NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

TTM4135 APPLIED CRYPTOGRAPHY AND NETWORK SECURITY, SPRING 2022

Web Security Lab
Group 7

Martin Skatvedt Sondre Kolberg Jens Lervold Henrik Backer

April 21, 2022

1 Introduction

> **Q1:** Generating a symmetric key k just for encrypting that one message seems like an unnecessarily complicated step. Why does GPG do that, instead of just encrypting the message with p?

A: This form of encryption is known as hybrid encryption, where a symmetric key is encrypted using a public key. GPG uses hybrid encryption, where the symmetric key is unique to each message. By doing this, even though if an attacker could decrypt the symmetric key, they would only be able to decrypt that one message. The symmetric key is also called a session key in GPG. Symmetric encryption is also far cheaper computationally, and saves resources.

> **Q2:** How many bytes does PGP use to store the private signing and public verification keys for each of the two signature types?

A: After generating the two keypairs with RSA and DSA, we found the public key sized by exporting them and checking filesizes. The secret keys were found by looking in the ~/.gnupg folder and checking the filesizes of the two private keys. This is our findings:

- DSA private key = 1.1Kb
- RSA private key = 1.2Kb
- DSA public key = 2Kb
- RSA public key = 1Kb

Listing 1: Exporting public keys

```
gpg --armor --export <fingerprint> > pubkey
```

> **Q3:** How many bytes does PGP use to store the signatures in each of the four cases (both long and short messages)?

A: By using the clearsign option we could get a file with just the signatures. By checking their filesized we found out that the size of the file does not matter and our results are:

- RSA = 310B,
- DSA = 119B

Listing 2: Signing using clearsign

```
gpg -u <fingerprint> --output <filename>.sig --clearsign <filename>.txt
```

> **Q4:** How long, on average, does PGP uses for signature generation and verification in each of the four cases?

A: By using the time command we could get the time it took for each of our cases. Our results are the following:

- 1kb DSA sign = 0.4s

- 1mb DSA sign = 0,45s
- 1kb DSA verify = 0.025s
- 1mb DSA verify = 0.025s
- 1kb RSA sign = 0.4s
- 1mb RSA sign = 0.45s
- 1kb RSA verify = 0.015s
- 1mb RSA verify = 0.016s

This meant that the signing process took on average the same amount of time, but RSA was faster at verifying.

> **Q5:** Discuss your results for the above three measurements. In particular, how well do they correspond to what you expect from what was studied in the lectures? Include an explanation of how the different elements of the keys are stored (such as modulus, exponents, generators).

A:

From the lectures we knew that RSA would be faster at verifying signatures, which was confirmed in the experiment. RSA took 0.015s while DSA took 0.025. DSA should be faster at generating a signature, but our results showed that both methods were equally fast. For both DSA and RSA verifying was faster than signing for both the small and large file. We found that the size of the file did not matter for the amount of bytes stored.

> **Q6:** What assurances does someone receive who downloads your public key from the server where you have uploaded it? What is the difference between the role of the certification authority in X509 and the key server in PGP, with regard to the security guarantees they give?

A: A certificate allows a user to verify that a public key belongs to a certain identity through a trusted authority(like X509). The PGP server we have used does not issue such a signature, it instead verifies trust level of a public key by looking at how many other users trust that key(Web of trust). This means that you can not have a high level of trust for public keys that are not used often.

> **Q7:** Who typically signs a software release like Apache? What do you gain by verifying such a signature?

A: A software release gets signed by someone in the company (one of the contributors) called a Release Manager. After verifying such signature you know that the files downloaded is not tampered with and is approved by one of the developers in the company.

> **Q8:** Why did you obtain a certificate from Let's Encrypt instead of generating one yourself?

A: The certificate we obtain from Let's Encrypt is a publicly signed certificate, which means that many browsers will trust it. However if the certificate was self signed it would hold no real value in the public web. And browsers would not trust it.

Q9: What steps does your browser take when verifying the authenticity of a web page served over `https`? Give a high-level answer.

A: First the browser fetches the certificate and verifies its integrity. This done with public key cryptography and verifies the signature. Next the browser checks its validity period. After it checks that the certificate is valid, it checks for the certificates revocation status. This is due to that a certificate revocation can take days to be fully revoked. Lastly the browser verifies the issuer. There are often two identities linked to a certificate, the issuer owning the signing key and the subject, which refers to the owner of the public key that the certificate authenticates. Browsers check that the issuer is the same as the subject of the previous certificate. [1]

Q10: Have a look at the screenshot. What does the string `TLS_DHE_RSA_WITH_AES_128_CBC_SHA` in the bottom left say about the encryption? Address all eight parts of the string.

A: TLS tells us that the encrypted connection was established using TLS. DHE stands for Diffie-Hellman Ephemeral and refers to the method used for sharing the symmetric key. RSA is the public key encryption algorithm. AES is the symmetric encryption algorithm using 128 bit key. CBC (Cipher Block Chaining) refers to the mode of operation used with the block cipher. Since it only says SHA we can conclude that it uses SHA-1.

Q11: The screenshot is obviously from a few years ago. Do you think the encryption specified by this string is still secure right now? Motivate your answer.

A: In 2017, two researchers demonstrated the first chosen prefix collision attack on SHA-1 [2]. Today sha-256 is the typical hashing algorithm used for internet traffic CBC mode has also be shown to be weak against plaintext-attacks in tls 1.0

Q12: What is the purpose and format of an SCT certificate field? Why might a certificate with an SCT not appear in any certificate transparency log?

A: The purpose of the Signed Certificate Timestamp is to add a certificate to the log within a certain amount of time. This is know as the Maximum Merge Delay. The reason for it not showing up in the transparency log is because of this delay. [3]

Q13: What restrictions on server TLS versions and ciphersuites are necessary in order to obtain an A rating at the SSL Labs site? Why do the majority of popular web servers not implement these restrictions?

A: In order to obtain an A rating you need ≥ 80 points which is a combined score from 3 categories; protocol -, key exchange -, and cipher support. To get rating A you need SSL version ≥ 3.0 and TSL version ≥ 1.2. To get more than 80% in cipher support you need ≥ 128 bits.

Q14: What are the values of the client and server nonces used in the handshake? How many bytes are they? Is this what you expect from the TLS 1.2 specification?

What is the value of the encrypted pre-master secret in the client key exchange field sent to the server? Is this the size that you would expect given the public key of your server?

A:

Server nonce: 535917290f2b679484330b0433860072ad43516a06e4f689444f574e47524401
Client nonce: 1b32eb9b457ee310527a0e9b45f3e235a1fa6e3222cf4c41d8ebaad4c5a93c80

Length: 32 bytes (28 bytes random + 4 bytes timestamp) which is what we expected from the TLS 1.2 specification.

Encrypted pre master:
510b144adc062deaa548a30a13054a4c16ad5626697ba74ba9bb6ca2e6f4b4a15acab
55644a9aeb96a011b7dddcb2de8d0d695170d7fe44b50390162460c1af3760c72d86b
a8e0ac39ca048cc277470cacf09c16273da0e0271572ebbd11450156db3b63293fa35
d4585fc51f354b6c34929c8c1f92f2afafec2644de9c7074affed8b2287b660abff98
91097562208f7b9ff6009ff8736c8c8b88b068032acd1bb030bd9bf34577680554372
6ddbe95c2665bee788d7ce262e8377c4b12d686df186da6b44d5ae28140ffaf7aca35
dd1cfe92737adb5b51015afd58a4668a2f762e8c3c52d192299010a050000b638daaf
788393f3bee85263646be212fe64

Length: 256 bytes

The public key of our server had a length of 2048 bits, which translates into 256 bytes. This is the same as the length of our encrypted pre master key, which it what we would have expected.

> **Q15:** What was the value of the pre-master secret in the session that you captured? Is this the size (in bytes) that you expect from the TLS specification? Explain how Task 16 shows that this session does not have forward secrecy.

A:

The value of the master secret was: 1349c0495de7a46d8dfc33a13723c2a9c67e7f03971da03b e95c6afe6786edd1716416e6c660c0dd5e52e9702812c641

The length of the master secret was 48 bytes, which was the size specified in the TLS specification [4].

The session did not have forward secrecy because the symmetric key was only encrypted with RSA, meaning if an attacker gets the private key of either the client or the server, the session key can be decrypted. If DHE had been used instead, an attacker could not in the future find the key derived using a private key.

2 Cooperation and Experience

The team worked well together and tried to divide the work equally. We met once a week and worked together on the tasks. Martin took a leading role in doing the milestones, but we all took part in the project, and was very happy with the cooperation among the group. We found the lab to be insightful to the relevant protocols used in this course. One of the challenges we faced during this project was to find useful sources when doing research online, as there is a lot of information on these topics. Overall, this project was a good experience.

References

[1] Nick Naziridis. Browsers and certificate validation. https://www.ssl.com/article/browsers-and-certificate-validation/, November 2018.

[2] Gaëtan Leurent and Thomas Peyrin. Sha-1 is a shambles. https://eprint.iacr.org/2020/014.pdf.

[3] E. Kasper B. Laurie, A. Langley. Certificate transparency. https://datatracker.ietf.org/doc/html/rfc6962.

[4] Eric Rescorla and Tim Dierks. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, August 2008. URL https://www.rfc-editor.org/info/rfc5246.