Exploring Encryption Algorithms and Network Protocols: A Comprehensive Survey of Threats and Vulnerabilities

Jemin Ahn, Rasheed Hussain, Kyungtae Kang, and Junggab Son

***Abstract*—Cryptographic network protocols play a crucial role in enabling secure data exchange over insecure media in modern network environments. However, even minor vulnerabilities can make protocols an easy target for cyber attackers. Therefore, it is essential to investigate the threats and vulnerabilities stemming from the cryptographic network protocols. Furthermore, it is necessary to comprehensively investigate the weaknesses of network protocols that use cryptographic primitives to inform users and developers about potential attack points. This comprehensive survey examines the relationship between encryption schemes and network protocols and presents an in-depth review of associated threats and vulnerabilities. Given that most cryptographic protocols operate in the Transport and Application layers of the Transmission Control Protocol/Internet Protocol (TCP/IP) protocol stack, our investigation primarily centers around encryption algorithms used by representative and notable cryptographic network protocols such as Transport Layer Security (TLS) and Secure Shell (SSH). Furthermore, we delve into the attackers' methods to exploit the already identified and existing vulnerabilities, seeking to understand the mechanisms employed to compromise these protocols. Through this survey, we aim to provide the readership with an in-depth understanding of the existing and new vulnerabilities associated with modern cryptographic protocols and provide valuable insights into securing them effectively. We also discuss the existing challenges and future research directions in this domain.**

***Index Terms*—Cryptographic Network Protocols, Vulnerabilities, Encryption Algorithms, Transport Layer Security (TLS), Secure Shell (SSH), HyperText Transport Protocol Secure (HTTPS)**

I. Introduction

Cryptographic Network Protocols (CNPs) are the backbone of secure communications over networks. Since the inception of the networks and Internet, security has always been an afterthought, at least for the TCP/IP protocol stack. Therefore, over time, either cryptographic protocols have been added to the communication protocols as add-ons, or cryptographic primitives have been embedded in the existing protocols. In the TCP/IP protocol stack, most of the communication protocols that use cryptographic primitives are used at the top two layers, i.e., the Application and Transport layers. Among others, a representative CNP is the Secure Socket Layer/Transport Layer Security (SSL/TLS) that employs Public Key Infrastructure (PKI), i.e., public/private key algorithms and hash functions to guarantee end-to-end security. By establishing secure network connections and encrypting the transmitted data, SSL/TLS protects network users from a range of cyber threats such as Man-in-the-Middle (MitM), Distributed Denial of Service (DDoS), SQL injection, and other application-level attacks. Google Chrome has observed that approximately 95% of monitored traffic utilizes SSL/TLS as a security solution [1].

While CNPs have proven to be effective solutions for network security and are widely implemented; even a minor vulnerability can result in significant damage to clients and systems in terms of cybersecurity risks. Many vulnerabilities may arise from different sources such as from the use of outdated encryption algorithms [2], [3], analysis of encryption patterns [4], [5], or exploitation of weaknesses in the network protocols [6]. Exploiting these vulnerabilities provides the attackers with opportunities to infiltrate or infect systems with malware and enables them to perform other malicious activities in the network ranging from eavesdropping to active attacks.

It is worth noting that certain vulnerabilities discovered in obsolete encryption schemes remain active in modern network environments due to the coexistence of diverse CNP versions on the Internet. While many servers adopt secure cryptographic algorithms and state-of-the-art protocols by following the latest version like TLS v1.3, numerous web servers still need to support outdated encryption schemes for compatibility reasons. A report reveals that over 68% of web servers continue to support TLS v1.0 and v1.1, both of which employ outdated encryption algorithms, despite the increasing adoption of TLS v1.3 [7]. This necessitates continued attention to old protocols and algorithms, even if they are not recommended.

Supporting older versions of network protocols ensures service compatibility, but it also poses a security risk as the old version inherit their known vulnerabilities. These protocols have evolved over various iterations, with frequent updates to encryption algorithms. While network service providers maintain support for previous versions to ensure compatibility, this practice results in the coexistence of diverse versions of network protocols in modern networks. However, the encryption algorithms supported by older versions may be inadequate in the contemporary network environments. Former CNPs

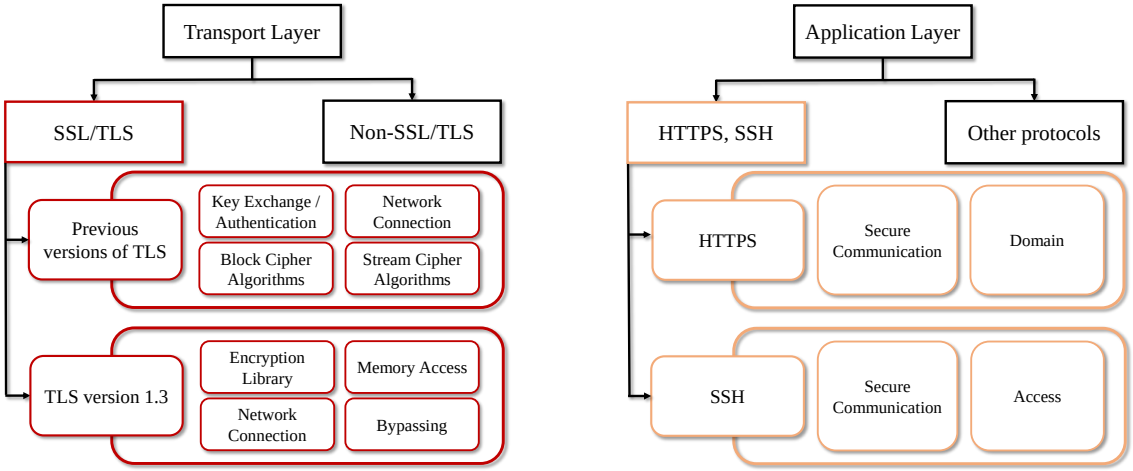
J. Ahn is with the Department of Computer Science and Engineering, Hanyang University, Seoul, Republic of Korea, e-mail: ahnjemin@hanyang.ac.kr

R. Hussain is with the Smart Internet Lab and Bristol Digital Futures Institute, University of Bristol, UK, e-mail: rasheed.hussain@bristol.ac.uk

K. Kang is with the Department of Artificial Intelligence, Hanyang University, Ansan, Republic of Korea, e-mail: ktkang@hanyang.ac.kr

J. Son is with the Department of Computer Science, University of Nevada, Las Vegas (UNLV), Las Vegas, USA, e-mail: junggab.son@unlv.edu

Manuscript first received February 27, 2024; revised October 7, 2024.



(a) Transport Layer Protocols (b) Application Layer Protocols

Fig. 1: Scope and objectives of our work. Red boxes in (a) and orange boxes in (b) represent topics covered in this paper.

were developed in different environments, and their encryption algorithms were tailored to the computing environments of that time. As computing environments advanced, certain encryption algorithms were deprecated due to their inherent weaknesses [8], [9]. Using previous versions of CNPs exposes Internet users to the vulnerabilities associated with outdated encryption algorithms. Exploiting these vulnerabilities in a mixed protocol environment is an enticing prospect for attackers [6], [10]. To this end, despite the introduction of TLS v1.3, a significant update aimed at addressing existing vulnerabilities, the classic approach of exploiting protocol mixtures to compromise secure communications persists [11].

A considerable number of surveys in the literature focus on the security of application domains such as Internet of Things [12], Intelligent Transportation System [13], Sensor Networks [14], and so on. Furthermore, despite the coexistence of diverse protocols and versions pertaining to the upper layers of the TCP/IP protocol stack, existing surveys tend to focus on narrow scopes such as encryption algorithms within specific domain or algorithm properties [15], [16]. This limited scope hinders the attainment of a comprehensive understanding of modern security channels where a multitude of protocols and algorithms are in use. Moreover, there is a notable scarcity of surveys specifically addressing vulnerabilities in TLS v1.3. While a few surveys have been conducted on encryption schemes of SSL/TLS post the release of TLS v1.3, they predominantly focus on its robustness, rather than its vulnerabilities [17], [18], [19]. It is also important to note that even seemingly secure protocol may yield unexpected results due to the intricate interactions of their complex software components. While there are no documented algorithmic vulnerabilities in TLS v1.3 so far, several implementation flaws have been identified in the literature [20], [21], [22], [23], [24]. In the modern heterogeneous and complex network environment with diverse CNPs, these flaws may be linked to existing vulnerabilities, highlighting the necessity for a comprehensive investigation of the existing efforts in this direction. To the best of our knowledge, there are few comprehensive studies that discuss CNP encryption algorithms' security in detail and provide a one-stop shop for practitioners, researchers, software developers, and security designers to make informed decisions in their specific fields. Recognizing this gap, our study explores the evolution of encryption algorithms and updates in encryption strategies within modern CNPs to enhance security. Furthermore, we delve into vulnerabilities that have necessitated security patches, providing insights into their exploitation and properties.

A. Scope and Contributions

The main objective of this paper is to conduct a comprehensive analysis of encryption schemes used in modern CNPs, focusing on data protection protocols operating in the Transport and Application layers of the TCP/IP protocol stack. Layers lower than the Transport layer (including network, link, and physical layers) employ cryptographic mechanisms including IPSec [61]. However, we do not address them for several reasons. First, security in the lower layers primarily focuses on protecting packets or signal transmission against spoofing, sniffing, or interference, whereas our emphasis is on the encryption of the actual information being transmitted. Second, the layers below the Transport layer are primarily managed by network service providers, meaning identifying their encryption and operational mechanisms can be challenging due to their security policies. Several custom encryption methods used in the Transport and Application layers often leverage lower layer protocols features [62], [63]. However, we do not include them, because encryption methods that leverage lower-layer features are typically designed for specific environments and have not been adopted as widely as encryption algorithms included in popular network security protocols. Therefore, in this paper, we focus on CNPs utilized in the Transport and Application layers, where we examine the security strategies and vulnerabilities associated with their encryption schemes. Figure 1 illustrates the scope of our

TABLE I: Related Surveys for the Past 12 Years

Category	Paper	Year	Covered Contents: Encryption Algorithms	Earlier TLS	TLS v1.3	Application Layer Protocols	Vulnerabilities	Description
Encryption Algorithm	[25]	2013	◒	○	○	○	◒	Encryption algorithms
	[26]	2014	◒	○	○	○	●	Symmetric encryption algorithms
	[27]	2014	◒	○	○	○	○	Asymmetric encryption algorithms
	[28]	2017	◒	○	○	○	●	Symmetric encryption algorithms
	[29]	2017	●	○	○	○	◒	Encryption algorithms
	[30]	2017	◒	○	○	●	◒	Cryptography algorithms in IoTs
	[31]	2017	●	◒	○	○	◒	Encryption algorithms on IoT
	[32]	2019	●	○	○	○	●	Encryption algorithms
	[33]	2019	●	○	○	○	◒	Encryption algorithms
	[34]	2020	◒	○	○	○	◒	Symmetric encryption algorithms
	[16]	2021	●	○	○	●	◒	Cryptography algorithms in IoTs
	[35]	2023	●	○	○	○	◒	A review of cryptographic algorithms
	[36]	2023	●	○	○	○	◒	Comparative analysis of encryption algorithms
	[37]	2024	●	○	○	○	◒	RSA and elliptic curve encryption systems
	[38]	2024	●	○	○	○	◒	Lightweight encryption algorithms for IoTs
	[39]	2024	●	◒	○	◒	◒	Lightweight encryption algorithms for IoTs
Secure Communication	[40]	2013	●	◒	○	○	●	SSL/TLS
	[41]	2014	◒	◒	○	○	◒	Server Forward Secrecy
	[42]	2014	◒	◒	○	◒	◒	Forward Secrecy
	[43]	2015	●	●	○	◒	◒	Secure communication for IoT
	[44]	2015	◒	◒	○	●	●	Security issues for Ad hoc
	[45]	2015	◒	●	○	●	◒	Security in software defined networks
	[46]	2016	●	◒	○	○	●	SSL/TLS
	[47]	2017	◒	○	○	◒	●	Secure communication for VANETs
	[48]	2019	◒	○	○	◒	●	Secure communication for Smart Grids
	[17]	2020	◒	◒	◒	◒	◒	HTTPS traffic
	[49]	2020	○	◒	○	●	●	Overall insight of IoT
	[50]	2020	●	○	○	●	●	Secure communication techniques for 5G
	[51]	2020	◒	○	○	●	●	Security in IoTs
	[18]	2021	◒	●	◒	●	●	Flaws summaries in TLS
	[52]	2021	◒	◒	○	●	●	Security in IoTs
	[53]	2021	●	○	●	○	○	TLS v1.3 handshake
	[54]	2022	○	◒	○	◒	○	DNS encryption
	[15]	2022	●	●	○	●	●	Security in IoTs
	[19]	2022	◒	◒	◒	◒	◒	QUIC security issues
	[55]	2023	◒	●	●	◒	◒	TLS interception mechanisms
	[56]	2023	◒	◒	○	◒	◒	Secure communications for UAV traffic
	[57]	2023	○	◒	○	◒	●	Internet measurement techniques for cyber security
	[58]	2024	◒	◒	●	○	◒	Post-Quantum TLS
	[59]	2024	◒	○	○	●	◒	Authentication protocol for IoTs
	[60]	2024	◒	◒	○	◒	◒	Secure data transmission in IoTs
Our Work			●	●	●	●	●	Encryption schemes and vulnerabilities in modern network communications

○ : Not Covered
◒ : Partially Covered
● : Covered

survey. In the Transport layer, SSL/TLS plays a crucial role in ensuring secure communication between network endpoints. Originally developed for Transmission Control Protocol (TCP) security, SSL/TLS serves a significant and pivotal role in Internet security. Furthermore, SSL/TLS has also been adopted by other protocols beyond TCP because of its convenience and reliability, and most modern communication protocols that do not include security functions themselves, such as encryption, rely on SSL/TLS for their secure communication. Consequently, understanding SSL/TLS is fundamental for assessing Transport layer security. It is also worth mentioning that TLS is the evolved version of SSL.

Earlier versions of TLS, v1.0 through v1.2, share the same handshaking process, with minor changes in the list of secure ciphers. On the other hand, TLS v1.3 introduced major changes in the handshaking process and a distinct set of cipher suites compared with previous versions. Thus, we classify TLS into two categories: TLS v1.0 through v1.2 (denoted as earlier versions) and TLS v1.3. After examining TLS, we explore vulnerabilities that exploit weaknesses in TLS encryption schemes or systemic vulnerabilities. Since several encryption schemes in the previous versions of TLS are deemed vulnerable, we explore their properties and the exploitation strategies of adversaries in detail. We divide the vulnerabilities into four different categories, depending on their exploitation target: *Key Exchange*, *Network Connection*, *Block Cipher Algorithms*, and *Stream Cipher Algorithms*. Meanwhile, TLS v1.3 does not have known insecure encryption schemes *(at the time of writing this paper)*, and thus, our focus shifts to investigating the attempts to weaken encryption algorithms and implementation bugs that may lead to unexpected results, e.g., protocol downgrading. These bugs are classified into four different categories depending on their attack strategies: *Encryption Library*, *Memory Access*, *Network Connection*, and *Bypassing*. We delve into their problems and effects, which lead to unexpected results while using TLS v1.3.

In the Application layer, numerous protocols support a wide range of services. However, it is common for many Application layer protocols to rely on SSL/TLS rather than implementing their specific security solutions. Hence, vulnerabilities presented in SSL/TLS are inherited by the Application layer protocols. Consequently, attacks specifically

targeting Application layer protocols are considered less significant compared with those focusing on SSL/TLS, which offer broader applicability. A report from Edgescan indicates that the share of attacks targeting specific Application layer protocols, aside from a few exceptions such as HyperText Transfer Protocol Secure (HTTPS) and Secure Shell (SSH), is relatively low [64]. Therefore, we focus on both protocols in the Application layer.

HTTPS employs SSL/TLS as a security protocol akin to other Application layer protocols. Given its widespread use, attacks on HTTPS are indicative of challenges faced by other protocols. This prevalence also makes it a prime target for attackers. HTTPS inherits vulnerabilities from SSL/TLS, and thus we exclude overlapping vulnerabilities caused by SSL/TLS. Therefore, our focus is on exclusive attacks exploiting HTTPS properties, categorized into two types: vulnerabilities related to insecure encryption algorithms and domain-specific vulnerabilities (denoted as Encryption Algorithms and Domain in Figure 1, respectively to avoid redundancy). In contrast, SSH stands out as a unique Application Layer protocol, providing security functions without relying on SSL/TLS. Its distinctive nature has contributed to its long-standing use with many systems supporting the protocol. This popularity, however, renders SSH a secondary target for attackers, leading to a variety of attacks on SSH [64]. While SSH manages its own encryption mechanisms and available algorithm list, the encryption algorithms mostly overlap with those used in SSL/TLS. Therefore, we focus on exploring vulnerabilities that arise from SSH's unique cryptographic mechanisms, not those from the encryption algorithms. To this end, we categorize the vulnerabilities into two groups: vulnerabilities related to SSH access and those originating from SSH communication (denoted as Access and Packet Transmission in Fig. 1, respectively).

Regarding the literature, we heavily relied on the available research databases including but not limited to IEEEXplore, ACM, Elsevier, Springer, Scopus, MDPI, ScienceDirect and Common Vulnerabilities and Exposure (CVE).

B. Existing Surveys

Our investigation traversed existing surveys conducted over the past 12 years, all of which pertain to encryption schemes and secure communication. The synthesis of our inquiry findings is succinctly presented in Table I. We define five distinct topics our investigation covers, represented in the "Covered Contents" column. These topics are classified into three tiers for comparison: The first tier, "Not Covered," is indicated by an empty circle symbol (○), signifying that a specific topic received no coverage within the survey. The second tier, "Partially Covered," is depicted as a half-filled circle (◒), denoting that the topic was touched upon but not as the primary focus. Finally, the third tier, "Covered," is represented by a filled circle symbol (●), denoting that the topic corresponds with our concentrated focus.

Existing surveys can be classified into two principal categories: *Encryption Algorithms* and *Secure Communication*. The surveys belonging to the *Encryption Algorithm* category delve into encryption algorithms used in diverse application domains, primarily emphasizing their operational details and vulnerabilities of the algorithms. As secure algorithms have been adopted in a plethora of application domains, researchers moved their focus from investigating general encryption algorithms to encryption algorithms used in specific application domains. In other words, the verticals have considerably expanded along with the technological developments, and the scope of existing works encompasses a wide range including Internet of Things (IoT) [30], [31], [16], [38], [39], comprehensive cipher algorithm analysis [25], [29], [32], [33], [35], [36] and specific algorithms such as symmetric [26], [28], [34], asymmetric [27], and authentication algorithms [37]. Nevertheless, most of the existing studies lack an examination of vulnerabilities that come from practical usage related to protocols. They focus on scrutinizing encryption algorithms and/or investigating encryption algorithms' vulnerabilities rather than vulnerabilities stemming from practical usages of encryption algorithms. Only a handful of surveys whose topics are related to networks partially cover encryption strategies of TLS and Application layer protocols [30], [31], [16], [39].

Surveys classified as *Secure Communication* examine security strategies and vulnerabilities in encrypted channels. These studies have been conducted across various application domains and verticals, offering an extensive perspective of encryption scheme implementations. They cover SSL/TLS [40], [46], [18], [65], [53], [55], its specific properties [41], [42] and various other protocols, such as QUIC [19], DNS [54], and HTTPS [17]. Furthermore, encryption and security methods have been extended to other domains with increased concerns of cybersecurity, such as IoT, Vehicular Ad hoc Networks (VANET), 5G networks, and so on. These extended security mechanisms were tailored to the unique characteristics of each domain and have been reviewed in existing surveys [43], [44], [45], [47], [48], [49], [50], [15], [52], [51], [56], [57], [58], [59], [60]. These surveys tangentially examine the encryption algorithms used to establish secure communications. Even though some of these surveys contain more topics relevant to our survey by covering the practical usage of encryption schemes, they limit the investigation of encryption strategies and protocols to a specific area of their interests. Thus, these surveys only partly cover the topics and do not explore them thoroughly.

In addition, both categories share a common trait: limited coverage of content pertaining to TLS v1.3. This lack of research addressing TLS v1.3 can be attributed to its relative novelty and a reduced number of identified vulnerabilities compared with its predecessors. Of all the conducted surveys, only a few within the "Secure Communication" category encompass content related to TLS v1.3 [18], [17], [19], [55]. Existing studies cover topics distinct from our survey, focusing on encryption schemes in specific network protocols and delving into their operations, algorithms, and vulnerabilities. For instance, Joarder and Fung examined the security of Quick UDP Internet Connections (QUIC) protocol, a TLS v1.3-compliant protocol [19]. They covered a similar scope to our interests including vulnerabilities in network protocols and

encryption schemes. However, their analysis only focuses on QUIC-specific properties, extending its scope to encompass broader protocol security matters unrelated to encryption. Similarly, Shbair et al. [17] and Levillain et al. [18] also align with our research interests; however, they offer cursory analyses of specific aspects pertinent to our focus. Particularly, they prioritize vulnerabilities in outdated TLS versions, omitting a thorough examination of the potential benefits of TLS v1.3 and its vulnerabilities. Furthermore, Carnavalet et al. discusss the differences between and advantages of TLS versions [55]. However, they emphasize on TLS mechanism rather than its encryption algorithms, and the covered vulnerabilities are also elaborated but limited to interception attacks. While these surveys may touch upon some topics of our interest, they lack an in-depth vulnerability examination stemming from within the overall encryption schemes.

C. Organization

The remainder of this paper is organized as follows: In Section II, we provide an overview of security in upper layers protocols of network including Transport and Application layer protocols. This is followed by Section III and IV that present attacks on previous versions of TLS and TLS v1.3. After exploring attacks on the Transport layer, we illustrate the attacks on Application layer protocols represented by HTTPS and SSH in Section V and VI, respectively. We discuss open research challenges in Section VII and conclude the paper with our final remarks in Section VIII.

II. Overview of Security in Upper Layer Network Protocols

This section introduces the background necessary to understand the vulnerabilities of encryption algorithms employed in the upper network layer protocols, the transport and application layers. First, we classify encryption algorithms and briefly discuss their encryption characteristics. After that, we introduce an overview of secure network protocols used in the Transport layer tracing the history of their available encryption algorithms alongwith their cryptographic features. Finally, cryptographic protocols used in application layer protocols, specifically SSH and HTTPS, are covered. The evolution of the encryption algorithms employed by both protocols is scrutinized with their cryptographic characteristics.

A. Cryptography for Network Security

Cryptography for network security can be categorized into two classes depending on the number of keys: asymmetric cryptography and symmetric cryptography. Asymmetric cryptography uses two keys, denoted as public and private keys, one for encryption and the other for decryption, while symmetric cryptography uses one key for both encryption and decryption. They have their own advantages and disadvantages and can be employed for different purposes to establish secure network environments. Furthermore, these two can be used in combination (as in TLS). As a prelude, this section briefly discusses these two classes of cryptography and their characteristics.

1) Asymmetric Cryptography

Prime number-based and elliptic curve-based algorithms are two representative examples of asymmetric cryptography and are widely used for network security. Rivest, Shamir, and Adleman (RSA), a prime number-based algorithm is one of the most used security mechanisms [68]. In the RSA algorithm, for a given message M, where $0 < M < N$, and a user's public key $PU_k = (e, N)$, a ciphertext C can be calculated as follows:

$$C = M^e mod N, \tag{1}$$

where N denotes a large integer, a multiplication of two large enough prime numbers, $N = p \times q$. The ciphertext C can be decrypted using the user's private key $PR_k = (d, N)$:

$$M = C^d mod N. \tag{2}$$

On the other hand, elliptic curve-based asymmetric algorithms employ elliptic curve graphs to generate key pairs [69]. Equation (3) illustrates the shape of the elliptic curve $E_p(a, b)$:

$$y^2 = x^3 + ax + b \quad mod \quad p. \tag{3}$$

Next, a generator G should be selected as a point on the $E_p(a, b)$, with an order of n. A user randomly selects a prime integer d smaller than n, followed by calculating a corresponding public key using the following equation:

$$PU_k = d \cdot G. \tag{4}$$

Elgamal encryption can be implemented based on the elliptic curve. To encrypt a message M, where M is a point on the $E_p(a, b)$, a user selects a random number k, where $0 < k < n$, and generates a pair of values as a ciphertext $(C = (C1, C2))$.

$$C_1 = k \cdot G, \tag{5}$$

$$C_2 = M + k \cdot PU_k. \tag{6}$$

A receiver can decrypt the ciphertext C with the private key as follows:

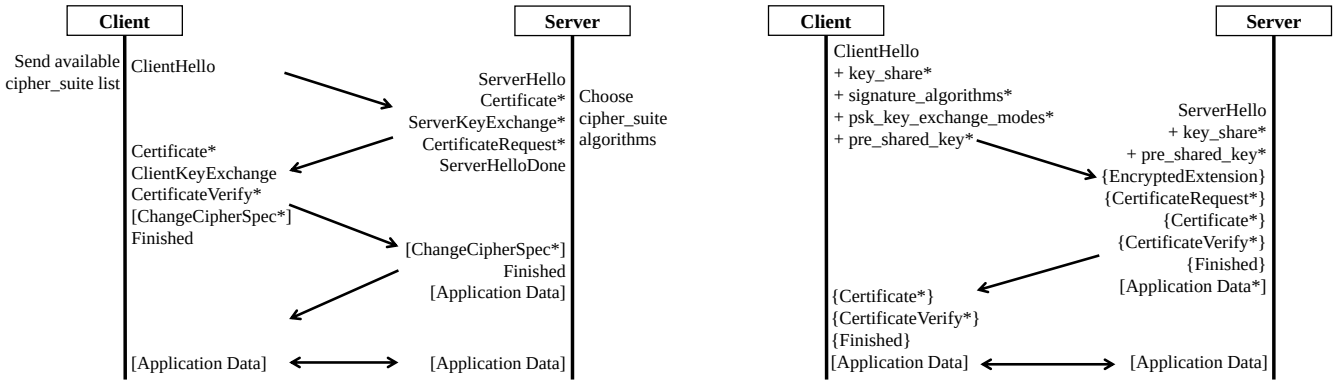
$$d \cdot C_1 = d \cdot (k \cdot G) = k \cdot (d \cdot G) = k \cdot PU_k, \tag{7}$$

$$C_2 - d \cdot C_1 = (M + k \cdot PU_k) - (k \cdot PU_k) = M. \tag{8}$$

Asymmetric cryptography serves two key functions. First, when a message is encrypted with a public key, only the owner of the corresponding private key can decrypt it, ensuring confidentiality. Second, if a message is encrypted with a user's private key, it can be verified that the user generated the ciphertext, which serves as a *digital signature*.

The Diffie-Hellman algorithm is another representative prime numbers-based asymmetric cryptography [70], which can be used only to exchange cryptographic keys. It requires two public parameters, a large enough prime number p and a primitive root g, which can generate all the non-zero residues when modulo operation is conducted on p. With public parameters, a sender and a receiver selects a random number a and b, respectively, for their private key, then both sender and receiver generate their public keys S_PU_k and R_PU_k as follows:

$$S\_PU_k = g^a \quad mod \quad p, \tag{9}$$



(a) TLS version 1.2 and earlier [66]

(b) TLS version 1.3 [67]

+ : Noteworthy Extensions
∗ : Optional Data
{ } : Messages Protected Using Key While Handshaking
[ ] : Messages Protected Using Key Derived from Application

Fig. 2: Comparison of TLS Handshaking Protocols

$$R\_PU_k = g^b \mod p. \tag{10}$$

The sender sends S_PU_k to the receiver, and the receiver sends R_PR_k to the sender. The public keys are each raised to the power of their corresponding private key and then modulo operation is conducted by p, resulting in generating shared secret key for them. The secret key has the same value due to the following property shown in the equation (11):

$$Shared\_Key = (R\_PU_k)^a \bmod p = (S\_PU_k)^b \bmod p. \tag{11}$$

Asymmetric cryptography presents lower risks of key leakage since it eliminates concerns over key distribution. This advantage is often leveraged as an additional measure for information protection, such as authentication. Consequently, asymmetric encryption algorithms are commonly employed to authenticate web server certificates, with Digital Signature Algorithm (DSA) and Diffie–Hellman-based algorithms are the most prevalent for authentication purposes. Another critical use of asymmetric cryptography is the exchange of keys used in symmetric cryptography. Due to the computational intensity of n-th power calculations required in asymmetric cryptography, decryption takes a longer time. Hence, asymmetric encryption is frequently used to exchange keys for symmetric encryption that is more appropriate for fast encryption. In the TLS encryption process, algorithms such as RSA are utilized to exchange symmetric encryption keys.

2) Symmetric Cryptography

Symmetric cryptography has two classes: block cipher and stream cipher. Block cipher encrypts plaintext of a fixed block size, while stream cipher encrypts continuously in the form of bits stream. Block cipher encrypts a plaintext by breaking it up into fixed size blocks (depending on the algorithm and its mode). Equations (12) and (13) illustrate the process.

$$Plaintext = block_1 \parallel block_2 \parallel block_3 \ldots \parallel block_n. \tag{12}$$

Every block is subjected to the encryption process (E) specified in each block cipher algorithm using a key and the blocks. The encrypted blocks are calculated once more according to the block cipher mode of operation (MO) and delivered to the receiver:

$$Ciphertext = E(block_1, k) \cdot MO \parallel E(block_2, k) \cdot MO \ldots \parallel E(block_n, k) \cdot MO. \tag{13}$$

Since the receiver knows both the MO and the k, it can decrypt the received blocks sequentially. Block ciphers employ a variety of operations, including substitutions, shifts, column shuffling, and additions, to render plaintext unintelligible. Each algorithm has its own unique set of operations. For example, Triple Data Encryption Standard (3DES) generates ciphertext by performing a single DES cipher operation three times in succession, while Advanced Encryption Standard (AES) employs a series of operations organized into rounds, requiring 10 rounds to produce a block cipher. These algorithms encrypt plaintext in blocks, which are then fragmented and reassembled into a continuous text at the receiver's end. To facilitate block-level encryption operations, block encryption methods utilize block cipher modes of operation that offer features such as error propagation prevention, parallel processing, performance enhancement, and attack mitigation. Consequently, block ciphers, in conjunction with these modes, provide more efficient, faster, and secure encryption functionalities. The primary block cipher algorithm used in TLS is AES, famous for its robustness and efficiency. As the mode used for AES, Galois/Counter Mode (GCM) and Counter with Cipher block chaining Message authentication code (CCM) are representative; both modes provide the Authenticated Encryption with Associated Data (AEAD) properties that are essential in modern TLS by generating secure authentication tags, which guarantee both data confidentiality and integrity.

Stream ciphers, on the other hand, generate ciphertext in the form of a bits stream, using an encryption key and a random number generator to create a key stream, which sequentially encrypts the plaintext into a byte stream:

$$Key\_stream = Key\_Gen(Key, IV), \tag{14}$$

where Key_Gen is a stream key generator and IV denotes an initial vector. Using the key stream, stream cipher algorithms encrypt the message M in sequence:

$$Ciphertext = Key\_Gen(Key, IV)_i \oplus M_i, \qquad (15)$$

where i denotes i-th value and $\oplus$ denotes a bit-wise or byte-wise exclusive OR (XOR) operation. When received by the receiver, the stream is decrypted in sequence by reversing the encryption process. ChaCha20 is the representative stream cipher algorithm in TLS, paired with the poly1305 Message Authentication Code (MAC) algorithm to provide AEAD property.

In TLS environments, symmetric cryptography is utilized to encrypt data transmitted over a network due to its relatively fast encryption speeds and lower computational complexity compared to asymmetric ciphers. However, symmetric cryptography faces several challenges. The primary issue is key distribution; for these algorithms to function correctly, the encryption key must be exchanged with the communicating parties, and this process is not inherently secure. A solution to this problem is the use of asymmetric encryption algorithms, which can securely exchange symmetric encryption keys. It enables secure key exchange to enable fast encrypted data transfer using the exchanged key. Another issue is the encryption function's reliability. Since the same key is used for both encryption and decryption, data collection is easier, making it more susceptible to attacks than asymmetric ciphers. As computing power increases, simpler block cipher algorithms are vulnerable to brute force attacks or downgrade attacks. Block ciphers can be compromised either through weaknesses in the algorithm itself or through specific features of the mode, such as influencing adjacent blocks or using consistent padding, which can introduce vulnerabilities. This allows attackers to decrypt ciphertext, rendering the encryption ineffective. Meanwhile, stream ciphers are vulnerable due to the importance of IVs in their encryption process; predictable generation patterns can expose stream cipher algorithms to attacks. These issues will be further examined in subsequent sections.

B. Transport Layer Security

This section provides an overview of TLS with a focus on encryption schemes utilized in the protocol. Commonly, secure communication in Transport layer protocols is achieved through the adoption of SSL/TLS or SSL/TLS-based mechanisms. Therefore, our main interest lies in the encryption schemes used in SSL/TLS.

SSL/TLS employs two types of encryption algorithms to ensure secure data transmission. Asymmetric cryptography, while slower, offers better security and is used to exchange encryption keys for symmetric encryption. Symmetric encryption, in contrast, provides faster encryption and decryption speeds, making it more efficient when handling large volumes of data to be securely transmitted. The exchanged encryption key, known as the session key, is used to encrypt and decrypt data during transmission and remains valid for the duration of the established SSL/TLS session. As it has been maintained for a long time, earlier versions of SSL/TLS, especially SSL, employ encryption algorithms that have become outdated and vulnerable to attacks. Most encryption algorithms in the list of SSL ciphers were identified as vulnerable, and SSL was banned in modern networks for this reason. Thus, we focus on the encryption schemes used in TLS, including earlier versions that are widely used in modern networks.

TABLE II: Encryption Algorithms Supported Until TLS v1.2

Type	Algorithms
Key Exchange Algorithms	– Diffie-Hellman (DH) – Elliptic-Curve Diffie-Hellman (ECDH) – Rivest Shamir Adleman (RSA) – Secure Remote Password (SRP)
Certificate Authentication Algorithms	– Some Digital Signature Standards (DSS) combined with non-ephemeral algorithms – ANONymous signature (ANON) – Kerberos
Encryption Algorithms	– Data Encryption Standard (DES) / 3DES – Rivest Cipher 2/4 (RC2/4) – International Data Encryption Algorithm (IDEA) – Camellia – SEED – ARIA
Mode of Operation	– Cipher Block Chaining (CBC)

TLS has undergone constant changes with each version, primarily involving updates to the available encryption algorithms. However, a significant change was made with version 1.3 to address a fundamental problem that arose from the operational aspect of the earlier TLS versions such as in the process of algorithm negotiation. Figure 2 illustrates the change in the handshaking process. In Figure 2a, the handshake protocol in earlier versions uses two Round Trip Time (RTT) for algorithm negotiation, allowing clients to establish secure connections with various counterparts. This negotiation process involves aligning a "cipher_suite" for algorithm matching that encompassed key exchange, Certificate Authentication (CA), encryption, and MAC algorithms. Once the "cipher_suite" is negotiated, the client generates a pre-master secret, which is used in combination with other information from the SSL/TLS handshake to derive the master secret. From this master secret and the agreed upon encryption settings, the symmetric encryption key (session key) is generated, which remains valid for the duration of the session. The session key is included in the "ClientKeyExchange" message and sent to the server. At this stage, the "ChangeCipherSpec" message is also sent, indicating that encryption is ready. Once this bit is activated, both parties begin transmitting data using the agreed-upon encryption algorithms. However, this negotiation strategy is vulnerable because malicious servers can force ordinary connections to change their versions or algorithms to which they can easily attack.

To resolve this problem, TLS v1.3 adopted several strategies distinct from the earlier versions, as shown in Figure 2b. The client sends public key exchange information in the "key_share" field, which includes available curve specifications and finite field parameters. Additionally, supported algorithms for digital signature exchange are commu-

TABLE III: Encryption Algorithms in TLS version 1.3

Type	Algorithms
Key Exchange Algorithms	– Diffie-Hellman Ephemeral (DHE) – Elliptic-Curve Diffie-Hellman Ephemeral (ECDHE)
Certificate Authentication Algorithms	– Rivest Shamir Adleman (RSA) – Elliptic-Curve digital signature algorithm (ECDSA) – Edwards-curve Digital Signature Algorithm (EdDSA)
Encryption Algorithms	– Advanced Encryption Standard (AES) – ChaCha20-Poly1305
Mode of Operation	– Galois/Counter Mode (GCM) – Counter with Cipher Block Chaining-Message Authentication Code (CCM)

nicated through the "signature_algorithms" field. If a Pre-Shared Key (PSK) has already been established, the client includes two additional fields: "psk_key_exchange_modes" and "pre_shared_key." The PSK is an encryption key generated during a previously established session, and if the session is still valid, the key can continue to be used. The "psk_key_exchange_modes" field specifies information such as the algorithm used and key length, while the "pre_shared_key" field contains the actual public PSK that has been exchanged. Based on the combination of fields received, the server responds with the "key_share" field if a new key exchange is required or with the "pre_shared_key" if the PSK is to be reused. In both cases, these fields serve the same function as those sent by the client.

One notable change is that TLS v1.3 no longer supports insecure algorithms and the TLS version negotiation. This change prevents TLS v1.3-based secure connections from being compromised by replacing trustworthy algorithms with vulnerable ones. However, to achieve this goal, the available algorithms must be trustworthy. This led to the next characteristic of TLS v1.3, where the available algorithms need to satisfy specific properties: Forward Secrecy and Authenticated Encryption with Associated Data (AEAD). They are related to key exchange and data encryption algorithms, respectively. Forward Secrecy guarantees that leakage of a single session key does not affect different sessions by using ephemeral keys during the key exchange [71]. In other words, Forward Secrecy protects the past sessions if the keys are compromised in the future, and in case of a key compromise, the damage is limited to the session that used the compromised session key. As a data encryption-related property, AEAD provides reinforced data encryption authentication. Transmitted data consists of ciphertexts and MAC that allows receivers to inspect the received data's correctness. Encryption algorithms satisfying the AEAD property utilize plaintexts and ciphertexts to generate associated data, which are used to calculate MAC. This additional authentication process makes it difficult to analyze and manipulate the data without an encryption key.

TLS v1.3 only requires a single RTT to establish a secure network channel and has a simplified "cipher_suite" that only has encryption and MAC algorithms to be used for secure communications. To achieve this, TLS v1.3 simplifies the list of available algorithms, resulting in making the use of the key exchange algorithm and digital certificate authentication algorithm predictable. As a result, only a few algorithms remain in each field in TLS v1.3; algorithms not commonly used or failed to meet the required properties were banned. Tables II and III enumerate the updates to the encryption algorithms used in TLS. Non-ephemeral algorithms have been removed from the list of available algorithms for key exchange and certificate authentication, and non-AEAD algorithms have been discarded from the list of algorithms for data encryption. Although algorithms like Camellia and ARIA are considered trustworthy, they have also been removed due to their infrequent uses.

On the other hand, this simplification also makes it challenging for new algorithms to be officially added in TLS v1.3. While a few algorithms have been suggested as drafts, they are still awaiting official acceptance [72], [73], [74], [75], [76], and currently, they can only be used through private extensions.

C. Application Layer Security

SSL/TLS stands as a de facto standard to establish secure network channels, protecting the Transport layer and above of the TCP/IP protocol stack. Within this landscape, an array of Application layer protocols serves diverse application services, many of which did not originally encompass encrypted communication. Nevertheless, the escalating concern over security vulnerabilities in network services has fueled a surge in secure communication practices. This has prompted most Application layer protocols to opt for integration with the SSL/TLS protocol, enabling a shielded communication framework.

The strategic union with SSL/TLS yields several encapsulation benefits. Notably, this approach offers significant convenience for application layer protocol developers, as a single security layer assumes the responsibility of safeguarding multiple protocols. This reduces the scope that the protocol designers must consider, enabling them to emphasize functionality without becoming embroiled in intricate security specifications. As an encapsulated layer, SSL/TLS specifications (RFC 5246 and 7301) show benefits when Application layer protocols adopt it. However, this symbiotic reliance on SSL/TLS encryption schemes exposes Application layer protocols to the same vulnerabilities that SSL/TLS has. It makes attackers concentrate on discovering vulnerabilities of SSL/TLS, not a specific Application layer protocol. Supporting this perspective, the report issued by Edgescan demonstrates that SSL/TLS is a main target of modern network attacks [64]. However, this report also indicates that SSH and HTTPS are also major targets, implying that investigating attacks and vulnerabilities is essential.

Our focus narrows down to these heavily targeted Application layer protocols due to their substantial prevalence. While these protocols share a plethora of vulnerabilities with SSL/TLS, we emphasize the vulnerabilities intrinsic to the characteristics and encryption paradigms of Application layer protocols, which is an attack point that the attackers predominantly exploit. Within this framework, we spotlight the two most frequently attacked protocols. Foremost is the HTTP based protocol, wherein HTTPS stands as the secure HTTP utilizing SSL/TLS. This widely used and targeted protocol

can represent other SSL/TLS-dependent application layer protocols, as they adopt the same channel encryption strategy as HTTPS and are vulnerable to similar attacks. Therefore, we delve into the intricacies of HTTPS, examining its processes and encryption-related vulnerabilities.

SSH, however, distinguishes itself by diverging from the typical SSL/TLS-reliant protocol model. Developed before SSL/TLS, SSH serves as a distinctive protocol designed to ensure secure communication between end points. Its unique attributes have made the protocol essential in numerous systems, resulted in becoming the the second most targeted Application layer protocol. Unlike its SSL/TLS counterparts, SSH operates with an independent encryption process, characterized by similar yet subtle distinctions from SSL/TLS. We examine the encryption algorithms used by SSH and their vulnerabilities that exploit the characteristics of the algorithms.

1) HyperText Transfer Protocol Secure (HTTPS)

HTTP serves as a foundation for the exchange of hypertext-based data across networks. In its early stage, HTTP was tailored for the transmission of text-based data, including formats like HyperText Markup Language (HTML) and Cascading Style Sheets (CSS). However, as web services grew more complex and heterogeneous, the need to accommodate diverse data types became more important. This evolution prompted over 15 years of development, wherein HTTP underwent continuous enhancements through the addition of extensions. This adaptability made HTTP popular so far, having been a basis for various modern web protocols.

Along with this long enhancement, a huge number of attacks against HTTP have been discovered. For the baseline HTTP, the attack tactics were relatively straightforward as transmitted data without encryption can be easily eavesdropped by passive attackers. The advent of HTTPS was a direct response to these vulnerabilities [77], resulting in its wide scale adoption. Serving as a fusion of HTTP and SSL/TLS, HTTPS usage has witnessed consistent growth. The report issued by Web Almanac highlights that the share of HTTPS exceeds 85% across both mobile and desktop environments in 2022 [78]. In the HTTPS paradigm, HTTP data is transferred to the Transport layer for delivery. The data are split by the proper Transport layer protocol rule and then encrypted by the encryption strategy of SSL/TLS. Other Application layer protocols that entrust their security to SSL/TLS also follow this process.

2) Secure Shell Protocol

The initial version of SSH was proposed in 1996 as a secure alternative to insecure network communication protocols like Telnet and Remote Shell [79], but it suffered from several vulnerabilities [80], [81]. To address these concerns, an improved version of SSH was introduced in 2006, denoted as SSH v2, serving as the primary version up to this day. Although subtle variations exist between applications, they primarily follow the SSH v2 specification [82]. Compared with its predecessor, SSH v2 incorporates more robust encryption algorithms and offers a wider array of functionalities, including file transfers. Over its extensive history, the SSH protocol has undergone multiple updates, incorporating not only resilient algorithms but also supplemental operations such as FTP-related extensions.

TABLE IV: Encryption Algorithms in SSH 2.0

Type	Algorithms
Key Exchange Algorithms	– Diffie-Hellman (DH) – *Elliptic-Curve Diffie-Hellman (ECDH) – *Elliptic Curve Menezes-Qu-Vanstone (ECMQV)
Certificate Authentication Algorithms	– Rivest Shamir Adleman (RSA) – Digital Signature Standard (DSS) – *Elliptic-Curve digital signature algorithm (ECDSA) – °Edwards-Curve Digital Signature Algorithm (EdDSA)
Encryption Algorithms	– Advanced Encryption Standard (AES) – Blowfish – Twofish – Triple Data Encryption Standard (3DES) – Serpent – ARCFOUR (RC4) – IDEA – Carlisle Adams and Stafford Tavares-128 (CAST-128)
Mode of Operation	– Cipher-Block Chaining (CBC) – †CounTeR (CTR) – ‡Galois Counter Mode (GCM)

∗ : Added in RFC 5656
∘ : Added in RFC 8709
† : Added in RFC 4344
‡ : Added in RFC 5647

Unlike other Application layer protocols, SSH stands as a unique protocol that facilitates encrypted connections without relying on SSL/TLS. Its operational principles resemble those of SSL/TLS. Like SSL/TLS, SSH generates a symmetric key and disseminates it using an asymmetric encryption algorithm. Moreover, it features authentication and integrity functions akin to those present in SSL/TLS. However, SSH distinguishes itself by executing its security functions on the Application layer in contrast to SSL/TLS-dependable protocols, and supports additional access control options. We concentrate on the features and vulnerabilities. These vulnerabilities typically stem from the use of insecure encryption algorithms and flaws in the flow of processes.

Table IV enumerates the encryption algorithms available within SSH with many also finding utility in SSL/TLS. Algorithms without any symbols in the table are initial algorithms incorporated in SSH v2 [83]. Subsequent updates introduced additional algorithms through various RFCs including 5656, 8709, 4344, and 5647. These updates aimed not only at adopting secure algorithms but also at disabling encryption algorithms considered as vulnerable with certain algorithms being officially recommended in RFC documents [84], [85].

Given the substantial overlap between encryption algorithms used in SSH and those in TLS, we refrain from going into their details. However, SSH has several encryption algorithms not included in SSL/TLS, such as CAST-128, Blowfish, Twofish, and Serpent. These, though distinct, are not frequently utilized in modern networks for various reasons. CAST-128 and blowfish, due to their utilization of 64-bit cipher blocks, are susceptible to birthday attacks and are considered unsafe [10]. Twofish and Serpent were finalist algorithms in the AES competition held from 1997 to 2000. In the competition, however, Rijndael emerged as the victorious algorithm and subsequently became the standard AES encryption method, constraining the influence of other contenders. Moreover, side-channel attack vulnerabilities were identified in the Twofish encryption algorithm owing to its key processing mecha-

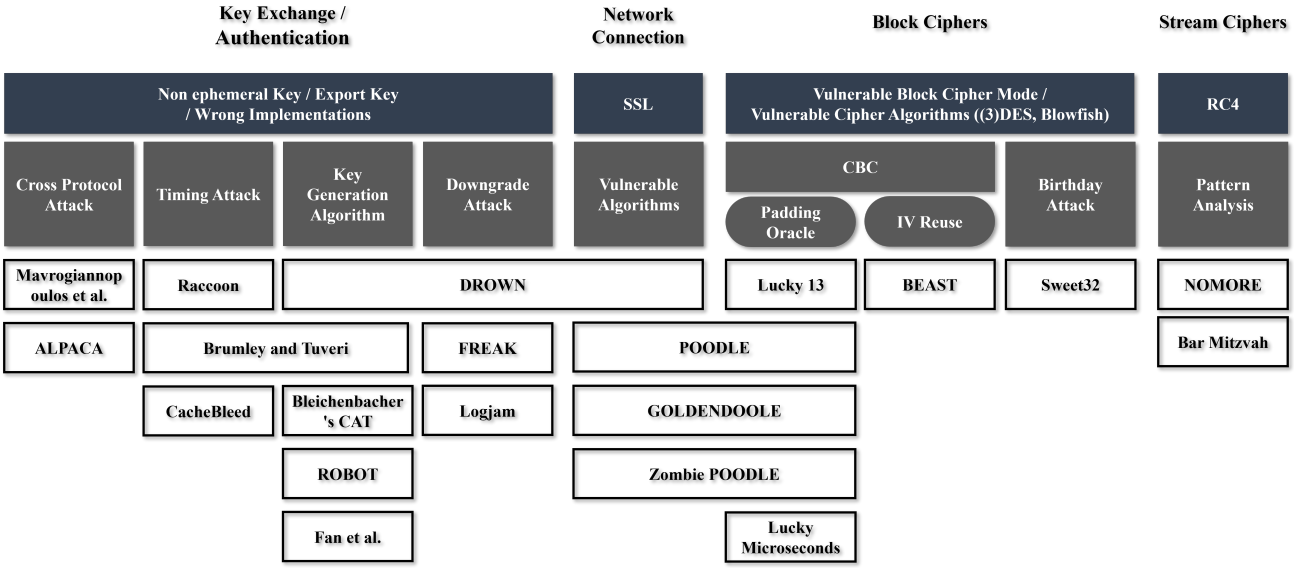


Fig. 3: Classification of vulnerabilities in the earlier versions of TLS. Targets are represented by navy boxes, strategies by gray boxes, and the names of publicly known attacks by white boxes. Wide boxes indicate attacks with multiple properties. These attacks are classified into four main cases: Key Exchange/Authentication, Network Connection, Block Cipher Algorithm, and Stream Cipher Algorithm.

nism [86]. While numerous efforts were made to address these vulnerabilities over time, Twofish and Serpent became obsolete after Rijndael was selected for the AES standard. Except for these vulnerable or infrequently used algorithms, SSH supports similar encryption algorithms as SSL/TLS does. Since vulnerabilities of common algorithms are covered in Sections III and IV, we concentrated on investigating other weaknesses linked to SSH's properties. Section VI introduces unveiled attack strategies aimed at exploiting SSH's encryption mechanisms.

III. Attacks on Earlier Versions of TLS

Attacks on earlier versions of TLS exploit various vulnerabilities, encompassing a range of strategies. For example, some attacks focus on disabling target systems by establishing numerous connections, and other attacks try to establish illegal connections by exploring commonly used ports [87], [64]. SHAttered and SLOTH exploit vulnerabilities in the MAC algorithms used in early versions of TLS [88], [89]. Heartbleed (CVE-2014-0160) exploits a bug related to incorrect memory bound checking, leading to potential data leaks when the request size differs from the actual size [90]. Furthermore, there is a category of attacks that leverages weaknesses in the TLS connection process. The triple handshake attack, also known as a MitM attack, is a notable representative, placing attackers in the middle of a secure connection [91].

Beyond these attacks, other vulnerabilities are exploited within or related to encryption algorithms utilized for establishing TLS channels. This constitutes our primary focus. Starting from the list of encryption algorithms in Table II, we comprehensively investigate various attack strategies that align with our research goals and systematically categorize them into four distinct classes: Key Exchange/Authentication, Network Connection, Block Cipher, and Stream Cipher. Figure 3 illustrates details of these classes.

A. Key Exchange Algorithms

The first category we discuss, involves vulnerabilities that exploit key exchange algorithms. These vulnerabilities primarily downgrade a key exchange algorithm to a vulnerable version, thereby granting attackers to obtain information or decipher the entire encrypted traffic.

1) Cross Protocol Attack

In the context of modern TLS attacks, algorithms that have been excluded from TLS v1.3 are noteworthy because they were banned for security apprehensions. Certain attacks exploit vulnerabilities in cross-protocol communication to execute malicious authentication, thereby leading to MitM attacks. A remarkable study by Mavrogiannopoulos et al. shows scenarios wherein a MitM attack can succeed by identifying an encryption key shared between a server and a client, which remains valid for more than two encryption algorithms [92]. This sophisticated attack builds upon the approach introduced by Wagner and Schneier, originally targeting SSL, and forces normal Elliptic Curve Diffie-Hellman (ECDH) based connections to use weaker algorithms [93]. The strategy causes the server and client to communicate with different algorithms, allowing the attacker to exploit the server as an oracle to recover the client's pre-master secret. By accessing this oracle, the attacker can obtain the key capable of decrypting every session key generated from the DH algorithm, since normal DH employs a static pre-master secret. The Application Layer Protocol Confusion Analyzing and Mitigating Cracks in TLS Authentication (ALPACA) attack also shows a MitM scenario [94]. This attack exploits immature identification in the certification process which only verifies domains excluding the

real content. By manipulating request messages, the attacker confuses clients, creating an opportunity for the MitM attack to succeed. The ALPACA attack provides empirical evidence of the vulnerable cases. The authors of the ALPACA investigated 1.4M distinct websites and found 114,197 websites that were susceptible to downgrading attacks, which means that around 8% of websites were vulnerable to this attack. We argue that even 8% is a significant number.

2) Timing Attack

Eavesdropping proves to be an efficient strategy for certain attacks, particularly targeting non-ephemeral key exchange algorithms vulnerable to timing or pattern analysis attacks. A noteworthy example of such an attack is Raccoon, which aims at exploiting these vulnerabilities in the context of DH algorithms used in TLS [95]. This attack is applicable to both ephemeral and static DH algorithms; however, the extractable information from secure connections using ephemeral DH algorithms is relatively insignificant. Raccoon attacks primarily focus on secure connections utilizing static DH algorithms for key exchange. The method of the attack requires capturing packets during the initial step of a TLS connection. These collected packets play a pivotal role in enabling the attacker to measure the timing, which serves as an oracle originating from the TLS server. Through this timing-related oracle, the attacker can infer the pre-master secret, which will subsequently be converted into a shared key. They found 3.33% of Alexa top 100k websites meet the conditions that the Raccoon attack needs to succeed. However, due to the complexity and many prerequisites, Raccoon is considered a difficult attack to perform. Nevertheless, it plays a crucial role in warning about the risks associated with outdated versions of TLS. As another example of a timing attack, Brumley and Tuveri discovered that elliptic curve-based encryption algorithms using Montgomery's ladder for scalar multiplication in OpenSSL are susceptible to a timing attack [96]. To prove this vulnerability, they represented an encryption key recovering scenario in a TLS handshaking process using the ECDSA key exchange algorithm. As a distinct scenario applicable to ephemeral key exchange cases, it could be considered a fatal attack strategy to incapacitate encryption algorithms. Most attacks on encryption schemes target software-level vulnerabilities, CacheBleed, a particular timing attack, turned out an exploitable point in an architectural domain by verifying a vulnerability stemmed from micro architecture [97]. This strategy is available when specific versions of Intel architecture access their memory, enabling attackers to reveal an RSA key by adopting an old cache collision attack [98]. Through experiments, they achieved a high recovery rate of 60% by observing 16,000 times of RSA decryption process. Both timing attacks are significant examples, because they work on hardware-related domains that had been considered vulnerability-free areas. These instances remind us to keep watching every component that composes the whole encryption steps.

3) Key Generation Algorithm

Meanwhile, encryption algorithms used in the Key Exchange and the certificate authentication steps underwent significant changes due to several security issues. The RSA algorithm serves as a representative target for attackers. A prevalent technique used in attacking RSA involves an old chosen cipher attack known as the Bleichenbacher vulnerability [99]. Through this attack, a message encrypted with 512-bit or 1024-bit keys between 300 thousand and 2 million trials can be decrypted. Noteworthy variants of this attack are the Return Of Bleichenbacher's Oracle Threat (ROBOT) and Bleichenbacher's new Cache ATtack (CAT), which were discovered decades later from the release of the original attack [100], [101]. The attack methodologies employed herein rely on the use of an RSA PKCS#1 v1.5 padding oracle, affording attackers the capability to retrieve plaintext from RSA-encrypted ciphertexts. These instances demonstrate that the outdated vulnerability can remain exploitable even within contemporary network environments. By exploring the top 1M websites on Alexa, more than 20,000 websites that could be vulnerable to a ROBOT attack were discovered, and the CAT showed that the attack can also affect TLS 1.3, albeit with a lower success rate for CAT attacks. Vulnerabilities in the Key Exchange algorithms have been discovered toward ECDSA as well as RSA [102], [96]. Fan et al. showed OpenSSL has a vulnerability due to insecure implementation in the ECDSA key generation step [102]. This vulnerability stemmed from a specific multiplication method and enables attackers to obtain information around half bit of a used key by a side channel attack. With the discovered key information, they proved that it is possible to recover an entire encryption key by using diverse arithmetic operations.

4) Downgrade Attack

Shifting a key exchange algorithm, rather than downgrading the connection protocol, represents another valid strategy against TLS. Remarkable examples of such attacks include Factoring RSA Export Keys (FREAK) and Logjam, both of which exploit vulnerabilities in the OpenSSL libraries [2], [3]. These attacks exploit the use of special cipher algorithms intended for export in the 1990s. Although they were forgotten, these algorithms continued to exist within the OpenSSL libraries until the 2010s. Their security flaw came from the usage of excessively short key sizes, rendering them susceptible to brute force attacks empowered by enhanced computing capabilities. In the FREAK attack, targets were coerced into using the RSA_export key [2], while Logjam manipulated the target into employing the DHE_export key [3]. Both attacks were identified at the time of publication with approximately 20% of popular websites exposed to the vulnerabilities. Despite the availability of stronger 1024-bit or 2048-bit keys, the attacks forced the use of the vulnerable 512-bit keys. The aftermath of these attacks led to the removal of export cipher algorithms from the OpenSSL libraries. Additionally, such incidents prompted the simplification of algorithms used in TLS v1.3, along with the adoption of a non-negotiable connection strategy. Furthermore, vulnerabilities associated with non-ephemeral algorithms in key exchange protocols resulted in removing them from the available list [8], [103]. This step was taken to enhance the overall security of TLS implementations.

5) Other Minor Attacks

Secure Remote Password (SRP) protocol was also supported in the key exchange process of earlier TLS [104]. As a

password-based algorithm, it has suffered from dictionary attacks and side channel attacks, which infer password related information [104], [105], [106]. In TLS connection, however, this protocol is not practical due to its low adoption rate. The password-based strategy requires a manual password setup, which is more suitable to manage private servers, not public services. It leads to a low acceptance rate of SRP in TLS connections, and this low utilization resulted in eliminating SRP in TLS v1.3. Since SSH protocol is a primary user of SRP, we discuss it in detail in Section VI.

B. Network Connection Protocols

Downgrading attacks target not only robust algorithms but also network connection protocols. These attacks convert TLS network connections to SSL, which incorporates many weak encryption algorithms. This gives attackers an opportunity to select a weak encryption algorithm for each field of "cipher_suite". In practical instances, observed attacks have demonstrated the shift of key exchange or block ciphers to vulnerable options.

1) Vulnerable Algorithms with CBC Mode (Padding Oracle)

One prominent attack leveraging the downgrade approach is Padding Oracle on Downgraded Legacy Encryption (POODLE) [107]. Previous versions of TLS support a negotiation mechanism to match algorithms supported by counterparts. When attackers conduct MitM attacks between servers and clients, they can establish SSL v3.0 connections with their targets if SSL v3.0 connection is supported for compatibility reasons. Once an SSL v3.0 connection is successfully established, the attacker can initiate a padding oracle attack. This attack exploits a padding process utilized in a block cipher mode of operation and exploits SSL v3.0's improper handling of padding responses. Block cipher algorithms employ padding schemes to fill the remaining space within a block when the data size does not align with the block size. Since attackers are placed in the middle, they can collect transmitted ciphertexts, which have the padding patterns. By analyzing these data, the attackers can deduce whether altered bits are for the padding or not. They can further determine exact bits by repeatedly making requests if the bits are not for the padding. This approach allows attackers to reveal one byte of encrypted messages only within 256 SSL v3.0 requests.

Although most systems disabled the use of SSL v3.0 after the introduction of the POODLE attack, new vulnerabilities related to POODLE have emerged [108]. New POODLE variants exploit the case of a valid padding with an invalid MAC, indicating that the Cipher Block Chaining (CBC) mode is still susceptible to a POODLE attack. Two variants, Zombie POODLE and GOLDENDOODLE, were discovered in a Citrix application delivery controller [109], [11]. Zombie POODLE was identified in the Citrix load balancer. This attack is available when the target was still using outdated operations that were recommended to be avoided due to the initial POODLE attack. It achieved a high success rate by exploiting information about invalid padding with a valid MAC. GOLDENDOODLE exhibited a similar exploit scenario and impact as Zombie POODLE did, but with significantly fewer requests to achieve success. A report indicates that over 2,500 out of 1 million Amazon Alexa top domains had vulnerabilities to Zombie POODLE attacks, and approximately 1,000 of them were susceptible to GOLDENDOODLE attacks [110]. Fortunately, security patches were implemented by network services vendors to mitigate these threats.

2) Vulnerable Algorithms with Downgrade Attack

Another attack that downgrades TLS to SSL is Decrypting RSA with Obsolete and Weakened eNcryption (DROWN) [6]. This cross-protocol attack enables attackers to decipher secure channels by exploiting multiple vulnerabilities of an unsafe connection protocol. DROWN attacks could be successful due to certain vulnerabilities in SSL v2.0. It exploits the TLS weakness using the same RSA private key for SSL v2.0 connections because it employs ephemeral keys for the RSA key exchange step [111], [112]. The attack exposed around 11 million servers to potential risks. Subsequently, most servers disabled the use of SSL v2.0 after the discovery of the DROWN attack, to block DROWN and its variants. The existence of such attacks underscores the significance of addressing old vulnerabilities and discontinuing support for outdated secure network connection protocols.

C. Block Cipher Algorithms

Among the block cipher algorithms, DES-based algorithms were banned earlier due to their inadequate key lengths. The increase in computing power facilitated breaking their encryption through straightforward brute force methods, rendering them insecure. In addition to these delisted algorithms, certain attacks focus on exploiting other block cipher characteristics used in TLS.

1) Birthday Attack

One such attack is Sweet32, which targets 64-bit block size encryption algorithms, including standard DES and Blowfish [10]. Notably, 3DES was the active TLS encryption algorithm at the time of Sweet32's publication. 3DES had a usage rate of just 1%, but nearly 90% of servers supported the encryption algorithm when this attack was discovered. This birthday attack on 64-bit block ciphers requires a large dataset and necessitates the attacker to monitor the target's secure connection for an extended period. Furthermore, several prerequisites must be met for a successful attack, including access to some plaintext and ciphertext and the issuance of a duplicated key. Although Sweet32 was not widely applicable, it brought attention to the vulnerability of short-length block ciphers, leading to the decision to delist 3DES, the second most widely used algorithm.

2) CBC Mode (IV Reuse)

Block ciphers generally require a mode operation to address data size (bigger files) and ensure data integrity. Among several modes, vulnerabilities and attacks against CBC mode were discovered. For example, the Browser Exploit Against SSL/TLS (BEAST) attack, introduced in 2011 (CVE-2011-3389), exploits a vulnerability in the old CBC mode system during the IV creation process [113]. Since some older systems used a fixed IV for CBC mode, allowing attackers to exploit this weakness and reuse the IV to break the CBC mode. This

led to the encryption of all blocks in CBC mode being impacted by the initial block created with the IV, facilitating the breaking of the algorithms by narrowing down the information required for attackers to guess.

3) CBC Mode (Padding Oracle)

Another attack on CBC mode is known as the Lucky13 attack, which utilizes a padding oracle from header information of the TLS MAC [5]. Employing a timing attack strategy with side-channel sniffing, Lucky13 exploits time differences between TLS messages with varying bytes of padding to infer the data of transmitted packets. Variants of Lucky13 have been studied by generating attack scenarios focusing on different environments or targets. Lucky Microseconds attack targets servers utilizing s2n-tls implementation, an open-source TLS developed by Amazon [114]. Another Lucky13 variant, named Lucky Strike [115], was proposed to attack cloud service environments. Vigorous activities to attack block cipher algorithms have led service providers to opt for stream cipher algorithms, known for fast and secure encryption when establishing secure connections [116].

The vulnerabilities highlighted previously have led to the obsolescence and deprecation of widely-used algorithms in TLS v1.3. In addition to these, certain block ciphers have faced deprecation for distinct reasons. Some of these ciphers are banned due to low utilization rates; they deemed secure algorithms when TLS v1.3 was developing, even though their vulnerabilities were discovered recently [117], [118]. Fortunately, these vulnerabilities have not resulted in significant damage as the attacks operate within constrained scenarios and the algorithms are seldom employed. Nevertheless, it is imperative to acknowledge that their potential impact could amplify with changing environmental conditions in the future, akin to the obsolete algorithms, necessitating ongoing monitoring and scrutiny.

D. Stream Cipher Algorithms

Stream cipher algorithms were often selected to avoid vulnerabilities in block cipher algorithms and to expedite the encryption process. Notably, Rivest Cipher 4 (RC4) was a representative stream cipher algorithm utilized in the earlier versions of TLS. Stream ciphers operate at the bit level, encrypting data by bit-level eXclusive OR (XOR) operations with a key stream. However, potential vulnerabilities in a key stream or a key stream generator were also introduced.

One significant attack against RC4 is the Numerous Occurrence MOnitoring & Recovery Exploit (NOMORE), which demonstrates a vulnerability in RC4 in practice [4]. The attack involves injecting a script to create connection requests and collecting a large volume of request data over 50 hours. Attackers can analyze a bias presented in the random number stream by utilizing collected data. While NOMORE requires considerable data accumulation time, attackers may still find the weakness within a few hours with a 94% success rate. Another bias-based attack on RC4 is the Bar Mitzvah attack [123], which originated from research papers introduced in 2001 [124], [125]. The attack leverages the poor initialization of RC4, resulting in a pattern in the creation

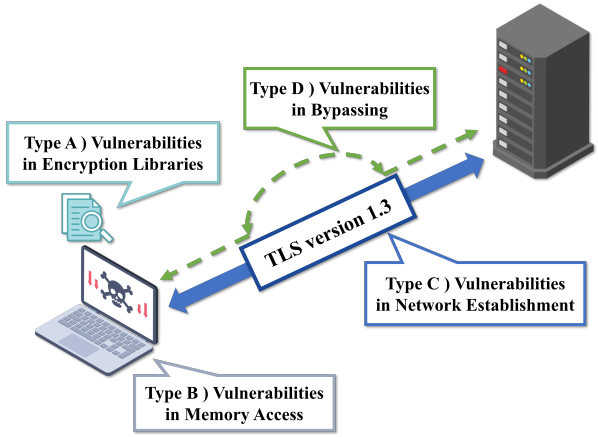


Fig. 4: Types of Attacks on TLS Version 1.3

of the key stream. By employing sniffing methods, attackers can calculate the bias in the key stream. These pattern-based attacks on RC4 had insisted that the algorithm should be deprecated [126], leading to a significant decline in using RC4 encryption algorithms compared with the its usage before the publication of these attacks [127]. Consequently, ChaCha20 was chosen as the default stream cipher for TLS v1.3.

To date, no stream bias-related weaknesses have been discovered in ChaCha20. However, a library vulnerability in ChaCha20 was reported when an ephemeral nonce was reused (CVE-2019-1543). Since the usage of a non-ephemeral algorithm conflicts with the policy of TLS v1.3, libraries related to this vulnerability were swiftly corrected. It was confirmed as a bug, and no stream bias-related vulnerabilities in ChaCha20 have been reported so far. Thus, ChaCha20 continues to be considered a robust algorithm, and its maintenance is expected to continue for the foreseeable future.

IV. Attacks on TLS v1.3

The attacks discussed in the preceding section are discovered strategies that exploit vulnerable encryption schemes utilized in earlier versions of TLS. TLS v1.3 addressed these concerns by simplifying its available algorithm list and eliminating their negotiation process during the connection setup stage. These inherent attributes thwart any attempts at network channel downgrading and firmly prohibit the use of insecure encryption algorithms. Notably, the adoption of TLS v1.3 has surged to nearly 50%, bolstered by its robust security measures. This adoption rate is anticipated to continue its upward tendency, according to the F5 report [128]. However, even the seemingly flawless protocol can encounter challenges. In addition, despite the release of TLS v1.3, numerous servers must support the earlier versions of TLS due to compatibility issues. Given the SSL exploitation examples, supporting old versions of TLS may turn into new vulnerabilities.

This section turns its focus to the vulnerabilities of TLS v1.3 since its release. While no algorithmic vulnerabilities have been detected in TLS v1.3 so far, our investigation focuses on the reported vulnerabilities. They are primarily classified as software bugs and conducting abnormal behaviors

TABLE V: Discovered Vulnerabilities on TLS v1.3

Title	Environment	Vulnerable Versions	Authority	Descriptions	Type
Jager et al. [20]	OpenSSL	1.0.2	-	Weaknesses in PKCS#1 v1.5 Encryption	C
CVE-2019-11727	firefox	68 before	Mozila	Force to use PKCS#1 v1.5 only for authentication	C
CVE-2019-17023	firefox	72 before	Mozila	Enable to downgrade version of TLS	C
CVE-2020-13777	gunTLS	3.6.x $\sim$ 3.6.14	MITRE	Incorrect cryptography for encrypting a session ticket	A, D
CVE-2020-24613	wolfSSL	4.5 before	MITRE	Mishandling TLS server data allow attackers conceal	A
CVE-2020-3285	Cisco FTD	6.4.0 $\sim$ 6.4.0.8	Cisco	Bypassing a TLS 1.3 to access blocked URLs	D
CVE-2020-8660	CNCF envoy	1.13.0 before	MITRE	Bypassing a TLS inspector in CNCF Envoy environments	D
Dunkelman et al. [119]	AES	-	-	An advanced boomerang attack on AES	A
Lee et al. [21]	OpenSSL GnuTLS BoringSSL	1.1.1 3.6.4 master	-	Downgrading attacks on TLS 1.3	C
Akhmetzyanova et al. [22]	OpenSSL	-	-	Vulnerabilities on psk_ke mode	C
CVE-2021-22890	curl	7.63.0 $\sim$ 7.75.0	HackerOne	Bad handling of TLS 1.3 session tickets	C, D
CVE-2021-22901	curl	7.75.0 $\sim$ 7.76.1	HackerOne	UAF vulnerability after get a session ticket	B
CVE-2021-3336	wolfSSL	4.7.0 before	MITRE	Not stop a certain anomalous peer behavior	A, C
CVE-2021-4160	OpenSSL	1.0.2 $\sim$ 1.0.2zb 1.1.1 $\sim$ 1.1.1l 3.0.0 before 3.0.1	OpenSSL	A carry propagation bug in MIPS environments	A
Coutinho and Souza Neto [120]	ChaCha	-	-	Differential-Linear attack against ChaCha	A
Drucker and Gueron [23]	OpenSSL	3.0.0	-	A reflection attack on TLS 1.3	C
Perianin et al. [24]	OpenSSL	1.1,x	-	A timing attack using CPU cache memory	A
CVE-2022-25638	wolfSSL	5.2.0 before	MITRE	Bypassing a authentication for abnormal connections	D
CVE-2022-25640	wolfSSL	5.2.0 before	MITRE	Not properly enforce a requirement	D
CVE-2022-38152	wolfSSL	5.5.0 before	MITRE	A bug of wolfSSL_clear() library function	B
CVE-2022-39173	wolfSSL	5.5.1 before	MITRE	Causing buffer overflow in a handshaking process	B
CVE-2022-42905	wolfSSL	5.5.2 before	MITRE	Enable to steal 5bytes in a special condition	B
CVE-2022-42961	wolfSSL	5.5.0 before	MITRE	ECDSA key disclosure via a Rowhammer exploit	B, C
CVE-2023-3724	wolfSSL	5.6.2 before	wolfSSL Inc.	Creating a predictable session master secret key	A, C
CVE-2023-4807	OpenSSL	-	OpenSSL	A bug of Poly1305 MAC implementation	A
CVE-2023-24609	Matrix SSL	4.x $\sim$ 4.6.0	MITRE	Wrong memory access in pre-shared key handling	A, B
CVE-2023-6129	PowerPC CPU	-	OpenSSL	A bug of Poly1305 MAC implementation	A
Bariant et al. [121]	AES128	-	-	An advanced boomerang attack on AES128	A
CVE-2023-6937	wolfSSL	5.5.6 before	wolfSSL Inc.	A bug of TLS message checkup	C
CVE-2024-28755	Mbed TLS	3.5.x $\sim$ 3.6.0	MITRE	TLS version change in the session reset process	C
CVE-2024-28836	Mbed TLS	3.5.x $\sim$ 3.6.0	MITRE	TLS version negotiation available	C
CVE-2024-37309	CreateDB	5.7.2 before	Github	Allowing client-initiated renegotiation	C
Fischlin et al. [122]	UDP	-	-	Compromise of the AEAD characteristic	D

not specified in TLS v1.3 such as protocol downgrading. Multiple vulnerabilities related to encryption schemes have been uncovered, and their collective overview is presented in Table V. The *Description* field of the table outlines the specific nature of these vulnerabilities, illustrating summaries of discovered vulnerabilities identified in TLS v1.3. They are primarily stemmed from software defects rather than inherent flaws of encryption schemes. These defects possess the potential to be combined with diverse attack scenarios, depending on the attacker's intentions. They may not have materialized into practical attacks so far, these vulnerabilities could be significant challenges in the future. We classify the vulnerabilities into four distinct categories of encryption libraries, memory access, network establishment, and bypassing, as depicted in Figure 4.

A. Encryption Libraries (Type A)

Vulnerabilities categorized as Type A in Fig 4 and Table V mean to have a weakness related to encryption libraries. Since these problems originate from wrong cipher processes, they can be easily combined with other types of vulnerabilities to bypass robust encryption schemes. An illustrative instance of this classification is CVE-2020-13777. In this case, an erroneous encryption key is employed due to an implementation flaw, replacing the key that should be utilized. This vulnerability is considered a dual-type weakness, as it has a potential to facilitate certificate bypassing.

As attack strategies have been advanced, hardware-related traits became an exploitable point to attack TLS v1.3. CVE-2021-4160 exposes a vulnerability linked to the Microprocessor without Interlocked Pipelined Stages (MIPS) architecture. In OpenSSL, a carry propagation bug was discovered within the MIPS32/64 squaring procedure, which significantly affected numerous elliptic curve algorithms. This weakness primarily stems from an architecture-oriented issue related to the key generation. However, its impact extends to various facets, notably targeting every elliptic curve-based algorithm. Another advanced attack by Perianin et al., combined a deep learning technique and hardware-related vulnerabilities to restore an encryption key [24]. They discovered that a cache operation could be vulnerable when a specific math library of OpenSSL is used. This library contains a scalar multiplication method named w-ary NonAdjacent Form (wNAF), which was identified as vulnerable [102]. Perianin et al. figured out that it

was still in use in generating an ECDSA certificate, leading to disclosing the key by integrating another cache attack [129]. The attack scenario is not a practical threat, because this protocol was already considered as a deprecated method. However, it is noteworthy that Perianin et al. made a successful encryption pattern recognition by using a deep learning model. CVE-2023-6129 highlights yet another implementation flaw arising from Performance Optimization With Enhanced RISC - Performance Computing (POWER PC) CPU-based platforms [130]. This bug stems from an issue within the OpenSSL library associated with Poly1305 on PowerPC CPUs, where vector register restoration occurs in an incorrect order. This erroneous restoration can enable attackers to execute a range of activities, spanning from DoS attacks to gaining full control over the system. While no instances of impact have been reported thus far, the broad spectrum of potential attacks underscores the critical importance of exercising caution and implementing appropriate safeguards.

Meanwhile, several other vulnerabilities were reported to be caused by wrong specific functions. A function named *SanityCheckTls13MsgReceived* proved a susceptibility to data leakage by malicious servers, as documented in CVE-2020-24613. Another weakness (CVE-2021-3336) came from the *DoTls13CertificateVerify* function, exhibiting an inability to handle abnormal situations. It sends several types of signatures without corresponding certificates, creating chances for non-authentication and potential MitM attacks. Both vulnerabilities were revealed within the wolfSSL application and subsequently remedied in later versions.

CVE-2023-3724 is related to session key generation. When a client does not receive a Pre-Shared Key (PSK) and Key Share Extension (KSE), a session key is generated through Input Keying Material (IKM). However, a predictable value can be selected as the IKM value, making a secure channel vulnerable. It does not violate a certificate process of TLS v1.3, but attackers can break channel reliability established by the IKM values if they can eavesdrop.

The last case of Type A, CVE-2023-4807, came from a bug related to Poly1305 implementations. It works when a user's X86_64 processor supports Advanced Vector Extensions 512-bit Integer Fused Multiply-Add (AVX512-IFMA) instructions. Through this bug, a specific register of systems can be changed by an attacker, causing diverse abnormal results from damaging applications to system breakdown. Fortunately, these exploits require conditions that TLS v1.3 fundamentally prevents, such as using incorrect algorithms or predictable key generation. This implies that TLS v1.3 becomes more secure over software updates. However, precedents such as export algorithms developed long ago, have already been observed causing vulnerabilities that later evolved into new exploits, such as FREAK, DROWN, and Logjam. Therefore, even if the vulnerabilities addressed in this section are already resolved through software updates, the possibility remains that they could re-emerge as new vulnerabilities when combined with other factors. This indicates that we need to keep paying attention to vulnerabilities that have already been resolved.

Apart from vulnerabilities arising from implementation flaws, notable cases in Type A involve attempts to compromise encryption algorithms previously considered secure. For example, Dunkelman et al. demonstrated an improved boomerang attack that significantly reduced computational complexity by employing data discarding strategies and variant tracing [119]. Briant et al. also introduced a new boomerang attack [121], proposing an advanced framework using a mixed-integer linear programming model to launch boomerang attacks against AES-based encryption algorithms. While both strategies have succeeded in reducing computational complexity in limited environments, they remain ineffective at fully recovering the encryption keys used in real-world scenarios.

The ChaCha encryption algorithm has also been subjected to attempts to compromise. Coutinho and Neto proposed an enhanced linear approximation strategy that simplifies cipher analysis [120]. Through this approach, they have improved the complexity of existing differential linear attacks on the ChaCha algorithm [131]. These advances suggest that encryption algorithms considered secure today may become vulnerable in the future. Thus, it is crucial to continuously monitor for emerging vulnerabilities and threats to ensure timely detection and mitigation.

B. Memory Access (Type B)

Type B vulnerabilities are rooted in memory-related issues, leveraging defects in memory management of applications. These vulnerabilities encompass a range of anomalies, including use-after-free bugs (CVE-2021-22901), segmentation faults (CVE-2022-38152), buffer overflows (CVE-2022-39173), and over-reads (CVE-2022-42905 and CVE-2023-24609). These flaws provide unauthorized access to memory addresses, furnishing attackers with a platform for executing more intricate exploits. These attacks exploit illicit memory access to load and execute malevolent code, which can conduct versatile capabilities, including encryption-related attacks.

By exploiting these unauthorized pathways, attackers can conduct many malicious actions, encompassing data manipulation, malware deployment, and beyond. While exploiting unauthorized memory access may pose a challenge, its successful execution empowers attackers to execute their malicious intentions. A noteworthy precedent of such vulnerabilities is the Heartbleed attack (CVE-2014-0160), unearthed in 2014. This attack misused OpenSSL heartbeat functions to gain unauthorized access to memory space. This attack works as follows: when the client specifies a request content size larger than the actual size, the server responds based on the manipulated size. As a result, an issue arises where data beyond the intended content size is unintentionally read from the server's memory and included in the response. It made a huge impact on modern security by showing the risks associated with memory leaks.

While the preceding examples elucidate vulnerabilities rooted in software, a distinctive attack stems from a hardware vulnerability. CVE-2022-42961 brought attention to a novel Rowhammer attack that exposed a private ECC signature key [132]. The leaked private key empowers attackers to recover an ECDSA key integral to network establishment. This poses a serious threat to the establishment of a secure network connection, given the insecure digital signature created by

the recovered key. The compromised certificate key, when exploited, has the potential to compromise encrypted channels, impeding the secure transmission of data. Memory-targeted attacks have a high privilege, which can exert most activities on a system. Due to their catastrophic potential, these Type B vulnerabilities demand persistent attention. Their capacity to undermine security, coupled with their potential for severe effects, emphasizes the need for continuous monitoring and proactive strategies.

C. Network Connection (Type C)

Type C vulnerabilities imply unauthorized operations of TLS v1.3 related to the network connection control. The protocols in TLS v1.3 mandate that certificate algorithms uphold forward secrecy and ephemeral key properties. To ensure their integrity, comprehensive investigations have been undertaken to inspect the resilience of encryption algorithms employed in earlier versions of TLS.

Jager et al. highlights a significant deficiency in PKCS#1 v1.5, demonstrating its failure to meet the forward secrecy characteristic [20]. They demonstrated that PSKs could be decrypted within the duration of a TLS session and within a time frame of 18 minutes to 202 hours for the QUIC protocol. Consequently, they strongly advocated for the exclusion of this PSK algorithm in the new TLS framework. Despite its popularity in TLS v1.2, this vulnerable PSK algorithm was banned from TLS v1.3 due to its susceptibility to exploitation across various scenarios. A dual approach was adopted that prohibits both vulnerable versions and the algorithms themselves. However, a subsequent revelation emerged in the form of CVE-2019-11727, suggesting a residual possibility of the same attack exploiting PKCS#1 v1.5. It forces a client and server to sign PKCS#1 v1.5 signatures in a TLS v1.3 connection when those are available at the server.

In the realm of certificate algorithms, additional vulnerabilities stemmed from PSKs' ephemeral key properties are discovered. Akhmetzyanova et al. explored cases wherein PSKs are shared among users in several potential scenarios [22]. Meanwhile, another vulnerability leveraged the property of PSKs, failing to validate certificate authentication when they are shared between a server and a client. Drucker and Gueron investigated this strategy, unveiling the potential for malicious servers to impersonate genuine TLS v1.3 servers. It was named a "selfie attack," spotlighting vulnerabilities in the PSKs' group_authentication mode [23].

A problem can also arise during the handshake message generation process. CVE-2023-6937 highlights a case in which the absence of boundary validation during handshake message generation allows intrusion into the session key space. While this issue does not compromise core protocol requirements, such as authentication or key negotiation, it raises concerns about potential anomalies. For instance, multiple encryption keys could be used within the same session, or an unencrypted ServerHello message might be transmitted.

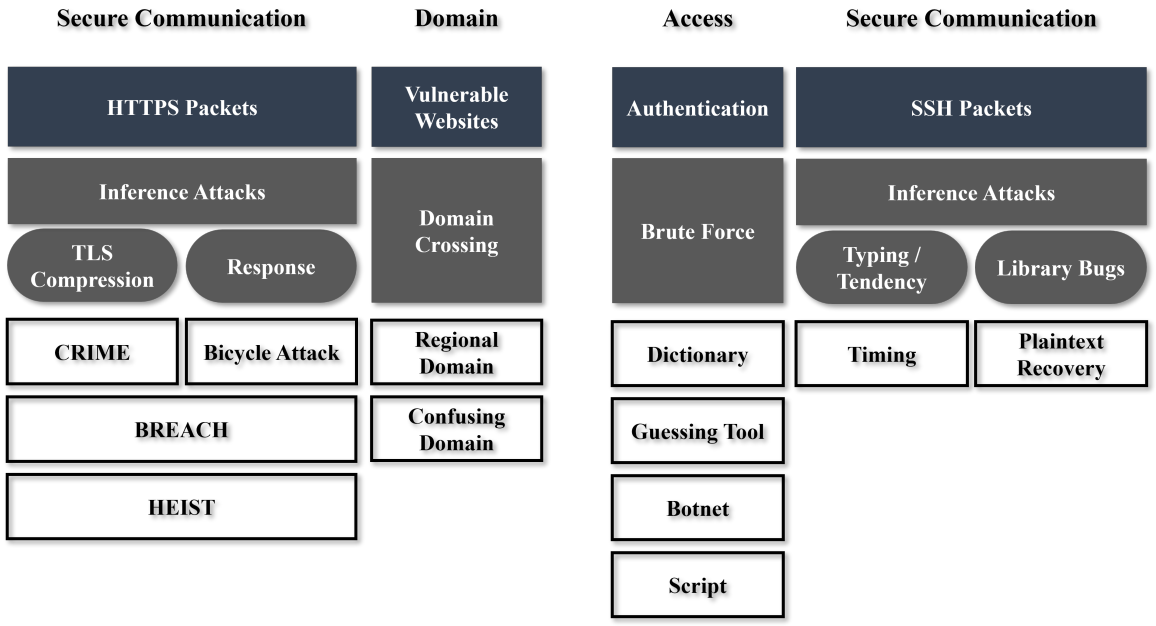
Certain vulnerabilities even precipitate TLS version downgrades, despite such actions being prohibited within TLS v1.3. Lee et al. brought to light new vulnerabilities in the network stacks of MacOS and Windows that enabled version downgrades [21]. Exploiting these flaws in the TLS fallback mechanism across multiple browser versions, they illuminated reasonable downgrading scenarios. They found that approximately 8.6% of servers do not support the Signaling Cipher Suite Value (SCSV) mechanism, which prevents downgrade attacks. Another instance of version downgrading emerged in CVE-2019-17023, targeting specific versions of the Firefox browser. In their retry request process, they negotiate a version of TLS, which was prohibited in TLS v1.3.

TLS version negotiation issues have been identified in CVE-2024-28836 and CVE-2024-37309. CVE-2024-28836 highlights a scenario in which a TLS v1.2 connection can be established even when TLS v1.2 is disabled. This vulnerability arises because the server may revert to its TLS v1.2 implementation. If the server is intended to allow only TLS v1.3, this fallback could not only facilitate a downgrade attack but also lead to Denial of Service (DoS) attacks. Similarly, CVE-2024-37309 reveals that several versions of the CreateDB client, which are used to create new PostgreSQL databases, permit negotiation of the TLS version when requesting security parameters, potentially resulting in a version downgrade or resource exhaustion DoS attacks, akin to the issue in CVE-2024-28836. Moreover, this downgrade issue was also observed during SSL/TLS session resets. CVE-2024-28755 indicates that the $mbedtls\_ssl\_session\_reset()$ function in Mbed, a development platform and operating system for IoT devices, may not preserve TLS v1.3 after a session reset. Consequently, a re-established SSL/TLS connection may revert to TLS v1.2, thereby indicating that a downgrade attack is possible in these situations.

D. Bypassing (Type D)

Previous vulnerabilities exploited algorithmic or implementation flaws, rendering encryption schemes used in TLS v1.3. However, a distinct class of vulnerabilities, denoted as Type D, circumvents the encryption schemes inherent to the new TLS paradigm. In earlier versions of TLS, attackers were able to bypass secure encryption schemes by negotiating algorithms or protocols. However, TLS v1.3 does not allow this circumstance through its robust mechanism, and has additional security restrictions such as managing a black list. In terms of this facet, it can be a critical strategy to bypass the reliability of TLS v1.3. We discuss this case in detail, to prevent future vulnerabilities.

Modern network ecosystems commonly employ TLS inspectors, specialized network monitoring systems designed to oversee various facets of network activities, encompassing handshake processes, cipher algorithm selection, and connection establishments. These insights serve several purposes, bolstering network security. Notably, CVE-2020-8660 exposed the potential for bypassing TLS inspectors within specific Cloud Native Computing Foundation (CNCF) Envoy environments, revealing a crack in modern network surveillance systems' armor. It exploits that TLS extensions are not inspected in the inspector, which enables attackers to violate security restrictions defined by the surveillance systems.



(a) Attacks on HTTPS

(b) Attacks on SSH

Fig. 5: Classification of attacks launched at the Application layer protocols. Targets are represented by navy boxes, strategies by gray boxes, and the names of publicly known attacks by white boxes. Wide boxes indicate attacks with multiple properties. Attacks on HTTPS are classified into secure communication and domain, and attacks on SSH are classified into access and secure communication.

Contemporary network defense mechanisms, managed by dedicated network enterprises, protect systems against external threats. Such systems maintain blacklists of malicious URLs, curbing access to these domains for network users. Nevertheless, CVE-2020-3285 demonstrated a lapse in this safeguarding. Exploiting an erroneous connection handling within the Snort application, this vulnerability facilitated the access of blacklisted URLs by network users.

Bypassing can extend beyond monitoring systems and manifest during the authentication process. CVE-2020-13777 and CVE-2021-22890 underscore how incorrect session ticket handling can lead to bypass vulnerabilities. In the case of CVE-2020-13777, certain versions of the Gnu TLS application persists in utilizing erroneous data that deviated from the correct encryption key. When communication is established with this incorrect data, the network connection fails to recover, culminating in authentication bypass. In a distinct scenario, CVE-2021-22890 exploited an interaction between HTTPS proxy and libcurl within specific environments. The libcurl library mishandled proxy session tickets from HTTPS proxy servers, thereby exposing a potential for MitM attacks orchestrated by malicious HTTPS proxy servers.

Some bypassing vulnerabilities are predicated upon the satisfaction of specific conditions rather than operational flaws. A striking illustration appeared in CVE-2022-25638, where the data in the ‘sig_algo’ field of a certificate message made a way for authentication bypass in certain versions of the wolfSSL. Certainly, an implementation flaw can lead to the bypass. A separate revelation emerged in CVE-2022-25640 attributed to an implementation lapse that bypassed the certificate verification process in the wolfSSL. It happened because the library was not checking requirements properly. These vulnerabilities were promptly addressed in subsequent updates. Fischlin et al. [122] argue that UDP environments compromise the security of TLS v1.3. They demonstrated that in UDP-based environments, such as QUIC and DTLS, indiscriminate and repetitive packet transmissions can lead to forgery issues related to the use of sliding-window techniques. This situation compromises the integrity of AEAD (Authenticated Encryption with Associated Data). To address this problem, they have proposed a feature called “robustness”, which is a channel’s property to filter out any misplaced ciphertexts and correctly receive ciphertexts that fit into the supported order.

V. Attacks on HTTPS

The popularity of SSL/TLS makes many attackers focus on discovering vulnerabilities to break the protocol. However, doing so is challenging because the latest verison of TLS (v1.3) recommends using secure and dependable algorithms. Thus, recently, the attackers attempted alternate approaches, exploiting the properties of Application layer protocols. HTTPS is one of the most attacked Application layer protocols as stated in the Edgescan report [64]. It has been updated through several versions, possessing diverse extensions to support numerous functions. Accordingly, various attack strategies targeting HTTPS have been discovered, including code injection [133], Denial-of-Services (DoS) [134], Phishing attacks [135], and one that extracts a user’s personal information by monitoring and analyzing a keyboard typing tendency [136]. More attacks can be launched on HTTPS by

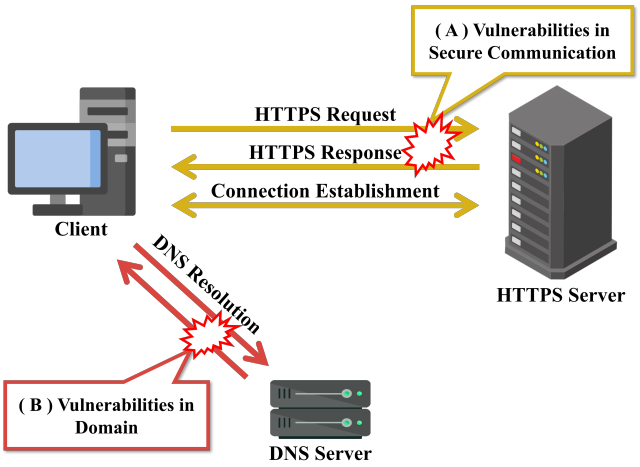


Fig. 6: Types of Attacks on HTTPS

leveraging different strategies. For instance, existing research results introduced vulnerabilities examined between HTTPS and the Wi-Fi protocol [137], [138], yet they predominantly stem from the Wi-Fi protocol, and only tangentially involving HTTPS. Clearly, these types of attacks can be critical threats against the security of HTTPS [64]. However, they are out of the scope of this paper because our main focus encompasses attacks and vulnerabilities undermining encryption algorithms by exploiting the specific attributes of HTTPS; they choose bypassing encryption schemes, not breaking them. In addition, HTTPS utilizes SSL/TLS protocol for secure network communication, and their vulnerabilities are covered in the previous sections. Therefore, this section only focuses on the exclusive vulnerabilities exploiting HTTPS properties. We conduct an in-depth analysis targeting properties of HTTPS, categorizing the strategies into two primary domains, Secure Communication and Domain-based attacks, as illustrated in Figure 5a. The vulnerabilities scrutinized within each category are depicted in Figure 5.

Type (A) in Fig 6, also denoted as Secure Communication in Figure 5a, encompasses the cases of deliberate manipulations of encryption algorithms. HTTPS, known as a secure version of HTTP, makes it harder for the attackers to decipher the ecnrypted transmitted data. For instance, attempting to decode encrypted data utilizing a brute force technique on AES-256 would require over a decade of relentless effort [139]. However, this complexity diminishes if attackers can pinpoint specific target data sizes or discern data transmission patterns. Termed an inference attack, this method exploits the characteristics of Application layer protocols to narrow down the target range and expedite the decryption process significantly [140]. Type (B) in Fig 6, also denoted as Domain in Figure 5a, focuses on circumventing strong encryption mechanisms through distinct attributes of HTTPS. These attacks induce clients to access vulnerable websites, subsequently leveraging the susceptibilities of these sites. This is feasible due to the diversity of cybersecurity standards across different countries, coupled with the utilization of domain names for HTTPS access. In the following sections, we explore both attack types, investigating the characteristics they exploit.

A. Secure Communication

1) Inference Attack (TLS compression)

The Compression Ratio Info-leak Made Easy (CRIME) attack operates as an inference attack, exploiting vulnerabilities in HTTPS compression techniques [140]. HTTPS uses compression to reduce the size of transmitted data. It was a useful strategy for a large volume of data and a tendency of increasing data size. However, if MitM attack is possible, CRIME can disclose particular data such as the length of transmitted data. An attacker inserts additional data on hijacked request, which changes the compression rate and size of compressed data. By analyzing data differences, the attacker can deduce the encrypted content accessed by victims. This inference enables to narrow down target ranges, facilitating efficient chosen-plaintext attacks [141]. The target ranges include numerous types of data; some of which have private information. For example, attackers can hijack web sessions if the data is an HTTP cookie that has session information.

2) Inference Attack (Response)

Inference strategies are also applicable to HTTP response messages. The original bicycle attack introduced the concept of inferring a particular component's length from the HTTP response [142]. Although this attack was impractical due to its excessive requirements, such as physical access and a fake CA, it can have dire consequences if it succeeds. A few years later, a practical bicycle attack emerged, showcasing scenarios wherein password length could be inferred, contingent upon knowledge of the IP address and user information [143]. This attack shows the potential case that can guess personal password details from secure AES-GCM traffic. As it provides the means to identify password length, this strategy significantly undermines secure encryption algorithms, rendering it susceptible to basic attacks like brute force. For instance, the success rate doubles when an attacker knows the length of the passwords. Even a secure encryption algorithm can be compromised if attackers can specify the data length, and it enables swift interpretation over a short period. These inference attacks show the reality that algorithmic security does not guarantee overall security if there are vulnerabilities in other components of a system or in the other layers of the network.

3) Inference Attack (Combination)

One simple strategy against CRIME attacks is disabling the TLS compression function in web browsers. However, this attack has recurred with the Browser Reconnaissance and Exfiltration via the Adaptive Compression of Hypertext (BREACH) attack [144]. Disabling the TLS compression makes HTTP protocol employ gzip for the reduction of the volume of network traffic. By exploiting gzip compression algorithms, attackers can infer characters within the HTTP response as the CRIME attack did. The similarity between gzip and TLS compression schemes allows attackers to infer content size, granting an interpretation of a single byte in the HTTP response per request. This vulnerability can be exploited for various vicious purposes, including extracting Cross-Site

Request Forgery (CSRF) token values from malicious servers and establishing connections with victims using stolen tokens, resulting in substantial damage; over 95% of one CSRF token data can be recovered, and it can sometimes be decrypted in as little as 30 seconds.

Regarding encryption schemes, the concern with CRIME and BREACH is applicable to any ciphersuites within TLS connections that employ compression techniques. This risk persists despite the activation of HTTP Strict Transport Security (HSTS), a protocol designed to protect against attacks such as cookie hijacking and downgrading. Fortunately, the vulnerabilities can be prevented by disabling compression techniques, and software updates appeared to deal with these inference attacks. Nonetheless, a more sophisticated strategy emerged, namely HTTP Encrypted Information can be Stolen through TCP-windows (HEIST) [145]. When users access malicious websites, their browsers execute malicious JavaScript that the attackers installed. It enables the attackers to trace a specific API call, sneaking TCP windows-related data, which can delimit a range of encrypted data. This extracted data is then combined with inference attack techniques to retrieve encrypted information from user request packets. On average, the HEIST attack is 280% faster than other compression vulnerability-based attacks, which is a remarkable advantage in the modern network connection environment having limited connection time. Notably, this attack does not necessitate a MitM approach and operates solely through JavaScript execution. Proposed countermeasures, such as disabling JavaScript and cookies in web browsers, impede the user experience in web environments; though they are effective.

B. Domain-related Attack

The second type of attack on HTTPS primarily exploits the inherent nature of HTTPS users accessing websites via domains. This method is attractive because individuals usually do not verify the domain names rigorously. It works like a downgrading attack by leveraging regional domain characteristics. Alashwali et al. investigated the security disparities across regional domains, uncovering vulnerabilities that came from security policies across various countries [146]. Such inconsistencies can be exploited to redirect users to vicious servers or IPs and induce them to use insecure security protocols. Some countries permit insecure protocols or algorithms based on their specific needs for complicated reasons. Alashwali et al. has categorized three cases where URL and HTTPS strategies are inconsistent and found over 1000 domains in each case. These security policy differences enable attackers to exploit these permissions, potentially transforming secure connections into downgraded TLS versions or even plain HTTP communications. Some attacks exploit domain names themselves. Unoccupied domains can be easily acquired by anyone, including attackers. They try to get recently expired domains or fake domains similar to trustworthy domains. These domains are frequently misused to inadvertently lead users to malicious websites.

An advanced variant exploits both domain and certificate information properties. Delignat-Lavaud and Bhargavan showed

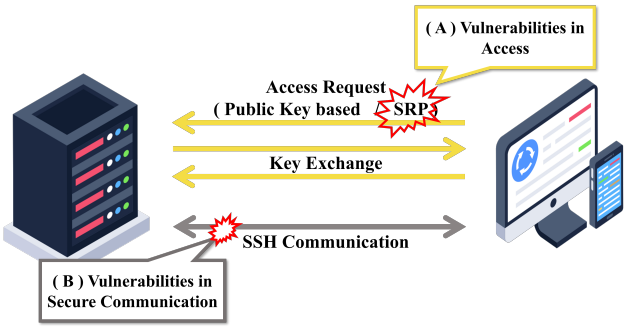


Fig. 7: Types of Attacks on SSH

vulnerabilities arising in multi-domain scenarios employing the same encryption key for authentication and session ticket generation [147]. Here, an authorization process of a server certificate is managed through a virtual host, which involves fallback algorithms. After investigation, 36% of Internet Information Services (IIS) servers, 35% of Apache servers, and 14% of NginX servers were exploitable via the fallback mechanism [147]. The authors also showed that this process can be used to open a backdoor to cross-site scripting (XSS), redirecting users to malicious sites. Such an attack can evade robust security measures akin to HSTS and effectively undermine encryption schemes by exploiting authentication process flaws.

C. Other Attacks Exploiting HTTP Properties

Some attacks aim to bypass security protection mechanisms rather than undermining encryption schemes. Notable examples include HTTPS Request Smuggling (HRS) and Cross-Site Request Forgery (CSRF), both of which exploit HTTPS properties to bypass encryption [148], [149]. HRS manipulates HTTP request parameters to confuse victims and initiate malicious activities, while CSRF exploits pre-validated session identifiers to covertly execute unauthorized requests. These attacks are further extended to various advanced versions [150], [151], [152] and are significant examples of attacks leveraging HTTPS properties to bypass strong encryption techniques. While HRS and CSRF are not directly related to vulnerabilities in encryption algorithms, we include them due to their potential to exploit HTTP characteristics, bypass SSL/TLS authentication, and access private data.

VI. Attacks on Secure SHell

Secure SHell (SSH), a distinctive application layer protocol, has a variety of identified vulnerabilities. Encryption algorithms allowed to use SSH are akin to the ones for SSL/TLS. Thus, SSH also suffers from the similar algorithmic vulnerabilities introduced in Section III. This section focuses on attacks independently designed to compromise the encryption schemes employed in SSH, which can be categorized into two classes: Access and Secure Communication, as illustrated in Figure 5b. Figure 7 depicts the points of these attacks.

The first category is related to its authentication process. Because of a unique characteristic of the protocol, SSH permits

SRP which is deprecated by other protocols due to security issues. It allows users to access identifications and passwords, causing diverse vulnerabilities related to their creation. We discuss the attacks that utilize this property to extort login authority by attackers. The second category encompasses cases exploiting the features of SSH packet communication. As a responsive protocol, SSH connections keep communicating during the lifetime of the sessions. Various features generated in these communications are exploitable by attackers. We demonstrate malicious activities capitalized on them.

Beyond these categories, one popular attack on SSH, Address Resolution Protocol (ARP) spoofing, has evolved into a primary strategy for MitM attacks, particularly as encryption ciphers grow more robust. However, we do not cover this type of attack in detail because it is conducted in lower layers of the TCP/IP protocol stack and does not engage with encryption algorithms.

A. Access

One of the most distinctive properties of SSH is its simple authentication mechanism. Despite the robust certificate authentication offered by SSH, many users select personal accounts for accessing SSH servers, and some users set their accounts with a weak text-based password. Many SSH servers use the default port number, causing several issues. Weak account IDs and passwords can be easily exposed to simple hacking attempts or security threats. The attacks become even easier when default port numbers are used. Through macro-based attacks, such as a bot, attackers can target SSH servers that meet the conditions, including using the default port number or weak account IDs and passwords. To counter such attacks, many systems employ Intrusion Detection Systems (IDSs) that block access if abnormal activities (including excessive login attempts) are detected. For applications supporting SSH functions, there are usually useful IDSs provided by the application publishers, but in a default SSH environment, users need to update the IDS settings using the configuration file. However, like the port setting, default IDS settings are often used. Such default IDS settings also render them vulnerable to brute force attacks using bots or scripts. For example, attackers can avoid detection by making just enough login attempts to not trigger the alarms and then disconnecting before trying again because they know the default IDS setting values. These brute force attacks can be categorized into the following methods: Dictionary attacks, Guessing tools, Botnets, and Scripts, as depicted in Figure 5b.

Dictionary attacks depend on a list containing commonly used account names and passwords, systematically attempting logins using these records [153]. Another variant relies on guessing tools that automatically predict user passwords or decipher password creation patterns, enabling attackers to target SSH users. Prominent password-cracking tools, such as Hydra, Ncrack, and Patator, excel in this task [154], [155], [156]. Similarly, Botnets constitute an automated brute-force strategy, similar to guessing tools. Attackers distribute these Botnets, which then search a target system thoroughly for valuable data to facilitate password guessing. This data can include parts of a dictionary or even direct passwords/accounts for login. Some Botnets are equipped with automated login functions or can download additional malware, enabling more advanced attacks. Notable examples include Brickerbot and Reaper, which primarily focus on IoT devices' network connections [157], [158]. The other method of brute-force attack is implementing malicious script code that is designed to automate attack attempts (denoted as *Script* in Figure 5b). Other automatic strategies require complicated techniques for spreading and monitoring their malicious works. However, script-based attacks can spread through simple activities, such as malicious website operations. Therefore, as a more straightforward approach compared with guessing tools or botnets, script attacks can be executed by numerous individual attackers. Their functionalities resemble those of previous automatic attacks, with BruteSpray being an illustrative example [159].

B. Secure Communication

Attacks on SSH communications often target vulnerabilities in network connections, particularly network packets. Similar to attacks on HTTPS, SSH communications can be susceptible to eavesdropping, allowing attackers to narrow down the range for decrypting ciphertexts. However, with the advancement of robust encryption algorithms, SSH is now largely immune to eavesdropping and inference-based attacks. The majority of modern SSH attacks are brute force in nature, as previously discussed. Nevertheless, it is important to consider all types of attacks, including outdated ones, since strategies like downgrading can still be exploited at any time.

1) Inference attack (Typing / Tendency)

The timing attack is a representative one that uses the SSH characteristic [136]. It exploits the fact that SSH sessions persist until they are terminated and collects keyboard typing (keystrokes) patterns or periodic activities. Attackers then deduce the progression of communication by analyzing the keystroke patterns and SSH packets intercepted during interactive sessions. Attackers can extrapolate specific information through the pattern analysis of SSH packet transmissions and tendencies. Essentially, this inference attack allows them to infer the size and segment of transmitted data by closely examining these behavioral patterns. This reduces the data size and makes it easy to decrypt, potentially incapacitating even robust encryption algorithms. Fortunately, this attack has been reported as non-practical, and the latest SSH incorporates enhanced robustness against attacks utilizing packet pattern analysis [160]. Due to these challenges, modern timing attacks target vulnerable network systems across various environments, rather than aiming for a specific protocol [161], [162]. Over time, they have been gradually upgraded to cope with defense systems [97], [24]. However, adopting robust encryption schemes and the development of diverse detection metrics have made timing attacks difficult to succeed, because they still need various conditions such as eavesdropping and decryption.

2) Inference attack (Library Bugs)

The other form of inference attack is shown in Fig. 5b, a OpenSSH library bug. It exploits vulnerabilities in an open

library to retrieve plaintext from ciphertext [163]. This type of attack shows that the SSH Block Packet Protocol (BPP), considered to be a secure protocol, can be exploited due to a design flaw introduced in RFC 4253. The bug in the OpenSSH library that implements BPP allowed attackers to deduce the size of the last block. Consequently, the SSH BPP inadvertently leaks information related to the transmitted data size, and this information can make it susceptible to a brute force attack. After revealing this vulnerability, the OpenSSH team released OpenSSH v5.2, incorporating a patch to mitigate this attack vector. However, the threat of plaintext recovery attacks persisted even after the bug fix. From OpenSSH version 5.2 to version 7.4, three additional bugs were discovered [164].

Over time, encryption techniques have significantly improved in reliability. Furthermore, innovative categorization methods have been proposed to classify SSH attacks more efficiently, enhancing the efficacy of security systems [165], [166]. Consequently, the efforts to reinforce cybersecurity have restrained recent successful attacks in modern SSH network environments.

VII. Open Research Challenges

This section introduces several research challenges derived from our comprehensive investigation of CNPs. We selected three distinct topics: network protocols that support secure communications, future encryption algorithms, and cyber threats exploiting encryption algorithms in modern networks.

A. Protocols for Secure Communication

Throughout this paper, we introduce investigation results of vulnerabilities related to the encryption schemes of CNPs. Old CNPs have out-of-date encryption schemes and algorithms, which are vulnerable to various attacks. However, the release of new versions resolves these problems by revising their available algorithm list. For example, the latest CNPs, such as TLS v1.3 and SSH 2.0, manage only robust encryption schemes. They are evaluated as trustworthy protocols, and only some implementation bugs have been reported, not algorithmic vulnerabilities. In this context, in the following, we discuss future challenges related to modern CNPs.

1) Software Bug and Vulnerability Detection

Since up-to-date CNPs are theoretically flawless, modern network environments can be more secure as the usage of the CNPs increases. However, the problem is that network protocols are implemented in complicated conditions where numerous libraries and programs are entangled. As shown in Table V, protocols have unexpected results because of their complexity, which may develop into a security weakness. Resolving bugs requires a deep understanding of the relevant knowledge, and they are typically reported by researchers or service providers after completing their analysis. However, this maintenance approach is too passive for Internet users who are exposed to a real-time attack. Therefore, developing detection techniques that perceive software bugs or vulnerabilities of protocol workloads in real-time can make a significant impact on future network security.

Another challenge is predicting the lifespan of encryption algorithms, which is influenced by evolving factors such as computing power and network environments. As these environments continuously change, research into the relationship between the lifespan of encryption algorithms and computing conditions has been critical. A theoretical framework that accounts for contemporary conditions is essential for predicting the lifespan of encryption algorithms. This type of analysis has long been conducted to proactively phase out algorithms before they become vulnerable [167], [9], and it is expected to remain important in the future. Moreover, with the development of next-generation cryptographic techniques and the advent of quantum computing, the need for research into encryption algorithm lifespans will likely intensify.

2) Versions Compatibility

One of the major problems with the modern CNPs is that they work simultaneously in various versions. It makes CNPs vulnerable to downgrade attacks, even though it has the advantage of service compatibility. Finding a solution to resolve the compatibility issues can therefore make a remarkable impact on modern network security; however, it is not a practical approach considering countless network service providers and Internet users. A compromise could involve detecting changes in encryption schemes such as version downgrading or encryption algorithm change. These malevolent changes can cause downgrade attacks that are the most disruptive attack to modern robust CNPs. Downgrading strategies aim to manipulate CNPs by changing protocol versions to obsolete ones or robust algorithms to vulnerable ones. It allows attackers to exploit existing vulnerabilities, which generates enormous expenses for cybersecurity by expanding the security coverage to consider. Through the detection, systems can prevent the source of downgrade attacks effectively by corresponding their threats depending on the security levels of changed schemes, such as disabling network connections or rolling back to safe versions. In this vein, security monitoring may play a pivotal role as well as the role of enabling technologies such as Threat Intelligence (TI) and Digital Twin (DT) can be investigated.

B. Future Encryption Algorithms

Due to the abundance availability of computing power and capabilities, various encryption algorithms that used to be secure have become vulnerable [9], [139]. Moreover, attack strategies also have evolved. While earlier attacks utilized naive methods like brute force, modern attacks use smart strategies, such as oracle exploitation [144] and deep learning [24]. In the past, algorithm robustness was improved by increasing the key size, which is the most convenient approach. However, the smart attack strategies promoted the advancement of encryption algorithms because they can break the simple countermeasure by pinpointing the target to decipher. Improvement in computing power and advancement of attack strategies threatened ordinary encryption algorithms, causing new types of encryption algorithms to counter the threats. Quantum computing capabilities will render the existing algorithms insecure as it can break the encryption by either brute-force or combing other methods such as Artificial Intelligence (AI).

1) Post-Quantum Cryptography

Quantum computing can break modern encryption algorithms that are generally considered as robust algorithms. Breaking them generally requires huge computing power to find the cryptographic keys, and quantum computing is known to have sufficient capability to do that. More precisely, prime-factor-based encryption algorithms, represented by public key cryptography are more vulnerable to such attacks [168]; fortunately, symmetric encryption algorithms resist quantum computing by expanding their encryption key [169]. It implies that modern key exchange algorithms will not be secure after Q-day. Since those algorithms have been adopted in the most up-to-date network protocols for secure communication, the emergence of quantum computing can cause the reconstruction of these protocols. In response to these threats, Post-Quantum Cryptography (PQC) has been studied, having 5 different types: Lattice-based Cryptography, Multivariate Cryptography, Hash-based Cryptography, Code-based Cryptography, and Isogeny-based Cryptography [170], [171]. These five types of PQC can replace existing public key cryptography in principle. However, these types of algorithms are expected to require additional time for enhanced security, because they are still under investigation and not in commercial use. Therefore, research in PQC stabilization is needed for future network security.

The advancements in quantum computing further necessitate changes not only in encryption algorithms, but also in mechanisms that utilize these algorithms. TLS is a representative example. It relies on public key cryptography to securely exchange keys used for encrypting data in transit. However, it faces significant vulnerabilities in the presence of quantum computing, which can easily break the underlying computational hardness assumptions [168]. To address this challenge, new key exchange mechanisms, such as KEMTLS, have been proposed; it utilizes a Key Encapsulation Mechanism (KEM) to enhance security against quantum attacks [172]. Since KEM was proposed and developed, various advanced strategies and analyses have been carried out [173], [174], [175], [176]. Furthermore, efforts to support a secure network environment against quantum computing-based attacks have been made by organizations such as the National Institute of Standards and Technology (NIST) and the Internet Engineering Task Force (IETF), requiting continuous researches [177], [178]. Nevertheless, further research is needed in this direction to deal with compatibility and efficiency.

2) Homomorphic Encryption

Homomorphic Encryption (HE) is considered to be the prospective standard in encryption algorithms. The groundbreaking concept of HE, unveiled in 2009, ensures consistent results between computations on ciphertext and plaintext [179]. Despite this remarkable advancement, practical limitations such as cipher performance and the effective management of encryption noise have emerged [180]. Numerous studies have been undertaken to address these constraints, yet they persist as unsolved challenges [181]. The realization of Fully Homomorphic Encryption (FHE) could position HE to replace contemporary standard algorithms, given the satisfaction of FHE conditions. FHE guarantees uniform results even after an infinite number of operations. Consequently, research focused on mitigating limitations or understanding the implications of applying this algorithm within modern network systems holds significant importance.

3) Attribute-based Encryption

Attribute-based Encryption (AE) leverages specific attributes or information to encrypt data [182]. This approach allows AE to use a wide range of attribute combinations as encryption keys, making it difficult for attackers to break the encryption. AE is particularly effective in modern network environments such as cloud and distributed systems, where nodes perform complex operations [183], [184]. By selecting appropriate attributes, AE can securely differentiate between nodes without the need for highly complex encryption operations.

This method is not only efficient in distributed systems, but also in cryptographic scenarios where temporary data is involved. Traditionally, encryption keys are generated using random values, but keys can also be derived from temporary information available in the system. For example, Gonde et al. proposed generating a key stream by using network information as the encryption key for session establishment [62]. However, since AE relies on information to generate encryption keys, the system security depends on the nature of that information. If easily predictable attributes are used, the encryption becomes vulnerable. Commonly used attributes may be scalable, but their generality increases the likelihood that attackers could predict them. Therefore, balancing between general and unique attributes is essential for the effective operation of AE [185]. The selection of proper attributes is the most critical factor in this encryption mechanism, and as different domains require different sets of attributes, identifying suitable attributes for specific use cases remains an important area of research.

C. Exploitation of Encryption Algorithms

Encryption brought about significant changes in diverse aspects, contributing to supporting secure network communication for Internet users. There have been positive aspects, but not always. Attackers are also able to use encryption algorithms to conceal their malicious activities, and diverse cyber attacks have been employing encryption tactics as this has become popular and easy to do.

1) Malware Exploiting Encryption

Ransomware is a representative attack exploiting encryption techniques for a main attack strategy. Ransomware aims to disable target systems by encrypting the victim's data for extorting ransom. Modern ransomware manages its encryption algorithms for data encryption and encryption key delivery, akin to TLS. They mainly exploit encryption algorithms considered hard to decipher, which include most encryption algorithms used in TLS and even private algorithms [186]. Except for ransomware, the other malware uses encryption strategies partially, not a main attack method [187], [188]. They encrypt transmitting data through encryption techniques to avoid surveillance systems. In addition, some other attacks exploit encryption schemes for obfuscation, which disable

target systems or specific data [189]. While these attacks have been commonly discovered in the modern network, no existing studies investigate encryption schemes adopted by malware. Existing surveys still focus on classifying malware with classic types, which are not suitable to address up-to-date cyber threats [190], [191]. Therefore, there is an urgent need to develop a countermeasure against these types of malware exploiting encryption algorithms.

2) Encrypted Traffic from Malware

Early studies in traffic analysis aimed at detecting malicious activities primarily focused on identifying significant features for traffic classification [192], [193]. However, these approaches have demonstrated limited effectiveness when faced with encrypted traffic [194], [195], [196]. Malicious actors could easily use encryption algorithms to obscure the content and address, making it difficult for surveillance systems to detect the malicious activities. In this vein, the emergence of Machine Learning-based methods has significantly improved traffic classification techniques, with models like Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) playing a crucial role [197], [198]. These models excel at processing large datasets and can simultaneously compute a wide array of features, helping to overcome challenges in traffic classification by effectively utilizing communication features such as packet data and handshake messages [199], [200]. Furthermore, in recent years, Natural Language Processing (NLP) models have demonstrated exceptional performance in classifying sequential data [201], [202], making them particularly suited for the task of malware traffic classification [203], [204], [205]. Given their strengths, NLP-based classification methods and other AI-based techniques are poised to become a promising direction for future traffic classification research.

VIII. Conclusion

Encryption schemes play a pivotal role against the multitude of threats that have proliferated in the modern network environment. Malicious activities have become sophisticated, and encrypted channels are the primary shield against threats. Nevertheless, modern studies neglect to investigate encryption schemes used in network communications as the stability of modern CNPs has improved. However, their vulnerabilities are a significant target that we must persistently focus on, and they can lead to catastrophic results in cybersecurity, even if their flaws are trivial.

Our investigation exhibits a new approach to exploring modern vulnerabilities stemming from modern network protocols, depending on encryption schemes. This approach is a novel classification methodology, considering that existing surveys for encryption schemes stand traditional classification approaches such as malware types, or restricted domains. In this paper, we organized encryption strategies and their weaknesses depending on network protocols, with SSL/TLS, HTTPS, and SSH serving as our principal subjects. Our presentation of the intricacies and vulnerabilities inherent in the mentioned protocols constitutes a distinctive and exhaustive survey. With this investigation, we can yield a profound understanding of modern encryption methodologies and vulnerabilities exploitable by attackers. In closing, we suggest various open research challenges that could be derived from our investigation.

This survey contributes to various aspects. The first is a novel classification approach. Our approach emphasizes encryption schemes in the exploration of contemporary network vulnerabilities, and it applies to other domains. For instance, malware using encryption schemes can be explored. Second, we investigated vulnerabilities of TLS v1.3, which have never been addressed in the existing surveys. It is considered a robust protocol, using only secure encryption algorithms so far. However, several implementation flaws and studies have been reported, and we scrutinized its causes and backwash. Finally, individuals can gain a more adept comprehension of the constantly evolving and dynamic field of cybersecurity by understanding the historical vulnerabilities comprehensively. With comprehension, they can proficiently cope with their problems when they need to figure out the encryption schemes of CNPs. Through the contributions, we expect that prospective investigations can analyze their research subjects from diverse points of view with a profound understanding of modern network protocol security.

Acknowledgement

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS 2023-00261068, Development of Lightweight Multimodal Anti-Phishing Models and Split-Learning Techniques for Privacy-Preserving Anti-Phishing), (No. RS-2022-00155885, Artificial Intelligence Convergence Innovation Human Resources Development (Hanyang University ERICA)), and (No. RS-2024-00431388, the Global Research Support Program in the Digital Field program). Junggab Son and Kyungtae Kang are corresponding authors of this paper.

References

[1] A. Bannister, "Google to bolster Chrome privacy protections with HTTPS-First Mode," Accessed: 01/03/2025. [Online]. Available: https://portswigger.net/daily-swig/google-to-bolster-chrome-privacy-protections-with-https-first-mode

[2] B. Beurdouche, K. Bhargavan, A. Delignat-Lavaud, C. Fournet, M. Kohlweiss, A. Pironti, P.-Y. Strub, and J. K. Zinzindohoue, "A messy state of the union: Taming the composite state machines of tls," *Communications of the ACM*, vol. 60, no. 2, pp. 99–107, 2017.

[3] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta *et al.*, "Imperfect forward secrecy: How Diffie-Hellman fails in practice," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 5–17.

[4] M. Vanhoef and F. Piessens, "All Your Biases Belong to Us: Breaking RC4 in WPA-TKIP and TLS," in *Proceedings of the 24th USENIX Security Symposium*, 2015, pp. 97–112.

[5] N. J. Al Fardan and K. G. Paterson, "Lucky thirteen: Breaking the TLS and DTLS record protocols," in *Proceedings of the 34th IEEE Symposium on Security and Privacy*, 2013, pp. 526–540.

[6] N. Aviram, S. Schinzel, J. Somorovsky, N. Heninger, M. Dankel, J. Steube, L. Valenta, D. Adrian, J. A. Halderman, V. Dukhovni *et al.*, "DROWN: Breaking TLS Using SSLv2," in *Proceedings of the 25th USENIX Security Symposium*, 2016, pp. 689–706.

[7] EnterpriseAppsToday, "24+ SSL Statistics To Secure Your Browsing in 2024," Accessed: 01/03/2025. [Online]. Available: https://www.enterpriseappstoday.com/stats/ssl-statistics.html

[8] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta *et al.*, "Imperfect forward secrecy: How diffie-hellman fails in practice," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 5–17.

[9] E. Barker and A. Roginsky, "Transitioning the use of cryptographic algorithms and key lengths," National Institute of Standards and Technology, Tech. Rep., 2018.

[10] K. Bhargavan and G. Leurent, "On the practical (in-) security of 64-bit block ciphers: Collision attacks on HTTP over TLS and OpenVPN," in *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 456–467.

[11] C. Young, "What is GOLDENDOODLE Attack?" Web Report, 2019, Accessed on 01/03/2025. [Online]. Available: https://www.tripwire.com/state-of-security/goldendoodle-attack

[12] J. Tournier, F. Lesueur, F. L. Mouël, L. Guyon, and H. Ben-Hassine, "A survey of iot protocols and their security issues through the lens of a generic iot stack," *Internet of Things*, vol. 16, p. 100264, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660520300986

[13] A. Lamssaggad, N. Benamar, A. S. Hafid, and M. Msahli, "A survey on the current security landscape of intelligent transportation systems," *IEEE Access*, vol. 9, pp. 9180–9208, 2021.

[14] B. Bhushan and G. Sahoo, *Requirements, Protocols, and Security Challenges in Wireless Sensor Networks: An Industrial Perspective*. Cham: Springer International Publishing, 2020, pp. 683–713. [Online]. Available: https://doi.org/10.1007/978-3-030-22277-2_27

[15] P. Williams, I. K. Dutta, H. Daoud, and M. Bayoumi, "A survey on security in internet of things with a focus on the impact of emerging technologies," *Internet of Things*, vol. 19, p. 100564, 2022.

[16] V. A. Thakor, M. A. Razzaque, and M. R. Khandaker, "Lightweight cryptography algorithms for resource-constrained IoT devices: A review, comparison and research opportunities," *IEEE Access*, vol. 9, pp. 28 177–28 193, 2021.

[17] W. M. Shbair, T. Cholez, J. Francois, and I. Chrisment, "A survey of HTTPS traffic and services identification approaches," *arXiv preprint arXiv:2008.08339*, 2020.

[18] O. Levillain, "Implementation flaws in TLS stacks: lessons learned and study of TLS 1.3 benefits," in *Proceedings of the 15th International Conference on Risks and Security of Internet and Systems*, 2021, pp. 87–104.

[19] Y. Joarder and C. Fung, "A Survey on the Security Issues of QUIC," in *Proceedings of the 6th Cyber Security in Networking Conference*, 2022, pp. 1–8.

[20] T. Jager, J. Schwenk, and J. Somorovsky, "On the security of TLS 1.3 and QUIC against weaknesses in PKCS# 1 v1. 5 encryption," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1185–1196.

[21] S. Lee, Y. Shin, and J. Hur, "Return of version downgrade attack in the era of TLS 1.3," in *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*, 2020, pp. 157–168.

[22] L. Akhmetzyanova, E. Alekseev, E. Smyshlyaeva, and A. Sokolov, "On post-handshake authentication and external PSKs in TLS 1.3," *Journal of Computer Virology and Hacking Techniques*, vol. 16, no. 4, pp. 269–274, 2020.

[23] N. Drucker and S. Gueron, "Selfie: reflections on TLS 1.3 with PSK," *Journal of Cryptology*, vol. 34, no. 3, p. 27, 2021.

[24] T. Perianin, S. Carré, V. Dyseryn, A. Facon, and S. Guilley, "End-to-end automated cache-timing attack driven by machine learning," *Journal of Cryptographic Engineering*, vol. 11, no. 2, pp. 135–146, 2021.

[25] G. Singh, "A study of encryption algorithms (RSA, DES, 3DES and AES) for information security," *International Journal of Computer Applications*, vol. 67, no. 19, 2013.

[26] M. Ebrahim, S. Khan, and U. B. Khalid, "Symmetric algorithm survey: a comparative analysis," *arXiv preprint arXiv:1405.0398*, 2014.

[27] S. Nithya and E. Raj, "Survey on asymmetric key cryptography algorithms," *Journal of Advanced Computing and Communication Technologies 2014*, vol. 2, no. 1, pp. 1–4, 2014.

[28] M. F. Mushtaq, S. Jamel, A. H. Disina, Z. A. Pindar, N. S. A. Shakir, and M. M. Deris, "A survey on the cryptographic encryption algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 11, 2017.

[29] M. B. Yassein, S. Aljawarneh, E. Qawasmeh, W. Mardini, and Y. Khamayseh, "Comprehensive study of symmetric key and asymmetric key encryption algorithms," in *Proceedings of the 13th international conference on engineering and technology*, 2017, pp. 1–7.

[30] I. Bhardwaj, A. Kumar, and M. Bansal, "A review on lightweight cryptography algorithms for data security and authentication in IoTs," in *Proceedings of the 4th International Conference on Signal Processing, Computing and Control*, 2017, pp. 504–509.

[31] S. Singh, P. K. Sharma, S. Y. Moon, and J. H. Park, "Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–18, 2017.

[32] M. Al-Shabi, "A survey on symmetric and asymmetric cryptography algorithms in information security," *International Journal of Scientific and Research Publications*, vol. 9, no. 3, pp. 576–589, 2019.

[33] A. M. Qadir and N. Varol, "A review paper on cryptography," in *Proceedings of the 7th International Symposium on Digital Forensics and Security*, 2019, pp. 1–6.

[34] M. N. Alenezi, H. Alabdulrazzaq, and N. Q. Mohammad, "Symmetric encryption algorithms: Review and evaluation study," *International Journal of Communication Networks and Information Security*, vol. 12, no. 2, pp. 256–272, 2020.

[35] Y. Salami, V. Khajevand, and E. Zeinali, "Cryptographic algorithms: a review of the literature, weaknesses and open challenges," *J. Comput. Robot*, vol. 16, no. 2, pp. 46–56, 2023.

[36] A. Olutola and M. Olumuyiwa, "Comparative analysis of encryption algorithms," *European Journal of Technology*, vol. 7, no. 1, pp. 1–9, 2023.

[37] M. Ugbedeojo, M. O. Adebiyi, O. J. Aroba, and A. A. Adebiyi, "Rsa and elliptic curve encryption system: A systematic literature review," *International Journal of Information Security and Privacy (IJISP)*, vol. 18, no. 1, pp. 1–27, 2024.

[38] P. Suryateja and K. V. Rao, "A survey on lightweight cryptographic algorithms in iot," *Cybernetics and Information Technologies*, vol. 24, no. 1, pp. 21–34, 2024.

[39] S. Singh, P. K. Sharma, S. Y. Moon, and J. H. Park, "Advanced lightweight encryption algorithms for iot devices: survey, challenges and solutions," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–18, 2024.

[40] K. Bhargavan, C. Fournet, M. Kohlweiss, A. Pironti, and P.-Y. Strub, "Implementing TLS with verified cryptographic security," in *Proceedings of the 34th IEEE Symposium on Security and Privacy*, 2013, pp. 445–459.

[41] Y. Suga, "SSL/TLS servers status survey about enabling forward secrecy," in *Proceedings of the 17th International Conference on Network-Based Information Systems*, 2014, pp. 501–505.

[42] L. S. Huang, S. Adhikarla, D. Boneh, and C. Jackson, "An experimental study of TLS forward secrecy deployments," *IEEE Internet Computing*, vol. 18, no. 6, pp. 43–51, 2014.

[43] K. T. Nguyen, M. Laurent, and N. Oualha, "Survey on secure communication protocols for the Internet of Things," *Ad Hoc Networks*, vol. 32, pp. 17–31, 2015.

[44] B. Mokhtar and M. Azab, "Survey on security issues in vehicular ad hoc networks," *Alexandria Engineering Journal*, vol. 54, no. 4, pp. 1115–1126, 2015.

[45] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2317–2346, 2015.

[46] A. Satapathy, J. Livingston *et al.*, "A Comprehensive Survey on SSL/TLS and their Vulnerabilities," *International Journal of Computer Applications*, vol. 153, no. 5, pp. 31–38, 2016.

[47] S. S. Manvi and S. Tangade, "A survey on authentication schemes in VANETs for secured communication," *Vehicular Communications*, vol. 9, pp. 19–30, 2017.

[48] C. Peng, H. Sun, M. Yang, and Y.-L. Wang, "A survey on security communication and control for smart grids under malicious cyber attacks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 8, pp. 1554–1569, 2019.

[49] R. Yugha and S. Chithra, "A survey on technologies and security protocols: Reference for future generation IoT," *Journal of Network and Computer Applications*, vol. 169, p. 102763, 2020.

[50] A. Kakkar, "A survey on secure communication techniques for 5G wireless heterogeneous networks," *Information Fusion*, vol. 62, pp. 89–109, 2020.

[51] R. Yugha and S. Chithra, "A survey on technologies and security protocols: Reference for future generation iot," *Journal of Network and Computer Applications*, vol. 169, p. 102763, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S108480452030237X

[52] J. Tournier, F. Lesueur, F. L. Mouël, L. Guyon, and H. Ben-Hassine, "A survey of iot protocols and their security issues through the lens

of a generic iot stack," *Internet of Things*, vol. 16, p. 100264, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660520300986

[53] B. Dowling, M. Fischlin, F. Günther, and D. Stebila, "A cryptographic analysis of the tls 1.3 handshake protocol," *Journal of Cryptology*, vol. 34, no. 4, p. 37, 2021.

[54] L. Minzhao, G. H. Habibi, and S. Vijay, "A Survey on DNS Encryption: Current Development, Malware Misuse, and Inference Techniques," *ACM Computing Surveys*, vol. 55, no. 8, pp. 1–28, 2022.

[55] X. de Carné de Carnavalet and P. C. van Oorschot, "A survey and analysis of tls interception mechanisms and motivations: Exploring how end-to-end tls is made "end-to-me" for web traffic," *ACM Computing Surveys*, vol. 55, no. 13s, pp. 1–40, 2023.

[56] R. Aissaoui, J.-C. Deneuville, C. Guerber, and A. Pirovano, "A survey on cryptographic methods to secure communications for uav traffic management," *Vehicular Communications*, p. 100661, 2023.

[57] M. S. Pour, C. Nader, K. Friday, and E. Bou-Harb, "A comprehensive survey of recent internet measurement techniques for cyber security," *Computers & Security*, vol. 128, p. 103123, 2023.

[58] N. Alnahawi, J. Müller, J. Oupickỳ, and A. Wiesmaier, "A comprehensive survey on post-quantum tls," *IACR Communications in Cryptology*, vol. 1, no. 2, 2024.

[59] M. K. Hasan, Z. Weichen, N. Safie, F. R. A. Ahmed, and T. M. Ghazal, "A survey on key agreement and authentication protocol for internet of things application," *IEEE Access*, 2024.

[60] M. Ozaif, S. Mustajab, and M. Alam, "Exploration of secured data transmission in internet of things: A survey," in *2024 IEEE International Conference on Computing, Power and Communication Technologies (IC2PCT)*, vol. 5. IEEE, 2024, pp. 106–112.

[61] Sheila Frankel, and Suresh Krishnan, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap," RFC, February 2011. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6071

[62] S. H. Gonde, C. Frisch, S. Duhovnikov, M. Kubisch, T. Meyerhoff, and D. Schupke, "Physical layer security in a private 5g network for industrial and mobility application," in *2023 IEEE 9th World Forum on Internet of Things (WF-IoT)*. IEEE, 2023, pp. 1–6.

[63] T. Assaf, A. Al-Dweik, Y. Iraqi, S. Jangsher, A. Pandey, J.-P. Giacalone, E. E. Abulibdeh, H. Saleh, and B. Mohammad, "High-rate secret key generation using physical layer security and physical unclonable functions," *IEEE Open Journal of the Communications Society*, vol. 4, pp. 209–225, 2023.

[64] Edgescan Research Team, "2022 Vulnerability Statistics Report," Accessed: 01/03/2025. [Online]. Available at https://www.edgescan.com/wp-content/uploads/2024/03/2022-Vulnerability-Statistic-Report.pdf.

[65] X. de Carné de Carnavalet and P. C. van Oorschot, "A survey and analysis of tls interception mechanisms and motivations: Exploring how end-to-end tls is made "end-to-me" for web traffic," *ACM Comput. Surv.*, vol. 55, no. 13s, jul 2023. [Online]. Available: https://doi.org/10.1145/3580522

[66] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," Network Working Group, RFC Editor, RFC 5246, August 2008, Accessed: 01/03/2025. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5246.txt

[67] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," Internet Engineering Task Force, RFC Editor, RFC 8446, August 2018, Accessed: 01/03/2025. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8446

[68] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[69] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, no. 177, pp. 203–209, 1987.

[70] W. Diffie and M. E. Hellman, "New directions in cryptography," in *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*, 2022, pp. 365–390.

[71] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta *et al.*, "Imperfect forward secrecy: How diffie-hellman fails in practice," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 5–17.

[72] G. S. Chudov and S. E. Leontiev, "GOST Cipher Suites for Transport Layer Security (TLS) Protocol Version 1.3," Internet Engineering Task Force, Internet-Draft, 2009, Accessed: 01/03/2025. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-chudov-cryptopro-cptls-04

[73] V. Popov, I. Kurepkin, S. Leontiev, G. Chudov, A. Afanasiev, N. Nikishin, B. Izotov, E. Minaeva, S. Murugov, I. Ovcharenko, I. Ustinov, and A. Erkin, "Additional Cryptographic Algorithms for Use with GOST 28147-89, GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms," Network Working Group, RFC Editor, RFC 4357, January 2006, Accessed: 01/03/2025. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc4357

[74] S. Smyshlyaev, E. Alekseev, E. Griboedova, A. Babueva, and L. Nikiforova, "GOST Cipher Suites for Transport Layer Security (TLS) Protocol Version 1.3," Internet-Draft, 2021, Accessed: 01/03/2025. [Online]. Available: https://www.ietf.org/archive/id/draft-smyshlyaev-tls13-gost-suites-05.html

[75] L. Bruckert, J. Merkle, and M. Lochter, "Elliptic Curve Cryptography (ECC) Brainpool Curves for Transport Layer Security (TLS) Version 1.3," Network Working Group, RFC 8734, February 2020, Accessed: 01/03/2025. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8734

[76] P. Yang, "ShangMi (SM) Cipher Suites for TLS 1.3," Network Working Group, RFC Editor, RFC 8998, March 2021, Accessed: 01/03/2025. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8998

[77] E. Rescorla, "HTTP Over TLS," RFC 2818, May 2000, Accessed: 01/03/2025. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc2818

[78] Saptak Sengupta, Liran Tal and Brian Clark, "Part2 Chapter14 Security," Web Almanac, Tech. Rep., 2022, Accessed: 01/03/2025. [Online]. Available: https://almanac.httparchive.org/en/2022/security#introduction

[79] T. Ylonen, "SSH–secure login connections over the Internet," in *Proceedings of the 6th USENIX Security Symposium*, 1996, pp. 40–52.

[80] M. Bellare, T. Kohno, and C. Namprempre, "Breaking and provably repairing the ssh authenticated encryption scheme: A case study of the encode-then-encrypt-and-mac paradigm," *ACM Transactions on Information and System Security*, vol. 7, no. 2, pp. 206–241, 2004.

[81] M. Abadi, "Explicit communication revisited: Two new attacks on authentication protocols," *IEEE Transactions on Software Engineering*, vol. 23, no. 3, pp. 185–186, 1997.

[82] T. Ylonen and C. Lonvick, "RFC 4253: The Secure Shell (SSH) transport layer protocol," *The Internet Society*, 2006.

[83] S. Lehtinen and C. Lonvick, "The Secure Shell (SSH) Protocol Assigned Numbers," SSH Communications Security Corp, RFC Editor, RFC 4250, January 2006, Accessed: 01/03/2025. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4250

[84] M. Friedl, N. Provos, and W. A. Simpson, "Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol," RFC Editor, RFC 4419, March 2006, Accessed: 01/03/2025. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4419

[85] L. Velvindron, "Deprecating RC4 in Secure Shell SSH," RFC Editor, RFC 8758, April 2020, Accessed: 01/03/2025. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8758.html

[86] J. J. G. Ortiz and K. J. Compton, "A simple power analysis attack on the twofish key schedule," *arXiv preprint arXiv:1611.07109*, 2016.

[87] R. Tandon, "A survey of distributed denial of service attacks and defenses," *arXiv preprint arXiv:2008.01345*, 2020.

[88] O. Shwartz, A. Cohen, A. Shabtai, and Y. Oren, "Shattered Trust: When Replacement Smartphone Components Attack." in *Proceedings of the 11th USENIX Workshop on Offensive Technologies*, 2017, pp. 1–13.

[89] K. Bhargavan and G. Leurent, "Transcript collision attacks: Breaking authentication in TLS, IKE, and SSH," in *Proceedings of the 25th Network and Distributed System Security Symposium*, 2016, pp. 1–18.

[90] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov, "The first collision for full SHA-1," in *Proceedings of the 37th Annual International Cryptology Conference*, 2017, pp. 570–596.

[91] K. Bhargavan, A. D. Lavaud, C. Fournet, A. Pironti, and P. Y. Strub, "Triple handshakes and cookie cutters: Breaking and fixing authentication over TLS," in *Proceedings of the 35th IEEE Symposium on Security and Privacy*, 2014, pp. 98–113.

[92] N. Mavrogiannopoulos, F. Vercauteren, V. Velichkov, and B. Preneel, "A cross-protocol attack on the TLS protocol," in *Proceedings of the 19th ACM Conference on Computer and Communications Security*, 2012, pp. 62–72.

[93] D. Wagner and B. Schneier, "Analysis of the ssl 3.0 protocol," in *The Second USENIX Workshop on Electronic Commerce Proceedings*, vol. 1, no. 1, 1996, pp. 29–40.

[94] M. Brinkmann, C. Dresen, R. Merget, D. Poddebniak, J. Müller, J. Somorovsky, J. Schwenk, and S. Schinzel, "ALPACA: Application layer protocol confusion-analyzing and mitigating cracks in TLS authentication," in *Proceedings of the 30th USENIX Security Symposium*, 2021, pp. 4293–4310.

[95] R. Merget, M. Brinkmann, N. Aviram, J. Somorovsky, J. Mittmann, and J. Schwenk, "Raccoon Attack: Finding and Exploiting Most-Significant-Bit-Oracles in TLS-DH(E)," in *Proceedings of the 30th USENIX Security Symposium*, 2021, pp. 213–230.

[96] B. B. Brumley and N. Tuveri, "Remote timing attacks are still practical," in *European Symposium on Research in Computer Security*. Springer, 2011, pp. 355–371.

[97] Y. Yarom, D. Genkin, and N. Heninger, "Cachebleed: a timing attack on openssl constant-time rsa," *Journal of Cryptographic Engineering*, vol. 7, pp. 99–112, 2017.

[98] D. A. Osvik, A. Shamir, and E. Tromer, "Cache attacks and countermeasures: the case of aes," in *Topics in Cryptology–CT-RSA 2006: The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2005. Proceedings*. Springer, 2006, pp. 1–20.

[99] D. Bleichenbacher, "Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS# 1," in *Proceedings of the 18th Annual International Cryptology Conference*, 1998, pp. 1–12.

[100] H. Böck, J. Somorovsky, and C. Young, "Return Of Bleichenbacher's Oracle Threat (ROBOT)," in *Proceedings of the 27th USENIX Security Symposium*, 2018, pp. 817–849.

[101] E. Ronen, R. Gillham, D. Genkin, A. Shamir, D. Wong, and Y. Yarom, "The 9 lives of Bleichenbacher's CAT: New cache attacks on TLS implementations," in *Proceedings of the 40th IEEE Symposium on Security and Privacy*, 2019, pp. 435–452.

[102] S. Fan, W. Wang, and Q. Cheng, "Attacking openssl implementation of ecdsa with a few signatures," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1505–1515.

[103] F. Weimer, "Factoring RSA keys with TLS perfect forward secrecy," *Red Hat Technical Report*, 2015.

[104] D. de Almeida Braga, P.-A. Fouque, and M. Sabt, "PARASITE: PAssword Recovery Attack against Srp Implementations in ThE wild," in *Proceedings of the 28th ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 2497–2512.

[105] A. Russon, "Threat for the Secure Remote Password Protocol and a Leak in Apple's Cryptographic Library," in *Proceedings of the 19th International Conference on Applied Cryptography and Network Security*, 2021, pp. 49–75.

[106] D. Bleichenbacher, "Breaking a Cryptographic Protocol with Pseudo-primes," in *Proceedings of the 8th International Workshop on Theory and Practice in Public Key Cryptography*, 2005, pp. 9–15.

[107] B. Möller, T. Duong, and K. Kotowicz, "This POODLE bites: exploiting the SSL 3.0 fallback," *Security Advisory*, vol. 21, pp. 34–58, 2014.

[108] C. Young, "TLS CBC Padding Oracles in 2019," Web Report, 2019, Accessed on 01/03/2025. [Online]. Available: https://www.tripwire.com/state-of-security/tls-cbc-padding-oracles/

[109] C. Young, "What is Zombie POODLE?" Web Report, 2019, Accessed on 01/03/2025. [Online]. Available: https://www.tripwire.com/state-of-security/zombie-poodle

[110] C. Young, "Zombie POODLE, GOLDENDOODLE & How TLSv1.3 Can Save Us All," *Black Hat Aisa*, 2019. [Online]. Available: https://i.blackhat.com/asia-19/Fri-March-29/bh-asia-Young-Zombie-Poodle-Goldendoodle-and-How-TLSv13-Can-Save-Us-All.pdf

[111] R. Bardou, R. Focardi, Y. Kawamoto, L. Simionato, G. Steel, and J.-K. Tsay, "Efficient padding oracle attacks on cryptographic hardware," in *Proceedings of the 32nd Annual Cryptology Conference*, 2012, pp. 608–625.

[112] D. Bleichenbacher, "Chosen ciphertext attacks against protocols based on the rsa encryption standard pkcs# 1," in *Proceedings of the 18th Annual International Cryptology Conference*, 1998, pp. 1–12.

[113] T. Duong and J. Rizzo, "Here come the ninjas," *Unpublished manuscript*, vol. 320, 2011.

[114] M. R. Albrecht and K. G. Paterson, "Lucky microseconds: A timing attack on amazon's s2n implementation of TLS," in *Proceedings of the 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2016, pp. 622–643.

[115] G. Irazoqui, M. S. Inci, T. Eisenbarth, and B. Sunar, "Lucky 13 strikes back," in *Proceedings of the 10th ACM symposium on information, computer and communications security*, 2015, pp. 85–96.

[116] N. Sullivan, "Do the chacha: better mobile performance with cryptography," Accessed: 01/03/2025. [Online]. Available at https://blog.cloudflare.com/do-the-chacha-better-mobile-performance-with-cryptography.

[117] M. Seddigh and H. Soleimany, "Cross-VM cache attacks on Camellia," *Journal of Computer Virology and Hacking Techniques*, vol. 18, no. 2, pp. 91–99, 2022.

[118] S. Milad and S. Hadi, "Flush+ Reload Attacks on SEED," *The Computer Journal*, vol. 65, no. 10, pp. 2769–2779, 2022.

[119] O. Dunkelman, N. Keller, E. Ronen, and A. Shamir, "The retracing boomerang attack," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2020, pp. 280–309.

[120] M. Coutinho and T. C. Souza Neto, "Improved linear approximations to arx ciphers and attacks against chacha," in *Advances in Cryptology–EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part I 40*. Springer, 2021, pp. 711–740.

[121] A. Bariant and G. Leurent, "Truncated boomerang attacks and application to aes-based ciphers," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2023, pp. 3–35.

[122] M. Fischlin, F. Günther, and C. Janson, "Robust channels: Handling unreliable networks in the record layers of quic and dtls 1.3," *Journal of Cryptology*, vol. 37, no. 2, p. 9, 2024.

[123] M. Itsik, "Bar-Mitzva Attack: Breaking SSL with 13-Year Old RC4 Weakness," *Black Hat Asia*, 2015.

[124] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of RC4," in *Proceedings of the 12th International Conference on Selected Areas in Cryptography*, 2001, pp. 1–24.

[125] I. Mantin and A. Shamir, "A practical attack on broadcast RC4," in *Proceedings of the 8th International Workshop on Fast Software Encryption*, 2001, pp. 152–164.

[126] N. AlFardan, D. J. Bernstein, K. G. Paterson, B. Poettering, and J. C. Schuldt, "On the Security of RC4 in TLS," in *Proceedings of the 22nd USENIX Security Symposium*, 2013, pp. 305–320.

[127] A. Razaghpanah, A. A. Niaki, N. Vallina-Rodriguez, S. Sundaresan, J. Amann, and P. Gill, "Studying TLS usage in Android apps," in *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*, 2017, pp. 350–362.

[128] D. Warburton, "The 2021 TLS Telemetry Report," Accessed: 01/03/2025. [Online]. Available: https://www.f5.com/labs/articles/threat-intelligence/the-2021-tls-telemetry-report

[129] D. Gruss, C. Maurice, K. Wagner, and S. Mangard, "Flush+flush: A fast and stealthy cache attack," 2016.

[130] IBM, "POWER family and PowerPC architecture overview," *IBM Documentation*, 2023. [Online]. Available: https://www.ibm.com/docs/en/aix/7.1?topic=storage-power-family-powerpc-architecture-overview

[131] A. R. Choudhuri and S. Maitra, "Significantly improved multi-bit differentials for reduced round salsa and chacha," *IACR Transactions on Symmetric Cryptology*, pp. 261–287, 2016.

[132] O. Mutlu and J. S. Kim, "Rowhammer: A retrospective," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 8, pp. 1555–1571, 2019.

[133] M. Kaur, M. Raj, and H.-N. Lee, "Cross channel scripting and code injection attacks on web and cloud-based applications: a comprehensive review," *Sensors*, vol. 22, no. 5, p. 1959, 2022.

[134] N. Tripathi and N. Hubballi, "Application layer denial-of-service attacks and defense mechanisms: a survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 4, pp. 1–33, 2021.

[135] Z. Alkhalil, C. Hewage, L. Nawaf, and I. Khan, "Phishing attacks: A recent comprehensive study and a new anatomy," *Frontiers in Computer Science*, vol. 3, p. 563060, 2021.

[136] D. X. Song, D. Wagner, and X. Tian, "Timing Analysis of Keystrokes and Timing Attacks on SSH," in *Proceedings of the 10th USENIX Security Symposium*, 2001, pp. 1–16.

[137] M. Vanhoef, "Fragment and Forge: Breaking Wi-Fi Through Frame Aggregation and Fragmentation," in *Proceedings of the 30th USENIX Security Symposium*, August 2021, pp. 161–178.

[138] M. Vanhoef and F. Piessens, "Key reinstallation attacks: Forcing nonce reuse in WPA2," in *Proceedings of the 24th ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1313–1328.

[139] V. Kananda, "Why You Should Use AES 256 Encryption to Secure Your Data," Accessed: 01/03/2025. [Online]. Available: https://www.ipswitch.com/blog/use-aes-256-encryption-secure-data

[140] J. Rizzo and T. Duong, "The CRIME Attack," Accessed: 01/03/2025. [Online]. Available: https://docs.google.com/presentation/d/11eBmGiHbYcHR9gL5nDyZChu_-lCa2GizeuOfaLU2HOU/edit#slide=id.g1d134dff_1_222

[141] J. Kelsey, "Compression and information leakage of plaintext," in *Proceedings of the 9th International Workshop on Fast Software Encryption*, 2002, pp. 263–276.

[142] G. Vranken, "HTTPS bicycle attack," Tech Report, 2015, Accessed: 01/03/2025. [Online]. Available: http://docs.alexomar.com/biblioteca/https-bicycle-attack.pdf

[143] B. Harsha, R. Morton, J. Blocki, J. Springer, and M. Dark, "Bicycle attacks considered harmful: Quantifying the damage of widespread password length leakage," *Computers & Security*, vol. 100, p. 102068, 2021.

[144] Y. Gluck, N. Harris, and A. Prado, "BREACH: reviving the CRIME attack," *Unpublished manuscript*, 2013.

[145] M. Vanhoef and T. Van Goethem, "HEIST: HTTP Encrypted Information can be Stolen through TCP-windows," in *The 19th Black Hat US Briefings*, 2016.

[146] E. S. Alashwali, P. Szalachowski, and A. Martin, "Exploring HTTPS security inconsistencies: A cross-regional perspective," *Computers & security*, vol. 97, p. 101975, 2020.

[147] A. Delignat-Lavaud and K. Bhargavan, "Network-based origin confusion attacks against HTTPS virtual hosting," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 227–237.

[148] PortSwigger, "HTTP request smuggling," Accessed: 01/03/2025. [Online]. Available: https://portswigger.net/web-security/request-smuggling

[149] J. Blatz, "Csrf: Attack and defense," *McAfee® Foundstone® Professional Services, White Paper*, 2007.

[150] B. Jabiyev, S. Sprecher, K. Onarlioglu, and E. Kirda, "T-reqs: HTTP request smuggling with differential fuzzing," in *Proceedings of the 28th ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 1805–1820.

[151] A. Klein, "HTTP request smuggling in 2020–new variants, new defenses and new challenges," *Black Hat Briefings USA*, 2020.

[152] E. Arshad, M. Benolli, and B. Crispo, "Practical attacks on Login CSRF in OAuth," *Computers & Security*, vol. 121, p. 102859, 2022.

[153] J. Owens and J. Matthews, "A study of passwords and methods used in brute-force SSH attacks," in *Proceedings of the 1st USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2008, pp. 1–8.

[154] "Hydra Tool Documentation," Kali Library Document, Accessed: 01/03/2025. [Online]. Available: https://www.kali.org/tools/hydra/

[155] "Ncrack Tool Documentation," Kali Library Document, Accessed: 01/03/2025. [Online]. Available: https://www.kali.org/tools/ncrack/

[156] "Patator Tool Documentation," Kali Library Document, Accessed: 01/03/2025. [Online]. Available: https://www.kali.org/tools/patator/

[157] G. Yen, "IoT_reaper: A Rappid Spreading New IoT Botnet," Accessed: 01/03/2025. [Online]. Available: https://blog.netlab.360.com/iot_reaper-a-rappid-spreading-new-iot-botnet-en/

[158] Trend Micro, "BrickerBot Malware Emerges, Permanently Bricks IoT Devices," Accessed: 01/03/2025. [Online]. Available: https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/brickerbot-malware-permanently-bricks-iot-devices

[159] J. R. Shane Young, "BruteSpray github repository," Accessed: 01/03/20253. [Online]. Available: https://github.com/x90skysn3k/brutespray

[160] M. A. Hogye, C. T. Hughes, J. M. Sarfaty, and J. D. Wolf, "Analysis of the feasibility of keystroke timing attacks over SSH connections," *Research Project at University of Virginia*, 2001.

[161] J. Jin and X. Wang, "On the effectiveness of low latency anonymous network in the presence of timing attack," in *Proceedings of the 39th IEEE/IFIP International Conference on Dependable Systems & Networks*, 2009, pp. 429–438.

[162] A. Aviram, S. Hu, B. Ford, and R. Gummadi, "Determinating timing channels in compute clouds," in *Proceedings of the ACM Cloud Computing Security Workshop*, 2010, pp. 103–108.

[163] M. R. Albrecht, K. G. Paterson, and G. J. Watson, "Plaintext recovery attacks against SSH," in *Proceedings of the 30th IEEE Symposium on Security and Privacy*, 2009, pp. 16–26.

[164] T. B. Hansen, "Cryptographic Security of SSH Encryption Schemes," Ph.D. dissertation, Royal Holloway, University of London, 2020.

[165] A. Boldyreva, J. P. Degabriele, K. G. Paterson, and M. Stam, "Security of symmetric encryption in the presence of ciphertext fragmentation," in *Proceedings of the 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2012, pp. 682–699.

[166] M. R. Albrecht, J. P. Degabriele, T. B. Hansen, and K. G. Paterson, "A surfeit of SSH cipher suites," in *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1480–1491.

[167] M. Abdalla and M. Bellare, "Increasing the lifetime of a key: a comparative analysis of the security of re-keying techniques," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2000, pp. 546–559.

[168] V. Mavroeidis, K. Vishi, M. D. Zych, and A. Jøsang, "The impact of quantum computing on present cryptography," *arXiv preprint arXiv:1804.00200*, 2018.

[169] D. J. Bernstein, "Grover vs. mceliece," in *Post-Quantum Cryptography: Third International Workshop, PQCrypto 2010, Darmstadt, Germany, May 25-28, 2010. Proceedings 3*. Springer, 2010, pp. 73–80.

[170] R. A. Perlner and D. A. Cooper, "Quantum resistant public key cryptography: a survey," in *Proceedings of the 8th Symposium on Identity and Trust on the Internet*, 2009, pp. 85–93.

[171] J. K. Cheng, E. M. Lim, Y. Y. Krikorian, D. J. Sklar, and V. J. Kong, "A survey of encryption standard and potential impact due to quantum computing," in *2021 IEEE Aerospace Conference (50100)*. IEEE, 2021, pp. 1–10.

[172] P. Schwabe, D. Stebila, and T. Wiggers, "Post-quantum tls without handshake signatures," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1461–1480.

[173] P. Schwabe, D. Stebila, and T. Wiggers, "More efficient post-quantum kemtls with pre-distributed public keys," in *Computer Security–ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part I 26*. Springer, 2021, pp. 3–22.

[174] C. Cremers, A. Dax, and N. Medinger, "Keeping up with the kems: Stronger security notions for kems and automated analysis of kem-based protocols," *Cryptology ePrint Archive*, 2023.

[175] V. Kolachana, D. Upmandewan, A. Giri, N. Pavan, A. Ahmed, M. Thippeswamy, and T. Vinay, "Application of quantum algorithms for network protocols," in *Proceedings of Third International Conference on Communication, Computing and Electronics Systems: ICCCES 2021*. Springer, 2022, pp. 439–461.

[176] N. Alnahawi, J. Müller, J. Oupický, and A. Wiesmaier, "SoK: Post-quantum TLS handshake," Cryptology ePrint Archive, Paper 2023/1873, 2023, https://eprint.iacr.org/2023/1873. [Online]. Available: https://eprint.iacr.org/2023/1873

[177] "Fifth PQC Standardization Conference," National Institute of Standards and Technology, Accessed: 01/03/2025. [Online]. Available: https://csrc.nist.gov/Events/2024/fifth-pqc-standardization-conference

[178] "Post-Quantum Cryptography," IEFT Community, Accessed: 01/03/2025. [Online]. Available: https://wiki.ietf.org/group/sec/PQCAgility

[179] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009, pp. 169–178.

[180] P. Martins, L. Sousa, and A. Mariano, "A survey on fully homomorphic encryption: An engineering perspective," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, pp. 1–33, 2017.

[181] C. Marcolla, V. Sucasas, M. Manzano, R. Bassoli, F. H. Fitzek, and N. Aaraj, "Survey on fully homomorphic encryption, theory, and applications," *Proceedings of the IEEE*, vol. 110, no. 10, pp. 1572–1609, 2022.

[182] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in Cryptology: Proceedings of CRYPTO 84 4*. Springer, 1985, pp. 47–53.

[183] Y. Zhang, R. H. Deng, S. Xu, J. Sun, Q. Li, and D. Zheng, "Attribute-based encryption for cloud computing access control: A survey," *ACM Computing Surveys (CSUR)*, vol. 53, no. 4, pp. 1–41, 2020.

[184] C. Ge, W. Susilo, J. Baek, Z. Liu, J. Xia, and L. Fang, "Revocable attribute-based encryption with data integrity in clouds," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 5, pp. 2864–2872, 2021.

[185] F. Akhter and M. U. Rahman, "Linear secret sharing scheme for attribute encryption."

[186] H. Oz, A. Aris, A. Levi, and A. S. Uluagac, "A survey on ransomware: Evolution, taxonomy, and defense solutions," *ACM Computing Surveys (CSUR)*, vol. 54, no. 11s, pp. 1–37, 2022.

[187] B. B. Rad, M. Masrom, and S. Ibrahim, "Camouflage in malware: from encryption to metamorphism," *International Journal of Computer Science and Network Security*, vol. 12, no. 8, pp. 74–83, 2012.

[188] I. You and K. Yim, "Malware obfuscation techniques: A brief survey," in *2010 International conference on broadband, wireless computing, communication and applications*. IEEE, 2010, pp. 297–300.

[189] Cybersecurity & Infrastructure Security Agency, "Update: Destructive Malware Targeting Organizations in Ukraine," Last Accessed: 01/03/2025. [Online]. Available at https://www.cisa.gov/news-events/cybersecurity-advisories/aa22-057a.

[190] M. N. Alenezi, H. Alabdulrazzaq, A. A. Alshaher, and M. M. Alkharang, "Evolution of malware threats and techniques: A review," *International journal of communication networks and information security*, vol. 12, no. 3, pp. 326–337, 2020.
[191] A. Qamar, A. Karim, and V. Chang, "Mobile malware attacks: Review, taxonomy & future directions," *Future Generation Computer Systems*, vol. 97, pp. 887–909, 2019.
[192] R. Lyda and J. Hamrock, "Using entropy analysis to find encrypted and packed malware," *IEEE Security & Privacy*, vol. 5, no. 2, pp. 40–45, 2007.
[193] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh, "A survey on heuristic malware detection techniques," in *The 5th Conference on Information and Knowledge Technology*. IEEE, 2013, pp. 113–120.
[194] B. Anderson and D. McGrew, "Identifying encrypted malware traffic with contextual flow data," in *Proceedings of the 2016 ACM workshop on artificial intelligence and security*, 2016, pp. 35–46.
[195] D. Bekerman, B. Shapira, L. Rokach, and A. Bar, "Unknown malware detection using network traffic classification," in *2015 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2015, pp. 134–142.
[196] T.-F. Yen and M. K. Reiter, "Traffic aggregation for malware detection," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2008, pp. 207–227.
[197] D. Ucci, L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis," *Computers & Security*, vol. 81, pp. 123–147, 2019.
[198] M. Shen, K. Ye, X. Liu, L. Zhu, J. Kang, S. Yu, Q. Li, and K. Xu, "Machine learning-powered encrypted network traffic analysis: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, 2022.
[199] Z. Okonkwo, E. Foo, Q. Li, and Z. Hou, "A cnn based encrypted network traffic classifier," in *Proceedings of the 2022 Australasian Computer Science Week*, 2022, pp. 74–83.
[200] H. Kashyap, A. R. Pais, and C. Kondaiah, "Machine learning-based malware detection and classification in encrypted tls traffic," in *International Conference on Security, Privacy and Data Analytics*. Springer, 2022, pp. 247–262.
[201] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
[202] A. Brack, E. Entrup, M. Stamatakis, P. Buschermöhle, A. Hoppe, and R. Ewerth, "Sequential sentence classification in research papers using cross-domain multi-task learning," *International Journal on Digital Libraries*, pp. 1–24, 2024.
[203] T. Liu, X. Ma, L. Liu, X. Liu, Y. Zhao, N. Hu, and K. Z. Ghafoor, "Lambert: Leveraging attention mechanisms to improve the bert fine-tuning model for encrypted traffic classification," *Mathematics*, vol. 12, no. 11, p. 1624, 2024.
[204] I. H. Ji, J. H. Lee, M. J. Kang, W. J. Park, S. H. Jeon, and J. T. Seo, "Artificial intelligence-based anomaly detection technology over encrypted traffic: a systematic literature review," *Sensors*, vol. 24, no. 3, p. 898, 2024.
[205] X. Zang, T. Wang, X. Zhang, J. Gong, P. Gao, and G. Zhang, "Encrypted malicious traffic detection based on natural language processing and deep learning," *Computer Networks*, p. 110598, 2024.

Jemin Ahn received his B.S. degree in Computer Science and Engineering in 2017 from Hanyang University, Ansan, Republic of Korea, and now he is currently pursuing the Ph.D. degree with Computer Science and Engineering from Hanyang University, Seoul, Republic of Korea.

He is interested in advancing the field of cybersecurity through the application of deep learning techniques. His research focuses on practical approaches for detecting and preventing cyber threats, with recent efforts centered on developing attack detection methods using natural language processing models.

Rasheed Hussain (Senior member, IEEE) received his B.S. Engineering degree in Computer Software Engineering from University of Engineering and Technology, Peshawar, Pakistan in 2007, MS and Ph.D. degrees in Computer Science and Engineering from Hanyang University, South Korea in 2010 and 2015, respectively. He worked as a Postdoctoral Fellow at Hanyang University, South Korea from March 2015 to August 2015. He also worked as a guest researcher and consultant at the University of Amsterdam (UvA), The Netherlands from September 2015 to May 2016 and as an Assistant Professor at Innopolis University, Innopolis, Russia from June 2016 until December 2018. He also served as an Associate Professor and the Director of the Institute of Information Security and Cyber-Physical Systems and the head of the Networks and Blockchain Lab at Innopolis University, Russia from December 2018 till December 2021. He served as an ACM Distinguished Speaker (2018-2021) and currently serving another term as ACM Distinguished Speaker (2022-2026). Currently, he is a Senior Lecturer with the Smart Internet Lab and Bristol Digital Futures Institute (BDFI) at the University of Bristol, UK. To date, he has delivered around 20 invited and keynote talks (including ACM DSP lectures) and serves as editorial board member in many journals including IEEE Communications Surveys & Tutorials. He served as a co-chair and organizing committee member as well as technical committee member of many renowned conferences including IEEE ICC, CNS, Globecom, FNWF, VTC, VNC, and more.

His research interests include Information, network, and cyber security, Future Networks (6G) security, Digital Twins security, AI security, Responsible AI, Explainable AI, and Fairness in AI.

Kyungtae Kang received his B.S. degree in Computer Science and Engineering in 1999, and his M.S. and Ph.D. degrees in Electrical Engineering and Computer Science in 2001 and 2007, respectively, from Seoul National University, Seoul, Korea. From 2008 to 2010, he was a postdoctoral research associate at the University of Illinois at Urbana-Champaign, IL, USA. In 2011, he joined the Department of Computer Science and Engineering at Hanyang University, Korea and is currently a tenured professor in the Department of Artificial Intelligence.

His research interests primarily focus on systems, including operating systems, mobile systems, distributed systems, and real-time embedded systems. His recent work delves into the interdisciplinary field of cyber-physical systems.

Junggab Son (SMIEEE'21) received the BSE degree in computer science and engineering from Hanyang University, Ansan, South Korea (2009), and the Ph.D. degree in computer science and engineering from Hanyang University, Seoul, South Korea (2014). From 2014 to 2016, he was a Postdoctoral Research Associate with the Department of Math and Physics, North Carolina Central University. From 2016 to 2018, he was a Research Fellow and a Limited-term Assistant Professor at Kennesaw State University. From 2018 to 2022, he was an Assistant Professor of Computer Science at Kennesaw State University. Currently, he is an Assistant Professor of Computer Science and a Director of Information and Intelligent Security (IIS) Laboratory at University of Nevada, Las Vegas. His research interests include applied cryptography, privacy preservation, blockchain and smart contract, AI-powered cybersecurity solutions, and security/privacy issues in artificial intelligent algorithms.