKI) [222] are used to manage and ensure the trustworthiness of certificates. To that end, a PKI contains a Certificate Authority (CA). When an entity wants a certificate issued, it contacts the CA, which verifies the identity of the entity, and ensures that the entity holds the private key associated to the public key on the certificate. If the identity and ownership of the private key are verified, the CA creates a certificate, binding the public key to the identity of the entity, and apposes its signature on the certificate, to guarantee its authenticity. Therefore, the PKI is the root-of-trust for authentication: it is necessary to trust the PKI, and its verification process, to trust the authenticity of certificates.

Asymmetric cryptography is also used for confidentiality and integrity of data, by encrypting data with the public key, and decrypting it with the private key. However, asymmetric encryption is slower than is symmetric encryption, and it is therefore often used for exchanging a symmetric key, before using a symmetric encryption scheme for exchanging data securely [223].

As with symmetric key encryption, there are extensions of asymmetric key encryption that offer extended operations.

Identity-based encryption [224], or attribute-based encryption [225], facilitate the integration of cryptography into authorization mechanisms, as discussed in section 2.4.6.

Functional encryption [226] enables the creation of 'function keys', using the private key, to enable the holder of a function key to learn only part of the data from its encryption.

With homomorphic encryption [227]–[229], it is possible to perform mathematical operations on encrypted data. The result of those mathematical operations on encrypted data is a ciphertext, which, decrypted, corresponds to the same mathematical operations performed on the original data.

Post-quantum cryptography [230], [231] aims at guaranteeing security against attackers disposing of quantum computing capabilities, without needing a quantum computer. In 2023, post-quantum cryptography was in a standardization process [232].

Lightweight cryptography [233]–[236] and PUF-based cryptography [237] are used in IoT devices to reduce the computing power necessary to communicate.

Table 2.5: Zero Trust Encryption Technologies.

Technology	Description
Symmetric Key Encryption	Preservation of confidentiality
Message Authentication Codes	Preservation of integrity
Searchable Symmetric Encryption	Preservation of confidentiality with research capability in encrypted data
Asymmetric Cryptography	Confidentiality and integrity of data
Digital signature	Authentication
Attribute-based encryption	Keys represent attributes
Functional encryption	Only a part of the data can be learnt
Homomorphic encryption	Computations on ciphertexts possible
Lightweight encryption	Low computational power needed

Synthesis

The confidentiality, integrity, and authentication of data and messages is preserved through encryption. Encryption technologies provide specificities for protecting data and messages in different contexts, and with different levels of guarantees, as summarized in table 2.5.

2.4.5 Zero Trust Capability: Isolation

A first step for preventing unauthorized access to, and unauthorized use of, resources, is to isolate resources, and to isolate communications in the network. Security isolation has two main usages: safely executing untrusted programs, referred as *sandboxing* [238], and running trusted programs which may have vulnerabilities, to protect the rest of the system [239]. In cloud computing, isolation is crucial as multi-tenancy implies many users may share the same computing platform, leading to security threats based on service co-location [240].

Isolation mechanisms differ depending on enforcement location and on isolation granularity [239]. The *isolation granularity* describes the scope of the subject being isolated, *e.g.*, a whole operating system (OS), an application, a group of applications, or even part of an application. The *enforcement location* defines where the access decision is performed. There are four main enforcement locations, depicted in figure 2.8: a physical host, a hardware component, a supervisor, or the application itself, detailed in the following.

Physical Host Isolation

Physical isolation protects whole operating systems from the threat of side-channel leaks and of covert channels. This is the strongest isolation mechanism. However, it prevents the sharing of resources [239].

Hardware Component Isolation

Specialized hardware components have been developed for providing isolation of resources on a physical host shared between different users and applications. One example are *Trusted Platform Modules (TPM)*, which are cryptographic

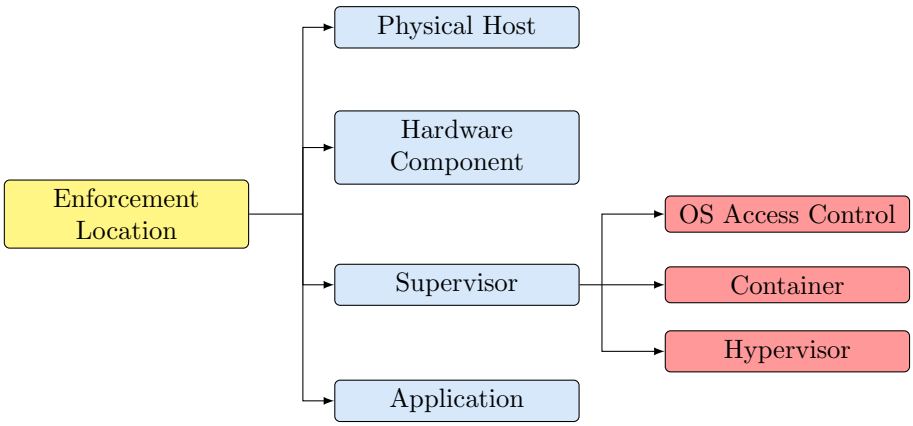


Figure 2.8: Security isolation mechanisms enforcement locations [239].

processors that store cryptographic material, *e.g.*, for creating trusted computing environments [241]. Similarly, *enclaves*, such as Intel SGX [242], isolate regions in memory and in a processor, to perform secure computations. However, numerous side-channel attack challenge the security provided by enclaves [243].

Supervisors

A *supervisor* is a centralized entity that isolates a subject from its environment. There are three groups of techniques to enable supervisor enforcement: OS Access Control, containerization, and hypervisors [239].

Traditional OSs separate two levels of privilege: the *kernel space*, offering processor operations for communicating with the hardware, *e.g.*, for reading and writing files, or for accessing network interfaces, and the *user space*, with least privileged processor operations. User space processes are allocated a virtual address space, to which they can write, and from which they can read. They access it using a *page table*, which converts this virtual address space to physical memory addresses, preventing them from accessing other processes or kernel memory regions [244]. The kernel space contains drivers and fundamental components, whereas the user space contains user applications.

OS Access Control When a user space application needs access to the hardware, it performs a *system call*, which the kernel authorizes or denies depending on the privileges of the user application [245]. This authorization procedure is called Mandatory Access Control (MAC) [246], a procedure inherited from the Orange Book and NSDD-145.

Several methods are used to perform MAC. A *library OS* is a library available to user space applications to perform system calls [247]. It restricts the set of system calls that user space applications can perform. It is also referred to as a *unikernel* [248].

For more flexibility, Linux Security Modules (LSMs) implement access control mechanisms, enabling dynamic and fine-grain access control for authorizing

processes to use system calls [249]. For example, AppArmor[20] enables system administrators to define application profiles, for defining which kernel resources applications can access.

Containers *Containers* isolate applications, or groups of applications, by providing a restricted execution environment on a physical host [250]. Container isolation is performed through *Linux namespaces* [251], with each container having its own namespaces. A Linux namespace partitions kernel resources, such that each container sees their own set of resources. Example Linux namespaces are the Unix Time-Sharing System (UTS) namespace, providing a hostname to the container, or the Process Identification (PID) namespace, separating the processes of the container from other processes running on the host or in other containers. Access to kernel functionalities for containers is restricted [249], *e.g.*, by Seccomp[21], which limits the list of system calls processes in a container can perform. Containers can be configured, on the host, to access more kernel resources (which are grouped into *Linux capabilities*, that can be added or removed from containers). However, containerization is vulnerable to co-located containers attacks [253], [254], and to attacks on the host [255].

A particular use for containerization is *microservices*, which enable intra-application isolation. Microservices are software units responsible for specific functions in distributed applications [256]. They are either deployed on a single machine, communicating via Inter-Process Communication [257], or run on different machines. One way to deploy microservices is to deploy them in containers, managed by orchestration tools and cluster managers, *e.g.*, Google Borg [258], Docker Swarm Manager [259], and Kubernetes [260]. Such a microservices oriented architecture is called a Service Oriented Architecture [261]. Several zero trust solutions are based on microservices and containerization, such as Cilium[22], Istio[23], or CyberGuarder [262].

Hypervisors Virtualization offers a stronger isolation mechanism compared to containerization [250]. A *virtual machine* (VM) is an isolated machine, composed of an OS and of applications, that run on another (physical) machine, called the *host*. OSs of VMs are called *guest OSs*. VMs are managed by a *hypervisor*, which monitors VMs, and acts as a link between VMs and the host. There are two types of hypervisor: *bare-metal* hypervisors, which interact directly with the hardware, and *hosted* hypervisors, which run on the host OS [245].

Synthesis

Isolation mechanisms prevent unauthorized access to resources. Isolation is either enforced physically, or through dedicated components, *e.g.*, a TPM, or a

[20] *Apparmor: Linux kernel security module*, 2009. [Online]. Available: https://apparmor.net/.

[21] J. Edge, *A seccomp overview*, Linux Plumbers Conference, 2015. [Online]. Available: https://lwn.net/Articles/656307/.

[22] Isovalent, *Cilium*, 2018. [Online]. Available: https://cilium.io/.

[23] Google, IBM, and Lyft, *The Istio service mesh*, 2018. [Online]. Available: https://istio.io/.

supervisor, preventing transfer of information or actions to and from isolated spaces.

2.4.6 Zero Trust Capabilities: Access Control and Authorization

Authorization is the process of deciding if a connection is compliant with an access control policy or not. NIST [83] defines the authorization function as a trust algorithm, taking as input the access request, the subject database and history, the asset database, the resource policy, threat intelligence, and logs, to decide if the connection is authorized or not.

There are several types of authorization systems, depending on the input used:

- *User-based* authorization is a static authorization system, which authorizes access to a resource depending on the identity of the entity (human or non-human) requesting access and its device. This authentication category is divided in several subcategories: *identity-based* authorization [182], [263] takes into account the identity of the entity, *role-based* authorization assigns roles to entities.
- *Context-aware* authorization takes into account environmental attributes [264], or the behavior of the subject requesting authorization [265]–[267].
- *Attribute-based* authorization assigns attributes to entities, to objects, to operations, and optionally to the environment, and defines a policy for evaluating those attributes, to authorize an entity to perform an operation on an object [268]–[271].
- *Trust/Risk-based* policies compute a trust score for each authorization request [272]–[277].
- *Usage control* has the action at the heart of the authorization, instead of the entity [278]. Formally specified by [279], usage control follows a need-to-know approach, and authorize access depending on the business needs. Access policies are pre-defined and static [280], consider the user intent [281], or are application-aware [282].

Synthesis

Several authorization systems leverage authentication and access control policies to authorize connections. As zero trust authorization requires to be dynamic, per-session, and least-privileged, authorization technologies need to take into account the context, and to have sufficiently fine-grain expressions of authorization.

2.4.7 Zero Trust Capabilities: Monitoring and Mitigation

Visibility is one of the pillars of zero trust, and is needed for assessing the security posture of assets, of devices, and of users. Security Incident and Event Management (SIEM) systems help organizations to detect, analyse, and react to threats, by providing organizations with a global visibility on activity in

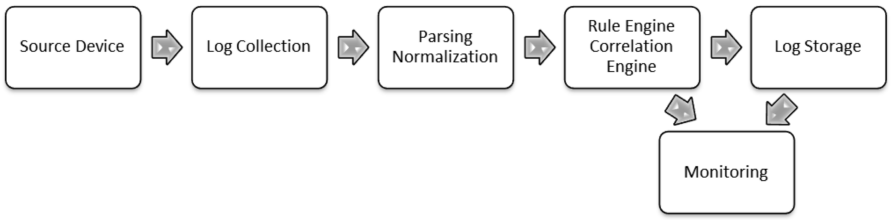


Figure 2.9: SIEM basic components (from [283]).

the organization. SIEM systems operate in three steps. First, sensors collect security events and normalize those events [284]. *Security events* are occurrences or activities in the organization that may indicate the presence of a threat, *e.g.*, system and network logs, or antivirus or IDS alerts [285]. To get a better understanding of the meaning of those events, situational awareness is necessary. This is provided by external threat knowledge [286]. Thus, the second step of SIEM systems is the integration of threat intelligence feeds. Finally, SIEM systems analyse and correlate events in the light of threat intelligence and organization knowledge, and raise alerts if abnormal activity is found. SIEM systems also offer visualization techniques for simplifying the processing of alerts and events by human operators [287]. An example of basic components within a basic SIEM solution is depicted in figure 2.9.

Event Collection

Continuous Diagnostics and Mitigation (CDM) systems gather information on managed devices[24]. A preliminary analysis may be performed on the device by the CDM system, before sending results for further processing to the SIEM. Enclaves, or TPM, deployed on devices enable SIEM systems to trust the local processing operation [288].

Anti-viruses scan files and applications to detect malicious software [289], by matching files against known signatures, or by running applications in sandboxes to detect suspicious behavior [183], [290].

The network is monitored with an IDS. IDSs detect intrusions, defined as an attempt to compromise the confidentiality, integrity, or availability of files and services [291]. There are three main categories of IDSs: signature-based, which detect known malware signatures in packets, anomaly-based, which detect abnormal network behavior, and stateful protocol analysis, which stores the state of flows for analysis [292]. Content Disarm and Reconstruction is a detection technique that analyses payload for finding vulnerabilities [293].

To prevent having a single point of failure, and to prevent the modification or erasure of logs, blockchain-based IDSs have been proposed [133], [294], for a decentralization of monitoring, while maintaining a coherent view of the network. Moreover, the immutability property of a blockchain, stating that transactions recorded on the blockchain cannot be modified, ensures logs are not modified or erased. However, blockchain-based solutions suffer from limited handling capacity in case of heavy traffic, from high energy and cost re-

[24]Cybersecurity & Infrastructure Security Agency (CISA), *Continuous diagnostics and mitigation (CDM) program*, 2012. [Online]. Available: https://cisa.gov/cdm.

quirements, from higher latency, and from higher vulnerability to denial of service [295].

In software-defined networks, network control is separated from the forwarding process. Thus, network analysis and monitoring is performed by the SDN controller, which has a global view of the network [296]–[299].

Threat Intelligence

Threat intelligence provides knowledge to organizations of current threats, for contextualizing events to detect attack patterns [283], [300]. Threat intelligence improves anti-viruses, IDSs, and vulnerability scanners [301].

There are two types of threat intelligence [302]: first, operational intelligence provides knowledge on ongoing attacks and campaigns, and helps incident response teams to detect and deter attacks[25]. Second, strategic threat intelligence provides an overview of the organization threat landscape.

Threat intelligence results are integrated to SIEM systems through the use of normalized languages, such as STIX [303], or through integrated threat intelligence sharing platforms [304], [305].

Analysis

SIEM systems analyse and correlate the collected data, regarding threat intelligence, to produce relevant alerts and visualizations.

The amount of collected data is a challenge for SIEM systems [286]. Machine learning is one technique for analyzing large amount of collected alerts, as proposed by [306], [307] for IoT networks, by [48] for cloud networks, by [308]–[310] for SDN, and by [311], [102] for general networks. For example, [312] builds a fingerprint of IoT devices, and uses graph neural networks to categorize the traffic behavior given the fingerprints. Big data analytics and statistics are also used for attack detection [313], [314], as is deep learning [315]–[317], which increases the accuracy and precision of threat detection.

Analysis results create alerts, and several SIEM systems automatically repair vulnerabilities or attacked systems [318].

Synthesis

Monitoring and mitigation require technologies for capturing relevant events in the architecture, and technologies for interpreting those events and for taking action accordingly. Such technologies enable capabilities for the monitoring zero trust pillar, thus increasing the level of awareness of the organization.

2.4.8 Zero Trust Capabilities: Automation and Orchestration

Automation and Orchestration is a core pillar of zero trust, and is responsible for automating consistent security responses across the zero trust architecture. This, through the use of Security Orchestration, Automation and Response (SOAR) [82].

[25] Osterman Research, Inc., "The value of threat intelligence," Spamhaus Technology, whitepaper, 2019. [Online]. Available: https://www.spamhaus.com/custom-content/uploads/2020/04/2019-The-Value-of-Threat-Intelligence-White-Paper-LR.pdf.

Security automation is the use of software or tools for completing security tasks, by reducing the involvement of human operators[26]. Security orchestration is the integration and coordination of heterogenous security tools from different vendors and experts, as a unified security solution, producing actions in response to security incidents. It has three key functionalities: acting as a middleware for security tools, orchestrating security activities, and enabling automated response [319].

An approach for automatic security is autonomic security: self-managing characteristics of distributed autonomous resources, adapting to changes in real-time, while hiding intrinsic complexity [320]. Google has proposed a set of philosophies and tools called Autonomic Security Operations[27], which apply to zero trust networks for supporting automatic authentication and dynamic access control [157]. These tools also apply to distributed cloud zero trust environments [322].

Artificial intelligence (AI) is one technology for automating tasks. AI and SDN help by creating access rules dynamically [323], and machine learning enables dynamic authorization [201], [324], [325]. Machine learning is a basis for implementing SOAR capabilities [326]. More generally, [132] presents how AI and automation techniques improve zero trust architectures.

Containerization and containers orchestration are another tool towards automation. For example, Kubernetes clusters provide control groups to isolate resources, and provide authentication and dynamic authorization capabilities [103]. The Istio service mesh[28] is another example of a deployed zero trust architecture using containerization for traffic management, telemetry, and for network security.

Synthesis

Technologies for automation and orchestration coordinate information from other technologies in the architecture, and apply rules for performing security operations, reducing the need for human operators, *e.g.*, by automating alert collection, alert prioritization, and repetitive processes.

2.4.9 Section Overview

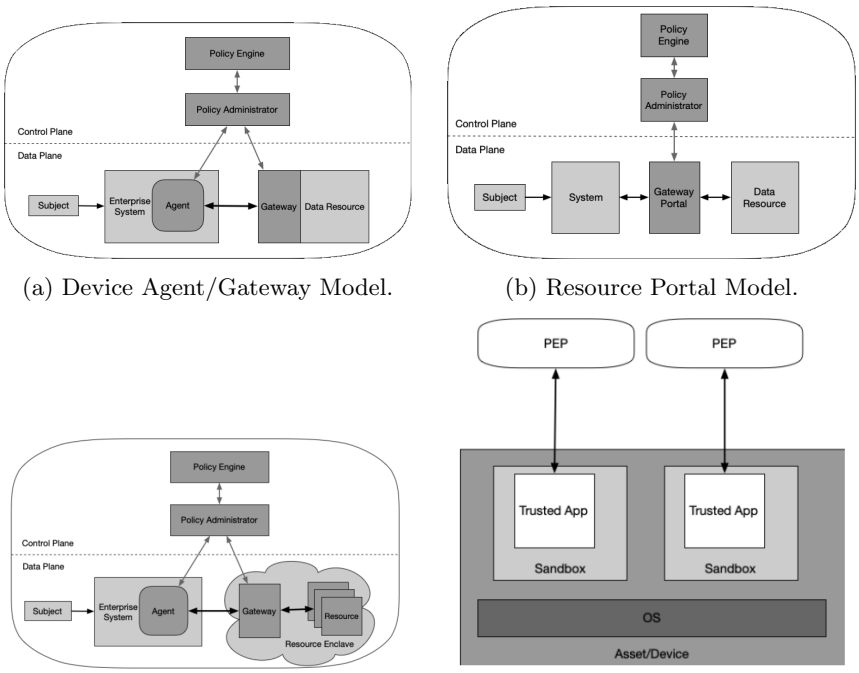
In order to attain higher levels of zero trust maturity, an architecture must provide several capabilities, which are obtained through the integration of technologies – such as an IdM, a CDM system, a PKI, a SIEM system, etc. – into the architecture.

This section has presented an overview of technologies for implementing zero trust capabilities. There is no single way to implement capabilities: the choice of technologies depend on the level of security, and of zero trust maturity,

[26] J. Trull and V. Agarwal, *Top 5 best practices to automate security operations*, blogpost, 2017. [Online]. Available: https://www.microsoft.com/en-us/security/blog/2017/08/03/top-5-best-practices-to-automate-security-operations/.

[27] I. Ghanizada and A. Chuvakin, "Modernizing SOC... introducing autonomic security operations," Google, Tech. Rep., 2021. [Online]. Available: https://cloud.google.com/blog/products/identity-security/modernizing-soc-introducing-autonomic-security-operations?hl=en.

[28] Google, IBM, and Lyft, *The Istio service mesh*, 2018. [Online]. Available: https://istio.io/.



(a) Device Agent/Gateway Model.

(b) Resource Portal Model.

(c) Enclave Gateway Model.

(d) Application Sandboxes.

Figure 2.10: NIST Deployment Models (from [83]).

that the organization desires to achieve, as well as existing constraints in the architecture.

2.5 Maturity Positioning of Zero Trust Architectures

The previous sections have shown that zero trust is a process of integrating several technologies into an architecture, in order to attain higher levels of zero trust maturity. This section evaluates and positions the zero trust maturity of a representative selection of architectures that label themselves as zero trust, with respect to the zero trust core principles, presented in section 2.2.

2.5.1 Categorization of Zero Trust Architectures

The abstract NIST architecture [83] defines the basis of zero trust, with segmented resources protected by a PEP, with dynamic authorization, with monitoring, and with the identity of entities as a key component for policy creation. It categorizes zero trust architectures in four deployment variations, presented in figure 2.10.

In the device agent/gateway model, depicted in figure 2.10a, the PEP is split into two components: a *device agent* installed on devices, and a *gateway* protecting resources. The device agent communicates with the Policy Administrator (PA), requesting access to a resource. If access is authorized, the PA configures the gateway, so the agent can connect to the resource.

In the resource portal model, depicted in figure 2.10b, the PEP is a single component acting as a gateway for subject requests. This model does not require the installation of an agent on every device accessing resources.

The enclave-based model, depicted in figure 2.10c, is similar to the device agent/gateway model, except that the gateway resides at the boundary of an enclave protecting the resource. This model is useful for organizations that cannot install individual gateways in front of each of its resources.

Application sandboxing, depicted in figure 2.10d, is another variation of the device agent/gateway model, except that only trusted applications, running in sandboxes, are allowed to query resources.

2.5.2 From Black Core to Software-Defined Perimeters

Following the concept of deperimeterization introduced by the Jericho Forum in 2004[29], the DoD published in 2007 an architecture dubbed Black Core [41]. This architecture aims at providing dynamic, responsive, and flexible communications and access to resources, while providing assurances that the information is correct, available, and protected. Technical solutions to create such infrastructures include intelligent gateways, which protect resources with a fine granularity, have a service-oriented approach, and offer AI capabilities for dynamically protecting information.

One property of Black Core is that the infrastructure is 'black', *i.e.*, the infrastructure is resilient against reconnaissance and cannot be mapped through scanning. This property was extended by the CSA in 2014, with an architecture called Software-Defined Perimeters (SDP) [94], which replaces the physical gateways of Black Core with software defined gateways. SDP is an architecture based on the need-to-know model, with components in the infrastructure cloaked against unauthorized entities.

Software-Defined Perimeter

SDP is based on Software-Defined Networking, decoupling the control plane from the data plane: in SDN, operations of deciding how data is to be forwarded towards their destination, *e.g.*, routing, network discovery, or traffic management, are decoupled from actual forwarding operations. The control plane is responsible for the former operations, whereas the data plane is responsible for the forwarding. This separation provides flexibility, with programmability for network application development, modularity, and scalability [104]. A central controller is in charge of managing control plane operations, having a global overview of the network.

SDP has a central SDP controller, which dynamically configures gateways to allow or deny traffic, following the zero trust access policy. Gateways are the components providing the hiding property of the infrastructure: by default, all traffic reaching gateways is dropped. In order to forward traffic through a gateway, a client first needs to authenticate to the gateway, with a process described further.

With the formalization of zero trust by NIST in 2020 [83], the SDP specification was refined by the CSA in 2022 [89]. The core of SDP was unchanged,

[29] P. Simmonds, *De-perimeterisation*, 2004. [Online]. Available: https://www.blackhat.com/presentations/bh-usa-04/bh-us-04-simmonds.pdf.

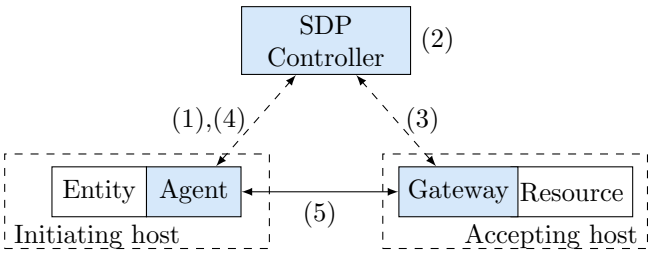


Figure 2.11: Architecture and access workflow of SDP [89]. Dashed arrows are control channels, and solid arrows are data channels.

but more specific details on the SDP protocol and its usage were provided, as well as abstract definitions for zero trust architectures.

The components of an SDP architecture are depicted in figure 2.11. Every device contains a device agent, monitoring the device and establishing connections with the SDP controller and with SDP resources. Devices and entities trying to access an SDP resource form *Initiating Hosts* (IH).

Resources, *e.g.*, a web or SSH server, as well as the SDP controller, are protected by SDP gateways, which filter every communication, and which communicate with the SDP Controller. By default, gateways drop every connection (except those with the controller). A resource and the gateway protecting it form an *Accepting Host* (AH).

When a resource or a device is on-boarded, its gateway or device agent creates a secure control channel to the SDP Controller. Only authorized entities can communicate with the SDP Controller and with resources. This filtering is performed by gateways, using *Single Packet Authorization* (SPA)[30]: before communicating with a resource, or with the SDP controller, behind a gateway, the host sends an authorization packet, called an SPA packet, containing cryptographic authentication information. Validating an incoming SPA packet is computationally lightweight, and does not require state to be stored on the gateway, thus improving the resiliency of the infrastructure against denial-of-service attacks. Then, once the SPA packet has been accepted, the host establishes a secure channel with the controller using mutual TLS (mTLS) [328].

When an IH is on-boarded, it first authenticates with the SDP controller, and the SDP controller determines which resources the IH can access. Those are steps 1 and 2 in figure 2.11. Then, the controller configures the gateway of AHs that the IH is allowed to access, and also communicates to the IH the list of resources it can access (steps 3 and 4). Finally, the IH can create a secure communication channel with the AH, based on SPA and mTLS (step 5).

The SDP specification [89] does not specify a zero trust access policy used by SDP controllers, and leaves the choice to each organization implementing SDP. The SDP controller may use internal tables for authentication, or connect to an external IdM, for example to enforce MFA. Similarly, authorization may be based on internal tables, or on dynamic access control systems, and may take into account contextual information.

[30] M. Rash, "Single packet authorization," *Linux Journal*, 2007. [Online]. Available: https://www.linuxjournal.com/article/9565.

Thus, SDP in itself is an abstract zero trust architecture, not an implementation: authentication, device validation, monitoring, and automation are not detailed in the specification, their implementation being left to each organization. To implement SDP, the SDP controller would require an authentication system, a device validation system, and an access control system, all following zero trust principles. Moreover, the infrastructure would require monitoring and analytics, data at rest encryption, and automation and orchestration.

SDP is nonetheless a starting point for implementing zero trust architectures, as it provides technologies for enabling dynamic and per-session authorization, with segmentation of access, and end-to-end encryption.

SDP-Based Implementations

As SDP is an architecture, not an implementation, design and technology choices left to organizations consequently, there are numerous zero trust implementations based on SDP. SDP is a basis for zero trust Infrastructure as a Service [105] and smart home networks [106]. SDP is used in the 5G smart healthcare [107], and for IoT systems [108]. SDP is the preferred [71] zero trust architecture for networks composed of numerous IoT devices with unstable communication channels.

Waverley Labs [109], whose authors are amongst the SDP specification authors, offers an open source implementation of an SDP architecture. It follows the SDP specification, and implements identity management using a trusted PKI, and uses a static role-based access control, defined in an SQL database. This implementation is not a complete zero trust solution: there is no device validation, policies are static, and there is no mention of monitoring.

In [110], an SDP solution is presented, in which policies are defined using the theoretical models of Bell-LaPadula and BIBA. A dynamic threshold-based access control policy based on the location of the request is proposed in [111]. SDP is extended with a Trusted Execution Environment (TEE) in [112] for IoT architectures.

As described in the previous section, SDP security is based on SPA followed by a secure mTLS connection. However, this leads to security issues [113], [141]: the correlation between the authentication packet and the subsequent secure connection is based on the IP address of the initiating host, which may lead to the hijacking of the secure session [142].

Thus, [113] proposes an extension of SDP, using extended Berkeley packet filters to include authorization tokens within mTLS packets, thus linking the authentication packet to the subsequent flow.

To increase the difficulty of a successful attack, [96] extends SDP by cloaking network properties (such as domain names or IP addresses), in order to also make reconnaissance more difficult, even for insiders.

Another challenge of SDP is that the control plane is assumed trusted, especially the controller. This may be an unrealistic assumption, as attackers – either an external threat or an insider threat, *e.g.*, a malicious employee – might be able to compromise control plane components [114]. Therefore, [115] proposes to use a verified and secure control plane based on DNS, instead of trusting an SDN control plane. Similarly, [114] proposes an architecture in which the controller is not trusted, and instead uses trust distribution techniques, such as survivable databases and survivable SSO.

As the controller is a central entity, it becomes a single point of failure: if it is compromised, security in the whole network is compromised. The SDP specification [89] proposes a protocol for duplicating the controller: a *primary* controller is onboarded into the architecture, and it then onboards subsequent controllers, for load balancing and for redundancy. The specification does not describe how the state between controllers is shared, accessed, and synchronized, as it is implementation specific. Duplicating controllers may limit incremental deployment of controllers, *e.g.*, for updates or change of hardware.

Several solutions exist for ensuring the consistency of multiple controllers in SDN, as summarized by [329]: one of them is the use of a publish-subscribe system between controllers, to exchange their states. A second solution is the use of application-specific consistency algorithms, such as clustering techniques. Similarly, a heterogeneous redundancy mechanism, based on endogenous security, is used to replicate an SDP controller [116]. Alternatively, the use of blockchain for ensuring consistency has been proposed, as one purpose of blockchain is to make every node reach a consensus [133], [330]. However, the performance of blockchain-based solutions is prohibitive for deployments, compared to centralized solutions, because of computation overhead due to the replication mechanism across all nodes, and of the low throughput of blockchain technologies [331].

2.5.3 Resource Portal Deployment Model

This section presents zero trust architectures following the resource portal deployment model depicted in figure 2.10b.

2010: Forrester Zero Trust Network [98]

In 2010, Forrester proposed the first definition for zero trust [77], as well as an implementation following this definition of zero trust, called the Forrester Zero Trust Network [98], depicted in figure 2.12. It is based on a special device, called a 'network segmentation gateway'. This device is at the nucleus of the network, and segments it securely, by creating parallel network segments, called MicroCore And Perimeters (MCAP), following a global access policy. All traffic is inspected and logged using a Data Acquisition Network (DAN). Policies are configured using a management (MGMT) server.

While the implementation follows the 2010 zero trust definition, zero trust has evolved, including more principles and pillars. Thus, the Forrester Zero Trust Network would not be considered a fully mature zero trust architecture, as policies are not necessarily dynamic, and authentication of devices and users is not mandatory. Moreover, the automation and orchestration of segments and of access policies is not considered.

BeyondCorp

Google BeyondCorp [78]–[80] is an architecture, depicted in figure 2.13, based on a special device called the '*access proxy*', which intercepts every connection before forwarding them (if authorized) to their destination backend. This centralized decision point enables easy logging of connection attempts, and

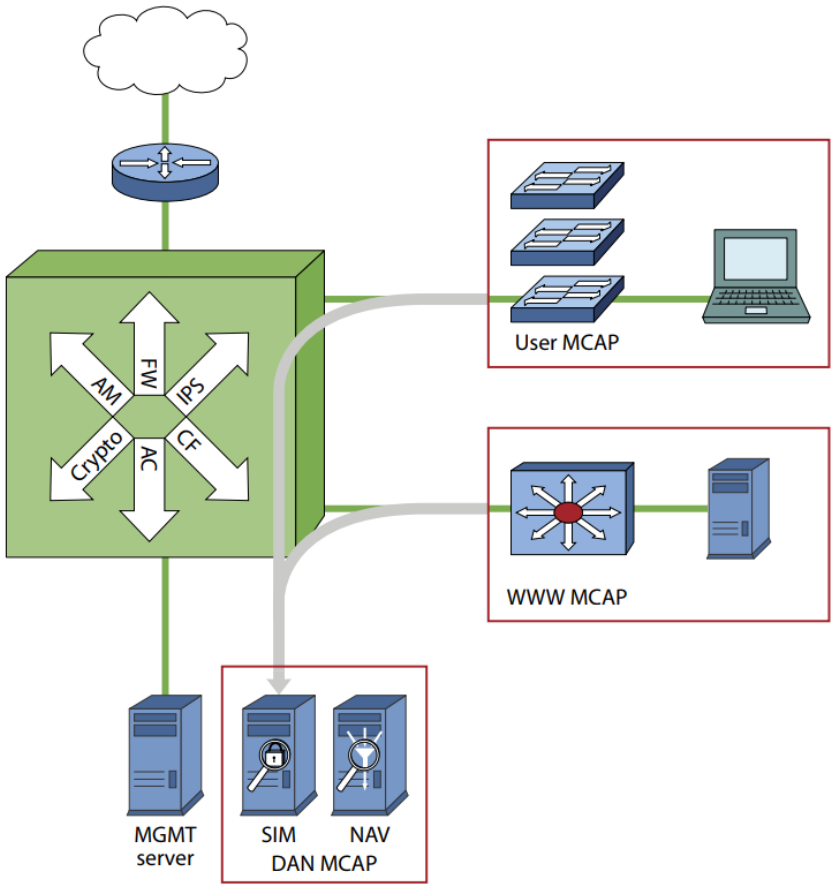


Figure 2.12: Forrester Zero Trust Network (from [98]).

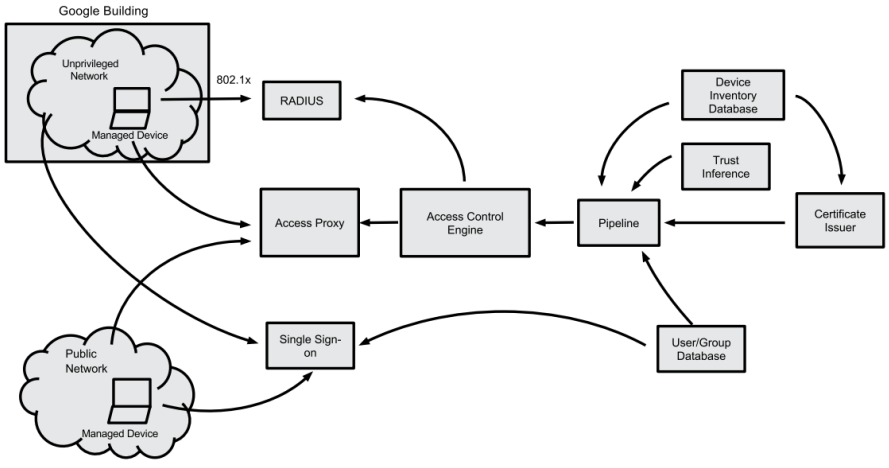


Figure 2.13: BeyondCorp components and access flow (from [78]).

enables a dynamic policy. The policy is based on an inventory of users and devices, on single sign-on, on an access control language based on Access Control Lists (ACLs), and on a trust inferrer evaluating if a connection can be trusted. Google uses a derivative thereof, called BeyondProd [87], for securing cloud services.

An implementation of BeyondCorp using Kubernetes has been proposed [103]. However, this implementation only uses a static access control policy, and no dynamic changes in the policy or monitoring are specified.

An extension of BeyondCorp focusing on data sensitivity is proposed in [332]. The proposed solution is not a fully mature zero trust architecture, as it lacks automation and monitoring.

The BeyondCorp access proxy is also used to secure industrial IoT devices [117], [49], [118].

BeyondCorp is limited to HTTP traffic, which prevents legacy services from being incorporated in the zero trust architecture [140], thus [119] proposes a solution that encapsulates general IP traffic and forwards it to the access proxy, using a Virtual Private Network (VPN).

2.5.4 Zero Trust as a Combination of Zero Trust Solutions

The numerous capabilities that are needed for an architecture to reach an 'optimal' zero trust maturity, are summarized in table 2.3 at the beginning of section 2.4. Each of the different architectures presented in the previous sections only include a subset of those capabilities.

Thus, in 2018, Forrester presented the concept of the Zero Trust eXtended (ZTX) ecosystem [84], as a framework for zero trust vendor solutions. According to this framework, a zero trust solution is considered part of the ecosystem if it offers an advanced or optimal maturity in at least three zero trust pillars, *and* if it provides an Application Programming Interface (API) to be interoperable with other vendors solutions. This ensures compatibility between vendors, and enables building a complete architecture, associating several vendors, to reach a full zero trust maturity.

More than 30 zero trust vendors and platforms were part of the ZTX ecosystem in 2018. The following provides a description for a selection of those zero trust products.

Gartner ZTNA [120]

The Gartner Zero Trust Network Architecture (ZTNA) [120] is based on the Continuous Adaptive Risk and Trust Assessment (CARTA) framework[31], an adaptive and risk-based architecture. Users and resources are first authenticated, and authorization is then performed using the context of the connection, with a continuous evaluation of the security postures of the user, the device, the application, and the network. It follows the NIST resource portal deployment model from figure 2.10b.

[31] K. Panetta, *The gartner it security approach for the digital age*, Gartner, Jun. 2017. [Online]. Available: https://www.gartner.com/smarterwithgartner/the-gartner-it-security-approach-for-the-digital-age.

Adding IoT-specific capabilities to ZTNA, *e.g.*, device discovery and compliance management, provides zero trust security to the smart manufacturing industry [135].

ZTNA is sold by vendors, *e.g.*, NetMotion, Cyolo, Cloudflare [88], or Zscaler, as a zero trust component performing authentication, monitoring, and access control. Device validation, monitoring, and data encryption need to be added to those vendor architectures.

Gartner ZTNA is also the base architecture for the zero trust solution of Perimeter 81[32]. This architecture combines cloud-based firewalls and network traffic control for defining policies, third party SIEM services, such as Amazon S3 or Splunk, third party identity providers (*e.g.*, Google, Azure), and VPN technologies for encryption.

Similarly, Akamai[33] uses a ZTNA-based solution, adding to the microsegmentation capabilities of ZTNA multifactor authentication and web-based protection.

In sum, ZTNA offers capabilities for reaching 'advanced' or 'optimal' zero trust maturity for the identity, network, applications, and visibility pillars (in figure 2.5). However, it does not offer capabilities for device monitoring, application-level protection, data-centric security, and advanced automation.

VMWare NSX [86]

The core of the VMWare NSX [86] zero trust architecture is workloads. Each workload is centrally evaluated by a controller, which creates, if the workload is authorized, a segment between the source and destination. Authorization depends on the workload itself, and its context. The target audience for VMWare NSX are data centres, on which a controller can have a global vision.

VMWare NSX provides micro-segmentation, transport encryption, and data-centric security, therefore providing advanced to optimal zero trust maturity for the network, application and workloads, and data zero trust pillars.

It is possible to combine it with other products to reach higher maturity for the user, device, analytics, and orchestration pillars, for creating mature zero trust architectures[34].

Cisco Secure Platform

Cisco offers a zero trust platform, and a processus for migration to zero trust following the CISA maturity model[35].

In their zero trust architecture reference [121], Cisco combines several products to build a zero trust architecture, *e.g.*, Umbrella for providing network monitoring and analytics, Duo for providing multifactor authentication, or

[32] P. 81, "Perimeter 81 zero trustnetwork access (ZTNA)," Tech. Rep., 2021. [Online]. Available: https://www.perimeter81.com/lp/zero-trust-network-access-datasheet-pdf.

[33] Akamai, *Zero trust security model*. [Online]. Available: https://www.akamai.com/our-thinking/zero-trust/zero-trust-security-model.

[34] VMware, *Introduction to vmware zero trust*, Apr. 2020. [Online]. Available: https://techzone.vmware.com/resource/introduction-vmware-zero-trust.

[35] Cisco, "Cisco's guide tozero trust maturity," Tech. Rep., 2022. [Online]. Available: https://www.cisco.com/c/dam/en/us/products/collateral/security/zero-trust-field-guide.pdf.

Thalos for providing threat intelligence. With this combination of products, an advanced to optimal zero trust maturity is reached in the identity, device, network, application, visibility and analytics, and automation pillars. However, configuration of dynamic policies is not clearly defined, as well as data at rest protection.

Illumio

Illumio[36] proposes a zero trust architecture whose core is dynamic network segmentation. Access is controlled by a Policy Compute Engine, which collects information from Virtual Enforcement Nodes, providing real-time information on the network.

Illumio mainly provides segmentation capabilities, reaching an advanced to optimal zero trust maturity for the network pillar, for automation, and for data-centric security. However, it needs to be combined to other solutions for reaching high maturity in the identity, devices, and visibility and analytics pillars.

2.6 Discussion

The zero trust paradigm provides benefits over the perimetric paradigm. However, it does not answer every security challenge. Migrating from an existing architecture to a zero trust architecture is difficult, and zero trust technologies themselves introduce new vulnerabilities.

2.6.1 Benefits of Zero Trust

With the application of the GDPR in 2016, demand for data sovereignty has globally spread, with more than 65% of state agencies searching for solutions for data-centric security (DCS), and with almost half of those agencies considering zero trust[37]. Thus, data sovereignty is a critical business case for zero trust[38].

Similarly, the increase of work-from-home following the COVID-19 pandemic fueled the adoption of zero trust: a survey [143] shows that in 2023, 30% of interviewed corporates were in the process of incorporating zero trust, and 60% were planning to.

Forrester presents eight business and security benefits for zero trust[39], which can be summarized as a higher visibility on the network for securing assets, resulting in higher security at a lower cost, and an easier implementation of security measures, of compliance initiatives, and of business transformation.

[36] Illumio, "Illumio architecture," Tech. Rep., 2020. [Online]. Available: https://resources.illumio.com/resources/illumio-architecture.

[37] P. Patterson, "Demand for data sovereignty is moving to local government," Cisco Services, Tech. Rep., 2022. [Online]. Available: https://blogs.cisco.com/services/demand-for-data-sovereignty-is-moving-to-local-government.

[38] D. Schaupner, "Enabling data sovereignty with zero trust," *Digital Security Magazine*, 2022. [Online]. Available: https://atos.net/en/lp/digital-sovereignty-cybersecurity-magazine/enabling-data-sovereignty-with-zero-trust.

[39] C. Cunningham, D. Holmes, and J. Pollard, "The eight business and security benefits of zero trust," *Forrester Reseach November*, 2019. [Online]. Available: https://www.kennisportal.com/wp-content/uploads/2022/06/Akamai-the-eight-business-and-security-benefits-of-zero-trust-report.pdf.

Each principle of zero trust increases the security of the architecture. Strong authentication protects resources against attacks and prevents breaches [129]. Least-privilege and dynamic authorization enable an easier and more precise definition of access policies, resulting in fewer misconfigurations [144]. Fine segmentation of access prevents lateral movement [145]. Mandatory encryption of data in transit and at rest follows the principles of data-centric security, preventing data exfiltration and tampering [70]. Monitoring provides higher visibility on the network, enabling the detection and prevention of attacks [83].

According to [146], zero trust reduces the cost of data breaches, thus directly balancing the cost of implementing zero trust, *e.g.*, by lowering insurance premiums.

In academic literature, several studies have formally evaluated the benefits of zero trust. Formal methods have been designed to formalize the security offered by zero trust architectures, such as meta attack languages [147], or trust loss effects analysis [148]. However, they have not been used to evaluate real-world zero trust implementations.

SDP was found to be resilient against denial-of-service attacks and port scanning attacks [75], to mitigate risks efficiently [141], to enhance access control with fine granularity, and to minimize the attack surface comparatively to perimetric access control [149].

However, if zero trust provides benefits for securing organizations, it protects them only against specific attacker models [150].

2.6.2 Challenges of Zero Trust Migration

Zero trust migration raises several challenges for organizations. A first challenge is the lack of clear capabilities that an architecture requires to attain a high zero trust maturity level [71], *e.g.*, for defining the granularity of segmentation, or for defining what needs to be validated in devices. This is mitigated by evaluation frameworks, which help define capabilities more precisely, *e.g.*, the framework from [151] evaluating the degree of micro-segmentation. However, such precise frameworks do not evaluate the maturity of all zero trust pillars, and several require to be used for evaluating the zero trust maturity of an architecture, and defining subsequent goals.

Alternatively, [152] proposes a machine learning model to evaluate the maturity of an architecture with regard to zero trust pillars. However, it only has a 95% accuracy, which prevents it from being used alone for evaluation, and it does not provide insight for defining precise goals for increasing the zero trust maturity.

A second challenge for zero trust migration is the lack of standards, coupled with vendor lock-in which may prevent an easy integration of different commercial solutions. Moreover, selecting the most suitable zero trust products, for an organization which may evolve, remains a challenge [71].

A third challenge with migration is the integration of legacy systems into the zero trust architecture [72]. Moreover, migration to zero trust often disrupts users and slows down the production flow, which is contrary to the business goals [153]. Thus, migration to zero trust necessitates a strong willingness to migrate – from all users, as well as from management [78]. Finally, technical issues may arise while implementing and deploying zero trust solutions, as

well as a lack of coordination between management teams, for defining access policies, and for the implementation [122].

2.6.3 Vulnerabilities Introduced by Zero Trust

While zero trust offers protection against attacks, it creates new vulnerabilities [157]. Those impact productivity and resilience.

NIST [83] enumerates several of those as follows: as access is granted following a global policy, subversion of the decision process may occur through a successful attack on the PDP, or through an insider threat. Moreover, a denial of service on critical zero trust elements, such as the PDP, creates a denial of service on every resource in the organization. As the decision process is centralized, logs also need to be centralized for taking access decisions. As logs contain critical information on the whole architecture, this centralized log storage creates a single point of vulnerability. Finally, reliance on external data or solutions harms zero trust, and weakens security. This makes the design of federated zero trust architectures difficult, as trust between federation members is difficult to acquire [154].

Moreover, chosen zero trust solution have an attack surface that can be a source of vulnerabilities, *e.g.*, several vulnerabilities have been found in SDP solutions [113], [141], or in blockchain-based zero trust solutions [333].

2.6.4 Real Zero Trust Architectures

The benefits and challenges of zero trust raised in this section consider fully matured zero trust architectures. However, as discussed, zero trust is also a migration process. Thus, architectures undertake a migration process, from traditional to zero trust maturity, over time.

Therefore, being able to position architectures relatively to their zero trust maturity level, and their following of zero trust core principles, is necessary to gain an understanding of the benefits, and risks, that non fully matured architectures bring to organizations. Transition periods bring fewer benefits and higher risks than fully matured architectures, which need to be taken into account while performing a zero trust migration.

2.7 Summary

So, is trust misplaced? As shown in section 1.1, trust in perimeter security is misplaced: because of insider threats and vulnerabilities, breaches occur; as perimeter security is mostly static, and enables lateral movements, those breaches have organization wide consequences.

Thus, the concept of zero trust has emerged, whose fundamental paradigm is that every access request needs to be explicitly verified, without relying on implicit trust. This chapter has shown how this fundamental paradigm is decomposed into several core principles: every entity needs to be authenticated, and authorization is granted dynamically, by taking into account the environment, for a limited time, with an as fine as possible granularity, and following a least-privilege policy. This implies that the infrastructure, entities, and resources are constantly monitored, and data and communications are always encrypted.

As discussed in section 2.3, migration from a perimeter security architecture to zero trust is performed incrementally, by adding capabilities to the architecture to improve the zero trust maturity, following zero trust pillars: identity, device, network, applications, data, visibility, automation, policy, and governance. Such a transition is challenging, both from the technological perspective and from governance, with business goals potentially hindered by zero trust deployment. Moreover, the need for multiple interoperable security technologies creates a high dependency in vendors of security solutions.

The formalization presented in this chapter enables positioning architectures relatively to zero trust principles and maturity, for a better understanding of the benefits and additional risks they incur on the organization. Reaching high levels of zero trust maturity requires the architecture to dispose of several zero trust capabilities, which are acquired through zero trust technologies. However, those additional technologies make architectures more complex, and expand their attack surface. As architectures become more complex, organizations may rely on third-party solutions, which may weaken security, as trust is granted to third-party products. Furthermore, zero trust new technologies introduce their own set of vulnerabilities and dependencies. Which leads to the question, *is trust in zero trust also misplaced?* To start answering this question, chapter 3 will be discussing the construction of a zero trust architecture.

Chapter 3

Building A Zero Trust Architecture

As described by section 2.2, a zero trust architecture must follow several principles: mandatory authentication, dynamic authorization, segmentation of access, encryption of data, and continuous monitoring.

This chapter describes a proof-of-concept zero trust architecture, which modifies and combines several open-source products. After describing how the architecture is built, its zero trust maturity is evaluated, and an insight on how it can be augmented to reach full zero trust maturity is provided.

3.1 Overview of the Proof-of-concept

The proof-of-concept zero trust architecture described in this section is based on Software-Defined Perimeters (SDP) [89], a zero trust architecture well-suited for networks comprising numerous IoT devices with unstable communication channels [71].

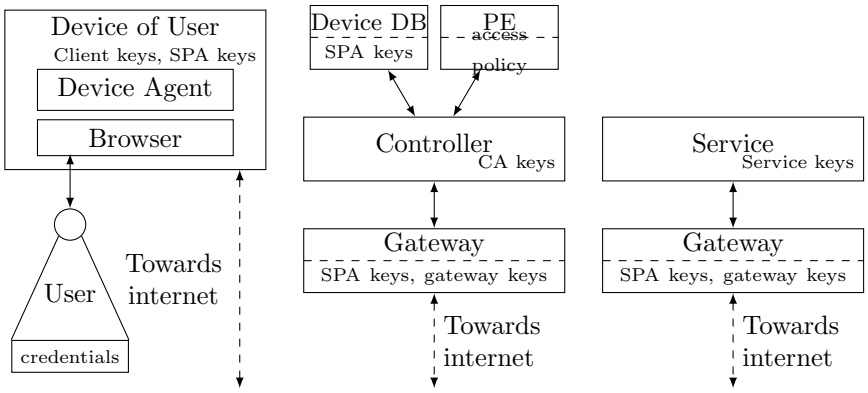
In an SDP architecture, the policy administrator and policy engine functions of a zero trust architecture are performed by a central component, called the *SDP Controller*. PEPs are split into two components: a *gateway* protecting the resource, preventing unauthorized access, and a *device agent* installed on the devices of requesters. An entity trying to access an SDP resource, and its device, which includes the device agent, form an '*Initiating Host*' (IH). A gateway and the resource it protects form an '*Accepting Host*' (AH). Secure communication channels in SDP are composed of two mechanisms: Single Packet Authorization (SPA) [327], which ensures that only pre-authorized entities can create a channel, and mutual Transport Layer Security (mTLS) [328], which creates end-to-end encrypted and authenticated channels.

In the proof-of-concept architecture, devices are simulated using Docker[1] containers, with Docker networks connecting containers to simulate network communications.

The base open-source implementation of SDP used in this proof-of-concept is from WaverleyLabs[2]. This implementation provides segmentation capabilities, as well as implementations of the SDP controller, of device agents, and of

[1] https://www.docker.com/

[2] https://github.com/WaverleyLabs/SDPcontroller



(a) Initiating Host. (b) SDP Controller. (c) Accepting Host.

Figure 3.1: Generic implementation of SDP components using Docker containers.

SDP gateways. It also includes communication protocols between components, including SPA. The implementation only defines access control for devices, through Access Control Lists (ACLs) stored in an SQL database. Specifically, for each authorized device accessing a service, there is a corresponding row in the SQL database representing this authorization.

3.2 Components of the Proof-of-concept

The proof-of-concept architecture implements three types of SDP components, depicted in figure 3.1.

An Initiating Host (IH), depicted in figure 3.1a, consists of a user (a human) and of a device (a Docker container). The device runs two processes: a browser, which the user interacts with to access services, and a device agent, that operates as superuser. The user does not have superuser access to the device. The device agent communicates with the SDP controller and with SDP gateways, to establish segments.

An Accepting Hosts (AH), depicted in figure 3.1c, is composed of two systems: one hosts the service, the other is the SDP gateway protecting the service. Traffic from other components must pass through the gateway to reach the service.

Finally, the SDP controller, depicted in figure 3.1b, consists of a system running the controller process, of an SDP gateway protecting the SDP controller, of a device database, used to store information about device agents and gateways, and of a Policy Engine (PE), described in section 3.2.4.

3.2.1 Onboarding and Secrets

Onboarding refers to the process of integrating an SDP component into the architecture. In the base SDP implementation, onboarding involves a manual process of adding each IH and AH to the database, followed by manual generation of key material and certificates.

In contrast, the proof-of-concept implementation automates this process. Entities are automatically added to the database from a JSON file describing them. The onboarding process also automatically generates key material, and stores it in a Docker volume, simulating the transmission of cryptographic material and of configurations into devices during onboarding. In a real-world implementation, cryptographic material and configurations would be generated by the SDP controller, uploaded on devices, either manually by a human operator, or automatically using an MDM system.

The architecture has its own PKI, with the Certificate Authority (CA) being the SDP Controller. The SDP controller uses the root CA key to sign certificates for all entities in the architecture. Devices are registered into the device database by the SDP controller, which records their SPA keys and certificates.

3.2.2 User Authentication

One of the AH in the architecture serves as the Identity Manager (IdM) of the domain. For this proof-of-concept implementation, the chosen open-source IdM is Keycloak[3].

Users are registered into the IdM, each uniquely identified by a username, and provided with their credentials to authenticate themselves. Although Keycloak enables multi-factor authentication, only one device is associated with each user in this proof-of-concept implementation, so multi-factor authentication has not been implemented.

Each user in Keycloak is described by attributes, *e.g.*, their name, email, or clearance level for accessing sensitive information. These attributes are used by the access policy, as described in section 3.2.4.

As detailed in section 3.3, when a user wants to access a resource, the device agent of their device contacts the SDP controller. The SDP controller generates a Security Assertion Markup Language (SAML)[4] request, containing a request for authenticating the user. This SAML request is forwarded to the user, who is redirected to the IdM. The user then logs in to the IdM, which generates a SAML assertion containing the attributes of the user, and is signed by the IdM.

An alternative to SAML requests is the use of OpenID Connect[5], which relies on JSON Web Tokens. According to [334], OpenID Connect provides more capabilities, *e.g.*, easy support of mobile devices. However, as it is a newer technology, transition from SAML to OpenID Connect has a high cost, and acceptance by the industry is slower.

In the proof-of-concept, the base SDP controller has been extended to generate SAML requests, to decode and verify SAML assertions, and to provide those attributes to the PE. To verify the authenticity of SAML assertions, the SDP controller retrieves, during onboarding, the certificate of the IdM.

[3] https://www.keycloak.org/

[4] OASIS, "Security assertion markup language (SAML) v2. 0 technical overview," Tech. Rep., Mar. 2008. [Online]. Available: https://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html.

[5] https://openid.net/developers/discover-openid-and-openid-connect/

3.2.3 Device Validation

Device monitoring and validation requires several components in the architecture: one device agent must be installed on every monitored device, and a server needs to be deployed to centralize monitoring results. For this proof-of-concept implementation, the open-source MDM system Fleet[6] has been chosen.

MDM device agents are services running on devices, that the MDM server queries to retrieve information about the monitored device. The device agent runs as superuser, preventing users from tampering with information produced by the device agent. Additional security can be provided by running the device agent inside a trusted execution environment.

As described in section 3.3, the SDP controller queries the MDM server to retrieve attributes describing the device of the user for access control. This has required modifications to the controller, but not to the MDM, as Fleet provides an API for querying device attributes.

3.2.4 Attribute-based Access Control

The base SDP implementation uses a static authorization system, stored in a database as an ACL: for every IH, the list of services it can access is stored.

In contrast, this proof-of-concept implementation uses an ABAC system, enabling the creation of flexible and dynamic access policies. The chosen open source PE is Open Policy Agent (OPA)[7].

While Keycloak also supports ABAC, the ABAC system in the proof-of-concept has been decoupled from the Identity Manager, as in chapter 6, access control is not necessarily performed by the domain providing identities. Moreover, decoupling the ABAC system from the Identity Manager enables the integration of monitoring attributes from the MDM system.

An ABAC system involves 'subjects', which are entities trying to perform an 'action', *e.g.*, read, write, or delete, on a 'resource', in an 'environment' [335]. Policies are expressed as a set of rules that take into account attributes from the subject, the resource, the action, and the environment, to authorize or deny the action. In this proof-of-concept architecture, 'resources' refer to services, *e.g.*, web servers, and the 'action' is a generic action 'access': the subject is either granted or denied access to the service. 'Subjects' are IHs, *i.e.*, both an entity and its device.

Therefore, the SDP controller retrieves attributes describing the user from the SAML assertion, attributes describing the device of the user from the MDM, and inputs them to the PE. Attributes describing AHs are configured into the PE during onboarding.

An example policy has been implemented. This policy describes services by two attributes: a `classification level` ('None', 'Confidential', 'Secret', or 'Top Secret'), and a whitelist of `authorized organizations` that can access the service. Users are described by three attributes: a `clearance level` with the same possible values as classification levels, an `affiliation`, and the `list of devices` they own. Devices are described by two attributes: a `list of`

[6] https://fleetdm.com/docs/get-started/why-fleet
[7] https://www.openpolicyagent.org/docs/latest/

`running processes`, and an `identifier`. The example access policy specifies the following conditions for an IH to access an AH:

1. the `clearance level` of the user must be greater or equal than the `classification level` of the service;
2. the `affiliation` of the user must be part of the `authorized organizations` listed for the service;
3. the `identifier` of the device must be included in the `list of devices` owned by the user;
4. the `running processes` of the device must include an antivirus software ('ClamAV'[8] was chosen);
5. access must be requested on working hours and days.

This policy enables the SDP controller to ensure that the device is owned by the user, that it complies with the policy 'all devices require a running antivirus', and that the user is authorized to access the service. The fifth condition is an example environmental attribute.

3.3 Access Workflow

Figure 3.2 depicts the access workflow that IHs initiate to connect to services protected by SDP.

Arrows 1 to 11 depict the authentication procedure. The device agent deploys a web application accessible through a browser, depicted in figure 3.3a. First, the user requests a list of available services from the device agent (arrow 1 in figure 3.2). The device agent then authenticates itself with the controller, using its SPA keys and TLS private key, creating a secure SDP channel (arrows 2 and 3). It then requests the list of services available to the user from the SDP controller (arrow 4).

As the user is not yet authenticated, the controller responds with an authentication request, containing information on how to reach the IdM (*e.g.*, its IP address, or domain name), and a login URL containing a SAML authentication request (arrow 5). Additionally, the controller configures the SDP gateway protecting the IdM to accept connections from the device of the user (arrow 6).

After receiving the authentication request, the device agent sends an SPA packet to the IdM to authorize connections (arrow 7), and redirects the user to the login URL (arrow 8). The user then opens an mTLS connection with the IdM (arrow 9), and authenticates themselves using their credentials (arrow 10), as depicted in figure 3.3b. The IdM returns an authenticated SAML assertion describing the attributes of the user (arrow 11).

This SAML assertion is then forwarded to and verified by the controller (arrows 12 and 13). Then, the controller queries the MDM server for device attributes (arrows 14 and 15). After receiving device attributes, the controller sends both the user and device attributes to the PE (arrow 16), which answers

[8] https://www.clamav.net/

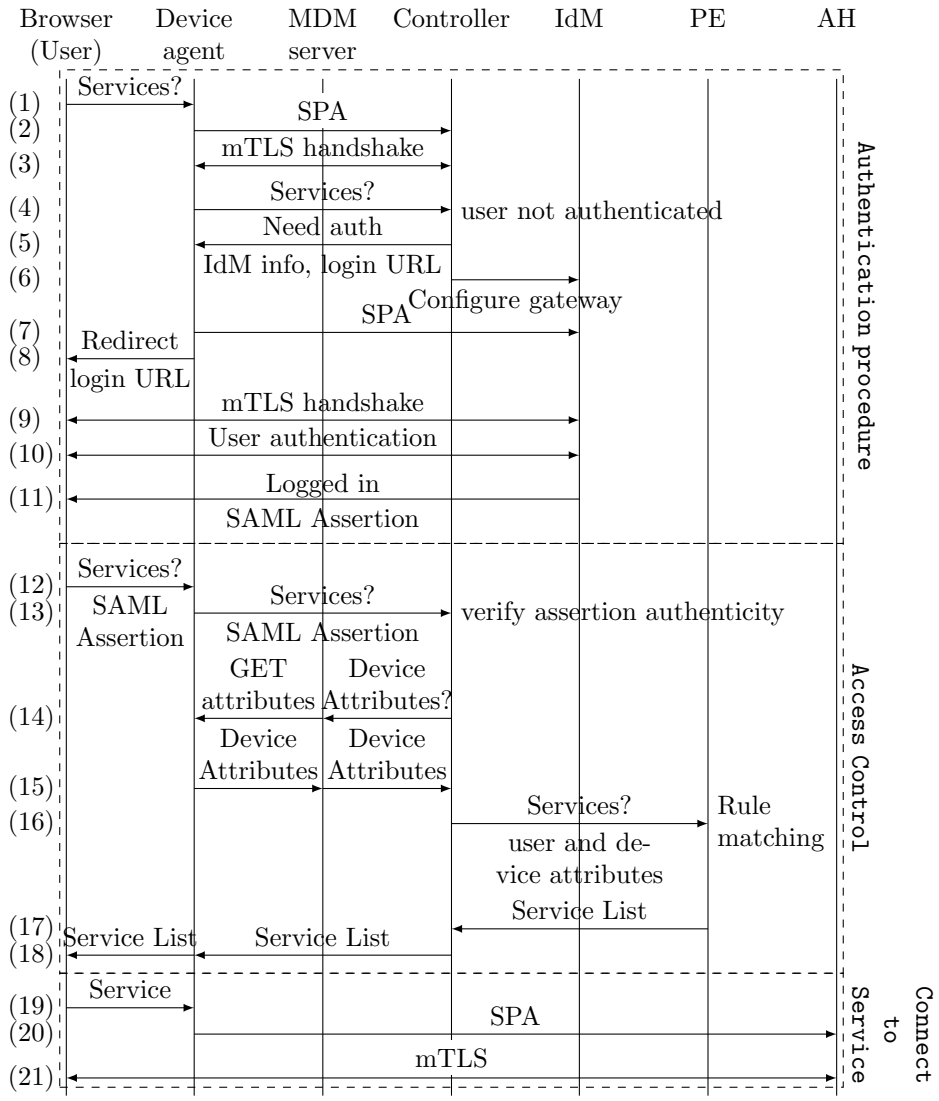


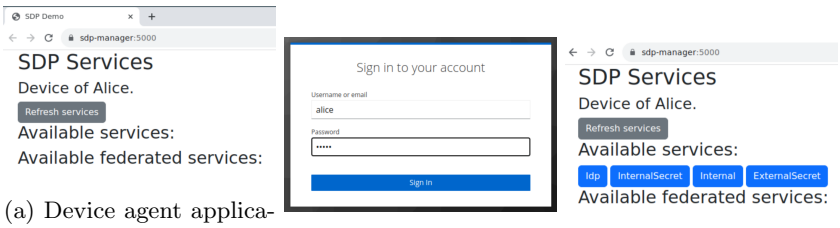
Figure 3.2: Access workflow of an IH in the proof-of-concept.

with the list of services available to the user (arrow 17). This list is forwarded back to the user (arrow 18), and displayed to them, as shown in figure 3.3c.

To access a service, the user selects the desired service in the browser (arrow 19). This prompts the device agent to send an SPA packet to the SDP gateway protecting the service (arrow 20). Finally, the user opens an mTLS connection with the service (arrow 21).

3.4 Zero Trust Maturity of the Proof-of-Concept

The WaverleyLabs SDP implementation provides segmentation capabilities based on IP addresses for IHs, resulting in a 'device' granularity. This granu-



(a) Device agent application. (b) Identity manager. (c) List of services.

Figure 3.3: Screenshots from the proof-of-concept.

larity is inherited in the proof-of-concept presented in this chapter. To achieve a finer granularity, an alternative zero trust technology would be required.

Moreover, the base SDP implementation does not provide user authentication capabilities, which is a core principle of zero trust. The introduction of the IdM into the proof-of-concept enables an initial maturity level for the identity zero trust pillar. As described in section 3.2.2, the IdM used in this proof-of-concept supports multi-factor authentication, enabling an advanced zero trust maturity level. To reach optimal maturity levels for identity, continuous authentication would be required. This would necessitate additional technologies to be added to the architecture.

The base SDP implementation provides device authentication. The introduction of MDM into the architecture enables device monitoring and validation, achieving an advanced zero trust maturity for the device pillar. Optimal zero trust maturity is outside the scope of this chapter, as it depends on organizational choices, *e.g.*, evaluating the supply chain trustworthiness.

In the base SDP implementation, authorization is static, based on ACLs. The introduced ABAC system enables the creation of flexible and dynamic policies, thus reaching an advanced maturity level.

The encryption provided by the base SDP implementation, based on mTLS, includes key management and rotation protocols, providing an advanced maturity level for network encryption. However, data at rest is not encrypted, making it necessary to improve Data-Centric Security (DCS), *i.e.*, primarily focusing on the security of data, instead of protecting servers and networks only.

No monitoring capabilities are provided by the base SDP implementation. Adding device monitoring is a starting point, but reaching an advanced maturity level for the monitoring pillar also requires network monitoring, *e.g.*, with an intrusion detection system, with resource monitoring, and with a SIEM system for centralized event analysis and for automated response after security incidents.

3.5 Summary

This chapter has presented a proof-of-concept implementation for a zero trust architecture. This proves the possibility of building a zero trust architecture from existing products and technologies. The benefits of this proof-of-concept have been assessed by evaluating its zero trust maturity, following the method

from section 2.3, by estimating its zero trust capabilities as listed in table 2.3 on page 30.

Results show that the proof-of-concept provides capabilities for user, application, and device authentication, identity management, device validation, micro-segmentation, end-to-end encryption, isolation, and dynamic authorization, achieving an advanced to optimal zero trust maturity for the identity, device, network, workload and applications, and policy zero trust pillars. This shows that zero trust is a framework improving the security posture of organizations.

However, the data, visibility and analytics, and automation and orchestration zero trust pillars only have an initial zero trust maturity level, requiring additional technologies to reach higher levels. Part of these challenges, *e.g.*, about the data pillar, are identified more precisely and addressed in part III of this manuscript. Moreover, part IV of this manuscript explores uses of zero trust architectures that are not explicitly covered by the zero trust framework, *e.g.*, the privacy of users and the interaction of zero trust architectures with other domains.

Part III

Improving the Maturity of Zero Trust Architectures

Chapter 4

Data-Centric Security Protections in Zero Trust Architectures

Many approaches for securing data rely on network-centric policies and methods [92]: systems on which data is stored are isolated, with limited entrypoints called *endpoints* [50]. An alternative approach, *data-centric security*, prioritizes safeguarding data itself, rather than prioritizing the protection of devices and networks that store and transmit data [215].

As presented in chapter 2, zero trust architectures are build upon several technology *pillars* [82], [97]. In the representation from the DoD, depicted in figure 2.7 (page 30), data serves as the central pillar, as protecting data is a primary goal of zero trust architectures [82]. According to [97], a data-centric security approach is necessary to follow zero trust principles.

Nevertheless, despite data protection being a core principle of zero trust, methods for protecting data at rest, and for performing dynamic, least-privilege access control for data, have seldom been studied in the zero trust literature [92].

This chapter introduces a data-centric approach to zero trust security, *i.e.*, which prioritizes the protection of data. The proposed method builds upon a base zero trust architecture, and adds data-at-rest encryption using Attribute-Based Encryption, combining access control and data protection. This allows for fine-grained access control at the data object level, thereby preventing data exfiltration and offering increased flexibility in restricting data access. A proof-of-concept implementation demonstrates the security benefits of this approach and highlights the trade-offs involved in employing data encryption.

4.1 Statement of Purpose

In a zero trust architecture, every resource is protected by a Policy Enforcement Point (PEP) [83], depicted in figure 2.1 on page 20. When a connection request is made, the PEP intercepts it, and the Policy Decision Point (PDP) evaluates it, granting or denying access to the resource based on information describing the requester, its device, the resource, and the environment. This information can be represented as *attributes*, enabling authorization through

Attribute-Based Access Control (ABAC) [336]. Everything 'behind' the PEP is considered trusted, as described in chapter 2. Therefore, one PEP can only protect data and services with the same level of sensitivity, as otherwise, a user with lower clearance level could potentially move laterally, and gain unauthorized access to more sensitive data.

This chapter explores how to isolate co-located data objects with varying levels of sensitivity, as well as the integration of data-centric security measures within a zero trust architecture.

4.1.1 Related Work

Data-Centric Security

Data-centric security implies several key properties [82]:

- Data Inventorying and Categorization, which automatically labels data with attributes describing it;
- Data Rights Management (DRM), *i.e.*, authorizing and blocking access to data dynamically;
- Data Loss Prevention (DLP), *i.e.*, blocking suspected data exfiltration;
- Dynamic Data Masking (DDM), *i.e.*, dynamically masking and altering data to prevent unauthorized data access.

For example, a military agency may store confidential documents on a server, accessible only to personnel with a 'secret' clearance level. Despite enforcing access control to access the server, data-centric security remains crucial for enforcing need-to-know policies. With DRM, access can be restricted only to documents needed for missions, regardless of the clearance level of the user. Furthermore, DDM can be used to dynamically redact classified information within documents. This may particularly be useful in a coalition network environment to mask sovereign information.

While those properties are needed to attain an optimal maturity level for the data pillar, most architectures presented in chapter 2 offer none of the aforementioned properties.

Data-Centric Security in Zero Trust Architectures

A potential solution for achieving data isolation and finer-than-system granularity is the 'device application sandboxing' deployment model proposed by NIST [83], depicted in figure 2.10c on page 45. In this model, an application executes within a compartment, *e.g.*, a container or a virtual machine, and communication with the compartment is protected by a PEP. Organizations must guarantee the integrity and isolation of compartments. However, this solution demands greater effort compared to other deployment models, as it necessitates securing, and dynamically creating, compartments for data and services [83].

An alternative approach to achieving data isolation is the encryption of stored data. Various methods have been proposed for encrypting data and managing keys in zero trust architectures [128]. In Horus [337], data is encrypted using keyed hash trees, which offers fine-grained security: regions of

data are encrypted with different keys, so if one key is compromised, only part of the data can be decrypted. In Tahoe [338], each file is split into different parts, distributed across multiple servers. The file can only be decrypted by entities that have access to at least K parts of the file. Moreover, blockchain technology has been proposed for developing a data storage scheme for Internet of Things (IoT) devices [339], however with efficiency issues. Those methods protect data from a malicious or compromised storage server, as an unauthorized entity gaining access to a storage server cannot decrypt stored data. However, those solutions do not prevent in themselves data exfiltration from a malicious or compromised authorized user, and they also do not enable data rights management for expressing fine-grained authorization.

Combining Attribute-Based Encryption with Attribute-Based Access Control

As Attribute-Based Access Control (ABAC) is a basis for zero trust authorization, [123] proposes a zero trust architecture for Industrial IoT, which uses Attribute-Based Encryption (ABE) to create a customized ABAC scheme.

Attribute-Based Encryption [340], [341] applies attribute-based policies to encrypt data, *i.e.*, only entities with matching attributes can decrypt ciphertexts. ABE schemes are defined by four cryptographic algorithms: `Setup` creates public parameters, a public key, and a master private key; `Encrypt` generates ciphertexts; `KeyGen` produces decryption keys, based on a set of attributes; and `Decrypt` decrypts ciphertexts.

ABE categorized into two types: Ciphertext-Policy Attribute-Based Encryption (CP-ABE), and Key-Policy Attribute-Based Encryption (KP-ABE). In CP-ABE, decryption keys are linked to user attributes, and ciphertexts contain an access policy. In KP-ABE, ciphertexts are labeled with a set of attributes, and decryption keys are associated with access policies. For the purpose of using ABE to create ABAC policies, CP-ABE is preferred, because users are granted decryption keys that correspond to their attributes, and data is encrypted to include the access policy.

Statement of Purpose

This chapter proposes a method for extending existing zero trust architectures to provide data-centric security, by leveraging Attribute-Based Encryption (ABE), enabling data rights management and data loss prevention. Continuous and automatic data inventorying is outside the scope of this chapter.

A proof-of-concept implementation that extends the base zero trust architecture presented in chapter 3 is described in this chapter. The addition of data encryption provides an advanced to optimal maturity level for the zero trust data pillar. Moreover, the proposed method further advances the 'deperimeterization' process of zero trust: instead of 'trusting everything' beyond the PEP, the proposed architecture offers a finer granularity, at the level of individual data objects.

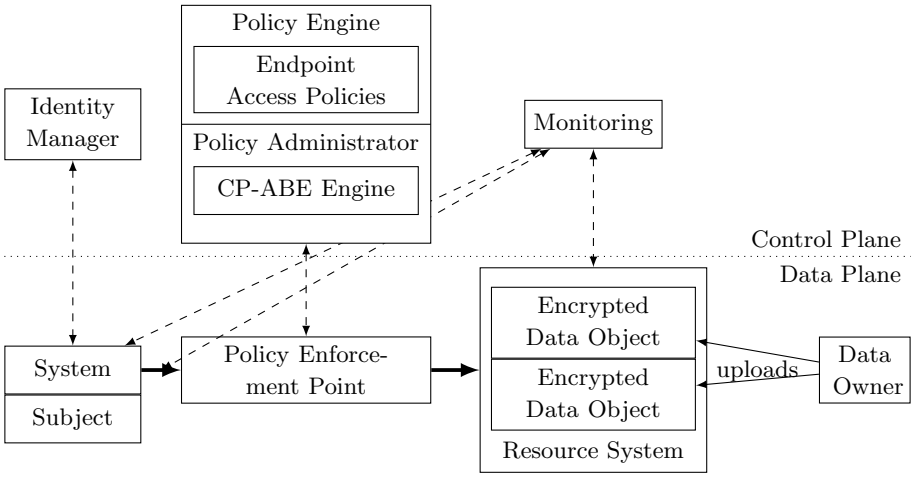


Figure 4.1: Data-centric zero trust architecture.

4.1.2 Paper Outline

The remainder of this chapter is organized as follows. Section 4.2 provides a detailed description of the method for integrating ABE into a base zero trust architecture. Section 4.3 presents an analysis of the security guarantees offered by this architecture, and discusses the benefits and limitations of ABE in the context of a zero trust architecture. In section 4.4, the proposed method is applied to the zero trust architecture – based on Software-Defined Perimeters – described in chapter 3, illustrating the necessary changes that need to be performed to zero trust components. Finally, section 4.5 concludes this chapter.

4.2 Augmenting Zero Trust Architectures with Attribute-Based Encryption

This section describes a method for integrating data-at-rest encryption into an existing zero trust architectures, using an Attribute-Based Encryption (ABE) scheme.

4.2.1 Overview

In addition to the zero trust components defined by NIST [83], which are depicted in figure 2.2 on page 21, the resulting architecture proposed in this chapter, depicted in figure 4.1, incorporates a *Data Owner* (DO) and a cryptographic engine.

Data is stored in an encrypted form on servers, thus preventing unauthorized access to data, even if the server is untrusted or compromised. This necessitates the DO to be able to encrypt data, and authorized entities to be able to decrypt data.

4.2.2 Data Owner and Data Upload

The DO is responsible for uploading data on organization systems and for defining data access policies. The trustworthiness of the DO is ensured through zero trust authentication and access control.

Data encryption prevents entities that have gained unauthorized access to a data server from accessing the data. Two cryptographic schemes are used: a Symmetric Encryption (SE) scheme, *e.g.*, the Advanced Encryption Standard (AES) [216], and an ABE scheme [340], [341].

An SE scheme is defined by three algorithms: `SE-KeyGen` generates a symmetric key; `SE-Enc` encrypts data using the key; and `SE-Dec` decrypts encrypted data using the key.

An ABE scheme applies attribute-based policies to encrypt data, *i.e.*, only entities with matching attributes can decrypt data. An ABE scheme is defined by four cryptographic algorithms: `ABE-Setup` generates master public and private keys; `ABE-Encrypt` encrypts data; `ABE-KeyGen` generates decryption keys based on a set of attributes; and `ABE-Decrypt` decrypts encrypted data. In the method described in this chapter, a Ciphertext-Policy ABE scheme (CP-ABE) is required, *i.e.*, decryption keys are linked to entity attributes, and ciphertexts contain an access policy.

When the architecture is deployed, the Policy Administrator (PA) uses the `ABE-Setup` algorithm to generate a master public key, pk, and a master secret key, mk. The public key is shared with every entity within the architecture.

The algorithm used by the DO to encrypt and upload data to organization systems is described in algorithm 4.1.

Algorithm 4.1: Algorithm for uploading encrypted data.

Input: data `d`, policy `ap`, public key `pk`, encryption time t_{enc}.
Output: encrypted data `c`.

1 `k` $\xleftarrow{\$}$ `SE-KeyGen()`
2 $c_0 \xleftarrow{\$}$ `SE-Enc(`k, d`)`
3 `ap` $\leftarrow$ `ap` $\wedge$ (key_time $> t_{\text{enc}}$)
4 $c_1 \xleftarrow{\$}$ `ABE-Encrypt(`pk, k, ap`)`
5 **return** $c \leftarrow (c_0, c_1)$

First, the DO generates a random symmetric key, k (step 1 of algorithm 4.1). Then, the DO uses key k to encrypt the data with the SE scheme (step 2). A special condition, 'key_time $> t_{\text{enc}}$', is added to the access policy in step 3. This step is performed to prevent key reuse, enabling a dynamic access control policy, as detailed in section 4.3.3. The key k is then encrypted using the ABE scheme (step 4). This ensures that only entities with attributes matching the access policy defined by the DO can decrypt the key k. Finally, the DO uploads both the encrypted data and the encrypted key to the data storage (step 5). The use of the SE scheme, instead of directly encrypting data with the ABE scheme, is for performance reasons.

4.2.3 Zero Trust Access To Data

Accessing data in the architecture presented in figure 4.1 is performed in two steps: first the subject gains access to the system where the data is stored;

then the PDP provides the subject with an ABE decryption key, enabling the subject to decrypt the data.

Access to the Data System

Access to the data system is verified following the authorization procedure of the base zero trust architecture: the access request is intercepted by the PEP, and is evaluated by the PDP, using ABAC. Attributes are obtained from the Identity Manager (IdM), which produces identity attributes for the subject and its system, and from monitoring components.

This access control mechanism only protects access to the data system. More granular access control, *e.g.*, for individual data objects, is implemented through data encryption and decryption.

Data Decryption

In addition to transmitting the list of attributes describing the subject and its device to the ABAC system, the PA generates an ABE decryption key corresponding to this list of attributes, L, using the master secret key, msk, of the ABE scheme:

$$\mathrm{k_L} \leftarrow \texttt{ABE-KeyGen}(\mathrm{pk}, \mathrm{msk}, \mathrm{L} \wedge (\mathrm{key\_time} = t_k)).$$

The generation time of the key, represented by the attribute 'key_time = t_k', is included in the attribute list, for matching conditions added in step 3 of algorithm 4.1. The decryption key is then transmitted to the subject on a secure control channel.

Having retrieved both the encrypted data and the decryption key, the subject decrypts the data, by first using the ABE decryption key $\mathrm{k_L}$ to decrypt – if authorized by the access policy contained in the encrypted data – the SE key that was used to encrypt each data object. Once SE keys are decrypted, data objects are decrypted using the `SE-Dec` algorithm.

4.3 Security Analysis

As access control to resource systems is unchanged by the architecture proposed in section 4.2, the security provided by the base zero trust architecture remains intact. This section analyses additional security benefits of the architecture, and discusses the trade-offs involved in achieving this enhanced security.

4.3.1 Data-centric Security

Data is stored in an encrypted form on organization systems. Because decryption keys are never present on these systems, entities having access to data systems cannot decrypt stored data beyond what they are authorized to access. Indeed, each encrypted data object encapsulates its own access policy, and only entities whose attributes match the access policy can decrypt it. Therefore, data can safely be stored on untrusted storage systems, *e.g.*, servers rented from cloud providers.

Two possibilities are considered for enabling access to data.

If the access policy for each data object is readable by the PEP, then the PEP evaluates the access policy of each data object with the attributes contained in the access request, and prevents the transmission of data when the attributes do not match the access policy. By filtering data at the PEP level, the subject only receives data objects they can decrypt, without revealing the existence of unauthorized data objects. However, the PEP needs to know the access policies of every data objects, which may reveal private information.

An alternative method is to allow subjects to access all encrypted data. Because of ABE, even if a subject has access to all encrypted data objects, they can only decrypt the ones for which their attributes match the access policy of data objects. This method is particularly useful for protecting files containing information with different sensitivities. To achieve this, each part of the file is encrypted with a different access policy. When a subject requests the file, the entire encrypted file is sent. The subject can only decrypt the authorized parts because of the ABE encryption. While access policies may remain private, the subject learns the existence, and the size, of data they are not authorized to decrypt.

4.3.2 Relationship between ABAC and ABE

ABAC policies are typically expressed using standard policy languages, *e.g.*, the eXtensible Access Control Markup Language (XACML) or Next Generation Access Control (NGAC) [269]. XACML is based on XML, and defines attributes as name-value pairs, which are combined in boolean conditions to create rules. In NGAC, attributes are defined as nodes in a graph, and rules are expressed as links in the graph. Authorization is performed by finding a path linking user attributes to object attributes in the policy graph.

ABE schemes rely on tree structures to define access control policies [342]. In an ABE policy tree, leaves represent attributes, and non-leaf nodes represent threshold gates. Given subject attributes, a leaf of the tree is *valid* if it corresponds to one of the subject attributes. A threshold gate with parameter k is valid if the gate has at least k valid children. Authorization is granted if the root of the tree is valid.

Configuration of ABE policies

Unlike ABAC, there is no standardized language for configuring ABE policies [341]. Thus, more significant setup work is required to implement ABE policies compared to ABAC policies. Furthermore, ABE policies need to be incorporated into each data object individually, which makes policy updates more challenging to apply. In contrast, in ABAC, policies are centralized in the PDP, thus updating policies is performed at a single location. To enable more flexibility and to simplify policy management, revocable ABE [343] enables the revocation of attributes in access policies, and ABE with policy updating [344], [345] updates the policies of ciphertexts.

Policy expressiveness

Base ABE lacks the expressiveness of ABAC policies [341]. To address this limitation, [346] extended tree structures of ABE, by adding information on

the leaves of policy trees, enabling the use of more operators, *e.g.*, negation and number comparisons. The generator of decryption keys evaluates the validity of extended leaves to generate keys. This negatively impacts the performance of ABE, as one decryption key has to be generated for each data object (or for groups of data objects if policies can be combined), instead of only one decryption key per subject.

To prevent the generation of a separate key for each data object in the architecture proposed in section 4.2, when a subject requests access to multiple data objects with different conditions 'key_time $> t_i$', it sends the list of encryption times $t_1, t_2, ..., t_s$ to the PA, which generates a single decryption key containing all attributes 'key_time $> t_i$'.

4.3.3 Dynamic Access Control

A core principle of zero trust architectures is dynamic access control. For instance, an access policy may require subjects to use up-to-date devices to access data, and the up-to-date status needs to be regularly assessed. In the architecture presented in section 4.2, when a subject is authorized to access data, the PA provides an ABE decryption key matching its attributes. Thus, if the device of the subject is up-to-date, the PA generates and provides to the subject a decryption key corresponding to the 'device up-to-date' attribute. However, subsequent data objects can still be decrypted, by using this same decryption key, regardless of device updates. Therefore, two solutions for achieving dynamic access control are proposed.

The first solution relies on the dynamic access control provided by the base zero trust architecture. In the above example, if the subject no longer has an up-to-date device, then it is denied access to the server. However, if the attributes of an entity change, but the entity is still allowed to access the server (*e.g.*, if a user changes job within the same organization), then dynamic access control to data is not ensured.

An alternative solution is the prevention of key reuse, by introducing time constraints on decryption keys. When encrypting a data object, the DO adds a time constraint on the time of generation of the key, ensuring that only keys generated after the encryption of the data object can decrypt it (line 3 in algorithm 4.1). While this method ensures dynamic access control, it requires rough time synchronisation between the DO and the PA, and the use of an expressive ABE scheme, which negatively impacts the performance of ABE operations as detailed in section 4.3.2.

4.4 Example Extension of a Zero Trust Architecture for Data-Centric Security

This section describes the implementation of the method from section 4.2 as a proof-of-concept architecture. The base zero trust architecture is the Software-Defined Perimeters (SDP) proof-of-concept presented in Chapter 3.

In an SDP architecture, the PA and PE functions are performed by the SDP controller. Therefore, a CP-ABE engine is integrated into the SDP controller. Moreover, every IH that needs to decrypt data can perform ABE decryption operations.

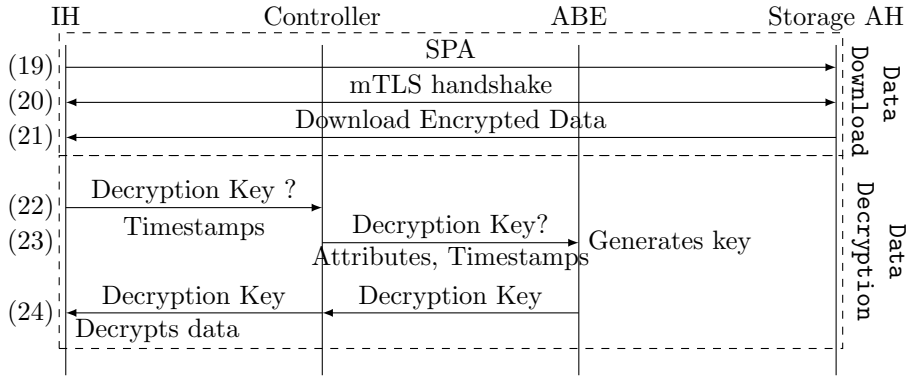


Figure 4.2: Access workflow of an IH for downloading and decrypting data.

4.4.1 Access Workflow

As presented in section 4.2, access control to the storage system remains unchanged. The difference with the base architecture is that data is stored encrypted on the storage system, and subjects must possess decryption keys to decrypt the data. To achieve this, the SDP controller includes an additional component: the CP-ABE engine. When the controller is deployed, the CP-ABE engine generates CP-ABE parameters: a master private key and associated public key, as well as cryptographic parameters. When a data owner is onboarded, the cryptographic parameters and the public key are sent to the data owner. To upload a data object on the storage server, the data owner first encrypts it, using the procedure defined in algorithm 4.1 described in section 4.2.

Figure 4.2 depicts the workflow for a subject to decrypt downloaded data. It is a direct extension of the access workflow for the base architecture, depicted in figure 3.2 on page 62. It illustrates the operations described in section 4.2.3 for the proof-of-concept architecture.

The original access workflow to gain access to the storage server remains unchanged: the user authenticates to the Identity Manager, the SDP controller retrieves device attributes from the MDM, and uses the ABAC engine for authorizing access (arrows 1 to 18 in figure 3.2).

Once the IH is authorized to access the storage server, it connects to it through SPA and mTLS (arrows 19 and 20), and downloads encrypted data objects (arrow 21).

Once encrypted data objects downloaded, the IH extracts timestamps from their embedded access policy, and requests a decryption key corresponding to these timestamps to the SDP controller (arrow 22). The SDP controller requests an ABE decryption key corresponding to the IH attributes, by sending them, as well as the requested timestamps, to the ABE system (arrow 23). The resulting decryption key is then transmitted to the IH (arrow 24), which decrypts data objects as the object access policies authorize.

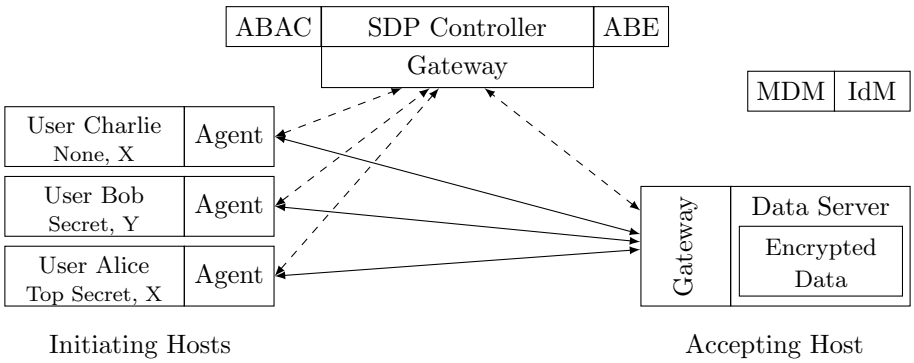


Figure 4.3: Architecture of the SDP-based proof-of-concept.

4.4.2 Proof-of-concept Implementation

An overview of the proof-of-concept architecture is depicted in figure 4.3. The architecture consists of several components: the SDP controller, which uses both an ABAC engine and an ABE engine, the Identity Manager, the MDM system, a storage system on which data is stored in an encrypted form, and three users: Alice, Bob, and Charlie.

The users have different clearance levels and departments: Alice and Charlie are both part of the X department, and Bob of the Y department. Alice has a top secret clearance level, Bob a secret clearance level, and Charlie no clearance level.

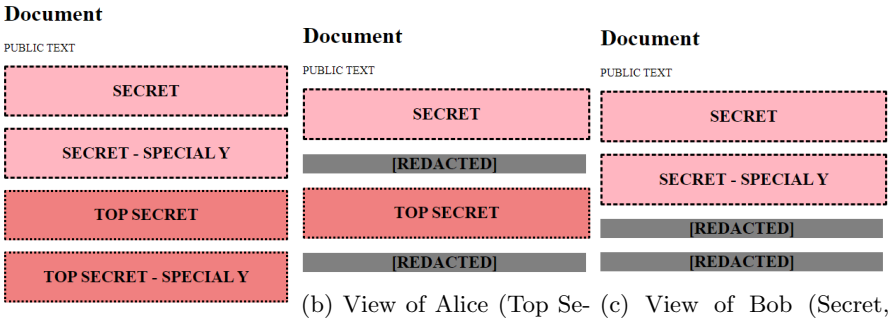
Access to the storage server is restricted to users with a clearance level 'secret' or higher, regardless of department. As a result, Charlie is denied access to the server, but both Alice and Bob can access it. The storage server contains an encrypted document, depicted in figure 4.4a. The document is composed of five parts: a public part, a secret part, a secret part reserved to Y personnel, a top secret part, and a top secret part reserved to Y personnel.

Adding Data Centric Security

Enhancing the base SDP architecture with data centric security requires the installation of ABE engines in several components: in the SDP controller, to generate ABE keys for IHs, on the device of the data owner, for encrypting data, and within IHs, to decrypt downloaded data. The ABE engine chosen for this proof-of-concept is OpenABE[1]. Only the SDP controller requires the master secret key of the ABE scheme, which is used for generating decryption keys. Other components only require the master public key to encrypt data, or to decrypt data using decryption keys generated by the SDP controller. To accommodate multiple access control policies in a single document, each part of the document is encrypted independently.

Figure 4.4 illustrates the fine-granularity of access control achieved by the proof-of-concept: from the same encrypted document, whose cleartext form is depicted in figure 4.4a, Alice and Bob each view a differently redacted document, respectively depicted in figures 4.4b and 4.4c. As Alice has a top secret

[1] https://github.com/zeutro/openabe



(a) Original document.

(b) View of Alice (Top Secret, X).

(c) View of Bob (Secret, Y).

Figure 4.4: Document uploaded on the data server in the proof-of-concept.

clearance level, she can access both secret and top-secret parts of the document, but not parts reserved to Y personnel. In contrast, Bob, with his secret clearance level, can only view secret parts of the document, but has access to parts reserved specifically for Y personnel.

4.5 Summary

A fundamental aspect of zero trust is data protection. However, state-of-the-art zero trust architectures primarily focus on securing networks and systems, neglecting comprehensive approaches to data-centric security and data-at-rest encryption.

This chapter has filled the gap between endpoint-oriented zero trust architectures and data-centric security, by presenting a method for enhancing an existing zero trust architecture with Attribute-Based Encryption. The resulting architecture inherits the security properties of the base zero trust architecture, while introducing novel capabilities, *e.g.*, data rights management and access control with finer granularity, ensuring data-centric security. A proof-of-concept implementation demonstrates how this method can be applied to extend the proof-of-concept architecture from chapter 3 to enhance it with data-centric security, addressing the data pillar in the zero trust maturity model.

Chapter 5

Continuous Authentication in Secure Messaging

As presented in chapter 2, identity is one of the core pillars of zero trust architectures [97]. To attain an optimal maturity level, a continuous authentication of entities is required. Moreover, the automation of procedures such as continuous authentication is also a zero trust goal, as it facilitates the deployment and use of zero trust architectures.

In multiple secure messaging schemes, continuous authentication relies on out-of-band channels to verify the authenticity of long-running communications, *i.e.*, requires manual actions to be performed by users. Such out-of-band checks are seldom performed by users in practice.

In this chapter, a novel method for performing continuous authentication automatically is proposed, without the need for an out-of-band channel. The proposed method leverages the long-term secrets of users, extending the authenticity of sessions as long as long-term secrets of users are not compromised, strengthening the post-compromise security of the Signal messaging protocol [347]. Moreover, the proposed mechanism further enables the detection of long-term secrets compromise using an out-of-band channel.

The proposed protocol comes with a novel, formal security definition capturing continuous authentication, a general construction for messaging protocols, and a security proof for the proposed protocol. An implementation of the protocol is described, which seamlessly integrates on top of the existing Signal library, together with bandwidth and storage overhead benchmarks.

5.1 Statement of Purpose

The Signal end-to-end encrypted messaging protocol[1] is used by billions of people[2], in the Signal app itself and other messengers such as Facebook Messenger[3]

[1] *Signal: Technical information.* [Online]. Available: https://signal.org/docs/.

[2] WhatsApp, *Whatsapp security advisories*, 2023. [Online]. Available: https://www.whatsapp.com/security/advisories.

[3] Facebook, *Messenger secret conversation, technical whitepaper*, 2016. [Online]. Available: https://about.fb.com/wp-content/uploads/2016/07/messenger-secret-conversations-technical-whitepaper.pdf.

and WhatsApp[4]. The security of encryption keys used in the Signal protocol relies on two composed cryptographic protocols. First, a Diffie–Hellman-style key exchange protocol involving long-term asymmetric keys (whose public part is distributed via a central Signal server) is used to derive a shared secret. This initial shared secret is then used by parties in Signal's Double Ratchet protocol[5] to derive symmetric keys, used to encrypt messages between the two communicating parties.

5.1.1 Related Work: Security of Signal

There have been numerous analyses of the security of the Signal protocol, as has been recapitulated in [136], which show that security properties for messaging protocols come in a variety of flavours with different adversary powers and strengths.

For these analyses, models separate different types of secrets: *session secrets* (like ephemeral randomness or state), which are used throughout the Double Ratchet protocol, and *long-term secrets*, used only in the initial key agreement. Some [347], [348] study the security of the Signal protocol in its entirety, including the X3DH key exchange. Others [349]–[354] focus exclusively on the ratcheting part of the protocol, thus considering only session secrets. Among other security properties, [348] and [349] confirm that against strong adversaries who control the network and can adaptively compromise session and long-term secrets, Signal offers forward secrecy (meaning that the secrecy of messages sent before a secret leakage are still secure) as well as post-compromise security [347] (meaning that after users exchange *unmodified* messages, security is restored or "healed").

The security of real secure messaging implementations is also evaluated in [355], with a focus on (de)synchronization. Somewhat similar to the setting of this chapter, their analysis involves an adversary trying to break post-compromise security by impersonating a compromised user and finding discrepancies between the implementations and the formal specifications.

Additionally, [352] proposes a construction for secure messaging by signing message transcripts, focusing though on healing communication under *passive* attacks while this chapter aims at detecting and preventing *active* attacks.

As pointed out by [356], the definition of post-compromise security is quite restrictive in the sense that the adversary needs to remain completely passive for security to be restored. Indeed, if the adversary remains active after a state leakage and continuously injects forged messages, authenticity is never restored. [356] instead proposes a protocol relying on an out-of-band channel (like email, SMS, or an in-person meeting) to detect such active adversaries, leveraging additional *fingerprints* computed by the protocol and compared over the out-of-band channel. However, while detection clearly is a good step towards mitigating attacks, it does not prevent the actual attack from continuing.

[4]WhatsApp, *Whatsapp security*, 2021. [Online]. Available: https://www.whatsapp.com/security/.

[5]T. Perrin and M. Marlinspike, *The Double Ratchet algorithm*, 2016. [Online]. Available: https://whispersystems.org/docs/specifications/doubleratchet/.

5.1.2 Locking Out Active Adversaries.

The question for this chapter goes one step further than [356]:

Is it possible, post-compromise, to lock out even an active adversary from a messaging communication?

Clearly, the answer in general is: No. An active network adversary that fully compromises a user's device, including all session and long-term secrets can, by design, fully impersonate that user subsequently. Instead, this chapter examines a weaker property, by distinguishing – and thus better leveraging – the difference between the use of session and long-term secrets in a messaging protocol. Indeed, long-term secrets may be harder to compromise, *e.g.*, due to stronger randomness sources or better protection in hardware. This is the case for WhatsApp or Signal, which can be deployed on devices which have access to secure hardware such as Trusted Platform Modules (TPM) [241], in which long-term secrets can be stored more safely[6]. Phones on the other hand can use smart-cards to store their long-term keys. A typical attack scenario are border searches, where a travelling user would need to give away their phone or laptop for analysis, risking the leak of their session secrets. An adversary could then remain an active Man-in-the-Middle (MitM), possibly on nation-controlled networks, yet long-term keys in smart cards or a TPM might not have been leaked. Such a breach of authenticity can have a high impact in the case of Signal as sessions are typically months or years long.

Thus, the problem statement for this chapter is as follows:

Is it possible, post-compromise, to lock out even an active adversary from a messaging communication that compromised session, but not long-term secrets?

Notably, the answer for Signal is still: No. This chapter demonstrates how to, generically, turn this into a Yes.

5.1.3 Chapter Outline

The remainder of this chapter is organized as follows. Section 5.2 presents a formal, game-based definition of *continuous authentication*, a post-compromise security property locking out active adversaries who have not compromised long-term secrets. Section 5.3 presents the Signal protocol, and proves that it does not meet the security requirement of section 5.2. Therefore, section 5.4 proposes a *generic extension* for messaging protocols, to provide them with provably-secure continuous authentication. The proof of security is established in section 5.5. A prototype implementation of this extension for Signal and an *analysis and benchmark* of the overhead it introduces is presented in section 5.6. While implementing the extension, a discrepancy between the post-compromise security offered by Signal's official library and what is claimed in the literature has been exposed, and is presented in section 5.7. Finally, section 5.8 concludes this chapter.

[6] WhatsApp, *How whatsapp enables multi-device capability*, 2021. [Online]. Available: https://engineering.fb.com/2021/07/14/security/whatsapp-multi-device/.

5.2 Continuous Authentication

This section presents what locking out an active adversary from a messaging communication formally means. The basis for this is a formal syntax, following [349], that captures generic *messaging schemes* that operate on an unreliable channel, derive an initial shared session secret using long-term keys, and then derive further session secrets from previous state, new randomness and possibly long-term keys. This in particular encompasses the Signal Double Ratchet protocol.

A game-based security definition for *continuous authentication* is then presented, which guarantees two core properties:

1. An active MitM adversary that compromised a user's communication state gets *locked out* of the communication, *unless* it has also compromised the user's long-term secrets.

2. Users can correctly *decide whether* long-term secrets have been compromised in an active attack, via an out-of-band channel.

The first property captures the desired strengthening of messaging protocols. In the Double Ratchet protocol[7], all secrets are derived from prior established secret state (the so-called *root* and *chain* keys) and new randomness generated by the users (the Diffie–Hellman *ratchet* keys). The former is revealed in a full state compromise, and the latter are unauthenticated, and can hence be impersonated by the adversary. Therefore, an adversary can conserve its Man-in-the-Middle position indefinitely, without being detected in-band.

The second property improves on a related issue: If long-term secrets *are* compromised, then security is not restored if users only close their current session and reopen a new one, but users will instead need to generate and distribute new long-term keys. This procedure however is cumbersome and typically involves manual effort, so ideally users would like to better know *when* it is indeed necessary. Continuous authentication offers such a checking mechanism, enabling users to only change their long-term secrets if they have indeed been leaked.

5.2.1 Messaging Schemes

A messaging scheme consists of several algorithms: The core algorithms, following [349], are used to create users (Register), initiate sessions between them (InitState) and let them send and receive messages (Send and Recv). This definition supports an arbitrary number of users, but only two-party sessions (*i.e.*, no group chats).

In addition to the four core messaging algorithms, the formalization introduces StartAuth, a procedure which can be used to initiate an in-band authentication step[8], and DetectOOB, a procedure that compares the states of two session participants out-of-band, and decides whether an adversary has used a long-term secret to avoid in-band detection.

[7] T. Perrin and M. Marlinspike, *The Double Ratchet algorithm*, 2016. [Online]. Available: https://whispersystems.org/docs/specifications/doubleratchet/.

[8] This procedure can leave the state unchanged if the messaging scheme, like the original Signal protocol, does not support in-band authentication.

Definition 5.1 (Messaging scheme)**.** *A* messaging scheme $\mathtt{MS} = (\textsc{Register}, \textsc{InitState}, \textsc{Send}, \textsc{Recv}, \textsc{StartAuth}, \textsc{DetectOOB})$ *consists of six probabilistic algorithms:*

- $\textsc{Register}$ *creates a user* U *outputting long- and medium-term information and secrets:* $(LTI_U, LTS_U, MTI_U, MTS_U) \xleftarrow{\$} \textsc{Register}()$.
- $\textsc{InitState}$ *takes as input the long- and medium-term secret of a user* U*, the long-term information of a user* V*, and optionally some public information, and creates an initial session state for* U *to communicate with* V*:* $\pi_U \xleftarrow{\$} \textsc{InitState}(LTS_U, MTS_U, LTI_V, PI)$.
- $\textsc{Send}$ *takes as input the state* π_U *and long-term secrets* LTS_U *of the sender as well as a message* m *and outputs a new state, a ciphertext and a message index* $(\pi'_U, c, idx) \xleftarrow{\$} \textsc{Send}(\pi_U, LTS_U, m)$.
- $\textsc{Recv}$ *takes as input the state* π_U *and long-term secrets* LTS_U *of the receiver as well as a ciphertext* c *and outputs a new state, a plaintext and an index* $(\pi'_U, m, idx) \xleftarrow{\$} \textsc{Recv}(\pi_U, LTS_U, c)$.

  $\textsc{Recv}$ *may return an error* $(\perp)$ *instead of the plaintext which signals the ciphertext has not been accepted. Moreover, it may raise a* **Close** *exception which signifies the user closes the connection.*
- $\textsc{StartAuth}$ *takes as input a state* π_U *and outputs a new state* $\pi'_U \xleftarrow{\$} \textsc{StartAuth}(\pi_U)$.
- $\textsc{DetectOOB}$ *takes as input two states* π_U *and* π_V *and outputs a bit* $d \xleftarrow{\$} \textsc{DetectOOB}(\pi_U, \pi_V)$.

Moreover, the session state contains an `auth` flag which is initially set to **None**. $\textsc{StartAuth}$ is a special procedure which may set this flag to a value other than **None** to indicate that the party is currently performing an authentication step. An authentication step is *passed* once the `auth` flags of both communication parties are back to **None**.

In a session between two parties, an *epoch* is defined as a flow of messages sent by one party without receiving a reply from their peer. Epochs are numbered: even epochs correspond to messages sent by the initiator of the conversation and odd epochs to messages sent by the responder. Within an epoch, messages are again numbered consecutively.

Before defining correctness for messaging schemes, the following definition introduces the notion of *matching states*. On a high-level, states are considered to be matching if either state would decrypt correctly a ciphertext sent by their matching partner.

Definition 5.2 (Matching states)**.** *Let* `MS` *be a messaging scheme and* A*,* B *two users created with* $(LTI_U, LTS_U, MTI_U, MTS_U) \xleftarrow{\$} \textsc{Register}()$ *for* $U \in \{A, B\}$*. Let* π_A *(resp.* π_B*) be a state of* A *(resp.* B*) during a protocol execution.*

π_A *and* π_B *are* matching *states if, for all message* m*, ciphertext* c *and index* idx*,* $(\pi'_A, c, idx) \xleftarrow{\$} \textsc{Send}(\pi_A, LTS_A, m)$ *implies that* B *would output* $(\pi'_B, m', idx') \xleftarrow{\$} \textsc{Recv}(\pi_B, LTS_B, c)$ *with* $m' = m$ *and* $idx' = idx$*.*

This enables the definition of correct messaging schemes.

Definition 5.3 (Correct messaging scheme)**.** *Let* `MS` *be a messaging scheme. Let also A and B be two users created with the* Register *algorithm.* `MS` *is* correct *if it follows the following properties:*

1. *The index of a message created by* Send *corresponds to its epoch and number within the epoch.*
2. *The index idx returned by* Recv *is efficiently computable from the ciphertext.*
3. Recv *returns plaintext $\perp$ if the ciphertext corresponds to an index which has already been decrypted.*
4. *If* Recv *returns plaintext $\perp$, then the state remains unchanged.*
5. *If states π_A and π_B are matching, A uses* Send *to create $(\pi'_A, c, idx) \leftarrow \textsc{Send}(\pi_A, LTS_A, m)$ from a plaintext m, and B inputs this ciphertext to create $(\pi'_B, m, idx) \leftarrow \textsc{Recv}(\pi_B, LTS_B, c)$ (the output message and index are equal because of the matching property), then π'_A and π'_B are still matching.*
6. *Given two matching states, if one of them has $\pi.auth \neq \textbf{None}$, then there exists a finite number of calls to* Send *and* Recv *such that both states get back to $\pi.auth = \textbf{None}$.*

Property 1 ensures numbering corresponds to the notion of epochs. Property 2 makes immediate decryption [349] possible. Moreover, property 3 makes sure messages decrypted correspond to different indexes and ciphertexts cannot be replayed. Property 4 ensures bad ciphertexts do not break the scheme. Property 5 ensures the soundness property propagates to all messages in the communication. Note that this does not fully capture immediate decryption. The reader may refer to [349] to get a more precise definition of the soundness property for the Signal protocol which is compatible with the remainder of the paper. Property 6 rules out schemes where an authentication step can never end.

5.2.2 Security Game

This section presents the formal security game capturing continuous authentication, represented in Definition 5.4.

The security game creates two users, Alice (A) and Bob (B), and lets the adversary interact with them using oracles to simulate a communication. As the final objective is to detect long-term secret compromise, long-term secrets are distributed honestly to parties. In contrast, medium-term secrets are generated by parties, but delivered on the communication channel, allowing the adversary to tamper with them. The adversary is active on the network and can corrupt devices, leaking their current state. The adversary can also compromise long-term secrets, which sets a flag *compromised* in the game, maintaining adversary knowledge within the game. When the adversary terminates, an out-of-band detection step (`detectTrial`, defined in algorithm 5.1) is triggered.

The adversary breaks continuous authentication (also referred to as the adversary 'winning') by (1) fooling the out-of-band detection DetectOOB

```
1  procedure createState-A(PI):
2      assert(¬π_A)
3      π_A ←$ InitState(LTS_A, MTS_A, LTI_B, PI)
```

```
1  procedure transmit-A(m):
2      assert(π_A)
3      (π_A, c, idx) ←$ Send(π_A, LTS_A, m)
4      if c ∈ inj_B ∪ authinj ∪ passinj :
5          inj_B ← inj_B \ {c}
6          authinj ← authinj \ {c}
7          passinj ← passinj \ {c}
8      trans_B.append(c)
9      return c
```

```
1  procedure corruptState-A():
2      if π_A:
3          return π_A
4      return MTS_A
```

```
1  procedure auth-A():
2      assert(lastrecv_A > 0 ∧ ¬π_A.auth ∧ ¬π_B.auth)
3      π_A ←$ StartAuth(π_A)
4      authinj, authidx ← inj_A, lastrecv_A
```

```
1  procedure corruptLTS-A():
2      compromised ← True
3      return LTS_A
```

```
1  procedure deliver-B(c):
2      assert(π_B)
3      try:
4          (π'_B, m, idx) ←$ Recv(π_B, LTS_B, c)
5          if m ≠ ⊥ ∧ idx > lastrecv_B:
6              lastrecv_B ← idx
7          if ¬π_B.auth ∧ π'_B.auth :
8              authinj ← authinj ∪ {c ∈ inj_B | c.idx ≤ authidx}
9          CheckAuthStepPassed()
10         π_B ← π'_B
11         if c ∉ trans_B ∧ m ≠ ⊥ :
12             inj_B[idx] ← c
13         return m
14     except Close:
15         closed ← True
```

```
1  procedure CheckAuthStepPassed():
2      if authinj ≠ ∅ ∧ ¬π_A.auth ∧ ¬π_B.auth:
3          passinj ← passinj ∪ authinj
4          inj_A ← inj_A \ authinj
5          inj_B ← inj_B \ authinj
6          authinj ← ∅
```

Figure 5.1: Oracles available to the adversary in the continuous authentication security game (cf. definition 5.4). The `MS.` prefixes for functions of the messaging scheme are omitted. The `CheckAuthStepPassed` function checks if the adversary succeeded in injecting a message which passed an authentication step. Each oracle has a counterpart whose implementation is similar by swapping A and B in the implementation.

to think it compromised the long-term keys when it actually did not, or (2) injecting a message and successfully passing an authentication step ($passinj \neq \emptyset$), without being detected.

This model conservatively grants the adversary more power than may seem reasonable in practice. In particular, the adversary can choose when in-band and out-of-band detection steps happen (by calling StartAuth and terminating). In practice, in-band detection steps may follow a predefined schedule, and out-of-band detection steps are performed at the discretion of users.

Oracles

The adversary has access to the following oracles, depicted in figure 5.1, with corresponding counterpart oracles for Bob:

- createState-A creates the initial state of Alice given some public information of Bob provided by the adversary.
- transmit-A takes a plaintext as input and simulates Alice sending it.
- deliver-A takes a ciphertext as input and simulates Alice receiving it.
- corruptState-A returns the current state of Alice.
- auth-A makes Alice request authentication.
- corruptLTS-A leaks Alice's long-term secret to the adversary.

The transmit-A/B oracles are a wrapper around SEND, which records ciphertexts created legitimately by users. Similarly, the deliver-A/B oracles are a wrapper around RECV, which add injected ciphertexts to the *inj* sets, described with the security game.

When Alice starts an authentication step (which happens when she receives an authentication message or when auth-A is called), *authinj* is filled with all messages that were injected to her. Authenticated messages will be those she has received in the last epoch and before.

Whenever the adversary calls deliver-A/B, the `CheckAuthStepPassed` function is called to check if the adversary has successfully injected a message and passed an authentication step. In that case, deliver-A/B adds the injected messages that were successfully authenticated in *passinj*, and removes them from *authinj* and *inj* sets.

The *win* flag can only be set to **True** in the `detectTrial` function. This happens either if parties output **True** in the out-of-band detection step, but the long-term secret was not compromised (*i.e.*, users produced a false positive), or if they output **False**, but communication was successfully tampered with and the authentication step passed (*i.e.*, the attacker was successful at avoiding detection).

Security Game

The security game itself and resulting security notions are defined as follows.

Definition 5.4 (Continuous authentication)**.** *Let $\mathcal{A}$ be a probabilistic polynomial-time adversary against a messaging scheme* `MS`*. It has access to oracles depicted in figure 5.1, abbreviated as oracles$_{\texttt{MS}}$. The security game is given in Algorithm 5.1.*

The advantage *of adversary $\mathcal{A}$ against the messaging scheme* `MS` *in the detection game is:*

$$\texttt{Adv}(\mathcal{A}) = \Pr\left[\texttt{Detection-Game}(\mathcal{A}, \texttt{MS}) = 1)\right].$$

The messaging scheme `MS` *is said to provide* continuous authentication *if, for all efficient adversaries $\mathcal{A}$,* $\texttt{Adv}(\mathcal{A})$ *is small.*

Algorithm 5.1: Security game capturing continuous authentication.

game `Detection-Game(`$\mathcal{A}$`, MS):`
  $(LTI_A, LTS_A, MTI_A, MTS_A) \xleftarrow{\$}$ *MS*.REGISTER()
  $(LTI_B, LTS_B, MTI_B, MTS_B) \xleftarrow{\$}$ *MS*.REGISTER()
  $\pi_A, \pi_B \leftarrow$ **None**, **None**
  $win \leftarrow$ **False**, $closed \leftarrow$ **False**, $compromised \leftarrow$ **False**
  $trans_A, trans_B \leftarrow \emptyset, \emptyset$
  $inj_A, inj_B, authinj, passinj \leftarrow \emptyset, \emptyset, \emptyset, \emptyset$
  $\mathcal{A}^{oracles_{MS}}(LTI_A, LTI_B, MTI_A, MTI_B)$
  `detectTrial()`
  return $win \wedge \neg closed$

procedure `detectTrial():`
  `assert(`$\pi_A \wedge \pi_B \wedge \neg\pi_A.auth \wedge \neg\pi_B.auth$`)`
  $d \leftarrow$ DETECTOOB(π_A, π_B)
  if $d \wedge \neg compromised$**:**
    $win \leftarrow$ **True**
  elif $\neg d \wedge passinj \neq \emptyset$**:**
    $win \leftarrow$ **True**

The game defines internal variables to keep track of the communication and of the adversary's actions:

- (LTI_U, LTS_U) is the long-term information and secret of user U and π_U its state.
- *win* is a flag representing if the adversary has met the winning conditions.
- *closed* is a flag representing the state of the connection (if it is closed or not).
- *compromised* records if the adversary has compromised either of the parties' long-term secrets.
- $trans_U$ is a set holding ciphertexts created by a legitimate user U.
- inj_U is a set containing messages injected to user U (which user U accepted) that are yet to be authenticated. *authinj* is a set used during authentication steps which holds all injected messages currently being authenticated. *passinj* is a set containing all injected messages that successfully passed authentication.

5.3 The Signal Protocol

Signal[9,10] is an asynchronous messaging protocol using an unreliable channel which aims to provide end-to-end encryption with additional security properties such as forward secrecy and post-compromise security.

[9] *Signal: Technical information.* [Online]. Available: https://signal.org/docs/.
[10] T. Perrin and M. Marlinspike, *The Double Ratchet algorithm*, 2016. [Online]. Available: https://whispersystems.org/docs/specifications/doubleratchet/.

On a high-level, the protocol consists of three phases:

1. A *registration phase* happening when users join the platform. Users then generate key material and upload their public key material.
2. A *session establishment* happening when one user wants to contact another. This phase uses long-term key material to establish a shared session secret.
3. The *data exchange phase* is the remainder of the communication. We emphasize that in this phase long-term secrets are not used.

Signal involves several cryptographic components:

- an elliptic curve group to perform Diffie–Hellman key exchange operations,
- a key derivation function (KDF) do derive new key material from established secrets,
- a signature scheme for signing uploaded key material, and
- an authenticated encryption (AEAD) scheme for message encryption.

Registration

Upon registration, a user U creates a *long-term key pair* (ik^U, ipk^U). It then generates a *pre-key pair* $(prek^U, prepk^U)$ as well as several *ephemeral key pairs* $((ek_i^U, epk_i^U))_i$. The public keys are all uploaded to the Signal server. This matches the Register procedure from Definition 5.1, with the long-term key-pair corresponding to long-term information and secrets (LTS_U, LTI_U) and the other keys to the medium-term information and secrets (MTS_U, MTI_U).

Session establishment

In order for a user Alice (A) to initiate a session with another user Bob (B), Alice retrieves Bob's public information from the server. She then uses Signal's Diffie–Hellman-based initial key agreement protocol X3DH to establish an initial shared secret, the so-called *root key* sk_0, by combining her private long-term key and an ephemeral private key with Bob's public keys. This can be modelled in terms of Definition 5.1 as $\pi_A \leftarrow \textsc{InitState}(LTS_A, MTS_A, LTI_B, MTI_B)$.

Alice will transmit to Bob in the associated data AD of her first message the ephemeral public keys she has used in X3DH. Thus, upon receiving this initial message, Bob will also be able to derive the exact same shared secret, which can be modelled as $\pi_B \leftarrow \textsc{InitState}(LTS_B, MTS_B, LTI_A, AD)$.

Data exchange

Recall that an epoch i is a sequence of messages sent by a user without having received any reply from their peer. When a user (for instance Alice) wants to send the first message in an epoch, she first chooses a new *ratchet key pair* (rk_i^A, rpk_i^A). Using her new ratchet key-pair and Bob's last

ratchet key pair[11], she derives a shared secret as the Diffie–Hellman secret $DH_i = \texttt{DH}(rk_i^A,\ rpk_{i-1}^B)$. From this she can derive a new *sending chain key*: $(sk_{i+1}, ck_{i,0}) \leftarrow \texttt{KDF}(sk_i, DH_i)$.

For all messages in the epoch, Alice creates message keys using the KDF with the chain key and a constant input: $(ck_{i,j+1}, mk_{i,j}) = \texttt{KDF}(ck_{i,j}, 1)$. Those message keys are used to encrypt messages using the AEAD scheme. The public ratchet key for this epoch is included in the associated data of all messages, so Bob can derive the same message keys to decrypt messages. Because messages may be lost or reordered, the message epoch and index are included in the associated data, so Bob can reconstruct the correct key to decrypt the message.

All those operations can be encapsulated in the SEND procedure, with root key, chain keys and message keys being held in the local state. Keys are deleted from the local state once they are no longer needed.

When Bob receives one message from Alice, knowing the root key sk_i shared with Alice and receiving in the associated data her new ratchet key, he can derive the same initial chain key $ck_{i,0}$, and from it the message keys to decrypt messages. Receiving is modelled through the RECV algorithm.

This shows succinctly that Signal fits the definition of a messaging scheme given in Section 5.2. We will see next that it does not achieve continuous authentication.

5.3.1 Signal Does Not Provide Continuous Authentication

For the original Signal protocol, STARTAUTH is a procedure that does nothing as no authentication steps are implemented. We will now show that no choice for implementing the DETECTOOB algorithm would yield a protocol which ensures continuous authentication. Formally, the following adversary against Signal succeeds in the security game for continuous authentication from Definition 5.4 with probability 1, no matter how DETECTOOB is defined.

Proposition 5.1 (Signal insecurity)**.** *Let $\mathcal{A}$ be the adversary from Algorithm 5.2. If the messaging scheme* MS *of Definition 5.4 is the Signal protocol, then adversary $\mathcal{A}$ wins the game with probability 1.*

Proof. The proof is quite straightforward. First the adversary opens a legitimate session between both users and uses transmit-A and deliver-B to send a message from Alice to Bob. Then, using transmit-B, the adversary makes Bob create message 0 of epoch 1 for plaintext m_1.

However, instead of delivering the created ciphertext, the adversary drops it and forges her own ciphertext instead. To do so, she corrupts the local state of Alice using corruptState-A and performs the same computations as what Bob would have done to transmit m_1' instead of m_1. We refer the reader to the Signal specification[12] to see that Line 12 leads to a correct ciphertext for m_1'. c_1 and c_1' correspond to different plaintexts thus $c_1 \neq c_1'$, which means $(c_1', AD_1) \notin trans_A$.

When the adversary calls auth-A, as (c_1', AD_1) has been successfully injected, $(c_1', AD_1) \in inj_A \subset authinj$. Then the adversary transmits one message

[11] For the very first epoch of the conversation, Alice uses Bob's pre-key pair as his last ratchet key-pair.

[12] T. Perrin and M. Marlinspike, *The Double Ratchet algorithm*, 2016. [Online]. Available: https://whispersystems.org/docs/specifications/doubleratchet/.

Algorithm 5.2: A description of a successful continuous authentication adversary against the Signal protocol.

adversary $\mathcal{A}(LTI_A, LTI_B, MTI_A, MTI_B)$**:**
  $m_0, m_1 \neq m_1', m_2 \in_R \{0,1\}^*$
  createState-A(MTI_B)
  $c_0 \xleftarrow{\$}$ transmit-A(m_0)
  createState-B($c_0.AD$)
  deliver-B(c_0)
  $c_1 \xleftarrow{\$}$ transmit-B(m_1)
  $(c_1, AD_1) \leftarrow c_1$
  $\pi_A \leftarrow$ corruptState-A()
  $(\_, ck) \leftarrow \texttt{KDF}(\pi_A.sk, \texttt{DH}(AD_1.rpk^B,\ \pi_A.rk^A))$
  $(\_, mk) \leftarrow \texttt{KDF}(ck, 1)$
  $c_1' \xleftarrow{\$} \texttt{Enc}(mk, m_1', AD_1)$
  deliver-A((c_1', AD_1))
  auth-A()
  $c_2 \xleftarrow{\$}$ transmit-A(m_2)
  deliver-B(c_2)

from Alice to Bob. When Bob receives the message, `CheckAuthStepPassed` is called. As StartAuth does not change the state, both conditions $authinj \neq \emptyset$ and $\neg\pi_A.auth \wedge \neg\pi_B.auth$ are satisfied, and therefore $(c_1', AD_1) \in passinj$.

When `detectTrial` is called, the adversary wins whatever the implementation of DetectOOB is. Indeed, if DetectOOB outputs $d = 1$, as $\mathcal{A}$ never compromised a long-term secret, *compromised* is **False** and therefore the adversary wins. If $d = 0$, as *passinj* is not empty the adversary also wins the game.

Thus, the adversary wins with probability 1. □

The above proof solely uses the fact that the adversary can forge valid messages by only knowing the session secrets of a party. The attack hence extends to any messaging scheme that, like Signal, only relies on session secrets and randomness to create new session secrets.

5.4 Introducing Authentication Steps

This section presents the proposed *Authentication Steps* protocol that generically extends messaging schemes to achieve continuous authentication.

The extension introduces authentication steps, that may happen regularly at defined epochs in a session or could be user-triggered. These authentication steps leverage long-term secrets. In Signal, the long-term secret of a user consists of their private identity key, a Diffie–Hellman exponent. The Authentication Steps protocol introduces a new type of long-term secret, which is a signing key $sigk^U$ [13].

The objectives of an authentication step are twofold:

[13] In practice, Signal already re-uses the identity key to sign a user's medium-term public key using the XEdDSA [357] signature scheme; therefore, an implementation may similarly

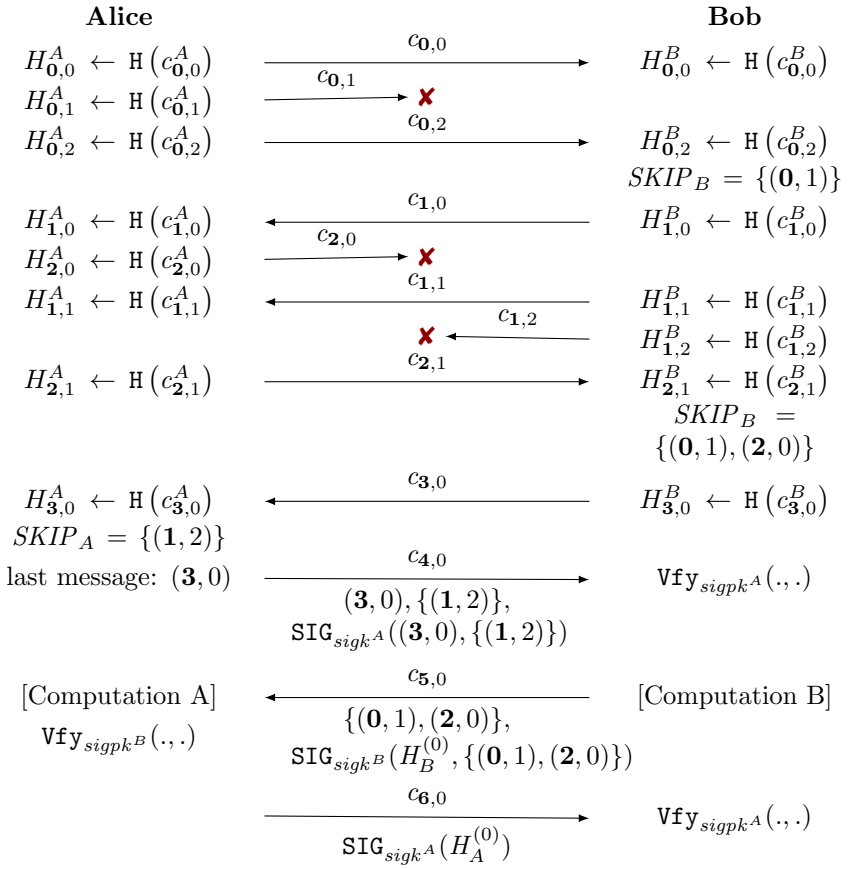


Figure 5.2: An example execution of an authentication step. The actual authentication step is performed during epochs **4** to **6**, with authenticated messages from epoch **0** (epoch numbers in **boldface**). Additional data sent for those messages is included below arrows. For epochs **4** to **6**, $H_{\mathbf{i},j}$ hashes are still computed by both parties, but they are omitted in this figure as they concern the next authentication step. [Computation U] for $U \in \{A, B\}$ corresponds to the computation $H_U^{(0)} \leftarrow 0||\mathtt{H}\left(\varepsilon||H^U_{\mathbf{0},0}||H^U_{\mathbf{0},2}||H^U_{\mathbf{1},0}||H^U_{\mathbf{1},1}||H^U_{\mathbf{2},1}||H^U_{\mathbf{3},0}\right)$.

1. to convince parties that they are communicating with the holder of their peer's private key, and

2. to detect tampering with messages since the last authentication step.

To that end, each party sends on the in-band channel their own view of the communication since the last authentication step. These messages are included alongside regular messages exchanged between users. This allows the authentication steps to seamlessly be integrated on top of the existing Signal protocol.

reuse that identity key as the signing long-term key for the authentication steps extension. In practice, this means only maintaining a single long-term secret for both Signal and the Authentication Steps protocol.

In order to maintain forward secrecy, the additional information is derived from the (public) ciphertexts sent. To save space, intermediate computations compress those ciphertexts as they are sent or received. Those intermediate computations and an authentication step are illustrated in figure 5.2.

5.4.1 Recording Ciphertexts

In order to perform authentication steps, parties need to store the transcript of ciphertexts sent and received. The order in which messages are received is not relevant as reordering may be caused by the unreliable channel.

Instead of storing ciphertexts as sent, each party computes digests of those ciphertexts using a hash function and stores those in a dictionary. Concretely, for every sent or received message, each user U computes and stores $H^U_{i,j} = \mathtt{H}\left(c^U_{i,j}\right)$, where $\mathtt{H}$ is a cryptographic hash function, and $c^U_{i,j}$ is the ciphertext corresponding to message j sent or received in epoch i by user U.

5.4.2 Authentication Steps

The stored (and hashed) ciphertexts are then used in the actual authentication step. An authentication step is a 3-pass message exchange, and therefore requires three epochs to complete. In the following, an authentication step is described with Alice sending the first authentication message. Figure 5.2 illustrates an authentication step, performed in epochs 4 to 6.

The authentication step information is included in every message of the epoch. That way, the peer receives the authentication information at least once, as if they do not receive it, the epoch number will not increase. If authentication information is missing from a message where it should have been included, then the receiving party should dismiss the message.

In the first epoch, Alice sends the following additional authentication information (encrypted along with actual plaintext):

- the indexes of messages that she should have received from Bob, but did not, denoted $SKIP_A$,
- the index of the most recent message she has received from Bob, denoted $authidx$, and
- a signature $\mathtt{SIG}_{sigk^A}((authidx, SKIP_A))$ over both values.

This allows Bob to know which messages Alice wants to authenticate. When Bob receives this message, he first verifies the signature, using Alice's signing public key. In case of success, Bob computes the following hash:

$$H_B^{(n_B)} = n_B || \mathtt{H}\left(H_B^{(n_B-1)} || \; ||_{(i,j)\in I_B^{(n_B)}} H^B_{i,j}\right),$$

where n_B is the number of authentication steps Bob has completed, and $H_B^{(n_B-1)}$ the hash computed in the previous authentication step (with the convention $H_B^{(-1)} = \varepsilon$ the empty string). The concatenation happens in lexicographic order over $I_B^{(n_B)}$, the set of all messages sent and received by Bob since last authentication step and until message $authidx$, and excluding messages with an index contained in $SKIP_A$.

In the second epoch (with Bob sending messages), Bob sends the following information (along with the regular message plaintexts):

- the indexes of messages that he should have received from Alice, denoted $SKIP_B$, and
- a signature $\mathtt{SIG}_{sigk^B}(H_B^{(n_B)}, SKIP_B)$ over the hash computed and the indexes of missed messages.

When Alice receives Bob's message, she extracts the list $SKIP_B$ and computes the following hash:

$$H_A^{(n_A)} = n_A||\mathtt{H}\left(H_A^{(n_A-1)}|| \; ||_{(i,j)\in I_A^{(n_A)}} H_{i,j}^A\right),$$

where, like for Bob, n_A is her number of completed authentication steps and $H_A^{(n_A-1)}$ is the previous hash (or $H_A^{(-1)} = \varepsilon$). Alice then checks the signature received from Bob, using Bob's public signing key, on data $(H_A^{(n_A)}, SKIP_B)$.

In the third epoch, Alice sends a signature $\mathtt{SIG}_{sigk^A}(H_A^{(n_A)})$ over her hashed collection of seen messages. When Bob receives it, he verifies the signature's validity on $H_B^{(n_B)}$, using Alice's signing public key.

If at some point a signature verification fails, the verifier closes the connection. Otherwise, Alice and Bob have *passed the authentication step*.

Deniable Signing.

Any unforgeable signature scheme can be used in the authentication step. In particular, to maintain Signal's deniability of the initial key agreement (cf. [358]), signatures can be generated using designated-verifier or 2-user ring signatures [359], [360], similarly to their deployment in recent proposals for Signal-like deniable key exchanges [361]–[364].

5.4.3 Detecting Compromised Long-term Secrets

In this section, it is assumed that Alice and Bob have passed at least one authentication step. At each authentication step, parties derive a hash $H_A^{(n_A)}$ or $H_B^{(n_B)}$. Authentication steps succeed if the signatures over those hashes match.

On a high-level, users execute the following protocol: Using the out-of-band channel, parties compare the last hash they have computed (which they store in their state until the next authentication step) as well as the number of authentication steps performed. If the hash values and authentication steps counters match, the users output **False**, indicating that they do not detect long-term key compromise, otherwise they output **True**.

If no adversary tampers with the communication, then exchanged hashes would match. Conversely, hashes not matching means that an adversary is present. Moreover, as at least one authentication step has been successful, the adversary must have been able to forge a signature to avoid in-band detection, which indicates they know at least one long-term secret. These two properties of the Authentication Steps protocol are formally proven in section 5.5.

5.4.4 Protocol Soundness

The described Authentication Steps protocol is correct, *i.e.*, it matches Definition 5.3, as shown by the following proposition.

Proposition 5.2 (Protocol soundness)**.** *If no adversary tampers with the communication, i.e., modifies or injects a ciphertext, then parties pass authentication steps.*

The idea of the proof is to show that in the absence of an adversary, users always receive authentication information when performing authentication steps. This leads them to compute the same sets $I_A^{(n_A)} = I_B^{(n_B)}$ and therefore they agree on the hash computations.

More precisely, properties 1 to 4 of the correctness definition are easy to prove and come directly from the properties of the underlying protocol.

To prove the remaining properties, first Lemma 5.3 proves that parties always receive authentication information if they continue to communicate. Then, Lemma 5.4 shows that parties agree on the same set of messages to authenticate ($I_A^{(n_A)} = I_B^{(n_B)}$, with the notations of Section 5.4). Finally, the proof of 5.2 shows that the hashes computed are the same and that each signature verification succeeds.

Lemma 5.3. *If no adversary tampers with the communication, then parties either stay in the same epoch or receive the authentication information from their peer.*

Proof. The authentication information is sent along all messages in an epoch. However, to advance to the next epoch, a party needs to receive at least one message from their peer's epoch, which includes the authentication information. As ciphertexts are not modified or injected in this context, the information received has been honestly generated by their communicating partner. □

Lemma 5.4. *If no adversary tampers with the communication, then computed sets $I_A^{(n_A)}$ and $I_B^{(n_B)}$ by each party are equal: i.e., $I_A^{(n_A)} = I_B^{(n_B)}$.*

Proof. We begin the proof by proving by induction that for every authentication step, Alice and Bob agree on the lower and upper bounds on the chosen set of messages, which we denote the *period* of messages.

For the first authentication step, Alice and Bob must agree on the lower bound as it is message $(0, 0)$. Then the initiator (for instance Alice) chooses the last message she wants to authenticate (one of Bob's messages, since the authentication step starts at the beginning of an epoch) and sends this index to Bob. Bob receives this index correctly thanks to Lemma 5.3. So for the first authentication step, Alice and Bob agree on the period of messages to authenticate.

Let's assume Alice and Bob agreed on the period of messages to authenticate for the last authentication step. Let's call (i, j) the index of the last message authenticated. As authentication steps last for three epochs, epoch i is finished for both users. They thus know which message follows (i, j) (either $(i, j + 1)$ or $(i + 1, 0)$), and they agree on this message as the first one to authenticate.

Once again, the initiator chooses the last message to authenticate and transmits this choice to the peer, so both agree on the upper bound.

By induction, for every authentication step, Alice and Bob agree on the same period of messages to authenticate.

Then for a given authentication step, $I_U^{(n_U)}$ is the set of message indexes that U has sent or received in this period, except for messages that their peer did not receive (whose indices are sent in $SKIP_V$).

In both cases, $I_A^{(n_A)}$ and $I_B^{(n_B)}$ are the union of the set of messages received by Alice and the set of messages received by Bob. This proves that $I_A^{(n_A)} = I_B^{(n_B)}$. □

With those lemmas, the soundness of the Authentication Steps protocol can be proven.

Proof of Proposition 5.2. By definition, authentication steps pass if the connection does not close. Given the implementation, **Close** exceptions occur only when a signature verification fails.

As no adversary actively injects or modifies messages, the first signature verification performed by Bob always succeeds as the signed data is sent alongside the signature. Similarly, if no active adversary is present, then ciphertexts received will be the same as ciphertexts sent: $\forall (i,j) \in \mathcal{R},\ c_{i,j}^A = c_{i,j}^B$ where $\mathcal{R}$ is the set of indices of received messages.

The hash function is deterministic, therefore:

$$\forall (i,j) \in \mathcal{R},\ H_{i,j}^A = \mathtt{H}\left(c_{i,j}^A\right) = \mathtt{H}\left(c_{i,j}^B\right) = H_{i,j}^B.$$

Let's assume that at the beginning of an authentication step, $n_A = n_B$ and $H_A^{(n_A)} = H_A^{(n_B)}$. Recall the next $I_U^{(n_U+1)}$ for user U is computed as: $I_U^{(n_U+1)} \leftarrow (n_U+1)||\ \mathtt{H}\left(I_U^{(n_U)}||\ ||_{(i,j)\in I_U^{(n_U+1)}} H_{i,j}^U\right)$. Following Lemma 5.4, Alice and Bob agree on sets $I_A^{(n_A+1)} = I_B^{(n_B+1)} = I \subset \mathcal{R}$. Moreover, ciphertexts are ordered correctly as the identification information is included in the ciphertext and was stored accordingly. We have seen above that for all $(i,j) \in I$, $H_{i,j}^A = H_{i,j}^B$. Therefore, after the authentication step, we still have $I_A^{(n_A+1)} = I_A^{(n_A+1)}$. As at the beginning of the communication, $n_A = n_B = 0$ and $I_A^{(-1)} = I_B^{(-1)} = \varepsilon$, by induction we conclude that for every authentication step, $I_A^{(n_A)} = I_B^{(n_B)}$.

In the second epoch, Bob signs $(I_B^{(n_B)},\ SKIP_B)$ and transmits $SKIP_B$. Because we have shown $I_A^{(n_A)} = I_B^{(n_B)}$, Alice succeeds in verifying the signature on the data $(I_A^{(n_A)},\ SKIP_B)$ she computes. In the third epoch, considering $I_A^{(n_A)} = I_B^{(n_B)}$, Bob also succeeds in verifying the signature. As all signature verifications succeed, Alice and Bob pass the authentication step. □

5.5 Security of the Authentication Steps Protocol

This section formally establishes the continuous authentication security (as per definition 5.4) of the Authentication Steps protocol extension presented in section 5.4.

Theorem 5.5. *Assuming a collision resistant hash function* H *and an existentially unforgeable signature scheme $\mathcal{S}$, the Authentication Steps protocol presented in section 5.4 provides continuous authentication as per definition 5.4.*

Formally, the advantage of any adversary $\mathcal{A}$ in the detection game against the Authentication Steps protocol is bounded as follows:

$$\mathtt{Adv}(\mathcal{A}) \leq \mathtt{Adv}^{coll}_{\mathcal{B}_1}(H) + 2 \cdot \mathtt{Adv}^{EUF\text{-}CMA}_{\mathcal{B}_2}(\mathcal{S}),$$

for reduction adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ given in the proof.

Proof Overview

The adversary wins the detection game in two cases:

1. users decide one of their long-term secrets is compromised when that is not the case; and
2. the adversary manages to inject a message and remains undetected.

In case 1, the adversary never corrupts the long-term secret, yet the parties decide that their long-term secret is compromised. Thus, the hashes that Alice and Bob exchanged at the end of the game must be different, but both Alice and Bob verified signatures hashes in the last authentication step. It follows then that either Alice or Bob received a signature that was not produced by their peer, and that the adversary must have successfully forged a message under one of their (non-compromised) signing key. This would violate the EUF-CMA security of the signature scheme, leading to the $2 \cdot \mathtt{Adv}^{EUF\text{-}CMA}_{\mathcal{B}_2}(\mathcal{S})$ term in the theorem bound.

In case 2, the adversary must have injected a message between Alice and Bob, but when Alice and Bob exchanged their hashes at the end of the game, the hash outputs matched. It follows that between Alice's or Bob's computations, there must be a hash collision, leading to the $\mathtt{Adv}^{coll}_{\mathcal{B}_1}(H)$ term in the theorem bound.

5.5.1 Formal Proof

This section proves theorem 5.5, which states that the Authentication Steps protocol is secure under the assumption that the underlying cryptographic primitives are secure, namely the hash function and the signature scheme.

Let $\mathtt{Adv}^{coll}_{\mathcal{A}}(H)$ be the advantage of an adversary trying to find a collision for a hash function $\mathtt{H}$, and $\mathtt{Adv}^{EUF\text{-}CMA}_{\mathcal{A}}(\mathcal{S})$ be the advantage of an adversary in the EUF-CMA (Existential UnForgeability in the Chosen Message Attack setting) game against a signature scheme $\mathcal{S}$.

False Positives and False Negatives.

Before proving theorem 5.5, several useful definitions are introduced.

From the specification, no authentication steps can overlap. Therefore, users will reject messages that start a new authentication step if they are currently performing one. Thus, authentication steps can be numbered from the point of view of a user U, from 1 to n_U.

The proof of theorem 5.5 splits the winning condition into two events, referred to as the false negative case and the false positive case. Propositions 5.7 and 5.8 each provide an upper bound on their respective probabilities.

Definition 5.5. *Given an adversary $\mathcal{A}$ playing the security game of definition 5.4, the following events are defined:*

- W *is the event that the $\mathcal{A}$ wins the game,*
- FP *is the event that at the end of the game, $\neg closed \wedge \neg compromise \wedge d$ is true,*
- FN *is the event that at the end of the game, $\neg closed \wedge passinj \neq \emptyset \wedge \neg d$ is true.*

FP *and* FN *respectively stand for false positive and false negative.*

Upper Bound for False Negatives

This section gives an upper bound on the probability Pr [FN] that an adversary produces a false negative in the game. First, lemma 5.6 is introduced, which is used to prove proposition 5.7.

Lemma 5.6. *Let $\mathcal{A}$ be an adversary playing the security game of definition 5.4 against the Authentication Steps protocol from section 5.4.*

If $passinj \neq \emptyset$ at the end of the game, it means that there exists some user $U \in \{A, B\}$, an authentication step j for U, and a message index $i \in I_U^{(j)}$, such that $c_i^A \neq c_i^B$ (where one of the ciphertext could be $\perp$ if the corresponding user has sent no ciphertext for index i).

Proof. If $passinj \neq \emptyset$ at the end of the game, then there exists an index $i \in passinj$.

$passinj$ is only filled in the `CheckAuthStepPassed` function (see figure 5.1), and only if $authinj$ is not empty. $authinj$ is filled only at two places: at line 4 of the auth-A/B oracle, or at line 8 of deliver-A/B (see figure 5.1). For both cases, this happens when a user U enters an authentication step (*e.g.*, authentication step j), and message i comes from inj_U.

Message i has already been received, because it is in inj_U when added to $authinj$. Therefore, when the authentication step begins, message i is not a skipped message. Moreover, $i \leq auth.authidx$ given the implementation of StartAuth.

By definition, $\pi_U.lastauth$ contains the index of the last message authenticated before starting the authentication step. As $authinj$ is cleared at the end of every authentication step, having the ciphertext corresponding to index i in $authinj$ means that it was not already authenticated in a previous authentication step. Therefore, $i > \pi_U.lastauth$.

This proves that for authentication step j, $i \in [\pi_U.lastauth, authinfo.authidx]$. As message i is not a skipped message, $i \in I_U^{(j)}$.

Additionally, because $i \in inj_U$, then user U received – and accepted – ciphertext c_i^U as a valid message (when it was added to inj_U). If V is the peer of U, then c_i^V – if it exists – cannot be equal to c_i^U, as otherwise it would have been an honest message. This proves the existence of $i \in I_U^{(j)}$, such that $c_i^A \neq c_i^B$. □

The following proposition gives an upper bound on the probability that the adversary produces a false negative.

In the construction given in Section 5.4, hashes are used to save space for ciphertexts. However, if no hashes were used and transcripts of actual ciphertexts were stored instead, false negatives could never happen.

Proposition 5.7 (False negatives)**.** *Let $\mathcal{A}$ be an adversary in the detection game of definition 5.4 playing against the Authentication Steps protocol presented in section 5.4.*

Then, $\Pr[\text{FN}] \leq \mathtt{Adv}^{coll}_{\mathcal{B}_1}(H)$*, for a reduction adversary $\mathcal{B}_1$ constructed in the proof.*

Proof. Let $\mathcal{A}$ be an adversary producing event FN. From definition 5.5, FN $=$ $\neg closed \wedge passinj \neq \emptyset \wedge \neg d$.

In particular, *passinj* is not empty at the end of the game. According to Lemma 5.6, this implies the existence of an authentication step j_0 for user $V \in \{A, B\}$, and of an index $i \in I_V^{(j_0)}$ such that $c_i^A \neq c_i^B$.

However, d is **False**. Given the computation of d in the DetectOOB procedure, this means that $\pi_A.H_A^{(n_A)} = \pi_B.H_B^{(n_B)}$.

By definition, for any user U, $H_U^{(n_U)} = n_U || H_U^{(n_U-1)}$. Therefore, $\pi_A.H_A^{(n_A)} = \pi_B.H_B^{(n_B)}$ implies in particular that $n_A = n_B$, which means that Alice and Bob have seen the same number of authentication steps.

Moreover, hashes $H_U^{(j)}$ are computed as follows:

$$H_U^{(j)} \leftarrow \mathtt{H}\left(H_U^{(j-1)} \;||\; ||_{k \in \mathtt{sorted}(I_U^{(j)})} \pi_U.H_k^U\right),$$

for any $j \geq 0$, and with $H_U^{(-1)} = \varepsilon$.

For any $j \geq 0$, if $H_A^{(j)} = H_B^{(j)}$, then there are only two possibilities:

1. either $H_A^{(j-1)} \;||\; ||_{k \in \mathtt{sorted}(I_A^{(j)})} \pi_A.H_k^A \neq H_B^{(j-1)} \;||\; ||_{k \in \mathtt{sorted}(I_B^{(j)})} \pi_B.H_k^B$,
2. or they are equal.

If they are different, because $H_A^{(j)} = H_B^{(j)}$, both values are a collision for the hash function. If they are equal, it means that in particular $H_A^{(j-1)} = H_B^{(j-1)}$.

As the equality $H_A^{(j)} = H_B^{(j)}$ is true for the last authentication step, by induction, either there is a hash collision, or for all authentication step j:

$$||_{k \in \mathtt{sorted}(I_A^{(j)})} \pi_A.H_k^A = ||_{k \in \mathtt{sorted}(I_B^{(j)})} \pi_B.H_k^B.$$

This is true in particular for $j = j_0$. As the elements of $\pi_U.H^U$ are hashes of ciphertexts computed on sending and receiving, and that the hash function produces outputs of the same length, there are exactly the same number of hashes in each concatenation.

Moreover, $i \in I_V^{(j_0)}$ so one hash corresponds to the ciphertext with index i. Thus, $\mathtt{H}(c_i^V) = \mathtt{H}(c^W)$ for a the ciphertext c^W in the same position as ciphertext i in the concatenation.

However, $c^W \neq c_i^V$, because by definition of i, $c_i^A \neq c_i^B$. Therefore, c_i^V and c^W are a hash collision.

To conclude, any case leading the adversary to a false negative produces an explicit hash collision, and therefore the reduction $\mathcal{B}_1$ from the detection game to the hash collision game is immediate:

$$\Pr[\text{FN}] \leq \mathtt{Adv}^{coll}_{\mathcal{B}_1}(H).$$

□

Upper Bound for False Positives

This section gives an upper bound on the probability $\Pr[\text{FP}]$ that the adversary produces a false positive in the game.

Proposition 5.8 (False positives)**.** *Let $\mathcal{A}$ be an adversary in the detection game of definition 5.4 playing against the Authentication Steps protocol of section 5.4.*

Then, $\Pr[\text{FP}] \leq 2 \cdot \mathtt{Adv}^{EUF-CMA}_{\mathcal{B}_2}(\mathcal{S})$*, for a reduction adversary $\mathcal{B}_2$ constructed in the proof.*

Proof. Let $\mathcal{A}$ be an adversary producing event FP. Having $\neg$*compromise* means that $\mathcal{A}$ never calls the corruptLTS-A/B oracles. Moreover, $\neg$*closed* means the communication never closes, which means that signature verifications always succeed.

The proof builds an adversary $\mathcal{B}_2$ for the EUF-CMA game against the signature scheme $\mathcal{S}$, as a wrapper around $\mathcal{A}$, which acts as a challenger in the detection game for $\mathcal{A}$.

$\mathcal{B}_2$ creates two users Alice and Bob, but embeds a public key provided by the EUF-CMA challenger into one party's signing key-pair, and uses the signing oracle to generate signatures.

As user U_1 is entirely generated by $\mathcal{B}_2$, the adversary can simulate the oracles concerning U_1, and therefore they are similar to the oracles defined in Figure 5.1.

$\mathcal{B}_2$ keeps track of signature forgeries. Every time $\mathcal{B}_2$ signs a message using the oracle provided by his challenger, $\mathcal{B}_2$ stores it. Moreover, every time a signature on the signing public key pk given by the EUF-CMA game is verified, $\mathcal{B}_2$ checks if the signature was produced by the signing oracle. If that is not the case, but the verification is successful, $\mathcal{B}_2$ stops and outputs the corresponding pair m^*, σ^*.

To simulate user U_0, whose private key is unknown, $\mathcal{B}_2$ uses the original oracles, except for transmit$-U_0$, which is the only oracle using the private signing key of U_0 in the SEND procedure. corruptLTS-A/B oracles are not called by adversary $\mathcal{A}$, and therefore $\mathcal{B}_2$ does not need to simulate those oracles when the event FP happens.

In order to create the signature, $\mathcal{B}_2$ queries its own challenger with message $\pi_{U_0}.auth$, to get the signature using the private key of U_0. Therefore, $\mathcal{B}_2$ is correctly defined and can act as a challenger for $\mathcal{A}$.

Moreover, the following proves that when $\mathcal{A}$ triggers the event FP, then $\mathcal{B}_2$ wins the EUF-CMA game with probability at least $\frac{1}{2}$.

During his last authentication step n, U_1 verified successfully a signature σ on $\pi_{U_1}.auth$ by using U_0's public signing key $sigpk^{U_0}$. $\pi_{U_1}.auth$ contains in particular $n_U = n$ and $H = H_1$ computed by U_1. Because U_0 and U_1 can

number their authentication steps, they will produce at most one signature on an *auth* set having $n_U = n$.

At the end of the game, parties output $d = \textbf{True}$. From the implementation of DetectTOOB, this means that $\pi_A.H_A^{(n_A)} \neq \pi_B.H_B^{(n_B)}$. Given the definition of $\pi_U.H_U^{(n_U)}$, this means that during the last authentication step of each party:

$$\pi_A.n_A||\pi_A.auth.H \neq \pi_B.n_B||\pi_B.auth.H\,.$$

There are two disjoint possibilities:

1. either $\pi_A.n_A = \pi_B.n_B$ but $\pi_A.auth.H \neq \pi_B.auth.H$;
2. or $\pi_A.n_A \neq \pi_B.n_B$;

In case 1, $\pi_A.n_A = \pi_B.n_B = n$ and $\pi_A.auth.H \neq \pi_B.auth.H$. Yet U_1's verification of σ succeeded on the data $\pi_{U_1}.auth$ which contains $n_U = n$ and $H = H_1$. However, as stated above, U_0 can produce and sign at most one set *auth* with $n_U = n$, and this set has $H = \pi_{U_0}.auth.H \neq \pi_{U_1}.auth.H = H_1$. Therefore, $\pi_{U_1}.auth$ was not submitted to the signing oracle, and yet σ verifies over $\pi_{U_1}.auth$, so $\mathcal{B}_2$ can output this forgery.

In case 2, $\pi_A.n_A \neq \pi_B.n_B$. Recall that U_0 and U_1 are chosen uniformly at random at the beginning of the game. Because the signing key-pair and signatures are sampled and created in the same way in the detection game and in the reduction when using the signing oracle, $\mathcal{A}$ cannot distinguish which key-pair is used in the signing game. Therefore, with probability $\frac{1}{2}$, $\pi_{U_0}.n_{U_0} < \pi_{U_1}.n_{U_1}$.

In that case, U_0 cannot have signed a set $\pi_{U_0}.auth$ with $n_{U_0} = \pi_{U_1}.n_{U_1}$ as it has not yet reached the correct number of authentication steps. This once again yields a valid signature forgery.

Therefore, with probability at least $\frac{1}{2}$, if $\mathcal{A}$ triggers FP then $\mathcal{B}_2$ wins the EUF-CMA game. This leads to the upper bound $\Pr[\mathrm{FP}] \leq 2 \cdot \mathtt{Adv}^{EUF\text{-}CMA}_{\mathcal{B}_2}(\mathcal{S})$. □

Security Proof

Proof of Theorem 5.5. Let $\mathcal{A}$ be an adversary in the game of definition 5.4. Because of the implementation of the `detectTrial` function, and because the *win* flag is only set in this function, it is immediate that $\mathrm{W} = \mathrm{FP} \sqcup \mathrm{FN}$, which are the events defined in definition 5.5.

Therefore:

$$\mathtt{Adv}(\mathcal{A}) = \Pr[\mathrm{W}] = \Pr[\mathrm{FP}] + \Pr[\mathrm{FN}]\,.$$

Moreover, proposition 5.8 states that $\Pr[\mathrm{FP}] \leq 2 \cdot \mathtt{Adv}^{EUF\text{-}CMA}_{\mathcal{B}_2}(\mathcal{S})$, and proposition 5.7 states that $\Pr[\mathrm{FN}] \leq \mathtt{Adv}^{coll}_{\mathcal{B}_1}(H)$, which proves theorem 5.5. □

5.6 Implementation and Benchmarks

A prototype Authentication Steps protocol, which integrates seamlessly *on top* of the official Signal Java library, was implemented. The full implementation can be found on GitHub[14], along with build instructions and benchmarking tests.

[14] https://github.com/apoirrier/libsignal-java-authsteps

5.6.1 Space and Computation Overhead

Authentication steps require additional data to be computed and stored, such as the ciphertexts hashes between authentication steps.

The storage and bandwidth overhead is a function of the channel reliability and the average number of messages per authentication step, the latter being the more influential parameter. Indeed, the sender cannot know in advance which messages the peer has received, thus there is no alternative but storing every ciphertext hash individually.

As for computational overhead, computing the ciphertext hashes involves one hash invocation; additionally, at most one signature and one verification operation is performed per epoch. The signature scheme employed by Signal is XEdDSA [357]. Signing and verifying data typically requires the same amount of computation as the Diffie-Hellman key computation happening in asymmetric ratchet steps. Thus, the computational overhead is at most the same magnitude as the original computations in Signal.

5.6.2 Benchmarking the Space Overhead

In order to give an estimate on the space overhead induced by the Authentication Steps extension, simulations of communication sessions were performed, to evaluate ciphertext and state sizes. The message inputs for the simulations are taken from the National University of Singapore SMS Corpus [365], an SMS dataset composed of English text messages[15].

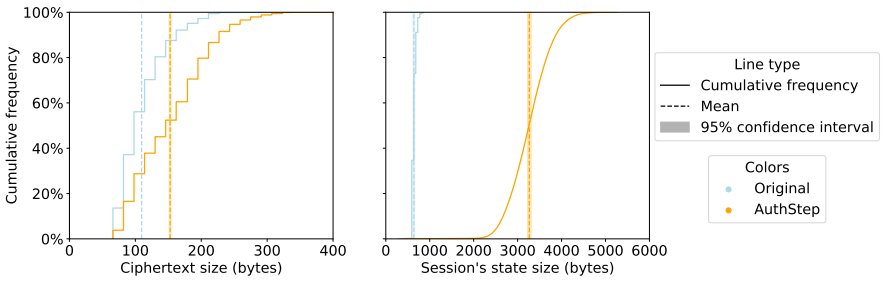


Figure 5.3: Space overhead of the Authentication Steps protocol with a 95% reliable channel.

At the example of a 95%-reliable channel, results, depicted in figure 5.3, show a mean increase of 43 bytes (+ 39%) in ciphertext size and 2.6 KB (+ 411%) in session state size compared to the unmodified Signal protocol. Overheads increase with longer communication epoch lengths and lower channel reliability.

This overhead can be optimized through the usage of trees (for instance Merkle trees [366]) to store hashes, and compress them if consecutive sequences of messages are received. This optimization would be interesting to implement and benchmark, at the same time it would make the underlying analysis and notions more complex. Furthermore, this optimization can only be performed on the receiver's side, as the sender has no way to know which sent messages

[15] T. Chen and M.-Y. Kan, *The national university of singapore sms corpus*, 2015. DOI: 10.25540/WVMO-4RNX.

will eventually be received. Thus, the compression can only happen on at most half of the conversation, and the space optimization is bound by a factor 2.

5.7 Observations on the Official Implementation

While implementing the proposed protocol, we found that the state deletion strategy in Signal's official Java implementation[16] is different from the strategy described in the formal analyses in the literature, such as [349] or [348], even if the latter claims to be based on the implementation. The official Signal specification[17] itself is unclear, and the strategy used in the implementation is implied but not made explicit.

In [349] or [348], post-compromise security kicks in after two epochs, which means that after two epochs of untampered communication after a state compromise, security is restored. This happens by the deletion of no longer necessary state once an epoch ends. However, the Signal implementation deletes this state only 5 epochs later, which is a hardcoded value[18].

Based on this, the following attack is possible, which demonstrates that the official Signal implementation achieves only slightly weaker post-compromise security than claimed in the literature. In the middle of a communication between Alice and Bob, an adversary leaks the state of Alice. Assume that during this epoch i, Alice sent n_i messages to Bob. The adversary can, by using the leaked state, create a valid ciphertext for message $(i, n_i + 1)$ (and even more messages).

Given the literature definition of post-compromise security, as the adversary remained passive, security should be restored at epoch $i+3$. However, with the Signal implementation, as Bob's state for epoch i is not yet deleted at epoch $i + 3$, the adversary can successfully inject messages (for epoch i) to Bob.

Security is however restored 5 epochs after compromise, therefore the implementation still guarantees a weaker post-compromise security property.

An Explanation of this Weaker Property, and Fixing it.

The Signal implementation disregards the total number of messages sent in the previous epoch, which is included alongside messages, and instead keeps the chain key without computing in advance message keys for missed messages. This saves computation time and space as the keys are not computed if those messages never arrive while the immediate decryption property is still valid as the chain key is kept and message keys can be derived if needed.

To fix this, when a new receiving epoch begins, the value of the total number of messages can be used to derive all message keys for this epoch and then delete the chain key from the state. This recovers the strong post-compromise security as claimed in the literature.

[16] Open Whisper Systems, *libsignal-protocol-java*, GitHub, 2021. [Online]. Available: https://github.com/signalapp/libsignal-protocol-java.

[17] T. Perrin and M. Marlinspike, *The Double Ratchet algorithm*, 2016. [Online]. Available: https://whispersystems.org/docs/specifications/doubleratchet/.

[18] Cf. line 210 in https://github.com/signalapp/libsignal-protocol-java/blob/fde96d22004f32a391554e4991e4e1f0a14c2d50/java/src/main/java/org/whispersystems/libsignal/state/SessionState.java#L210.

Both demonstrations of the attack and the proposed fix are made available on Github[19].

5.8 Summary

Messaging protocols such as Signal, that only use their long-term secrets for session initiation, allow for state-compromising adversaries to permanently take over a connection as a Man-in-the-Middle. This chapter offers a strengthened security notion, continuous authentication, which locks out an active adversary post-compromise who has not compromised long-term keys, and enables detection of long-term secret compromises using an out-of-band channel. The proposed Authentication Steps protocol extension generically enables this security in a provably-secure way, adding regular authentication steps in the protocol that leverages long-term keys to authenticate users and ensure no tampering has occurred. Moreover, an out-of-band protocol can be used on top of that to detect adversaries having used long-term secrets to avoid in-band detection.

The overhead introduced by authentication steps was analyzed, benchmarking the prototype implementation which seamlessly integrates on top of the official Signal library. While implementing those benchmarks, we remarked that the official implementation has a weaker post-compromise security property than claimed in the literature.

While this chapter focuses mainly on the Signal protocol, the concept of continuous authentication as well as the Authentication Steps protocol is generic. It can be adapted to other messaging protocols or protocols with long-lived connections, like TLS 1.3 resumption sessions, to provide stronger authenticity guarantees. More generally, this chapter has studied how it is possible to extend an existing cryptographic mechanism to improve the maturity of a zero trust technology, by providing automatic continuous authentication.

[19] https://github.com/apoirrier/libsignal-java-authsteps

Part IV

Beyond Zero Trust

Chapter 6

Building a Zero Trust Federation

In a zero trust architecture, every access to a resource is explicitly verified, without assuming trust based on origin or identity. However, real-world contexts may require organizations to share their resources, *e.g.*, data or services, with other organizations. To facilitate this sharing, they may agree on standards of operation, thus creating a *federation*. In a federated environment composed of multiple domains, ensuring zero trust guarantees for accessing shared resources is a challenge, as information on requesters is generated by their originating domain, yet requires explicit verification from the domain owning the resource.

This chapter proposes a method for federating zero trust architectures, ensuring the preservation of zero trust guarantees when accessing federated resources. The proposed approach relies on remote attestation, enabling continuous authentication and monitoring of requesters, without requiring intrusive software installations on every device within the federation. The feasibility of the proposed federation method is demonstrated by extending the proof-of-concept zero trust architecture from chapter 3, and federating two domains implementing this architecture. This provides detailed information on the federation procedure and its implementation.

6.1 Statement of Purpose

During the Afghanistan intervention, initiated in 2001, the necessity to establish a coalition mission network arose, to facilitate information sharing among allied nations. A first level of interoperability was attained in 2010 with the Afghanistan Mission Network (AMN), enabling informed decision-making through access to comprehensive information[1]. Subsequently, the North Atlantic Treaty Organization (NATO) Military Committee proposed in 2012 an initiative to improve the level of interoperability provided by the AMN, called 'Federated Mission Networking' (FMN)[2]. Widely accepted by NATO nations, the NATO FMN Implementation Plan (NFIP) was designed, and the fourth version endorsed in 2015 [51]. The need for interoperability has been confirmed

[1] J. Stoltenberg, *The secretary general's annual report 2014*, 2014. [Online]. Available: https://www.nato.int/cps/en/natohq/opinions_116854.htm.

[2] NATO, *Federated mission networking*. [Online]. Available: https://dnbl.ncia.nato.int/FMNPublic/SitePages/Home.aspx.

to be fundamental for armed forces by NATO leaders in 2016[3]. Moreover, a need for Multi-Domain Operations (MDO) has been identified by NATO, which expands the requirements for interoperability across the five operational domains and between nations[4].

A Need for Security

Following the Solarwinds attacks in 2021[5], the U.S. government mandated a transition to zero trust[6,7], requiring federal agencies to meet zero trust standards by the end of 2024. This includes the necessity of integrating classified data from all domains in international federations within a zero trust framework[8].

What is a Federation?

A *Federation* consists of a group of organizations that establish agreements with each other, to share resources and services with entities from one another. Within a federation, each organization retains its autonomy, *i.e.*, they manage their resources, services, and entities. In this context, each organization is referred to as a *domain*.

When an entity, known as the *requester*, requests access to a resource located in another domain within the federation, the domain responsible for managing the requester is referred to as the '*requester domain*', while the domain managing the requested resource is known as the '*resource domain*'.

Problem Statement

Zero trust requires every access to a resource to be explicitly verified: access must not be granted based on implicit trust.

In a federation, each domain is responsible for authenticating and monitoring its own entities, devices, network, and resources. Therefore, when an entity requests access to a federated resource, the requester domain provides information about the entity, device, and context of the request. The resource domain must evaluate the trustworthiness of that information. If the resource domain implicitly trusts this information, it would compromise zero trust security guarantees [154].

[3] W. B. King, "Army europe highlights interoperability at communications conference," *U.S. Army*, 2016. [Online]. Available: https://www.army.mil/article/161837/.

[4] NATO, *Multi-domain operations: Enabling NATO to out-pace and out-think its adversaries*, Jul. 2022. [Online]. Available: https://www.act.nato.int/articles/multi-domain-operations-out-pacing-and-out-thinking-nato-adversaries.

[5] J. Rundle, "Solarwinds, microsoft hacks prompt focus on zero-trust security," *The Wall Street Journal*, 2021. [Online]. Available: https://www.wsj.com/articles/solarwinds-microsoft-hacks-prompt-focus-on-zero-trust-security-11619429402.

[6] J. Biden, *Improving the nation's cybersecurity*, Executive order 14028, 2021. [Online]. Available: https://www.federalregister.gov/documents/2021/05/17/2021-10460/improving-the-nations-cybersecurity.

[7] S. D. Young, *Moving the U.S. government toward zero trust cybersecurity principles*, Memorandum M-22-09, 2022. [Online]. Available: https://www.whitehouse.gov/wp-content/uploads/2022/01/M-22-09.pdf.

[8] G. Magram, *Zero-trust interoperability for global defense alliances: 5 ways oracle technology enables classified data integration*, Oracle Cloud Infrastructure Blog, Dec. 2023. [Online]. Available: https://blogs.oracle.com/cloud-infrastructure/post/zerotrust-interoperability-defence-alliances.

6.1.1 Related Work

The creation of a zero trust architecture that can be federated with other architectures, regardless of their zero trust maturity, is evaluated in [155] for the U.S. Air Force. Six solutions are proposed. The first four solutions (ZTE Federation, ZTE-like Federation, Identity Credential Federation, and Weak Identity) involve the requester domain providing information about the requester and its device, which is then trusted by the resource domain. The difference between these architectures is the zero trust maturity level of the federated architecture: a more advanced maturity level offers higher security guarantees. The Ad Hoc Federation solution involves a central authority determining which resource can be shared with which entity. The Person-to-Person sharing solution enables users to share data with other users following a chain of command. However, all these solutions rely on implicit trust between federation members, or trust in the central authority for Ad Hoc Federation. This contradicts the core principles of zero trust, as the trustworthiness of the requester is not explicitly verified.

More generally, there are three types of solutions for creating a zero trust federation, without relying on implicit trust between federated members [156]:

1. Installation of a trusted component in every device accessing resources, including devices from partner domains, to evaluate the security posture of devices.

2. Standardized hierarchical trust architecture: trust is established through a supervising organization.

3. Third-party negotiation: a trusted third-party collects and shares information on every architecture.

The installation of monitoring agents in devices, *e.g.*, through Mobile Device Management (MDM)[9], enables the incorporation of non-managed devices into a zero trust architecture, making access to resources compliant with the zero trust principles [124]. This solution provides a basis for creating a zero trust federation. However, this solution is also intrusive, as every device requires the installation of a monitoring software component from all federated domains.

Federated Identity Management associates multiple Service Providers (SP) and Identity Providers (IdP) for authentication and authorization [367]. It enables users to connect to different SPs, without directly authenticating to each one. Instead, users authenticate themselves to an IdP, which provides them with an authentication token with limited validity, describing their identity. Federated Identity, along with a federation wide Certification Authority, provides a basis for building hierarchical-based federations [125]. Single Sign-On systems enable users to authenticate and to communicate their identities to distinct domains [210]. However, this solution requires domains to trust the federated identity manager [368], which may be compromised during the mission. Therefore, this solution does not follow zero trust principles, as authentication is not explicitly verified by resource owners.

[9] L. Mearian, "What's the difference between MDM, MAM, EMM and UEM?," *Computer World*, 2017. [Online]. Available: https://www.computerworld.com/article/3206325/whats-the-difference-between-mdm-mam-emm-and-uem.html.

The architecture proposed in [126] involves a third party component, called the Context Attribute Provider (CAP), that installs an agent on every device within the federation, monitoring those devices. The CAP then provides PDPs with contextual information in access requests. The CAP can be split into two components: one responsible for collecting contextual information, and another for exploiting it [127]. Similarly to hierarchical architectures, domains need to implicitly trust the CAP. Therefore, access to resources does not follow zero trust principles.

6.1.2 Statement of Purpose

Existing solutions for federating zero trust architectures either rely on trusting a third party or federation partners, or require intrusive techniques such as installing monitoring software on every device. These solutions inherently assume trust, which is never challenged during the mission execution, contradicting the zero trust principles that mandate the explicit verification of access requests. Moreover, in the context Federated Mission Networking, the need for sovereignty and control over systems precludes the use of intrusive techniques and the establishment of trust between nations, or in a third party.

This chapter proposes a method for federating zero trust architectures, that preserves the zero trust security guarantees while limiting the intrusiveness of the solution. The zero trust guarantees of the resulting federation, and its zero trust maturity level, are formally evaluated. The feasibility of building the federation is assessed through a proof-of-concept implementation.

6.1.3 Chapter Outline

The remainder of this chapter is organized as follows. Section 6.2 presents the abstract construction for federating zero trust architectures based on remote attestation, and provides a formal demonstration of how verifying the integrity of zero trust core components only enables zero trust federated access. Section 6.3 applies the method from section 6.2 to the zero trust architecture presented in chapter 3, demonstrating the feasibility of building of a mature zero trust federation. Section 6.4 discusses the deployment of this architecture for military organizations. Finally, section 6.5 concludes this chapter.

6.2 Proposed Zero Trust Federation

This section presents a detailed description of the federation framework proposed in this chapter, highlighting how it enables the federation of multiple zero trust architectures, while ensuring that each domain maintains its inherent zero trust guarantees.

6.2.1 Overview

Figure 6.1 depicts a zero trust federation consisting of two domains: domain A and domain B. Each domain operates autonomously, managing the identity, authentication, and monitoring of infrastructure, devices, and entities. They adhere to zero trust principles, with their own resources being protected by PEPs enforcing access decisions made by PDPs. In the scenario depicted in

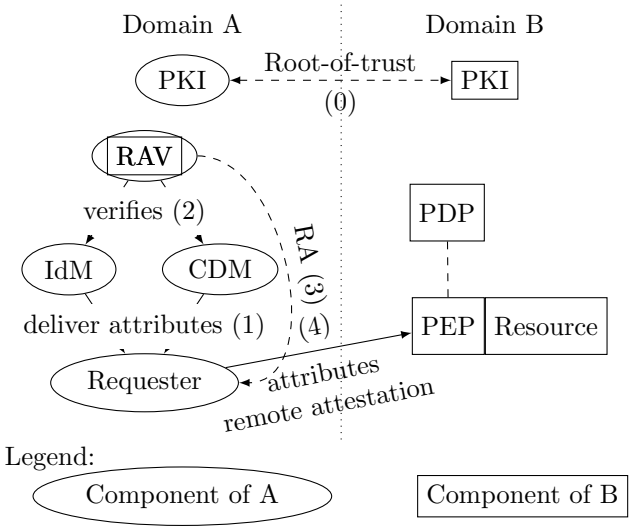


Figure 6.1: Zero trust federated architecture. The architecture is presented asymmetrically, with domain A being the requester domain, and domain B the resource domain.

figure 6.1, a single resource is managed by domain B, and an entity managed by domain A, referred to as the 'requester', requests access to this federated resource. To grant access, the PDP of domain B requires information describing the requester, its device, and environmental context. This information is generated by zero trust components managed by domain A (step 1 in figure 6.1).

Nevertheless, the PDP of domain B must *explicitly verify* the trustworthiness of the provided information, without relying on implicit trust in either domain A or a third party intermediary, and without requiring intrusive software to be installed on the device of the requester. This is achieved through *remote attestation*: domain B installs monitoring components and a Remote Attestation Verifier (RAV) in domain A. Monitoring components observe the IdM and the CDM systems of domain A (step 2), and the RAV verifies their integrity, based on observations received from the monitoring components, producing a Remote Attestation of trustworthiness (step 3). This attestation is then transmitted to the PDP of domain B along with the access request (step 4).

The PDP verifies the attributes and attestation results by relying on the root-of-trust shared between domains, established at the creation of the federation (step 0). Once verification is complete, the PDP proceeds with the authorization procedure.

6.2.2 Authentication and Monitoring

Domain A, as a zero trust architecture, continuously authenticates the requester and its device through the IdM and monitors them using CDM systems, as depicted by arrow 1 in figure 6.1. This produces information describing the requester and its device, which can be represented as attributes for an Attribute-Based Access Control (ABAC) scheme [335].

Interoperability of attributes between domains in the federation is achieved through standardization efforts. For the NATO coalition discussed in the introduction, the FMN defines standards for attribute interoperability, *e.g.*, STANAG 1059 which defines country codes, and STANAG 2116 which defines universal ranks for military personnel. Alternatively, if a specific standard does not exist, bilateral agreements between federation members can be established, similar to the identity mappings proposed in [155].

6.2.3 Root-of-trust Establishment

In a zero trust architecture, entities and devices are initially trusted upon onboarding. This initial trust, referred to as the *root-of-trust*, is composed of secrets in various forms, *e.g.*, cryptographic material, or passwords, providing for instance an identity for onboarded entities and devices, or for configuring multifactor authentication. After an entity or device is onboarded, it is no longer automatically trusted, as it may have been compromised. Instead, trust is explicitly verified through continuous authentication and monitoring.

A zero trust federation requires a similar root-of-trust, which is exchanged when domains federate (arrow 0 in figure 6.1). For the federation method described in this chapter, each domain shares its master certificate with other domains, enabling them to verify authenticated information produced within that domain. Additionally, information about shared resources and requesters can be exchanged.

Moreover, as depicted in figure 6.1, for a requester entity to access a federated resource, remote attestation components must be deployed by the resource domain into the requester domain. These components, described in section 6.2.4, form part of the root-of-trust between federation members, representing the initial trust they have when federating. The initial trust for installing remote attestation components from a foreign domain can be achieved through certification, similar to the process that verifies the trustworthiness of supply chain components, *e.g.*, software or devices, used in the domain. For example, remote attestation components can be designed and implemented by the organization of the requester, and deployed into their domain. The resource domain must ensure, before federating, that requirements for remote attestation components, *e.g.*, ensuring that attestation results cannot be modified or forged, are met.

Similarly, the root-of-trust shared between domains, like certificates and keys shared with entities and devices, represents an initial trust that needs to be reevaluated continuously.

6.2.4 Remote Attestation

This section describes the remote attestation procedure, which consists of two steps: (i) verifying the integrity of zero trust components, and (ii) producing remote attestation results (arrows 2 and 3 of figure 6.1).

When a requester requests access to a resource from another domain, it provides attributes representing its identity, its device, and the context of the request. These attributes are generated and authenticated by the requester domain.

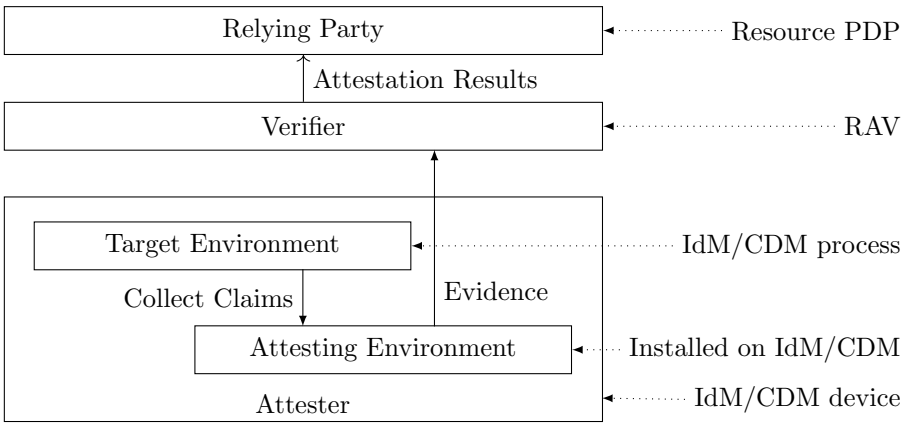


Figure 6.2: RATS general architecture [369]. The right part of the figure shows which role components from figure 6.1 have in the remote attestation general architecture.

To provide zero trust guarantees within the federation, there should be no inherent trust between domains. The root-of-trust does not ensure trust between domains, as entities or systems from the requester domain may have been compromised during the mission. Therefore, the PDP protecting the resource must continuously verify the integrity of attributes provided by the requester, which is enabled through remote attestation.

Remote ATtestation procedureS (RATS) [370] is a framework that enables a peer, called the *Relying Party*, to determine if another peer, called the *Attester*, is trustworthy. The general architecture for RATS is depicted in figure 6.2.

In the context of the federation method presented in this chapter, Relying Parties are the PDPs of resources, and Attesters are the IdM and CDM systems in requester domains. Attestation Results represent the trustworthiness of the Attesters for the PDP. The relationships between the general architecture for RATS and the components of a zero trust federation from figure 6.1 are depicted to the right in figure 6.2.

To demonstrate its trustworthiness to the Relying Party, the Attester produces information about itself, called the *Evidence*, and transmits it to an additional party, called the *Verifier*. The Verifier appraises the Evidence, generating *Attestation Results* that contribute to the decision-making process of the Relying Party regarding trustworthiness [369], [371].

The Attester is divided into two components co-located within the same entity: the *Attesting Environment* and the *Target Environment*, as depicted in figure 6.2. The Attesting Environment collects values and information from the Target Environment by observing it. Results from those observations are represented as *Claims*. These Claims are then typically signed, using cryptographic material contained in the Attesting Environment, to produce Evidence. There are several methods for building Attesting Environments, *e.g.*, Trusted Execution Environments (TEEs) [372], Trusted Platform Modules (TPMs) [373], BIOS firmware, or even software [374].

Figure 6.2 depicts the communication flow within a RATS architecture.

There are two main remote attestation models: the *Passport Model*, and the *Background Check Model* [369]. In the Passport Model, the Attester provides Evidence to the Verifier, which verifies the Evidence and generates Attestation Results. The Verifier then returns the Attestation Results to the Attester, allowing it to present them to Relying Parties for trust evaluation. This enables the Verifier to use the same Attestation Results for multiple Relying Parties. Alternatively, in the *Background Check Model*, the Attester directly transmits Evidence to the Relying Party, which then forwards it to a Verifier to obtain Attestation Results. This model enables different Relying Parties to use their own appraisal policies. These two models can be combined or modified to meet specific implementation requirements.

Attestation Results can take one of two forms: either a boolean value, indicating whether the Attester complies with the appraisal policy of a Verifier, or a richer set of Claims, providing more detailed information about the Attester. The Relying Party takes specific actions based on Attestation Results, *e.g.*, granting partial or full access to a resource.

In the zero trust federation presented in this chapter, remote attestation follows the Passport model: Attesting Environments are installed on the IdM and CDM systems for collecting Claims, and a Verifier is integrated into the requester architecture, generating Attestation Results. The Verifier is trusted by the resource domain, possessing secrets from the resource domain to authenticate Attestation Results. This trust originates from the certification process performed during the creation of the federation, as described in section 6.2.3.

6.2.5 Access Requests

Following the root-of-trust exchange and the installation of remote attestation components presented in section 6.2.4, the federation process is finalized, enabling requesters to initiate queries for federated resources. At this point, every access request must be explicitly verified.

When a requester requires access to a federated resource, it sends an access request to the PEP protecting the resource, as depicted by arrow 4 in figure 6.1. This access request includes attributes that describe the identity of the requester, their device, and the environment. Access requests are authenticated, and may be encrypted, using keys derived from the root-of-trust. Attributes contained in access requests are generated by the IdM and CDM within the requester domain, which correspond to Attesters in the RATS architecture. Therefore, the PDP of the resource domain, which corresponds to the Relying Party in the RATS architecture, can evaluate the trustworthiness of the IdM and CDM of the requester domain through remote attestation. If deemed trustworthy, it implies that the attributes in the access request are also trustworthy, thus explicitly verifying the access request. If access is granted, an end-to-end secure channel is established between the requester and the resource, using secrets based on the shared root-of-trust.

The solution presented requires each domain in the federation to host a set of remote attestation components for every other domain in the federation. This design choice improves scalability and reduces the attack surface, by eliminating the need to install foreign device agents on every device in the federation. Alternatively, if relying on a third party federated IdM is possible,

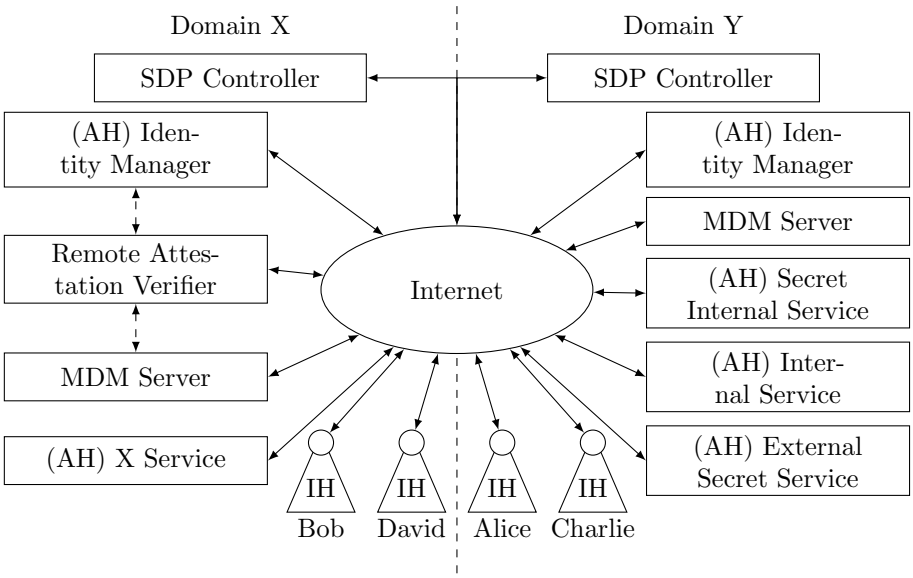


Figure 6.3: Architecture of the proof-of-concept of a zero trust federation.

further scalability can be achieved by installing (and certifying) only one set of remote attestation components, to monitor the third party IdM.

6.2.6 Zero Trust Guarantees

The federated architecture presented in this section adheres to the zero trust principles: every requester is authenticated and monitored by their origin domain, and the resource domain explicitly verifies, through remote attestation, the integrity and authenticity of these attributes.

This construction enables least-privilege, per-session, and dynamic authorization. Every domain retains control over access to their resources, ensuring that resources are isolated, and that segments are dynamically established between authorized requesters and resources, using encrypted channels. As a result, every zero trust principle is strictly followed in the federation, without relying on implicit trust between domains.

6.3 Proof-of-Concept Zero Trust Federation

This section describes a proof-of-concept for a zero trust *federation*, demonstrating the feasibility of federating two separate two zero trust architectures, each built upon the implementation from chapter 3. This section details the necessary adjustments needed for federating them.

The proposed zero trust federation is depicted in figure 6.3. It consists of two domains: domain X and domain Y. Within each domain are deployed an SDP controller, respectfully referred to as 'controller X' and 'controller Y', an IdM, an MDM server, services, and users. Each user is assigned a single device.

Three services are hosted by domain Y, each with unique characteristics: the 'Secret Internal Service' has a classification level of 'Secret' and is accessible

only to Y personnel; the 'Internal Service' has a classification level of 'None' and is restricted to Y personnel; and the 'Secret External Service' has a classification level of 'Secret', and allows users from any organization to access it. Two users are managed by domain Y: 'Alice', with a clearance level of 'Secret', and 'Charlie', with a clearance level of 'None'.

A single service, 'X Service', is hosted by domain X, having a classification level of 'None' and restricted to X personnel. Two users are managed by domain X: 'Bob', with a clearance level of 'Secret', and 'David', with a clearance level of 'None'.

As described in section 6.2, federating both architectures requires the establishment of a root-of-trust, the installation of remote attestation components, and the implementation of extended access requests for enabling federated access. To illustrate which components are required for federating domains, this proof-of-concept is a unidirectional federation, permitting only users from domain X to access services from domain Y, but not the inverse.

6.3.1 Federating SDP Controllers

An intuitive solution for federating architectures is that clients from domain X directly contact the SDP controller from domain Y to obtain a list of federated services, and then connect to those services as an IH from domain Y. However, this would require clients to be onboarded into domain Y as an IH, *e.g.*, by installing secrets on the device of the user, which may not be feasible.

An alternative solution is proposed, where users, devices, and services are onboarded within their own domain, receiving secrets and configurations from the SDP controller of their domain. For federating domains, only SDP controllers share secrets and configurations directly, which can then be forwarded to entities within the domains.

In this proof-of-concept, since only clients from domain X access federated services, controller X is onboarded into domain Y as an IH. Therefore, controller X is granted SPA keys, a private key, and an associated certificate from the PKI of domain Y. Moreover, SDP controllers exchange their respective PKI root certificates, which are installed on every system within each domain, enabling the verification of identities for devices and for services, of SAML assertions, and of MDM assertions.

6.3.2 Remote Attestation

Federating architectures following the method from section 6.2 requires remote attestation. In the proof-of-concept, the remote attestation procedure follows the Challenge/Response interaction model [369], depicted in figure 6.4. An attester process is installed on the IdM and MDM systems in domain X, continuously gathering evidence about the integrity of those systems. A verifier is also installed in domain X, which regularly queries the attester, collecting the evidence and appraising it to generate attestation results. These results are then sent to controller Y. The verifier is an IH for domain Y, provided with SPA keys and private key from the PKI of domain Y during onboarding. The remote attestation procedure serves as a verification mechanism for the controller of domain Y, ensuring that the attributes produced by the IdM

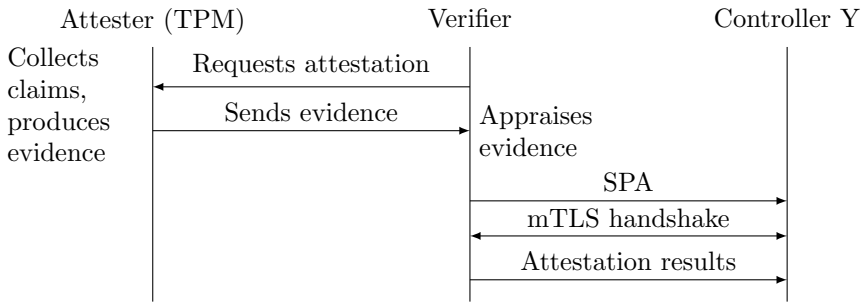


Figure 6.4: Remote attestation protocol in the proof-of-concept, performed regularly.

and by the MDM server of domain X are correctly generated, and are thereby trustworthy.

The chosen remote attestation implementation is based on simulated TPMs, as provided by the IETF Working Group on Remote Attestation: CHARRA[10]. The evidence appraised by the verifier includes Platform Configuration Registers [375], *i.e.*, read-only registers that record software state. In the event of attestation failure, access rights for federated clients are revoked. No remediation policy is in place for attestation failures in this proof-of-concept.

6.3.3 Protocol Workflow

Figure 6.5 depicts the process by which an authenticated user obtains a list of available federated services, and accesses a federated service. It is a direct extension of the non-federated workflow presented in figure 3.2 on page 62, for the proof-of-concept architecture of chapter 3.

Following arrows 1 to 17 in figure 3.2, where the user is authenticated and controller X retrieves internal authorized services, controller X establishes a secure communication channel with controller Y, using SPA and mTLS (arrows 18 and 19 in figure 6.5). Controller X then sends the user and device attributes to controller Y (arrow 20), which verifies the SAML assertion and the signature of device attributes, using certificates from the IdM and MDM server of domain X. If verification is successful, controller Y queries the PE of domain Y with the attributes (arrow 21), and receives a list of federated services available to the user from domain X (arrow 22). This list is then forwarded to controller X (arrow 23).

Once controller X receives the list of federated services, it forwards the combined list of both internal and federated services to the user (arrow 24). A screenshot of the view of each user in the federation is provided in figure 6.6, illustrating that each user views a different set of available services, depending on their attributes. Notice that Bob has access to the 'External Secret' federated service from domain Y.

To access a federated service, the user selects the corresponding service in their browser, which makes the device agent sending a request to access this

[10] https://github.com/Fraunhofer-SIT/charra

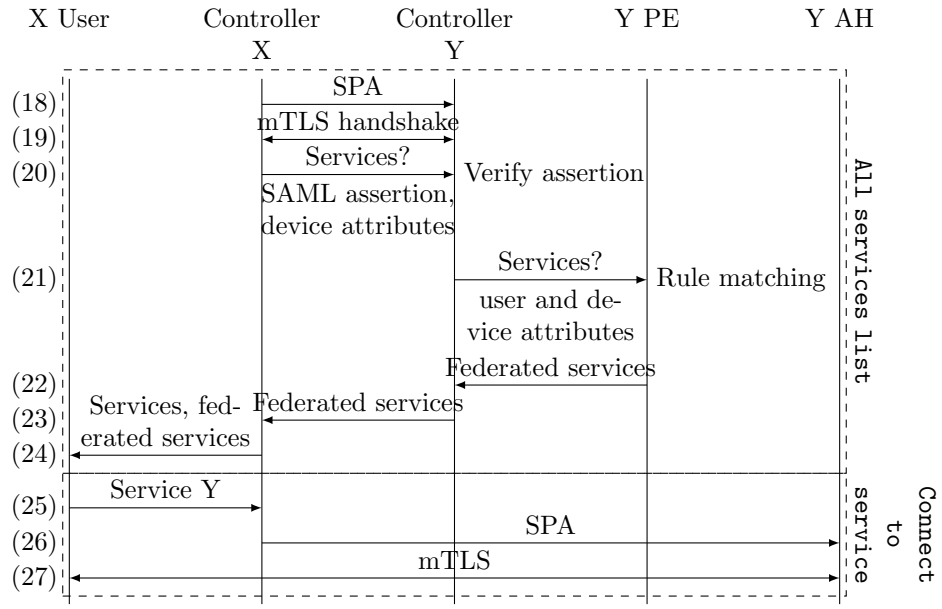


Figure 6.5: Access workflow in a federation, as an extension of the access workflow of the single proof-of-concept architecture from chapter 3, depicted in figure 3.2 on page 62, with the user from domain X accessing a service of domain Y.

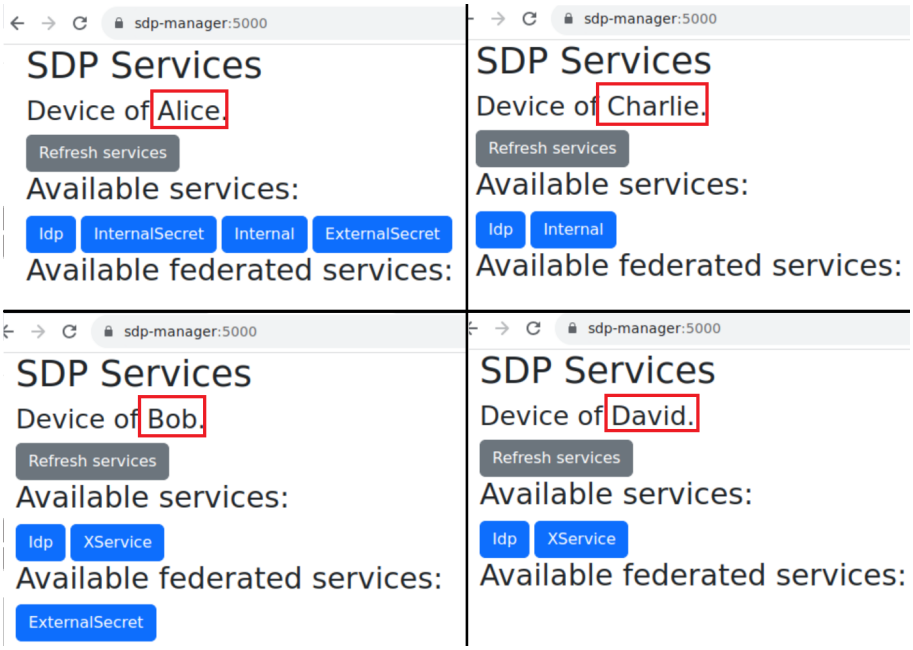


Figure 6.6: Screenshot of the list of available services for each user in the federation.

service to controller X (arrow 25). Subsequently, controller X acts as an IH for the federated service, sending an SPA packet to the SDP gateway which contains the IP address of the device of the user as authorized device (arrow 26). This authorizes the user to establish an mTLS connection with the federated service (arrow 27).

6.4 Applications in Military Networks

In the proof-of-concept presented in this chapter, devices, *e.g.*, laptops, phones, IoT devices, or servers, are simulated using Docker containers, which are networked together.

In a military context, clients and services are deployed on the battlefield. Connections between them are part of the *data plane*, and other communications are part of the *control plane*. Two models have been considered for the control plane.

The first model involves deploying every SDP component on the battlefield, including the SDP controller, the PE, the IdM, and the MDM server. This requires servers with sufficient computing power to run these processes, as well as stable bandwidth between hosts and the controller. This solution is relevant in the context of MDO, where activities across multiple domains are orchestrated in joint operations.

The second model involves deploying control plane elements in a combat cloud [52], *i.e.*, a network dedicated to data distribution and information sharing within a battlefield, at the edge of the tactical network. In this model, communication between SDP controllers, verifiers, MDM servers, and IdMs is expected to be more stable, with fewer resource constraints on these components. However, connectivity with IHs and services may be less stable, which could result in clients being denied access to services when a client or a service fails to connect to the control plane, even if it should be authorized and if connectivity between client and service is available. This problem can be mitigated by increasing the validity of policy enforcement, *e.g.*, gateway configurations or authentication tokens, thereby reducing the need for communications between the tactical network and the combat cloud, at the expense of less dynamic policies.

6.5 Summary

The desire for inter-domain collaboration – as exemplified in this chapter through international military cooperation – creates a need for interoperable federated architectures for data and service sharing. The zero trust paradigm provides principles for securing data and services by implementing fine-grain compartmentalization and least privilege access policies.

However, state-of-the-art zero trust federated architectures either require the installation of intrusive software on all devices, raising concerns about privacy and sovereignty, or assume an inherent trust between federation partners, which contradicts the core principle of zero trust.

To address this paradoxical issue of 'trust in a zero trust world', this chapter has presented a novel method for federating zero trust architectures. This method leverages remote attestation in a novel way, to constantly monitor

only core zero trust components (*i.e.*, Identity Manager, and monitoring systems). With this approach, every access request is explicitly verified, even when authorizing access to an entity from another domain in the federation, without implicitly trusting federation partners. Thus, zero trust principles are preserved without requiring intrusive software on every requester device.

This chapter has also demonstrated how the zero trust implementation from chapter 3 can be extended to implement a zero trust federation, with detailed workflows illustrating the necessary steps for building it.

Chapter 7

Privacy-Preserving Forwarding

In the zero trust paradigm, *trust* is no longer implicitly granted, but is explicitly and continuously verified. Such trustworthiness is evaluated to allow access to resources, *e.g.*, a service or data. In particular, one assumption of zero trust is that networks are considered an untrusted zone, with subjects and asserts needing to assume their local network hostile [83].

Nevertheless, the zero trust framework mostly focuses on protecting the confidentiality, integrity, and authenticity of communications and assets. This is aligned with the definitions of 'security' from [1], [2] as the prevention of unauthorized use or access of systems. However, this definition of security does not prevent potentially sensitive information from leaking, as metadata, *e.g.*, the fact that two entities are communicating, can be sensitive information. The protection of metadata is referred to as *privacy*, *i.e.*, the ability to determine whether, when, and to whom personal information is released [1]. For example, the zero trust architectures studied in chapter 2 do not ensure the anonymity of communications, especially on untrusted networks.

This chapter provides an example study of privacy that can be achieved for network communications. In an IP-based computer network, routers make forwarding decisions based on their internal state (*e.g.*, their routing table), as well as on information (*e.g.*, the source and the destination) conveyed in the header of each incoming datagram. This header information being unencrypted implies that an adversary, observing incoming and outgoing datagrams over the interfaces of a router, can learn both information about the communication patterns of the network, and about the network topology. In particular, if the network topology is considered sensitive information, this also implies that compromising a single router, and extracting and inspecting its state, constitutes an efficient attack.

This chapter explores different privacy-preserving datagram-based forwarding mechanisms, as an interchangeable mechanism for the IP forwarding mechanism, and evaluates to what extent they preserve privacy properties such as anonymity, unlinkability and topology-hiding, both from external observers recording traffic passing through a router, and from the router itself. Those privacy properties are formally defined. This chapter proposes novel forwarding mechanisms, formally proves their privacy guarantees, and compares them to existing privacy-preserving forwarding mechanisms. Additionally, a physical benchmarking platform consisting of a Raspberry Pi functioning as a router

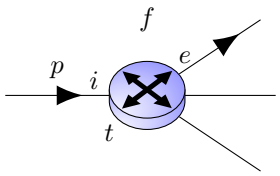
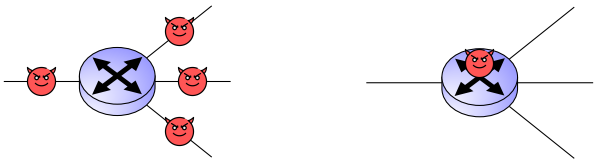


Figure 7.1: A datagram being forwarded by a router.



(a) External observer. (b) Internal observer.

Figure 7.2: Types of observers.

is used to evaluate the performance of forwarding mechanisms. The trade-offs between privacy benefits and performance is discussed.

7.1 Statement of Purpose

A *unicast forwarding mechanism* is a function, f, in routers which, as depicted in figure 7.1, takes as input a forwarding table, t, a datagram, p, an ingress interface, i, and which returns either an egress interface, $e = f(t, p, i)$, onto which the datagram is to be forwarded – or, if the datagram is to be silently dropped [53].

Traffic passing through routers can be observed by unauthorized individuals or organizations, as illustrated in figure 7.2a, which allows them to access sensitive information transmitted over the network. In IP datagrams headers, the source and destination addresses are carried in an unencrypted form, making it possible for an observer observing the traffic entering and leaving the router to learn the sources and destinations of datagrams, and thus learn the communication patterns as well as parts of the network topology. Furthermore, if the observer has compromised a router, and thus has access to its memory, as depicted in figure 7.2b, it can acquire the forwarding table of the router and can extract both information on the network topology and the communication patterns.

This chapter studies, and proposes, forwarding mechanisms that prevent observers from learning the network topology and the communication patterns. Two types of observers are considered: the *external observer*, which records datagrams entering and leaving the observed router, and the *internal observer*, which has access to the internal state of the router. Note that an internal observer is a more powerful observer than the external observer: all the information that an external observer can infer from its observations can also be inferred by an internal observer.

When communication occurs, two types of information can be gathered by an observer: information about the content of messages – the payload – and meta-information (information that does not include the content of messages).

The payload can be obfuscated, for example by encrypting it at the network layer using IPsec [376], or at the transport layer with TLS [177]. However, meta-data cannot be directly obfuscated, as all routers along the path between source and destination need access to that information to correctly forward datagrams. As there are already are solutions to provide payload opacity, this chapter considers the protection of meta-data only.

Privacy [54], [55] represents the prevention of meta-information being gathered by observers. It is different from confidentiality, which protects the content of messages [56].

This chapter considers several *privacy properties*, in the context of message exchanges:

- *Anonymity* [57], which is the impossibility for an observer to know the sender or recipient of a message.
- *Unlinkability* [57] between messages, which means that an observer cannot decide if two messages are related (for example, have the same sender or recipient).
- *Topology-hiding* [58], which means that the topology of the network is opaque.

When external or internal observers are present, anonymity is not guaranteed with IP forwarding, as datagrams headers contain the source and destination addresses of messages unencrypted. Therefore, unlinkability is also not guaranteed. Indeed, unlinkability depends on anonymity: if the source or destination of messages can be inferred by an observer, the observer can link related messages. Topology-hiding is only partially guaranteed: although the observer is not able to fully determine the topology of the network from the internal state of the router and observed datagrams, the observations help to define what topologies are plausible.

This chapter explores to what extent forwarding mechanisms can obfuscate the forwarding information to preserve privacy, from both external observers and internal observers, while maintaining their forwarding functionality.

7.1.1 Related Work

On the topic of evaluating privacy, [59] and [57] both measure privacy as a probability that an observer acquires private information. Privacy is achieved if the probability of the observer discovering the private information is no greater than the probability of discovering the private information without knowledge of any observation.

There are several solutions for achieving privacy of communication for forwarding mechanisms.

A first category of solutions protect the privacy of the communication by making use of an overlay network, changing the route of datagrams. A first example is IPsec [376], which extends the IP protocol. It operates in two modes: the *transport mode* encrypts only the payload, whereas the *tunnel mode* completely encrypts and encapsulates the original IP datagram between two endpoints. Tunnel mode provides anonymity and unlinkability from external observers located between the endpoints of the tunnel, as the source and

destination of the message are encrypted in the tunnel. This is however not the case before datagrams enter and after they leave the tunnel. Moreover, the tunnel provider, or an internal observer having access to the encryption keys, knows the sources and destinations of the encapsulated messages. Similarly, a framework that creates encrypted tunnels between peers to protect communication anonymity is proposed in [377].

An alternative to tunnel-based approaches for providing anonymity is *mix networks* [378], which were originally designed for email exchanges, but can also be used for IP traffic anonymisation [379]. Mix networks use proxy servers, called *mixes*, which accumulate datagrams from multiple senders before forwarding them in a random sequence towards their destinations. Observers are not able to link an egress datagram to its corresponding ingress datagram. Mist routing [380] and onion routing [381]–[383] combine the tunnel and mix networks ideas. These techniques use intermediate servers to route datagrams along a randomly chosen path, to hide their sources and destinations. However, privacy is not guaranteed on the first and last segments of datagrams routes. Mix, mist and onion routing shuffle traffic exchanged in the network to prevent datagrams to be linked. Therefore, they are dependent on the amount of traffic being exchanged in the network: if there are few datagrams, there is a higher probability to link datagrams and to find their sources or destinations. To alleviate this dependency, 'fake' traffic can be added [384]. Techniques similar to differential privacy [13], [385], [386] can be used to determine the right amount of fake traffic to add, and to formally prove privacy properties.

Cryptographic techniques are also employed for preserving the privacy of communications for forwarding mechanisms. Functional encryption [226] is an encryption technique which, given a message, x, and a function, f, can encrypt x, yielding a ciphertext, c, and derive a *function key*, k_f, such that given the function key and the ciphertext, it is possible to retrieve $f(x)$, but without learning any more information on x. Example functions for functional encryption schemes include arithmetic operations on boolean circuits [387] or inner products [388], an inner product being a way to multiply vectors together, with the result of the multiplication being a scalar. The use of functional encryption for forwarding is stated in [60], but no technical details are given. Homomorphic encryption [227] is an encryption scheme for performing computations on ciphertexts, without needing decryption. It is used by [61] to obfuscate datagrams for privacy-preserving routing. Similarly, [62] uses homomorphic encryption to protect the anonymity of communications for inter-domain forwarding. For every flow of datagrams, a first initialisation datagram is sent towards the destination to create a private route between the source and the destination, using homomorphic encryption. This route is stored in subsequent datagrams, which no longer need encryption operations.

Without using cryptography, label switching, such as Multi-Protocol Label Switching (MPLS) [63], can be used to obfuscate the sources and destinations of datagrams for an external observer. MPLS relies on a dictionary mapping IP addresses to a label. IP datagrams are encapsulated into a labelled datagram when they enter the MPLS architecture. This label is then used by *label switching routers* to forward datagrams, using the forwarding table of the router which maps labels to their next hops. For an external observer to compromise anonymity, both the labelled datagram and the mapping between IP addresses and labels is needed. However, datagrams with the same destination

have the same label, so MPLS does not guarantee unlinkability.

The topology-hiding problem is introduced by [58], which provides theoretical results for Multi-Party Computation (MPC) schemes (*i.e.*, schemes which involves participation for a computation of every node in the network), which are improved by [389], [390]. In [391], [137], topology-hiding routing protocols are studied in the context of Mobile Ad-Hoc NETworks (MANET). Datagrams do not carry routing information, and instead rely on peer-to-peer communication between every node to create routes, while keeping the network topology private.

To summarize, there exists several methods for privacy-preserving forwarding mechanisms. A first category of solutions relies on overlay networks, such as tunnel-based solutions and onion routing. However, these techniques detour traffic on potentially inefficient paths, and are thus not studied in this chapter. Other techniques rely on obfuscation, either through cryptography, which incurs an overhead for forwarding datagrams, or through labelling. The privacy guarantees and performance of those obfuscation techniques is evaluated, which composes the baseline for comparison with forwarding mechanisms proposed in this chapter.

7.1.2 Statement of Purpose

This chapter presents privacy-preserving forwarding mechanisms as an interchangeable mechanism for the forwarding mechanisms used in classic IP forwarding. Several mechanisms are presented, that allow forwarding packets along a shortest path from source to destination, and which rely on encryption and datagram header obfuscation to preserve anonymity, unlinkability, and to provide topology-hiding, against both external observer and internal observers. Forwarding mechanisms proposed in this chapter enable a configurable trade-off between privacy guarantees and performance. Formal analyses of privacy guarantees are provided, with an experimental evaluation of the overhead introduced by each forwarding mechanism, including comparison with existing techniques.

7.1.3 Chapter Outline

The remainder of this chapter is organized as follows. Section 7.2 details and formalises the context of networks, and of forwarding therein, as studied in this chapter. Section 7.3 then formally defines privacy in the context of forwarding mechanisms, and details the three privacy properties considered: anonymity, unlinkability and topology-hiding.

Section 7.4 proposes a set of different forwarding mechanisms and evaluates the privacy properties guaranteed by those mechanisms. Performance comparisons for the forwarding mechanisms are presented in section 7.5. The practical use of presented forwarding mechanisms is discussed in section 7.6. Finally, section 7.7 concludes the chapter.

7.2 Networks and Forwarding Mechanisms

This section presents networks and forwarding mechanisms as they are studied throughout this chapter.

A *network* is a connected undirected graph of *routers* and *hosts*. Every host has exactly one link, towards a router. Routers are devices, performing forwarding as depicted in figure 7.1. This chapter does not focus on routing protocols: forwarding tables are supposed configured before forwarding occurs in the network. Hosts are endpoints for communication, and are not performing forwarding operations. Thus, the remainder of this chapter considers networks simply as graphs of routers, with some routers also connected to a set of hosts.

Definition 7.1 (Network)**.** *A network is a graph $\mathcal{G} = (\mathcal{R}, \mathcal{E})$ where $\mathcal{R}$ is a set of routers connected by links in $\mathcal{E}$. Every router $r \in \mathcal{R}$ is linked to a (potentially empty) set of hosts $\mathcal{H}[r]$.*

If $r \in \mathcal{R}$, let $\mathcal{N}[r] = \{s \in \mathcal{R} | (r, s) \in \mathcal{E}\} \cup \mathcal{H}[r]$ be the set of *neighbors* of router r (other routers and linked hosts). Given a router, r, and a neighbor $s \in \mathcal{N}[r]$, there exists a connection point on the router, called an *interface*, that enables r to communicate with s. If traffic can enter the router through that interface, it is referred as an *ingress* interface, and if traffic can leave through that interface, it is referred as an *egress* interface.

7.2.1 Communication

Communication in the graph is represented as a set of *messages*.

Definition 7.2 (Message)**.** *A message is a tuple $m = (\text{src}, \text{dst}, \text{payload})$ which represents the source, destination, and content of a message. The source and destination are distinct hosts.*

A message is physically forwarded on the network using *datagrams*.

Definition 7.3 (Datagram)**.** *A datagram is a pair $p = (\texttt{header}, \texttt{payload})$: it is composed of a* header, `header`*, defined by the forwarding mechanism used in the network, and of the* payload *of the message,* `payload`*. Datagrams transmitting a same message m have the same payload. The header of datagrams contains information used for forwarding datagrams towards their destination.*

7.2.2 Forwarding Mechanisms

Definition 7.4 (Forwarding mechanisms)**.** *A* forwarding mechanism *is an algorithm,* `Forward`*, running on every router. When a host sends a message towards another host, it sends the message to the router to which it is connected. The payload of the message is then encapsulated into a datagram. This encapsulation operation is denoted* `Send`*.*

Forwarding is the operation depicted in figure 7.1. When a router receives a datagram p from an ingress interface i, the router calls $\texttt{Forward}(\texttt{state}[r], p, i)$, with $\texttt{state}[r]$ being its internal state (*i.e.*, forwarding table). $\texttt{Forward}(\texttt{state}[r], p, i)$ outputs an egress interface onto which the datagram is to be forwarded, or silently drops the datagram. Datagram headers may be modified by the router. This formalism allows capturing header modifications such as IP Time-to-live update [64], or IPv6 hop-by-hop options processing [65]. The payload remains unchanged. For a given router r, $\texttt{datagrams}[r]$ denotes the set of datagrams

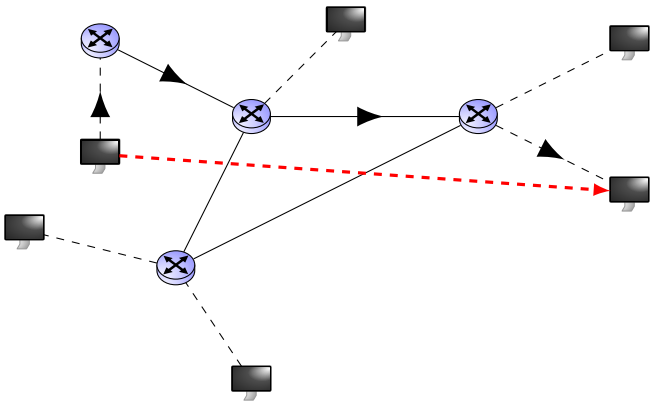


Figure 7.3: An example network. Simple black lines represent the links in the network, dashed lines represent the links between routers and hosts. The red dashed arrow represents a message between two hosts. Arrows on the links represent the datagrams used to forward the message towards its destination.

passing through r. An example network with a message is illustrated in figure 7.3.

As stated in section 7.1.2, this chapter only considers forwarding mechanisms that forward datagrams along one of the shortest path from source to destination. It is assumed that the forwarding tables of routers are a-priori configured such that this property is satisfied.

7.3 Privacy

This section formalizes the privacy properties considered in this chapter, derived from the probability-based definitions of [57], [59].

7.3.1 Privacy Games

Privacy means the prevention of meta-information being gathered by observers. Observers can be external entities, and are in this case referred to as *external observers*, or can be a router which passively records datagrams, referred to as an *internal observer*.

Any meta-information to keep secret from observers has an associated *privacy property*: Given a network, the communication happening in the network, and an observer, the privacy property is said *verified* if the observer does not learn the meta-information associated with the privacy property.

To reason on privacy properties, a *game* between a *network model* and an observer is used, illustrated in figure 7.4. Given a public set of hosts and routers, the network model is a 'black box', simulating a random communication happening in a random network. The observer can query the network model once, requiring observations performed on a router, r, of its choice:

- For an external observer, the network model replies with the neighborhood $\mathcal{N}[r]$ of the observed router, as well as the set $\texttt{datagrams}[r]$ of datagrams passing through r.

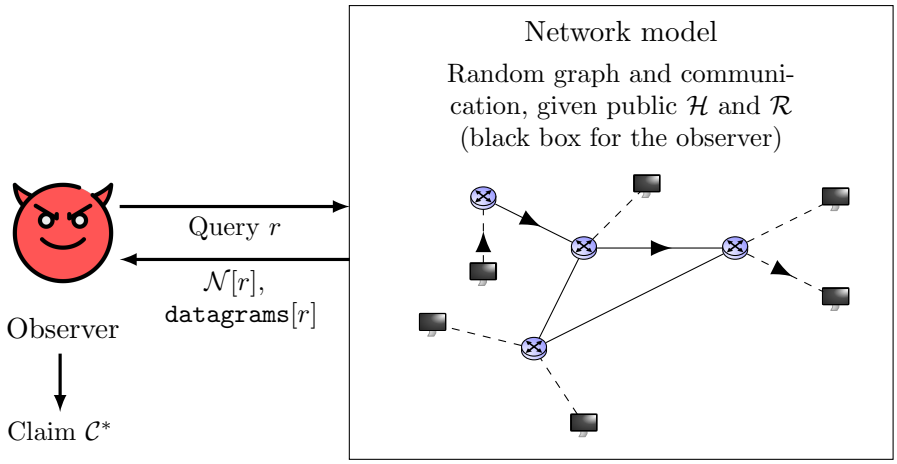


Figure 7.4: Privacy game for external observers.

- For an internal observer, the network model replies with the same information as is given to an external observer, as well as the state $\texttt{state}[r]$ of the observed router.

The observer then produces a claim, $\mathcal{C}^*$, which represents the meta-information that it infers from its observations. The claim corresponds to the private information that the observer tries to deduce from its observations. The *advantage* of an observer is defined as the probability that it produces a correct claim.

Definition 7.5 (Privacy guarantee)**.** *Given a forwarding mechanism, its* privacy guarantee *for a privacy property is defined as* $1 - A$*, where* A *is the upper bound on advantages for every possible observer.*

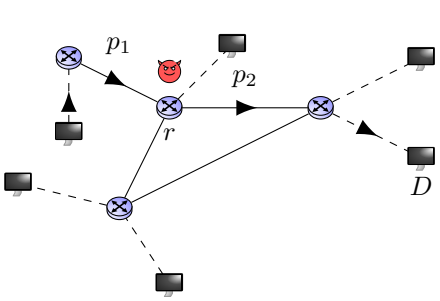
The privacy guarantee is a number between 0 and 1 which represents a level of privacy that is guaranteed by the forwarding mechanism. A value of 0 means that no privacy is guaranteed, and there exists an observer who always infers the meta-information from its observations, whereas a value of 1 means that no inference is possible.

7.3.2 Privacy Properties

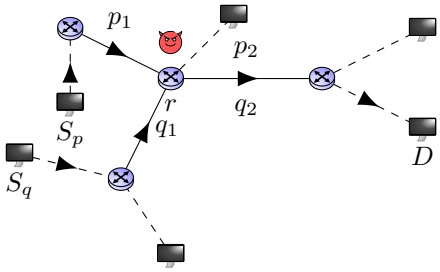
This chapter considers three kinds of privacy properties: anonymity, unlinkability and topology-hiding. The corresponding claims for their privacy games are presented in this section, with illustrations provided in figure 7.5.

Anonymity

Anonymity is a privacy property capturing that an observer cannot infer what the source or destination of a datagram are, except if the source or destination of the datagram is a host directly connected by way of the observed router. Anonymity is formally defined using a privacy game, where the claim of the observer, $\mathcal{C}^* = (b^*, d^*, h^*) \in \{0,1\} \times \texttt{datagrams}[r] \times \mathcal{H}$, is a statement that one of the observed datagrams, d^*, has a host h^* as source or destination (b^*



(a) The anonymity claim ($b^* = 1, p^* = p_2, h^* = D$) is valid as the destination ($b^* = 1$) of p_2 is D.



(b) The unlinkability claim ($b^* = 1$, $p_1^* = p_2, p_2^* = q_2$) is valid as p_2 and q_2 belong to different messages and their destination ($b^* = 1$) is identical (D).

Figure 7.5: Illustrations of valid claims.

is 0 if the claim concerns a source, 1 if it concerns a destination). The claim is correct if d^* is a datagram corresponding to a message m^*, such that both its source and destination are not connected by way of the observed router r and its destination (resp. source) is h^* if b^* is 1 (resp. 0). An example of a correct claim is depicted in figure 7.5a.

Unlinkability

Unlinkability captures that an observer cannot perform statistics on the observed datagrams: given a set of observed datagrams, the observer cannot infer which datagrams have the same source or destination. Similarly to anonymity, unlinkability is defined as a privacy game, in which the observer outputs a claim, $\mathcal{C}^* = (b^*, p_1^*, p_2^*) \in \{0,1\} \times \texttt{datagrams}[r]^2$.. The claim represents that two observed datagrams, p_1^* and p_2^*, which do not correspond to the same message, have the same source or destination ($b^* = 1$ for the same destination, $b^* = 0$ for the same source). The claim is correct if the observed router r is not directly connected to the source or destination of p_1^* and p_2^*, if p_1^* and p_2^* correspond to two distinct messages $m_1^* \neq m_2^*$, and if they have the same source or destination. This is illustrated in figure 7.5b.

Topology-hiding

The third privacy property captures that an observer cannot infer the network topology from the observed communication. Similarly to anonymity and unlinkability, topology-hiding is defined using a privacy game. The observer outputs a claim $\mathcal{C}^* = \mathcal{G}^*$ representing its guess that the network graph on which the communication occurs is $\mathcal{G}^*$. The observer wins the game if the networks are equal: $\mathcal{G}^* = \mathcal{G}$.

It may be infeasible for an observer to learn the entire topology of the graph. However, the observer can learn partial information of the graph from their observations, which increases its probability to find the exact graph used by the network model. As privacy is measured as a probability, the advantage of the observer reflects the amount of information learnt by the observer. This

is true for every privacy property: privacy games are useful for assessing the level of privacy guaranteed by forwarding mechanisms. In other words, privacy games measure the effectiveness of forwarding mechanisms to preserve privacy.

7.3.3 Minimal Advantage

Privacy is defined in [57] by comparing the probabilities of the observer learning the secret meta-information before and after it has performed observations. Privacy is achieved if these probabilities are equal. This definition is not directly applicable in the context of forwarding mechanisms: the claim of the observer may depend on the observations performed, thus there cannot be a probability, pre-observations. For example, the anonymity claim involves an observed datagram, which exists only if the system runs. Nevertheless, it is possible to define a base probability, as the probability that a claim chosen at random is correct.

Anonymity and Unlinkability

For anonymity and unlinkability, this minimal probability is $p_{\min} = \frac{1}{H}$, where $H = |\mathcal{H} \setminus \mathcal{H}[r]|$ is the number of hosts not directly linked to the observed router.

Indeed, if the observer selects at random a datagram whose source and destination is not a directly connected host, then the destination of the datagram is in $\mathcal{H} \setminus \mathcal{H}[r]$. Because the communication is generated randomly, a random choice of a host in $\mathcal{H} \setminus \mathcal{H}[r]$ would yield a correct anonymity claim with probability $p_{\min}$.

Similarly, for unlinkability, if the observer observes two random datagrams, corresponding to distinct messages and whose source and destination are not a directly connected host, then their destinations are in $\mathcal{H} \setminus \mathcal{H}[r]$. Thus, there is a probability $p_{\min}$ that their destinations are equal.

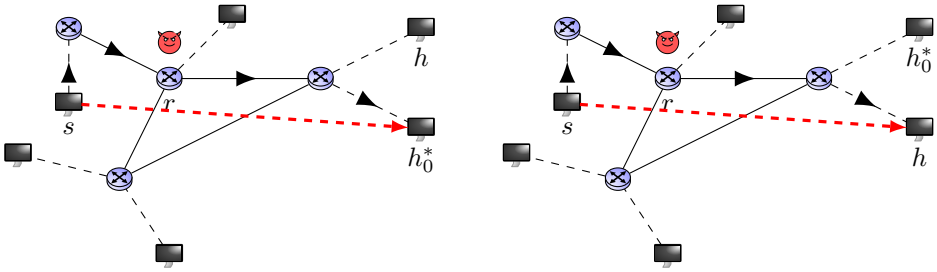
Topology-hiding

There is a finite number of routers and hosts, thus there is a finite number of possible graphs for the network. An observer producing a random graph amongst all possible has a non-zero probability, $p_{\min}$, of guessing the correct graph for the network.

7.3.4 Minimal Leakage

Definition 7.6 (Minimal leakage)**.** *Forwarding leaks information to the observer, independently of the forwarding mechanism in use. This information is denoted the* minimal leakage.

A *leakage game* is a game, similar to privacy games, except that the network model only gives partial information to the observer, called the *leakage*, and not the complete set of datagrams passing through the observed router. Leakage does not depend on the underlying forwarding mechanism. Leakage games are used to represent the probability of observers breaking privacy properties, independently of the underlying forwarding mechanism.



(a) $\texttt{Game}_{h_0^*}$: a message between s and h_0^*.

(b) $\texttt{Game}_h$: a message between s and h.

Figure 7.6: Illustration of the proof of minimal leakage for external observer. Between both games, h and h_0^* have been swapped both in the communication and the network. An observer located on router r would observe the same datagram paths, but the destinations are not the same in the games.

External Observers and Datagram Paths

An external observer, $\mathcal{O}$, observing a router, r, sees datagrams being forwarded by r. This is a leakage, referred to as the *datagram paths*.

Every datagram $p \in \texttt{datagrams}[r]$ being forwarded by r has an ingress interface, i_p, and an egress interface, e_p. Given an observed router r, the observed datagram paths are the set of ingress and egress interfaces for every observed datagram, $\{(i_p, e_p)\}_{p \in \texttt{datagrams}[r]}$.

Anonymity

Proposition 7.1 (Minimal leakage for an external observer for anonymity.)**.** *An external observer having only access to datagram paths cannot win the anonymity game with a probability higher than the minimal winning probability,* $p_{\min} = \frac{1}{H}$*, defined in section 7.3.3.*

Proof. Let $\mathcal{O}$ be an observer playing the anonymity leakage game in which the network model only gives $\mathcal{O}$ the datagram paths. Further, let r be the observed router.

If an observer $\mathcal{O}$ wins the game, then it has produced a correct claim, $\mathcal{C}^* = (b_0^*, p_0^*, h_0^*)$, meaning that $\mathcal{O}$ claims that the destination of an observed datagram p_0^* is the host h_0^* (if $b_0^* = 0$, the claim concerns the source instead of the destination). This game win by $\mathcal{O}$ is later referred as the *original game*, and denoted $\texttt{Game}_{h_0^*}$.

There exists at least $H = |\mathcal{H} \setminus \mathcal{H}[r]|$ scenarios (distinct graphs and communications) that lead to the same datagram paths, which is the input of $\mathcal{O}$ in the game. Those games are described below. Thus, $\mathcal{O}$ would output the claim $\mathcal{C}^*$ with the same probability as in the original game, but this claim is correct only in one game amongst the H different ones.

For every host $h \in \mathcal{H} \setminus \mathcal{H}[r]$, let $\texttt{Game}_h$ be the same game as $\texttt{Game}_{h_0^*}$, except that hosts h and h_0^* have been swapped (both in the network, and in the communication). As h and h_0^* have been swapped both in the network and in the

communication, datagrams in $\texttt{Game}_h$ follow the exact same path as datagrams in $\texttt{Game}_{h_0^*}$, as illustrated on figure 7.6. In both those cases, $\mathcal{O}$ is given the same input, but only wins the game in $\texttt{Game}_{h_0^*}$.

Therefore, the probability that $\mathcal{O}$ outputs a correct claim is, at most, $\frac{1}{H}$, which proves that an external observer having only access to datagram paths cannot win the anonymity game with a probability higher than the minimal winning probability. □

Unlinkability Contrary to anonymity, an external observer having only access to datagram paths has a higher probability of linking two datagrams than the minimal probability. Indeed, as stated in section 7.2.2, datagrams are forwarded along one of the shortest path between source and destination. Therefore, if the observer selects two different datagrams being forwarded on the same egress interface, the probability that those two datagrams have the same destination is greater than the probability for two random datagrams. More precisely, if there are k possible destinations behind that interface, then the probability of the observer to win the game is $\frac{1}{k}$, which is strictly better than the random probability if only a strict subset of destinations is reachable by that interface ($k < |\mathcal{H} \setminus \mathcal{H}[r]|$).

Topology-hiding Datagram paths add constraints on a plausible graph $\mathcal{G}$: if there is a packet coming from a router $x \in \mathcal{N}[r]$ and leaving towards $y \in \mathcal{N}[r]$, it means that (x, y) is not an edge in the graph.

Therefore, the probability of winning the game of an observer, taking those constraints into account, is strictly greater than the random probability.

Internal Observers and Unobfuscated Forwarding Table

Internal observers have access to the state of the observed router. This state depends on the underlying forwarding mechanism.

However, independently of the forwarding mechanism, routers are able to perform forwarding of initial datagrams using only their internal state. Therefore, even if the forwarding table is obfuscated, an internal observer can infer from the state an *unobfuscated forwarding table*, which associates to every destination possible next-hops: $\texttt{nexthops}[r] = \{d : X_d\}_{d \in \mathcal{H}}$, where for a host d, X_d is a set of possible next-hops for that destination from router r.

Given only a datagram path and an unobfuscated forwarding table, an observer can find the source or destination of a datagram coming from, or going to, next-hop x with probability $\frac{1}{|\mathcal{D}_x|}$ where $\mathcal{D}_x$ is the set of destinations such that x is a possible next-hop. Thus, if, in the leakage game, the observer selects datagrams going towards x with $\mathcal{D}_x$ having few members, it increases the probability of breaking anonymity and unlinkability compared to the random probability. Moreover, $\mathcal{O}$ needs only to randomly choose sub-graphs in every $\mathcal{D}_x$ in order to reconstruct a plausible graph $\mathcal{G}$, thus having a higher probability of success than when the unobfuscated forwarding table is not available.

Table 7.1: Summary of Forwarding Mechanisms.

Forwarding mechanism	**Datagram header**	**Router memory**	**Forwarding algorithm**
IP	destination IP	routing table	longest prefix matching (LPM)
Encrypted IP	encrypted destination IP	key, routing table	decryption, then LPM
MPLS and k-labels	label	forwarding table	read line in table
Encrypted k-labels	encrypted label	key, forwarding table	decryption, read line in table
Functional encryption	encrypted destination	function key	apply function decryption

Table 7.2: Privacy guarantees of considered forwarding mechanisms.
Green tick: minimal leakage. Red cross: no privacy. Orange approximately symbol: partial leakage.
EIP: Encrypted IP. k-l: k-labels. Ek-l: Encrypted k-labels. FE: Functional Encryption.

	IP	EIP	MPLS	k-l	E k-l	FE
Anonymity (ext. obs)	✘	✔	✔	✔	✔	✔
Anonymity (int. obs)	✘	✘	✔	✔	✔	✔
Unlinkability (ext. obs)	✘	✔	✘	≈	✔	✔
Unlinkability (int. obs)	✘	✘	✘	≈	≈	✔
Topology (ext. obs)	≈	✔	≈	≈	✔	✔
Topology (int. obs)	✔	✔	✔	✔	✔	✘

7.4 Privacy-preserving Forwarding Mechanisms

This section describes several methods for creating privacy-preserving forwarding mechanisms. Studied forwarding mechanisms are summarized in table 7.1, which describes for each forwarding mechanism the forwarding algorithm, whose inputs are the datagram header and the router internal memory. Privacy guarantees of each forwarding mechanism are evaluated and formally proven, as summarized in table 7.2.

7.4.1 Header Encryption

In order to enhance the privacy guarantees offered by a forwarding mechanism, it is possible to encrypt the header of datagrams, making the carried meta-data no longer readable by external observers.

To achieve unlinkability, two identical datagram headers should never be encrypted such that the same ciphertext results. To do so, a nonce-based encryption scheme [392] is used: In the simplest possible configuration, every router uses a shared symmetric key. When a host sends a message, the router it is connected to calls `Send` from the underlying forwarding mechanism, and

encrypts the datagram header using the symmetric key before forwarding it onto the network.

When a router receives a datagram, it first decrypts its header using the symmetric key, then calls `Forward` to retrieve the egress interface of the datagram. If the datagram header changes (*e.g.*, because of IPv6 hop-by-hop options [65]), the header is encrypted again before being forwarded.

Header Encryption Satisfies Privacy Properties Against External Observers

Proposition 7.2 (Privacy with Header Encryption)**.** *Header encryption prevents external observers from gaining more information than the datagram paths, as defined in section 7.3.4.*

Proof. The proof is based on the *indistinguishable from random bits* property [392] of nonce-based encryption schemes, which states that ciphertexts cannot be distinguished from random bits. Therefore, external observers only see datagrams passing through the observed router, with headers appearing to be random. Thus, headers are not useful to the observer for deducing private information. Therefore, the observer does not learn any additional information than they would by observing empty datagrams. □

This proves that protocols using encryption do not leak information other than the datagram paths to external observers, and therefore preserve anonymity, unlinkability and topology-hiding to those, as described in section 7.3.2. However, each router holds the encryption key, and thus can decrypt the encrypted headers. Therefore, header encryption does not provide additional privacy guarantees against internal observers than without encryption.

7.4.2 MPLS for Anonymity

Label switching, *e.g.*, MPLS, can be used to offer anonymity, using labels to hide the identity of destinations.

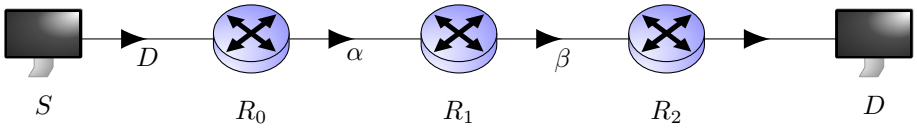
When a host, s, sends a message to a destination $d \in \mathcal{H}$, `Send` converts the public identity of d to a label $\hat{d}$. This label is then used as an initial label, for MPLS to forward the message towards its destination.

For traditional use of MPLS, labels are either assigned manually by a network administrator, or automatically using signalisation protocols such as the Label Distribution Protocol (LDP) [66].

For the purpose of this chapter, labels are distributed randomly, such that destinations are replaced with labels. Thus, each destination is identified on a given router by a random label, preventing observers from deducing from labels and forwarding tables the real destination of datagrams.

More precisely, labels are distributed as follows. First, let $\mathcal{H}_Z = \{h^{(i)}\}_{h\in\mathcal{H}, i\in[\![0,Z-1]\!]}$ be a set containing Z copies of $\mathcal{H}$. This set will be used to randomly assign labels.

The state of every router $r \in \mathcal{R}$ is composed of a *destination table* and a *forwarding table*. These tables are used to forward datagrams towards their destination, while preserving privacy properties. The forwarding protocol is illustrated in figure 7.7. The destination table is a table used by `Send` to convert destinations to labels. Labels are used for forwarding: when a router



Destination table of R_0		
Dest	Next	Label
D	R_1	α
⋮	⋮	⋮

Forwarding table of R_1		
Input	Next	Label
α	R_2	β
⋮	⋮	⋮

Forwarding table of R_2		
Input	Next	Label
β	D	-
⋮	⋮	⋮

Figure 7.7: Illustration of MPLS and of the k-labels protocol. Host S sends a message to host D. It forwards it to its linked router R_0. R_0 reads the next-hop R_1 and label α in its destination table at entry D. Upon reception, R_1 reads the entry with input label α of its forwarding table: it contains the next hop R_2 and new label β. The same happens with R_2, which forwards the datagram to host D.

receives a datagram, the label contained in the datagram header is used to find, in the forwarding table, the next-hop and next label of the datagram.

On the assumption that there exists routes between a router, r, and every host, both tables of r are filled as follows:

1. A permutation $\sigma_r : \mathcal{H}_2 \to [\![1, 2|\mathcal{H}|]\!]$ is chosen uniformly at random.
2. For each destination $d \in \mathcal{H}$, not connected by r, whose next hop is $x \in \mathcal{N}[r]$, set:
   - $\texttt{destination}_r[d] = (x, \sigma_x(d^{(0)}))$;
   - $\texttt{forwarding}_r[\sigma_r(d^{(0)})] = (x, \sigma_x(d^{(1)}))$;
   - $\texttt{forwarding}_r[\sigma_r(d^{(1)})] = (x, \sigma_x(d^{(1)}))$.
3. Entries are added in the destination table and forwarding table for hosts connected by r, signifying that forwarding is immediate.

With this construction, for every destination $d \in \mathcal{H}$, there are exactly two labels in the forwarding table that correspond to d: $\sigma_r(d^{(0)})$ and $\sigma_r(d^{(1)})$. Moreover, this construction ensures that datagrams entering r, and whose destination is d, have one of those two labels as header.

Routers do not keep the permutation as part of their internal state, only the tables. This enables a separation of labels between the destination table and the forwarding table, which guarantees anonymity.

MPLS Ensures Anonymity Against Internal Observers

This section proves that internal observers are not able to learn more information than the unobfuscated forwarding table, as defined in section 7.3.2. For anonymity, it means that given a datagram p whose next-hop is x, the internal observer knows only that the destination of the datagram is in $\mathcal{D}_x$, the set of destinations such that x is a possible next-hop.

Given an internal observer, $\mathcal{O}$, which guesses that some datagram p^* with next-hop x has a host d^* as destination, it is possible to construct $|\mathcal{D}_x|$ states such that only one instance (p^*, d^*) is a correct guess. Indeed, for any destination $d \in \mathcal{D}_x$, it is possible to create an alternative forwarding table by swapping the labels for d and d^* in the forwarding table. Because labels are chosen uniformly and randomly, every forwarding table is equally probable. However, the guess of $\mathcal{O}$ is right only in the case where $d = d^*$, which proves that its probability of winning is lower than $\frac{1}{|\mathcal{D}_x|}$.

This proves that internal observers cannot distinguish datagram destinations from anonymity sets $\mathcal{D}_x$ for $x \in \mathcal{N}[r]$. A similar argument be made, which shows that MPLS does not reveal information about the network topology to the observer. However, two datagrams corresponding to two different messages, but going to the same destination, share the same label. Therefore, unlinkability is not guaranteed. As shown in the previous section, header encryption (with MPLS as base forwarding mechanism) can provide unlinkability, but only against external observers.

7.4.3 k-labels Forwarding Mechanism

With both IP forwarding and MPLS, datagrams passing through a router, and going to the same destination, carry the same identifier. This prevents unlinkability.

To counter this, the *k-labels mechanism* associates more labels (for example, k) to each destination. With this technique, given k datagrams with distinct labels, it is not possible to know the number of distinct destinations of those datagrams. For example, it is not possible to distinguish if those k datagrams have k distinct destinations, or if they are all going to the same destination. This results in an increased number of datagrams needed for an observer to link datagrams. However, if $k + 1$ datagrams have the same destination, according to the pigeonhole principle [67], at least two datagrams share the same label, so unlinkability is limited.

The k-labels mechanism extends MPLS. Its construction is similar to MPLS, except that at each router, $2k$ labels are used for a given destination.

As with MPLS, a random permutations are used to generate labels. The permutations transpose $2k$ versions of $\mathcal{H}$ instead of 2: for every router $r \in \mathcal{R}$, a permutation $\sigma_r : \mathcal{H}_{2k} \to [\![1, 2k|\mathcal{H}|]\!]$ is chosen uniformly and randomly. Every router, r, uses k labels per destination to create initial datagrams. For forwarding, there are $2k$ distinct labels used for the same destination. The forwarding table and destination table of a router r are populated as follows:

- $\texttt{destination}_r[d] = (x, \{\sigma_x(d^{(i)})\}_{i<k})$ for every destination d with next-hop x.
- $\texttt{forwarding}_r[\sigma_r(d^{(i)})] = (x, \sigma_x(d^{(k+i\%k)}))$ for every destination d and every $i \in [\![0, 2k-1]\!]$.

`Send` randomly selects a label from $\texttt{destination}_r[d]$. Forwarding for the k-labels mechanism works exactly like MPLS forwarding.

k-labels Unlinkability

If two labels are reused, then an observer can infer that those two datagrams have the same destination. Nevertheless, an internal observer cannot produce definitive statistics on the observed communication, in the sense that given l datagrams with distinct labels going towards the same next-hop, the observer cannot infer which datagrams have the same destination.

k-labels Topology-hiding

For IP and MPLS, every datagram seen by an external observer adds a constraint on the plausible network as the observer learns the existence of a destination behind that interface.

The observer learns this information immediately for IP and for MPLS, whereas it requires at least $k+1$ datagrams for the k-labels mechanism.

Internal observers already know the distribution of destinations behind every next-hop: they cannot learn more information from observing the traffic from the k-labels mechanism.

7.4.4 Functional Encryption

Functional encryption [226] is an encryption technique which enables to securely transmit messages, while allowing holders of a functional key to learn specific information on the message, but without learning more than is authorized.

Functional encryption is leveraged for privacy-preserving forwarding. To do so, datagrams carry information on the path they are following. This information is encrypted using the functional encryption scheme, and each router can only reveal the local information on the path of the datagram (*i.e.*, the next hop).

More precisely, inner-product functional encryption is leveraged for privacy-preserving forwarding. In an inner product functional encryption scheme [388], messages are vectors, $\mathbf{x}$, and functions are inner products: $f_{\mathbf{y}} : \mathbf{x} \mapsto < \mathbf{x}, \mathbf{y} >$, where $\mathbf{y}$ is a vector.

To configure a communication network, every router, r, first assigns a random identifier (an integer) to their neighbors. This assignment is a permutation, $\sigma_r : \mathcal{N}[r] \to \mathbb{N}$, associating to every neighbor of r an integer. Then, for every possible host $d \in \mathcal{H}$, let $\mathbf{x}^{(d)}$ be a vector of size $|\mathcal{R}|$. For every router r, the r-th coordinate of $\mathbf{x}^{(d)}$ is the integer representing the egress interface of r on which a datagram going towards d must be forwarded: $\mathbf{x}_r^{(d)} = \sigma_r(y)$, where y is the next hop for d from r.

Those vectors are stored in each router, and act as the destination table of the router: for every possible destination, they describe an encoded version, due to the permutation used in each router, of the complete path to reach the destination. Additionally, every router, r, is configured with a secret key corresponding to the function $f_{\mathbf{e}_r}$, with $\mathbf{e}_r$ the vector all 0, except coordinate r being 1.

When a datagram is to be sent to d, $\mathbf{x}^{(d)}$ is encrypted using functional encryption. Let $\mathbf{y}^{(d)}$ be the ciphertext that results. This ciphertext is included in the header of the datagram. When a router, r, receives a datagram whose

Table 7.3: Internal memory required in a router for studied forwarding mechanisms.

IP	Encrypted IP	k-labels	Encrypted k-labels	Functional encryption
$O(\|\mathcal{N}[n]\|)$	$O(\|\mathcal{N}[n]\|)$	$2k\|\mathcal{V}\|$	$2k\|\mathcal{V}\|$	$O(\|\mathcal{H}\|\cdot\|\mathcal{R}\|)$

header contains $\mathbf{y}^{(d)}$, it computes, using its secret key:

$$f_{\mathbf{e}_r}(\mathbf{x}^{(d)}) =< \mathbf{e}_r, \mathbf{x}^{(d)} >= \mathbf{x}_r^{(d)} = \sigma_r(y).$$

As the router knows σ_r, it learns the correct next hop, and forwards the datagram.

Privacy Properties Guaranteed by Functional Encryption

Functional encryption is a form of encryption. Therefore, the results from section 7.4.1 apply, and the forwarding mechanism preserves anonymity and unlinkability against external observers.

Moreover, the functional encryption security property ensures that given a functional key, no information, besides the return value from the function, can be learnt from the ciphertext. When forwarding, functions only provide the next hop for a given datagram. Therefore, functional encryption also preserves anonymity and unlinkability against internal observers.

However, every router stores a copy of the route vectors, which contain information on the route of datagrams. This provides an internal observer with information on the network topology, for example the degree of routers in the graph.

7.5 Performance Evaluation

This section presents an evaluation of the performance of the different forwarding mechanisms, presented in section 7.4, including the memory necessary to store the state for every forwarding mechanism in each router and, experimentally, the time necessary to forward datagrams.

7.5.1 Memory Computation

The IP protocol [64] uses a forwarding table to forward incoming datagrams. Forwarding tables are designed to be compressed using tree algorithms for better efficiency, leading on average to $O(|\mathcal{N}[r]|)$ entries for a router r [68].

MPLS uses memory in each router, proportional to $6|\mathcal{H}|$: the destination table has $|\mathcal{H}|$ entries consisting of a label and a next-hop, and the forwarding table has $2|\mathcal{H}|$ entries each composed of a label and a next-hop. More generally, the k-labels mechanism requires memory proportional to $(1+5k)|\mathcal{H}|$: $|\mathcal{H}|$ entries of size $1+k$ in the destination table and $2k|\mathcal{H}|$ entries of size 2 in the forwarding table.

Encryption only adds a symmetric key to the state, but does not change the orders of magnitude of the required memory.

Functional encryption requires to store functional encryption keys and destination vectors. Both keys and destination vectors are vectors of size $|\mathcal{R}|$: their size is proportional to the number of routers in the network. Each router needs to store $|\mathcal{H}|$ destination vectors. Thus the required memory, $O(|\mathcal{H}|\cdot|\mathcal{R}|)$, is very high compared to the other considered protocols. Moreover, ciphertexts in functional encryption are also vectors of size $|\mathcal{R}|$, which need to be transmitted with every datagram. With current functional encryption schemes, this is unpractical for networks larger than dozens of routers.

These results are summarized in table 7.3.

7.5.2 Experimental Benchmarking Environment

A test-bed for evaluating the performance of forwarding mechanisms is constructed, which consists of a Raspberry Pi, acting as a router, and a client. The Raspberry Pi is a Raspberry Pi 4 Model B with 4GB of RAM. The router is connected to the client using a single Ethernet cable. This physical interface is associated with several virtual interfaces, each on a different VLAN. The router listens on every virtual interface and forwards datagrams on those virtual interfaces.

Performance is measured as the CPU time used by the forwarding process to forward each datagram. The forwarding process runs on an isolated core to prevent other processes from interfering with the measures. As the code is not optimized for the specific CPU architecture and runs in user space, the raw value of the processing time is not particularly relevant: only the comparison between protocols is considered in this chapter.

In the experiment, the router acts as a member of a fat tree data centre architecture [393], which is an architecture used in the financial, medical, government and military sectors [69], where privacy may be desirable. Therefore, if the router has k interfaces, there are $\frac{k^3}{4}$ hosts in the data centre, exchanging messages passing through the router.

Traffic

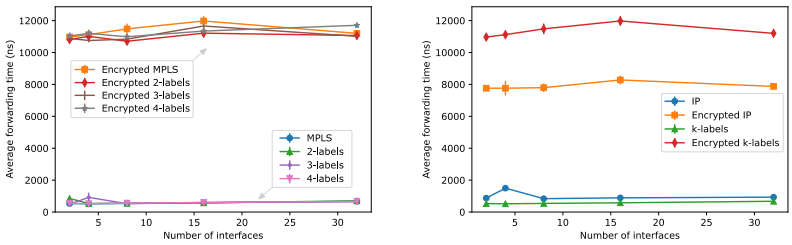
The router is exposed to UDP datagrams containing messages going from a random host to another random host. 300 datagrams per interface are generated, to measure forwarding time. All traffic used in this simulation is based on IPv6. An IPv6 routing header [65] is used to store the labels or encrypted IP addresses. For protocols requiring nonce-based encryption, AES-CTR with a key length of 128 bits is used. The full implementation is available on GitHub[1].

7.5.3 Results

Figure 7.8 depicts the measured performance of the studied forwarding mechanisms. 95% confidence intervals for the average forwarding time are displayed as vertical bars on the figures.

Figure 7.8a depicts the performance of the k-labels mechanism for different values of k. The only differentiating factor is the presence, or absence, of encryption. Performance is similar for different values of k. Indeed, label

[1] https://github.com/apoirrier/encrypted-router



(a) Comparing k-labels between themselves.

(b) Comparison of IP and k-labels, with or without encryption.

Figure 7.8: Comparison of average forwarding times.

switching is performed in constant time, which does not depend on the forwarding table size. The main impact of increasing the number of labels per destination is on the size of the state in every router. Therefore, in figure 7.8b, MPLS represents the performance of the k-labels mechanism.

Without encryption, the k-labels mechanism is faster than IP, as next-hops can be directly accessed by performing a simple pointer operation, as opposed to tree algorithms used by IP. Note that the k-labels mechanism uses more space than plain IP forwarding in those instances, as described in table 7.3. Encrypted IP is around 8 times slower than regular IP forwarding, as for every packet forwarded, one AES decryption operation is needed. Encrypted k-labels are around 20 times slower than unencrypted k-label forwarding, as there needs to be two AES operations: one decryption of the incoming label and one encryption for the outgoing label. Functional encryption has not been shown on the figure, as the forwarding time for a datagram is more than 100000 times larger than for 'classic' IP forwarding.

7.6 Discussion

As presented in table 7.2, IP forwarding does not guarantee anonymity nor unlinkability. There are several ways to address this.

7.6.1 Datagram Header Encryption

A first solution is to encrypt datagram headers using a shared key. This guarantees privacy properties, albeit only against external observers, as discussed in section 7.4.1. This comes at the expense of a forwarding operation 8 times slower, as shown in figure 7.8b.

Having a slower forwarding operation has several impacts on an operational network. The electricity consumption of routers increases, thus requiring more batteries if the router is a portable device. Routers can also process fewer datagrams per time unit, which may increase network congestion in case of heavy traffic.

7.6.2 Datagram Header Obfuscation

The second solution presented in this chapter is to replace destination IP addresses with labels. As discussed in table 7.2, this provides anonymity, but only partial unlinkability, both against external observers and internal observers.

Contrary to datagram header encryption, the k-labels mechanism forwards datagrams with a time equivalent as IP forwarding, as presented in figure 7.8b. The degree of unlinkability corresponds to the amount of traffic that an observer needs to observe, before being able to know which datagrams have the same source or destination. This degree is configurable by way of the number of labels per destination (k). Whereas increasing the number of labels does not impact forwarding time, as depicted in figure 7.8a, the required memory in each router increases linearly with k, as shown in table 7.3.

Finally, it is possible to encrypt labels in the k-labels mechanism. This provides 'perfect' unlinkability against external observers at the expense of a forwarding operation 20 times slower than the unencrypted k-labels mechanism.

7.6.3 Functional Encryption

While using functional encryption enables to preserve anonymity and unlinkability, even against internal observers, functional encryption schemes are not practical, because of the prohibitive additional memory in routers and datagrams, as well as forwarding times more than 100000 larger than for 'classic' IP forwarding.

7.6.4 Related Work Analysis

Overlay solutions such as tunnel-based solutions and onion routing protect the anonymity of the communication, and, in some extent, the unlinkability. However, the network topology is supposed known. In terms of performance, datagrams are encrypted, which adds an encryption overhead. Moreover, datagrams follow longer paths than the optimal path, which may incur significant delay for the communication. Nevertheless, this strategy enables stronger privacy properties to be achieved, as it is no longer assumed that datagrams take a direct route between source and destination, enabling the anonymity set to be greater than the one presented in section 7.3.4.

Solutions based on encryption, such as homomorphic routing [62], preserve the anonymity of the communication by creating temporary routes for each flow. Per-flow state is not needed in routers, however the size of datagrams is increased similarly to the functional encryption technique presented in this chapter. For performance reasons, unlinkability is not guaranteed for datagrams in a same flow.

7.7 Summary

This chapter has proposed, and evaluated, three privacy properties: anonymity, unlinkability, and topology-hiding, in the context of forwarding mechanisms. Those privacy properties represent the fact that some information is kept secret from observers while forwarding datagrams.

This chapter has proposed new forwarding mechanisms, which do not impact the datagram paths, and can be used as a direct replacement for forwarding mechanisms such as IP. They are derived from IP and MPLS, to offer anonymity, unlinkability and topology-hiding against external and internal observers.

The first forwarding mechanism is derived from MPLS and uses multiple labels to represent the same destination. This enables a forwarding performance comparable to IP or MPLS, while augmenting privacy, at the expense of a memory overhead in every router. The second forwarding mechanism uses encryption. This hinders forwarding performance as cryptographic operations are needed for every datagram, but provides optimal privacy against external observers. Finally, the use of functional encryption for providing optimal privacy against both external and internal attackers has been evaluated. While it theoretically provides optimal privacy, the memory overhead in datagrams and routers make it impossible to use.

Experimental benchmarks have been performed to evaluate performance, and serve as a proof-of-concept for preserving privacy in forwarding operations.

Part V

Conclusion

Chapter 8

Conclusion

The increasing complexity of IT services has driven organizations to outsource these functions, allowing them to leverage outsourced expertise without assuming responsibility for implementation and maintenance. This is characterized by the adoption of cloud hosted Software-as-a-Service, Platform-as-a-Service, and Infrastructure-as-a-Service offerings. Furthermore, the rise of remote work and Bring-Your-Own-Device policies has transformed the use of IT services, as employees increasingly rely on unmanaged devices or untrusted networks to access organization resources. These evolutions of IT usage have profoundly impacted IT architectures, transitioning from controlled, perimeter-based systems to dynamic frameworks that incorporate untrusted systems and networks. Concurrently, the frequency and complexity of cyberattacks have escalated, with insider threats and Advanced Persistent Threats posing a significant risk to cybersecurity. The concomitant evolution of cyberattacks, IT infrastructures, and IT usage has led to the emergence of a new security paradigm: *zero trust*.

This approach is founded on the principle 'never trust, always verify', abandoning traditional perimeter-based security models in favor of continuous authentication and dynamic authorization. Zero trust establishes robust security principles for securing organizations, by creating micro-perimeters to protect resources with a fine granularity, while also leveraging enhanced identity, dynamic access control, and real-time monitoring of users, systems, networks, and assets.

However, migrating from traditional architectures to zero trust architectures presents significant challenges. Implementing zero trust requires the integration of multiple technologies and capabilities, which is difficult due to the lack of clear guidelines for achieving high maturity levels in zero trust implementations. Furthermore, evaluating the formal security posture of an architecture is complex, as is its adequation with real-world threats. These challenges are addressed in this manuscript.

This thesis has, in part II, provided an in-depth analysis of zero trust.

Specifically, chapter 2 has presented a comprehensive survey of zero trust definitions, technologies, and existing architectures, studying extensively its core principles and their manifestations in migration strategies, capabilities, and technologies. This enabled the development of a formalized framework,

for positioning architectures relatively to their compliance with the zero trust paradigm. This framework was at the origin of improvements for zero trust proposed in part III and of discussions on IT security presented in part IV. While zero trust increases the security posture of organizations following its principles, it also introduces new vulnerabilities and does not guarantee protection against every threat, therefore requiring formal analyses for understanding the benefits and trade-offs of zero trust.

Chapter 3 demonstrated how to construct a practical zero trust architecture by modifying and integrating several open-source components. The resulting construction yields a proof-of-concept implementation with advanced zero trust maturity, which was used as a base zero trust architecture throughout this manuscript.

Then, part III investigated the discrepancies between existing zero trust literature and the underlying principles.

Specifically, chapter 4 highlighted that current zero trust architectures primarily focus on securing endpoints to resources and data, while neglecting the protection of data at rest and effective management of encryption keys. To address this gap, chapter 4 proposed a method for enhancing a base zero trust architecture with data-centric security, by incorporating Attribute-Based Encryption capabilities within the architecture. This approach encrypts data to maintain confidentiality and integrity, only allows authorized entities to decrypt it, and offering greater flexibility in storing sensitive data. Overall, the proposed method effectively addresses the data pillar of zero trust maturity models.

Chapter 5 explored part of the identity and automation pillars of zero trust maturity models. In the context of secure messaging, an extension to the messaging protocol is proposed, that integrates automation into continuous authentication. This approach reduces the need for manual interventions in securing communication channels, while also enhancing the formal security of continuous authentication.

Finally, part IV examined the practical applications of zero trust architectures in real world scenarios, assessing the effectiveness and relevance of the security posture achieved through zero trust.

Chapter 6 explored how it is possible to federate zero trust architectures. Indeed, even if an organization is secure through zero trust, it may still need to share data or services with another organization. However, the zero trust paradigm prevents the implicit trust of the foreign organization. To address this challenge of 'trust in a zero trust world', an approach that leverages the remote attestation technology was proposed, to explicitly verify the trustworthiness of the foreign organization.

Chapter 7 took a step further by postulating that confidentiality, integrity, availability, and authenticity of data and services are insufficient for ensuring security, as metadata can also reveal sensitive information. Therefore, the chapter investigated how to perform operations, *e.g.*, network layer forwarding, while preserving the privacy of the communication.

In sum, the work and results presented in this thesis consist of:

1. A comprehensive survey and taxonomy of zero trust definitions, technologies, and architectures, resulting in an evaluation framework for assessing the zero trust maturity of architectures, submitted to a journal;
2. A proof-of-concept zero trust architecture, based on open-source components, demonstrating how to build zero trust architectures through modification and combination of various products – this work has been presented in a workshop and submitted to a journal;
3. A method for integrating data-centric security within a zero trust architecture, submitted to a conference;
4. An extension to messaging protocols that automates continuous authentication, further enhancing post-compromise security properties – this work has been presented in a workshop and published in a conference;
5. A method for enabling multiple organizations using zero trust architectures to share resources and data, while not implicitly trusting other organizations, published in a conference and submitted to a journal;
6. A study on privacy considerations on untrusted network infrastructures, submitted to a journal.

The results achieved in this thesis have demonstrated the value of formalizing IT security in designing precise and relevant goals for improving the security of organizations. This contribution complements existing studies on IT security and zero trust, by providing a framework for positioning architectures relative to their compliance with the zero trust paradigm, as well as deepening the understanding of zero trust and IT security.

To conclude, the approach proposed in this thesis helps bridge the gap between existing zero trust architectures and the zero trust model. By extending the definition of IT security to better suit with real-world threats, this thesis helps organizations in developing strategies for improving their security posture, predicting and mitigating threats, and assessing precisely their maturity.

However, studies performed throughout this thesis have shown that, even though zero trust enhances the security and visibility within organizations, it also introduces additional threats. The numerous additional products and technologies needed to reach higher levels of zero trust maturity increase the attack surface and dependencies in potentially uncontrolled products. Furthermore, performing zero trust without initial trust – or trust in administrators, or in the system itself – is not feasible, as a root-of-trust is always required. The paradigm is called 'zero trust' because the system does not trust users – but conversely, users of the system need to implicitly trust the system and its administrators.

8.1 Perspectives

The evaluation framework developed in this thesis offers a tool for identifying potential research directions for enabling zero trust within architectures, therefore to enhance the security and privacy of individuals and organizations. More precisely, the following improvements can be considered:

- Chapter 2 has provided an overview of the fundamental principles of zero trust, as well as their potential benefits, limitations, and drawbacks. Investigating in more details the vulnerabilities brought by zero trust technologies, as well as the implications of transitioning to zero trust on users and administrators, would benefit the organizations undertaking a zero trust migration. Furthermore, whereas this thesis has focused on technical aspects of zero trust, research related to zero trust governance is scarce in the academic and industrial literature.

- Part III presents strategies for enhancing the zero trust capabilities of existing architectures through technology integration. Further research is necessary to reach higher levels of maturity. For example, Endpoint Detection and Response (EDR) and eXtended Detection and Response (XDR) have been developed to detect and analyze threats on devices. Research into improving EDR and XDR capabilities by integrating Artificial Intelligence (AI) techniques can enhance the precision of threat detection.

- The proof-of-concept presented in chapter 3 has demonstrated the challenges involved in building an architecture with high zero trust maturity, as multiple developments are required to integrate various technologies. Therefore, the establishment and wide-spread adoption of zero trust standards are essential for improving the interoperability of zero trust technologies, *e.g.*, the development – and adoption by the industry – of a standard zero trust data format.

- Part IV presents examples of limitations inherent to the zero trust framework. Research is needed to address the vulnerabilities introduced in zero trust architectures, *e.g.*, developing strategies to prevent attacks on single points of vulnerability, including the protection of the controller in the device agent/gateway model, and of the access proxy in the resource portal model.

Appendix A

Top-of-Rack-Assisted Load-Aware and Server-Agnostic Load-Balancing

While this thesis focuses on IT security, additional work was performed on how to improve network resources, which is presented in this appendix. While it is not directly relevant to the topic of this thesis, it enables to better understand constraints and requirements of networks, especially data centre networks.

Network load balancers optimize the use of computing resource in data centres, reducing response time and improving quality of service, by distributing requests among application instances running on servers. This chapter proposes ToRLB, a lightweight load balancing strategy that infers server occupancy to dispatch requests, without requiring cooperation from servers, computing per-packet statistics, or storing per-flow state.

ToRLB splits load balancing decisions between load balancers and between Top-of-Rack switches, which are on the path between load balancers and servers. This improves scalability, as load balancers remain server agnostic and perform simple operations. The performance of ToRLB is evaluated through extensive experiments on a simulation platform, demonstrating that it approaches the performance of state-of-the-art load balancing strategies, despite having limited information available.

A.1 Statement of Purpose

Cloud services expose data centres to substantial traffic volumes, with some single IP addresses handling over 100 Gbps [394]. To meet Service Level Agreements (SLA) and Quality-of-service (QoS) guarantees negotiated with cloud services consumers [395], [396], applications are replicated and distributed on several servers within data centres, using virtualization and containerization for providing scalable services [260].

Several data centre switching fabrics architectures have been designed to achieve high throughput, while reducing costs for large scale deployments. Among these, Clos networks [397], *e.g.*, fat-tree [393] or VL2 [398], are hierarchical architectures composed of four stages, depicted in figure A.1. Nodes

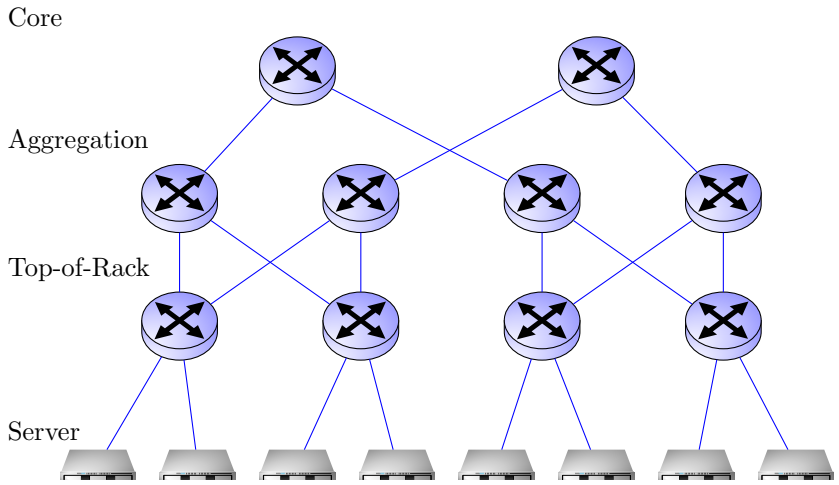


Figure A.1: Example Clos-based data centre fabrics architecture.

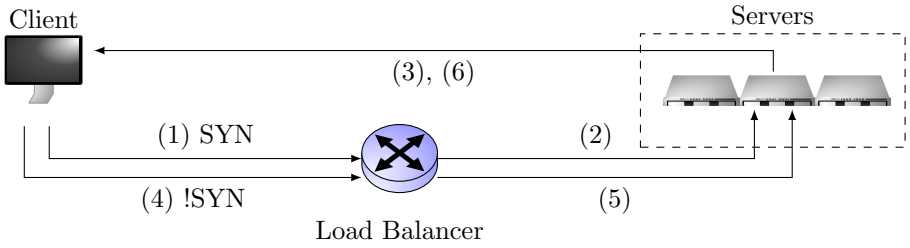


Figure A.2: Workflow of a network load balancer.

in the lowest tier are servers, which are grouped into *racks*. Each server in a rack is connected to a *Top-of-Rack* (ToR) switch, part of the *Edge* layer. ToR switches are in turn connected to one or more switches, part of the *Aggregation* layer. Finally, *Core* switches connect aggregation switches.

When a client initiates a TCP request[1] for an application hosted in a data centre, it sends a SYN packet with a destination IP address that uniquely identifies the application, referred to as the *Virtual IP address* (VIP). This packet is forwarded to a *Load Balancer* (LB), which selects a server responsible for processing the request [400]. Servers are identified within the data centre by their *Direct IP addresses* (DIP).

LBs are required to ensure *Per-Connection Consistency* (PCC) [401], *i.e.*, ensuring that packets from the same flow are forwarded to the same server, even if the number of servers or of LBs changes.

The workflow of an LB is depicted in figure A.2. Upon receiving a TCP SYN packet (arrow 1), the LB selects a server to dispatch the flow to (arrow 2). The chosen server responds using Direct Server Return (DSR) mode [400], bypassing the LB (arrow 3). Subsequent packets in the flow directed to the VIP

[1]Load balancing protocols are presented in this chapter for TCP connections, but are also applicable to connection-oriented transport protocols, *e.g.*, QUIC [399].

pass through the LB (arrow 4), and are forwarded towards the same application instance (arrow 5), ensuring PCC. Answers from the server are also returned using DSR mode (arrow 6).

Additionally, LBs aim to optimize *fairness*, *i.e.*, balancing flows on servers to prevent under- or over-loading of provisioned resources, thereby optimizing QoS [402], [403].

A.1.1 Related Work

One approach for load balancing is Equal-Cost Multi-Path (ECMP) [404], which distributes flows evenly on servers.

Several load balancing solutions, *e.g.*, Rubik [405], Silkroad [406], and Duet [407], employ ECMP as their load balancing strategy. To ensure PCC, these solutions store per-flow state in each LB, mapping each flow identifier (*e.g.*, the TCP 5-tuple, composed of source and destination IP addresses, of source and destination ports, and of the protocol number) to the DIP of the server that handles the flow. However, when multiple LBs are used simultaneously, this state needs to be replicated across all LBs to ensure PCC.

To avoid state replication and synchronisation between LBs, Ananta [394] and Maglev [408] use hashing mechanisms to preserve PCC. Upon receiving a TCP SYN packet, each LB hashes the TCP 5-tuple of the packet and divides the result by the number of servers hosting an application instance. The remainder of that division identifies the server that will handle the flow. As subsequent packets in the flow have the same TCP 5-tuple as the SYN packet, performing the same operation ensures that any LB sends packets to the same server, guaranteeing PCC without per-flow state, even if several LBs are in use. However, this approach tries to approach, but does not guarantee PCC if the number of servers changes, as the remainder of the division also changes. Alternative hashing mechanisms, *e.g.*, consistent hashing [409], reduce the number of flows that are closed when the number of servers changes.

Alternatively, Faild [410] and Beamer [411] use *daisy chaining* [412] to further reduce the number of reset connections when the number of servers changes. This approach involves replicating application state for every flow across two servers, with packets in the flow visiting each server in sequence (the daisy chain). Thus, if a server fails, the other takes over the connection.

While ECMP distributes flows uniformly to servers, it does not account for heterogeneous servers, *e.g.*, in terms of computing capacity. In such cases, ECMP creates an imbalanced distribution of flows on servers, thus not optimizing fairness [408]. Therefore, Ananta [394], Maglev [408], Concury [413], and Prism [414] rely on Weighted-Cost Multi-Path (WCMP) [415] instead. In WCMP, weights are statically assigned to servers based on their computing capacities. However, as weights are static, they can diverge from actual computing capacities if these change, resulting in suboptimal fairness [408].

To overcome weights divergence, Spotlight [402], LBAS [416], and Yoda [417] actively probe servers to estimate their computing capacities in real time. This load balancing strategy, referred to as Active WCMP (AWCMP), induces additional traffic in the data centre, and requires server modifications for signalling their computing capacities.

To remove the need for active signalling, INCAB [418] and HLB [403] passively observe traffic, and leverage statistical analysis to infer the computing

capacities of servers – the former using Bloom filters [419], the latter using Kalman filters [420].

WCMP, AWCMP, and statistics-based strategies are stateful, requiring per-flow state to be stored into LBs for ensuring PCC. This impacts the throughput of LBs [421], and reduces their resilience against Denial-of-Service (DoS) attacks, which exhaust LB memory [411].

Instead of storing flow mappings into LBs, CHEETAH [401] stores servers identifiers into packets, by adding 'cookies' to packets. These cookies are stored in TCP Timestamps [422] or QUIC header `connection-id` fields [399], which are mirrored by clients, establishing a covert channel between servers and LBs. This enables LBs to extract the information stored into the cookie, and to forward packets to the correct server.

Similarly, SHELL [423] and Charon [424] leverage this covert channel for stateless load balancing. Instead of relying on LBs to estimate computing capacities, SHELL [423] and 6LB [425] offload the decision-making process to servers by leveraging the power of two choices [426]: LBs select a set of candidate servers for each flow (*e.g.*, with consistent hashing), which are daisy chained using IPv6 Segment Routing (SR) [427]. When a server receives a SYN packet, it decides whether to accept the flow or to forward it to the next server in the chain, based on its computing capacity and current load.

A.1.2 Statement of Purpose

In SHELL and 6LB, all servers must be modified to support load balancing: they need to be SR-capable, and to monitor their load. Moreover, they cannot be turned off, as they are required to participate in the load balancing protocol. Additionally, their acceptance policy, which decides whether an incoming flow is handled by the server or sent to the next server in the chain, may not always make optimal decisions: just because a server is able to process a request does not mean that it is the best one for doing so.

This chapter proposes ToRLB, a stateless and server-agnostic network load balancer. Unlike SHELL and 6LB, which offload load balancing decisions to servers, ToRLB delegates part of the decision-making process to ToR switches. This enables the use of estimated server loads without modifying the servers themselves, as well as allowing servers to be shut down when needed. By using estimated server loads, ToRLB ensures fairer load balancing, and saves resources in the data centre by requiring fewer active servers. Additionally, ToRLB enables incremental deployability, by being able to operate seamlessly in a heterogeneous data centre that combines regular ToR switches with ToRLB ToR switches.

Figure A.3 depicts the key components of the ToRLB architecture. Clients initiate requests that are sent to a data centre with a Clos network topology. Requests are forwarded to LBs, at the transport layer. In ToRLB, modified ToR switches infer server loads, based on passive observations of network flows, without requiring explicit signalling from servers. When a new flow is received by an LB, the LB selects a rack to which the flow is forwarded. The ToR switch of that rack then chooses a server to which the flow is sent. PCC is ensured through a covert channel in TCP packets, similar to the solution proposed in Charon [424].

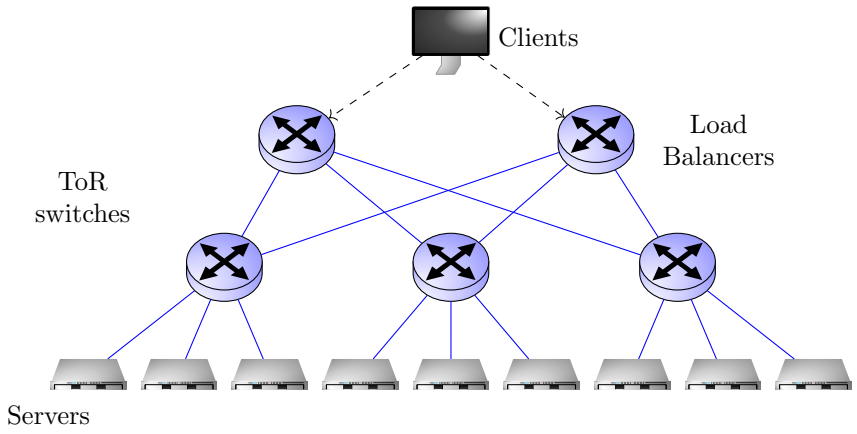


Figure A.3: Data centre architecture considered in this chapter.

The fairness of ToRLB is evaluated and compared to other load balancing strategies through simulations. The chapter also discusses the impact of the additional requirements for ToR switches on the overall system.

A.1.3 Chapter Outline

The remainder of this paper is organized as follows. Section A.2 provides an overview of the proposed load balancing strategy, ToRLB. Sections A.3 and A.4 delve deeper into the key components of ToRLB, LBs and ToR switches. Several policies are considered for both LBs and ToR switches, each with distinct requirements. Section A.5 describes the simulation environment used to compare ToRLB policies, and to compare ToRLB with other load balancing strategies. It also provides an analysis of the fairness offered by ToRLB. Finally, section A.6 concludes this chapter.

A.2 Overview of ToRLB

ToRLB is a load balancing system with two main components: load balancers (LBs), and Top-of-Rack (ToR) switches. When an incoming flow is forwarded to an LB, the decision on which server to assign the request for processing is distributed between the LB and ToR switches.

All traffic entering or exiting a rack passes through its corresponding ToR switch, allowing ToR switches to passively monitor traffic, for estimating the load of each server within their rack.

A.2.1 ToRLB Workflow

Figure A.4 depicts the workflow of a TCP flow handled by a data centre. The process begins when a SYN packet is forwarded to an LB (arrow 1). The LB selects a rack based on its load balancing policy, as described in section A.3.1. In this example, the LB chooses rack 2, and the SYN packet is forwarded to the ToR switch of rack 2 (arrow 2).

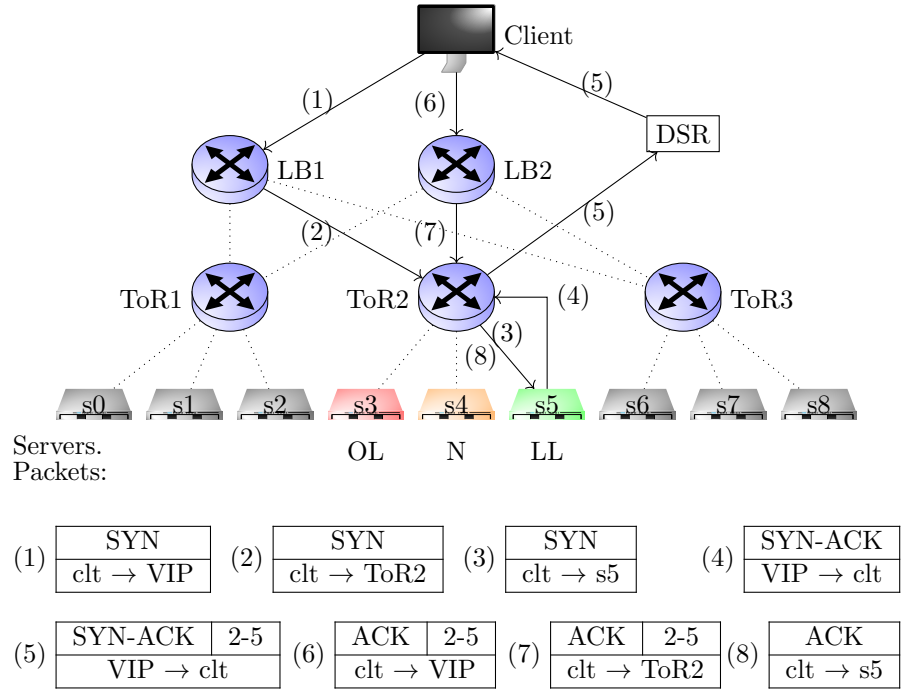


Figure A.4: Workflow of ToRLB.

Upon receiving the SYN packet, the ToR switch chooses a server within the rack for processing the flow, as detailed in section A.4.1. In this example, ToR 2 forwards the flow to server 5, because it is the least loaded (arrow 3).

Following the TCP state machine, the application responds to the SYN packet with a SYN-ACK packet. ToR 2, being connected to server 5, intercepts the SYN-ACK packet (arrow 4), and inserts, into a covert channel in the packet, the index of the rack and a local identifier for the chosen server. The procedure is similar to methods presented in [401], [423], and is described in section A.4.2. The SYN-ACK is then sent to the client, using DSR mode (arrow 5).

The value inserted by the ToR switch is mirrored by the client in the ACK packet (arrow 6). When an LB – which may be distinct from the LB having received the SYN packet, *e.g.*, if the set of LBs changes and packets are distributed to LBs using consistent hashing – receives the ACK packet, it extracts the index of the rack from the covert channel, and forwards the packet to this rack (arrow 7). Upon receiving the packet, the ToR switch extracts, from the covert channel, the identifier for the server, and forwards the packet to the designated server (arrow 8).

Subsequent packets in the flow follow the same path as the SYN-ACK and ACK packets (arrows 4 to 8).

Table A.1: LB policies in ToRLB.

Policy	Description
ECMP	Select a rack randomly following a uniform distribution.
LSQ	Record the number of active flows in each rack, and select the least loaded.
AWCMP	Select a rack using weights, given by ToR switches.
WLSQ	Combination of LSQ and AWCMP.

A.3 ToRLB Load Balancer

The ToRLB Load Balancer needs to perform two tasks: selecting a suitable rack for incoming flows, and ensuring PCC for ongoing connections.

A.3.1 Load Balancer Policies

When a SYN packet arrives at the data centre, the LB selects a rack to forward the packet to. This section explores four policies for making this selection, summarized in table A.1.

The first policy is a random selection of a rack with uniform probability, similar to an ECMP strategy, *e.g.*, using consistent hashing on racks. This policy is the easiest to configure, and does not require additional state to be saved on LBs.

The second policy is Local Shortest Queue (LSQ), in which LBs keeps track of the number of active flows they have forwarded to each rack, by maintaining one counter per rack. This additional state is of the same magnitude as state already used by LBs for forwarding. When a SYN packet arrives at an LB, the LB forwards the packet to the rack with lowest counter value, and increments the corresponding counter. When a TCP connection closes, observed with RST or FIN packets, the counter is decremented. As LBs need to monitor the start and end of each connection, LSQ is only relevant if all packets in a flow are forwarded to the same LB, *e.g.*, if packets are forwarded to LBs using consistent hashing.

The third policy is based on AWCMP. Weights correspond to the number of servers in each rack. This information is known by ToR switches, as they are directly connected to servers in the rack. Thus, if a server is shut down, or is brought up, the ToR switch is aware of this change. In ToRLB, weights are updated passively, by including server counts in the covert channel of each packet leaving a rack, along with PCC information. LBs retrieve this information when packets arrive at the data centre. This method produces less precise weights than traditional AWCMP, but does not require active probing from LBs.

The fourth policy is Weighted LSQ (WLSQ), which combines LSQ and AWCMP. LBs use both rack occupancy from LSQ and rack computing capacities from AWCMP, by selecting the rack with the lowest ratio occupancy over computing capacity.

Table A.2: ToR switch policies in ToRLB.

Policy	Description
ECMP	Select a rack randomly following a uniform distribution.
WCMP	Select a rack randomly following manually configured weights.
LSQ	Record the number of active flows in each server, and select the least loaded.
WLSQ	Combines WCMP and LSQ.

A.3.2 Maintaining Per-Connection Consistency

If the policy is ECMP, consistent hashing enables PCC to be maintained if the set of racks remains unchanged. To maintain PCC for other policies, each rack is assigned a unique 1-byte identifier, stored on the ToR switch associated with that rack. When a packet leaves a rack, the ToR switch inserts into the corresponding rack identifier into the covert channel of the packet. When a non-SYN packet from a flow reaches an LB, the LB extracts the rack identifier from the covert channel, and forwards the packet towards the corresponding rack. The nature of the covert channel is explained in more details in section A.4.2.

A.3.3 Incremental Deployability

Modifying ToR switches is required for ToRLB to maintain PCC and to select servers. However, ToRLB can still function if only a subset of racks have ToRLB-enabled ToR switches. In this case, LBs need to be aware of servers in racks with standard ToR switches. LBs then act as virtual ToR switches for those servers, applying both the LB and the ToR switch policies.

To ensure PCC, DSR mode is disabled for these servers. This enables LBs to include the server identifiers into the covert channel of packets.

A.4 ToRLB Top-of-Rack Switches

In the ToRLB architecture, ToR switches serve as the second layer of decision-making. When a ToR switch receives a SYN packet forwarded by an LB, its task is to select a target server for the flow.

A.4.1 ToR switches Policy

Similarly to LBs, ToR switches use policies to select which servers flows are forwarded to. Four ToR switches policies are considered in this chapter, summarized in table A.2.

The simplest policy for ToR switches is ECMP, where the ToR switch randomly selects a server in the rack. If servers have heterogeneous computing capacity, WCMP can be used instead with manually configured weights.

As all traffic flowing into and out of servers within a rack passes through the corresponding ToR switch, ToR switches can monitor this traffic. This enables for an LSQ policy, tracking active connections into servers in the rack. Additionally, computing capacities may be manually configured in ToR switches,

enabling a WLSQ policy that tracks the number of active connections in every server while being aware of their computing capacities.

A.4.2 Maintaining Per-Connection Consistency

To maintain PCC, ToR switches assign 1-byte local identifiers to servers in their rack, and store the mapping between server and identifier. When a packet leaves a server in a rack, it is forwarded to the ToR switch of that rack. The ToR switch then inserts the rack identifier and the server identifier into the covert channel of the packet

The covert channel follows the method described in [401], [423], where identifiers are inserted into TCP timestamps. Applications in the data centre are configured to produce timestamp values with the same 16 most significant bits. When a packet leaves a server, the ToR switch replaces the 16 most significant bits of the `TSval` field with the rack and server identifiers. This value is mirrored into the `TSecr` field of packets sent to the data centre by the client.

When a subsequent packet in the flow reaches the data centre, the LB extracts the rack identifier from the `TSecr` field and forwards the packet to the corresponding rack. Then, the ToR switch extracts the server identifier to forward the packet towards the correct server. The ToR switch also replaces the 16 most significant bits in the `TSecr` field with the common value configured for every application.

This covert channel is only viable for up to 2^8 racks and 2^8 servers per rack. As proposed by [401], QUIC `connection-id` may be used instead, which enables 160 bits of covert channel. If the LB policy is AWCMP, the size of identifiers needs to be reduced, to accommodate for weights to be inserted into the covert channel, *e.g.*, with 4 bits for the rack identifier, and 6 bits each for the server identifier and the weight. As remarked by [401], storing identifiers directly into packets makes the data centre more vulnerable to denial-of-service attacks. Thus, [401] proposes to obfuscate them, using a process similar to hash-based Message Authentication Codes (HMAC), which requires additional computation per packet.

A.4.3 Overhead of ToR switches

ToRLB introduces overhead for ToR switches in several ways. First, ToR switches require additional memory to store rack and server identifiers, as well as the number of active connections. This additional memory is comparable to the state already stored in ToR switches, as each ToR switch has one entry per server in its forwarding table.

Second, ToR switches need to perform additional processing operations on every packet that passes through them. This includes inspecting packets for detecting SYN, FIN and RST packets, applying the ToRLB policy to select servers for SYN packets, and inserting or extracting identifiers from the covert channel for other packets. These additional operations create computational overhead.

Additionally, if the policy is based on LSQ, finding the least loaded server in the rack for each SYN packet requires an integer comparison for every server in the rack, as well as incrementing a counter. If a FIN or RST packet passes

through the ToR switch, a counter needs to be decremented. Moreover, rewriting the TCP timestamps for every TCP packet passing through a ToR switch requires recomputing the TCP checksum of the packet. These line-rate computations can negatively impact the latency and throughput of ToR switches.

A.5 Evaluating Fairness and Performance of Load Balancing Strategies

A test environment has been created to simulate data centre communications, enabling the comparison of different load balancing strategies in terms of fairness and performance. Fairness is measured as the distribution of load in servers, and performance is measured as the average response time of requests.

A.5.1 Testing Environment

The testing environment uses a discrete event-based network simulator, to simulate data centres as depicted in figure A.3. The testing environment can be found on GitHub[2]. In these simulations, only one application in the data centre is considered. The application is CPU intensive: requests are composed of a single data packet that requires the application to perform computations, using processor resources, before sending the response.

Client

Clients of the application are modeled in the simulator by a single client, that generates a stream of TCP requests to the application. This stream is represented by a Poisson Pareto Burst Process [428], which models Internet traffic: incoming flows follow a Poisson distribution with parameter λ, and each request requires a processing time following a long-tail Pareto distribution. The λ parameter is normalized, and represents the rate of requests, with $\lambda > 1$ representing an overwhelming request rate that exceeds the capacity of the data centre, even under optimal load balancing.

For each request, the client measures the time necessary to receive a response from the application. The client also has a timeout setting: if no response is received within this timeframe, the client sends an RST packet and discards the flow. At the end of each simulation, the client processes the response times for requests, and returns their distribution. Requests without responses are counted with a time `timeout`.

In the simulations presented in this chapter, the client continuously sends requests for 11s. Because the data centre starts without any request being processed, requests sent during the first second are not taken into account for measuring performance, and initialize the data centre. The Pareto policy has a mean of $350 \cdot 10^9$ cycles (*i.e.*, requests require on average $350 \cdot 10^9$ CPU cycles to complete), with a Hurst parameter of 0.7, as proposed in [429]. The `timeout` value is set to 40s.

[2] https://github.com/apoirrier/sr-tor

Servers

Servers in the simulator accept incoming connections, and process them. Each server is composed of multiple CPU threads, each with a processing speed of 1 GHz. Requests are distributed across these threads, and if m requests are processed simultaneously on a single thread, the processing speed for each request is divided by m.

Servers have a limited capacity to process requests concurrently, defined by the configuration variable `max_connections`. If the limit is reached, incoming requests are added to a backlog queue of length `backlog_length`. If the backlog queue is full, the request is discarded and an RST packet is sent back to the client. In the simulations presented in this chapter, `max_connections` is set to 8 times the number of threads, and `backlog_length` is set to 32 times the number of threads.

The results presented in the following sections are averaged across at least 60 simulations. The graphs depict the average values obtained from these simulations, with the *standard error* displayed on the graphs.

A.5.2 ToR Switches Policies in ToRLB

As presented in sections A.3 and A.4, ToRLB is defined by two load balancing policies: one performed by LBs, the other performed by ToR switches. This section studies the impact of ToR switches policies on performance.

ToR switches only influence the distribution of requests within their respective rack. Therefore, the impact of ToR switches policies is evaluated in data centres containing a single rack.

Three data centres are considered, to evaluate whether the distribution of servers within the rack impacts performance and fairness. Each data centre contains two types of servers: 4-threads servers and 8-threads servers. The computing capacity in all three data centres is identical, with 168 threads available in the rack. However, their distributions differ: data centre A consists of 42 servers with 4 threads each, data centre B has 20 servers with 4 threads each, as well as 11 servers with 8 threads each, and data centre C has 21 servers with 8 threads each.

Performance Evaluation

Performance results are depicted in figure A.5, comparing the ToR switches policies listed in table A.2. The horizontal axis represents the normalized request rate, λ, and the vertical axis represents the average response time of requests on a logarithmic scale.

The solid curves represent the overall performance with ECMP policy, dashed curves with WCMP, curves with dashes and dots with LSQ, and dotted curves with WLSQ. Green curves (with crosses) are for data centre A, blue curves (with squares) for data centre B, and yellow curves (with triangles) for data centre C.

Each curve is increasing: as the request rate increases, the average processing time also increases, starting from 350ms (the mean of the Pareto law).

Unsurprisingly, WLSQ demonstrates the best performance, since it is the load balancing policy where ToR switches have the most information, including

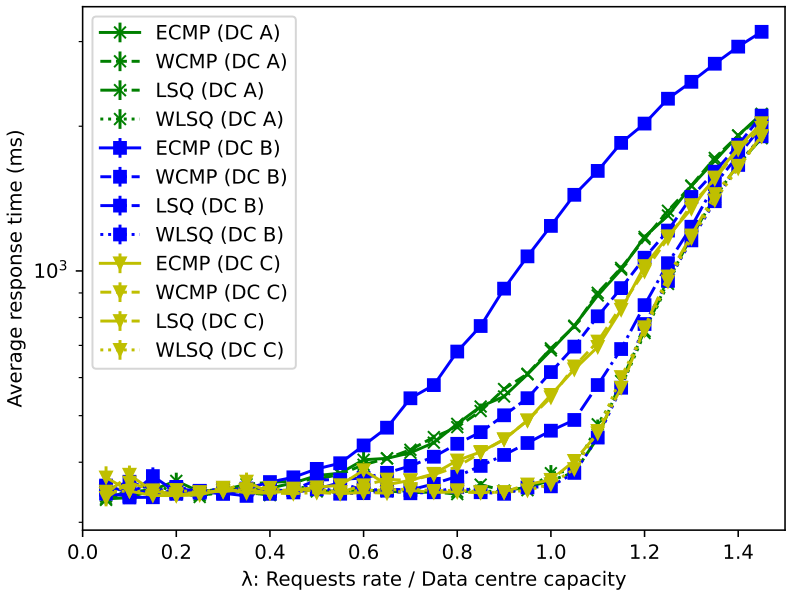


Figure A.5: Performance comparison of ToRLB ToR switches policies, in one rack data centres (DC). DC A has 42 servers with 4 threads, DC B has 20 servers with 4 threads and 11 servers with 8 threads, and DC C has 21 servers with 8 threads. Data centres are represented by the color and shape of points. Policies are represented by the style of lines. Y axis is log scale, lower is better.

server occupation and computing capacity. The distribution of servers does not impact performance for WLSQ.

On homogeneous data centres, ECMP and WCMP have the same performance, as do LSQ and WLSQ. This is expected because identical servers imply equal weights. However, in the heterogeneous data centre, ECMP performs poorly due to its lack of differentiation between 8-threads and 4-threads servers, resulting in under-loading the former and overloading the latter. LSQ does not achieve the same level of performance as WLSQ, which demonstrates the importance of weights in making good decisions. LSQ performs better than WCMP, demonstrating that pre-configured weights are insufficient to optimize the request distribution onto servers, and an adaptative strategy is required.

The performance of WCMP depends on the number of servers: the lower the number of servers, the better the performance. This can be attributed to servers optimizing request processing by scheduling them on threads. Therefore, servers with more threads process more queries simultaneously, balancing queries more efficiently.

Performance is Explained by Fairness

Figure A.6 depicts, averaged across 60 simulations, thread occupancy over time, in data centre B, with the ToR switch policy LSQ, for various values of

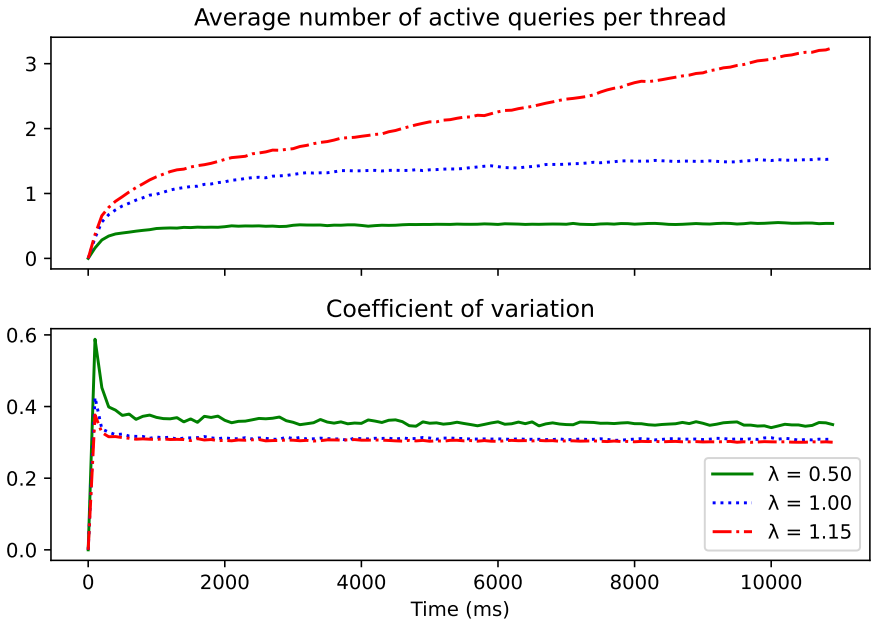


Figure A.6: Active requests per thread, with LSQ as ToR switches policy, in data centre B.

λ. The top graph represents, over time, the average number of active requests processed by each thread. The graph below represents, over time, the coefficient of variation of threads occupancy over time, computed as the standard deviation divided by the mean.

If $\lambda > 1$, the number of active requests diverges, as the number of requests exceeds the capacity of the data centre. Conversely, if $\lambda \leq 1$, it converges. This validates *a posteriori* the simulation time of 11s and waiting period of 1s, as these enable to reach a stationary state and to remove the transition period.

The coefficient of variation represents how spread requests are distributed among threads: larger variations imply a higher coefficient of variation. This value also stabilizes over time.

Figure A.7 depicts the stabilized values of average thread occupancy and coefficient of variation depending on λ, for the three considered data centres, and for all ToR switches policies. The coefficient of variation, which measures fairness, is directly correlated with the performance depicted in figure A.5. This is due to a causal link between fairness and performance: when requests are dispatched more fairly among servers, the overall processing time decreases.

Analysis

The results from this section demonstrate a direct correlation between performance, measured by average response time of requests, and fairness, defined as the even distribution of requests on threads within the data centre.

In heterogeneous data centres, providing weights to ToR switches yields

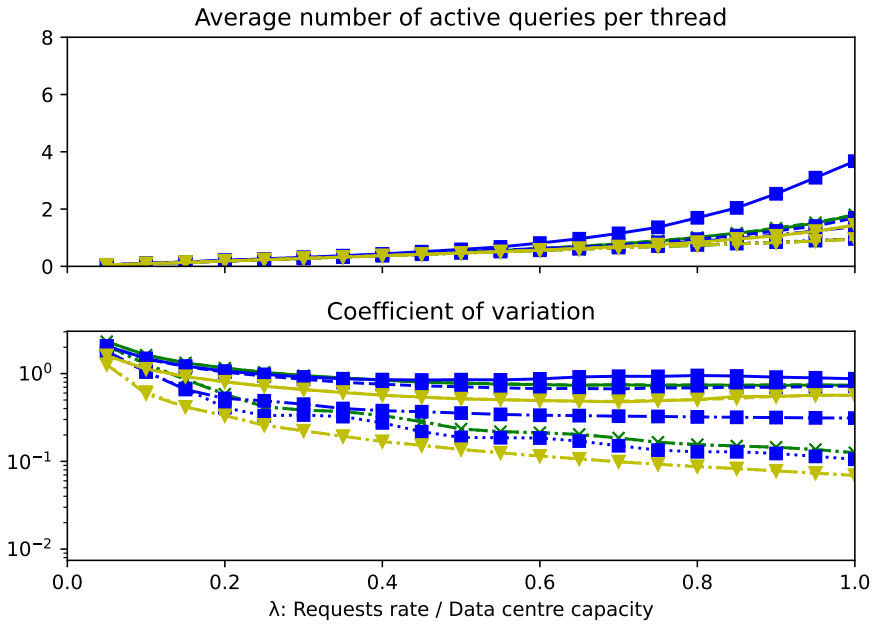


Figure A.7: Fairness comparison of ToRLB ToR switches policies. Lower coefficient of variation means better fairness. Legend is the same as in figure A.5.

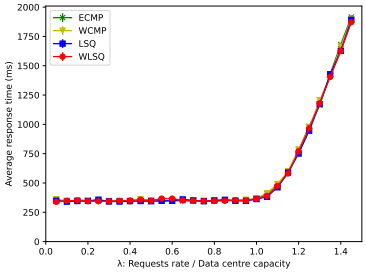
the best results in terms of performance. However, this approach requires manual configurations, which can be error-prone, and limit scalability. Given that LSQ only offers a moderate improvement over WLSQ, and considering its ease of deployment, the recommended policy for ToR switches is LSQ. Unless otherwise stated, the remainder of this chapter assumes that LSQ is the chosen ToR switch policy in ToRLB.

A.5.3 LB Policies in ToRLB

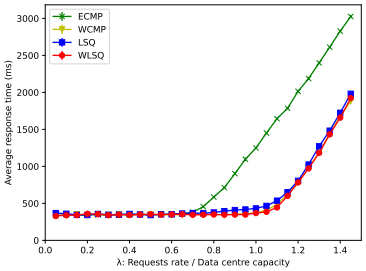
This section examines how LB policies impact the overall performance of ToRLB. The impact on performance may depend on the distribution of servers in racks, so LB policies are compared across three data centres. Each data centre consists of two racks, with a total of 42 identical servers, each with 4 threads. The only difference between data centres is the server distribution within the racks: data centre α is a balanced data centre, with each rack containing 21 servers, data centre β has one rack with 14 servers and a second rack with 28 servers, and data centre γ has one rack with 2 servers and the other rack with 40 servers.

Performance Evaluation

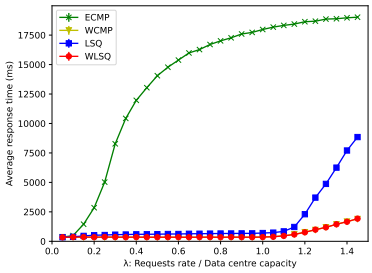
Figure A.8 compares the performance of ToRLB LB policies, listed in table A.1. The impact of LB policies becomes more significant as the imbalance between racks in the data centre increases: in the balanced data centre, each LB policy



(a) 2 racks of 21 servers each.



(b) 2 racks with 14 and 28 servers respectively.



(c) 2 racks with 2 and 40 servers respectively.

Figure A.8: Performance comparison of LB policies in 2 racks data centres. Lower is better.

produces an evenly balanced distribution of requests across each rack, resulting in no difference in performance, as depicted in figure A.8a. For unbalanced data centres, ECMP distributes requests evenly, leading to the worst performance since one rack is overloaded and the other under-loaded. The difference between LSQ and WLSQ is negligible: distributing requests proportionally to the computing capacities of racks is the most crucial parameter for optimizing performance.

Figure A.9 presents the distribution of response times to requests in data centre γ for $\lambda = 1.2$. The red (plain) line is the reference, and represents the Pareto law, which is the distribution of processing times if each query was processed alone on threads. Although the curves appear not to reach the value 1 in the graph, they do so at the right edge of the graph, which is the value of the timeout for the client. This means that the value reached just before the timeout represents the proportion of requests that have received a response.

The ECMP (green, dashed) and LSQ (blue, dotted) lines show a spike in requests around 10s processing time. These are the requests processed in the rack with only 2 servers, while the remainder are processed in the other rack. Response time distributions follow a scaled version of the Pareto law, due to each thread processing multiple requests simultaneously, which multiplies the processing time of each request.

This distribution explains the evolution of the LSQ and WCMP curves on

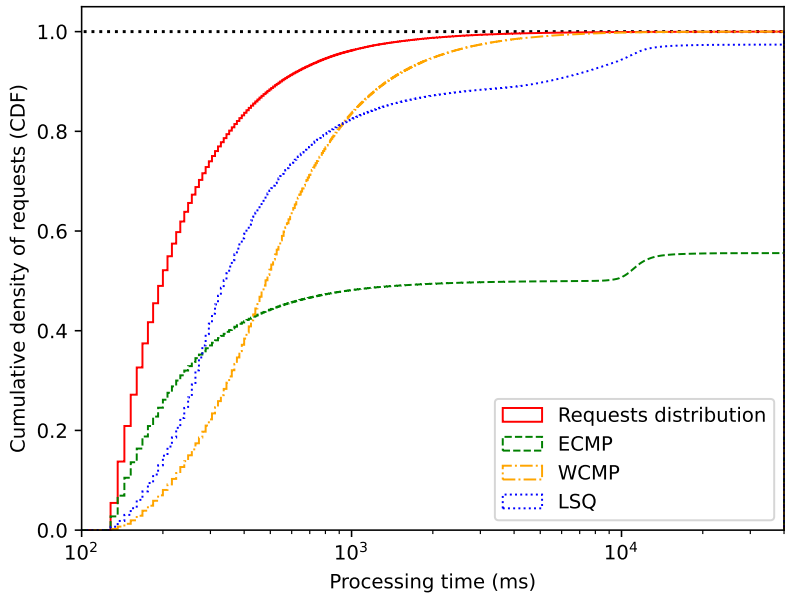


Figure A.9: Distribution of response times, for $\lambda = 1.2$, in the datacenter having racks with 2 and 40 servers respectively.

figure A.8. Processing time linearly increases in a first step for $\lambda \leq 1$, as the average number of requests per threads increases. This is represented in figure A.9 by curves moving from the Pareto distribution 'to the right'. Then, a second, quicker, linear increase happens, with $\lambda > 1$: this represents more and more requests timing out and being discarded. This is represented in figure A.9 by curves moving 'lower', with the proportion of answered requests dropping.

Increasing the Number of Racks and Load Balancers

Increasing the number of LBs does not impact performance when requests are distributed uniformly across LBs. This is because each LB receives the same distribution of requests, and thus makes similar decisions as a result. For randomness-based policies (ECMP and WCMP), using multiple LBs requires LBs to have de-correlated randomness. Experimental results presented in figure A.10 validate this statement: the number of LBs does not impact performance.

The behaviors described in figure A.9 are similarly observed when there are more than two racks, with the same explanations applying.

Analysis

This section has demonstrated that optimal performance is achieved when requests are balanced across racks in proportion to their computing capacities. The best policy for this is AWCMP, which eliminates the need for LBs to compute statistics on traffic.

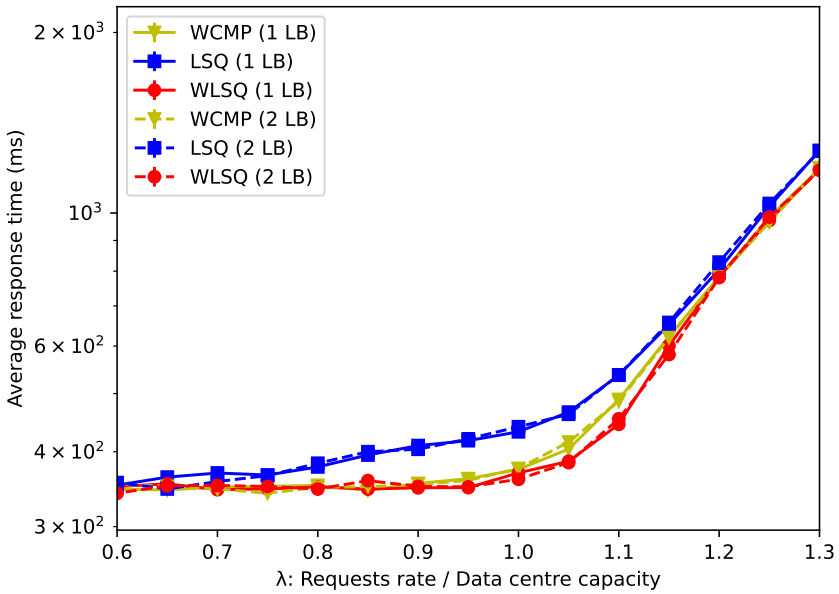


Figure A.10: Comparison of ToRLB policies with one LB or 2 LBs, in a 2 racks data centre.

Therefore, the optimal policies for ToRLB are AWCMP for LBs, and LSQ for ToR switches. The main role of LBs is to distribute queries into racks proportionally to their computing capacities. Any further imbalance, *e.g.*, caused by requests variations, is handled by ToR switches, which distribute requests among servers in each rack by tracking active connections on each server.

A.5.4 Comparing ToRLB to Reference Load Balancing Strategies

This section compares ToRLB with reference load balancing strategies, that do not use ToR switches for load balancing. The reference load balancing strategies used for comparison are as follows:

- *ECMP*: LBs select a server uniformly at random;
- *LSQ*: LBs track the number of active requests in each server, and select the least loaded;
- *Shortest Expected Delay (SED)*: LBs are aware of server loads, computing capacity, and processing times of flows in servers.

Simulations were performed on a data centre consisting of 2 LBs and 42 homogeneous 4-threads servers, unevenly distributed across 6 racks.

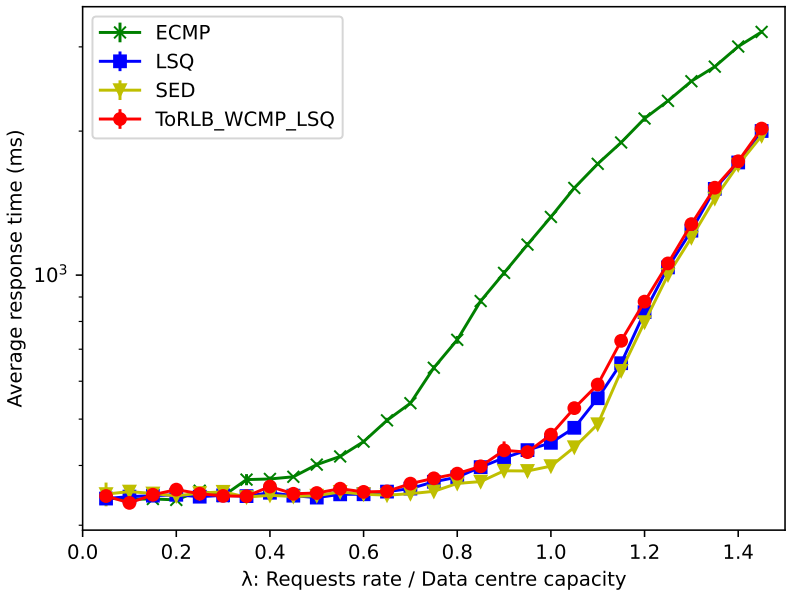


Figure A.11: Comparison of ToRLB with reference load balancing strategies.

Performance Evaluation

Figure A.11 presents a performance comparison between ToRLB, with AWCMP as LB policy and LSQ as ToR switch policy (referred to as ToRLB_WCMP_LSQ), and reference load balancing strategies.

It is relevant to compare ToRLB_WCMP_LSQ and ECMP – instead of WCMP – because weights in ToRLB_WCMP_LSQ correspond to the number of servers in each rack, whereas weights in WCMP as an overall load balancing strategy would correspond to the computing capacity of servers, which are all identical in this simulation.

The results depicted in figure A.11 show that ECMP has the worst performance. Even though all servers have the same computing capacity, requests are not identical, leading to an imbalance in request processing, and thus resulting in some servers being overloaded while others are under-loaded.

LSQ and SED adapt to this imbalance, demonstrating better performance. As SED has access to more information than LSQ, it performs better.

ToRLB has an intermediate performance, closely matching the performance of LSQ. It is expected that ToRLB would have worse performance than SED, as it has access to less information. The use of ToRLB, instead of LSQ, enables to spread the computations to ToR switches, improving scalability.

A.6 Summary

This chapter has introduced ToRLB, a lightweight, network-layer load balancer that offloads part of the load balancing process to ToR switches. By doing so,

ToRLB enables lightweight LBs that only need to perform WCMP for incoming flows, and to inspect subsequent packets for ensuring PCC. Additionally, servers are not part of the load balancing process, allowing them to be dynamically halted or activated as needed, thereby enabling dynamic adjustment of the number of active servers to meet demand and reducing costs. ToRLB also offers increased flexibility, as servers do not require special modifications. Furthermore, ToRLB enables incremental deployability of ToRLB ToR switches. While ToRLB may not achieve the same level of performance as stateful network load balancers like HLB, it approaches the performance of an LSQ-based strategy with less information available, and a more lightweight load balancing system.

Appendix B

Résumé en Français

Cette thèse étudie l'utilisation d'architectures de sécurité pour protéger les systèmes d'information contre les menaces informatiques. Elle examine en particulier le concept de zéro confiance, et se pose la question de comment ce paradigme peut être intégré au sein d'architectures existantes afin de renforcer leur sécurité. De plus, cette thèse explore comment étendre le cadre zéro confiance afin qu'il réponde au mieux aux menaces réelles. Ce manuscrit comprend cinq parties et neuf chapitres, structurés comme suit.

La partie I est une mise en contexte introductive. Dans le chapitre 1 (soumis comme partie de [430]), il est présenté le contexte historique qui a mené à l'émergence du paradigme de zéro confiance. Le chapitre explore les défis liés à la définition de zéro confiance et ses potentiels avantages en ce qui concerne la sécurisation des systèmes d'information.

La partie II offre une analyse approfondie du concept de zéro confiance. Le chapitre 2 (soumis également comme partie de [430]) présente une étude exhaustive des définitions et principes du paradigme zéro confiance, issus de publications universitaires, gouvernementales et industrielles. A partir ce cette analyse, une définition complète du paradigme zéro confiance est proposée. De plus, le chapitre présente une taxonomie des stratégies de migration, qui permettent l'ajout de capacités additionnelles pour sécuriser les systèmes d'information, et une taxonomie des multiples technologies qui peuvent être mises en oeuvre pour fournir ces capacités. Enfin, une étude d'un ensemble représentatif des architectures zéro confiance existantes offre une analyse plus approfondie du concept, et permet de proposer une méthode pour positionner ces architectures par rapport à leur adhésion aux principes zéro confiance afin de proposer des axes précis d'amélioration.

Le chapitre 3 (soumis comme partie de [431]) montre comment une architecture zéro confiance peut être construite en combinant et modifiant divers produits open-source, en proposant un démonstrateur. Grâce à la méthode développée au chapitre 2, la maturité zéro confiance du démonstrateur est évaluée. Ce démonstrateur est utilisé dans le reste de la thèse comme architecture de base, à laquelle peuvent être incorporées des améliorations.

L'analyse des architectures zéro confiances et des recherches existantes présentées dans la partie II révèle des lacunes dans l'état de l'art pour construire des architectures à haut niveau de maturité. La Partie III vise à combler certaines de ces lacunes, en fournissant des méthodes pour intégrer des technologies additionnelles au sein d'une architecture existante, afin d'améliorer son niveau de maturité zéro confiance. Une partie de ces améliorations est mise en œuvre au sein du démonstrateur présenté au chapitre 3, illustrant comment il est possible d'intégrer des technologies au sein d'une architecture existante, tout en assurant son interopérabilité avec les autres composants.

En particular, le chapitre 4 (soumis comme [432]) présente une méthode pour ajouter de la sécurité centrée sur les données à une architecture zéro confiance existante, en exploitant une méthode de chiffrement, le Chiffrement par Attributs, qui protège la confidentialité des données avec un contrôle d'accès basé sur les attributs, offrant une flexibilité supplémentaire aux propriétaires des données. La méthode proposée permet de stocker des données protégées par des politiques d'accès distinctes sur un seul serveur. Dans ce cas, même si le serveur est compromis, ou n'est pas de confiance, comme par exemple un fournisseur cloud, les données qui y sont stockées sont restent confidentielles.

Le chapitre 5 (présenté au symposium Real World Crypto[1] et publié dans [159]) étudie en détail une des propriétés fondamentales du zéro confiance, l'authentification continue, dans le contexte des protocoles de messagerie sécurisée. Dans ce chapitre, la procédure d'authentification continue du protocole de messagerie Signal est analysée en détails, puis étendue pour améliorer et automatiser la sécurité post-compromission, qui permet la re-sécurisation des sessions de communication après que les secrets d'un des membres ont été révélés.

La partie IV confronte le paradigme zéro confiance avec divers scénarios du monde réel. Elle montre que le paradigme seul n'est pas suffisant, et qu'il est nécessaire d'une part d'étendre les architectures afin d'obtenir des garanties de sécurité suffisantes, et d'autre part d'étendre le modèle de menace pour englober l'intégralité des menaces qui pèsent sur les systèmes d'information.

Dans le chapitre 6 (présenté à la conférence C&ESAR[2], publié dans [93], et soumis comme [431]), une méthode pour fédérer plusieurs architectures zéro confiance qui maintienne des niveaux équivalents de sécurité est proposée. Le principal défi est que, dans une fédération, la vérification de l'identité et la surveillance des entités voulant accéder à des données et services sont effectuées par le domaine d'origine de ces entités. Or, pour suivre les principes zéro confiance, quand une entité demande l'accès à une ressource fédérée, le domaine protégeant la ressource doit explicitement vérifier l'identité et estimer la sécurité de l'environment du demandeur. Comme ces informations sont recueillies et maîtrisées par le domaine du demandeur, la solution usuelle est de faire implicitement confiance au domaine du demandeur, et de ne pas vérifier explicitement les informations transmises, ce qui est contraire aux principes zéro confiance. Ceci peut mener à des attaques, en particulier si le domaine implicitement considéré comme de confiance a été compromis. Pour pallier à ce problème, ce chapitre propose une méthode qui permet au domaine de la ressource d'explicitement vérifier ces informations, grâce à la technologie

[1] https://rwc.iacr.org/2022/program.php
[2] https://2023.cesar-conference.org/

d'attestation à distance. Ceci permet une vérification explicite sans exiger de méthode intrusive qui ne serait pas applicable à des déploiements réels.

Le chapitre 7 (soumis comme [433]) étudie comment l'infrastructure sous-jacente, par exemple les réseaux, peut être considérée comme non digne de confiance. Plusieurs méthodes sont proposées pour permettre aux routeurs d'acheminer les packets vers leur destination, sans toutefois leur permettre d'obtenir des informations sur ces paquets, par exemple quelle sont leurs sources et destinations. La solution proposée ajoute un niveau supplémentaire de sécurité, en permettant l'anonymisation des communications en plus de la confidentialité et de l'intégrité.

Enfin, la partie V clôt ce manuscrit. L'annexe A présente une contribution supplémentaire réalisée au cours de cette thèse, montrant comment distribuer de manière optimale les ressources et les requêtes dans un centre de données, en équilibrant la charge du réseau.

List of Figures

List of Tables

List of Algorithms

Bibliography

Background References

[1] J. D. Howard, *An analysis of security incidents on the internet 1989-1995*. Carnegie Mellon University, 1997. [Online]. Available: https://insights.sei.cmu.edu/library/an-analysis-of-security-incidents-on-the-internet/.

[2] J. Saltzer and M. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975, ISSN: 0018-9219. DOI: 10.1109/proc.1975.9939.

[3] K. A. Scarfone and P Hoffman, "Guidelines on firewalls and firewall policy," National Institute of Standards and Technology, Tech. Rep. NIST SP 800-41-1, 2009. DOI: 10.6028/nist.sp.800-41r1.

[4] Industrial Control Systems Cyber Emergency Response Team (ICS-CERT), "Recommended practice: Improving industrial control system cybersecurity with defense-in-depth strategies," Department of Homeland Security, Tech. Rep., 2016. [Online]. Available: https://www.cisa.gov/uscert/sites/default/files/recommended_practices/NCCIC_ICS-CERT_Defense_in_Depth_2016_S508C.pdf.

[5] N. Mhaskar, M. Alabbad, and R. Khedri, "A formal approach to network segmentation," *Computers & Security*, vol. 103, p. 102 162, Apr. 2021. DOI: 10.1016/j.cose.2020.102162.

[6] C. Smith, "Understanding concepts in the defence in depth strategy," in *IEEE 37th Annual 2003 International Carnahan Conference onSecurity Technology, 2003. Proceedings.*, IEEE, 2003. DOI: 10.1109/ccst.2003.1297528.

[7] D. Puthal, S. P. Mohanty, P. Nanda, and U. Choppali, "Building security perimeters to protect network systems against cyber threats [future directions]," *IEEE Consumer Electronics Magazine*, vol. 6, no. 4, pp. 24–27, Oct. 2017. DOI: 10.1109/mce.2017.2714744.

[8] Z. R. Alashhab, M. Anbar, M. M. Singh, Y.-B. Leau, Z. A. Al-Sai, and S. A. Alhayja'a, "Impact of coronavirus pandemic crisis on technologies and cloud computing applications," *Journal of Electronic Science and Technology*, vol. 19, no. 1, p. 100 059, Mar. 2021. DOI: 10.1016/j.jnlest.2020.100059.

[9] P. Sun, "Security and privacy protection in cloud computing: Discussions and challenges," *Journal of Network and Computer Applications*, vol. 160, p. 102 642, Jun. 2020. DOI: 10.1016/j.jnca.2020.102642.

[10] S. Dhar, "From outsourcing to cloud computing: Evolution of it services," *Management Research Review*, vol. 35, no. 8, pp. 664–675, Jul. 2012, ISSN: 2040-8269. DOI: 10.1108/01409171211247677.

[11] A. F. Westin, "Privacy and freedom," *Washington and Lee Law Review*, vol. 25, no. 1, 1968. [Online]. Available: https://scholarlycommons.law.wlu.edu/cgi/viewcontent.cgi?article=3659&context=wlulr.

[12] M. Douriez, H. Doraiswamy, J. Freire, and C. T. Silva, "Anonymizing nyc taxi data: Does it matter?" In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, IEEE, Oct. 2016. DOI: 10.1109/dsaa.2016.21.

[13] C. Dwork, "Differential privacy: A survey of results," in *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2008, pp. 1–19. DOI: 10.1007/978-3-540-79228-4_1.

[14] J. P. Titus, "Security and privacy," *Communications of the ACM*, vol. 10, no. 6, pp. 379–381, Jun. 1967, ISSN: 1557-7317. DOI: 10.1145/363332.363421.

[15] J. K. Reynolds, *Helminthiasis of the Internet*, RFC 1135, Dec. 1989. DOI: 10.17487/RFC1135. [Online]. Available: https://www.rfc-editor.org/info/rfc1135.

[16] T. M. Chen and J.-M. Robert, "The evolution of viruses and worms," in *Statistical methods in computer security*, CRC press, 2004. [Online]. Available: https://ivanlef0u.fr/repo/madchat/vxdevl/papers/avers/statmethods2004.pdf.

[17] D. E. Bell and L. J. L. Padula, "Secure computer system: Unified exposition and MULTICS interpretation," The MITRE Corporation, Tech. Rep. ESD-TR-75-306, 1976. [Online]. Available: https://csrc.nist.gov/csrc/media/publications/conference-paper/1998/10/08/proceedings-of-the-21st-nissc-1998/documents/early-cs-papers/bell76.pdf.

[18] C. S. Technology, Ed., *Audit and Evaluation of Computer Security*, National Bureau of Standards, 1977. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nbsspecialpublication500-19.pdf.

[19] B. Middleton, *A History of Cyber Security Attacks*. Auerbach Publications, Jul. 2017. DOI: 10.1201/9781315155852.

[20] M. Warner, "Notes on the evolution of computer security policy in the US government, 1965-2003," *IEEE Annals of the History of Computing*, vol. 37, no. 2, pp. 8–18, Apr. 2015. DOI: 10.1109/mahc.2015.25.

[21] J. R. Yost, "The origin and early history of the computer security software products industry," *IEEE Annals of the History of Computing*, vol. 37, no. 2, pp. 46–58, Apr. 2015. DOI: 10.1109/mahc.2015.21.

[22] S. B. Lipner, "The birth and death of the orange book," *IEEE Annals of the History of Computing*, vol. 37, no. 2, pp. 19–31, Apr. 2015. DOI: 10.1109/mahc.2015.27.

[23] H. J. Highland, "The brain virus: Fact and fantasy," *Computers & Security*, vol. 7, no. 4, pp. 367–370, Aug. 1988, ISSN: 0167-4048. DOI: 10.1016/0167-4048(88)90576-7.

[24] K. R. van Wyk, "The lehigh virus," *Computers & Security*, vol. 8, no. 2, pp. 107–110, Apr. 1989, ISSN: 0167-4048. DOI: 10.1016/0167-4048(89)90064-3.

[25] H. J. Highland, "A history of computer viruses — introduction," *Computers & Security*, vol. 16, no. 5, pp. 412–415, Jan. 1997. DOI: 10.1016/s0167-4048(97)82245-6. [Online]. Available: https://www.cs.umd.edu/class/spring2018/cmsc414-0101/papers/3viruses.pdf.

[26] D. Denning, "An intrusion-detection model," *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 222–232, Feb. 1987. DOI: 10.1109/tse.1987.232894.

[27] L. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood, and D. Wolber, *A network security monitor*. Nov. 1989. DOI: 10.2172/6223037.

[28] H. Orman, "The morris worm: A fifteen-year perspective," *IEEE Security & Privacy*, vol. 1, no. 5, pp. 35–43, Sep. 2003, ISSN: 1558-4046. DOI: 10.1109/msecp.2003.1236233.

[29] I. Urzay, "Collective intelligence approaches to malware recognition," *Network Security*, vol. 2008, no. 5, pp. 14–16, May 2008. DOI: 10.1016/s1353-4858(08)70065-5.

[30] K. A. Rhodes, "The melissa computer virus demonstrates urgent need for stronger protection over systems and sensitive data," United States General Accounting Office, Tech. Rep., 1999. [Online]. Available: https://www.govinfo.gov/content/pkg/GAOREPORTS-T-AIMD-99-146/pdf/GAOREPORTS-T-AIMD-99-146.pdf.

[31] US Executive Office of the President, *Defending america's cyberspace: National plan for information systems protection version 1.0: An invitation to a dialogue*, Legislative Hearing/Committee Report, 2000. [Online]. Available: http://www.fas.org/irp/offdocs/pdd/CIP-plan.pdf.

[32] Department of Defense, *Information operations roadmap*, Oct. 2003. [Online]. Available: https://nsarchive2.gwu.edu/NSAEBB/NSAEBB177/info_ops_roadmap.pdf.

[33] F. Avolio, "Firewalls and internet security, the second hundred (internet) years," *The Internet Protocol Journal*, vol. 2, no. 2, pp. 24–32, Jun. 1999. [Online]. Available: https://ipj.dreamhosters.com/wp-content/uploads/issues/1999/ipj02-2.pdf.

[34] J. R. Vacca and S. Ellis, *Firewalls, Jumpstart for Network and Systems Administrators*. Digital Press, 2004, p. 448, ISBN: 9781555582975.

[35] C. J. May, J. Hammerstein, J. Mattson, and K. Rush, "Defense in depth: Foundation for secure and resilient IT enterprises," Carnegie-Mellon University Pittsburgh PA Sofware Engineering INST, Tech. Rep., 2006. [Online]. Available: https://apps.dtic.mil/sti/citations/ADA460375.

[36] D. E. E. 3rd and C. W. Kaufman, *Domain Name System Security Extensions*, RFC 2065, Jan. 1997. DOI: 10.17487/RFC2065. [Online]. Available: https://www.rfc-editor.org/info/rfc2065.

[37] S. Sarkar, G. Choudhary, S. K. Shandilya, A. Hussain, and H. Kim, "Security of zero trust networks in cloud computing: A comparative review," *Sustainability*, vol. 14, no. 18, p. 11 213, Sep. 2022. DOI: 10.3390/su141811213.

[38] N. Sheikh, M. Pawar, and V. Lawrence, "Zero trust using network micro segmentation," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, IEEE, May 2021, pp. 1–6. DOI: 10.1109/infocomwkshps51825.2021.9484645. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9484645.

[39] R. Vanickis, P. Jacob, S. Dehghanzadeh, and B. Lee, "Access control policy enforcement for zero-trust-networking," in *2018 29th Irish Signals and Systems Conference (ISSC)*, IEEE, IEEE, Jun. 2018, pp. 1–6. DOI: 10.1109/issc.2018.8585365. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8585365.

[40] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines, "5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges," *Computer Networks*, vol. 167, p. 106 984, Feb. 2020. DOI: 10.1016/j.comnet.2019.106984.

[41] J. G. Grimes, "Vision for a net-centric, service-oriented dod enterprise," Department of Defense Global Information Grid, Tech. Rep., 2007. [Online]. Available: https://www.acqnotes.com/Attachments/DoDGIGArchitecturalVision,June07.pdf.

[42] B. Embrey, "The top three factors driving zero trust adoption," *Computer Fraud & Security*, vol. 2020, no. 9, pp. 13–15, Jan. 2020. DOI: 10.1016/s1361-3723(20)30097-x.

[43] S. Dennis and D. B. Rogers, "Zero trust: Enabling the last mile of biomedical AI research," *Blockchain in Healthcare Today*, vol. 6, no. 1, Jan. 2023. DOI: 10.30953/bhty.v6.259.

[44] C. A. Iordache, A. V. Dragomir, and C. V. Marian, "Public institutions updated enhanced biometric security, zero trust architecture and multi-factor authentification," in *2022 International Symposium on Electronics and Telecommunications (ISETC)*, IEEE, Nov. 2022. DOI: 10.1109/isetc56213.2022.10010127.

[45] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "MITRE ATT&CK®: Design and philosophy," The MITRE Corporation, Tech. Rep., 2018. [Online]. Available: https://www.mitre.org/sites/default/files/2021-11/prs-19-01075-28-mitre-attack-design-and-philosophy.pdf.

[46] A. Bogdanov, D. Khovratovich, and C. Rechberger, "Biclique cryptanalysis of the full aes," in *Advances in Cryptology – ASIACRYPT 2011*. Springer Berlin Heidelberg, 2011, pp. 344–371, ISBN: 9783642253850. DOI: 10.1007/978-3-642-25385-0_19.

[47] D. Boneh and V. Shoup, *A Graduate Course in Applied Cryptography*. Jan. 2023. [Online]. Available: https://toc.cryptobook.us/book.pdf.

[48] Z. Abaid, A. Shaghaghi, R. Gunawardena, S. Seneviratne, A. Seneviratne, and S. Jha, "Health access broker: Secure, patient-controlled management of personal health records in the cloud," in *13th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2020)*, Springer International Publishing, Aug. 2020, pp. 111–121. DOI: 10.1007/978-3-030-57805-3_11.

[49] E. B. Boumhaout and A. Danielsen, "Towards zero trust for critical infrastructure: Rethinking the industrial demilitarized zone," in *Computer Science*, 2020. [Online]. Available: https://www.semanticscholar.org/paper/Towards-Zero-Trust-For-Critical-Infrastructure:-The-Boumhaout-Danielsen/2502ca47bb757357d958d14ba17c60e152254afe.

[50] Q. Shen and Y. Shen, "Endpoint security reinforcement via integrated zero-trust systems: A collaborative approach," *Computers & Security*, p. 103 537, Oct. 2023. DOI: 10.1016/j.cose.2023.103537.

[51] M. R. Brannsten, F. T. Johnsen, T. H. Bloebaum, and K. Lund, "Toward federated mission networking in the tactical domain," *IEEE Communications Magazine*, vol. 53, no. 10, pp. 52–58, Oct. 2015. DOI: 10.1109/mcom.2015.7295463.

[52] A. Kiser, J. Hess, E. M. Bouhafa, and S. Williams, "The combat cloud: Enabling multi-domain command and control across the range of military operations," AIR COMMAND and STAFF COLLEGE, Tech. Rep. AD1042210, Mar. 2017. [Online]. Available: https://apps.dtic.mil/sti/tr/pdf/AD1042210.pdf.

[53] F. Baker, *Requirements for IP Version 4 Routers*, RFC 1812, Jun. 1995. DOI: 10.17487/RFC1812. [Online]. Available: https://www.rfc-editor.org/info/rfc1812.

[54] T. Sharma, T. Wang, C. D. Giulio, and M. Bashir, "Towards inclusive privacy protections in the cloud," in *Lecture Notes in Computer Science*, Springer International Publishing, 2020, pp. 337–359. DOI: 10.1007/978-3-030-61638-0_19.

[55] NIST, "NIST PRIVACY FRAMEWORK:" NIST, Tech. Rep., Jan. 2020, https://www.nist.gov/privacy-framework. DOI: 10.6028/nist.cswp.01162020.

[56] J. Domingo-Ferrer, O. Farràs, J. Ribes-González, and D. Sánchez, "Privacy-preserving cloud computing on sensitive data: A survey of methods, products and challenges," *Computer Communications*, vol. 140-141, pp. 38–60, May 2019. DOI: 10.1016/j.comcom.2019.04.011.

[57] A. Pfitzmann and M. Köhntopp, "Anonymity, unobservability, and pseudonymity — a proposal for terminology," in *Designing Privacy Enhancing Technologies*, Springer Berlin Heidelberg, 2001, pp. 1–9. DOI: 10.1007/3-540-44702-4_1.

[58] T. Moran, I. Orlov, and S. Richelson, "Topology-hiding computation," in *Theory of Cryptography*, Springer Berlin Heidelberg, 2015, pp. 159–181. DOI: 10.1007/978-3-662-46494-6_8.

[59] C. Díaz, S. Seys, J. Claessens, and B. Preneel, "Towards measuring anonymity," in *Privacy Enhancing Technologies*, Springer Berlin Heidelberg, 2003, pp. 54–68. DOI: 10.1007/3-540-36467-6_5.

[60] A. Kaminsky, M. Kurdziel, S. Farris, M. Lukowiak, and S. Radziszowski, "Solving the cross domain problem with functional encryption," in *MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM)*, IEEE, IEEE, Nov. 2021, pp. 49–54. DOI: 10.1109/milcom52596.2021.9652958.

[61] A. Guellier, C. Bidan, and N. Prigent, "Homomorphic cryptography-based privacy-preserving network communications," in *Applications and Techniques in Information Security*, Springer Berlin Heidelberg, 2014, pp. 159–170. DOI: 10.1007/978-3-662-45670-5_15.

[62] F. Tusa, D. Griffin, and M. Rio, "Homomorphic routing," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Future of Internet Routing & Addressing*, ACM, Sep. 2023. DOI: 10.1145/3607504.3609287.

[63] E. Rosen, A. Viswanathan, and R. Callon, *Multiprotocol label switching architecture*, 2001.

[64] J. Postel, "Internet protocol," Tech. Rep., Sep. 1981, RFC 791. DOI: 10.17487/rfc0791.

[65] D. S. E. Deering and B. Hinden, *Internet Protocol, Version 6 (IPv6) Specification*, RFC 8200, Jul. 2017. DOI: 10.17487/RFC8200. [Online]. Available: https://www.rfc-editor.org/info/rfc8200.

[66] B. Thomas, L. Andersson, and I. Minei, *LDP Specification*, RFC 5036, Oct. 2007. DOI: 10.17487/RFC5036. [Online]. Available: https://www.rfc-editor.org/info/rfc5036.

[67] I. N. Herstein, *Topics in algebra*. John Wiley & Sons, 1964.

[68] H. Liu, "Routing table compaction in ternary CAM," *IEEE Micro*, vol. 22, no. 1, pp. 58–64, 2002. DOI: 10.1109/40.988690.

[69] T. Curry, D. Callahan, B. Fuller, and L. Michel, "DOCSDN: Dynamic and optimal configuration of software-defined networks," in *Information Security and Privacy*, Springer International Publishing, 2019, pp. 456–474. DOI: 10.1007/978-3-030-21548-4_25.

Zero Trust Architectures References

[70] Cybersecurity Information, "Embracing a zero trust security model," National Security Agency, Tech. Rep., 2021. [Online]. Available: https://media.defense.gov/2021/Feb/25/2002588479/-1/-1/0/CSI_EMBRACING_ZT_SECURITY_MODEL_UOO115131-21.PDF.

[71] K. Uttecht, "Zero trust (ZT) concepts for federal government architectures," MASSACHUSETTS INST OF TECH LEXINGTON, Tech. Rep., 2020. [Online]. Available: https://apps.dtic.mil/sti/citations/AD1106904.

[72] P. Phiayura and S. Teerakanok, "A comprehensive framework for migrating to zero trust architecture," *IEEE Access*, vol. 11, pp. 19 487–19 511, 2023. DOI: 10.1109/access.2023.3248622.

[73] E. Gilman and D. Barth, *Zero Trust Networks*. O'Reilly Media, Incorporated, 2017. [Online]. Available: https://www.usenix.org/sites/default/files/conference/protected-files/lisa16_slides_gilman.pdf.

[74] C. DeCusatis, P. Liengtiraphan, A. Sager, and M. Pinelli, "Implementing zero trust cloud networks with transport access control and first packet authentication," in *2016 IEEE International Conference on Smart Cloud (SmartCloud)*, IEEE, IEEE, Nov. 2016, pp. 5–10. DOI: 10.1109/smartcloud.2016.22.

[75] A. Moubayed, A. Refaey, and A. Shami, "Software-defined perimeter (SDP): State of the art secure solution for modern networks," *IEEE Network*, vol. 33, no. 5, pp. 226–233, Sep. 2019. DOI: 10.1109/mnet.2019.1800324.

[76] P. Simmonds, *De-perimeterisation*, 2004. [Online]. Available: https://www.blackhat.com/presentations/bh-usa-04/bh-us-04-simmonds.pdf.

[77] J. Kindervag, "No more chewy centers: Introducing the zero trust model of information security," Forrester, Tech. Rep., 2010. [Online]. Available: https://media.paloaltonetworks.com/documents/Forrester-No-More-Chewy-Centers.pdf.

[78] R. Ward and B. Beyer, *Beyondcorp: A new approach to enterprise security*, 2014. [Online]. Available: https://www.usenix.org/system/files/login/articles/login_dec14_02_ward.pdf.

[79] B. Spear, L. Cittadini, M. Saltonstall, and B. Beyer, *Beyondcorp: The access proxy*, 2016. [Online]. Available: https://www.usenix.org/system/files/login/articles/login_winter16_05_cittadini.pdf.

[80] B. Osborn, J. McWilliams, B. Beyer, and M. Saltonstall, "Beyondcorp: Design to deployment at google," 2016. [Online]. Available: https://www.usenix.org/system/files/login/articles/login_spring16_06_osborn.pdf.

[81] Cybersecurity and I. S. Agency, "Zero trust maturity model," Cybersecurity and Infrastructure Security Agency (CISA), Tech. Rep., 2021. [Online]. Available: https://www.cisa.gov/zero-trust-maturity-model.

[82] R. Freter, "Zero trust reference architecture," Department of Defense, Tech. Rep., 2022. [Online]. Available: https://dodcio.defense.gov/Portals/0/Documents/Library/(U)ZT_RA_v2.0(U)_Sep22.pdf.

[83] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," National Institute of Standards and Technology, Tech. Rep. NIST SP 800-207, Aug. 2020. DOI: 10.6028/nist.sp.800-207. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-207/final.

[84] C. Cunningham, "The zero trust eXtended (ZTX) ecosystem," Forrester, Tech. Rep., 2018. [Online]. Available: https://www.cisco.com/c/dam/m/en_sg/solutions/security/pdfs/forrester-ztx.pdf.

[85] Microsoft, "Evolving zero trust," Microsoft, Tech. Rep., 2021. [Online]. Available: https://go.microsoft.com/fwlink/p/?LinkID=2170762.

[86] S. Vanveerdeghem, "Vmware nsx - context-aware microsegmentation," VMware, Tech. Rep., 2018. [Online]. Available: https://www.vmware.com/content/dam/digitalmarketing/vmware/fr/pdf/products/nsx/vmware-context-aware-micro-segmentation-solution-overview.pdf.

[87] Google, *BeyondProd*, Nov. 2022. [Online]. Available: https://cloud.google.com/docs/security/beyondprod.

[88] Cloudflare, "Cloudflare one," Cloudflare, Tech. Rep., 2022. [Online]. Available: https://www.cloudflare.com/static/e9ea5dfaa69c554cc1cbaa7f3e441acf/Cloudflare_One_at_a_glance.pdf.

[89] J. Garbis and J. Koilpillai, "Software-defined perimeter (SDP) specification v2.0," Cloud Security Alliance, Tech. Rep., 2022. [Online]. Available: https://cloudsecurityalliance.org/artifacts/software-defined-perimeter-zero-trust-specification-v2/.

[90] P. Assunção, "A zero trust approach to network security," in *Proceedings of the Digital Privacy and Security Conference 2019*, 2019. [Online]. Available: https://privacyandsecurityconference.pt/conference2019/Proceedings_Digital_Privacy_and_Security_Conference_2019.pdf.

[91] M. Campbell, "Beyond zero trust: Trust is a vulnerability," *Computer*, vol. 53, no. 10, pp. 110–113, Oct. 2020. DOI: 10.1109/mc.2020.3011081.

[92] J. M. Pittman, S. Alaee, C. Crosby, T. Honey, and G. M. Schaefer, "Towards a model for zero trust data," *American Journal of Science & Engineering*, vol. 3, no. 1, pp. 18–24, Jun. 2022. DOI: 10.15864/ajse.3103.

[93] A. Poirrier, L. Cailleux, and T. H. Clausen, "An interoperable zero trust federated architecture for tactical systems," in *MILCOM 2023 - 2023 IEEE Military Communications Conference (MILCOM)*, IEEE, Oct. 2023. DOI: 10.1109/milcom58377.2023.10356247.

[94] B. Bilger, J. Schweitzer, J. Koilpillai, *et al.*, "SDP specification v1.0," Cloud Security Alliance, Tech. Rep., 2014. [Online]. Available: https://cloudsecurityalliance.org/artifacts/sdp-specification-v1-0/.

[95] E. Bertino and K. Brancik, "Services for zero trust architectures - a research roadmap," in *2021 IEEE International Conference on Web Services (ICWS)*, IEEE, Sep. 2021. DOI: 10.1109/icws53863.2021.00016.

[96] Y. Chen, H.-c. Hu, and G.-z. Cheng, "Design and implementation of a novel enterprise network defense system bymaneuveringmulti-dimensional network properties," *Frontiers of Information Technology & Electronic Engineering*, vol. 20, no. 2, pp. 238–252, Feb. 2019. DOI: 10.1631/fitee.1800516.

[97] Cybersecurity and I. S. Agency, "Zero trust maturity model version 2.0," Tech. Rep., Apr. 2023. [Online]. Available: https://www.cisa.gov/zero-trust-maturity-model.

[98] J. Kindervag, S. Balaouras, and L. Coit, "Build security into your network's DNA: The zero trust architecture," *Forrester Research Inc*, 2010. [Online]. Available: https://www.virtualstarmedia.com/downloads/Forrester_zero_trust_DNA.pdf.

[99] S. Dhar and I. Bose, "Securing IoT devices using zero trust and blockchain," *Journal of Organizational Computing and Electronic Commerce*, vol. 31, no. 1, pp. 18–34, Nov. 2021. DOI: 10.1080/10919392.2020.1831870.

[100] M. Samaniego and R. Deters, "Zero-trust hierarchical management in IoT," in *2018 IEEE International Congress on Internet of Things (ICIOT)*, IEEE, IEEE, Jul. 2018, pp. 88–95. DOI: 10.1109/iciot.2018.00019.

[101] Google, IBM, and Lyft, *The Istio service mesh*, 2018. [Online]. Available: https://istio.io/.

[102] Y. Fu, F. Lou, F. Meng, Z. Tian, H. Zhang, and F. Jiang, "An intelligent network attack detection method based on RNN," in *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, IEEE, Jun. 2018. DOI: 10.1109/dsc.2018.00078.

[103] D. D'Silva and D. D. Ambawade, "Building a zero trust architecture using Kubernetes," in *2021 6th International Conference for Convergence in Technology (I2CT)*, IEEE, IEEE, Apr. 2021, pp. 1–8. DOI: 10.1109/i2ct51068.2021.9418203.

[104] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 27–51, 2015. DOI: 10.1109/comst.2014.2330903.

[105] J. Singh, A. Refaey, and J. Koilpillai, "Adoption of the software-defined perimeter (SDP) architecture for infrastructure as a service," *Canadian Journal of Electrical and Computer Engineering*, vol. 43, no. 4, pp. 357–363, 2020. DOI: 10.1109/cjece.2020.3005316.

[106] A. Sallam, A. Refaey, and A. Shami, "Securing smart home networks with software-defined perimeter," in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, IEEE, IEEE, Jun. 2019, pp. 1989–1993. DOI: 10.1109/iwcmc.2019.8766686.

[107] B. Chen, S. Qiao, J. Zhao, *et al.*, "A security awareness and protection system for 5g smart healthcare based on zero-trust architecture," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10 248–10 263, Jul. 2021. DOI: 10.1109/jiot.2020.3041042.

[108] D. Puthal, L. T. Yang, S. Dustdar, *et al.*, "A user-centric security solution for internet of things and edge convergence," *ACM Transactions on Cyber-Physical Systems*, vol. 4, no. 3, pp. 1–19, May 2020. DOI: 10.1145/3351882.

[109] J. Koilpillai, "Software defined perimeter (SDP), a primer for CIOs," Waverley Labs LLC, Tech. Rep., 2017. [Online]. Available: https://www.waverleylabs.com/wp-content/uploads/2017/10/waverleylabs-sdp-white-paper.pdf.

[110] X. Tian and H. Song, "A zero trust method based on BLP and BIBA model," in *2021 14th International Symposium on Computational Intelligence and Design (ISCID)*, IEEE, IEEE, Dec. 2021, pp. 96–100. DOI: 10.1109/iscid52796.2021.00031.

[111] S. Mandal, D. A. Khan, and S. Jain, "Cloud-based zero trust access control policy: An approach to support work-from-home driven by COVID-19 pandemic," *New Generation Computing*, vol. 39, no. 3-4, pp. 599–622, Jun. 2021. DOI: 10.1007/s00354-021-00130-6.

[112] F. Federici, D. Martintoni, and V. Senni, "A zero-trust architecture for remote access in industrial IoT infrastructures," *Electronics*, vol. 12, no. 3, p. 566, Jan. 2023. DOI: 10.3390/electronics12030566.

[113] L. Zhang, H. Li, J. Ge, *et al.*, "EDP: An eBPF-based dynamic perimeter for SDP in data center," in *2022 23rd Asia-Pacific Network Operations and Management Symposium (APNOMS)*, IEEE, IEEE, Sep. 2022, pp. 01–06. DOI: 10.23919/apnoms56106.2022.9919966.

[114] L. Ferretti, F. Magnanini, M. Andreolini, and M. Colajanni, "Survivable zero trust for cloud computing environments," *Computers & Security*, vol. 110, p. 102 419, Nov. 2021. DOI: 10.1016/j.cose.2021.102419.

[115] L. Csikor, S. Ramachandran, and A. Lakshminarayanan, "ZeroDNS: Towards better zero trust security using DNS," in *Proceedings of the 38th Annual Computer Security Applications Conference*, ACM, Dec. 2022, pp. 699–713. DOI: 10.1145/3564625.3567968.

[116] J. Guo and M. Xu, "ZTESA – a zero-trust endogenous safety architecture: Gain the endogenous safety benefit, avoid insider threats," in *International Symposium on Computer Applications and Information Systems (ISCAIS 2022)*, M. Sarfraz and M. Cen, Eds., SPIE, vol. 12250, SPIE, May 2022, pp. 192–202. DOI: 10.1117/12.2639540.

[117] A. Alagappan, S. K. Venkatachary, and L. J. B. Andrews, "Augmenting zero trust network architecture to enhance security in virtual power plants," *Energy Reports*, vol. 8, pp. 1309–1320, Nov. 2022. DOI: 10.1016/j.egyr.2021.11.272.

[118] C. Zanasi, F. Magnanini, S. Russo, and M. Colajanni, "A zero trust approach for the cybersecurity of industrial control systems," in *2022 IEEE 21st International Symposium on Network Computing and Applications (NCA)*, IEEE, Dec. 2022. DOI: 10.1109/nca57778.2022.10013559.

[119] B. G. Jung, Y.-S. Yoo, K. Kim, B.-S. Kim, H. Lee, and H. Park, "ZTA-based federated policy control paradigm for enterprise wireless network infrastructure," in *2022 27th Asia Pacific Conference on Communications (APCC)*, IEEE, IEEE, Oct. 2022, pp. 1–5. DOI: 10.1109/apcc55198.2022.9943635.

[120] S. Riley, N. MacDonald, and L. Orans, "Zero trust network access (ztna)," Gartner, Tech. Rep., Apr. 2019. [Online]. Available: https://www.gartner.com/en/documents/3912802.

[121] Cisco, "Cisco security reference architecture," Cisco, Tech. Rep., Jul. 2023. [Online]. Available: https://www.cisco.com/c/dam/en/us/products/se/2023/7/SC_Ops/Security_Reference_Architecture_Overview_and_Use_Cases_072023_PDF.pdf.

[122] C. Katsis, F. Cicala, D. Thomsen, N. Ringo, and E. Bertino, "Can i reach you? do i need to? new semantics in security policy specification and testing," in *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*, ACM, Jun. 2021. DOI: 10.1145/3450569.3463558.

[123] W. Huang, X. Xie, Z. Wang, J. Feng, G. Han, and W. Zhang, "Zt-access: A combining zero trust access control with attribute-based encryption scheme against compromised devices in power iot environments," *Ad Hoc Networks*, vol. 145, p. 103 161, Jun. 2023, ISSN: 1570-8705. DOI: 10.1016/j.adhoc.2023.103161.

[124] J. Anderson, Q. Huang, L. Cheng, and H. Hu, "Byoz: Protecting byod through zero trust network security," in *2022 IEEE International Conference on Networking, Architecture and Storage (NAS)*, IEEE, Oct. 2022. DOI: 10.1109/nas55553.2022.9925513.

[125] Y. Palmo, S. Tanimoto, H. Sato, and A. Kanai, "Optimal federation method for embedding internet of things in software-defined perimeter," *IEEE Consumer Electronics Magazine*, vol. 12, no. 5, pp. 68–75, Sep. 2023, ISSN: 2162-2256. DOI: 10.1109/mce.2022.3207862.

[126] K. Hatakeyama, D. Kotani, and Y. Okabe, "Zero trust federation: Sharing context under user control towards zero trust in identity federation," in *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, IEEE, IEEE, Mar. 2021, pp. 514–519. DOI: 10.1109/percomworkshops51409.2021.9431116.

[127] M. Hirai, D. Kotani, and Y. Okabe, "Linking contexts from distinct data sources in zero trust federation," in *Lecture Notes in Computer Science*, Springer Nature Switzerland, 2023, pp. 136–144. DOI: 10.1007/978-3-031-25467-3_9.

Zero Trust Surveys References

[128] N. F. Syed, S. W. Shah, A. Shaghaghi, A. Anwar, Z. Baig, and R. Doss, "Zero trust architecture (ZTA): A comprehensive survey," *IEEE Access*, vol. 10, pp. 57 143–57 179, 2022. DOI: 10.1109/access.2022.3174679.

[129] X. Yan and H. Wang, "Survey on zero-trust network security," in *Communications in Computer and Information Science*, Springer Singapore, 2020, pp. 50–60. DOI: 10.1007/978-981-15-8083-3_5.

[130] C. Buck, C. Olenberger, A. Schweizer, F. Völter, and T. Eymann, "Never trust, always verify: A multivocal literature review on current knowledge and research gaps of zero-trust," *Computers & Security*, vol. 110, p. 102 436, Nov. 2021. DOI: 10.1016/j.cose.2021.102436.

[131] Y. He, D. Huang, L. Chen, Y. Ni, and X. Ma, "A survey on zero trust architecture: Challenges and future trends," *Wireless Communications and Mobile Computing*, vol. 2022, Y. Huo, Ed., pp. 1–13, Jun. 2022. DOI: 10.1155/2022/6476274.

[132] Y. Cao, S. R. Pokhrel, Y. ZHU, R. Ram Mohan Doss, and G. Li, "Automation and orchestration of zero trust architecture: Potential solutions and challenges," 2022. [Online]. Available: https://dro.deakin.edu.au/articles/preprint/Automation_and_Orchestration_of_Zero_Trust_Architecture_Potential_Solutions_and_Challenges/21385929.

[133] L. Alevizos, V. T. Ta, and M. H. Eiza, "Augmenting zero trust architecture to endpoints using blockchain: A state-of-the-art review," *SECURITY AND PRIVACY*, vol. 5, no. 1, e191, Nov. 2021. DOI: 10.1002/spy2.191.

[134] W. Yeoh, M. Liu, M. Shore, and F. Jiang, "Zero trust cybersecurity: Critical success factors and a maturity assessment framework," *Computers & Security*, vol. 133, p. 103 412, Oct. 2023. DOI: 10.1016/j.cose.2023.103412.

[135] B. Paul and M. Rao, "Zero-trust model for smart manufacturing industry," *Applied Sciences*, vol. 13, no. 1, p. 221, Dec. 2022. DOI: 10.3390/app13010221.

[136] N. Unger, S. Dechand, J. Bonneau, *et al.*, "SoK: Secure messaging," in *2015 IEEE Symposium on Security and Privacy*, IEEE, May 2015. DOI: 10.1109/sp.2015.22.

[137] K. J. Abhilash and K. S. Shivaprakasha, "Secure routing protocol for MANET: A survey," in *Lecture Notes in Electrical Engineering*, Springer Singapore, Dec. 2019, pp. 263–277. DOI: 10.1007/978-981-15-0626-0_22.

Zero Trust Evaluations References

[138] B. Scott, "How a zero trust approach can help to secure your AWS environment," *Network Security*, vol. 2018, no. 3, pp. 5–8, Mar. 2018. DOI: 10.1016/s1353-4858(18)30023-0.

[139] S. Teerakanok, T. Uehara, and A. Inomata, "Migrating to zero trust architecture: Reviews and challenges," *Security and Communication Networks*, vol. 2021, Q. Li, Ed., pp. 1–10, May 2021. DOI: 10.1155/2021/9947347.

[140] G. Gonçalves, K. O'Malley, B. A. E. Beyer, and M. Saltonstall, "Beyondcorp and the long tail of zero trust," *;login:*, 2023. [Online]. Available: https://www.usenix.org/publications/loginonline/beyondcorp-and-long-tail-zero-trust.

[141] L. Dong, Z. Niu, Y. Zhu, and W. Zhang, "Specifying and verifying SDP protocol based zero trust architecture using TLA+," in *Proceedings of the 7th International Conference on Cyber Security and Information Engineering*, ACM, Sep. 2022, pp. 35–43. DOI: 10.1145/3558819.3558826.

[142] O. Zheng, J. Poon, and K. Beznosov, "Application-based TCP hijacking," in *Proceedings of the Second European Workshop on System Security*, ACM, Mar. 2009, pp. 9–15. DOI: 10.1145/1519144.1519146.

[143] A. Z. Alalmaie, P. Nanda, X. He, and M. S. Alayan, "Why zero trust framework adoption has emerged during and after covid-19 pandemic," in *Advanced Information Networking and Applications*, Springer International Publishing, 2023, pp. 181–192. DOI: 10.1007/978-3-031-28694-0_17.

[144] P. Kumar, A. Moubayed, A. Refaey, A. Shami, and J. Koilpillai, "Performance analysis of SDP for secure internal enterprises," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, IEEE, IEEE, Apr. 2019, pp. 1–6. DOI: 10.1109/wcnc.2019.8885784.

[145] S. Mehraj and M. T. Banday, "Establishing a zero trust strategy in cloud computing environment," in *2020 International Conference on Computer Communication and Informatics (ICCCI)*, IEEE, IEEE, Jan. 2020, pp. 1–6. DOI: 10.1109/iccci48352.2020.9104214.

[146] Y. Bobbert and T. Timmermans, "How zero trust as a service (ZTaaS) reduces the cost of a breach," in *Proceedings of the Future Technologies Conference (FTC) 2023, Volume 4*, Springer Nature Switzerland, 2023, pp. 433–454. DOI: 10.1007/978-3-031-47448-4_33.

[147] E. Thiele and E. Hachichou, "ztLang: A modelling language for zero trust networks," M.S. thesis, KTH Royal Institute of Technology, 2022. [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2:1703897&dswid=-8661.

[148] D. L. V. Bossuyt, N. Papakonstantinou, B. Hale, and R. Arlitt, "Trust loss effects analysis method for zero trust assessment," in *2023 Annual Reliability and Maintainability Symposium (RAMS)*, IEEE, Jan. 2023. DOI: 10.1109/rams51473.2023.10088265.

[149] R. R. Omar and T. M. Abdelaziz, "A comparative study of network access control and software-defined perimeter," in *Proceedings of the 6th International Conference on Engineering & MIS 2020*, ACM, Sep. 2020, pp. 1–5. DOI: 10.1145/3410352.3410754.

[150] J. H. Hurley, "Zero trust is not enough: Mitigating data repository breaches," *International Conference on Cyber Warfare and Security*, vol. 18, no. 1, pp. 137–144, Feb. 2023. DOI: 10.34190/iccws.18.1.1068.

[151] N. Basta, M. Ikram, M. A. Kaafar, and A. Walker, "Towards a zero-trust micro-segmentation network security strategy: An evaluation framework," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, IEEE, IEEE, Apr. 2022, pp. 1–7. DOI: 10.1109/noms54207.2022.9789888.

[152] H. Teymourlouei, "A machine learning approach to the evaluation of zero trust compliance in network infrastructure," in *2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, IEEE, Jul. 2023. DOI: 10.1109/iceccme57830.2023.10253205.

[153] P. Robinson, "Why is zero trust so difficult?" *Computer Fraud & Security*, vol. 2023, no. 3, Mar. 2023. DOI: 10.12968/s1361-3723(23)70014-6.

[154] K. Strandell and S. Mittal, "Risks to zero trust in a federated mission partner environment," *arXiv preprint arXiv:2211.17073*, Nov. 2022. DOI: 10.48550/ARXIV.2211.17073. arXiv: 2211.17073 [cs.CR].

[155] W. R. Simpson and K. E. Foltz, "Maintaining zero trust with federation," *International Journal of Emerging Technology and Advanced Engineering*, vol. 11, no. 5, pp. 17–32, May 2021. DOI: 10.46338/ijetae0521_03.

[156] K. Olson and E. Keller, "Federating trust," in *Proceedings of the SIGCOMM '21 Poster and Demo Sessions*, ACM, Aug. 2021. DOI: 10.1145/3472716.3472865.

Zero Trust Technologies References

[157] D. Eidle, S. Y. Ni, C. DeCusatis, and A. Sager, "Autonomic security for zero trust networks," in *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, IEEE, IEEE, Oct. 2017, pp. 288–293. DOI: 10.1109/uemcon.2017.8249053.

[158] S. Xiao, Y. Ye, N. Kanwal, T. Newe, and B. Lee, "SoK: Context and risk aware access control for zero trust systems," *Security and Communication Networks*, vol. 2022, A. Fu, Ed., pp. 1–20, Jun. 2022. DOI: 10.1155/2022/7026779.

[159] B. Dowling, F. Günther, and A. Poirrier, "Continuous authentication in secure messaging," in *Computer Security – ESORICS 2022*, Springer Nature Switzerland, 2022, pp. 361–381. DOI: 10.1007/978-3-031-17146-8_18.

[160] P. A. Grassi, M. E. Garcia, and J. L. Fenton, "Digital identity guidelines: Revision 3," National Institute of Standards and Technology, Tech. Rep. SP 800-63-3, Feb. 2020. DOI: 10.6028/nist.sp.800-63-3. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-63/3/final.

[161] M. Campbell, "Putting the passe into passwords: How passwordless technologies are reshaping digital identity," *Computer*, vol. 53, no. 8, pp. 89–93, Aug. 2020. DOI: 10.1109/mc.2020.2997278.

[162] S. W. Shah and S. S. Kanhere, "Recent trends in user authentication – a survey," *IEEE Access*, vol. 7, pp. 112 505–112 519, 2019. DOI: 10.1109/access.2019.2932400.

[163] Q. Xiao, "Security issues in biometric authentication," in *Proceedings from the Sixth Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop, 2005.*, IEEE, 2005. DOI: 10.1109/iaw.2005.1495927.

[164] A. Ometov, S. Bezzateev, N. Mäkitalo, S. Andreev, T. Mikkonen, and Y. Koucheryavy, "Multi-factor authentication: A survey," *Cryptography*, vol. 2, no. 1, p. 1, Jan. 2018. DOI: 10.3390/cryptography2010001.

[165] S. W. Shah and S. S. Kanhere, "Smart user identification using cardiopulmonary activity," *Pervasive and Mobile Computing*, vol. 58, p. 101 024, Aug. 2019. DOI: 10.1016/j.pmcj.2019.05.005.

[166] C. M. Lonvick and T. Ylonen, *The Secure Shell (SSH) Authentication Protocol*, RFC 4252, Jan. 2006. DOI: 10.17487/RFC4252. [Online]. Available: https://www.rfc-editor.org/info/rfc4252.

[167] I. Matiushin and V. Korkhov, "Continuous authentication methods for zero-trust cybersecurity architecture," in *Computational Science and Its Applications – ICCSA 2023 Workshops*, Springer Nature Switzerland, 2023, pp. 334–351. DOI: 10.1007/978-3-031-37120-2_22.

[168] H. Saevanee, N. Clarke, S. Furnell, and V. Biscione, "Continuous user authentication using multi-modal biometrics," *Computers & Security*, vol. 53, pp. 234–246, Sep. 2015. DOI: 10.1016/j.cose.2015.06.001.

[169] W. Louis, M. Komeili, and D. Hatzinakos, "Continuous authentication using one-dimensional multi-resolution local binary patterns (1dmrlbp) in ECG biometrics," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 12, pp. 2818–2832, Dec. 2016. DOI: 10.1109/tifs.2016.2599270.

[170] A. Mosenia, S. Sur-Kolay, A. Raghunathan, and N. K. Jha, "CABA: Continuous authentication based on BioAura," *IEEE Transactions on Computers*, vol. 66, no. 5, pp. 759–772, May 2017. DOI: 10.1109/tc.2016.2622262.

[171] G. R. da Silva, D. F. Macedo, and A. L. dos Santos, "Zero trust access control with context-aware and behavior-based continuous authentication for smart homes," in *Anais do XXI Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg 2021)*, SBC, Sociedade Brasileira de Computação - SBC, Oct. 2021, pp. 43–56. DOI: 10.5753/sbseg.2021.17305.

[172] F. Juefei-Xu, C. Bhagavatula, A. Jaech, U. Prasad, and M. Savvides, "Gait-ID on the move: Pace independent human identification using cell phone accelerometer dynamics," in *2012 IEEE Fifth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, IEEE, Sep. 2012. DOI: 10.1109/btas.2012.6374552.

[173] H. M. Thang, V. Q. Viet, N. D. Thuc, and D. Choi, "Gait identification using accelerometer on mobile phone," in *2012 International Conference on Control, Automation and Information Sciences (ICCAIS)*, IEEE, Nov. 2012. DOI: 10.1109/iccais.2012.6466615.

[174] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song, "Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 1, pp. 136–148, Jan. 2013. DOI: 10.1109/tifs.2012.2225048.

[175] L. S. Dasu, M. Dhamija, G. Dishitha, A. Vivekanandan, and S. V, "Defending against identity threats using adaptive authentication," in *2023 IEEE 8th International Conference for Convergence in Technology (I2CT)*, IEEE, Apr. 2023. DOI: 10.1109/i2ct57861.2023.10126295.

[176] O. Michel and E. Keller, "Policy routing using process-level identifiers," in *2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*, IEEE, IEEE, Apr. 2016, pp. 7–12. DOI: 10.1109/ic2ew.2016.10.

[177] E. Rescorla, *The Transport Layer Security (TLS) Protocol Version 1.3*, RFC 8446, Aug. 2018. DOI: 10.17487/RFC8446. [Online]. Available: https://www.rfc-editor.org/info/rfc8446.

[178] H. Chang, M. Kodialam, T. Lakshman, and S. Mukherjee, "Microservice fingerprinting and classification using machine learning," in *2019 IEEE 27th International Conference on Network Protocols (ICNP)*, IEEE, Oct. 2019. DOI: 10.1109/icnp.2019.8888077.

[179] K. A. Torkura, M. I. Sukmana, and C. Meinel, "Integrating continuous security assessments in microservices and cloud native applications," in *Proceedings of the10th International Conference on Utility and Cloud Computing*, ACM, Dec. 2017. DOI: 10.1145/3147213.3147229.

[180] K.-Y. Lam and C.-H. Chi, "Identity in the internet-of-things (IoT): New challenges and opportunities," in *Information and Communications Security*, Springer International Publishing, 2016, pp. 18–26. DOI: 10.1007/978-3-319-50011-9_2.

[181] K. Reaz and G. Wunder, "ASOP: A sovereign and secure device onboarding protocol for cloud-based IoT services," in *2022 6th Cyber Security in Networking Conference (CSNet)*, IEEE, IEEE, Oct. 2022, pp. 1–3. DOI: 10.1109/csnet56116.2022.9955610.

[182] A. Kanuparthi, R. Karri, and S. Addepalli, "Hardware and embedded security in the context of internet of things," in *Proceedings of the 2013 ACM workshop on Security, privacy & dependability for cyber vehicles*, ACM, Nov. 2013. DOI: 10.1145/2517968.2517976.

[183] A. Hamza, D. Ranathunga, H. H. Gharakheili, M. Roughan, and V. Sivaraman, "Clear as MUD," in *Proceedings of the 2018 Workshop on IoT Security and Privacy*, ACM, Aug. 2018. DOI: 10.1145/3229565.3229566.

[184] I. Ali, S. Sabir, and Z. Ullah, "Internet of things security, device authentication and access control: A review," *International Journal of Computer Science and Information Security (IJCSIS), Vol. 14, No. 8, August 2016*, Jan. 2019. DOI: 10.48550/ARXIV.1901.07309. arXiv: 1901.07309 [cs.CR].

[185] M. Fagan, J. Marron, P. Watrobski, M. Souppaya, W. Barker, *et al.*, "Trusted internet of things (iot) device network-layer onboarding and lifecycle management: Enhancing internet protocol-based iot device and network security," National Institute of Standards and Technology, Tech. Rep. 1800-36, May 2023. [Online]. Available: https://csrc.nist.gov/pubs/sp/1800/36/iprd.

[186] F. Restuccia, S. D'Oro, and T. Melodia, "Securing the internet of things in the age of machine learning and software-defined networking," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4829–4842, Dec. 2018. DOI: 10.1109/jiot.2018.2846040.

[187] Z. Huang and Q. Wang, "A PUF-based unified identity verification framework for secure IoT hardware via device authentication," *World Wide Web*, vol. 23, no. 2, pp. 1057–1088, Apr. 2019. DOI: 10.1007/s11280-019-00677-x.

[188] A. Braeken, "PUF based authentication protocol for IoT," *Symmetry*, vol. 10, no. 8, p. 352, Aug. 2018. DOI: 10.3390/sym10080352.

[189] M. Á. Prada-Delgado, I. Baturone, G. Dittmann, J. Jelitto, and A. Kind, "PUF-derived IoT identities in a zero-knowledge protocol for blockchain," *Internet of Things*, vol. 9, p. 100 057, Mar. 2020. DOI: 10.1016/j.iot.2019.100057.

[190] J. R. Wallrabenstein, "Practical and secure IoT device authentication using physical unclonable functions," in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, IEEE, Aug. 2016. DOI: 10.1109/ficloud.2016.22.

[191] D. Chen, N. Zhang, Z. Qin, *et al.*, "S2m: A lightweight acoustic fingerprints-based wireless device authentication protocol," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 88–100, Feb. 2017. DOI: 10.1109/jiot.2016.2619679.

[192] B. Kelley and I. Ara, "An intelligent and private 6g air interface using physical layer security," in *MILCOM 2022 - 2022 IEEE Military Communications Conference (MILCOM)*, IEEE, Nov. 2022. DOI: 10.1109/milcom55135.2022.10017638.

[193] T. Langehaug and S. Graham, "Towards hardware-based application fingerprinting with microarchitectural signals for zero trust environments," in *Proceedings of the 56th Hawaii International Conference on System Sciences*, Jan. 2023, pp. 6613–6622. [Online]. Available: https://hdl.handle.net/10125/103434.

[194] Y. Colomb, P. White, R. Islam, and A. Alsadoon, "Applying zero trust architecture and probability-based authentication to preserve security and privacy of data in the cloud," in *Emerging Trends in Cybersecurity Applications*, Springer International Publishing, Nov. 2022, pp. 137–169. DOI: 10.1007/978-3-031-09640-2_7.

[195] E. Hayashi, S. Das, S. Amini, J. Hong, and I. Oakley, "CASA," in *Proceedings of the Ninth Symposium on Usable Privacy and Security*, ACM, Jul. 2013. DOI: 10.1145/2501604.2501607.

[196] K. Benzekki, A. E. Fergougui, and A. E. ElAlaoui, "A context-aware authentication system for mobile cloud computing," *Procedia Computer Science*, vol. 127, pp. 379–387, 2018. DOI: 10.1016/j.procs.2018.01.135.

[197] M. Jakobsson, E. Shi, P. Golle, R. Chow, *et al.*, "Implicit authentication for mobile devices," in *Proceedings of the 4th USENIX conference on Hot topics in security*, vol. 1, 2009, pp. 25–27.

[198] L. Li, X. Zhao, and G. Xue, "Unobservable re-authentication for smartphones," in *NDSS*, Citeseer, vol. 56, 2013, pp. 57–59. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=9ba5c640b8d52cce96975672307c9d7560cc8535.

[199] Y. Ge and Q. Zhu, "MUFAZA: Multi-source fast and autonomous zero-trust authentication for 5g networks," in *MILCOM 2022 - 2022 IEEE Military Communications Conference (MILCOM)*, IEEE, Nov. 2022. DOI: 10.1109/milcom55135.2022.10017839.

[200] S.-H. Kim, D. Choi, S.-H. Kim, S. Cho, and K.-S. Lim, "Context-aware multimodal FIDO authenticator for sustainable IT services," *Sustainability*, vol. 10, no. 5, p. 1656, May 2018. DOI: 10.3390/su10051656.

[201] P. Fu, J. Wu, X. Lin, and A. Shen, "ZTEI: Zero-trust and edge intelligence empowered continuous authentication for satellite networks," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, IEEE, Dec. 2022. DOI: 10.1109/globecom48099.2022.10000958.

[202] S. Ding, J. Cao, C. Li, K. Fan, and H. Li, "A novel attribute-based access control scheme using blockchain for IoT," *IEEE Access*, vol. 7, pp. 38 431–38 441, 2019. DOI: 10.1109/access.2019.2905846.

[203] S. W. A. Shah, N. F. Syed, A. Shaghaghi, A. Anwar, Z. Baig, and R. Doss, "Towards a lightweight continuous authentication protocol for device-to-device communication," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, IEEE, Dec. 2020. DOI: 10.1109/trustcom50675.2020.00148.

[204] S. W. Shah, N. F. Syed, A. Shaghaghi, A. Anwar, Z. Baig, and R. Doss, "LCDA: Lightweight continuous device-to-device authentication for a zero trust architecture (ZTA)," *Computers & Security*, vol. 108, p. 102 351, Sep. 2021. DOI: 10.1016/j.cose.2021.102351.

[205] J. Torres, M. Nogueira, and G. Pujolle, "A survey on identity management for the future network," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 787–802, 2013. DOI: 10.1109/surv.2012.072412.00129.

[206] T. E. Maliki and J.-M. Seigneur, "A survey of user-centric identity management technologies," in *The International Conference on Emerging Security Information, Systems, and Technologies (SECUREWARE 2007)*, IEEE, 2007. DOI: 10.1109/secureware.2007.4385303.

[207] N. Thomas Lam, T. J. Clancy, P. T. Ranks, and J. J. W. Wilmer III, "Enterprise identity, credential and access management reference architecture," Department of Defense, Tech. Rep., 2020. [Online]. Available: https://dodcio.defense.gov/Portals/0/Documents/Cyber/DoD_Enterprise_ICAM_Reference_Design.pdf.

[208] Y. Cao and L. Yang, "A survey of identity management technology," in *2010 IEEE International Conference on Information Theory and Information Security*, IEEE, Dec. 2010. DOI: 10.1109/icitis.2010.5689468.

[209] J. Sermersheim, *Lightweight Directory Access Protocol (LDAP): The Protocol*, RFC 4511, Jun. 2006. DOI: 10.17487/RFC4511. [Online]. Available: https://www.rfc-editor.org/info/rfc4511.

[210] F. Alaca and P. C. V. Oorschot, "Comparative analysis and framework evaluating web single sign-on systems," *ACM Computing Surveys*, vol. 53, no. 5, pp. 1–34, Sep. 2020. DOI: 10.1145/3409452.

[211] K. Rhee, W. Jeon, and D. Won, "Security requirements of a mobile device management system," *International Journal of Security and Its Applications*, vol. 6, no. 2, pp. 353–358, 2012. [Online]. Available: https://www.researchgate.net/profile/Dongho-Won-2/publication/267227402_Security_Requirements_of_a_Mobile_Device_Management_System/links/55ca889508aeca747d69ea6e/Security-Requirements-of-a-Mobile-Device-Management-System.pdf.

[212] P. Steiner, "Going beyond mobile device management," *Computer Fraud & Security*, vol. 2014, no. 4, pp. 19–20, Apr. 2014. DOI: 10.1016/s1361-3723(14)70483-x.

[213] V. A. Desnitsky, I. V. Kotenko, and S. B. Nogin, "Detection of anomalies in data for monitoring of security components in the internet of things," in *2015 XVIII International Conference on Soft Computing and Measurements (SCM)*, IEEE, May 2015. DOI: 10.1109/scm.2015.7190452.

[214] D. Weissman and A. Jayasumana, "Integrating IoT monitoring for security operation center," in *2020 Global Internet of Things Summit (GIoTS)*, IEEE, Jun. 2020. DOI: 10.1109/giots49054.2020.9119680.

[215] J. Bayuk, "Data-centric security," *Computer Fraud & Security*, vol. 2009, no. 3, pp. 7–11, Mar. 2009, ISSN: 1361-3723. DOI: 10.1016/s1361-3723(09)70032-6.

[216] M. J. Dworkin, "Advanced encryption standard (AES)," National Institute of Standards and Technology, Tech. Rep. FIPS 197, May 2023. DOI: 10.6028/nist.fips.197-upd1.

[217] B. M.P. and K. R. Babu, "Secure cloud storage using aes encryption," in *2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)*, IEEE, Sep. 2016. DOI: 10.1109/icacdot.2016.7877709.

[218] S. Patranabis and D. Mukhopadhyay, "Forward and backward private conjunctive searchable symmetric encryption," in *NDSS Symposium 2021 (virtual)*, 2021.

[219] R. Bost, B. Minaud, and O. Ohrimenko, "Forward and backward private searchable encryption from constrained cryptographic primitives," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1465–1482.

[220] B. R. Waters, D. Balfanz, G. Durfee, and D. K. Smetters, "Building an encrypted and searchable audit log.," in *NDSS*, Citeseer, vol. 4, 2004, pp. 5–6.

[221] W. Diffie, "The first ten years of public-key cryptography," *Proceedings of the IEEE*, vol. 76, no. 5, pp. 560–577, May 1988. DOI: 10.1109/5.4442.

[222] S. Boeyen, S. Santesson, T. Polk, R. Housley, S. Farrell, and D. Cooper, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, RFC 5280, May 2008. DOI: 10.17487/RFC5280. [Online]. Available: https://www.rfc-editor.org/info/rfc5280.

[223] M. Hellman, "An overview of public key cryptography," *IEEE Communications Magazine*, vol. 40, no. 5, pp. 42–49, May 2002, ISSN: 0163-6804. DOI: 10.1109/mcom.2002.1006971.

[224] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Annual international cryptology conference*, Springer, 2001, pp. 213–229. [Online]. Available: https://crypto.stanford.edu/~dabo/papers/bfibe.pdf.

[225] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*, 2006, pp. 89–98.

[226] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," in *Theory of Cryptography*, Springer Berlin Heidelberg, 2011, pp. 253–273. DOI: 10.1007/978-3-642-19571-6_16.

[227] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, ACM, May 2009, pp. 169–178. DOI: 10.1145/1536414.1536440.

[228] D. Boneh, C. Gentry, S. Gorbunov, *et al.*, "Fully key-homomorphic encryption, arithmetic circuit abe and compact garbled circuits," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 2014, pp. 533–556.

[229] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Tfhe: Fast fully homomorphic encryption over the torus," *Journal of Cryptology*, vol. 33, no. 1, pp. 34–91, 2020.

[230] L. Chen, S. Jordan, Y.-K. Liu, *et al.*, "Report on post-quantum cryptography," National Institute of Standards and Technology, Tech. Rep. NISTIR 8105, Apr. 2016. DOI: 10.6028/nist.ir.8105.

[231] Y. Shen, X. Tang, X. Zhang, Y. Zhou, and H. Zou, "A flexible continuous-wave quantum cryptography scheme with zero-trust security for internet of things," *International Journal of Distributed Sensor Networks*, vol. 18, no. 11, p. 155 013 292 211 369, Nov. 2022. DOI: 10.1177/15501329221136978.

[232] National Institute of Standards and Technology, *Module-Lattice-Based Key-Encapsulation Mechanism Standard*. Aug. 2023. DOI: 10.6028/nist.fips.203.ipd.

[233] H. Hui, C. Zhou, S. Xu, and F. Lin, "A novel secure data transmission scheme in industrial internet of things," *China Communications*, vol. 17, no. 1, pp. 73–88, Jan. 2020. DOI: 10.23919/jcc.2020.01.006.

[234] M. S. Turan, K. A. McKay, Ç. Çalık, D. Chang, and L. Bassham, "Status report on the first round of the NIST lightweight cryptography standardization process," National Institute of Standards and Technology, Tech. Rep. NISTIR 8268, Oct. 2019. DOI: 10.6028/nist.ir.8268.

[235] W. J. Buchanan, S. Li, and R. Asif, "Lightweight cryptography methods," *Journal of Cyber Security Technology*, vol. 1, no. 3-4, pp. 187–201, Oct. 2017. DOI: 10.1080/23742917.2017.1384917.

[236] K. A. McKay, L. Bassham, M. S. Turan, and N. Mouha, "Report on lightweight cryptography," National Institute of Standards and Technology, Tech. Rep. NISTIR 8114, Mar. 2017. DOI: 10.6028/nist.ir.8114.

[237] U. Chatterjee, R. S. Chakraborty, and D. Mukhopadhyay, "A PUF-based secure communication protocol for IoT," *ACM Transactions on Embedded Computing Systems*, vol. 16, no. 3, pp. 1–25, Apr. 2017. DOI: 10.1145/3005715.

[238] C. Greamo and A. Ghosh, "Sandboxing and virtualization: Modern tools for combating malware," *IEEE Security & Privacy Magazine*, vol. 9, no. 2, pp. 79–82, Mar. 2011. DOI: 10.1109/msp.2011.36.

[239] R. Shu, P. Wang, S. A. G. III, *et al.*, "A study of security isolation techniques," *ACM Computing Surveys*, vol. 49, no. 3, pp. 1–37, Oct. 2016. DOI: 10.1145/2988545.

[240] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud," in *Proceedings of the 16th ACM conference on Computer and communications security*, ACM, Nov. 2009. DOI: 10.1145/1653662.1653687.

[241] A. Tomlinson, "Introduction to the TPM," in *Smart Cards, Tokens, Security and Applications*, Springer International Publishing, 2017, pp. 173–191. DOI: 10.1007/978-3-319-50500-8_7.

[242] V. Costan and S. Devadas, *Intel SGX explained*, Cryptology ePrint Archive, Paper 2016/086, Feb. 2016. [Online]. Available: https://eprint.iacr.org/2016/086.

[243] J. V. Bulck, M. Minkin, O. Weisse, *et al.*, "Breaking virtual memory protection and the SGX ecosystem with foreshadow," *IEEE Micro*, vol. 39, no. 3, pp. 66–74, May 2019. DOI: 10.1109/mm.2019.2910104.

[244] A. S. Tanenbaum, *Modern Operating Systems (2nd Edition)*. Prentice Hall, 2001, p. 976, ISBN: 9780130313584.

[245] M. J. D. Lucia, "A survey on security isolation of virtualization, containers, and unikernels," US Army Research Laboratory, Tech. Rep., 2017. [Online]. Available: https://apps.dtic.mil/sti/pdfs/AD1035194.pdf.

[246] P. A. Loscocco, S. D. Smalley, P. A. Muckelbauer, R. C. Taylor, S. J. Turner, and J. F. Farrell, "The inevitability of failure: The flawed assumption of security in modern computing environments," in *In Proceedings of the 21st National Information Systems Security Conference*, 1998. [Online]. Available: https://www.d.umn.edu/~pahp/readings/inevitability_of_failure-1998.pdf.

[247] D. R. Cheriton and K. J. Duda, "A caching model of operating system kernel functionality," *ACM SIGOPS Operating Systems Review*, vol. 29, no. 1, pp. 83–86, Jan. 1995. DOI: 10.1145/202453.202476.

[248] A. Madhavapeddy, R. Mortier, C. Rotsos, *et al.*, "Unikernels," *ACM SIGARCH Computer Architecture News*, vol. 41, no. 1, pp. 461–472, Mar. 2013. DOI: 10.1145/2490301.2451167.

[249] C. Wright, C. Cowan, J. Morris, S. Smalley, and G. Kroah-Hartman, "Linux security modules: General security support for the linux kernel," in *11th USENIX Security Symposium (USENIX Security 02)*, IEEE, 2002. DOI: 10.1109/fits.2003.1264934.

[250] J. Watada, A. Roy, R. Kadikar, H. Pham, and B. Xu, "Emerging trends, techniques and open issues of containerization: A review," *IEEE Access*, vol. 7, pp. 152 443–152 472, 2019. DOI: 10.1109/access.2019.2945930.

[251] S. M. Jain, "Namespaces," in *Linux Containers and Virtualization*, Apress, 2020, pp. 31–43. DOI: 10.1007/978-1-4842-6283-2_3.

[252] J. Edge, *A seccomp overview*, Linux Plumbers Conference, 2015. [Online]. Available: https://lwn.net/Articles/656307/.

[253] T. Combe, A. Martin, and R. D. Pietro, "To docker or not to docker: A security perspective," *IEEE Cloud Computing*, vol. 3, no. 5, pp. 54–62, Sep. 2016. DOI: 10.1109/mcc.2016.100.

[254] M. G. Xavier, M. V. Neves, and C. A. F. D. Rose, "A performance comparison of container-based virtualization systems for MapReduce clusters," in *2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, IEEE, Feb. 2014. DOI: 10.1109/pdp.2014.78.

[255] S. Sultan, I. Ahmad, and T. Dimitriou, "Container security: Issues, challenges, and the road ahead," *IEEE Access*, vol. 7, pp. 52 976–52 996, 2019. DOI: 10.1109/access.2019.2911732.

[256] A. Hannousse and S. Yahiouche, "Securing microservices and microservice architectures: A systematic mapping study," *Computer Science Review*, vol. 41, p. 100 415, Aug. 2021. DOI: 10.1016/j.cosrev.2021.100415.

[257] J. L. Peterson and A. Silberschatz, *Operating System Concepts*. Addison-Wesley Longman Publishing Co., Inc., 1985.

[258] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at google with borg," in *Proceedings of the Tenth European Conference on Computer Systems*, ACM, Apr. 2015. DOI: 10.1145/2741948.2741964.

[259] F. Soppelsa and C. Kaewkasi, *Native Docker Clustering with Swarm*. Packt Publishing - ebooks Account, 2016, p. 248, ISBN: 9781786469755.

[260] D. Bernstein, "Containers and cloud: From LXC to docker to kubernetes," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81–84, Sep. 2014. DOI: 10.1109/mcc.2014.51.

[261] K. B. Laskey and K. Laskey, "Service oriented architecture," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 1, no. 1, pp. 101–105, Jul. 2009. DOI: 10.1002/wics.8.

[262] J. Li, B. Li, T. Wo, *et al.*, "CyberGuarder: A virtualization security assurance architecture for green cloud computing," *Future Generation Computer Systems*, vol. 28, no. 2, pp. 379–390, Feb. 2012. DOI: 10.1016/j.future.2011.04.012.

[263] M. Shahzad and M. P. Singh, "Continuous authentication and authorization for the internet of things," *IEEE Internet Computing*, vol. 21, no. 2, pp. 86–90, Mar. 2017. DOI: 10.1109/mic.2017.33.

[264] M. Miettinen, S. Heuser, W. Kronz, A.-R. Sadeghi, and N. Asokan, "ConXsense," in *Proceedings of the 9th ACM symposium on Information, computer and communications security*, ACM, Jun. 2014. DOI: 10.1145/2590296.2590337.

[265] Q. Yao, Q. Wang, X. Zhang, and J. Fei, "Dynamic access control and authorization system based on zero-trust architecture," in *2020 International Conference on Control, Robotics and Intelligent System*, ACM, Oct. 2020, pp. 123–127. DOI: 10.1145/3437802.3437824.

[266] Z. Xiaojian, C. Liandong, F. Jie, W. Xiangqun, and W. Qi, "Power IoT security protection architecture based on zero trust framework," in *2021 IEEE 5th International Conference on Cryptography, Security and Privacy (CSP)*, IEEE, IEEE, Jan. 2021, pp. 166–170. DOI: 10.1109/csp51677.2021.9357607.

[267] K. Olejnik, I. Dacosta, J. S. Machado, K. Huguenin, M. E. Khan, and J.-P. Hubaux, "SmarPer: Context-aware and automatic runtime-permissions for mobile devices," in *2017 IEEE Symposium on Security and Privacy (SP)*, IEEE, May 2017. DOI: 10.1109/sp.2017.25.

[268] A. L. M. Neto, A. L. F. Souza, I. Cunha, *et al.*, "AoT," in *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, ACM, Nov. 2016. DOI: 10.1145/2994551.2994555.

[269] D. Ferraiolo, R. Chandramouli, R. Kuhn, and V. Hu, "Extensible access control markup language (XACML) and next generation access control (NGAC)," in *Proceedings of the 2016 ACM International Workshop on Attribute Based Access Control*, ACM, Mar. 2016, pp. 13–24. DOI: 10.1145/2875491.2875496.

[270] B. Bezawada, K. Haefner, and I. Ray, "Securing home IoT environments with attribute-based access control," in *Proceedings of the Third ACM Workshop on Attribute-Based Access Control*, ACM, Mar. 2018. DOI: 10.1145/3180457.3180464.

[271] J. Wang, H. Wang, H. Zhang, and N. Cao, "Trust and attribute-based dynamic access control model for internet of things," in *2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, IEEE, Oct. 2017. DOI: 10.1109/cyberc.2017.47.

[272] Y. Ge and Q. Zhu, "Trust threshold policy for explainable and adaptive zero-trust defense in enterprise networks," in *2022 IEEE Conference on Communications and Network Security (CNS)*, IEEE, IEEE, Oct. 2022, pp. 359–364. DOI: 10.1109/cns56114.2022.9947263.

[273] P. Lv, X. Sun, H. Huang, J. Qian, C. Sun, and H. Dai, "Dynamic trust continuous evaluation-based zero-trust access control for power grid cloud service," *Journal of Physics: Conference Series*, vol. 2402, no. 1, p. 012 008, Dec. 2022. DOI: 10.1088/1742-6596/2402/1/012008.

[274] J. Huang and D. M. Nicol, "Trust mechanisms for cloud computing," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 2, no. 1, p. 9, 2013. DOI: 10.1186/2192-113x-2-9.

[275] T. Dimitrakos, T. Dilshener, A. Kravtsov, *et al.*, "Trust aware continuous authorization for zero trust in consumer internet of things," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, IEEE, IEEE, Dec. 2020, pp. 1801–1812. DOI: 10.1109/trustcom50675.2020.00247.

[276] A. S. A. Saleh, E. M. R. Hamed, and M. Hashem, "Building trust management model for cloud computing," in *2014 9th International Conference on Informatics and Systems*, IEEE, Dec. 2014. DOI: 10.1109/infos.2014.7036688.

[277] T. Khan, K. Singh, K. Ahmad, and K. A. B. Ahmad, "A secure and dependable trust assessment (SDTS) scheme for industrial communication networks," *Scientific Reports*, vol. 13, no. 1, Feb. 2023. DOI: 10.1038/s41598-023-28721-x.

[278] A. L. Marra, F. Martinelli, P. Mori, and A. Saracino, "Implementing usage control in internet of things: A smart home use case," in *2017 IEEE Trustcom/BigDataSE/ICESS*, IEEE, Aug. 2017. DOI: 10.1109/trustcom/bigdatase/icess.2017.352.

[279] U. Schöpp, C. Xu, A. Ibrahim, F. Faghih, and T. Dimitrakos, "Specifying a usage control system," in *Proceedings of the 28th ACM Symposium on Access Control Models and Technologies*, ACM, May 2023. DOI: 10.1145/3589608.3593843.

[280] Z. Niu, L. Dong, and Y. Zhu, "The runtime model checking method for zero trust security policy," in *Proceedings of the 7th International Conference on Cyber Security and Information Engineering*, ACM, Sep. 2022, pp. 8–12. DOI: 10.1145/3558819.3558821.

[281] A. Almehmadi and K. El-Khatib, "On the possibility of insider threat prevention using intent-based access control (IBAC)," *IEEE Systems Journal*, vol. 11, no. 2, pp. 373–384, Jun. 2017. DOI: 10.1109/jsyst.2015.2424677.

[282] H. Mekky, F. Hao, S. Mukherjee, Z.-L. Zhang, and T. Lakshman, "Application-aware data plane processing in SDN," in *Proceedings of the third workshop on Hot topics in software defined networking*, ACM, Aug. 2014, pp. 13–18. DOI: 10.1145/2620728.2620735.

[283] G. González-Granadillo, S. González-Zarzosa, and R. Diaz, "Security information and event management (SIEM): Analysis, trends, and usage in critical infrastructures," *Sensors*, vol. 21, no. 14, p. 4759, Jul. 2021. DOI: 10.3390/s21144759.

[284] S. Bhatt, P. K. Manadhata, and L. Zomlot, "The operational role of security information and event management systems," *IEEE Security & Privacy*, vol. 12, no. 5, pp. 35–41, Sep. 2014. DOI: 10.1109/msp.2014.103.

[285] A. D'Amico and K. Whitley, "The real work of computer network defense analysts," in *VizSEC 2007*, Springer Berlin Heidelberg, 2008, pp. 19–37. DOI: 10.1007/978-3-540-78243-8_2.

[286] M. Cinque, D. Cotroneo, and A. Pecchia, "Challenges and directions in security information and event management (SIEM)," in *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, IEEE, Oct. 2018. DOI: 10.1109/issrew.2018.00-24.

[287] E. Novikova and I. Kotenko, "Analytical visualization techniques for security information and event management," in *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, IEEE, Feb. 2013. DOI: 10.1109/pdp.2013.84.

[288] V. Karande, E. Bauman, Z. Lin, and L. Khan, "SGX-log," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ACM, Apr. 2017. DOI: 10.1145/3052973.3053034.

[289] H. Venter and J. Eloff, "A taxonomy for information security technologies," *Computers & Security*, vol. 22, no. 4, pp. 299–307, May 2003. DOI: 10.1016/s0167-4048(03)00406-1.

[290] P. Shakarian, J. Shakarian, and K. Kashihara, "Systems and methods for vulnerability-based cyber threat risk analysis and transfer," pat. US20220078203A1, US Patent App. 17/201,869, Mar. 2022. [Online]. Available: https://patents.google.com/patent/US20220078203A1/en.

[291] K. A. Scarfone and P. M. Mell, "Guide to intrusion detection and prevention systems (IDPS)," National Institute of Standards and Technology, Tech. Rep. NIST SP 800-94, 2007. DOI: 10.6028/nist.sp.800-94.

[292] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, Jan. 2013. DOI: 10.1016/j.jnca.2012.09.004.

[293] R. Dubin, "Content disarm and reconstruction of PDF files," *IEEE Access*, vol. 11, pp. 38 399–38 416, 2023. DOI: 10.1109/access.2023.3267717.

[294] S. M. Awan, M. A. Azad, J. Arshad, U. Waheed, and T. Sharif, "A blockchain-inspired attribute-based zero-trust access control model for IoT," *Information*, vol. 14, no. 2, p. 129, Feb. 2023. DOI: 10.3390/info14020129.

[295] W. Meng, E. W. Tischhauser, Q. Wang, Y. Wang, and J. Han, "When intrusion detection meets blockchain technology: A review," *IEEE Access*, vol. 6, pp. 10 179–10 188, 2018, ISSN: 2169-3536. DOI: 10.1109/access.2018.2799854.

[296] D. He, S. Chan, X. Ni, and M. Guizani, "Software-defined-networking-enabled traffic anomaly detection and mitigation," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1890–1898, Dec. 2017. DOI: 10.1109/jiot.2017.2694702.

[297] S. S. Bhunia and M. Gurusamy, "Dynamic attack detection and mitigation in IoT using SDN," in *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, IEEE, Nov. 2017. DOI: 10.1109/atnac.2017.8215418.

[298] J. Singh and S. Behal, "Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions," *Computer Science Review*, vol. 37, p. 100 279, Aug. 2020. DOI: 10.1016/j.cosrev.2020.100279.

[299] J. Galeano-Brajones, J. Carmona-Murillo, J. F. Valenzuela-Valdés, and F. Luna-Valero, "Detection and mitigation of DoS and DDoS attacks in IoT-based stateful SDN: An experimental approach," *Sensors*, vol. 20, no. 3, p. 816, Feb. 2020. DOI: 10.3390/s20030816.

[300] S. Bhattacharjee and S. K. Das, "Building a unified data falsification threat landscape for internet of things/cyberphysical systems applications," *Computer*, vol. 56, no. 3, pp. 20–31, Mar. 2023. DOI: 10.1109/mc.2022.3198599.

[301] Y. Makino and V. Klyuev, "Evaluation of web vulnerability scanners," in *2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, IEEE, Sep. 2015. DOI: 10.1109/idaacs.2015.7340766.

[302] Z. Pokorny, *The Threat Intelligence Handbook, Second Edition*. CyberEdge Group, LLC, 2018, ISBN: 978-1-948939-06-5. [Online]. Available: https://paper.bobylive.com/Security/threat-intelligence-handbook-second-edition.pdf.

[303] S. Appala, N. Cam-Winget, D. McGrew, and J. Verma, "An actionable threat intelligence system using a publish-subscribe communications model," in *Proceedings of the 2nd ACM Workshop on Information Sharing and Collaborative Security*, ACM, Oct. 2015. DOI: 10.1145/2808128.2808131.

[304] P. Amthor, D. Fischer, W. E. Kühnhauser, and D. Stelzer, "Automated cyber threat sensing and responding," in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ACM, Aug. 2019. DOI: 10.1145/3339252.3340509.

[305] E. Bou-Harb, W. Lucia, N. Forti, S. Weerakkody, N. Ghani, and B. Sinopoli, "Cyber meets control: A novel federated approach for resilient CPS leveraging real cyber threat intelligence," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 198–204, May 2017. DOI: 10.1109/mcom.2017.1600292cm.

[306] S. H. Haji and S. Y. Ameen, "Attack and anomaly detection in IoT networks using machine learning techniques: A review," *Asian Journal of Research in Computer Science*, pp. 30–46, Jun. 2021. DOI: 10.9734/ajrcos/2021/v9i230218.

[307] I. Kotenko, I. Saenko, A. Kushnerevich, and A. Branitskiy, "Attack detection in IoT critical infrastructures: A machine learning and big data processing approach," in *2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, IEEE, Feb. 2019. DOI: 10.1109/empdp.2019.8671571.

[308] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Machine-learning techniques for detecting attacks in SDN," in *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*, IEEE, Oct. 2019. DOI: 10.1109/iccsnt47585.2019.8962519.

[309] M. M. Oo, S. Kamolphiwong, T. Kamolphiwong, and S. Vasupongayya, "Advanced support vector machine- (ASVM-) based detection for distributed denial of service (DDoS) attack on software defined networking (SDN)," *Journal of Computer Networks and Communications*, vol. 2019, pp. 1–12, Mar. 2019. DOI: 10.1155/2019/8012568.

[310] J. Ye, X. Cheng, J. Zhu, L. Feng, and L. Song, "A DDoS attack detection method based on SVM in software defined network," *Security and Communication Networks*, vol. 2018, pp. 1–8, 2018. DOI: 10.1155/2018/9804061.

[311] Z. Long, L. Tan, S. Zhou, C. He, and X. Liu, "Collecting indicators of compromise from unstructured text of cybersecurity articles using neural-based sequence labelling," in *2019 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Jul. 2019. DOI: 10.1109/ijcnn.2019.8852142.

[312] J. He, S. Li, C. Zhang, *et al.*, "Construction of security attribute portraits for power internet of things," in *2023 3rd International Symposium on Computer Technology and Information Science (ISCTIS)*, IEEE, Jul. 2023. DOI: 10.1109/isctis58954.2023.10213074.

[313] M. Saleem, M. Warsi, and S. Islam, "Secure information processing for multimedia forensics using zero-trust security model for large scale data analytics in SaaS cloud computing environment," *Journal of Information Security and Applications*, vol. 72, p. 103 389, Feb. 2023. DOI: 10.1016/j.jisa.2022.103389.

[314] S. Marchal, X. Jiang, R. State, and T. Engel, "A big data architecture for large scale security monitoring," in *2014 IEEE International Congress on Big Data*, IEEE, Jun. 2014. DOI: 10.1109/bigdata.congress.2014.18.

[315] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly, "Deep learning-based intrusion detection for IoT networks," in *2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)*, IEEE, Dec. 2019. DOI: 10.1109/prdc47002.2019.00056.

[316] O. Brun, Y. Yin, J. Augusto-Gonzalez, M. Ramos, and E. Gelenbe, "Iot attack detection with deep learning," in *ISCIS Security Workshop*, 2018. [Online]. Available: https://hal.laas.fr/hal-02062091.

[317] M. E. Aminanto, R. Choi, H. C. Tanuwidjaja, P. D. Yoo, and K. Kim, "Deep abstraction and weighted feature selection for wi-fi impersonation detection," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 3, pp. 621–636, Mar. 2018. DOI: 10.1109/tifs.2017.2762828.

[318] R. Dubin, "Content disarm and reconstruction of RTF files a zero file trust methodology," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 1461–1472, 2023. DOI: 10.1109/tifs.2023.3241480.

[319] C. Islam, M. A. Babar, and S. Nepal, "A multi-vocal review of security orchestration," *ACM Computing Surveys*, vol. 52, no. 2, pp. 1–45, Apr. 2019. DOI: 10.1145/3305268.

[320] A. Saxena, M. Lacoste, T. Jarboui, U. Lücking, and B. Steinke, "A software framework for autonomic security in pervasive environments," in *Information Systems Security*, Springer Berlin Heidelberg, 2007, pp. 91–109. DOI: 10.1007/978-3-540-77086-2_8.

[321] I. Ghanizada and A. Chuvakin, "Modernizing SOC... introducing autonomic security operations," Google, Tech. Rep., 2021. [Online]. Available: https://cloud.google.com/blog/products/identity-security/modernizing-soc-introducing-autonomic-security-operations?hl=en.

[322] M. Compastie, R. Badonnel, O. Festor, R. He, and M. Kassi-Lahlou, "A software-defined security strategy for supporting autonomic security enforcement in distributed cloud," in *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, IEEE, Dec. 2016. DOI: 10.1109/cloudcom.2016.0079.

[323] P. Krishnan, K. Jain, A. Aldweesh, P. Prabu, and R. Buyya, "OpenStackDP: A scalable network security framework for SDN-based OpenStack cloud infrastructure," *Journal of Cloud Computing*, vol. 12, no. 1, Feb. 2023. DOI: 10.1186/s13677-023-00406-w.

[324] K. Ramezanpour and J. Jagannath, "Intelligent zero trust architecture for 5g/6g networks: Principles, challenges, and the role of machine learning in the context of o-RAN," *Computer Networks*, vol. 217, p. 109 358, Nov. 2022. DOI: 10.1016/j.comnet.2022.109358.

[325] Y. Tao, Z. Lei, and P. Ruxiang, "Fine-grained big data security method based on zero trust model," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, IEEE, IEEE, Dec. 2018, pp. 1040–1045. DOI: 10.1109/padsw.2018.8644614.

[326] J. Kinyua and L. Awuah, "AI/ML in security orchestration, automation and response: Future research directions.," *Intelligent Automation & Soft Computing*, vol. 28, no. 2, pp. 527–545, 2021. [Online]. Available: login.aspx?direct=true&profile=ehost&scope=site&authtype=crawler&jrnl=10798587&AN=151028404.

[327] M. Rash, "Single packet authorization," *Linux Journal*, 2007. [Online]. Available: https://www.linuxjournal.com/article/9565.

[328] B. Campbell, J. Bradley, N. Sakimura, and T. Lodderstedt, *OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens*, RFC 8705, Feb. 2020. DOI: 10.17487/RFC8705. [Online]. Available: https://www.rfc-editor.org/info/rfc8705.

[329] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, "A survey on software defined networking with multiple controllers," *Journal of Network and Computer Applications*, vol. 103, pp. 101–118, Feb. 2018, ISSN: 1084-8045. DOI: 10.1016/j.jnca.2017.11.015.

[330] J. J. D. Rivera, T. A. Khan, W. Akbar, A. Muhammad, and W.-C. Song, "Zt&t: Secure blockchain-based tokens for service session management in zero trust networks," in *2022 6th Cyber Security in Networking Conference (CSNet)*, IEEE, IEEE, Oct. 2022, pp. 1–7. DOI: 10.1109/csnet56116.2022.9955614.

[331] M. Dabbagh, K.-K. R. Choo, A. Beheshti, M. Tahir, and N. S. Safa, "A survey of empirical performance evaluation of permissioned blockchain platforms: Challenges and opportunities," *Computers & Security*, vol. 100, p. 102 078, Jan. 2021, ISSN: 0167-4048. DOI: 10.1016/j.cose.2020.102078.

[332] I. Ahmed, T. Nahar, S. S. Urmi, and K. A. Taher, "Protection of sensitive data in zero trust model," in *Proceedings of the International Conference on Computing Advancements*, ACM, Jan. 2020, pp. 1–5. DOI: 10.1145/3377049.3377114.

[333] S. Rouhani and R. Deters, "Blockchain based access control systems: State of the art and challenges," in *IEEE/WIC/ACM International Conference on Web Intelligence*, ACM, Oct. 2019. DOI: 10.1145/3350546.3352561.

[334] N. Naik and P. Jenkins, "Securing digital identities in the cloud by selecting an apposite federated identity management from saml, oauth and openid connect," in *2017 11th International Conference on Research Challenges in Information Science (RCIS)*, IEEE, May 2017. DOI: 10.1109/rcis.2017.7956534.

[335] V. C. Hu, D. Ferraiolo, R. Kuhn, *et al.*, *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*. Jan. 2014. DOI: 10.6028/nist.sp.800-162.

[336] E. Yuan and J. Tong, "Attributed based access control (abac) for web services," in *IEEE International Conference on Web Services (ICWS'05)*, IEEE, 2005. DOI: 10.1109/icws.2005.25.

[337] Y. Li, N. S. Dhotre, Y. Ohara, T. M. Kroeger, E. Miller, and D. D. E. Long, "Horus: Fine-Grained Encryption-Based security for Large-Scale storage," in *11th USENIX Conference on File and Storage Technologies (FAST 13)*, San Jose, CA: USENIX Association, Feb. 2013, pp. 147–160. [Online]. Available: https://www.usenix.org/conference/fast13/technical-sessions/presentation/li_yan.

[338] Z. Wilcox-O'Hearn and B. Warner, "Tahoe: The least-authority filesystem," in *Proceedings of the 4th ACM international workshop on Storage security and survivability*, ser. CCS08, ACM, Oct. 2008. DOI: 10.1145/1456469.1456474.

[339] J. Wang, J. Chen, N. Xiong, O. Alfarraj, A. Tolba, and Y. Ren, "S-bds: An effective blockchain-based data storage scheme in zero-trust iot," *ACM Transactions on Internet Technology*, vol. 23, no. 3, pp. 1–23, Aug. 2023, ISSN: 1557-6051. DOI: 10.1145/3511902.

[340] Y. Zhang, R. H. Deng, S. Xu, J. Sun, Q. Li, and D. Zheng, "Attribute-based encryption for cloud computing access control: A survey," *ACM Computing Surveys*, vol. 53, no. 4, pp. 1–41, Aug. 2020, ISSN: 1557-7341. DOI: 10.1145/3398036.

[341] A. Chiquito, U. Bodin, and O. Schelen, "Attribute-based approaches for secure data sharing in industrial contexts," *IEEE Access*, vol. 11, pp. 10 180–10 195, 2023, ISSN: 2169-3536. DOI: 10.1109/access.2023.3240000.

[342] S. Wang, J. Zhou, J. K. Liu, J. Yu, J. Chen, and W. Xie, "An efficient file hierarchy attribute-based encryption scheme in cloud computing," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1265–1277, Jun. 2016, ISSN: 1556-6021. DOI: 10.1109/tifs.2016.2523941.

[343] J. Hur and D. K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 7, pp. 1214–1221, Jul. 2011, ISSN: 1045-9219. DOI: 10.1109/tpds.2010.203.

[344] Y. Jiang, W. Susilo, Y. Mu, and F. Guo, "Ciphertext-policy attribute-based encryption supporting access policy update and its extension with preserved attributes," *International Journal of Information Security*, vol. 17, no. 5, pp. 533–548, Aug. 2017, ISSN: 1615-5270. DOI: 10.1007/s10207-017-0388-7.

[345] Y. Zhang, J. Li, X. Chen, and H. Li, "Anonymous attribute-based proxy re-encryption for access control in cloud computing," *Security and Communication Networks*, vol. 9, no. 14, pp. 2397–2411, Jul. 2016, ISSN: 1939-0122. DOI: 10.1002/sec.1509.

[346] B. Lang, R. Xu, and Y. Duan, "Self-contained data protection scheme based on cp-abe," in *Communications in Computer and Information Science*. Springer Berlin Heidelberg, 2014, pp. 306–321, ISBN: 9783662447888. DOI: 10.1007/978-3-662-44788-8_18.

[347] K. Cohn-Gordon, C. Cremers, and L. Garratt, "On post-compromise security," in *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, IEEE, Jun. 2016. DOI: 10.1109/csf.2016.19.

[348] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila, "A formal security analysis of the signal messaging protocol," *Journal of Cryptology*, vol. 33, no. 4, pp. 1914–1983, Sep. 2020. DOI: 10.1007/s00145-020-09360-1.

[349] J. Alwen, S. Coretti, and Y. Dodis, "The double ratchet: Security notions, proofs, and modularization for the signal protocol," in *Advances in Cryptology – EUROCRYPT 2019*, Springer International Publishing, 2019, pp. 129–158. DOI: 10.1007/978-3-030-17653-2_5.

[350] M. Bellare, A. C. Singh, J. Jaeger, M. Nyayapati, and I. Stepanovs, "Ratcheted encryption and key exchange: The security of messaging," in *Advances in Cryptology – CRYPTO 2017*, Springer International Publishing, 2017, pp. 619–650. DOI: 10.1007/978-3-319-63697-9_21.

[351] B. Poettering and P. Rösler, "Towards bidirectional ratcheted key exchange," in *Lecture Notes in Computer Science*, Springer International Publishing, 2018, pp. 3–32. DOI: 10.1007/978-3-319-96884-1_1.

[352] J. Jaeger and I. Stepanovs, "Optimal channel security against fine-grained state compromise: The safety of messaging," in *Lecture Notes in Computer Science*, Springer International Publishing, 2018, pp. 33–62. DOI: 10.1007/978-3-319-96884-1_2.

[353] F. B. Durak and S. Vaudenay, "Bidirectional asynchronous ratcheted key agreement with linear complexity," in *Advances in Information and Computer Security*, Springer International Publishing, 2019, pp. 343–362. DOI: 10.1007/978-3-030-26834-3_20.

[354] D. Jost, U. Maurer, and M. Mularczyk, "Efficient ratcheting: Almost-optimal guarantees for secure messaging," in *Advances in Cryptology – EUROCRYPT 2019*, Springer International Publishing, 2019, pp. 159–188. DOI: 10.1007/978-3-030-17653-2_6.

[355] C. Cremers, J. Fairoze, B. Kiesl, and A. Naska, "Clone detection in secure messaging: Improving post-compromise security in practice," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20, ACM, Oct. 2020. DOI: 10.1145/3372297.3423354.

[356] B. Dowling and B. Hale, "Secure messaging authentication against active man-in-the-middle attacks," in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, Sep. 2021. DOI: 10.1109/eurosp51992.2021.00015.

[357] T. Perrin, "The XEdDSA and VXEdDSA signature schemes," Signal, Tech. Rep., 2016. [Online]. Available: https://whispersystems.org/docs/specifications/xeddsa/.

[358] N. Vatandas, R. Gennaro, B. Ithurburn, and H. Krawczyk, "On the cryptographic deniability of the signal protocol," in *Lecture Notes in Computer Science*. Springer International Publishing, 2020, pp. 188–209, ISBN: 9783030578787. DOI: 10.1007/978-3-030-57878-7_10.

[359] M. Jakobsson, K. Sako, and R. Impagliazzo, "Designated verifier proofs and their applications," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1996, pp. 143–154, ISBN: 9783540683391. DOI: 10.1007/3-540-68339-9_13.

[360] R. L. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2001, pp. 552–565, ISBN: 9783540456827. DOI: 10.1007/3-540-45682-1_32.

[361] N. Unger and I. Goldberg, "Deniable key exchanges for secure messaging," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS'15, ACM, Oct. 2015. DOI: 10.1145/2810103.2813616.

[362] N. Unger and I. Goldberg, "Improved strongly deniable authenticated key exchanges for secure messaging," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 1, pp. 21–66, Jan. 2018, ISSN: 2299-0984. DOI: 10.1515/popets-2018-0003.

[363] K. Hashimoto, S. Katsumata, K. Kwiatkowski, and T. Prest, "An efficient and generic construction for signal's handshake (x3dh): Post-quantum, state leakage secure, and deniable," *Journal of Cryptology*, vol. 35, no. 3, May 2022, ISSN: 1432-1378. DOI: 10.1007/s00145-022-09427-1.

[364] J. Brendel, R. Fiedler, F. Günther, C. Janson, and D. Stebila, "Post-quantum asynchronous deniable key exchange and the signal handshake," in *Lecture Notes in Computer Science*. Springer International Publishing, 2022, pp. 3–34, ISBN: 9783030971311. DOI: 10.1007/978-3-030-97131-1_1.

[365] T. Chen and M.-Y. Kan, "Creating a live, public short message service corpus: The NUS SMS corpus," *Language Resources and Evaluation*, Aug. 2012. DOI: 10.1007/s10579-012-9197-9.

[366] R. C. Merkle, "Secrecy, authentication, and public key systems," Ph.D. dissertation, Stanford Electronic Laboratories, 1979. [Online]. Available: https://www.ralphmerkle.com/papers/Thesis1979.pdf.

[367] S. Shim, G. Bhalla, and V. Pendyala, "Federated identity management," *Computer*, vol. 38, no. 12, pp. 120–122, Dec. 2005. DOI: 10.1109/mc.2005.408.

[368] D. W. Chadwick, "Federated identity management," in *Foundations of Security Analysis and Design V*, Springer Berlin Heidelberg, 2009, pp. 96–120. DOI: 10.1007/978-3-642-03829-7_3.

[369] H. Birkholz, D. Thaler, M. Richardson, N. Smith, and W. Pan, *Remote ATtestation procedureS (RATS) Architecture*, RFC 9334, Jan. 2023. DOI: 10.17487/RFC9334. [Online]. Available: https://www.rfc-editor.org/info/rfc9334.

[370] L. Gu, X. Ding, R. H. Deng, B. Xie, and H. Mei, "Remote attestation on program execution," in *Proceedings of the 3rd ACM workshop on Scalable trusted computing*, ACM, Oct. 2008. DOI: 10.1145/1456455.1456458.

[371] G. Coker, J. Guttman, P. Loscocco, *et al.*, "Principles of remote attestation," *International Journal of Information Security*, vol. 10, no. 2, pp. 63–81, Apr. 2011. DOI: 10.1007/s10207-011-0124-7.

[372] M. Pei, H. Tschofenig, D. Thaler, and D. Wheeler, *Trusted Execution Environment Provisioning (TEEP) Architecture*, RFC 9397, Jul. 2023. DOI: 10.17487/RFC9397. [Online]. Available: https://www.rfc-editor.org/info/rfc9397.

[373] Trusted Computing Group, *Trusted platform module library, part 1: Architecture*, Nov. 2019. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/TCG_TPM2_r1p59_Part1_Architecture_pub.pdf.

[374] A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla, "SWATT: Software-based attestation for embedded devices," in *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, IEEE, 2004. DOI: 10.1109/secpri.2004.1301329.

[375] W. Arthur, D. Challener, and K. Goldman, "Platform configuration registers," in *A Practical Guide to TPM 2.0*, Apress, 2015, pp. 151–161. DOI: 10.1007/978-1-4302-6584-9_12.

[376] S. Kent and K. Seo, *Security architecture for the internet protocol*, 2005.

[377] M. J. Freedman, E. Sit, J. Cates, and R. Morris, "Introducing tarzan, a peer-to-peer anonymizing network layer," in *Peer-to-Peer Systems*, Springer Berlin Heidelberg, 2002, pp. 121–129. DOI: 10.1007/3-540-45748-8_12.

[378] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–90, Feb. 1981. DOI: 10.1145/358549.358563.

[379] O. Berthold, H. Federrath, and M. Köhntopp, "Project "anonymity and unobservability in the internet"," in *Proceedings of the tenth conference on Computers, freedom and privacy: challenging the assumptions*, ACM, Apr. 2000, pp. 57–65. DOI: 10.1145/332186.332211.

[380] J. Al-Muhtadi, R. Campbell, A. Kapadia, M. Mickunas, and S. Yi, "Routing through the mist: Privacy preserving communication in ubiquitous computing environments," in *Proceedings 22nd International Conference on Distributed Computing Systems*, IEEE, IEEE Comput. Soc, 2002, pp. 74–83. DOI: 10.1109/icdcs.2002.1022244.

[381] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," Naval Research Lab Washington DC, Tech. Rep., 2004, https://apps.dtic.mil/sti/citations/ADA465464. [Online]. Available: https://apps.dtic.mil/sti/citations/ADA465464.

[382] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a type III anonymous remailer protocol," in *Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405)*, IEEE, IEEE Comput. Soc, 2003, pp. 2–15. DOI: 10.1109/secpri.2003.1199323.

[383] D. M. Goldschlag, M. G. Reed, and P. F. Syverson, "Hiding routing information," in *Information Hiding*, Springer Berlin Heidelberg, 1996, pp. 137–150. DOI: 10.1007/3-540-61996-8_37.

[384] J. Y. Koh, D. Leong, G. W. Peters, I. Nevat, and W.-C. Wong, "Optimal privacy-preserving probabilistic routing for wireless networks," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 9, pp. 2105–2114, Sep. 2017. DOI: 10.1109/tifs.2017.2698424.

[385] C. Dwork and K. Nissim, "Privacy-preserving datamining on vertically partitioned databases," in *Advances in Cryptology – CRYPTO 2004*, Springer Berlin Heidelberg, 2004, pp. 528–544. DOI: 10.1007/978-3-540-28628-8_32.

[386] I. Dinur and K. Nissim, "Revealing information while preserving privacy," in *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, ACM, Jun. 2003, pp. 202–210. DOI: 10.1145/773153.773173.

[387] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits," *SIAM Journal on Computing*, vol. 45, no. 3, pp. 882–929, Jan. 2016. DOI: 10.1137/14095772x.

[388] M. Abdalla, F. Bourse, A. D. Caro, and D. Pointcheval, "Simple functional encryption schemes for inner products," in *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2015, pp. 733–751. DOI: 10.1007/978-3-662-46447-2_33.

[389] A. Akavia, R. LaVigne, and T. Moran, "Topology-hiding computation on all graphs," *Journal of Cryptology*, vol. 33, no. 1, pp. 176–227, Mar. 2019, https://ia.cr/2017/296. DOI: 10.1007/s00145-019-09318-y.

[390] R. LaVigne, C.-D. Liu-Zhang, U. Maurer, T. Moran, M. Mularczyk, and D. Tschudi, "Topology-hiding computation beyond semi-honest adversaries," in *Theory of Cryptography*, Springer International Publishing, 2018, pp. 3–35. DOI: 10.1007/978-3-030-03810-6_1.

[391] Y. Zhang, T. Yan, J. Tian, Q. Hu, G. Wang, and Z. Li, "TOHIP: A topology-hiding multipath routing protocol in mobile ad hoc networks," *Ad Hoc Networks*, vol. 21, pp. 109–122, Oct. 2014. DOI: 10.1016/j.adhoc.2014.05.012.

[392] P. Rogaway, "Nonce-based symmetric encryption," in *Fast Software Encryption*, Springer Berlin Heidelberg, 2004, pp. 348–358. DOI: 10.1007/978-3-540-25937-4_22.

Load Balancing References

[393] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63–74, Aug. 2008. DOI: 10.1145/1402946.1402967.

[394] P. Patel, D. Bansal, L. Yuan, *et al.*, "Ananta: Cloud scale load balancing," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 207–218, Aug. 2013, ISSN: 0146-4833. DOI: 10.1145/2534169.2486026.

[395] D. Serrano, S. Bouchenak, Y. Kouki, *et al.*, "Sla guarantees for cloud services," *Future Generation Computer Systems*, vol. 54, pp. 233–246, Jan. 2016, ISSN: 0167-739X. DOI: 10.1016/j.future.2015.03.018.

[396] N. Dragoni, S. Giallorenzo, A. L. Lafuente, *et al.*, "Microservices: Yesterday, today, and tomorrow," in *Present and Ulterior Software Engineering*. Springer International Publishing, 2017, pp. 195–216, ISBN: 9783319674254. DOI: 10.1007/978-3-319-67425-4_12.

[397] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks (The Morgan Kaufmann Series in Computer Architecture and Design)*. Morgan Kaufmann, 2003, p. 550, ISBN: 9780122007514.

[398] A. Greenberg, J. R. Hamilton, N. Jain, *et al.*, "Vl2: A scalable and flexible data center network," in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, ser. SIGCOMM '09, ACM, Aug. 2009. DOI: 10.1145/1592568.1592576.

[399] J. Iyengar and M. Thomson, *QUIC: A UDP-Based Multiplexed and Secure Transport*, RFC 9000, May 2021. DOI: 10.17487/RFC9000. [Online]. Available: https://www.rfc-editor.org/info/rfc9000.

[400] T. Bourke, *Server load balancing*. O'Reilly, 2001, p. 175, ISBN: 0596000502.

[401] T. Barbette, C. Tang, H. Yao, *et al.*, "A High-Speed Load-Balancer design with guaranteed Per-Connection-Consistency," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, Santa Clara, CA: USENIX Association, Feb. 2020, pp. 667–683. [Online]. Available: https://www.usenix.org/conference/nsdi20/presentation/barbette.

[402] A. Aghdai, C.-Y. Chu, Y. Xu, D. H. Dai, J. Xu, and H. J. Chao, "Spotlight: Scalable transport layer load balancing for data center networks," *IEEE Transactions on Cloud Computing*, vol. 10, no. 3, pp. 2131–2145, Jul. 2022, ISSN: 2372-0018. DOI: 10.1109/tcc.2020.3024834.

[403] Z. Yao, Y. Desmouceaux, J.-A. Cordero-Fuertes, M. Townsley, and T. Clausen, "Hlb: Toward load-aware load balancing," *IEEE/ACM Transactions on Networking*, vol. 30, no. 6, pp. 2658–2673, Dec. 2022, ISSN: 1558-2566. DOI: 10.1109/tnet.2022.3177163.

[404] C. Hopps and D. Thaler, *Multipath Issues in Unicast and Multicast Next-Hop Selection*, RFC 2991, Nov. 2000. DOI: 10.17487/RFC2991. [Online]. Available: https://www.rfc-editor.org/info/rfc2991.

[405] R. Gandhi, Y. C. Hu, C.-k. Koh, H. H. Liu, and M. Zhang, "Rubik: Unlocking the power of locality and end-point flexibility in cloud scale load balancing," in *2015 USENIX Annual Technical Conference (USENIX ATC 15)*, Santa Clara, CA: USENIX Association, Jul. 2015, pp. 473–485. [Online]. Available: https://www.usenix.org/conference/atc15/technical-sessions/presentation/gandhi.

[406] R. Miao, H. Zeng, C. Kim, J. Lee, and M. Yu, "Silkroad: Making stateful layer-4 load balancing fast and cheap using switching asics," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17, ACM, Aug. 2017. DOI: 10.1145/3098822.3098824.

[407] R. Gandhi, H. H. Liu, Y. C. Hu, *et al.*, "Duet," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 27–38, Aug. 2014. DOI: 10.1145/2740070.2626317.

[408] D. E. Eisenbud, C. Yi, C. Contavalli, *et al.*, "Maglev: A fast and reliable software network load balancer," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, 2016, pp. 523–535.

[409] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, "Consistent hashing and random trees," in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing - STOC '97*, ACM Press, 1997. DOI: 10.1145/258533.258660.

[410] J. T. Araujo, L. Saino, L. Buytenhek, and R. Landa, "Balancing on the edge: Transport affinity without network state," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, Renton, WA: USENIX Association, Apr. 2018, pp. 111–124. [Online]. Available: https://www.usenix.org/conference/nsdi18/presentation/araujo.

[411] V. Olteanu, A. Agache, A. Voinescu, and C. Raiciu, "Stateless datacenter load-balancing with beamer," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, Renton, WA: USENIX Association, Apr. 2018, pp. 125–139. [Online]. Available: https://www.usenix.org/conference/nsdi18/presentation/olteanu.

[412] V. Olteanu and C. Raiciu, "Datacenter scale load balancing for multipath transport," in *Proceedings of the 2016 workshop on Hot topics in Middleboxes and Network Function Virtualization*, ser. SIGCOMM '16, ACM, Aug. 2016. DOI: 10.1145/2940147.2940154.

[413] S. Shi, Y. Yu, M. Xie, *et al.*, "Concury: A fast and light-weight software cloud load balancer," in *Proceedings of the 11th ACM Symposium on Cloud Computing*, ser. SoCC '20, ACM, Oct. 2020. DOI: 10.1145/3419111.3421279.

[414] R. Cohen, M. Kadosh, A. Lo, and Q. Sayah, "Lb scalability: Achieving the right balance between being stateful and stateless," *IEEE/ACM Transactions on Networking*, vol. 30, no. 1, pp. 382–393, Feb. 2022, ISSN: 1558-2566. DOI: 10.1109/tnet.2021.3112517.

[415] J. Zhou, M. Tewari, M. Zhu, *et al.*, "Wcmp: Weighted cost multipathing for improved fairness in data centers," in *Proceedings of the Ninth European Conference on Computer Systems*, ser. EuroSys 2014, ACM, Apr. 2014. DOI: 10.1145/2592798.2592803.

[416] J. Zhang, S. Wen, J. Zhang, *et al.*, "Fast switch-based load balancer considering application server states," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1391–1404, Jun. 2020, ISSN: 1558-2566. DOI: 10.1109/tnet.2020.2981977.

[417] R. Gandhi, Y. C. Hu, and M. Zhang, "Yoda," in *Proceedings of the Eleventh European Conference on Computer Systems*, ACM, Apr. 2016. DOI: 10.1145/2901318.2901352.

[418] A. Aghdai, M. I.-C. Wang, Y. Xu, C. H.-P. Wen, and H. J. Chao, "In-network congestion-aware load balancing at transport layer," in *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, IEEE, Nov. 2019. DOI: 10.1109/nfv-sdn47374.2019.9040109.

[419] M. Mitzenmacher, "Compressed bloom filters," in *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*, ser. PODC01, ACM, Aug. 2001. DOI: 10.1145/383962.384004.

[420] D. Simon, "Kalman filtering," in *Embedded Systems Programming*, Jun. 2001, pp. 72–79. [Online]. Available: https://abel.math.harvard.edu/archive/116_fall_03/handouts/kalman.pdf.

[421] F. Németh, M. Chiesa, and G. Rétvári, "Normal forms for match-action programs," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, ser. CoNEXT '19, ACM, Dec. 2019. DOI: 10.1145/3359989.3365417.

[422] D. Borman, R. T. Braden, V. Jacobson, and R. Scheffenegger, *TCP Extensions for High Performance*, RFC 7323, Sep. 2014. DOI: 10.17487/RFC7323. [Online]. Available: https://www.rfc-editor.org/info/rfc7323.

[423] B. Pit-Claudel, Y. Desmouceaux, P. Pfister, M. Townsley, and T. Clausen, "Stateless load-aware load balancing in p4," in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, IEEE, Sep. 2018. DOI: 10.1109/icnp.2018.00058.

[424] C. Rizzi, Z. Yao, Y. Desmouceaux, M. Townsley, and T. Clausen, "Charon: Load-aware load-balancing in p4," in *2021 17th International Conference on Network and Service Management (CNSM)*, IEEE, Oct. 2021. DOI: 10.23919/cnsm52442.2021.9615535.

[425] Y. Desmouceaux, P. Pfister, J. Tollet, M. Townsley, and T. Clausen, "6lb: Scalable and application-aware load balancing with segment routing," *IEEE/ACM Transactions on Networking*, vol. 26, no. 2, pp. 819–834, Apr. 2018, ISSN: 1558-2566. DOI: 10.1109/tnet.2018.2799242.

[426] M. Mitzenmacher, "The power of two choices in randomized load balancing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 10, pp. 1094–1104, 2001, ISSN: 1045-9219. DOI: 10.1109/71.963420.

[427] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, *Segment Routing Architecture*, RFC 8402, Jul. 2018. DOI: 10.17487/RFC8402. [Online]. Available: https://www.rfc-editor.org/info/rfc8402.

[428] M. Zukerman, T. Neame, and R. Addie, "Internet traffic modeling and future technology implications," in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, ser. INFCOM-03, IEEE, 2003. DOI: 10.1109/infcom.2003.1208709.

[429] I. A. Kaisina, D. S. Vasiliev, A. V. Abilov, A. E. Kaisin, D. D. Meitis, and A. I. Nistyuk, "Ns-3 simulation of poisson-pareto burst process in multi-source fanet scenario with network coding," in *2020 Moscow Workshop on Electronic and Networking Technologies (MWENT)*, IEEE, Mar. 2020. DOI: 10.1109/mwent47943.2020.9067512.

Publications submitted during the PhD

[430] A. Poirrier, L. Cailleux, and T. H. Clausen, "Is trust misplaced?" Submitted to *Proceedings of the IEEE*, 2024.

[431] A. Poirrier, L. Cailleux, and T. H. Clausen, "Building a zero trust federation," Submitted to *IEEE Journal on Selected Areas in Communications*, 2024.

[432] A. Poirrier, Laurent, and T. H. Clausen, "Data-centric security protections in zero trust architectures," Submitted to *MILCOM 2024 - 2024 IEEE Military Communications Conference (MILCOM)*, 2024.

[433] A. Poirrier and T. H. Clausen, "Privacy-preserving forwarding," *Under submission*, 2024.

[434] A. Poirrier and T. H. Clausen, "Top-of-rack-assisted load-aware and server-agnostic load-balancing," Submitted to *2024 IEEE 32nd International Conference on Network Protocols (ICNP)*, 2024.

ECOLE
DOCTORALE

Titre : Sécurité Formelle des Architectures Zéro Confiance

Mots clés : Architecture, Cryptographie, Cybersécurité, Sécurité des Réseaux, Zéro Confiance

Résumé : La sécurité des systèmes d'information repose traditionnellement sur le modèle de *sécurité du périmètre*, qui compartimentalise le réseau en différents périmètres isolés qui regroupent les ressources. Un système accède à un périmètre en s'authentifiant, et une fois au sein d'un périmètre, il est implicitement considéré comme étant *de confiance*, jouissant d'un accès non restreint à toutes les ressources de ce périmètre. Cependant, de nombreuses attaques envers les architectures périmétriques ont montré les limites de cette confiance. De plus, la transformation des systèmes d'information, par exemple l'émergence des services en nuage (*cloud services*), le télétravail ou l'usage de prestataires, a remis en cause la vision d'un réseau monolithique de confiance. Ainsi, un nouveau paradigme de sécurité a émergé, appelé *zéro confiance*. Fondé sur le principe « ne jamais faire confiance, toujours vérifier », ce paradigme transforme la notion de périmètre, en établissant un ensemble de principes de sécurité reposant sur une autorisation contextuelle et dynamique. Cependant, mettre en œuvre une architecture zéro confiance pose de nombreux défis, en particulier car il n'existe pas de définition claire du paradigme zéro confiance.
Dans ce contexte, cette thèse explore comment développer un cadre pratique et formel pour raisonner sur la sécurité des systèmes d'information. Tout d'abord, une étude approfondie du modèle zéro confiance est conduite, puis une taxonomie des technologies et architectures zéro confiance existantes est établie, ce qui permet une compréhension exhaustive du paradigme zéro confiance. Cela mène à la création d'une méthode d'évaluation des architectures, qui permet également d'identifier des lacunes dans la recherche sur le zéro confiance. Cette thèse propose plusieurs contributions pour combler ces lacunes, afin d'améliorer l'état de l'art pour le zéro confiance. Ces améliorations sont intégrées au sein d'un démonstrateur d'architecture zéro confiance, illustrant comment une architecture zéro confiance peut être complémentée par de nouvelles technologies pour améliorer sa maturité. Enfin, cette thèse prend du recul et évalue comment le paradigme zéro confiance répond aux problèmes de sécurité auxquels font face les entreprises, démontrant qu'il n'est pas suffisant seul pour protéger les données et services sensibles.

Title : Formal Security of Zero Trust Architectures

Keywords : Architecture, Cryptography, Cybersecurity, Network Security, Zero Trust

Abstract : The security architecture of Information Technology (IT) systems has traditionally been based on the *perimeter security* model, in which resources are grouped into perimeters isolated through network mechanisms, and devices are authenticated to access perimeters. Once within a perimeter, devices are *implicitly trusted*, and enjoy unrestricted access to resources within that perimeter. However, history has shown that such trust is misplaced, as numerous security threats and successful attacks against perimeter-based architectures have been documented. Furthermore, the emergence of new IT usages, such as cloud services, work-from-home, and service providers and subsidiaries relationships, has challenged the relevance of considering a monolithic, trusted network for accessing resources. These considerations have led to the emergence of a novel security paradigm, called by *zero trust*. Founded on the principle 'never trust, always verify', this approach transforms the notion of perimeter and establishes a set of security principles that prioritize context-aware and dynamic authorization. Nevertheless, implementing zero trust poses significant challenges, due to a lack of clear guidelines for defining zero trust.
In this context, this thesis investigates whether, and how, it is possible to develop a practical and formal framework for reasoning about the security of IT architectures. First, a thorough survey of zero trust is conducted, and a taxonomy of existing zero trust technologies and architectures is developed, enabling a comprehensive understanding of zero trust. This leads to the development of an evaluation framework, that is used to identify gaps within zero trust research. This thesis provides contributions aiming to address some of these gaps, for enhancing the state of zero trust technology development. These improvements are integrated into a proof-of-concept zero trust architecture implemented for this thesis, illustrating a method for extending an existing zero trust architecture. Finally, the thesis takes a step back, and evaluates the extent to which the zero trust framework addresses real-world problems, demonstrating that the zero trust framework alone is not sufficient for protecting sensitive data and services.

Institut Polytechnique de Paris
91120 Palaiseau, France