Network Security Knowledge Area Version 2.0.0

Christian Rossow | CISPA Helmholtz Center for Information Security
Sanjay Jha | University of New South Wales

EDITOR
Andrew Martin | Oxford University

REVIEWERS
Amir Herzberg | University of Connecticut
Matthias Hollick | Technical University of Darmstadt
Sanjay Jha | University of New South Wales
Shishir Nagaraja | University of Strathclyde
Gene Tsudik | University of California

COPYRIGHT

CHANGELOG

Version date	Version number	Changes made
July 2021	2.0	Major revision with restructuring to place the network protocol discussion in a wider context and mention of a broader range of network architectures.
October 2019	1.0	

INTRODUCTION

The ubiquity of networking allows us to connect all sorts of devices and gain unprecedented access to a whole range of applications and services anytime, anywhere. However, our heavy reliance on networking technology also makes it an attractive target for malicious users who are willing to compromise the security of our communications and/or cause disruption to services that are critical for our day-to-day survival in a connected world. In this chapter, we will explain the challenges associated with securing a network under a variety of attacks for a number of networking technologies and widely used security protocols, along with emerging security challenges and solutions. This chapter aims to provide the necessary background in order to understand other knowledge areas, in particular the Security Operations & Incident Management CyBOK Knowledge Area [1] which takes a more holistic view of security and deals with operational aspects. An understanding of the basic networking protocol stack and popular network protocols is assumed. Standard networking text books explain the fundamentals of the layered Internet Protocol suite [2, 3].

This chapter is organized as follows. In Section 1, we lay out the foundations of this chapter and define security goals in networked systems. As part of this, we also outline attackers and their capabilities that threaten these goals. In Section 2, we describe six typical networking scenarios that nicely illustrate why security in networking is important, and achieving it can be non-trivial. We then discuss the security of the various networking protocols in Section 3, structured by the layered architecture of the Internet protocol stack. In Section 4, we present and discuss several orthogonal network security tools such as firewalls, monitoring and Software Defined Networking (SDN). We complete this chapter with a discussion on how to combine the presented mechanisms in Section 5.

CONTENT

1 SECURITY GOALS AND ATTACKER MODELS

[2, c8] [4, c1] [5, c1] [6, c6] [3, c8]

We want and need secure networks. But what does *secure* actually mean? In this chapter, we define the fundamentals of common security goals in networking. Furthermore, we discuss capabilities, positions and powers of attackers that aim to threaten these security goals.

1.1 Security Goals in Networked Systems

When designing networks securely, we aim for several orthogonal *security goals* [4]. The most commonly used security goals are summarized in the CIA triad: confidentiality, integrity and availability. *Confidentiality* ensures that untrusted parties cannot leak or infer sensitive information from communication. For example, in a confidential email communication system, (i) only the sender and recipient of an email can understand the email content, not anyone on the communication path (e.g., routers or email providers), and (ii) no one else ought to learn that email was sent from the sender to the recipient. *Integrity* ensures that untrusted parties cannot alter information without the recipient noticing. Sticking to our email example, integrity guarantees that any in-flight modification to an email (e.g., during its submission, or on its way between email providers) will be discovered as such by the recipient. Finally, *availability* ensures that data and services should be accessible by their designated users all the time. In our email scenario, a Denial of Service (DoS) attacker may aim to threaten the availability of email servers in order to prevent or delay email communication.

Next to the CIA triad, there are more subtle security goals, not all of which apply in each and every application scenario. *Authenticity* is ensured if the recipient can reliably attribute the origin of communication to the sender. For example, an email is authentic if the recipient can ensure that the claimed sender actually sent this email. *Non-repudiation* extends authenticity such that we can prove authenticity to arbitrary third parties, i.e., allowing for public verification. In our email scenario, non-repudiation allows the email recipient to prove to anyone else that a given email stems from a given sender. *Anonymity* means that communication cannot be traced back to its sender (sender anonymity) and/or recipient (recipient anonymity). For example, if an attacker sends a spoofed email that cannot be reliably traced back to its actual sender (e.g., the correct personal identity of the attacker), it is anonymous. There are further privacy-related guarantees such as *unlinkability* that go beyond the scope of this chapter and are defined in the Privacy & Online Rights CyBOK Knowledge Area [7].

To achieve security goals, we will heavily rely on cryptographic techniques such as public and symmetric keys for encryption and signing, block and stream ciphers, hashing, and digital signature, as described in the Cryptography CyBOK Knowledge Area [8] and the Applied Cryptography CyBOK Knowledge Area [9]. Before showing how we can use these techniques for secure networking, though, we will discuss attacker models that identify capabilities of possible attackers of a networked system.

1.2 Attacker Models

Attacker models are vital to understand the security guarantees of a given networked system. They define the capabilities of attackers and determine their access to the network.

Often, the Dolev-Yao [10] attacker model is used for a formal analysis of security protocols in the research literature. The Dolev-Yao model assumes that an attacker has complete control over the entire network, and concurrent executions of the protocol between the same set of two or more parties can take place. The Dolev-Yao model describes the worst possible attacker: an attacker that sees all network communication, allowing the attacker to read any message, prevent or delay delivery of any message, duplicate any message, or otherwise synthesise any message for which the attacker has the relevant cryptographic keys (if any).

Depending on the context, real attackers may have less power. For example, we can distinguish between *active* and *passive* attackers. Active attackers, like Dolev-Yao, can manipulate packets. In contrast, passive attackers (*eavesdroppers*) can observe but not alter network communication. For example, an eavesdropper could capture network traffic using packet sniffing tools in order to extract confidential information such as passwords, credit card details and many other types of sensitive information from unprotected communication. But even if communication is encrypted, attackers may be able to leverage communication patterns statistics to infer sensitive communication content ("traffic analysis", see Section 3.1.6).

We furthermore distinguish between *on-path* and *off-path* attackers. The prime example for an on-path attacker is a person-in-the-middle (PITM), where an attacker is placed between two communication parties. In contrast, off-path attackers can neither see nor directly manipulate the communication between parties. Still, off-path attackers can cause severe harm. For example, an off-path attacker could spoof TCP packets aiming to maliciously terminate a suspected TCP connection between two parties. Similarly, off-path attackers could abuse high-bandwidth links and forge Internet Protocol (IP) headers (IP spoofing, see Section 3.2.4) to launch powerful and anonymous Denial of Service (DoS) attacks. Using forged routing protocol message, off-path attackers may even try to become on-path attackers.

In addition, the position of attackers heavily influences their power. Clearly, a single Internet user has less power than an entire rogue Internet Service Provider (ISP). The single user can leverage their relatively small bandwidth to launch attacks, while an ISP can generally also sniff on and alter communication, abuse much larger bandwidths, and correlate traffic patterns. Then again, as soon as attackers aggregate the power of many single users/devices (e.g., in form of a botnet), their overall power amplifies. Attackers could also be in control of certain Internet services, routers, or any combination thereof. We also distinguish between insider and outsider attackers, which are either inside or outside of a trusted domain, respectively.

Overall, we model (i) where attackers can be positioned, (ii) who they are, and (iii) which capabilities they have. Unfortunately, in strong adversarial settings, security guarantees diminish way too easily. For example, strong anonymity may not hold against state actors who can (theoretically) control major parts of the Internet, such as Tier-1. Similarly, availability is hard to maintain for spontaneous and widely distributed DoS incidents.

2 NETWORKING APPLICATIONS

[2, c1] [3, c1]

We now turn to a selection of popular concrete networking applications that help us illustrate why achieving security is far from trivial. This chapter is less about providing solutions—which are discussed in the subsequent two chapters—but more about outlining how security challenges differ by application and context. To this end, we will introduce various networking applications, starting with typical local networks and the Internet, and continuing with similarly ubiquitous architectures such as bus networks (e.g., for cyber-physical system), fully-distributed networks, wireless networks, and finally, Software Defined Networking (SDN).

2.1 Local Area Networks (LANs)

Arguably, a Local Area Network (LAN) is the most intuitive and by far predominant type of network with the (seemingly) lowest security requirements. Local networks connect systems within an internal environment, shaping billions of home networks and corporate networks alike. A typical fallacy of LAN operators is to blindly trust their network. However, the more clients (can possibly) connect to a LAN, the harder it is to secure a LAN environment. In fact, the clients themselves may undermine the assumed default security of a LAN. For example, without further protection, existing clients can find and access unauthorized services on a LAN, and possibly exfiltrate sensitive information out of band. This becomes especially problematic if the clients are no longer under full control of the network/system operators. A prime example is the recent rise of the Bring-Your-Own-Device (BYOD) principle, which allows employees to integrate their personal untrusted devices into corporate networks—in the worst case, even creating network bridges to the outside world. Similarly, attackers may be able to add malicious clients, such as by gaining physical access to network plugs. Finally, attackers may impersonate other LAN participants by stealing their loose identity such as publicly-known hardware addresses (e.g., cloning MAC addresses in Ethernet).

Consequently, even though a LAN is conceptually simple, we still have uncertainties regarding LAN security: Can we control which devices become part of a network to exclude untrusted clients and/or device configurations? (Sections 3.4.1 and 4.5) Can we monitor their actions to identify attackers and hold them accountable? (Section 4.3) Can we partition larger local networks into multiple isolated partitions to mitigate potential damage? (Section 3.4.5)

2.2 Connected Networks and the Internet

Securing communication becomes significantly more challenging when connecting local networks. The most typical scenario is connecting a LAN to the Internet, yet also LAN-to-LAN communication (e.g., two factories part of a joint corporate network using a Virtual Private Network (VPN)) is common. With such connected networks, we suddenly face an insecure end-to-end channel between the networks which is no longer in control of blindly trusted network operators. When communicating over the Internet, the traffic will pass several Autonomous Systems (ASs), each of which can in principle eavesdrop and manipulate the communication. Worse, senders typically have little to no control over the communication paths. In fact, even ASs cannot fully trust all paths in their routing table. Without further precautions, any other (misbehaving) AS on the Internet can reroute a target's network traffic by hijacking Internet routes. For example, a state actor interested in sniffing on the communication sent to a particular network can send malicious route announcements to place itself as PITM between all Internet client and the target network. Finally, especially corporate networks may choose to "include" third-party services into their network, such as data centers for off-site storage or clouds for off-site computations. Suddenly, external networks are integrated into corporate networks, and organizations may send sensitive data to servers outside of their control.

This leads to several questions: Can two parties securely communicate over an insecure channel, i.e., with guaranteed confidentiality and integrity? (Section 3.2.1) How should we securely connect networks, e.g., in a corporate setting? (Section 3.3.1) Can we ensure that the underlying routing is trusted, or at least detect attacks? (Section 3.3.3)

2.3 Bus Networks

Cyber-physical systems regularly use bus networks to communicate. For example, industrial control systems may use Modbus [11] to exchange status information and steer cyber-physical devices. Home automation networks (e.g., Konnex Bus (KNX) [12]) allow to sense inputs in houses (temperature, brightness, activity) which in turn govern actuators (e.g., heating/cooling, light, doors). Vehicular networks (e.g., Controller Area Network (CAN) [13]) connect dozens if not hundreds of components that frequently communicate, e.g., sensors such as a radar to control actuators such as the engine or brakes. All these bus networks typically also form local networks, yet with subtle differences to the aforementioned LANs. Frequently, bus clients require real-time guarantees for the arrival and processing time of data. Clearly, even safety is affected if the signal of a car's brake pedal arrives "too late" at its destination (the brake system). This need for real-time guarantees are often in direct conflict to simple security demands such as authenticity that may add "expensive" computations. Furthermore, by design, bus networks introduce a shared communication medium, which allows clients to both, sniff and manipulate communication. On top, many of the bus protocols were designed without paying particular attention to security, partially because they predate security best practices we know today. Finally, bus clients can have limited computing resources, having only limited capability to perform expensive security operations.

We summarize our discussion with a set of questions on bus security: Can we retrospectively add security to unprotected bus networks without breaking compatibility? Is it possible to gain security without violating real-time guarantees? (Section 3.4.7) Which additional mechanisms can we use to reduce the general openness of bus networks? (Section 3.4.5)

2.4 Wireless Networks

When moving from wired to wireless networks, we do not necessarily face fundamentally new threats, but increase the likelihood of certain attacks to occur. In particular, wireless networks (e.g., Wireless LAN) are prone to eavesdropping due to the broadcast nature of their media. Similarly, while simple physical access control may still partially work to protect a cable-connected LAN network, it fails terribly for wireless networks that—by construction—have no clear boundary for potential intruders. For example, walls around a house do not stop signals from and to wireless home networks, and without further protection, attackers can easily join and sniff on networks. In wireless settings, we thus have to pay particular attention to both access control and secure communication (including securing against traffic analysis). As it turns out, though, defining secure wireless standards is far from trivial, and several wireless and cellular standards have documented highly-critical vulnerabilities or design flaws.

To better understand their deficiencies, we will study the following questions: How can we enforce access control in wireless networks, while not sacrificing usability? How can we prevent eavesdropping attacks in wireless communication? (Section 3.4.6)

2.5 Fully-Distributed Networks: DHTs and Unstructured P2P Networks

Centralized client-server architectures have clear disadvantages from an operational point of view. They have a single point of failure, and require huge data centers in order to scale to many clients. Furthermore, centralized architectures have a central entity that can control or deliberately abandon them. Fully-distributed networks provide scalability and resilience by design. Not surprisingly, several popular networks chose to follow such distributed designs, including cryptocurrencies such as Bitcoin or file sharing networks. Decentralized network can be roughly grouped in two types. Distributed Hash Tables (DHTs) such as Kademlia [14] and Freenet [15] have become the de-facto standard for *structured* Peer-to-Peer (P2P) networks, and offer efficient and scalable message routing to millions of peers. In *unstructured* P2P networks, peers exchange data hop-by-hop using gossip protocols. Either way, P2P networks are designed to *welcome* new peers at all times. This lack of peer authentication leads to severe security challenges. A single entity may be able to aggressively overpopulate a network if no precautions are taken. DHTs further risk that attackers can potentially disrupt routing towards certain data items. Similarly, gossip networks risk that single entities flood the network with malicious data. Finally, due to the open nature of the networks, we also face privacy concerns, as network participants can infer which data items a peer retrieves.

All in all, fully-distributed networks raise fundamentally different security concerns: Can we reliably route data (or data requests) to their target? Can we authenticate peers even in distributed networks? What are the security implications of storing data in and retrieving data from a publicly accessible network? (Section 3.1.5)

2.6 Software-Defined Networking and Network Function Virtualisation

Software Defined Networking (SDN) is our final use case. SDN is strictly speaking not a networking application, but more of a technology to enable for dynamic and efficient network configuration. Yet it concerns similarly many security implications as other applications do. SDN aims to ease network management by decoupling packet forwarding (data plane) and packet routing (control plane). This separation and the underlying flow handling have enabled for drastic improvements from a network management perspective, especially in highly-dynamic environments such as data centers. The concept of Network Functions Virtualisation (NFV) complements SDN, and allows to virtualize network node functions such as load balancers or firewalls.

We will revisit SDN by discussing the following questions: How can SDN help in network designs and better monitoring? Can NFV help securing networks by virtualizing security functions? Are there new, SDN-specific threats? (Section 4.4)

3 NETWORK PROTOCOLS AND THEIR SECURITY

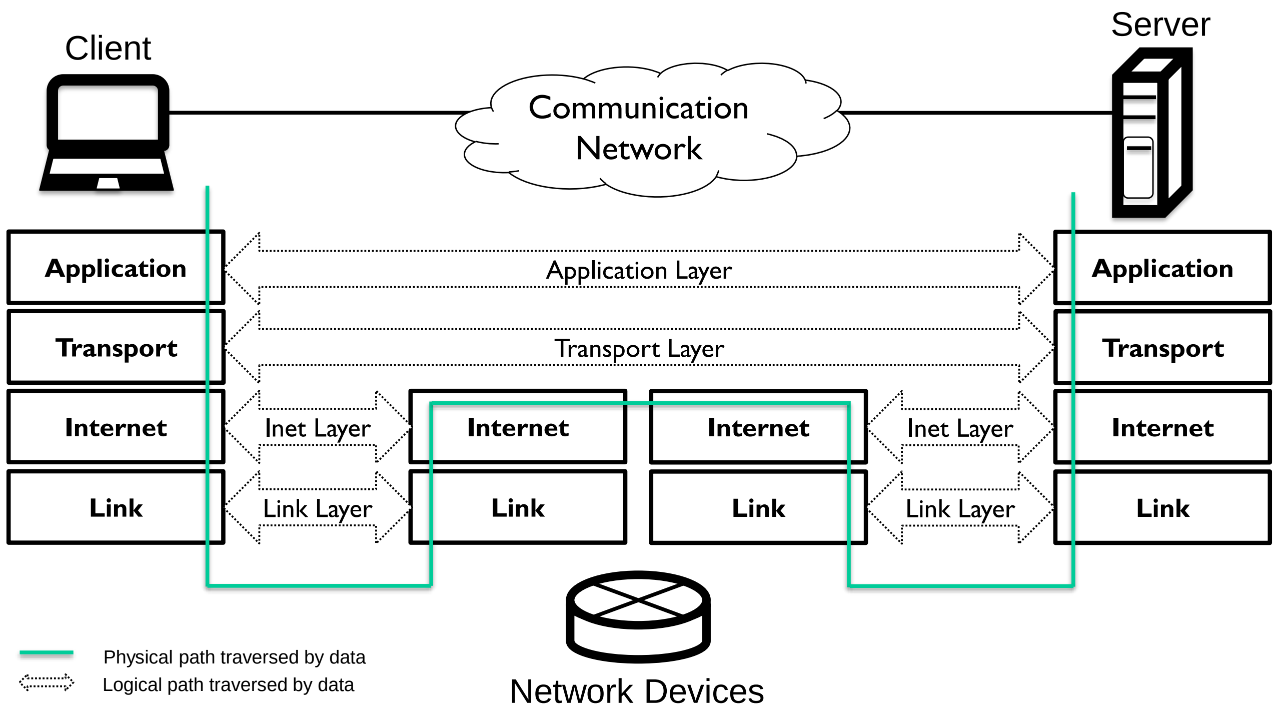

Figure 1: Four-Layer Internet Protocol Stack (a.k.a. *TCP/IP stack*).

After having introduced several networking applications, we now turn to the security of networking protocols. To guide this discussion, we stick to a layered architecture that categorizes protocols and applications. Indeed, a complex system such as distributed applications running over a range of networking technologies is best understood when viewed as layered architecture. Figure 1 shows the 4-layer Internet protocol suite and the interaction between the various layers. For each layer, we know several network protocols—some of which are quite generic, and others that are tailored for certain network architectures. The Internet is the predominant architecture today, and nicely maps to the TCP/IP model. It uses the Internet Protocol (IP) (and others) at the Internet layer, and UDP/TCP (and others) at the transport layer—hence, the Internet protocol suite is also known as the *TCP/IP stack*.

Other networking architectures such as automotive networks use completely different sets of protocols. Not always it is possible to directly map their protocol to the layered architecture of the TCP/IP model. Consequently, more fine-grained abstractions such as the ISO/OSI model extend this layered architecture. For example, the ISO/OSI model splits the link layer into two parts, namely the data link layer (node-to-node data transfer) and the physical layer (physical transmission and reception of raw data). The ISO/OSI model defines a network layer instead of an Internet layer, which is more inclusive to networks that are not connected to the Internet. Finally, ISO/OSI defines two layers below the application layer (presentation and session).

The vast majority of topics covered in this chapter do not need the full complexity of the ISO/OSI model. In the following, we therefore describe the security issues and according countermeasures at each layer of the TCP/IP model. We thereby follow a top-down approach, starting with application-layer protocols, and slowly going down to the lower layers until the link layer. Whenever possible, we abstract from the protocol specifics, as many discussed network security principles can be generically applied to other protocols.

3.1 Security at the Application Layer

[2, c8] [5, c6,c15,c19–c22] [3, c8]

We first seek to answer how application protocols can be secured, or how application-layer protocols can be leveraged to achieve certain security guarantees.

3.1.1 Email and Messaging Security

As a first example of an application-layer security protocol, we will look at secure email. Given its age, *the* protocol for exchanging emails, Simple Mail Transfer Protocol (SMTP), was not designed with security in mind. Still, businesses use email even now. Communication parties typically want to prevent others from reading (confidentiality) or altering (integrity) their emails. Furthermore, they want to verify the sender's identity when reading an email (authenticity). Schemes like Pretty Good Privacy (PGP) and Secure Multipurpose Internet Mail Extensions (SMIME) provide such end-to-end security for email communication. Their basic idea is that each email user has their own private/public key pair–see the Cryptography CyBOK Knowledge Area [8] for the cryptographic details, and Section 3.2.2 for a discussion how this key material can be shared. The sender signs the hash of a message using the sender's private key, and sends the hash along with the (email) message to the recipient. The recipient can then validate the email's signature using the sender's public key. Checking this signature allows for an integrity check and authentication at the same time, as only the sender knows their private key. Furthermore, this scheme provides non-repudiation as it can be publicly proved that the hash (i.e., the message) was signed by the sender's private key. To gain confidentiality, the sender encrypts the email before submission using "hybrid encryption". That is, the sender creates a fresh symmetric key used for message encryption, which is significantly faster than using asymmetric cryptography. The sender then shares this symmetric key with the recipient, encrypted under the recipient's public key.

This very same scheme can be applied to other client-to-client communication. For example, instant messengers (e.g., WhatsApp, Threema or Signal) or video conference systems can use this general principle to achieve the same end-to-end guarantees. One remaining challenge for such strong guarantees to hold is that user identities (actually, their corresponding key material) have to be reliably validated [16]. The Applied Cryptography CyBOK Knowledge Area [9] has more details.

Not all email users leverage such client-to-client security schemes, though. Both PGP and SMIME have usability challenges (e.g., key distribution, difficulty of indexed searches, etc.) that hamper wide adoption [17]. To address this issue, we can secure mail protocols (SMTP, but also Internet Message Access Protocol (IMAP) and Post Office Protocol (POP)) with the help of TLS (see Section 3.2.1). By wrapping them in TLS, we at least achieve hop-by-hop security, e.g., between client and their mail submission server or between mail servers during email transfer. Consequently, we can protect email submission, retrieval and transport from on-path adversaries. However, even though communication is protected hop-by-hop, curious mail server operators can see emails in plain. Only end-to-end security schemes like PGP/SMIME protect against untrusted mail server operators.

There are other challenges to secure email, such as phishing and spam detection, which are described in depth in the Adversarial Behaviours CyBOK Knowledge Area [18].

3.1.2 Hyper Text Transfer Protocol Secure (HTTPS)

The most prominent application-layer protocol, the Hypertext Transfer Protocol (HTTP), was designed without any security considerations. Yet, the popularity of HTTP and its unprecedented adoption for e-commerce imposed strict security requirements on HTTP later on. Its secure counterpart HTTPS wraps HTTP using a security protocol at the transport layer (TLS, see Section 3.2.1), which can be used to provide confidentiality and integrity for the entire HTTP communication—including URL, content, forms and cookies. Furthermore, HTTPS allows clients to implicitly authenticate web servers using certificates. HTTPS is described in much greater detail in the Web & Mobile Security CyBOK Knowledge Area [19].

3.1.3 DNS Security

In its primary use case, the Domain Name System (DNS) translates host names to their corresponding IP addresses. A hierarchy of authoritative name servers (NSs) maintain this mapping. Resolving NSs (*resolvers*) iteratively look up domain names on behalf of clients. In such an iterative lookup, the resolver would first query the root NSs, which then redirect the resolver to NSs lower in the DNS hierarchy, until the resolver contacts an NS that is authoritative for the queried domain. For example, in a lookup for a domain `sub.example.com`, a root NS would redirect the resolver to a NS that is authoritative for the `.com` zone, which in turn tell the resolver to contact the NS authoritative for `*.example.com`. To speed up these lookups, resolvers cache DNS records according to a lifetime determined by their authoritative NSs. To minimize privacy leaks towards NSs in the upper hierarchy, resolvers can minimize query names such that NSs higher up in the hierarchy do not learn the fully-qualified query name [20].

Unfortunately, multiple attacks aim to abuse the lack of authentication in plain text DNS. A PITM attacker can impersonate a resolver, return bogus DNS records and divert traffic to a malicious server, thus allowing them to collect user passwords and other credentials. In a DNS cache poisoning attack, adversaries aim to implant bogus name records, thus diverting a user's traffic towards the target domain to attacker-controlled servers. Learning from these attacks, the IETF introduced the DNS Security Extensions (DNSSEC). DNSSEC allows authoritative name servers to sign DNS records using their private key. The authenticity of the DNS records can be verified by a requester using the corresponding public key. In addition, a digital signature provides integrity for the response data. The overall deployment of DNSSEC at the top-level domain root name servers—a fundamental requirement to deploy DNSSEC at lower levels in the near future—steadily increases [21].

DNSSEC explicitly does not aim to provide confidentiality, i.e., DNS records are still communicated unencrypted. DNS over TLS (DoT) and DNS over HTTPS (DoH) address this problem. They provide end-to-end security between the DNS client and its chosen resolver, by tunneling DNS via secure channels, namely TLS (see Section 3.2.1) or HTTPS (see Section 3.1.2), respectively. More and more popular Web browsers (e.g., Chrome, Firefox) enable DoH by default, using selected resolvers preconfigured by the browser vendors. This resulted in a massive centralization of DNS traffic towards just a few resolvers received. Such a centralization puts resolvers in a quite unique position with the power of linking individual clients (by IP addresses) to their lookups. Oblivious DNS Over HTTPS (ODoH) addresses this issue by adding trusted proxies between DNS clients and their chosen resolvers [22].

Irrespective of these security protocols, resolvers are in a unique situation to monitor name resolutions of their clients. Resolver operators can leverage this in order to protect clients by offering some sort of blocklist of known "misbehaving" domains which have a bad reputation.

Such DNS filtering has the potential to mitigate cyber threats, e.g., by blocking phishing domains or command & control domains of known malware variants.

Finally, DNS is prone to Distributed Denial of Service (DDoS) attacks [23]. DNS authoritative servers can be targeted by NXDOMAIN attacks, in which an IP-spoofing client looks up many unassigned subdomains of a target domain at public (open) resolvers. Subdomains are typically chosen at random and are therefore not cached, hence sometimes referred to as *random subdomain attack*. Consequently, the resolvers have to forward the lookups and hence flood the target authoritative name server. In another type of DDoS attack, DNS servers (both resolvers and authoritatives) are regularly abused for amplification DDoS attacks, in which they reflect IP-spoofed DNS requests with significantly larger responses [24]. Reducing the number of publicly-reachable open DNS resolvers [25] and DNS rate limiting can mitigate these problems.

3.1.4 Network Time Protocol (NTP) Security

The Network Time Protocol (NTP) is used to synchronise devices (hosts, server, routers etc.) to within a few milliseconds of Coordinated Universal Time (UTC). NTP clients request times from NTP servers, taking into account round-trip times of this communication. In principle, NTP servers use a hierarchical security model implementing digital signatures and other standard application-layer security mechanisms to prevent transport-layer attacks such as replay or PITM. However, these security mechanisms are rarely enforced, easing attacks that shift the time of target system [26] in both, on-path and off-path attacks. In fact, such time shifting attacks may have severe consequences, as they, e.g., allow attackers to use outdated certificates or force cache flushes. The large number of NTP clients that rely on just a few NTP servers on the Internet to obtain their time are especially prone to this attack [27]. To counter this threat, network operators should install local NTP servers that use and compare multiple trusted NTP server peers. Alternatively, hosts can use NTP client implementations that offer provably secure time crowdsourcing algorithms [28].

3.1.5 Distributed Hash Table (DHT) Security

There are two main threats to DHTs: (i) *Eclipse* and (ii) *Sybil* attacks. An Eclipse attacker aims to poison routing tables to isolate target nodes from other, benign overlay peers. Redundancy helps best against Eclipse attacks. For example, systems like Kademlia foresee storage and routing redundancy, which mitigate some low-profile attacks against DHTs. In the extreme, DHT implementations can use dedicated routing tables with verified entries [29]. Central authorities—which lower the degree of distribution of DHTs, though—can solve the underlying root problem and assign stable node identifiers [30].

In a Sybil attack, an adversary introduces malicious nodes with self-chosen identifiers to subvert DHT protocol redundancy [31]. To prevent such Sybils, one can limit the number of nodes per entity, e.g., based on IP addresses [29]—yet causing collateral damage to nodes sharing this entity (e.g., multiple peers behind a NAT gateway). Others suggested to use peer location as identifier validation mechanisms, which, however, prevents that nodes can relocate [32]. Computational puzzles can slow down the pace at which attackers can inject malicious peers [33], but are ineffective against distributed botnet attacks. Finally, reputation systems enable peers to learn trust profiles of their neighbors [34], which ideally discredit malicious nodes [34].

Unfortunately, all these countermeasures either restrict the generality of DHTs, or introduce a centralized component. Therefore, most defenses have not fully evolved from academia into practice. A more complete treatment of DHT security is provided by Urdaneta and van Steen [30] and in the Distributed Systems Security CyBOK Knowledge Area [35].

3.1.6 Anonymous and Censorship-Free Communication

Anonymity in communication is a double-edged sword. On the one hand, we want to hold malicious communication parties accountable. On the other hand, other scenarios may indeed warrant for anonymous communication. For example, democratic minds may want to enable journalists to communicate even in suppressing regimes.

Anonymity can best be achieved when mixing communication with others and rerouting it over multiple parties. Onion routing represents the *de facto* standard of such Anonymous Communication Networks (ACNs). *Tor* is the most popular implementation of this general idea, in which parties communicate via (typically three) onion routers in a multi-layer encrypted overlay network [36]. A Tor client first selects (typically three) onion routers, the entry node, middle node(s), and exit node. The client then establishes an end-to-end secure channel (using TLS) to the entry node, which the client then uses to create another end-to-end protected between stream to the middle node. Finally, the client uses this client-to-middle channel to initiate an end-to-end protected stream to the exit node. The resulting path is called a circuit. Tor clients communicate over their circuit(s) to achieve sender anonymity. Any data passed via this circuit is encrypted three times, and each node is responsible for decrypting one layer.

Onion routing provides quite strong security guarantees. First of all, the entry and middle node cannot decrypt the communication passed over the circuit, only the exit node can. Furthermore, none of the proxies can infer *both* communication endpoints. In fact, only the entry node knows the client, and only the exit node knows the server. This also means that servers connected via onion routing networks do not learn the true addresses of their clients.

This general concept can be expanded to achieve recipient anonymity, i.e., protect the identity of servers. For example, Tor allows for so-called *onion services* which can only be contacted by Tor clients that know the server identity (a hash over their public key). As onion services receive data via Tor circuits and can never be contacted directly, their identity remains hidden.

While Tor gives strong anonymity guarantees, it is not fully immune against deanonymisation. In particular, traffic analysis and active traffic delay may help to infer the communication partners, especially if entry and exit node collaborate. In fact, it is widely accepted that powerful adversaries can link communication partners by correlating traffic entering and leaving the Tor network [37, 38]. Furthermore, patterns such as inter-arrival times or cumulative packet sizes were found sufficient to attribute encrypted communication to a particular website [39]. Consequently, attackers may be able to predict parts of the communication content even though communication is encrypted and padded. As a response, researchers explored countermeasures such as constant rate sending or more efficient variants of it [40].

Orthogonal to ACNs, censorship-resistant networks aim to prevent that attackers can suppress communication. The typical methodology here is to blend blocklisted communication into allowed traffic. For example, decoy routing uses on-path routers to extract covert (blocklisted) information from an overt (allowed) channel and redirects this hidden traffic to the true destination [41]. Similarly, domain fronting leverages allowed TLS endpoints to forward a covert stream—hidden in an allowed TLS stream—to the actual endpoint [42]. Having said this, nation state adversaries have the power to turn off major parts (or even all) of the communication to radically subvert these schemes at the expense of large collateral damage.

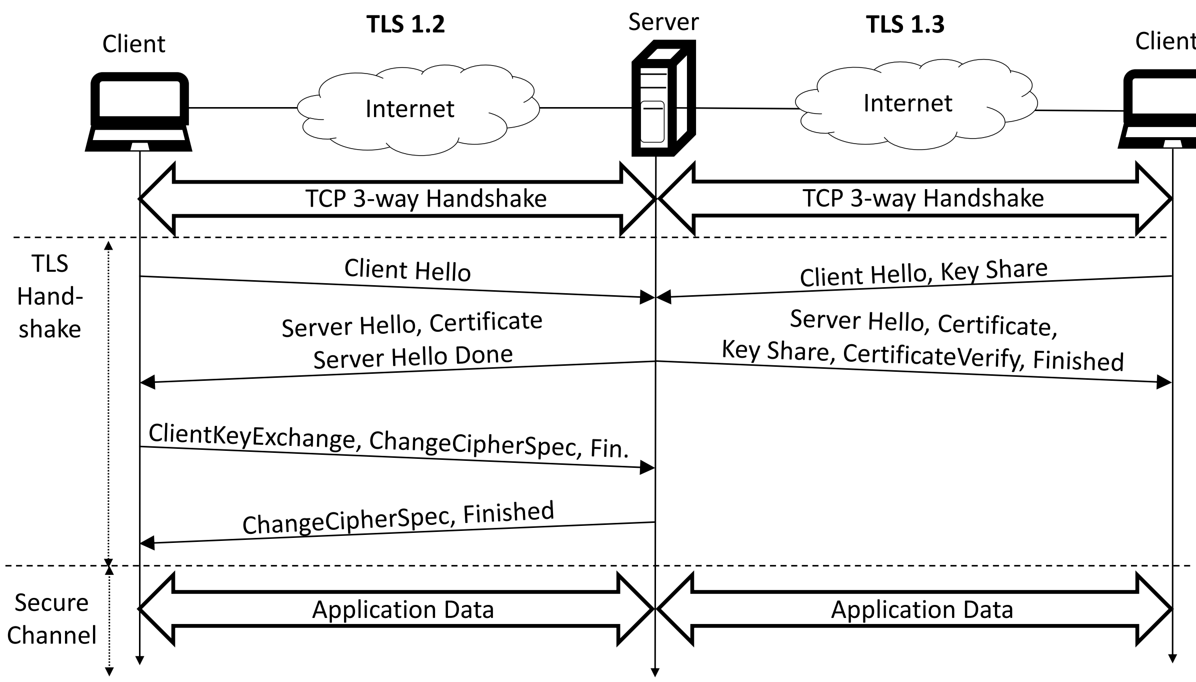

Figure 2: TLS Handshake: Comparison between TLS 1.2 (on the left) and TLS 1.3 (on the right), excluding the optional steps for client authentication.

3.2 Security at the Transport Layer

[2, c8] [4, c4,c6] [3, c8]

In this subsection, we discuss the security on the transport layer which sits below the application layer in the protocol stack.

3.2.1 TLS (Transport Layer Security)

Application-layer protocols rely on the transport layer to provide confidentiality, integrity and authentication mechanisms. These capabilities are provided by a shim layer *between* the application and transport layers, called the Transport Layer Security (TLS). In this section, our discussions will be hugely simplified to just cover the basics of the TLS protocol. For a more detailed discussion, including the history of TLS and past vulnerabilities, see Applied Cryptography CyBOK Knowledge Area [9].

We discuss the most recent and popular TLS versions 1.2 and 1.3, with a particular focus on their handshakes. Irrespective of the TLS version, the handshake takes care of cryptographic details that application-layer protocols otherwise would have to deal with themselves: authenticating each other, agreeing on cryptographic cipher suites, and deriving key material.

The handshakes differ between the two TLS versions, as shown in Figure 2. We start discussing TLS 1.2, as shown on the left-hand side of the figure. First, client and server negotiate which TLS version and cipher suites to use in order to guarantee compatibility even among heterogeneous communication partners. Second, server and client exchange certificates to authenticate each other, whereas client authentication is optional (and for brevity, omitted in Figure 2). Certificates contain communication partner identifiers such as domain names for web servers, and include their vetted public keys (see Section 3.2.2 for details). Third, the communication partners derive a symmetric key that can be used to secure the data transfer. To derive a key, the client can encrypt a freshly generated symmetric key under the server's public (e.g., RSA) key. Alternatively, the partners can derive a key using a Diffie-Hellman Key Exchange (DHKE). The DHKE provides TLS with perfect forward secrecy that prevents attackers from decrypting communication even if the server's private key leaks. As a final step, the handshake then validates the integrity of the handshake session. From now on, as part of the data transfer phase, TLS partners use the derived key material to encrypt and authenticate the subsequent communication.

TLS 1.3, as shown on the right of Figure 2, designs this handshake more efficiently. Without sacrificing security guarantees, TLS 1.3 reduces the number of round-trip times to one (1-RTT). TLS 1.3 no longer supports RSA-based key exchanges in favor of DHKE. The client therefore guesses the chosen key agreement protocol (e.g., DHKE) and sends its key share right away in the first step. The server would then respond with the chosen protocol, its key share, certificate and a signature over the handshake (in a *CertificateVerify* message). If the client was connected to the server before, TLS 1.3 even supports a handshake without additional round-trip time (0-RTT)—at the expense of weakening forward secrecy and replay prevention. Finally, as Formal Methods for Security CyBOK Knowledge Area [43] explores, TLS 1.3 has the additional benefit that it is formally verified to be secure [44, 45].

We now briefly discuss how TLS successfully secures against common network attacks. First, consider an eavesdropper that wants to obtain secrets from captured TLS-protected traffic. As the user data is encrypted, no secrets can be inferred. Second, in an IP spoofing attack, attackers may try any of the TLS partners to accept bogus data. However, to inject data, attackers lack the secret key to inject encrypted content. Third, also data cannot be altered, as TLS protects data integrity using authenticated encryption or message authentication codes. Finally, even a strong PITM attack is prevented by the help of certificates that authenticate the parties—unless the PITM attacker can issue certificates that the TLS partners trust, as discussed next. The TLS protocol also guarantees that payload arrives at the application in order, detects dropped and modified content, and also effectively prevents replay attacks that resend the same encrypted traffic to duplicate payload. Having said this, TLS does not prevent attackers from delaying parts or all of the communication.

3.2.2 Public Key Infrastructure

So far we have simply assumed that communication partners can reliably obtain trustworthy public keys from each other. However, in presence of active on-path attackers, how can one trust public keys exchanged via an insecure channel? The fundamental "problem" is that, conceptually, everyone can create public/private key pairs. Public-Key Infrastructure (PKI) provides a solution for managing *trustworthy* public keys (and, implicitly, their private key counterparts). Government agencies or standard organisations appoint registrars who issue and keep track of so-called *certificates* on behalf of entities (individuals, servers, routers etc).

Assume a user wants to obtain a trusted certificate and the corresponding key material. To this end, the user first generates a public/private key pair on their own hardware. The private key is never shared with anyone. The public key becomes part of a certificate signing request (CSR) that the user sends to a registration authority. Before this authority signs the certificate as requested, the user has to prove their identity (e.g., possession of a domain name for an HTTPS certificate, or personal identifiers for an S/MIME certificate) to registrars. The registrar's signature prevents forgery, as anyone can now verify the certificate using the (publicly known or similarly verifiable) registrar's public key. The resulting certificate contains the user's identity and public key, as well as CA information and a period of certificate validity.

Its format and PKI management specifications are specified in RFC 1422 and the ITU-X.509 standard.

The existing PKI model has faced several challenges, as evidenced by cases where CAs have issued certificates in error, or under coercion, or through their own infrastructure being attacked. As a response, CAs publish a list of revoked/withdrawn certificates, which can be queried using the Online Certificate Status Protocol (OCSP) as defined in RFC 6960, or is piggy-backed ("stapled") in TLS handshakes. To avoid wrong (but validated) certificates being issued, browsers temporarily started "pinning" them. However, this practice that was quickly abandoned and deprecated in major browsers, as it turned out to be prone to human errors (in case of key theft or key loss). Instead, big players such as Google or Cloudflare started collecting any observed and valid certificates in public immutable logs. TLS client such as browsers can then opt to refuse non-logged certificates. This scheme, known as Certificate Transparency (CT) [46], forces attackers publishing their rogue certificates. Consequently, certificate owners can notice whether malicious parties started abusing their identifies (e.g., domains).

The web of trust is an alternative, decentralized PKI scheme where users can create a community of trusted parties by mutually signing certificates without needing a registrar. The PGP scheme we discussed in Section 3.1.1 and its prominent implementation GNU Privacy Guard (GPG) is a good example, in which users certify each others' key authenticity.

A more detailed PKI discussion is part of the Applied Cryptography CyBOK Knowledge Area [9].

3.2.3 TCP Security

TLS does a great deal in protecting the TCP payloads and prevents session hijacks and packet injection. Yet what about the security of TCP headers of TLS connections or other, non-TLS connections? In fact, attackers could try launching TCP reset attacks that aim to maliciously tear down a target TCP connection. To this end, they guess or bruteforce valid sequence numbers, and then spoof TCP segments with the RST flag being set. If the spoofed sequence numbers hit the sliding window, the receiving party will terminate the connection. There are mainly two orthogonal solutions to this problem deployed in practice: (i) TCP/IP stacks have to ensure strong randomness for (initial) sequence number generation. (ii) Deny RST segments with sequence numbers that fall in the middle of the sliding window. Conceptually, these defenses are ineffective against on-path attackers that can reliably manipulate TCP segments (e.g., dropping payload and setting the RST flag). Having said this, Weaver et al. [47] show that race conditions allow for detecting RST attacks launched by off-path attackers even if they can infer the correct sequence number.

A *SYN Flooding attacker* keeps sending TCP SYN segments and forces a server to allocate resources for half-opened TCP connections. When servers limit the number of half-opened connections, benign clients can no longer establish TCP connections to the server. To mitigate this session exhaustion, servers can delete a random half-opened session whenever a new session needs to be created—potentially deleting benign sessions, though. A defence known as SYN Cookies has been implemented by operating systems as a more systematic response to SYN floods [RFC4987]. When enabled, the server does not half open a connection right away on receiving a TCP connection request. It selects an Initial Sequence Number (ISN) using a hash function over source and destination IP addresses, port numbers of the SYN segment, a timestamp with a resolution of 64 seconds, as well as a secret number only known to the server. The server then sends the client this ISN in the SYN/ACK message. If the request is from a legitimate sender, the server receives an ACK message with an acknowledgment number which is ISN plus 1. To verify if an ACK is from a benign sender, the server thus again computes the SYN cookie using the above-mentioned data, and checks if the acknowledge number in the ACK segment minus one corresponds to the SYN cookie. If so, the server opens a TCP connection, and only then starts using resources. A DoS attacker would have to waste resources themselves and reveal the true sending IP address to learn the correct ISN, hence, leveling the fairness of resource consumption.

3.2.4 UDP Security

What TLS is for TCP, Datagram TLS (DTLS) is for UDP. Yet again there are additional security considerations for UDP that we briefly discuss next. In contrast to its big brother TCP, UDP is designed such that application-layer protocols have to handle key mechanisms themselves (or tolerate their absence), including reordering, reliable transport, or identifier recognition.

Furthermore, being a connection-less protocol, UDP endpoints do not implicitly verify each others' IP address before communication starts. Consequently, if not handled at the application layer, UDP protocols are prone to IP spoofing attacks. We already showcased the consequences of this at the example of DNS spoofing. In general, to protect against this threat, any UDP-based application protocol must gauge the security impact of IP spoofing.

Reflective DDoS attacks are a particular subclass of IP spoofing attacks. Here, attackers send IP packets in which the source IP address corresponds to a DDoS target. If the immediate recipients (called *reflectors*) reply to such packets, their answers overload the victim with undesired replies. We mentioned this threat already in the context of DNS (Section 3.1.3). The general vulnerability boils down to the lack of IP address validation in UDP. Consequently, several other UDP-based protocols are similarly vulnerable to reflection [24]. Reflection attacks turn into amplification attacks, if the responses are significantly larger than the requests, which effectively amplifies the attack bandwidth. Unless application-level protocols validate addresses, or enforce authentication, reflection for UDP-based protocols will remain possible. If protocol changes would break compatibility, implementations are advised to rate limit the frequency in which clients can trigger high-amplification responses. Alternative, non-mandatory instances of amplifying services can be taken offline [25].

3.2.5 QUIC

QUIC is a new transport-level protocol that saw rapid deployment by popular Web browsers. QUIC offers faster communication using UDP instead of HTTP over TCP. QUIC was originally designed by Google, and was then standardized by the IETF in 2021 [48]. Its main goal is increasing communication performance using multiplexed connections. Being a relatively new protocol, in contrast to other protocols, QUIC was designed to be secure. Technically, QUIC uses most of the concepts described in TLS 1.3, but replaces the TLS Record Layer with its own format. This way, QUIC cannot only encrypt payload, but also most of the header data. QUIC, being UDP-based, "replaces" the TCP three-way handshake by its own handshake, which integrates the TLS handshake. This eliminates any round-trip time overhead of TLS. With reference to Figure 2 (page 14), QUIC integrates the only two TLS 1.3 handshake messages in its own handshake. When serving certificates and additional data during the handshake, QUIC servers run the risk of being abused for amplification attacks (cf. Section 3.2.4), as server responses are significantly larger than initial client requests. To mitigate this problem, QUIC servers verify addresses during the handshake, and must not exceed certain amplification prior to verifying addresses (the current IETF standard draft defines a factor of three).

3.3 Security at the Internet Layer

[2, c8] [4, c5,c9] [5, c17] [3, c8]

Although application-layer and transport-layer security help to provide end-to-end security, there is also merit in adding security mechanisms to the network layer. First, higher-layer security mechanisms do not necessarily protect an organisation's internal network links from malicious traffic. If and when malicious traffic is detected at the end hosts, it is too late, as the bandwidth has already been consumed. The second major issue is that the higher-layer security mechanisms described earlier (e.g., TLS) do not conceal or protect IP headers. This makes the IP addresses of the communicating end hosts visible to eavesdroppers and even modifiable to PITM attackers.

3.3.1 IPv4 Security

IP Spoofing: IP spoofing, as discussed for UDP and DNS (sections 3.2.4 and 3.1.3, respectively), finds its root in the Internet Protocol (IP) and affects both IPv4 and IPv6. In principle, malicious clients can freely choose to send traffic with any arbitrary IP address. Thankfully, most providers perform egress filtering and discard traffic from IP addresses outside of their domain [49]. Furthermore, Unicast Reverse Path Forwarding (uRPF) enables on-path routers to drop traffic from IP addresses that they would have expected entering on other interfaces [49].

Fragmentation Attacks: IPv4 has to fragment packets that do not fit the network's Maximum Transmission Unit (MTU). While fragmentation is trivial, defragmentation is not so, and has led to severe security problems in the past. For example, a Teardrop attack abuses the fact that operating systems may try to retain huge amounts of payload when trying to reassemble highly-overlapping fragments of a synthetic TCP segment. Fragmentation also eases DNS cache poisoning attacks in that attackers need to bruteforce a reduced search space by attacking only the non-starting fragments [50]. Finally, fragmentation may assist attackers in evading simple payload matches by scattering payload over multiple fragments.

VPNs and IPsec: Many organisations prefer their traffic to be fully encrypted as it leaves their network. For example, they may want to connect several islands of private networks owned by an organisation via the Internet. Also, employers and employees want a flexible work environment where people can work from home, or connect from a hotel room or an airport lounge without compromising their security. If only individual, otherwise-internal web hosts need to made available, administrators can deploy web proxies that tunnel traffic (sometimes referred to as *WebVPN*). In contrast, a full-fledged Virtual Private Network (VPN) connects two or more otherwise-separated networks, and not just individual hosts.

There are plenty of security protocols that enable for VPNs, such as Point-to-Point Tunneling Protocol (PPTP) (deprecated), TLS (used by, e.g., OpenVPN [51]), or Secure Socket Tunneling Protocol (SSTP). We will illustrate the general VPN concept at the example of the Internet Protocol Security (IPsec) protocol suite. Figure 4 shows that an employee working from home accesses a server at work, the VPN client in their host encapsulates IPv4 datagrams into IPsec and encrypts IPv4 payload containing TCP or UDP segments, or other control messages. The corporate gateway detects the IPsec datagram, decrypts it and decapsulates it back to the IPv4 datagram before forwarding it to the server. Every response from the server is

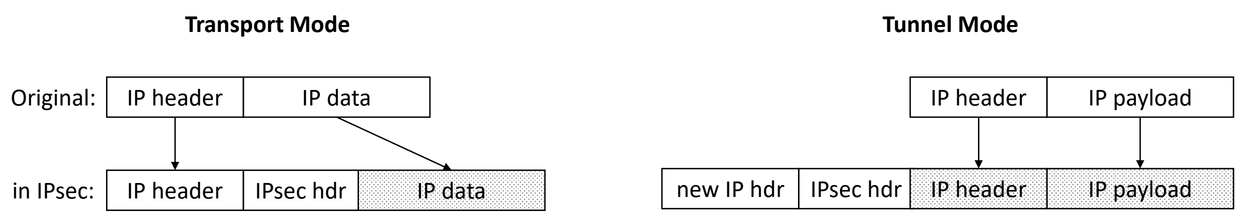

Figure 3: Comparison between IPsec transport mode and tunnel mode. All parts of the original packets that are shaded in gray are protected in the respective mode.

also encrypted by the gateway. IPsec also provides data integrity, origin authentication and replay attack prevention. These guarantees depend on the chosen IPSec protocol, though. Only the recommended and widely-deployed Encapsulation Security Payload (ESP) protocol (part of IPSec) provides these guarantees, including confidentiality and origin authentication. In contrast, the less popular Authentication Header (AH) protocol just provides integrity. Similarly, several tunneling protocols such as Generic Routing Encapsulation (GRE), Layer 2 Tunneling Protocol (L2TP) or Multiprotocol Label Switching (MPLS) do not provide CIA guarantees. Should those be required in untrusted networks, e.g., due to GRE's multi-protocol or multi-casting functionality, it is advisable to use them in combination with IPSec.

The entire set of modes/configurations/standards provided by IPsec is extensive [52]. Here, we only briefly introduce that IPsec supports two modes of operation: *tunnel mode* and *transport mode*, as compared in Figure 3. In transport mode, only the IP payload—not the original IP header—is protected. The tunnel mode represent a viable alternative if the edge devices (routers/gateways) of two networks are IPsec aware. Then, the rest of the servers/hosts need not worry about IPsec. The edge devices encapsulate every IP packet including the header. This virtually creates a secure tunnel between the two edge devices. The receiving edge device then decapsulates the IPv4 datagram and forwards within its network using standard IP forwarding. Tunnel mode simplifies key negotiation, as two edge devices can handle connections on behalf of all hosts in their respective networks. An additional advantage is that also IP headers (including source/destination address) gets encrypted.

When a large number of endpoints use IPsec, manually distributing the IPsec keys becomes challenging. RFC 7296 [53] defines the Internet Key Exchange protocol (IKEv2). Readers will observe a similarity between TLS (Section 3.2) and IKE, in that IKE also requires an initial handshake process to negotiate cryptographic algorithms and other values such as nonces and exhange identities and certificates. We will skip the details of a complex two-phase protocol exchange which results in the establishment of a quantity called SKEYSEED. These SKEYSEEDs are used to generate the keys used during a session (Security Associations (SAs)). IKEv2 uses the Internet Security Association and Key Management Protocol (ISAKMP) [54], which defines the procedures for authenticating the communicating peer, creation and management of SAs, and the key generation techniques.

NAT: Due to the shortage of IPv4 address space, Network Address Translation (NAT) was designed so that private IP addresses could be mapped onto an externally routable IP address by the NAT device [2]. For an outgoing IP packet, the NAT device changes the private source IP address to a public IP address of the outgoing link. This has implicit, yet unintentional security benefits. First, NAT obfuscates the internal IP address from the outside world. To a potential attacker, the packets appear to be coming from the NAT device, not the real host behind the NAT device. Second, unless loopholes are opened via port forwarding or via UPnP,

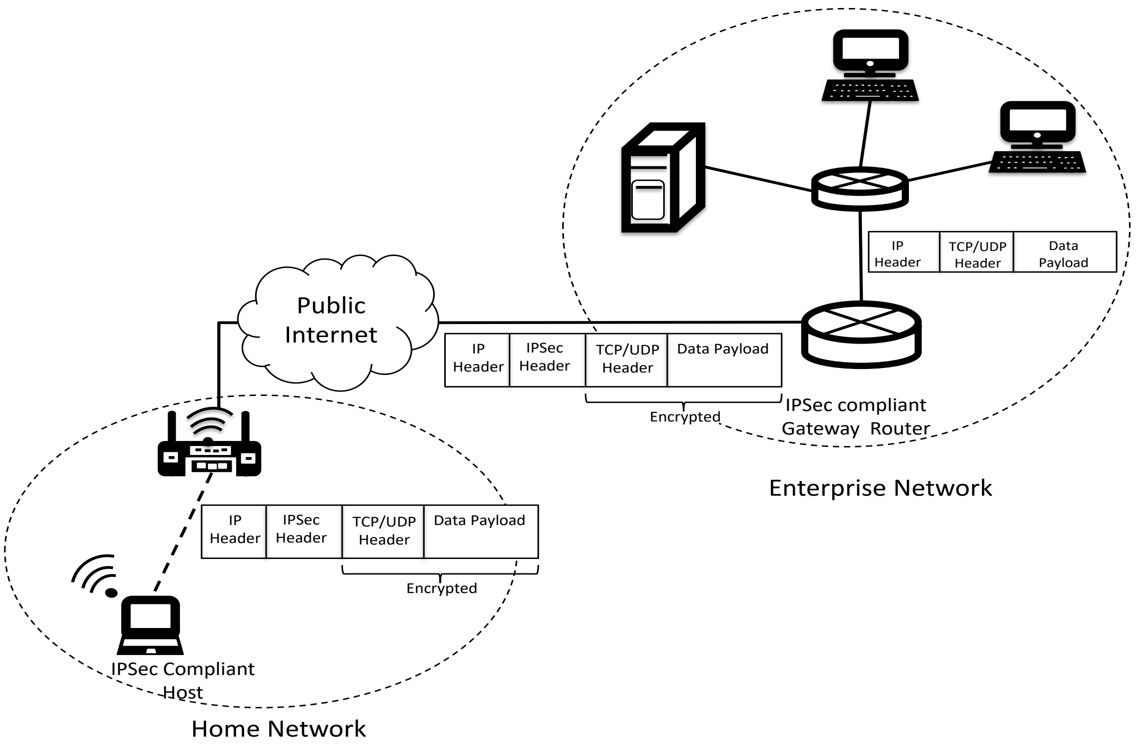

Figure 4: IPsec client-server interaction in transport mode (no protection of IP headers).

NAT gateways such as home routers prevent attackers from reaching internal hosts.

3.3.2 IPv6 Security

Although conceptually very similar from a security perspective, IPv6 brings a few advantages over IPv4. For example, IPv6's 128-bit address space slows down port scans, as opposed to IPv4, where the entire 32-bit address space can be scanned in less than an hour [55]. Similarly, IPv6 comes with built-in encryption in form of IPsec. IPsec that was initially mandated in the early IPv6 standard. Yet nowadays, due to implementation difficulties, IPsec remains a recommendation only. Furthermore, in contrast to IPv4, IPv6 has no options in its header—these were used for attacks/exploits in IPv4.

The community has debated many years over the potential security pitfalls with IPv6. As a quite drastic change, the huge address space in IPv6 obsoletes NATing within the IPv6 world, including all its implicit security benefits. In particular, NAT requires state tracking, which devices often couple with a stateful firewall (which we will discuss in Section 4.1) that brings additional security. Furthermore, NAT hides the true IP addresses and therefore complicates IP-based tracking—providing some weak form of anonymity. Having said this, experts argue that these perceived advantages also come with lots of complexity and disadvantages (e.g., single point of failure), and that eliminating NAT by no means implies that Internet-connected devices no longer have firewalls [56]. Furthermore, having large networks to choose addresses from, IPv6 may allow to rotate IP addresses more frequently to complicate address-based tracking. Summarizing this debate, as long as we do not drop firewalls, and are careful with IP address assignment policies, IPv6 does not weaken security.

Finally, another important aspect to consider is that we are still in a steady transition from IPv4 to IPv6. Hence, many devices feature a so-called *dual stack*, i.e., IPv4 and IPv6 connectivity.

This naturally asks for protecting both network accesses simultaneously.

3.3.3 Routing Security

IPv4/IPv6 assume that Internet routers reliably forward packets from source to destination. Unfortunately, a network can easily be disrupted if either the routers themselves are compromised or they accept spurious routing exchange messages from malicious actors. We will discuss these threats in the following, distinguishing between internal and external routing.

Within an Autonomous System (AS): Interior Gateway Protocols (IGPs) are used for exchanging routing information within an Autonomous System (AS). Two such protocols, Routing Information Protocol (RIPv2) and Open Shortest Path First (OSPFv2), are in widespread use with ASs for IPv4 networks. The newer RIPng and OSPFv3 versions support IPv6. These protocols support no security by default but can be configured to support either plain text-based authentication or MD5-based authentication. Authentication can avoid several kinds of attacks such as bogus route insertion or modifying and adding a rogue neighbour. Older routing protocols, including RIPv1 or Cisco's proprietary Interior Gateway Routing Protocol (IGRP)—unlike its more secure successor, the Enhanced Interior Gateway Routing Protocol (EIGRP)—do not offer any kind of authentication, and hence, should be used with care.

Across ASs: The Internet uses a hierarchical system where each AS exchanges routing information with other ASs using the Border Gateway Protocol (BGP) [57, 58]. BGP is a path vector routing protocol. We distinguish between External BGP used across ASs, and Internal BGP that is used to propagate routes *within* an AS. From now on, when referring to BGP, we talk about External BGP, as it comes with the most interesting security challenges. In BGP, ASs advertise their IP prefixes (IP address ranges of size /24 or larger) to peers, upstreams and customers [2]. BGP routers append their AS information before forwarding these prefixes to their neighbors. Effectively, this creates a list of ASs that have to be passed to reach the prefix, commonly referred to as the *AS path*.

High-impact attacks in the past have highlighted the security weakness in BGP due to its lack of integrity and authentication [59]. In particular, in a *BGP prefix hijacking attack* [60], a malicious router could advertise an IP prefix, saying that the best route to a service is through its network. Once the traffic starts to flow through its network, it can drop (DoS, censorship), sniff on (eavesdrop) or redirect traffic in order to overload an unsuspecting AS. As a countermeasure, the Resource Public Key Infrastructure (RPKI) [61], as operated by the five Regional Internet Registrys (RIRs), maps IP prefixes to ASs in so-called Route Origin Authorization (ROA). When neighbors receive announcements, RPKI allows them to discard BGP announcements that are not backed by an ROA or are more specific than allowed by the ROA. This process, called Route Origin Validation (ROV), enables to drop advertisements in which the AS that owns the advertised prefix is not on the advertised path.

RPKI *cannot* detect bogus advertisements where the owning AS is on path, but a malicious AS aims to reroute the target's AS traffic as an intermediary. BGPsec partially addresses this remaining security concern [62]. Two neighbouring routers can use IPsec mechanisms for point-to-point security to exchange updates. Furthermore, BGPsec enables routers to verify the incremental updates of an announced AS path. That is, they can verify which on-path AS has added itself to the AS path, preventing bogus paths that include a malicious AS that lacks the according cryptographic secrets. However, BGPsec entails large overheads, such as verifying a larger number of signatures on booting, and splitting up bulk announcements into many smaller ones. Furthermore, BGPsec only adds security if all systems on the AS path support it. Hence, not many routers deploy BGPsec yet, fueled by the lack of short-term benefits [63]—and it is likely to take years until it will find wide adoption, if ever.

Despite the fact that BGP prefix hijacks are a decade-old problem, fixing them retroactively remains one of the great unsolved challenges in network security. In fact, one camp argues that the BGP design is inherently flawed [64], and entire (yet not widely deployed) Internet redesigns such as SCION [65] indeed provide much stronger guarantees. Others did not give up yet, and hope to further strengthen the trust in AS paths by the help of ongoing initiatives such as Autonomous System Provider Authorization (ASPA) [66].

3.3.4 ICMP Security

Internet Control Message Protocol (ICMP) is a supportive protocol mainly used for exchanging status or error information. Unfortunately, it introduced several orthogonal security risks, most of which are no longer present but still worth mentioning. Most notably, there many documented cases in which ICMP was an enabler for Denial of Service (DoS) attacks. The *Ping of Death* abused a malformed ICMP packet that triggered a software bug in earlier versions of the Windows operating system, typically leading to a system crash at the packet recipient. In an ICMP flood, an attacker sends massive amounts of ICMP packets to swamp a target network/system. Such floods can be further amplified in so-called *smurf attacks*, in which an attacker sends IP-spoofed ICMP ping messages to the broadcast address of an IP network. If the ICMP messages are relayed to all network participants using the (spoofed) address of the target system as source, the target receives ping responses from all active devices. Smurf attacks can be mitigated by dropping ICMP packets from outside of the network, or by dropping ICMP messages destined to broadcast addresses.

But also outside of the DoS context, ICMP is worth considering from a security perspective. Insider attackers can abuse ICMP as covert channel to leak sensitive data unless ICMP is closely monitored or forbidden. ICMP reachability tests allow attackers to perform network reconnaissance during network scans (see also Section 4.3). Many network operators thus balance pros and cons of ICMP in their networks, often deciding to drop external ICMP messages using a firewall (see also Section 4.1).

3.4 Security on the Link Layer

[2, c8] [4, c7] [3, c8]

In this section, we are confining our attention to the security of the link layer. We mostly focus on the logical part of the link layer. The physical part is addressed in the Physical Layer and Telecommunications Security CyBOK Knowledge Area [67].

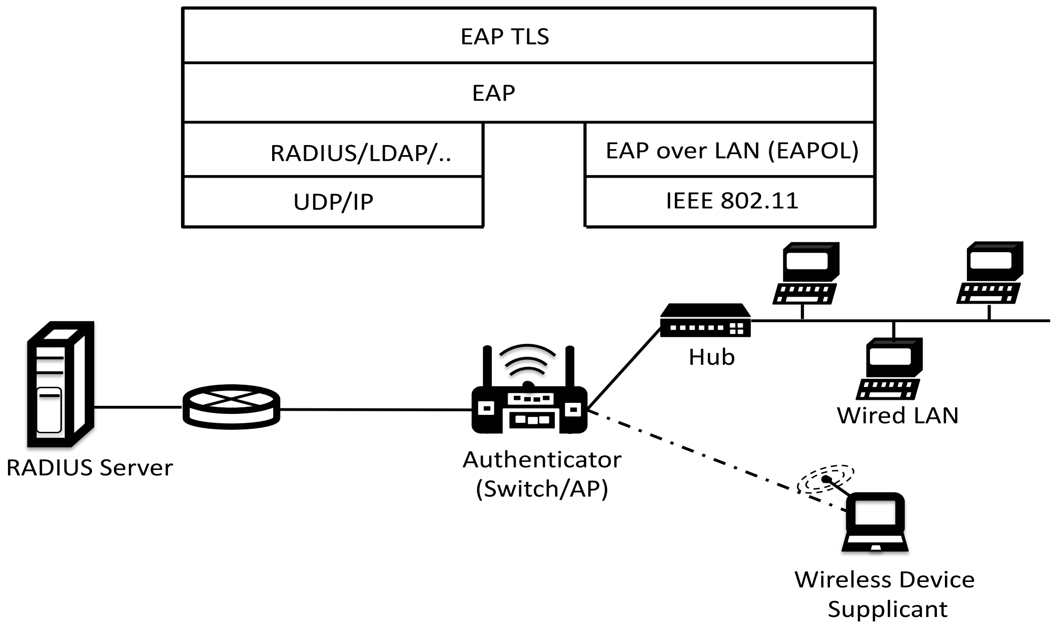

Figure 5: Extensible Authentication Protocol (EAP)

3.4.1 Port-based Network Access Control (IEEE 802.1X)

The IEEE 802.1X is a port-based authentication for securing both wired and wireless networks. Before a user can access a network at the link layer, it must authenticate the switch or access point (AP) they are attempting to connect to, either physically or via a wireless channel. As with most standards bodies, this group has its own jargon. Figure 5 shows a typical 802.1X setup. A user is called a supplicant and a switch or AP is called an authenticator. Supplicant software is typically available on various OS platforms or it can also be provided by chipset vendors. A supplicant (client) wishing to access a network must use the Extensible Authentication Protocol (EAP) to connect to the Authentication Server (AuthS) via an authenticator. The EAP is an end-to-end (client to authentication server) protocol. When a new client (supplicant) is connected to an authenticator, the port on the authenticator is set to the 'unauthorised' state, allowing only 802.1X traffic. Other higher layer traffic, such as TCP/UDP is blocked. The authenticator sends out the EAP-Request identity to the supplicant. The supplicant responds with the EAP-response packet, which is forwarded to the AS, and typically proves the supplicant possesses its credentials. After successful verification, the authenticator unblocks the port to let higher layer traffic through. When the supplicant logs off, the EAP-logoff to the authenticator sets the port to block all non-EAP traffic.

There are a couple of pitfalls when deploying EAP and choosing the wrong mode of operation. Certain EAP modes are prone to PITM attacks, especially in a wireless setting. It therefore is advised to use any sort of TLS-based EAP variant, such as EAP-TLS [68] or the Protected Extensible Authentication Protocol (PEAP). Similarly, dictionary attacks can weaken the security guarantees of certain EAP modes (e.g., EAP-MD5) that should be avoided.

3.4.2 WAN Link-Layer Security

What IEEE 802.1X is for local networks, protocols like Point-to-Point Protocol (PPP), its sibling PPP over Ethernet (PPPoE), or High-Level Data Link Control (HDLC) are for Wide Area Networks (WANs). They offer clients means to connect to the Internet by the help of their ISPs. PPP(oE) is the most widely used protocol in this context, used by billions of broadcast devices worldwide. Although optional in its standard, in practice, ISPs usually mandate client authentication to hold unauthorized users off. Popular examples of such authentication protocols within PPP are Password Authentication Protocol (PAP), Challenge Handshake Authentication Protocol (CHAP), or any of the authentication protocols supported by EAP. Usage of PAP is discouraged, as it transmits client credentials in plain. Instead, CHAP uses a reasonbly secure challenge-response authentication, which is however susceptible to offline bruteforce attacks against recorded authentication sessions that contain weak credentials.

3.4.3 Attacks On Ethernet Switches

Ethernet switches maintain forwarding table entries in a Content Addressable Memory (CAM). As a switch learns about a new destination host, the switch includes this host's address and its physical port in the CAM. For all future communications, this table entry is looked up to forward a frame to the correct physical port. MAC spoofing allow attackers to manipulate this mapping by forging their Message Authentication Code (MAC) addresses when sending traffic. For example, to poison the forwarding table, an attacker crafts frames with random addresses to populate an entire CAM. If successful, switches have to flood all the incoming data frames to all the outgoing ports, as they can no longer enter new address-to-port mappings. This makes the data available to the attacker attached to any of the switch ports.

Such MAC spoofing attacks can also be more targeted. Assume attackers want to steal traffic destined to one particular target host only, instead of seeing all traffic. The attacker then copies the target's MAC address. This way, the attacker may implicitly rewrite the target's entry in the switch forwarding table. If so, the switch will falsely forward frames to the attacking host that were actually destined for the target host.

Mitigating these MAC spoofing attacks requires authenticating the MAC addresses before populating the forwarding table entry. For example, IEEE 802.1X (see Section 3.4.1) mitigates such attack, vetting hosts before they can connect. Furthermore, switches may limit the number of MAC addresses per interface or enforce MAC bindings, as described next.

3.4.4 Address Resolution Protocol (ARP) / Neighbor Discovery Protocol (NDP)

The Address Resolution Protocol (ARP) translates IPv4 addresses to link layer addresses (e.g., MAC addresses in Ethernet). ARP spoofing is similar to MAC spoofing, yet does not (only) target the switch's address mappings. Instead, ARP spoofing target the IP-to-MAC address mappings of all network participants (possibly including the switch) in the same segment. To this end, ARP spoofing attackers send fake ARP messages over a LAN. For example, they can broadcast crafted ARP requests and hope participants learn wrong IP-to-MAC mappings on the fly, or reply with forged replies to ARP request. Either way, attackers aim to (re-)bind the target's IP address to their own MAC address. If successful, attackers will receive data that were intended for the target's IP address. ARP spoofing is particularly popular for session hijacking and PITM attacks. Similar attacks are possible for the Reverse Address Resolution Protocol (RARP), which—by now rarely used—allows hosts to discover their IP address. To mitigate ARP spoofing, switches employ (or learn) a trusted database of static IP-to-MAC address mappings, and refuse to relay any ARP traffic that contradicts these trusted entries. Alternatively, network administrators can spot ARP anomalies [69], e.g., searching for suspicious cases in which one IP address maps to multiple MAC addresses.

What ARP is for IPv4, the Neighbor Discovery Protocol (NDP) is for IPv6. NDP is based on ICMPv6 and is more feature-rich than ARP. Conceptually, NDP underlies the same spoofing risks as ARP, though, and requires the same countermeasures. Furthermore, there is one more caveat due to automatic IPv6 address assignment. In IPv6's most basic (yet common) IP address autoconfiguration scheme, layer-3 addresses are derived directly from layer-2 addresses without any need for address resolution. Knowledge of the MAC address may allow attackers to infer information about the host/servers which can be handy when launching attacks, or to track devices even if they change network prefixes. Using hash function for address generation is recommended as a mitigation technique. Further, RFC 4982 extends IPv6 by allowing for a Cryptographically Generated Address (CGA) where an address is bound to a public signature key. Orthogonal to this, RFC 7217 proposes to have stable addresses within a network prefix, and change them when clients switch networks to avoid cross-network tracking.

3.4.5 Network Segmentation

MAC spoofing and ARP spoofing nicely illustrate how fragile security on the network layer is. Consequently, network architects aim to split their physical network into several smaller networks—a practice known as *network segmentation*. Highly-critical environments such as sensitive military or control networks use *physical* segmentation. As the required change of cables and wires is quite expensive, *virtual* network segmentation has become a popular alternative. Virtual LANs (VLANs) are the *de facto* standard for virtual segmentation on Ethernet networks. VLANs can split sensitive (e.g., internal servers) from less sensitive (guest WiFi) network segments. VLANs enforce that routers can see and react upon traffic between segments, and limit the harm attackers can do to the entire LAN. It is important to note that network segmentation (e.g., via VLANs) does not necessarily require VPNs to bridge the networks. If all network segments are local, a router that is part of multiple subnetworks can connect them, ideally augmented with secure firewall policies (cf. Section 4.1) that control inter-network communication at the IP layer.

VLAN hopping attacks allow an attacking host on a VLAN to gain access to resources on other VLANs that would normally be restricted. There are two primary methods of VLAN hopping: switch spoofing and double tagging. In a switch spoofing attack, an attacking host impersonates a trunking switch responding to the tagging and trunking protocols (e.g., IEEE 802.1Q or Dynamic Trunking Protocol) typically used in a VLAN environment. The attacker now succeeds in accessing traffic for multiple VLANs. Vendors mitigate these attacks by proper switch configuration. For example, the ports are assigned a trunking role explicitly and the others are configured as access ports only. Also, any automatic trunk negotiation protocol can be disabled. In a double tagging attack, an attacker succeeds in sending its frame to more than one VLAN by inserting two VLAN tags to a frame it transmits. However, this attack does not allow them to receive a response. Again, vendors provide recommended configuration methods to deal with these possible attacks. A comprehensive survey of Ethernet attacks and defence can be found in [70].

Organizations like hosting providers that heavily virtualize services quickly reach the limitation of having a maximum of little less than 4096 VLANs when trying to isolating their services.

Virtual eXtensible LAN (VXLAN) tackles this limitation by introducing an encapsulation scheme for multi-tenant environments [71]. Unlike VLANs, which work on the link layer, VXLANs strictly speaking operate at the network layer to emulate link-layer networks. VXLAN allows creating up to ≈16M virtually separated networks, which can additionally be combined with VLAN functionality. Like VLANs, VXLANs also do not aim to provide confidentiality or integrity in general. Instead, they are means to segment networks. Worse, however, being on the network layer, VXLAN packets can traverse the Internet, and may allow attackers to inject spoofed VXLAN packets into "remote" networks. Thus, care has to be taken, e.g., by ingress filtering at the network edge to drop VXLAN packets that carry a valid VXLANs endpoint IP address.

3.4.6 Wireless Security

Wireless LAN are more vulnerable to security risks due to the broadcast nature of media, which simplifies eavesdropping. There have been several failed attempts to add integrity and confidentiality to WLANs communication. First, the Wired Equivalent Privacy (WEP) protocol used a symmetric key encryption method where the host shares a key with an Access Point (AP) out of band. WEP had several design flaws. First, a 24-bit IV introduced a weakness in that ≈16 million unique IVs can be exhausted in high-speed links in less than 2 hours. Given that IVs are sent in plaintext, an eavesdropper can easily detect this reuse and mount a known plaintext attack. Furthermore, using RC4 allowed for the Fluhrer, Martin and Shamir (FMS) attacks, in which an attacker can recover the key in an RC4 encrypted stream by capturing a large number of messages in that stream [72, 73]. Furthermore, WEP's linear CRC was great for detecting random link errors, but failed to reliably reveal malicious message modifications.

An interim standard called the Wi-Fi Protected Access (WPA) was quickly developed for backward hardware compatibility, while WPA2 was being worked out. WPA uses the Temporal Key Integrity Protocol (TKIP) but maintains RC4 for compatibility. The Pre-Shared Key (PSK), also known as WPA-Personal, is similar to the WEP-Key. However, the PSK is used differently, a nonce, and PSK are hashed to generate a temporal key. Following this, a cryptographic mixing function is used to combine this temporal key, the Temporal MAC (TMAC), and the sequence counter resulting in one key for encryption (128 bits) and another key for integrity (64 bits). As a consequence, every packet is encrypted with a unique encryption key to avoid FMS-style attacks. Also, the WPA extends the WEP IV to 48 bits. Several new fields include a new Frame Check Sequence (FCS) field, a CRC-32 checksum for error correction and a hash function for a proper integrity check. Due to compromises it made with respect to backwards compatibility, the WPA has had its own share of attacks, though [74].

The Wifi alliance then standardized WPA2 in 2004. WPA2 relies on more powerful hardware supporting a 128-bit AES Counter Mode with the Cipher Block Chaining Message Authentication Code Protocol (CCMP), obsoleting RC4. It also provides an improved 4-way handshake and temporary key generation method (which does not feature forward secrecy, though). While implementations of this handshake were shown insecure [75], the general handshake methodology was formally verified and is still believed to be secure [76, 77].

In 2018, a new WPA3 standard was accepted to make a gradual transition and eventually replace the WPA2. WPA3 overcomes the lack of perfect forward secrecy in WPA and WPA2. The PSK is replaced with a new key distribution called the Simultaneous Authentication of Equals (SAE) based on the IETF Dragonfly key exchange. The WPA3-Personal mode uses a 128-bit encryption, whereas the WPA3-Enterprise uses 192-bit encryption.

The discussion so far assumed that there is a shared secret between WLAN users and APs from which session keys can be derived. In fact, enterprise settings usually handle WLAN access control using perform strong authentication such as 802.1X (Section 3.4.1). Ideally, WLAN users have their own client certificates that provide much stronger security than any reasonably user-friendly password. In contrast, for openly accessible networks such as at airports or restaurants, there may neither be PSKs nor certificates. Consequently, the lack of strong encryption would leave communication unprotected. Opportunistic Wireless Encryption (OWE) tackles this open problem [78]. Instead of using a PSK during the WPA2/3 four-way handshake, the client and AP use a pairwise secret derived from an initial DHKE.

3.4.7 Bus Security

Bus networks follow a special topology in that all nodes are directly connected to a shared medium (the bus). Securing a bus is inherently complex, especially if we assume that an insider attacker is connected to the bus. In order to illustrate this, we will focus on the Controller Area Network (CAN) standard, which despite its age is still quite commonly used in cars today. CAN nicely reveals many issues that can arise on bus networks in general. CAN connects so-called Electronic Control Units (ECUs), such as a car's wheel, break pedal or the radio. CAN is a real-time protocol designed to give priority to more urgent ECUs (e.g., brake pedal) over less pressing ones (e.g., multimedia control). Sadly, CAN suffers from severe security vulnerabilities. They become especially problematic if ECUs are or turn malicious (e.g., after compromise). First, CAN does not authenticate messages, i.e., any compromised ECU (e.g., multimedia system) can easily spoof messages of critical components (e.g., wheel speed sensor). Second, compromised bus components can receive and invalidate all messages of any arbitrary other ECUs on the same bus. For example, a compromised ECU could suppress the signals sent by an activated brake pedal. Finally, and a little less concerning than the previous examples, CAN is unencrypted, providing no confidentiality against sniffing.

A radical protocol change could solve all these problems. In fact, there are new standards like AUTomotive Open System ARchitecture (AUTOSAR) [79] that provide improved security principles. Yet, as always, such radical changes take a long time in practice, as they break compatibility of existing devices. Also, devices have a years-long development cycle and usage time. Vendors are aware of these issues and aim to mitigate the problem by segmenting critical components from less critical ones (segmentation in general is discussed in Section 3.4.5). While certainly a vital step, as it physically disconnects more complex and vulnerable devices such as multimedia systems from safety-critical devices, this only reduces and not entirely eliminates the attack surface. A star topology would solve many of these issues, as the medium is no longer shared and address spoofing could be validated by a central entity. Yet star topologies incur significant additional physical cables, and thus, higher costs, manufacturing complexity, and weight. Academia explored several approaches to add message authenticity to CAN and to prevent spoofing on CAN without breaking backwards-compatibility [80, 81]. None of them found wide deployment in practice yet, though, possibly due to costs and the need to adapt ECUs. Alternative approaches aim to detect spoofed messages by learning and modeling the per-ECU voltage of bus messages. Unfortunately, such classifiers were proven unreliable [82]. A wider popularity of CAN-FD [13], which offers a flexible data rate and larger messages (64B instead of 8B in CAN) will decrease overhead of security add-ons and may thus ease the development of more secure CAN communication in the future.

Many of the observed problems generalize beyond CAN to any bus system or even shared-medium network. Rogue components on a bus can suppress messages by invalidating them, anyone on a bus can see all messages, and there are no built-in protection against spoofing.

Physical separation and segmentation of bus networks remains one of the key concepts to securing them. In addition, to add security guarantees to insecure bus protocols, we sometimes see complete protocol overhauls that typically break backward compatibility. For example, the insecure Modbus standard from 1979 [11] has a secure alternative (Modbus/TCP Security Protocol [83]) since 2018, which wraps bus messages in secure TLS-protected channels.

4 NETWORK SECURITY TOOLS

[2, c8] [4, c5,c8,c11,c12] [5, c23] [6, c6] [3, c8]

Until now we have discussed attacks and defenses at the protocol level. We will now introduce additional and orthogonal tools to these protocol-level defenses. Many of these tools have become de facto standards on top of the aforementioned security schemes at the protocol level. We only provide a brief overview here. The effective deployment of these tools is covered in detail in the Security Operations & Incident Management CyBOK Knowledge Area [1].

4.1 Firewalling

Firewalls can be co-located with routers or implemented as specialised servers. In either case, they are gatekeepers, inspecting all incoming/outgoing traffic. Firewall systems are typically configured as bastion hosts, i.e., minimal systems hardened against attacks. They apply traffic filters based on a network's security policy and treat all network packets accordingly. The term filter is used for a set of rules configured by an administrator to inspect a packet and perform a matching action, e.g., let the packet through, drop the packet, drop and generate a notification to the sender via ICMP messages. Packets may be filtered according to their source and destination network addresses, protocol type (TCP, UDP, ICMP), TCP or UDP source/destination port numbers, TCP Flag bits (SYN/ACK), rules for traffic from a host or leaving the network via a particular interface and so on. Traditionally, firewalls were pure packet filters, which worked on inspecting header field only. By now, firewalls can also be stateful, i.e., they retain state information about flows and can map packets to streams. While stateful firewalls allow to monitor related traffic and can map communication to flows, this comes at the cost of maintaining (possibly lots of) state.

Rule	State	Src IP	Src Port	Dst IP	Dst Port	Proto	Action
#1	NEW	172.16.0.0/24	*	*	80, 443	TCP	ACCEPT
#2	NEW	*	*	172.16.20.5	22	TCP	ACCEPT
#3	ESTABLISHED	*	*	*	*	TCP	ACCEPT
#4	*	*	*	*	*	*	DROP

Figure 6: Firewalling example. Rule #1 allows outgoing HTTP(S), rule #2 allows incoming SSH.

Figure 6 shows a simple example firewall configuration. All internal hosts (here, in network `172.16.0.0/24`) are allowed to communicate to TCP ports 80/443 for HTTP/HTTPS to external hosts (rule #1). External hosts can connect to an internal SSH server via TCP on port 22 (rule #2). All follow-up communication of these connections is granted (rule #3). Any other communication is dropped (rule #4). In reality, firewall configurations can become incredibly more complex than this minimal example. Specifying complete and coherent policies is typically hard. It typically helps to first lay out a firewall decision diagram, which is then—ideally automatically—transferred into concrete, functionally equivalent firewall policies [84]. Tools like Firewall Builder [85] or Capirca [86] can assist in this process.

Application Gateway (AG): Application gateways, aka application proxies, perform access control and thus facilitate any additional requirements of user authentication before a session is admitted. These AGs can also inspect content at the application layer, unless fully encrypted. In a typical setting, the application gateway will use a firewall's services after performing authentication and policy enforcement. A client wanting to access an external service would connect to the AG first. The AG would prompt them for authentication before initiating a session to the external server. The AG would now establish the connection with the destination acting as a relay on behalf of the client, essentially creating two sessions like a PITM. Another interesting application of an AG is TLS termination. An incoming webserver TLS connection could be terminated at the AG, so that it could do the resource intensive encryption/decryption and pass the un-encrypted traffic to the back-end servers. In practice, the AGs are also configured to inspect encrypted outbound traffic where the clients are configured with corresponding certificates installed at the AG.

Circuit-level Gateway (CG): A CG is a proxy that functions as a relay for TCP connections, thus allowing hosts from a corporate Intranet to make TCP connections over the Internet. CGs are typically co-located with a firewall. The most widely used CG today is SOCKS. For end user applications, it runs transparently as long as the hosts are configured to use SOCKS in place of a standard socket interface. A CG is simple to implement compared to an AG, as it does not need to understand application layer protocols.

DMZ: Network design ensures careful firewall placements by segmenting networks. Typically, the Demilitarised Zone (DMZ) (aka a perimeter network) is created. All external untrusted users are restricted from using the services available in this zone. Typically, an organisation's public web server and authoritative DNS would reside in the DMZ. The rest of the network is partitioned into several security zones by a security architect. For example, a payment database would be deployed to an isolated network, so would an internal file server.

4.2 Intrusion Detection and Prevention Systems

Intrusion Detection Systems (IDSs) can provide valuable information about anomalous network behaviour. They inspect payload, higher-layer information and many more attributes of sessions beyond what a firewall can do. An IDS would monitor network traffic with the help of agents/sensors/monitors on the network and sets off alarms when it detects (or thinks it has) suspicious activity. Essentially, the IDS would compare the traffic against what it considers normal traffic and, using a range of techniques, would generate an alert. IDSs can operate purely on traffic statistics that can be derived from header data. Alternatively, Deep Packet Inspection (DPI) allows to inspect transport- or application-layer payloads to recognize known malicious communication patterns (e.g., of malware). There are several widely-used IDSs such as Snort [87], Zeek [88] or Suricata [89]. IDSs have been used in numerous contexts, such as for detecting malware [90, 91, 92, 93], attacks in automotive networks [94] or against unmanned vehicles [95], software exploits [96], DoS attacks [97] or attacks in wireless ad-hoc networks [98].

The accuracy of an IDS remains a fundamental operational challenge. False alarms are a huge problem for network/security administrators despite decades of research. An IDS may generate false positives for legitimate hosts carrying out suspicious yet benign behaviour.

Likewise, a false negative occurs when malicious activity remains undetected.

Signature-based IDSs compare monitored traffic against a database of known malicious communication patterns. The database has to be continually updated, and despite all efforts, will never be complete. Signatures can be as simple as a source/destination IP address or other protocol headers, or match payload patterns. Rule specification goes beyond the scope of this chapter and is covered by more detailed textbooks [99, 100]. The following toy rule checks generates an alert if the payload of TCP/80 connection to network 192.168.5.7/24 contains a 'GET' string.

```
alert tcp any any -> 192.168.5.7/24 80
(content:"GET"; msg:"GET has been detected";)
```

A signature-based IDS generates a heavy workload, as it has to compare huge numbers of signatures. Speed of detection plays a key role in preventing these attacks. Several systems deploy parallel and distributed detection systems that can cope with high traffic rates on large networks and allow online detection; others exploit parallelism at the hardware level in order to overcome processing delays so that packets and flows can be processed at high speeds.

Anomaly-based IDSs compare monitored traffic to behavioral models built previously "normal" traffic during a learning phase. Instead of blocking certain patterns, anomaly detection allows all communication it deems as benign, and blocks the rest. The fundamentally hard problem here is to capture normal traffic that is both clean (i.e., does not contain malicious behavior) and sufficiently representative (i.e., it also captures benign behavior that will arise in the future). For example, an anomaly-based system based on statistical features could capture bandwidth usage, protocols, ports, arrival rate and burstiness [101]. In this example, a large percentage of port scans would be anomalous and generate an alert. Despite using machine learning techniques, anomaly-based IDSs accuracy remains unsatisfying in practical deployments [102].

Another way of classifying IDSs is the point of monitoring for malicious behaviour. A Host Intrusion Detection System (HIDS) runs on individual hosts in the network. Most virus scan software would have this feature where they also monitor inbound and outbound traffic in addition to the usual virus scanning. This can be particularly helpful if the hosts have been compromised and form part of a bot to attack other servers/networks. In contrast, a network intrusion detection system is deployed at strategic locations within the network to monitor inbound and outbound traffic to and from the devices in various segments of the network.

Intrusion Prevention System (IPS): An IPS distinguishes itself from an IDS in that it can be configured to block potential threats by setting filtering criteria on routers/switches at various locations in the network. IPS systems monitor traffic in real time dropping any suspected malicious packets, blocking traffic from malicious source addresses or resetting suspect connections. In most cases, an IPS would also have IDS capabilities.

4.3 Network Security Monitoring

Several other network monitoring tools help to understand the security situation of a network. We will briefly discuss these monitoring methodologies and mention example uses cases.

Flow monitoring standards such as NetFlow [103] or IPFIX [104] aggregate statistical information of all communication streams within a network. They provide a sweet spot between recording all network communication and nothing at all. Flow aggregation typically requires little computational resources and has low storage demands, enabling for long-term storage. Flow data comes handy during network forensics, or even as input for anomaly detection [105].

Network forensics enable administrators to extract application payload observed on their networks. When used as sniffer or when applied on recorded traffic, frameworks such as NetworkMiner [106] or Xplico [107] can, e.g., extract files, emails and HTTP sessions. They often come with further functionality to fingerprint the network hosts and to map network hosts to locations. Having said this, unless augmented with the according key material, these forensic tools are limited to analyzing non-secure communication.

Network scans allow network administrators to enumerate hosts and services within their network (or, optionally, the entire Internet). There are numerous tools such as Nmap [108] or Zmap [55] that can send, e.g., ICMP and SYN probes at scale.

IP telescopes are publicly reachable network ranges that do not host any service or client. Given these networks are still routed, though, one can monitor any traffic sent to them and derive interesting observations. For example, IP telescopes help observing network scans by others [109]. Similarly, they allow to spot backscatter [110], i.e., responses of traffic that attackers have provoked when using the telescope's IP addresses in IP spoofing attacks (e.g., when assigning random IP addresses during SYN floods).

Honeypots are system used by defenders to trap attackers. They are intentionally vulnerable yet well-isolated client or server systems that are exposed to attackers. There is a wide diversity of client-side honeypots (e.g., to emulate browser vulnerabilities [111]) and server-side honeypots (e.g., to emulate service vulnerabilities [112, 113], or to attract DDoS attacks [114]). Observing the techniques attackers use to exploit these honeypots gives valuable insights into tactics and procedures.

Network reputation services can help to assess the trustworthiness of individual network entities such as IP addresses or domain names. Based on past behaviour observed from an entity, these mostly commercial providers publish a score that serves as reputation for others. Identifying badly reputed hosts in network traffic can help to detect known attackers or connections to botnets. Reputation services are, however, limited in coverage and accuracy due to volatile domaind and IP address usage of attacking hosts.

Finally, Security Information and Event Management (SIEM) systems collect events from security-critical sensors (e.g., IDS, firewalls, host-based sensors, system log files). A SIEM system then analyses these events to distill and raise security-critical incidents for further inspection. It is particularly the combination of multiple data sources (system log files, host-based anomaly sensors, firewall or IDS events) that makes SIEM so successful in detecting, e.g., brute force attacks, worm propagations, or scans.

4.4 SDN and NFV Security

The SDN architecture enables for novel threat detection and attack prevention capabilities [115, 116]. For example, the central SDN controller(s) can more accurately infer DDoS attacks, and automate mitigation strategies by dynamically reprogram switches to drop malicious traffic flows. Furthermore, infected hosts can automatically be routed to an isolated network region (sometimes called a *walled garden*) issuing quarantine notifications to users of infected systems. SDN allows for such an immediate network isolation via software-controlled changes in the network—which was tedious reconfiguration labor before. Similarly, network designs can be rapidly scaled to higher loads.

At the same time, the SDN management plane offers a unique attack vector. Intruders gaining power over the SDN controller undermine any security guarantees that SDN bring, and have additionally the power to reconfigure networks at will. It is therefore of utmost importance that the SDN controller software and the underlying platform follow strong security best practices, including hardened software and strict access control. Furthermore, the SDN platform has to be protected against new SDN-specific threats. For example, the SDN controllers use a Spanning Tree Algorithm (SPTA) for topology updates. In a DoS attack, an adversary could advertise a fake link and force the SPTA to block legitimate ports. Similarly, being in a central position, SDN controllers can be target of DoS attacks [117]. Furthermore, Hong et al. [118] provide a number of attack vectors on practical SDN switch implementations. SDN switches are also prone to a *timing side channel attack* [119]. For example, attackers can send a packet and measure the time it takes the switch to process this packet. For a new packet, the switch will need to fetch a new rule from the controller, thus resulting in additional delay over the flows that already have rules installed at the switch. Consequently, the attacker can determine whether an exchange between an IDS and a database server has taken place, or whether a host has visited a particular website. A possible countermeasure would introduce delay for the first few packets of every flow even if a rule exists [120]. A more extensive analysis of SDN vulnerabilities in general can be found in a study by Zerkane et al. [121].

Network Functions Virtualisation (NFV) aims to reduce capex and allow for the rapid introduction of new services to the market. Specialised network middleboxes such as firewalls, encoders/decoders, DMZs and deep packet inspection units are typically closed black box devices running proprietary software [122]. NFV researchers have proposed the deployment of these middleboxes entirely as virtualised software modules and managed via standardised and open APIs. These modules are called Virtual Network Functions (VNFs). A large number of possible attacks concern the Virtual Machine (Hypervisor) as well as configuring virtual functions. Lal et al. [123] provide a table of NFV security issues and best practice for addressing them. For example, an attacker can compromise a VNF and spawn other new VNFs to change the configuration of a network by blocking certain legitimate ports. The authors suggest hypervisor introspection and security zoning as mitigation techniques. Yang et al. [124] provide a comprehensive survey on security issues in NFV.

4.5 Network Access Control

Networks are typically quite lax about which devices can become part of them. Section 3.4.1 described port-based device authentication devices, which demands secrets from trusted devices. Unfortunately, this still gives no security guarantees on the trustworthiness of network devices. For example, while a device may have been deemed trustworthy at times, system compromises may have caused the system to boot into an untrusted state. Network Access Control, usually implemented using the Trusted Network Connect (TNC) architecture [125], enforces configurable security policies of devices when they join networks. These policies enforce that network clients have been booted into trustworthy configurations. This may even enable firewalls to precisely attribute which client software has caused certain traffic [126]. Technically, to perform remote attestation, the verifier relies on unforgeable trusted hardware on the proving device. Technical details can be found in the Hardware Security CyBOK Knowledge Area [127].

The main practical drawback of such policies is that they attestate only the initial system state. Malicious runtime modifications to the system are not possible with TNC. Validating if a device was compromised *after* it entered a trusted boot state is still subject to research, e.g., in runtime-attestation schemes via remote control-flow enforcement [128, 129].

4.6 Zero Trust Networking

Zero trust networks radically give up the idea of blindly trusting devices within a assumed-to-be trusted part of a network. In zero trust networks, all devices are untrusted unless proven otherwise. This represent a paradigm shift which arose out of the challenges in defining centralized perimeters (e.g., a firewall) that split networks into trusted/untrusted domains. The main motivation here is that traditional networks keep losing control over which devices join the seemingly trusted side of a network (fueled by, e.g., bring-your-own-device). At the same time, devices temporarily jump from trusted to untrusted networks (e.g., work-from-home), before rejoining the trusted networks in a potentially modified state.

Migrating traditional network designs to zero trust networks is not trivial. NCSC provides a comprehensive tutorial which can serve as a great starting point [130]. In essence, a transition to zero trust networks first requires a deep understanding of the assets in a network, such as users, devices, services and data. Similarly, administrators require capabilities to measure the security state of these assets. Any request to services must be authorized using strong multi-factor authentication (described in the Authentication, Authorisation & Accountability CyBOK Knowledge Area [131]), ideally paired with a single sign-on scheme not to frustrate users. Not all legacy services can readily be plugged into such a zero-trust setting, requiring adaptations to make them compatible to standard authentication schemes (e.g., OpenID Connect, OAuth, SAML).

One popular example of such a zero trust network design is *BeyondCorp* [132]. It leverages network access control (see Section 4.5) to identify devices, and rigorously enforces user identification using a centralized single sign-on system. In BeyondCorp, previously internal application services become external ones, protected by an access proxy that rigorously enforces strong encryption and access control.

4.7 DoS Countermeasures

Denial of Service (DoS) attacks can roughly be categorized into two categories, depending on which resources they aim to exhaust. First, in volumetric DoS attacks, adversaries aim to exhaust the network bandwidth of a victim. Amplification attacks (see Section 3.2.4) are the most dominant instance of such attacks, but also large-scale Distributed Denial of Service (DDoS) attacks from remote-controlled botnets can leverage high attack bandwidths. Attack targets are typically individual services or networks, yet can also be entire links in the upper Internet hierarchy (and their depending ASs) that become congested [133, 134]. Volumetric attacks can be mitigated most effectively when traffic is stopped as early as possible before it reaches the target network. For example, commercial so-called scrubbing services help to filter malicious network traffic before it reaches the target. Technically, scrubbing services are high-bandwidth network providers that—with the help of their customers—place themselves between the Internet and an organization's perimeter. Alternatively, attack victims can null route traffic towards certain subnetworks via BGP advertisements to drop their traffic, or use BGP FlowSpec to filter traffic at powerful edge routers.

Second, in application-level DoS attacks, miscreants aim to cripple resources at the software layer. They typically aim to exhaust memory or computation resources (e.g., CPU). Here, defenses are quite application specific. For example, SYN cookies (see Section 3.2.3) and rate limiting protect TCP-based applications against connection floods. Also, CAPTCHAs may help to further distinguish between human- and or computer-generated communication, which is especially useful in the Web context.

5 CONCLUSION

[4, c5] [6, c8,c11] [3, c6]

5.1 The Art of Secure Networking

We covered a broad arsenal of network security instruments and schemes that network architects, network operators and communication partners can employ. It is important to understand that there is no silver bullet to obtain "a secure" network or communication. Unfortunately, it is rarely even possible to *guarantee* that none of the security goals will be broken ever. Consequently, it requires a thorough combination of these principles to obtain a reasonable and satisfactory level of security.

Having said this, there are fundamentals that we should strive for, and for which we have proven and standardized means. Endpoints can securely communicate using TLS. Sometimes these endpoints do however not represent the final recipients of sensitive data, such as for email or messenger servers which may "buffer" messages until the recipient fetches them. In these cases, we can deploy asynchronous end-to-end security schemes at the application layer (e.g., PGP/SMIME) that tolerate untrusted middle hops. Network operators must be prepared for attackers from within their network, and from outsiders. To protect against external threats, they can deploy zero trust networking, or use firewalls for more centralized architectures. On top, an IDS helps to identify threats at the payload level that were unnoticed by the firewall, and network monitoring in general will allow for *a posterio* network forensics. Insider attacks are much harder to mitigate, especially if trusted devices have been compromised by attackers, yet port-based authentication or even network access control are good starting points. In any case, decent network defenses require a fundamental interplay of several security best practices.

5.2 Further Network Security Topics

Network security is too broad to be fully captured at reasonable detail within a single knowledge area. Therefore, we have excluded several topics that are closely related, yet (still) relatively special. We will very briefly outline them in the following.

Cloud and Data Center Security: As soon as organizations outsource computations (cloud) or information (data center), this immediately triggers security demands. They are only partially connected to network security, though. For example, to not expose data and computations to cloud operators, clients can store data in hardware-backed secure containers such as Intel SGX [135]. Data centers and clouds run into risks that adversaries may abuse side channel to leak sensitive data from co-located services or systems—a topic that is discussed in detail in the Operating Systems & Virtualisation CyBOK Knowledge Area [136].

Delay-Tolerant Networks and Ad-hoc Sensors Networks: Not all networks guarantee that communication partners are online, reachable and responsive all the time. Sensor networks are one such example, where energy-constrained sensors just wake up periodically to exchange information, and usually hibernate. Similarly, the speed of light implies that devices in networks in space have significant message inter-arrival times (e.g., already over 2 seconds between Earth and Moon). In general, this requires delay-tolerant networks, which are incompatible with many of the aforementioned security principles, most of which assume reliable and quick responsiveness of communication endpoints. A detailed treatment of this subject goes beyond this KA. A great starting point for further reading is Ivancic' security analysis on delay-tolerant networks [137].

Network Covert Channels: Network covert channels aim to hide the pure existence of communication, e.g., using steganography. They allow two or more collaborating attacker processes to leak sensitive information despite network policies that should prevent such leakage. For example, attackers may encode sensitive information in TCP headers that will remain unnoticed by IDS [138]. Similar covert channels are possible for other protocols, such as DNS [139] or IP [140]. Covert channels can be confined by carefully modeling and observing all protocols fields or patterns in general that could be abused for hiding information [141].

Payment Networks: The banking sector foresees its own proprietary standards and network protocols. Exploring those in detail goes beyond the scope of this document, particularly also because protocols can be even specific to certain regions (e.g., *FinTS* in Germany) or special purposes (e.g., *3-D Secure* for securing credit card transactions). The rise of digital currencies such as Bitcoin which implement several protocols on their own add further complexity. Finally, stock exchanges nowadays heavily depend on reliable networks, and are extremely sensitive to timing attacks that require careful Quality-of-Service assurances [142, 143].

Physical-Layer Security: Our security analyses stopped at the logical part of the link layer. The physical part of this layer deserves further attention and indeed is a subject on its own. In fact, we witnessed several recent advancements in this field, such as Bluetooth Low Energy, distance bounding and positioning protocols, Near-Field Communication (NFC) or cellular networks. For a detailed treatment of this subject, we refer to the Physical Layer and Telecommunications Security CyBOK Knowledge Area [67].

Networking Infrastructure Security: We have so far assumed that networking components are fully trusted. However, with global supply chains that involve dozens of parties and countries during manufacturing a component, such assumption may be easily invalidated in practice. What happens if network infrastructure, which often is part of critical infrastructures, *cannot* be trusted, e.g., due to backdoors or software vulnerabilities? Answering this question is far from trivial, as it depends on which components and which security guarantees are at stake. One recent real-world example of such an analysis happens in 5G networks, where some countries ban hardware that is delivered by some other countries, simply because of lacking trust. This quickly turns into a non-networking issue that finds its solutions in other chapters, such as in the Software Security CyBOK Knowledge Area [144], the Secure Software Lifecycle CyBOK Knowledge Area [145] or Hardware Security CyBOK Knowledge Area [127]. Discussing non-trustworthy networking components goes beyond the scope of this chapter.

Cross-Border Regulations: Networks that span several countries and thus legislations are quite interesting from a law perspective. There may be conflicts of law, e.g., regarding patents, export restrictions, or simply the question whether or not a digital signature is legally binding. These topics are addressed in depth in the Law & Regulation CyBOK Knowledge Area [146].

CROSS-REFERENCE OF TOPICS VS REFERENCE MATERIAL

	[2]	[4]	[5]	[6]	[3]
1 Security Goals and Attacker Models	c8	c1	c1	c6	c8
2 Networking Applications	c1				c1
3.1 Security at the Application Layer	c8		c6,c15,c19–c22		c8
3.2 Security at the Transport Layer	c8	c4,c6			c8
3.3 Security at the Internet Layer	c8	c5,c9	c17		c8
3.4 Security on the Link Layer	c8	c7			c8
4 Network Security Tools	c8	c5,c8,c11,c12	c23	c6	c8
5 Conclusion		c5		c8,c11	c6

REFERENCES

[1] H. Debar, *The Cyber Security Body of Knowledge*. University of Bristol, 2021, ch. Security Operations & Inicident Management, version 1.0.2. [Online]. Available: https://www.cybok.org/

[2] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach (7th Edition)*, 7th ed. Pearson, 2017.

[3] A. S. Tanenbaum and D. Wetherall, *Computer Networks*, 5th ed. Prentice Hall, 2011.

[4] W. Stallings, *Network Security Essentials, Applications and Standards (6th Edition)*, 6th ed. Pearson, 2016.

[5] C. W. Kaufman, R. Perlman, and M. Speciner, *Network Security: Private Communication in a Public World*. Englewood Cliffs, New Jersey: Prentice-Hall, March 1995.

[6] C. P. Pfleeger, S. L. Pfleeger, and J. Margulies, *Security in Computing (5th Edition)*, 5th ed. USA: Prentice Hall Press, 2015.

[7] C. Troncoso, *The Cyber Security Body of Knowledge*. University of Bristol, 2021, ch. Privacy & Online Rights, version 1.0.2. [Online]. Available: https://www.cybok.org/

[8] N. Smart, *The Cyber Security Body of Knowledge*. University of Bristol, 2021, ch. Cryptography, version 1.0.1. [Online]. Available: https://www.cybok.org/

[9] K. G. Paterson, *The Cyber Security Body of Knowledge*. University of Bristol, 2021, ch. Applied Cryptography, version 1.0.0. [Online]. Available: https://www.cybok.org/

[10] D. Dolev and A. C. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–207, 1983. [Online]. Available: https://doi.org/10.1109/TIT.1983.1056650

[11] "Modbus," https://www.modbus.org/specs.php, accessed: 2021-03-23.

[12] "Knx," https://www.knx.org, accessed: 2021-03-23.

[13] I. O. for Standardization, "Road vehicles — Controller area network (CAN) — Part 1: Data link layer and physical signalling," International Organization for Standardization, Geneva, CH, Standard, Dec. 2015.

[14] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *International Workshop on Peer-to-Peer Systems*. Springer, 2002, pp. 53–65.

[15] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," in *Designing privacy enhancing technologies*. Springer, 2001, pp. 46–66.

[16] N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith, "Sok: Secure messaging," in *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*. IEEE Computer Society, 2015, pp. 232–249. [Online]. Available: https://doi.org/10.1109/SP.2015.22

[17] A. Whitten and J. D. Tygar, "Why johnny can't encrypt: A usability evaluation of pgp 5.0." in *USENIX Security Symposium*, vol. 348, 1999, pp. 169–184.

[18] G. Stringhini, *The Cyber Security Body of Knowledge*. University of Bristol, 2021, ch. Adversarial Behaviours, version 1.0.1. [Online]. Available: https://www.cybok.org/

[19] S. Fahl, *The Cyber Security Body of Knowledge*. University of Bristol, 2021, ch. Web & Mobile Security, version 1.0.1. [Online]. Available: https://www.cybok.org/

[20] S. Bortzmeyer, "DNS Query Name Minimisation to Improve Privacy," RFC 7816, Mar. 2016. [Online]. Available: https://rfc-editor.org/rfc/rfc7816.txt

[21] "Internet society — dnssec deployment maps," https://www.internetsociety.org/deploy360/dnssec/maps/, accessed: 2021-02-19.

[22] S. Singanamalla, S. Chunhapanya, J. Hoyland, M. Vavruša, T. Verma, P. Wu, M. Fayed, K. Heimerl, N. Sullivan, and C. Wood, "Oblivious dns over https (odoh): A practical privacy enhancement to dns," in *DNS Privacy Workshop 2021*, 2021.

[23] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 2046–2069, Fourth 2013.

[24] C. Rossow, "Amplification Hell: Revisiting Network Protocols for DDoS Abuse," in *Proceedings of the 2014 Network and Distributed System Security (NDSS) Symposium*, February 2014.

[25] M. Kührer, T. Hupperich, C. Rossow, and T. Holz, "Exit from Hell? Reducing the Impact of Amplification DDoS Attacks," in *Proceedings of the 23rd USENIX Security Symposium*, August 2014.

[26] A. Malhotra, I. E. Cohen, E. Brakke, and S. Goldberg, "Attacking the network time protocol," *IACR Cryptology ePrint Archive*, vol. 2015, p. 1020, 2015.

[27] T. Rytilahti, D. Tatang, J. Köpper, and T. Holz, "Masters of Time: An Overview of the NTP Ecosystem," in *IEEE EuroS&P*, 2018.

[28] O. Deutsch, N. R. Schiff, D. Dolev, and M. Schapira, "Preventing (network) time travel with chronos," in *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018.

[29] M. Castro and B. Liskov, “Practical byzantine fault tolerance and proactive recovery,” *ACM Trans. Comput. Syst.*, vol. 20, no. 4, p. 398–461, Nov. 2002. [Online]. Available: https://doi.org/10.1145/571637.571640

[30] G. Urdaneta, G. Pierre, and M. van Steen, “A survey of dht security techniques,” *ACM Comput. Surv.*, vol. 43, p. 8, 01 2011.

[31] J. R. Douceur, “The sybil attack,” in *Peer-to-Peer Systems*, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 251–260.

[32] R. A. Bazzi and G. Konjevod, “On the establishment of distinct identities in overlay networks,” *Distrib. Comput.*, vol. 19, no. 4, p. 267–287, Mar. 2007. [Online]. Available: https://doi.org/10.1007/s00446-006-0012-y

[33] N. Borisov, “Computational puzzles as sybil defenses,” in *Sixth IEEE International Conference on Peer-to-Peer Computing (P2P’06)*, 2006, pp. 171–176.

[34] G. Danezis, C. Lesniewski-Laas, M. F. Kaashoek, and R. Anderson, “Sybil-resistant dht routing,” in *Computer Security – ESORICS 2005*, S. d. C. di Vimercati, P. Syverson, and D. Gollmann, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 305–318.

[35] N. Suri, *The Cyber Security Body of Knowledge*. University of Bristol, 2021, ch. Distributed Systems Security, version 1.0.1. [Online]. Available: https://www.cybok.org/

[36] P. Syverson, R. Dingledine, and N. Mathewson, “Tor: The secondgeneration onion router,” in *Usenix Security*, 2004, pp. 303–320.

[37] S. J. Murdoch and G. Danezis, “Low-cost traffic analysis of tor,” in *2005 IEEE Symposium on Security and Privacy (S P’05)*, 2005.

[38] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. F. Syverson, “Users get routed: traffic correlation on tor by realistic adversaries,” in *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS’13, Berlin, Germany, November 4-8, 2013*, A. Sadeghi, V. D. Gligor, and M. Yung, Eds. ACM, 2013, pp. 337–348.

[39] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, “Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail,” in *2012 IEEE Symposium on Security and Privacy*, 2012, pp. 332–346.

[40] V. Shmatikov and M.-H. Wang, “Timing analysis in low-latency mix networks: Attacks and defenses,” in *Proceedings of the 11th European Conference on Research in Computer Security*, ser. ESORICS’06. Berlin, Heidelberg: Springer-Verlag, 2006, p. 18–33.

[41] J. Karlin, D. Ellard, A. W. Jackson, C. E. Jones, G. Lauer, D. Mankins, and W. T. Strayer, “Decoy routing: Toward unblockable internet communication.” in *FOCI*, 2011.

[42] D. Fifield, C. Lan, R. Hynes, P. Wegmann, and V. Paxson, “Blocking-resistant communication through domain fronting,” *Proceedings on Privacy Enhancing Technologies*, vol. 2015, no. 2, pp. 46–64, 2015.

[43] D. Basin, *The Cyber Security Body of Knowledge*. University of Bristol, 2021, ch. Formal Methods for Security, version 1.0. [Online]. Available: https://www.cybok.org/

[44] K. Bhargavan, B. Blanchet, and N. Kobeissi, “Verified models and reference implementations for the TLS 1.3 standard candidate,” in *IEEE Symposium on Security and Privacy (Oakland)*, 2017, pp. 483–503.

[45] C. Cremers, M. Horvat, J. Hoyland, S. Scott, and T. van der Merwe, “A comprehensive symbolic analysis of tls 1.3,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1773–1788.

[46] B. Laurie, A. Langley, and E. Kasper, “Certificate Transparency,” RFC 6962, Jun. 2013. [Online]. Available: https://rfc-editor.org/rfc/rfc6962.txt

[47] N. Weaver, R. Sommer, and V. Paxson, “Detecting forged tcp reset packets,” in *in Proceedings of the Network and Distributed System Security Symposium, NDSS 2009*. The

Internet Society, 09 2009.
[48] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," RFC 9000, May 2021. [Online]. Available: https://rfc-editor.org/rfc/rfc9000.txt
[49] F. Baker and P. Savola, "Ingress Filtering for Multihomed Networks," RFC 3704, Mar. 2004. [Online]. Available: https://rfc-editor.org/rfc/rfc3704.txt
[50] A. Herzberg and H. Shulman, "Fragmentation considered poisonous, or: One-domain-to-rule-them-all.org," in *2013 IEEE Conference on Communications and Network Security (CNS)*, 2013, pp. 224–232.
[51] "Openvpn," https://openvpn.net/, accessed: 2021-03-23.
[52] S. Frankel and S. Krishnan, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap," RFC 6071, Feb. 2011. [Online]. Available: https://rfc-editor.org/rfc/rfc6071.txt
[53] C. Kaufman, P. E. Hoffman, Y. Nir, P. Eronen, and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)," RFC 7296, Oct. 2014. [Online]. Available: https://rfc-editor.org/rfc/rfc7296.txt
[54] J. Turner, M. J. Schertler, M. S. Schneider, and D. Maughan, "Internet Security Association and Key Management Protocol (ISAKMP)," RFC 2408, Nov. 1998. [Online]. Available: https://rfc-editor.org/rfc/rfc2408.txt
[55] Z. Durumeric, E. Wustrow, and J. A. Halderman, "Zmap: Fast internet-wide scanning and its security applications," in *22nd {USENIX} Security Symposium ({USENIX} Security 13)*, 2013, pp. 605–620.
[56] C. Grundemann, "Ipv6 security myth – no ipv6 nat means less security," https://www.internetsociety.org/blog/2015/01/ipv6-security-myth-3-no-ipv6-nat-means-less-security/, accessed: 2021-03-23.
[57] "Border Gateway Protocol (BGP)," RFC 1163, Jun. 1990. [Online]. Available: https://rfc-editor.org/rfc/rfc1163.txt
[58] "Border Gateway Protocol 3 (BGP-3)," RFC 1267, Oct. 1991. [Online]. Available: https://rfc-editor.org/rfc/rfc1267.txt
[59] K. Butler, T. R. Farley, P. McDaniel, and J. Rexford, "A survey of bgp security issues and solutions," *Proceedings of the IEEE*, vol. 98, no. 1, pp. 100–122, 2010.
[60] M. Lad, R. Oliveira, B. Zhang, and L. Zhang, "Understanding resiliency of internet topology against prefix hijack attacks," in *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, 2007, pp. 368–377.
[61] M. Lepinski and S. Kent, "An Infrastructure to Support Secure Internet Routing," RFC 6480, Feb. 2012. [Online]. Available: https://rfc-editor.org/rfc/rfc6480.txt
[62] M. Lepinski and K. Sriram, "BGPsec Protocol Specification," RFC 8205, Sep. 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8205.txt
[63] C. Hall, R. Anderson, R. Clayton, E. Ouzounis, and P. Trimintzios, "Resilience of the internet interconnection ecosystem," in *Economics of Information Security and Privacy III*. Springer, 2013, pp. 119–148.
[64] Q. Li, Y.-C. Hu, and X. Zhang, "Even rockets cannot make pigs fly sustainably: Can bgp be secured with bgpsec?" in *Proceedings of the NDSS Workshop on Security of Emerging Network Technologies (SENT'14)*, Feb. 2014.
[65] X. Zhang, H. Hsiao, G. Hasker, H. Chan, A. Perrig, and D. G. Andersen, "SCION: scalability, control, and isolation on next-generation networks," in *32nd IEEE Symposium on Security and Privacy, S&P 2011, 22-25 May 2011, Berkeley, California, USA*. IEEE Computer Society, 2011, pp. 212–227. [Online]. Available: https://doi.org/10.1109/SP.2011.45
[66] A. Azimov, E. Bogomazov, R. Bush, K. Patel, and J. Snijders, "Verification of AS_PATH Using the Resource Certificate Public Key Infrastructure and Autonomous

System Provider Authorization," Internet Engineering Task Force, Internet-Draft draft-ietf-sidrops-aspa-verification-07, Feb. 2021, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-sidrops-aspa-verification-07

[67] S. Čapkun, *The Cyber Security Body of Knowledge*. University of Bristol, 2021, ch. Physical Layer & Telecommunications Security, version 1.0.1. [Online]. Available: https://www.cybok.org/

[68] D. Simon, R. Hurst, and D. B. D. Aboba, "The EAP-TLS Authentication Protocol," RFC 5216, Mar. 2008. [Online]. Available: https://rfc-editor.org/rfc/rfc5216.txt

[69] "arpwatch - keep track of ethernet/ip address pairings," https://linux.die.net/man/8/arpwatch, accessed: 2021-02-25.

[70] T. Kiravuo, M. Sarela, and J. Manner, "A survey of Ethernet LAN security," *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1477–1491, Third 2013.

[71] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks," RFC 7348, Aug. 2014. [Online]. Available: https://rfc-editor.org/rfc/rfc7348.txt

[72] A. Stubblefield, J. Ioannidis, and A. D. Rubin, "Using the Fluhrer, Mantin, and Shamir attack to break WEP," in *NDSS*, 2002.

[73] S. R. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of RC4," in *Selected Areas in Cryptography, 8th Annual International Workshop, SAC 2001 Toronto, Ontario, Canada, August 16-17, 2001, Revised Papers*, ser. Lecture Notes in Computer Science, S. Vaudenay and A. M. Youssef, Eds., vol. 2259. Springer, 2001, pp. 1–24.

[74] E. Tews and M. Beck, "Practical attacks against WEP and WPA," in *Proceedings of the Second ACM Conference on Wireless Network Security*, ser. WiSec '09. New York, NY, USA: ACM, 2009, pp. 79–86.

[75] M. Vanhoef and F. Piessens, "Key reinstallation attacks: Forcing nonce reuse in WPA2," in *Proceedings of the 24th ACM Conference on Computer and Communications Security (CCS)*. ACM, 2017.

[76] C. He, M. Sundararajan, A. Datta, A. Derek, and J. C. Mitchell, "A modular correctness proof of IEEE 802.11i and TLS," in *Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS 2005, Alexandria, VA, USA, November 7-11, 2005*, V. Atluri, C. A. Meadows, and A. Juels, Eds. ACM, 2005, pp. 2–15.

[77] J. Jonsson, "On the security of CTR + CBC-MAC," in *Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002. Revised Papers*, ser. Lecture Notes in Computer Science, K. Nyberg and H. M. Heys, Eds., vol. 2595. Springer, 2002, pp. 76–93.

[78] D. Harkins and W. A. Kumari, "Opportunistic Wireless Encryption," RFC 8110, Mar. 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8110.txt

[79] "Internet society — dnssec deployment maps," https://www.autosar.org/standards/, accessed: 2021-03-20.

[80] S. Nürnberger and C. Rossow, "Vatican - vetted, authenticated can bus," in *Conference on Cryptographic Hardware and Embedded Systems (CHES)*, 2016.

[81] J. Van Bulck, J. T. Mühlberg, and F. Piessens, "Vulcan: Efficient component authentication and software isolation for automotive control networks," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, ser. ACSAC 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 225–237.

[82] R. Bhatia, V. Kumar, K. Serag, Z. B. Celik, M. Payer, and D. Xu, "Evading voltage-based intrusion detection on automotive can," in *Proceedings of the 2021 Network and Distributed*

System Security (NDSS) Symposium, February 2021.

[83] "Modbussecure," https://modbus.org/docs/MB-TCP-Security-v21_2018-07-24.pdf, accessed: 2021-05-20.

[84] M. G. Gouda and A. X. Liu, "Structured firewall design," *Computer networks*, vol. 51, no. 4, pp. 1106–1120, 2007.

[85] "Firewall builder," https://github.com/fwbuilder/fwbuilder, accessed: 2021-03-23.

[86] "Capirca," https://github.com/google/capirca, accessed: 2021-03-23.

[87] M. Roesch, "Snort: Lightweight intrusion detection for networks," in *Proceedings of the 13th Conference on Systems Administration (LISA-99), Seattle, WA, USA, November 7-12, 1999*, D. W. Parter, Ed. USENIX, 1999, pp. 229–238. [Online]. Available: http://www.usenix.org/publications/library/proceedings/lisa99/roesch.html

[88] V. Paxson, "Bro: a System for Detecting Network Intruders in Real-Time," *Computer Networks*, vol. 31, no. 23-24, pp. 2435–2463, 1999. [Online]. Available: http://www.icir.org/vern/papers/bro-CN99.pdf

[89] "Suricata," https://suricata-ids.org, accessed: 2021-03-23.

[90] K. Rieck, G. Schwenk, T. Limmer, T. Holz, and P. Laskov, "Botzilla: detecting the "phoning home" of malicious software," in *Proceedings of the 2010 ACM Symposium on Applied Computing (SAC), Sierre, Switzerland, March 22-26, 2010*, S. Y. Shin, S. Ossowski, M. Schumacher, M. J. Palakal, and C. Hung, Eds. ACM, 2010, pp. 1978–1984.

[91] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA*, P. C. van Oorschot, Ed. USENIX Association, 2008, pp. 139–154.

[92] G. Gu, J. Zhang, and W. Lee, "Botsniffer: Detecting botnet command and control channels in network traffic," in *Proceedings of the Network and Distributed System Security Symposium, NDSS 2008, San Diego, California, USA, 10th February - 13th February 2008*. The Internet Society, 2008.

[93] G. Gu, P. A. Porras, V. Yegneswaran, and M. W. Fong, "Bothunter: Detecting malware infection through ids-driven dialog correlation," in *Proceedings of the 16th USENIX Security Symposium, Boston, MA, USA, August 6-10, 2007*, N. Provos, Ed. USENIX Association, 2007.

[94] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network," in *2016 International Conference on Information Networking (ICOIN)*, 2016, pp. 63–68.

[95] G. Choudhary, V. Sharma, I. You, K. Yim, R. Chen, and J.-H. Cho, "Intrusion detection systems for networked unmanned aerial vehicles: a survey," in *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE, 2018, pp. 560–565.

[96] N. Provos and T. Holz, *Virtual honeypots: from botnet tracking to intrusion detection*. Pearson Education, 2007.

[97] A. M. Lonea, D. E. Popescu, and H. Tianfield, "Detecting ddos attacks in cloud computing environment," *International Journal of Computers Communications & Control*, vol. 8, no. 1, pp. 70–78, 2012.

[98] Y. Zhang and W. Lee, "Intrusion detection in wireless ad-hoc networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '00. New York, NY, USA: Association for Computing Machinery, 2000, p. 275–283.

[99] A. Orebaugh, S. Biles, and J. Babbin, *Snort cookbook - solutions and examples for Snort administrators*. O'Reilly, 2005. [Online]. Available: http://www.oreilly.de/catalog/

snortckbk/index.html

[100] K. Cox and C. Gerg, *Managing security with Snort and IDS tools - intrusion detection with open source tools*. O'Reilly, 2004. [Online]. Available: http://www.oreilly.de/catalog/snortids/index.html

[101] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys Tutorials*, vol. 10, no. 4, pp. 56–76, Fourth 2008.

[102] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *2010 IEEE Symposium on Security and Privacy*, May 2010, pp. 305–316.

[103] B. Claise, "Cisco Systems NetFlow Services Export Version 9," RFC 3954, Oct. 2004. [Online]. Available: https://rfc-editor.org/rfc/rfc3954.txt

[104] P. Aitken, B. Claise, and B. Trammell, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information," RFC 7011, Sep. 2013. [Online]. Available: https://rfc-editor.org/rfc/rfc7011.txt

[105] Z. Jadidi, V. Muthukkumarasamy, E. Sithirasenan, and M. Sheikhan, "Flow-based anomaly detection using neural network optimized with gsa algorithm," in *2013 IEEE 33rd international conference on distributed computing systems workshops*. IEEE, 2013, pp. 76–81.

[106] "Networkminer," https://www.netresec.com/?page=networkminer, accessed: 2021-03-23.

[107] "Xplico," https://www.xplico.org, accessed: 2021-03-23.

[108] "Nmap: the network mapper," https://nmap.org, accessed: 2021-03-23.

[109] D. Moore, C. Shannon, G. Voelker, S. Savage *et al.*, "Network telescopes: Technical report," Cooperative Association for Internet Data Analysis (CAIDA), Tech. Rep., 2004.

[110] G. Yao, J. Bi, and A. V. Vasilakos, "Passive ip traceback: Disclosing the locations of ip spoofers from path backscatter," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 471–484, 2014.

[111] C. Seifert, I. Welch, P. Komisarczuk *et al.*, "Honeyc-the low-interaction client honeypot," in *NZCSRCS*, vol. 6. Citeseer, 2007.

[112] A. Mairh, D. Barik, K. Verma, and D. Jena, "Honeypot in network security: a survey," in *Proceedings of the 2011 international conference on communication, computing & security*, 2011, pp. 600–605.

[113] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "IoTPOT: Analysing the Rise of IoT Compromises," in *Proceedings of the 9th USENIX Workshop on Offensive Technologies (WOOT '15)*, August 2015.

[114] L. Krämer, J. Krupp, D. Makita, T. Nishizoe, T. Koide, K. Yoshioka, and C. Rossow, "AmpPot: Monitoring and Defending Amplification DDoS Attacks," in *Proceedings of the 18th International Symposium on Research in Attacks, Intrusions and Defenses*, November 2015.

[115] S. Taha Ali, V. Sivaraman, A. Radford, and S. Jha, "A survey of securing networks using software defined networking," *IEEE Transactions on Reliability*, vol. 64, pp. 1–12, 09 2015.

[116] A. Shaghaghi, M. A. Kaafar, and S. Jha, "Wedgetail: An intrusion prevention system for the data plane of software defined networks," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '17. New York, NY, USA: ACM, 2017, pp. 849–861.

[117] S. M. Mousavi and M. St-Hilaire, "Early detection of ddos attacks against sdn controllers," in *2015 International Conference on Computing, Networking and Communications (ICNC)*, 2015, pp. 77–81.

[118] S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning network visibility in software-defined

networks: New attacks and countermeasures," in *Proceedings 2015 Network and Distributed System Security Symposium*. Internet Society, 2015.

[119] S. Liu, M. K. Reiter, and V. Sekar, "Flow reconnaissance via timing attacks on SDN switches," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, June 2017, pp. 196–206.

[120] H. Cui, G. O. Karame, F. Klaedtke, and R. Bifulco, "On the fingerprinting of software-defined networks," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 10, pp. 2160–2173, 2016.

[121] S. Zerkane, D. Espes, P. Le Parc, and F. Cuppens, "Vulnerability analysis of software defined networking," in *Foundations and Practice of Security*, F. Cuppens, L. Wang, N. Cuppens-Boulahia, N. Tawbi, and J. Garcia-Alfaro, Eds. Cham: Springer International Publishing, 2017, pp. 97–116.

[122] Z. G. A. Gember, P. Prabhu and A. Akella, "Toward software defined middlebox networking," in *In Proceedings of the 11th ACM Workshop on Hot Topics in Networks (HotNets)*, Redmond, WA, 10 2012, pp. 7–12.

[123] S. Lal, T. Taleb, and A. Dutta, "NFV: Security threats and best practices," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 211–217, Aug 2017.

[124] W. Yang and C. Fung, "A survey on security in network functions virtualization," in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, June 2016, pp. 15–19.

[125] T. C. Group, "TNC Architecture for Interoperability," Trusted Computing Group, Specification, Oct. 2017. [Online]. Available: https://trustedcomputinggroup.org/resource/tnc-architecture-for-interoperability-specification/

[126] F. Schwarz and C. Rossow, "SENG, the SGX-Enforcing Network Gateway: Authorizing Communication from Shielded Clients," in *29th USENIX Security Symposium (USENIX Security 20)*. Boston, MA: USENIX Association, Aug. 2020.

[127] I. Verbauwhede, *The Cyber Security Body of Knowledge*. University of Bristol, 2021, ch. Hardware Security, version 1.0.1. [Online]. Available: https://www.cybok.org/

[128] T. Abera, N. Asokan, L. Davi, J.-E. Ekberg, T. Nyman, A. Paverd, A.-R. Sadeghi, and G. Tsudik, "C-flat: control-flow attestation for embedded systems software," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 743–754.

[129] G. Dessouky, S. Zeitouni, T. Nyman, A. Paverd, L. Davi, P. Koeberl, N. Asokan, and A.-R. Sadeghi, "Lo-fat: Low-overhead control flow attestation in hardware," in *Proceedings of the 54th Annual Design Automation Conference 2017*, 2017, pp. 1–6.

[130] "Zero trust principles," https://www.ncsc.gov.uk/blog-post/zero-trust-principles-beta-release, accessed: 2021-05-21.

[131] D. Gollmann, *The Cyber Security Body of Knowledge*. University of Bristol, 2021, ch. Authentication, Authorisation & Accountability, version 1.0.2. [Online]. Available: https://www.cybok.org/

[132] V. M. Escobedo, F. Zyzniewski, B. A. E. Beyer, and M. Saltonstall, "Beyondcorp: The user experience," *Login*, vol. tbd, p. tbd, 2017.

[133] A. Studer and A. Perrig, "The coremelt attack," in *European Symposium on Research in Computer Security*. Springer, 2009, pp. 37–52.

[134] M. S. Kang, S. B. Lee, and V. D. Gligor, "The crossfire attack," in *2013 IEEE Symposium on Security and Privacy*, 2013, pp. 127–141.

[135] F. Schuster, M. Costa, C. Fournet, C. Gkantsidis, M. Peinado, G. Mainar-Ruiz, and M. Russinovich, "Vc3: Trustworthy data analytics in the cloud using sgx," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 38–54.

[136] H. Bos, *The Cyber Security Body of Knowledge*. University of Bristol, 2021, ch. Operating

Systems & Virtualisation, version 1.0.1. [Online]. Available: https://www.cybok.org/

[137] W. D. Ivancic, "Security analysis of dtn architecture and bundle protocol specification for space-based networks," in *2010 IEEE Aerospace Conference*. IEEE, 2010, pp. 1–12.

[138] S. H. Sellke, C.-C. Wang, S. Bagchi, and N. Shroff, "Tcp/ip timing channels: Theory to implementation," in *IEEE INFOCOM 2009*. IEEE, 2009, pp. 2204–2212.

[139] V. Paxson, M. Christodorescu, M. Javed, J. Rao, R. Sailer, D. L. Schales, M. Stoecklin, K. Thomas, W. Venema, and N. Weaver, "Practical comprehensive bounds on surreptitious communication over {DNS}," in *22nd {USENIX} Security Symposium ({USENIX} Security 13)*, 2013, pp. 17–32.

[140] N. B. Lucena, G. Lewandowski, and S. J. Chapin, "Covert channels in ipv6," in *International Workshop on Privacy Enhancing Technologies*. Springer, 2005, pp. 147–166.

[141] S. Wendzel, S. Zander, B. Fechner, and C. Herdin, "Pattern-based survey and categorization of network covert channel techniques," *ACM Computing Surveys (CSUR)*, vol. 47, no. 3, pp. 1–26, 2015.

[142] F. Massacci and N. C. Ngo, "Distributed financial exchanges: Security challenges and design principles," *IEEE Security & Privacy*, 2020.

[143] F. Massacci, C. N. Ngo, J. Nie, D. Venturi, and J. Williams, "Futuresmex: secure, distributed futures market exchange," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 335–353.

[144] F. Piessens, *The Cyber Security Body of Knowledge*. University of Bristol, 2021, ch. Software Security, version 1.0.1. [Online]. Available: https://www.cybok.org/

[145] L. Williams, *The Cyber Security Body of Knowledge*. University of Bristol, 2021, ch. Secure Software Lifecycle, version 1.0.2. [Online]. Available: https://www.cybok.org/

[146] R. Carolina, *The Cyber Security Body of Knowledge*. University of Bristol, 2021, ch. Law & Regulation, version 1.0.2. [Online]. Available: https://www.cybok.org/

ACRONYMS

ACN Anonymous Communication Network.

AES Advanced Encryption Standard.

AG Application Gateway.

AH Authentication Header.

AP Access Point.

API Application Programming Interface.

ARP Address Resolution Protocol.

AS Autonomous System.

ASPA Autonomous System Provider Authorization.

AuthS Authentication Server.

AUTOSAR AUTomotive Open System ARchitecture.

BGP Border Gateway Protocol.

BYOD Bring-Your-Own-Device.

CA Certification Authority.

CAM Content Addressable Memory.

CAN Controller Area Network.

CBC Cipher Block Chaining.

CCMP Cipher Block Chaining Message Authentication Code Protocol.

CDS Cross-Domain Solutions.

CG Circuit-level Gateway.

CGA Cryptographically Generated Address.

CHAP Challenge Handshake Authentication Protocol.

CRC Cyclic Redundancy Check.

CSR certificate signing request.

CT Certificate Transparency.

DDoS Distributed Denial of Service.

DHKE Diffie-Hellman Key Exchange.

DHT Distributed Hash Table.

DMZ Demilitarised Zone.

DNS Domain Name System.

DNSSEC DNS Security Extensions.

DoH DNS over HTTPS.

DoS Denial of Service.

DoT DNS over TLS.

DPI Deep Packet Inspection.

DTLS Datagram TLS.

E2E End-to-End.

EAP Extensible Authentication Protocol.

EAP-PEAP EAP Protected Authentication Protocol.

EAP-SIM EAP for GSM Subscriber Identity.

EAP-TLS EAP-Transport Layer Security.

EAPoL Extensible Authentication Protocol over LAN.

ECU Electronic Control Unit.

EIGRP Enhanced Interior Gateway Routing Protocol.

ESP Encapsulation Security Payload.

FCS	Frame Check Sequence.
FMS	Fluhrer, Martin and Shamir.
GCM	Galois Counter Mode.
GPG	GNU Privacy Guard.
GRE	Generic Routing Encapsulation.
GSM	Global System for Mobile Communications.
HDLC	High-Level Data Link Control.
HIDS	Host Intrusion Detection System.
HTTP	Hypertext Transfer Protocol.
HTTPS	Hypertext Transfer Protocol Secure.
ICMP	Internet Control Message Protocol.
ICV	Integrity Check Value.
IDS	Intrusion Detection System.
IETF	Internet Engineering Task Force.
IGP	Interior Gateway Protocol.
IGRP	Interior Gateway Routing Protocol.
IKE	Internet Key Exchange.
IMAP	Internet Message Access Protocol.
IoT	Internet of Things.
IP	Internet Protocol.
IPS	Intrusion Prevention System.
IPsec	Internet Protocol Security.
ISAKMP	Internet Security Association and Key Management Protocol.
ISN	Initial Sequence Number.
ISO	International Organization for Standardization.
ISP	Internet Service Provider.
IV	Initialisation Vector.
KA	Knowledge Area.
KNX	Konnex Bus.
L2TP	Layer 2 Tunneling Protocol.
LAN	Local Area Network.

MAC	Message Authentication Code.
MAC	Media Access Control.
MIME	Multipurpose Internet Mail Extensions.
MK	Master Key.
MPLS	Multiprotocol Label Switching.
MSK	Master Session Key.
MTU	Maximum Transmission Unit.
NAT	Network Address Translation.
NDP	Neighbor Discovery Protocol.
NFC	Near-Field Communication.
NFV	Network Functions Virtualisation.
NIST	National Institute for Standards and Technology.
NS	name server.
NTP	Network Time Protocol.
OCSP	Online Certificate Status Protocol.
OS	Operating System.
OSI	Open Systems Interconnection.
OSPF	Open Shortest Path First.
OWE	Opportunistic Wireless Encryption.
P2P	Peer-to-Peer.
PAP	Password Authentication Protocol.
PEAP	Protected Extensible Authentication Protocol.
PGP	Pretty Good Privacy.
PITM	Person In The Middle.
PKC	Public Key Crytography.
PKC	Public Key Certificate.
PKI	Public-Key Infrastructure.
PMK	Pairwise Master Key.
PMS	Pre-Master Secret.
POP	Post Office Protocol.
PPP	Point-to-Point Protocol.
PPPoE	PPP over Ethernet.

PPTP Point-to-Point Tunneling Protocol.

PRF Pseudo-Random Function.

PRNG Pseudo Random Number Generator.

PSK Pre-Shared Key.

PTK Pairwise Transient Key.

QUIC Quick UDP Internet Connections.

RARP Reverse Address Resolution Protocol.

RIP Routing Information Protocol.

RIR Regional Internet Registry.

ROA Route Origin Authorization.

ROV Route Origin Validation.

RPKI Resource Public Key Infrastructure.

RSA Rivest-Shamir-Adleman.

RSN Robust Secure Networking.

SA Security Association.

SAD Security Association Database.

SAE Simultaneous Authentication of Equals.

SCADA Supervisory Control and Data Acquisition.

SDN Software Defined Networking.

SIEM Security Information and Event Management.

SMIME Secure Multipurpose Internet Mail Extensions.

SMTP Simple Mail Transfer Protocol.

SPTA Spanning Tree Algorithm.

SSL Secure Sockets Layer.

SSTP Secure Socket Tunneling Protocol.

TA Transmitter MAC Address.

TCP Transmission Control Protocol.

TFC Traffic Flow Confidentiality.

TK Temporal Key.

TKIP Temporal Key Integrity Protocol.

TLS Transport Layer Security.

TMAC Temporal MAC.

TNC Trusted Network Connect.

UDP User Datagram Protocol.

URL Uniform Resource Locator.

uRPF Unicast Reverse Path Forwarding.

USB Universal Serial Bus.

UTC Coordinated Universal Time.

VLAN Virtual LAN.

VNF Virtual Network Function.

VPN Virtual Private Network.

VXLAN Virtual eXtensible LAN.

WAN Wide Area Network.

WEP Wired Equivalent Privacy.

WLAN Wireless LAN.

WPA Wi-Fi Protected Access.

GLOSSARY

CyBOK Refers to the Cyber Security Body of Knowledge.

Internet The Internet is the single, interconnected, worldwide system of commercial, governmental, educational, and other computer networks that share (a) the protocol suite specified by the Internet Architecture Board (IAB), and (b) the name and address spaces managed by the Internet Corporation for Assigned Names and Numbers (ICANN).(Source = NIST IR 7298r2).

INDEX