

[Home](#)[About](#)[Posts](#)[Tags](#)

# Understanding TLS protocol

2019-07-24 :: Prakhar Srivastav

#tls #security

## Introduction

I have been reading up over this summer on the infamous man-in-the-middle (MITM) attack. Wikipedia defines an MITM attack as:

- 
- > an attack where the attacker secretly relays and possibly alters the communications between two parties who believe they are directly communicating with each other.
- 

The attacker puts itself between two parties to eavesdrop and alter the traffic. In this blog, we will understand how a MITM attack works. We will then discuss, how TLS (Transport Layer Security) tries to reduce the risk of a MITM attack. We will also explore how mTLS can help us to further secure communication.

## Busting some myths

For a long time, I had the wrong notion of how machines communicate over the internet. I conveniently assumed it to be point-to-point. Instead, it is far more complicated. Let's check that by sending a `traceroute` request to facebook.com from my local machine.

```
prakhar@tardis % traceroute facebook.com
traceroute to facebook.com (31.13.72.36), 30 hops max, 60 byte packets
 1  10.149.0.1 (10.149.0.1)  16.916 ms  17.523 ms  17.523 ms
 2  46.246.1.129 (46.246.1.129)  18.220 ms  18.221 ms  18.241 ms
 3  be-2-570.pe2.sto1.se.portlane.net (80.67.2.133)  18.231 ms  18.241 ms  18.216 ms
 4  be-5.cr1.sto1.se.portlane.net (80.67.4.198)  19.205 ms  18.180 ms  18.179 ms
 5  as32934-2-100g-1x1.sthix.net (192.121.80.77)  19.841 ms  19.843 ms  20.095 ms
 6  po104.psw02.arn2.tfbnw.net (157.240.42.249)  19.376 ms po104.psw04.arn2.tfbnw.net (157.240.42.253)  20.648 ms  18.499 ms
 7  173.252.67.171 (173.252.67.171)  18.638 ms 173.252.67.119 (173.252.67.119)  19.438 ms 173.252.67.109 (173.252.67.109)  18.647 ms
 8  edge-star-mini-shv-01-arn2.facebook.com (31.13.72.36)  20.157 ms  20.157 ms  20.613 ms
prakhar@tardis %
```

Now that is not a point-to-point communication. The packets sent

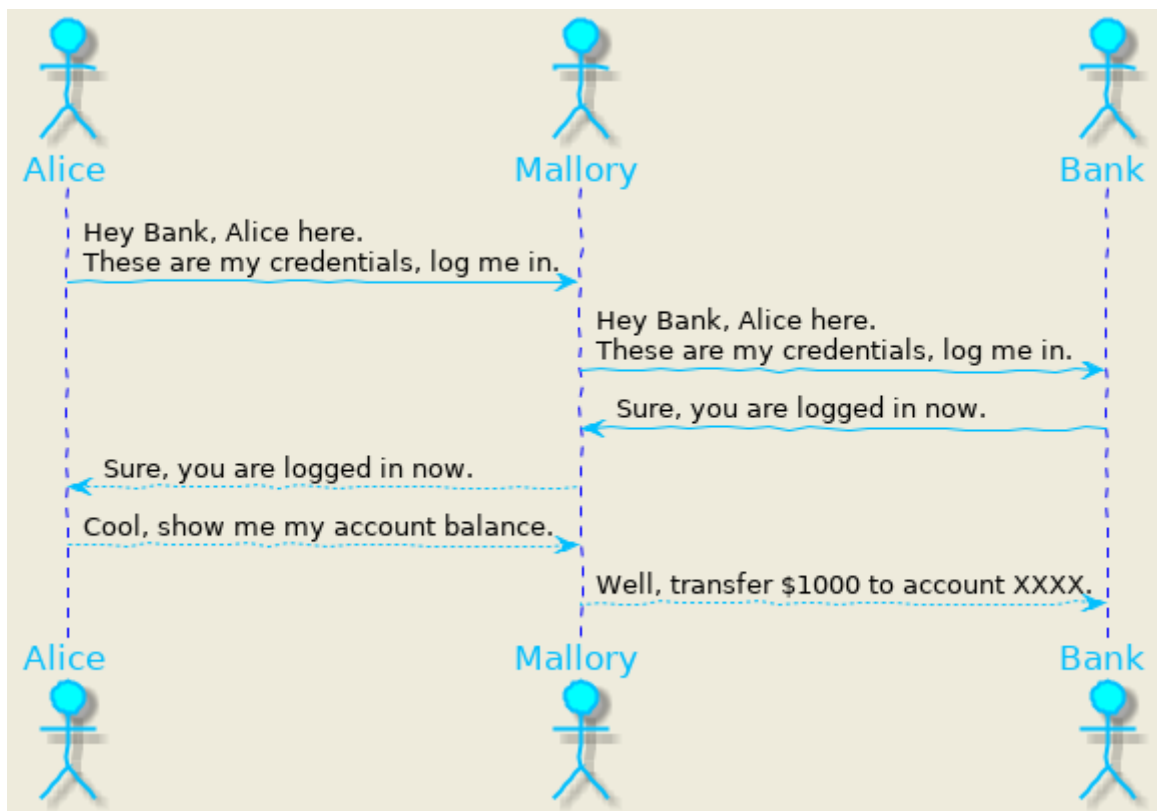
from my local machine route through different hosts before they reach facebook.com. This happens, with every packet that leaves my machine. On top of that, to avoid network congestion, the packets follow different routes.

**Note:** Check out this amazing animation for a better understanding of packet routing.

## MITM mechanics

The crux of a MITM attack lies in the fact that the server will never receive the TCP packets directly from the client machine. Instead, the packets route through many hosts before they reach the server. This provides an attacker a chance to place themselves in the packet route and intercept the traffic.

Let us try to understand this with a simple example. Mallory wants to hijack Alice's conversation with her bank over the internet. Mallory intercepts the traffic between Alice and the bank and establishes herself as the bank to Alice. The image below explains a modus operandi that Mallory could use for a successful attack.



1. Alice sends her credentials to Mallory, assuming that Mallory is her bank.
2. Mallory intercepts the traffic and tries to log in to bank with

Alice's credentials.

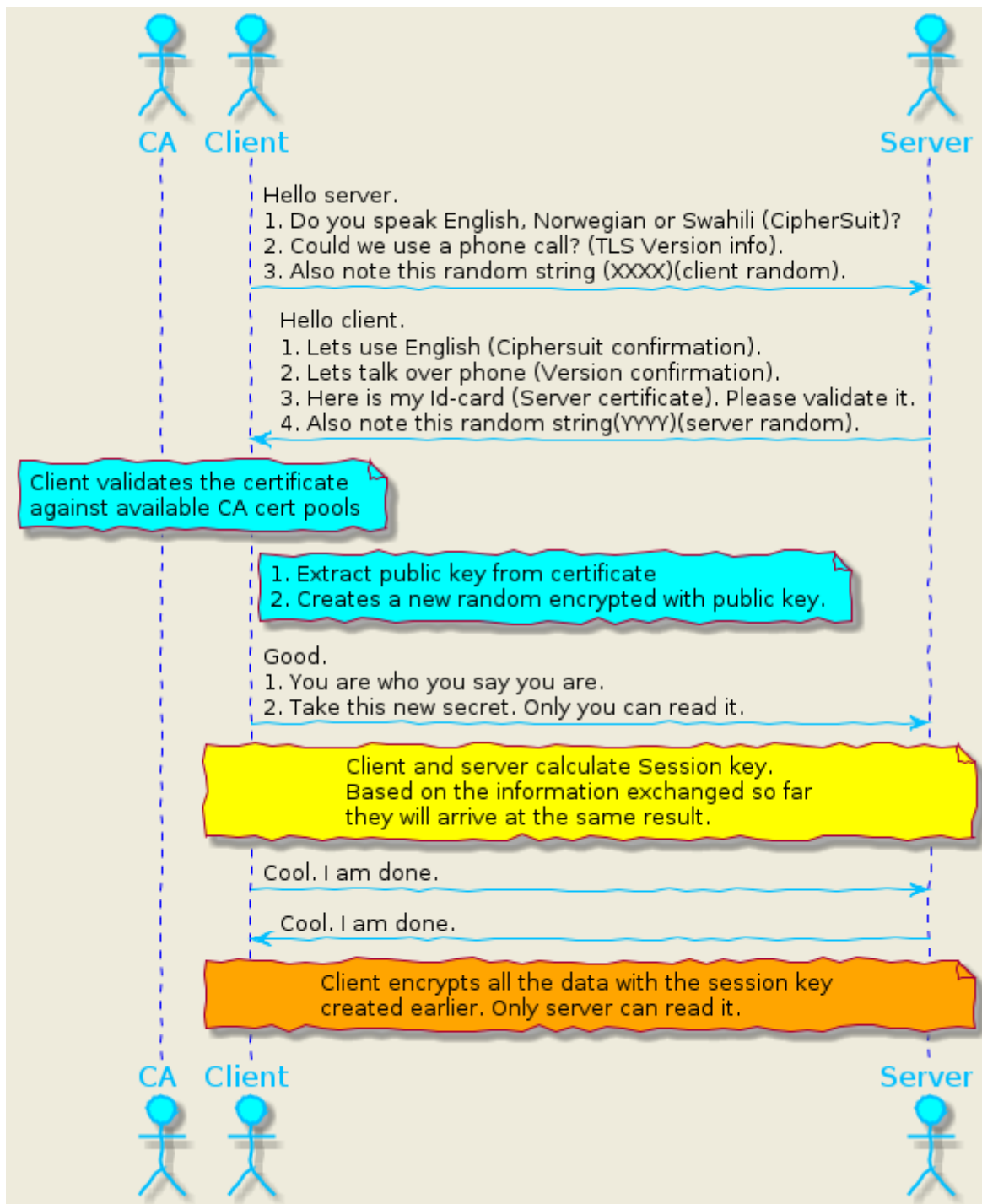
3. Bank validates Alice's credentials and tells Mallory that Alice has logged in.
4. At this point, Mallory has access to Alice's account.

## TLS to the rescue

The primary goal of the TLS Protocol is to provide privacy and data integrity between two communicating applications. It follows a mechanism to establish trust between the server and the client (TLS Handshake). TLS handshake is an agreement between parties on the below factors:

- ▶ The version of TLS to use.
- ▶ Encryption Algorithm (Cipher suite) to use.
- ▶ Authenticate server's identity to the client.
- ▶ Generate symmetric keys to encrypt data after the handshake is complete.

Let us understand it in detail with the help of the below image.



**1.client-hello:** The client starts the communication by sending a message to the server, with the following information:

- ▶ A Cipher suit: A list of algorithms that the client supports.
- ▶ TLS Version information: Client supported TLS version (1.0,1.1,1.2 or 1.3).
- ▶ Client random: A random number that the server will use later to generate the session key.

**2.server-hello:** The server replies to the client with the following details:

- ▶ **Algorithm:** The server confirms one of the supported algorithms from the cipher suite.
- ▶ **Certificate:** To prove the server identity. It embeds the server's public key.
- ▶ **Server random:** A random number generated by the server that the client will use to create a session key.

**3.authentication:** The client verifies the certificate with the certificate authority (CA) that issued it. The client aborts the handshake if CA denies the server's authenticity.

**4.generate pre-master secret:** This is the final part of the handshake. At this stage, the client generates a new random number. It then encrypts it with the public key (received in the certificate) and sends it to the server.

**5.calculate session key:** Using the client/server randoms and the pre-master secret, both parties generate a session key. Interestingly, both sides arrive at the same secret key that can be used for encrypting traffic.


**6.client-finished:** The client sends a "finished" message encrypted with the session key.


**7.server-finished:** The server sends a "finished" message encrypted with the session key.


**8.communication starts:** Client and server can now encrypt the traffic with the session key with confidence that no one can intercept them.


Let us assume that the user connects to facebook via a web browser but over a secured TLS connection. If you see the details of the server certificate from facebook, you will find that it is issued by DigiCert Inc. DigiCert Inc is a trusted Certificate Authority, that issues certificates to domain owners after a thorough check.

Page Info - https://www.facebook.com/?ref=logo

General

Media

Permissions

Security

### Web Site Identity

Web site: [www.facebook.com](https://www.facebook.com)

Owner: This web site does not supply ownership information.

Verified by: DigiCert Inc [View Certificate](#)

Expires on: 4 September 2019

### Privacy & History

Have I visited this web site before today?	Yes, 273 times	
Is this web site storing information on my computer?	Yes, cookies and 160 kB of site data	<a href="#">Clear Cookies and Site Data</a>
Have I saved any passwords for this web site?	Yes	<a href="#">View Saved Passwords</a>

### Technical Details

Connection Encrypted (TLS\_AES\_128\_GCM\_SHA256, 128 bit keys, TLS 1.3)

The page you are viewing was encrypted before being transmitted over the Internet. Encryption makes it difficult for unauthorised people to view information travelling between computers. It is therefore unlikely that anyone read this page as it travelled across the network.

[Help](#)

If you have DigiCert Inc's public key, you can easily verify that facebook's server certificate was cryptographically signed by DigiCert Inc. This way an attacker cannot sign a certificate himself and incorrectly claim to be facebook.com. When the certificate has been modified by even one bit, the sign will be incorrect and the client will reject it.

Now, there is one more question left to be answered. How does the browser access DigiCert Inc's public key? The answer is simple. When you installed your operating system or browser, a list of trusted CAs probably came with it. You also get a list of new CAs through the operating system and the browser updates. That is one of the reasons that you should always keep your browser and operating system updated.

## Can a CA make me trust any server they want!❏

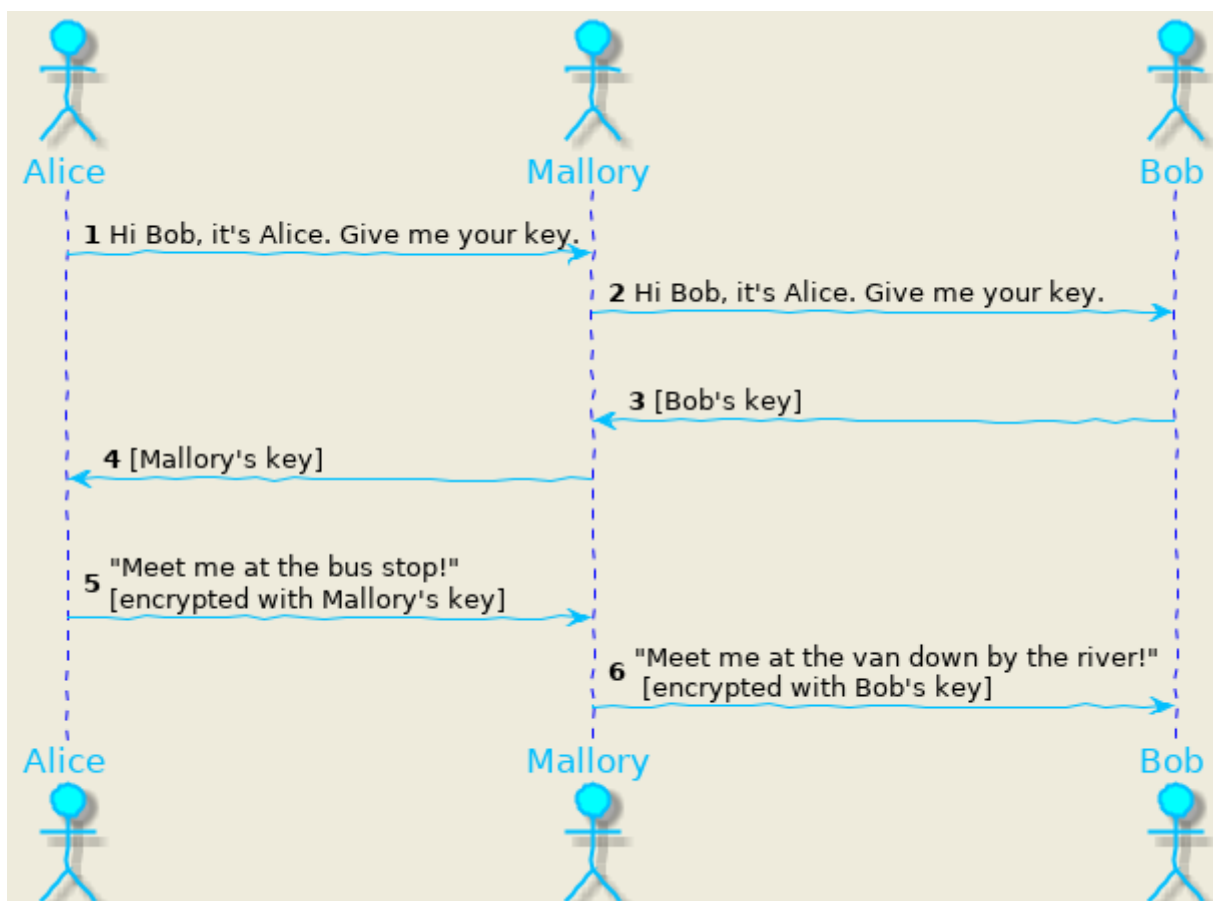
Yes, and that is where the trust comes in. You have to trust the CA not to make certificates as they please. When organizations like Facebook, Microsoft, Apple, and Mozilla trust a CA though, the CA must have audits; another organization checks on them periodically to make sure everything is still running according to the rules.

Issuing a certificate is done if, and only if, the registrant can prove they own the domain that the certificate is issued for.

## Is it enough ?

Well, there are a few notable incidents in the past where the CA has been compromised (check 2011 Diginotar attack in reference). In this scenario, the attackers can issue fraudulent certificates and try to harvest your login credentials. Another common scenario is large organizations, where network admins maintain their private CA pool valid within the organization and if you are using a work laptop, they add the CA to the CA list on your operating system.

In these cases, the attacker can present a fake certificate that will validate against the root CA. Let us examine this vulnerability using this example from Wikipedia. The attacker Mallory, wishes to intercept the communication and deliver a false message to Bob.



1. Alice requests Bob for his public key.
2. Mallory intercepts the traffic, declares herself to be Alice and requests for Bob's public key.
3. Bob thinks that the request came from Alice, gives his public key to Mallory.
4. Now Mallory has Bob's key, and she can start an MITM attack. Mallory forges the message and sends Alice her public key instead.
5. Alice, thinking that she has Bob's key encrypts her message with Mallory's key.
6. Because the message was encrypted with Mallory's key, Mallory can decrypt it, read it, modify it (if desired), re-encrypt with Bob's key, and forward it to Bob.

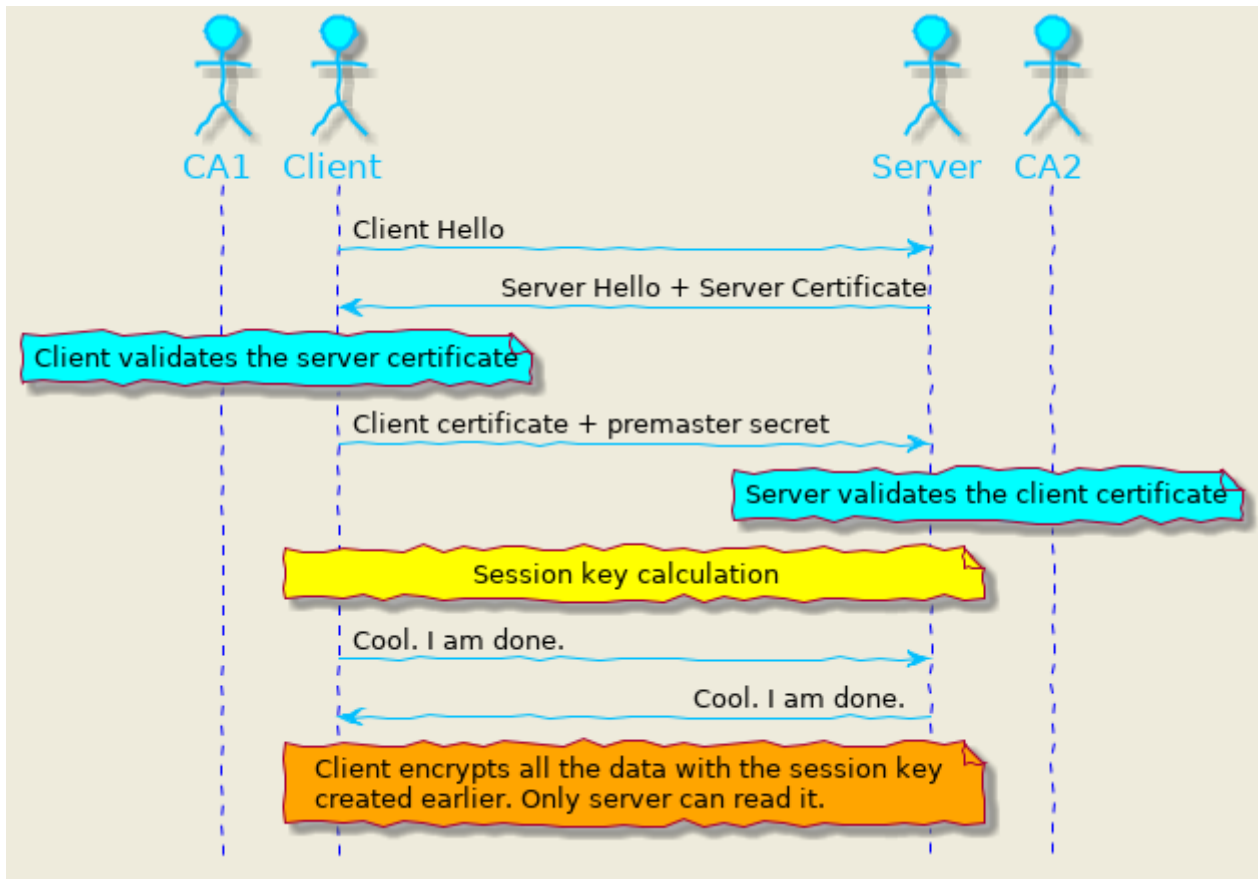
Since the Mallory's certificate is signed by same CA as Bob's, Alice will never notice the attack in progress.

## mTLS for further security

The TLS handshake provides a way for the client to send its certificate to the server. This allows the server to authenticate the client certificate with CA that issued it. This mechanism is termed mutualTLS or mTLS. Using mTLS the server can authenticate the client the same way as the client authenticated the server. The mechanism of the mTLS is exactly the same as TLS except that while sending the pre-master secret.

- ▶ the client additionally provides its public certificate to the server.
- ▶ the server then validates the client certificate through CA.
- ▶ if the CA denies the certificate, the connection is aborted.





Using mTLS, both the client and the server can validate the identity of each other. This could be greatly effective in applications dealing with financial or health-related data. mTLS is also a very common authentication technique in microservice architectures. Combined with other techniques like multi factor authentication, strict transport security (HSTS), access controls, etc. TLS/mTLS could prevent a lot of security threats.

## Conclusion#

Although TLS and mTLS provide security to a large extent, they are no silver bullet to prevent MITM attacks. The efforts should be made to defend the infrastructure and network at all OSI layers. In this blog, we have investigated the TLS protocol from a theoretical viewpoint. I will follow up on this topic from a programmatic perspective, where I will explain how to implement TLS on the server and the client. I also intend to explain the algorithm (Diffie-Hellman in particular) for key exchange and how do both the client and the server arrive at the same secret key in one of the upcoming posts.

## References#

### TLS Specifications:

- ▶ [TLS-1.2 RFC](#)
- ▶ [TLS-1.3 RFC](#)
- ▶ [X.509 PKI RFC](#)

### Interesting reads on MITM attacks:

- ▶ [2011 Diginotar attack](#)
- ▶ [Lenovo security incident](#)
- ▶ [NSA impersonation of Google](#)
- ▶ [Vulnerability on IoT devices](#)

### Other links:

- ▶ [movement-of-data](#)
- ▶ [wikipedia-tls](#)
- ▶ [wikipedia-mitm](#)
- ▶ [IBM-security-report](#)
- ▶ [this interesting question on stack exchange](#)
- ▶ [tls-vulnerabilities](#)
- ▶ [ways-to-defend-network](#)
- ▶ [what-happens-in-a-tls-handshake](#)

READ OTHER POSTS

[← Remoto RPC Framework](#)

[Golang http request h... →](#)

[Github](#) | [LinkedIn](#)

© 2022 Powered by Hugo :: Theme made by panr