Lecture Note 4: Network Security Principles

LIU Yanting 22037667R
JIANG Jinan 22116936R
ZHU Shuntao 22068136R

March 24, 2024

In this lecture note, we discuss the mechanisms related to Network Security Principles, including authenticated key exchange, public key infrastructure (PKI), and certification authorities. These concepts play a central role that underlie the security of the field of network security, which enables the secure communication on the Internet.

1 Authenticated Key Exchange

Before we start, we should declare the goal of authentication and authenticated key exchange. According to Diffie [DVOW92], "the goal of an authentication protocol is to provide the communicating parties with some assurance that they know each other's true identities. In an authenticated key exchange, there is the additional goal that the two parties end up sharing a common key known only to them."

1.1 Diffie-Hellman Key Exchange

Among all the AKE algorithms, the Diffie-Hellman Key Exchange [DH22] is one of the first. The core part of this encryption is the public key pair p and g. Suppose there are two clients A and B using private keys a and b respectively, for the basic model, A sends B g^a mod p and B ends A g^b mod p, and the shared key is g^{ab} mod p.

Under these procedures, A knows a and g^b mod p, which means A can compute g^{ab} mod p easily. B knows b and g^a mod p, which means B can also compute g^{ab} mod p easily. For the eavesdropper, it only knows g, p, g^a mod p, and g^a mod p, which means it needs to solve discrete logarithm problems. For the Brute-Force algorithm, its time complexity should be O(p), for the baby-step giant-step algorithm, its time complexity should be $O(\sqrt{p})$. Therefore, for security, the selection of p should be extremely large.

However, considering the basic Diffie-Hellman Key Exchange algorithm has no authentication section, both A and B cannot authenticate each other, which means it is insecure under man-in-the-middle attack. The basic settings are the same as above, while an eavesdropper intercepts all information between A and B. The eavesdropper has two fake private keys a_f and b_f, when receiving g^a mod p from A, it sends g^{b_f} mod p to A. And send g^{a_f} mod p to B when receiving g^b mod p from B. Now the eavesdropper has shared key g^{ab_f} mod p

with A, and shared key $g^{a_f b}$ mod p with B. Without notifying A or B, the eavesdropper can monitor the whole communication session between A and B.

1.2 Authenticated Key Exchange

To avoid the circumstances mentioned above, we need authentication during the key exchange procedure. The generation and distribution of certifications corresponding to each user are handled by the Trusted Third Party (TTP). The lecture has introduced two basic schemes of AKE, two-way/mutual authentication and one-sided AKE. The difference is that mutant authentication requires both clients to transmit their certifications to each other and verify the received certification, while the one-sided AKE only requires one client (suppose is client A) to send its certificate to B and B to verify the identity of A.

1.2.1 Transport Layer Security

The Transport layer security (TLS) is currently widely used in the World Wide Web, its first version TLS 1.0 was developed through the standardized Secure Sockets Layer (SSL) in 1995 [CHH+17]. According to section 1.1 (DHE), the security of DHE is dominant by the value of p, and it is soluble by quantum computer using Shor's Algorithm [Sho94], and somebody may crack it using a modern computer in a long period but limited time, which means the protocol #1 shown in lecture slides has no Forward secrecy. The protocol #2 adds an Advanced Encryption Standard (AES) mechanism to ensure forward secrecy.

Compared with previous versions, TLS 1.3 achieves significant improvements. One of the great improvements is the support of 1-RTT and 0-RTT handshake protocol. The TLS 1.3 has 3 key exchange modes, namely, Diffie–Hellman exchange (DHE), pre-shared key (PSK) exchange, and PSK coupled with DHE, the following figures are examples [CHH+17]:

ClientHello: Contains a random nonce and a list of symmetric algorithms.

ServerHello: Contains a server-generated random nonce, an indication of the selected parameters, and potentially some other extensions.

KeyShare: Contains a set of DH key shares and the associated groups.

EncryptedExtensions: Message contains material unnecessary for determining cryptographic parameters.

CertificateRequest: Indicates that the server requests client authentication in the mutual authentication case.

Certificate: Contains the certificate of the client/server.

CertificateVerify: A digital signature over the current transcript.

PSK: The pre-shared key issued by the server after the first successful handshake with the client.

The second one is PSK exchange, which only costs 1-RTT. After a successful handshake between the client and server, the server will assign a New Session Ticket (NST) to the client which is adopted by the client as PSKs for subsequent handshakes. Figure 2 is an example.

The TLS 1.3 also provides 0-RTT handshake as shown in Figure 3. The client uses PSK to send ClientHello with application data. However, this data is not protected against

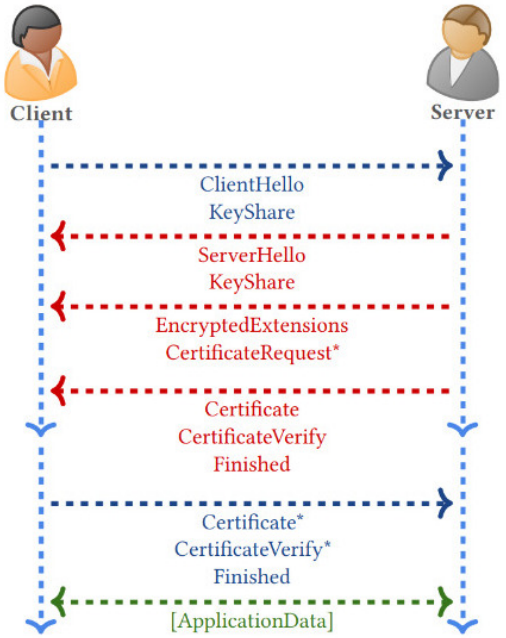

Figure 1: A full TLS 1.3 handshake (Section 2.1.1)

Figure 1: [CHH+17]

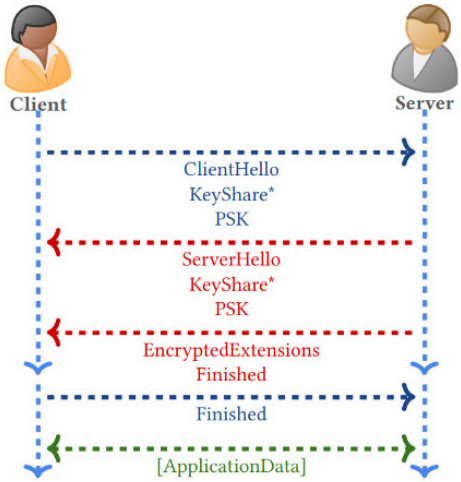

Figure 2: A PSK resumption handshake (Section 2.1.7)

Figure 2: [CHH+17]

replay attacks as mentioned by TLS 1.3 draft specification [CHH+17]. Therefore, this 0-RTT mechanism is only available for those entities equipped with replay protection at the application layer.

2 Public Key Infrastructure (PKI)

2.1 Motivations of the PKI

We begin with explaining the intuition behind the introduction of the Public Key Infrastructure (PKI) mechanism. Imagine this scenario, where we are trying to contact Bob in an encrypted channel via public key encryption, a distant friend from another country. In particular, we are sending some sensitive information $INFO_{sens}$. Here comes a critical question: how do we know Bob's public key? We might just look it up from the Internet, and Bob might have established a personal website on which he posts his public key. However, there is a question that how can we trust the website? How do we know it indeed came from Bob? Can anyone vouch for that?

As we can see from the above dilemma, there is an apparent issue of a lack of trust anchor. We have to rely on something to trust Bob's public key and thus to encrypt our sensitive information $INFO_{sens}$ to him. This motivates the design of the PKI, where we incorporate some form of a trust anchor into the public key identification system.

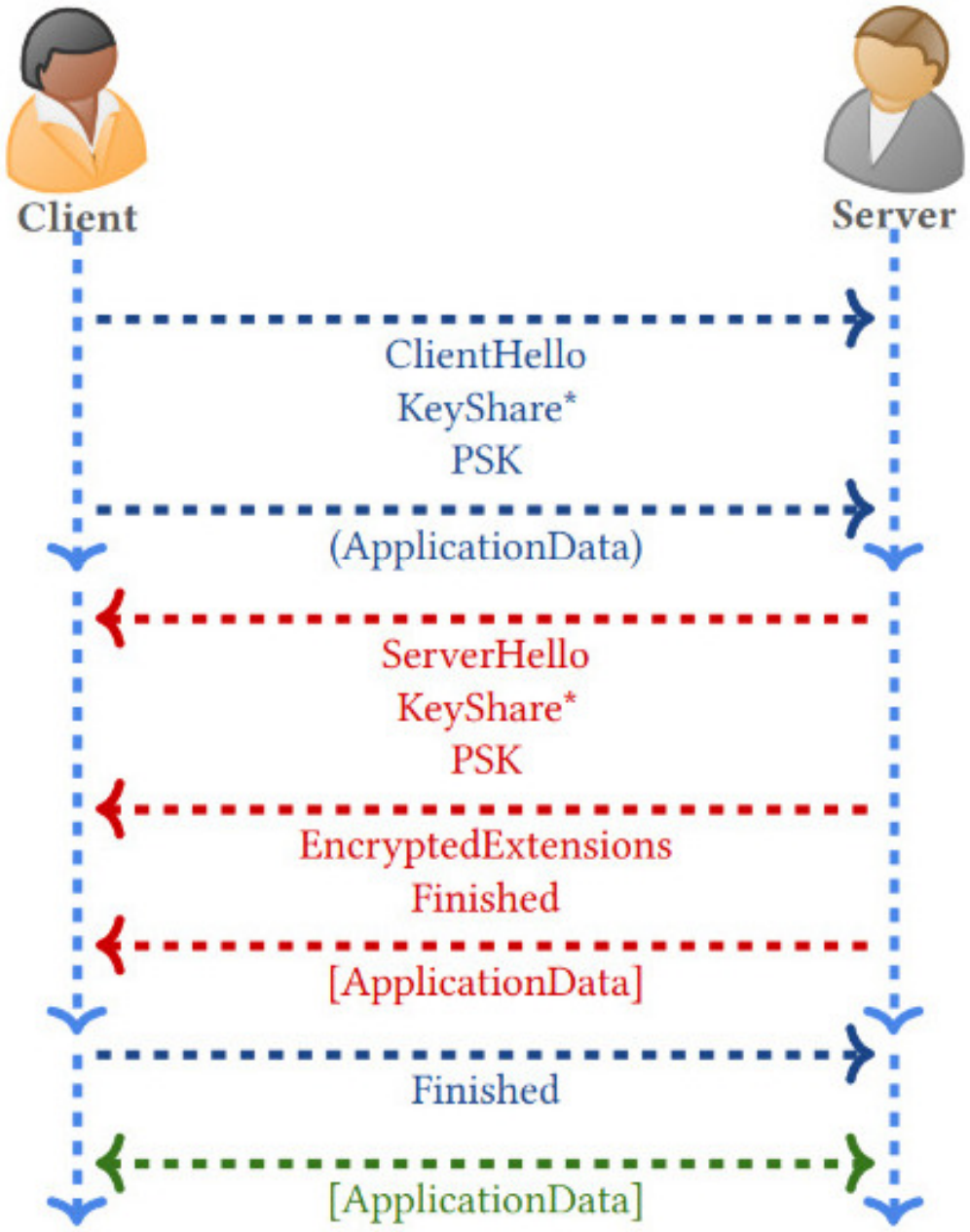

Figure 3: A 0-RTT handshake (Section 2.1.8)

Figure 3: [CHH+17]

2.2 The design of the PKI

As discussed above, a trust anchor is needed to ensure that there exists a source of trust in order for us to believe that a public key indeed belongs to Bob and therefore we could send some sensitive information $INFO_{sens}$ to him.

To accomplish this, the Public Key Infrastructure (PKI) structure has been devised. In particular, an illustration is shown in Figure 4. In this structure, we centralize our trust on a trusted third party (TTP), which is responsible for delegating digital signatures vouching for other people's public keys, possibly via some form of a hierarchy. By doing so, we implicitly trust a chain of players who delegate trust to downstream parties. This concept is best illustrated via the following example.

2.3 Example

As an example to illustrate the PKI infrastructure, consider the case of a poly class, where a student, Alice, is trying to convince the teacher, Haiyang, that his public key is 0x123.

1. **Trusted Third Party:** First, we establish the trust anchor. In this example, we assume that all parties trust the PolyU university. In this case, we consider the PolyU as a trusted third party, where both the student and the teacher trust what the university tells them. The public key $PUBKEY_{polyu}$ of the PolyU is already known and trusted by everyone on the campus, and only the PolyU authority has the corresponding private key $PRIVKEY_{polyu}$.

2. **Delegating to the COMP department:** Next, the PolyU delegates trust to lower levels. In particular, we assume that the PolyU uses its own private key $PRIVKEY_{polyu}$ to sign a digital certificate saying that the public key of the COMP department is $PUBKEY_{comp}$. Then, this effectively means that anyone that obtains a copy of the digital signature of such a form can verify its authenticity using the public key of PolyU, i.e., $PUBKEY_{polyu}$, to verify that this signature is valid. Since everyone in PolyU know and trust the public key of PolyU, they therefore trust that the public key $PUBKEY_{comp}$ of the COMP department is indeed what was vouched for in the digital signature. This is an effective example of the trust anchor: we trust PolyU, and if it tells us that we should trust COMP, then we just trust COMP.

3. **Delegating to the office:** As a next step, the COMP department, which via the chain of trust that we already trust, delegates the trust to the office that Alice works at (let's say it is office 123). This is done by signing a digital signature for the public key of the office 123, using the private key of the COMP department. Since we assume that only the COMP department has the private key, then if that signature checks out by verifying it against the public key of the COMP department, then that effectively means that it is indeed the COMP department that is providing proof for the validness of the public key of the office 123.

4. **Trusting Alice:** Finally, in order for the teacher to trust Alice's public key, the teacher has a final step to take: checking that the digital signature from office 123,

which vouches for the public key of Alice, is valid. This is conveniently done by checking against the public key of the office 123, which we should have already established via the trust chain all the way from the PolyU, to the COMP department, and then eventually to the office 123. Overall, if the signature for Alice's public key checks out, the teacher, Mr. Haiyang, could safely trust Alice's public key. This is all conveniently done by simply trusting the public key of a central trust anchor, i.e., PolyU in our example, and then delegating trust all the way through the chain of trust.

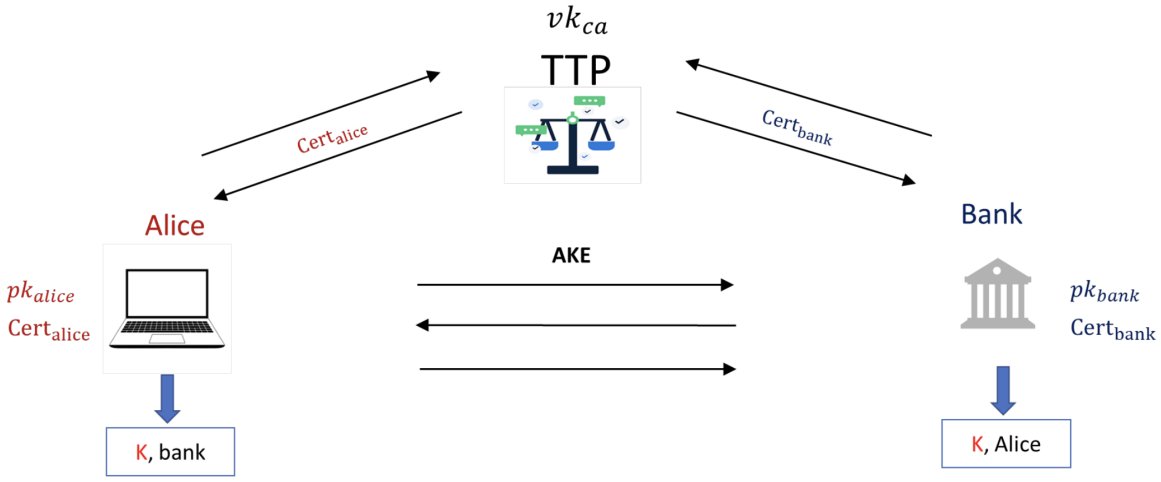

Figure 4: Source: from the class slides

3 certification authorities

Certification Authorities is a Trusted Third Party (TTP) that issues and manages digital certification. Digital certification is based on digital signature technology. The vk_{ca} of a certification authority is public for normal users to verify the digital certification. While the sk_{ca} of a certification authority is private for issuing the digital certification. This section is divided into two subsections. The first subsection introduces digital certification, including the process of checking a digital certification and the content of a digital certification. The second subsection introduces the authentication chain.

3.1 Digital Certification

3.1.1 Validation Procedure

The digital certification is an electronic document to prove the ownership of a public key. Suppose the user checking the digital certification trusts the issuer and finds that the digital certificate's signature is a valid signature of that issuer. In that case, this user can use the

contained public key to securely communicate with the subject of the certification. This process usually happens when a user visits any website that implements the HTTP Secure Protocol. When a user accesses a website or an online service that employs digital certification, their web browser verifies the authenticity of the digital certificate presented by the server. This verification process involves checking the digital signature of the certification against the vk_{ca}, ensuring that the certification has not been tampered with or forged. By leveraging digital certification, organizations can establish secure connections with their users to protect sensitive information. Additionally, digital certificates help users verify the legitimacy of websites and services, mitigating the risk of phishing attacks and online fraud. The following example happens when a user accesses the Hang Seng Bank's homepage.

Example:

Suppose a user visits Hang Seng Bank's webpage through a web browser. The browser first establishes a secure HTTPS connection with the Hang Seng Bank's webpage server. This ensures that the data exchanged between the user's device and the bank's server is encrypted and cannot be intercepted by malicious actors. As part of the HTTPS handshake process, the Hang Seng Bank's server presents its digital certification to the user's browser. This certification contains information about the bank's identity, such as its name, public key, URL, and the digital signature of a trusted certification authority. Then the user's browser verifies the authenticity of the bank's digital certification by checking its digital signature against the vk_{ca} of the trusted certification authority. If the signature is valid and the certification hasn't expired or been revoked, the browser trusts the identity of Hang Seng Bank. By incorporating digital certification into its online platform, Hang Seng Bank ensures that users can access their financial information and conduct transactions securely over the Internet, safeguarding against unauthorized access and potential cyber threats.

3.1.2 Formats and Contents

Digital certifications have several formats, each serving specific purposes and compatible with different systems and applications. These formats typically contain standardized sets of information known as certificate fields, which provide essential details about the certification holder and the certification authority issuing it. The most common format of digital certification is X.509 defined by the International Telecommunication Union (ITU) [JM20]. This format is commonly employed in SSL/TLS encryption for securing web communications. X.509 certificates are encoded using the ASN.1 (Abstract Syntax Notation One) standard and can be stored in different file formats such as PEM (Privacy Enhanced Mail), DER (Distinguished Encoding Rules), or PFX (Personal Information Exchange). A digital certification in X.509 format usually contains the following contents:

- **Subject Name:** This field identifies the entity to which the certificate is issued, typically in the form of a distinguished name (DN) containing information such as the organization name, common name (CN), and organizational unit (OU).
- **Issuer Name** This field specifies the entity that issued the certificate, usually a Certification Authority. It contains information similar to the subject field, indicating its identity. In addition, this field also indicates which signature algorithm is used to sign.

- **Public Key Information:** This field includes the public key of the certification holder; which encryption algorithm is used; The scope of use of this public key; and the key size.
- **Validity Period:** This field indicates the duration during which the certification is considered valid. It includes a start date and time (Not Before) and an expiration date and time (Not After).
- **Digital Signature:** To ensure the integrity and authenticity of the certification, it is digitally signed by the issuing Certification Authority using its private key. The users can verify the signature using the vk_{ca} to confirm the certificate's authenticity.

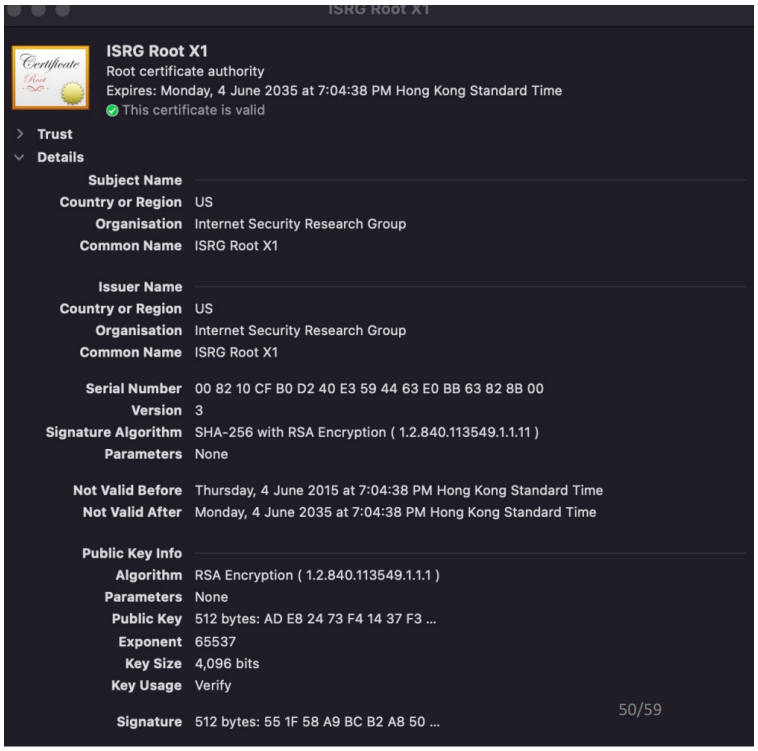

Figure 5: Digital Certification Example

Figure 5 is an example of a digital certification from the PowerPoint of our lecture. In this example, the Organization Name of the subject is *Internet Security Research Group* and the Common Name is *ISRG Root X1*. While the issuer and the subject are the same, which means this certification is self-signed. In the Issuer Name field, this certification also indicates that its Signature Algorithm is *SHA-256 with RSA Encryption*. The valid duration of this certification is between *4 June 2015 7:04:38 PM Hong Kong Standard Time* to *4 June 2035 7:04:38 PM Hong Kong Standard Time*. Besides, the public key info field specifies the *RSA Encryption* algorithm is used to encrypt; the key size is *4096 bits*; and this public key is used to *verify*.

3.2 Authentication Chain

There are multiple certification authorities in the real world since having multiple certification authorities increases the resilience of the PKI. If one certification authority experiences failure issues, subjects can still obtain certifications from other trusted certification authorities. However, it is unrealistic to store the vk_{ca} of all certification authorities in advance since the number of certification authorities is extremely large, and certification authorities can be added, removed, or change their vk_{ca} over time. Therefore, the certification authorities also need digital certifications to prove that the vk_{ca} belongs to the corresponding certification authorities. Authentication Chain is used to ensure the integrity and trustworthiness of digital certifications. In the concept of the Authentication Chain, certification authorities are divided into three hierarchical levels. At the top of the authentication chain, the certification authority is called the root certification authority with a self-signed certification. It is a trusted third party that issues intermediate certifications and signs them using its private key. Root certifications are pre-installed or manually trusted by operating systems, web browsers, and other software. After that, intermediate certification authorities sit between the end entity and the root certification authority. Their certifications are issued by the trusted root certification authorities and they will issue some certifications to the end entity.

In the above example, the certification of Heng Seng Bank's website may be issued by an untrusted intermediate certification authority. In this case, the user will try to access the intermediate certification to find which certification authority signed it. Note that this certification may be also issued by another untrusted intermediate certification authority. Hence, this procedure may repeatedly happen. Finally, this certification chain will end with a self-signed certification. If the final self-signed certification is signed by a trusted certification authority, then the whole chain can be trusted. Otherwise, the user needs to consider whether to trust this unfamiliar root certification authority or not.

References

[CHH+17] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. A comprehensive symbolic analysis of tls 1.3. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 1773–1788, 2017.

[DH22] Whitfield Diffie and Martin E Hellman. New directions in cryptography. In *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*, pages 365–390. 2022.

[DVOW92] Whitfield Diffie, Paul C Van Oorschot, and Michael J Wiener. Authentication and authenticated key exchanges. *Designs, Codes and cryptography*, 2(2):107–125, 1992.

[JM20] Chelsea Jean-Mary. An overview of x.509 certificates. `https://www.ibm.com/support/pages/system/files/inline-files/An_Overview_of_x.509_certificates.pdf`, 2020. Accessed: 2024-03-21.

[Sho94] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.