How does HTTPS work? (Episode 6)

ALEX XU
MAY 5, 2022

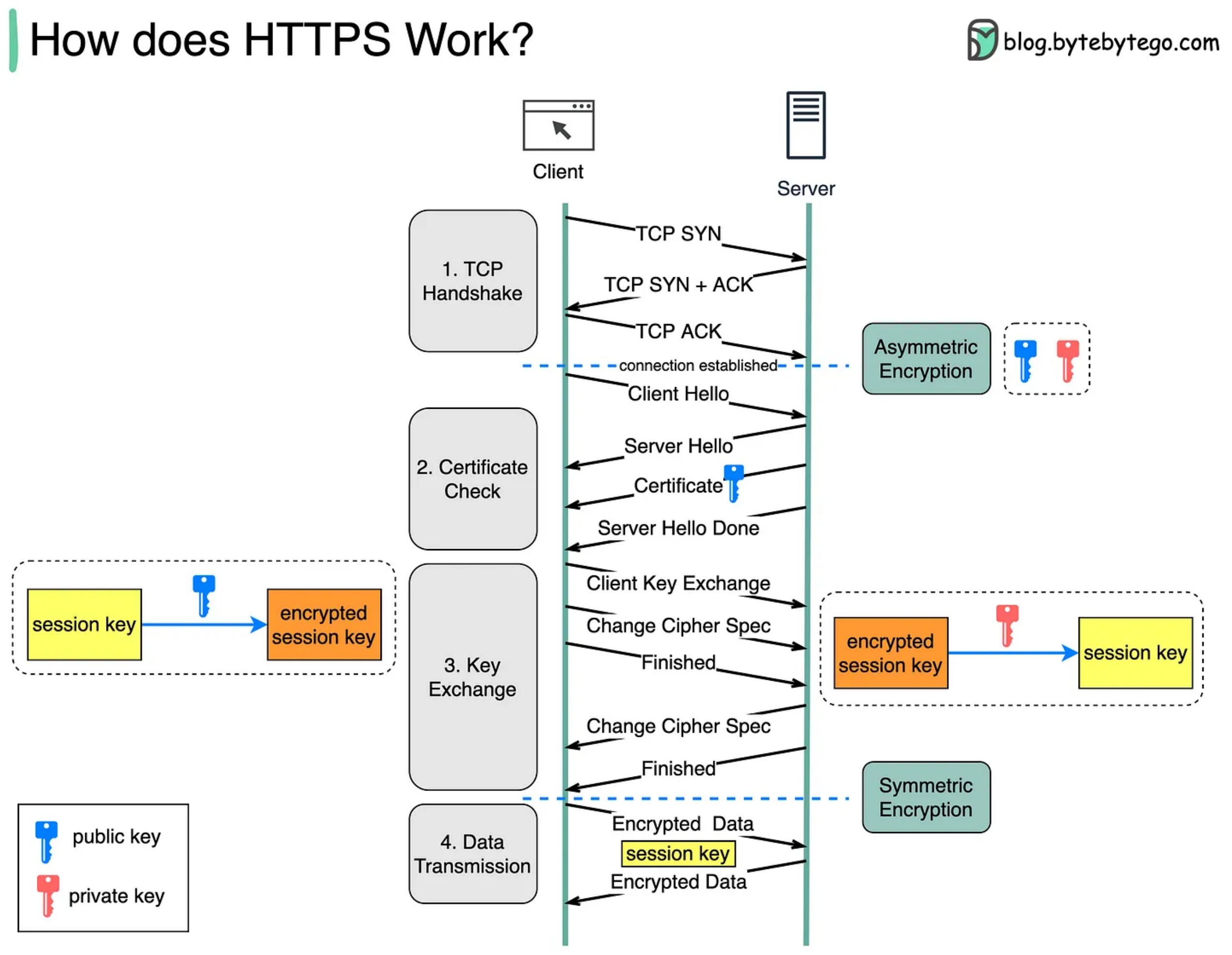
153 8 Share

In this newsletter, we will talk about the following:

- How does HTTPS work?
- How to store passwords safely in the database and how to validate a password?
- How to learn design patterns?

How does HTTPS work?

Hypertext Transfer Protocol Secure (HTTPS) is an extension of the Hypertext Transfer Protocol (HTTP.) HTTPS transmits encrypted data using Transport Layer Security (TLS.) If the data is hijacked online, all the hijacker gets is binary code.



How is the data encrypted and decrypted?

Step 1 - The client (browser) and the server establish a TCP connection.

Step 2 - The client sends a "client hello" to the server. The message contains a set of necessary encryption algorithms (cipher suites) and the latest TLS version it can support. The server responds with a "server hello" so the browser knows whether it can support the algorithms and TLS version.

The server then sends the SSL certificate to the client. The certificate contains the public key, hostname, expiry dates, etc. The client validates the certificate.

Step 3 - After validating the SSL certificate, the client generates a session key and encrypts it using the public key. The server receives the encrypted session key and decrypts it with the private key.

Step 4 - Now that both the client and the server hold the same session key (symmetric encryption), the encrypted data is transmitted in a secure bi-directional channel.

Why does HTTPS switch to symmetric encryption during data transmission? There are two main reasons:

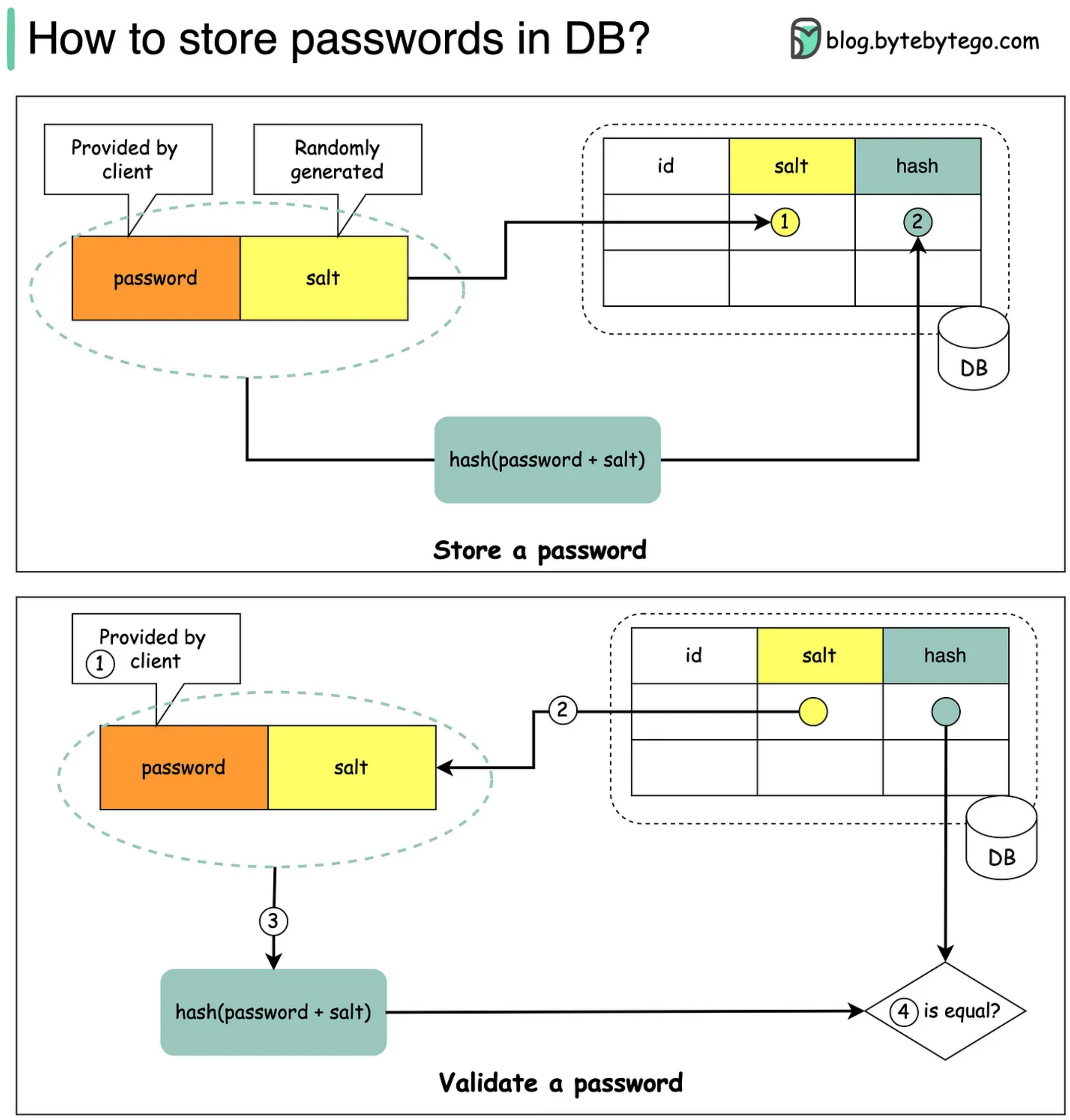
1. Security: The asymmetric encryption goes only one way. This means that if the server tries to send the encrypted data back to the client, anyone can decrypt the data using the public key.

2. Server resources: The asymmetric encryption adds quite a lot of mathematical overhead. It is not suitable for data transmissions in long sessions.

Over to you: how much performance overhead does HTTPS add, compared to HTTP?

How to store passwords safely in the database and how to validate a password?

Let’s take a look.



Things Not to do

- Storing passwords in plain text is not a good idea because anyone with internal access can see them.

- Storing password hashes directly is not sufficient because it is pruned to precomputation attacks, such as rainbow tables.

- To mitigate precomputation attacks, we salt the passwords.

What is salt?

According to OWASP guidelines, “a salt is a unique, randomly generated string that is added to each password as part of the hashing process”.

How to store a password and salt?

1️⃣ A salt is not meant to be secret and it can be stored in plain text in the database. It is used to ensure the hash result is unique to each password.

2️⃣ The password can be stored in the database using the following format: *hash(password + salt)*

How to validate a password?

To validate a password, it can go through the following process:

1️⃣ A client enters the password.

2️⃣ The system fetches the corresponding salt from the database.

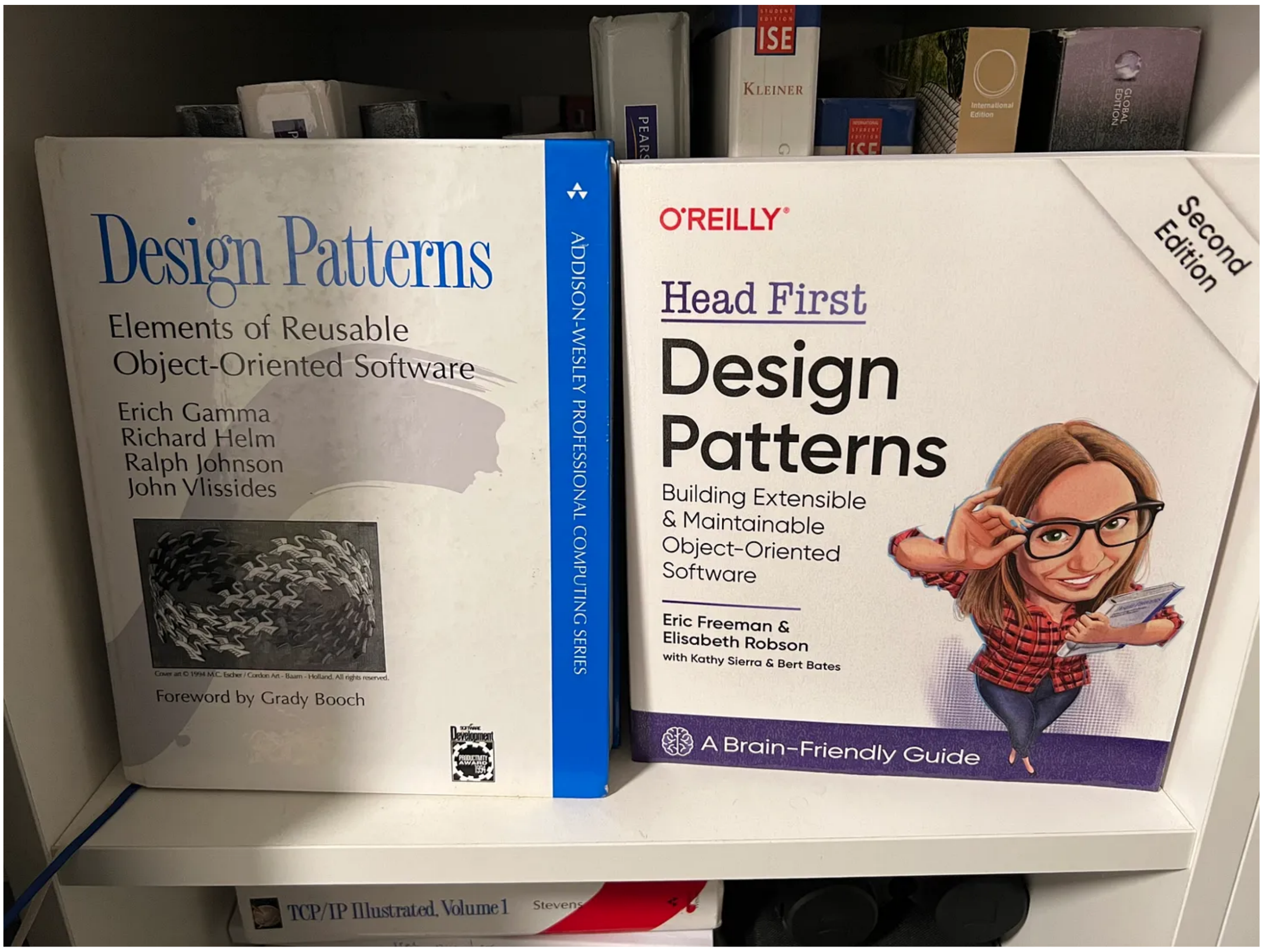
3️⃣ The system appends the salt to the password and hashes it. Let’s call the hashed value H1.

4️⃣ The system compares H1 and H2, where H2 is the hash stored in the database. If they are the same, the password is valid.

Over to you: what other mechanisms can we use to ensure password safety?

How to learn design patterns?

Besides reading a lot of well-written code, a good book guides us like a good teacher.


Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES
O'REILLY
Head First
Design Patterns
Building Extensible & Maintainable Object-Oriented Software
Eric Freeman & Elisabeth Robson
with Kathy Sierra & Bert Bates
Second Edition
A Brain-Friendly Guide
TCP/IP Illustrated, Volume 1

Head First Design Patterns, second edition, is the one I would recommend.

When I began my journey in software engineering, I found it hard to understand the classic textbook, **Design Patterns**, by the GOF. Luckily, I discovered Head First Design Patterns in the school library. This book solved a lot of puzzles for me. When I went back to the Design Patterns book, everything looked familiar and more understandable.

Last year, I bought the second edition of Head First Design Patterns and read through it. Here are a few things I like about the book:

🔹 This book solves the challenge of software's abstract, "invisible" nature. Software is difficult to build because we cannot see its architecture; its details are embedded in the code and binary files. It is even harder to understand software design patterns because these are higher-level abstractions of the software. The book fixes this by using visualization. There are lots of diagrams, arrows, and comments on almost every page. If I do not understand the text, it's no problem. The diagrams explain things very well.

🔹 We all have questions we are afraid to ask when we first learn a new skill. Maybe we think it's an easy one. This book is good at tackling design patterns from the student's point of view. It guides us by asking our questions and clearly answering them. There is a Guru in the book and there's also a Student.

Over to you: which book helped you understand a challenging topic? Why do you like it?

Other things we made:

153 Likes

8 Comments

Write a comment...

Albert May 6, 2022 Liked by Alex Xu

Thanks for writing this article. The content is easy to read and is very valuable. I have a comment on the password storage part. While the use of hashing is conceptionally correct, there are different kinds of hashing algorithms and most people would assume the simplest one (e.g. SHAS256). The practical one that is suitable for password hashing is discussed in https://en.wikipedia.org/wiki/Cryptographic_hash_function#Password_verification, which includes PBKDF2, scrypt and Argon2.

LIKE (2) REPLY SHARE

Rahul Vutukuri May 13, 2022

I really like the way HTTP vs HTTPS are described, in eay language, please do post negative scenarios or in which cases HTTPS is must or cases where HTTP outperform HTTPS

LIKE (1) REPLY SHARE

6 more comments...