Data and network Security

moshahmed@gmail
hmi-tech 2021

Password login

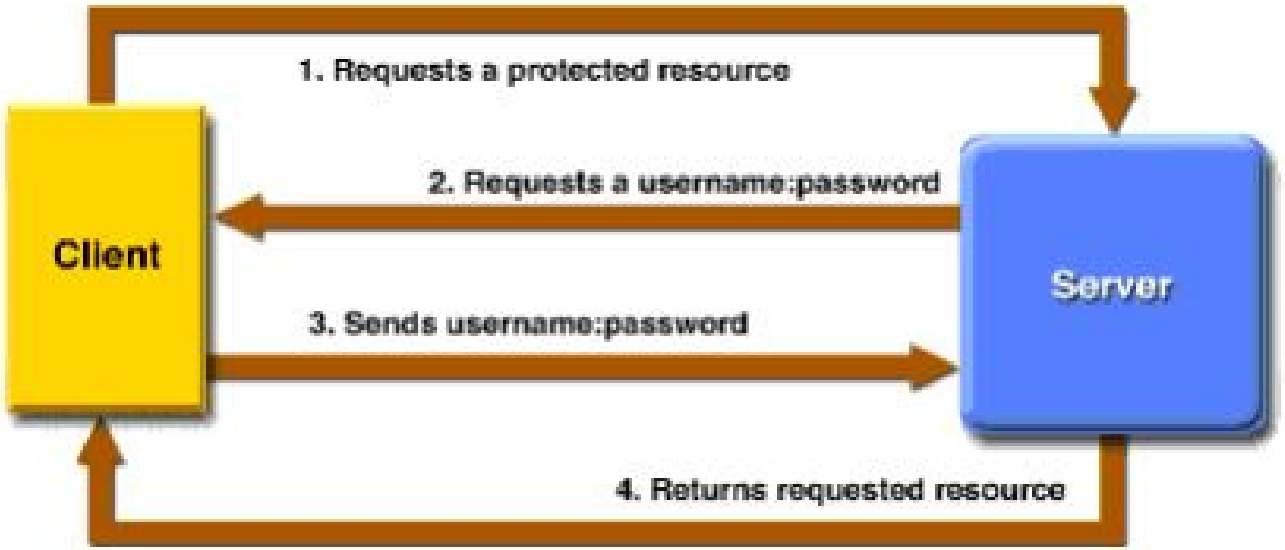


Figure 32-2 HTTP Basic Authentication

Certificates

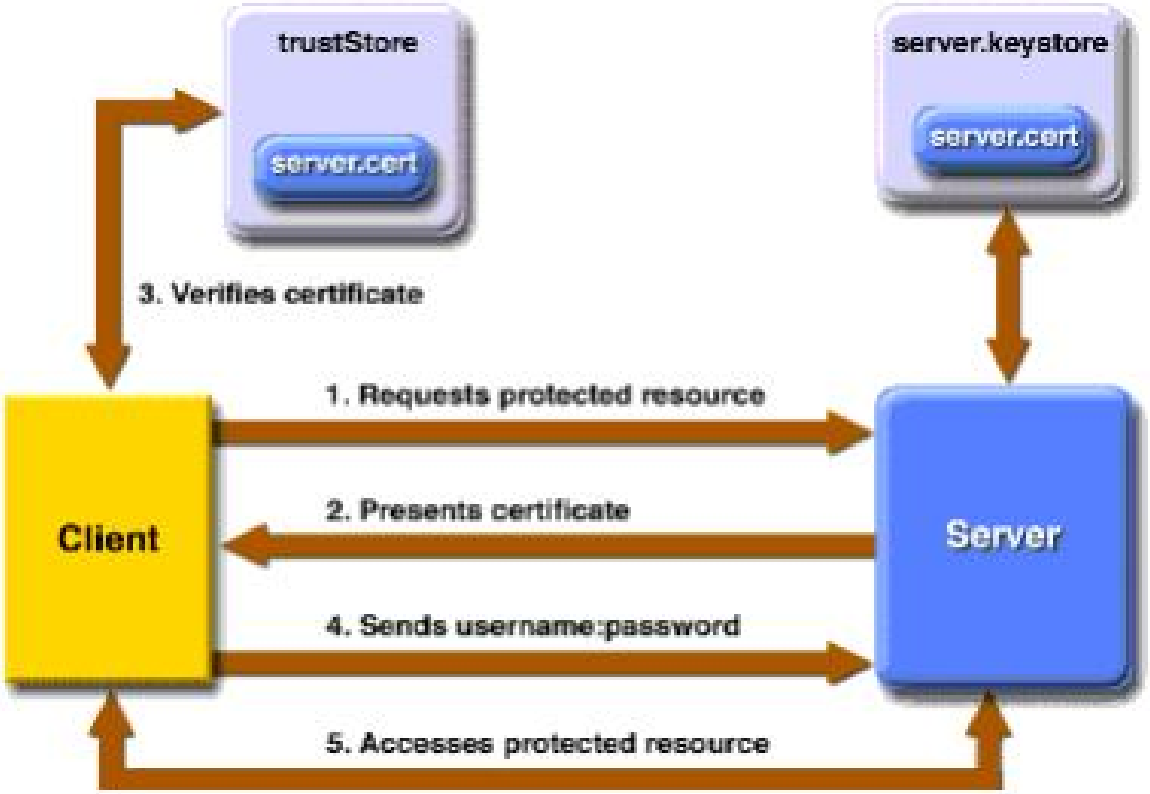
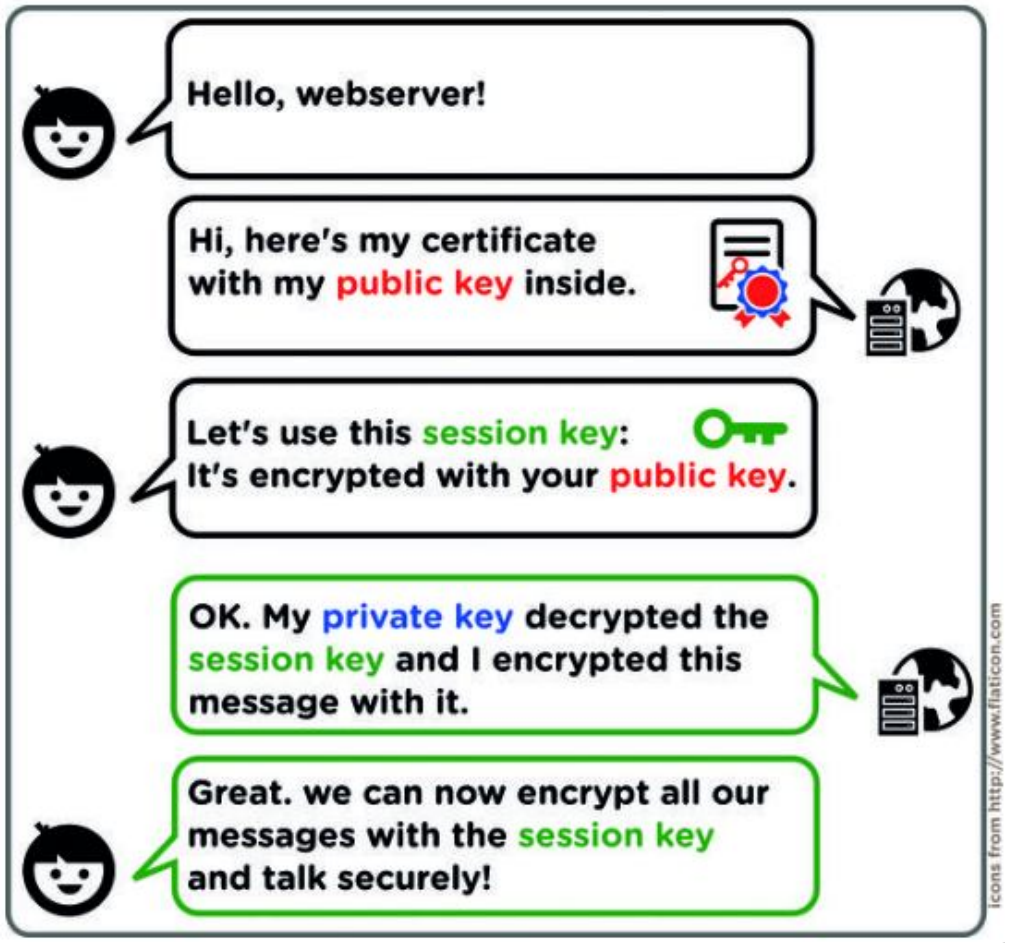
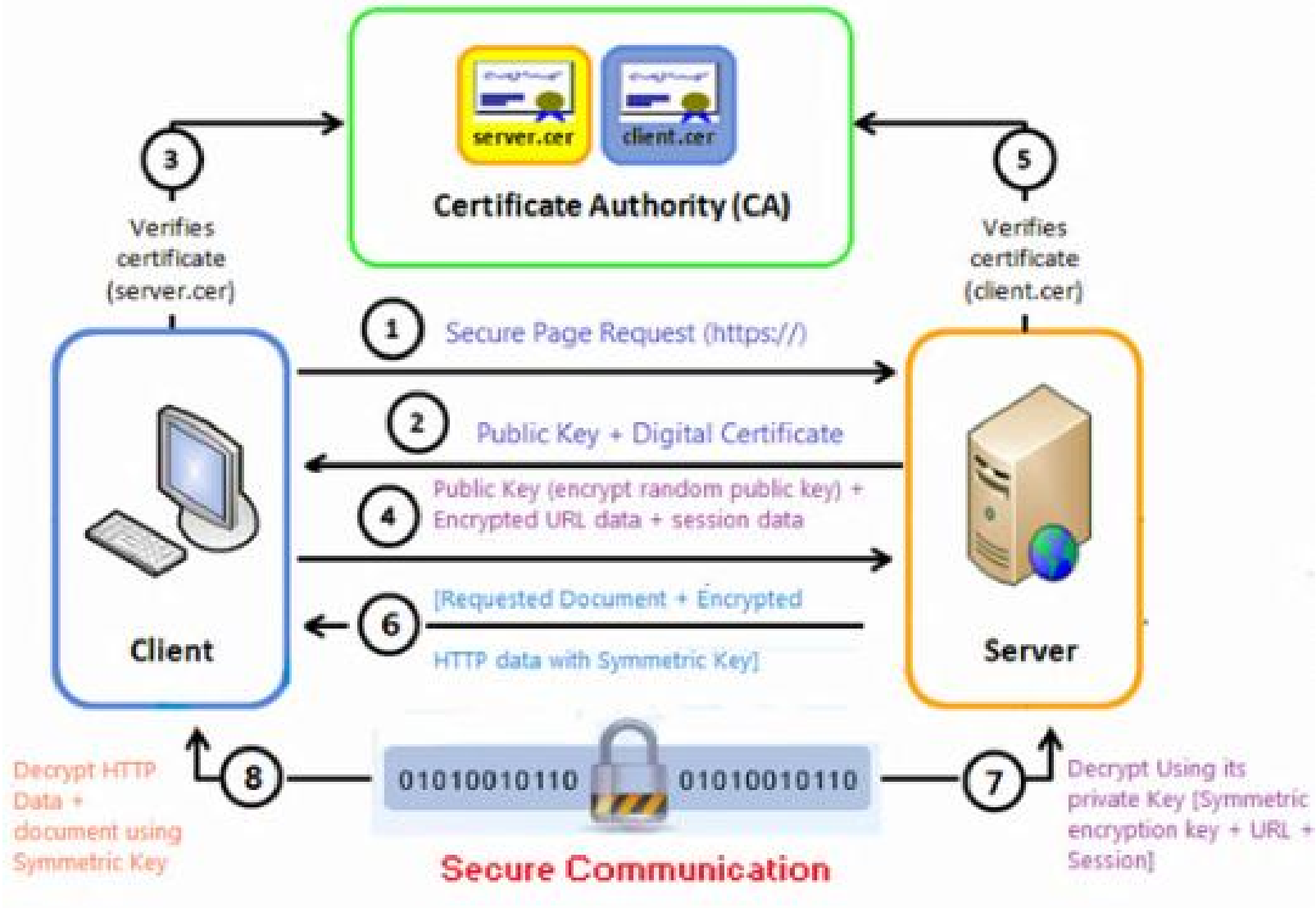


Figure 32-5 User Name- and Password-Based Mutual Authentication

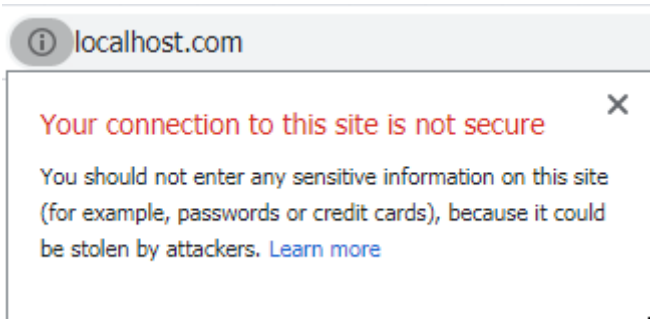
Web server Certificate

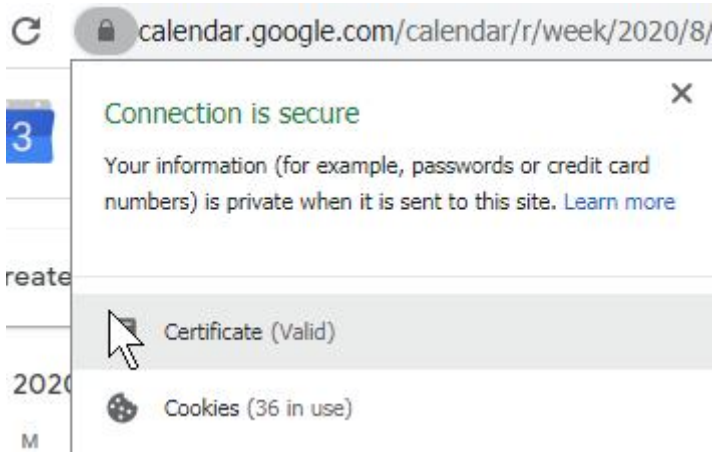


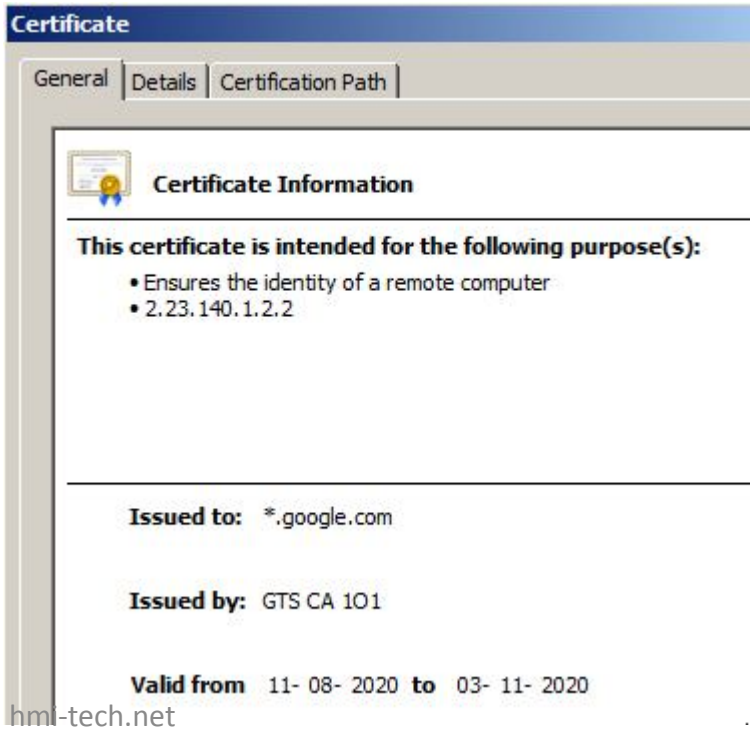
Signed certificate



Website certificates







Complexity of passwords

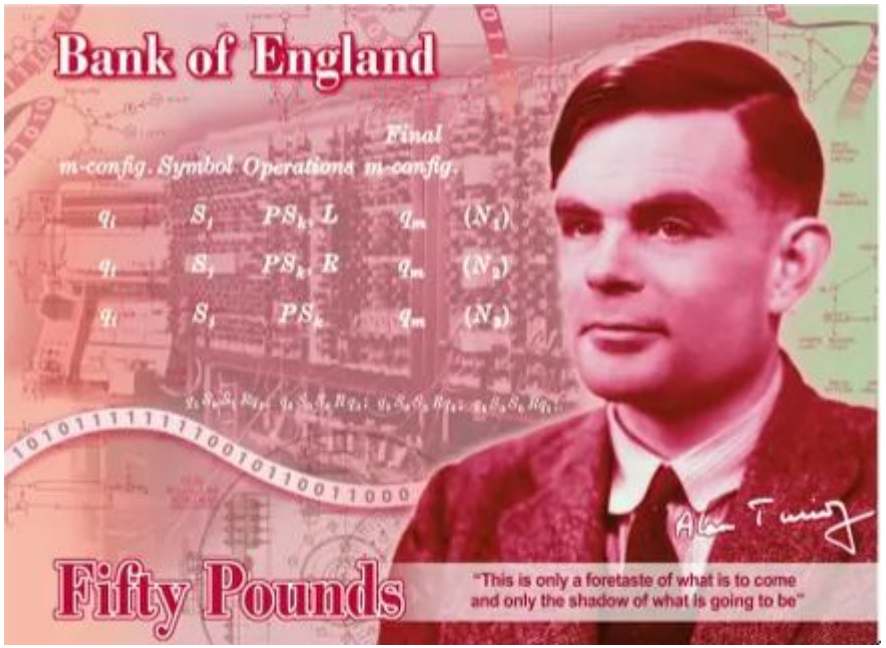
Brute force - Number of choices to try:

- alphabets = 26 x 2 (uppercase+lower) + 10 (digits)= 6 bits/char
- Multiplied by Length of password
- 5 char password = 5x6=30 bits ~ 9digits ~ 1 billion choices

Issues

- Time needed to try each password, 1M/second
- Rate limit number of (online) tries, 3 tried and blocked for 1min
- Server has to store password or one way hash. If hacker gets the db, they can build tables of all guesses (expensive). So add fixed random digits (salt) to each password.
- Users cannot remember long passwords, they use common words/pet name/address/DOB.
- MITM 3rd party listener on the network can reuse password

Who cracked the Enigma encryption in WW2?



Minimal Practices

- Never share passwords.
- Don’t put passwords on internet.
- Atleast 8 characters + 3 digits + special chars.
- Different passwords on different sites
- Change passwords every quarter
- Randomly generated, not from known words.

```
$ dd if=/dev/urandom bs=12 count=1 status=none |  base64
      ylF8JjZSz9Kcvv5w
```

- Use MFA on aws, linux, gmail, fb, insta, twilio, hotmail

Good Practices

- Lock laptop screen when away.
- Encrypt dev machine hard-disks
- Don’t run unrelated streaming/games on dev machine
- Don’t install unknown software from unknown sites
- Regularly audit software running on your machine.
- Audit network connections on your machine.

PII data

- PII data – name, phone number, email, DOB, images, Id, addresses.
- Don’t download PII data onto machine, except for support / business needs as per *Terms of Use*.
- Delete PII data after usage. Track what PII data you accessed and why.
- Never log PII/security tokens in log files /papertrail / github.

OTP - One time password

Userid + Password + 2FA/MFA

- Token changes every minute / login

Difficult to reuse, except by social engineering.

1. Userid + password + SMS.
2. Userid + password + RSAId

   Used by Banks

3. Google Authenticator App

UserId + Password + MFA - changes every minute.

Difficult to reuse, except by social engineering.

Login: username, password, 6 digit code

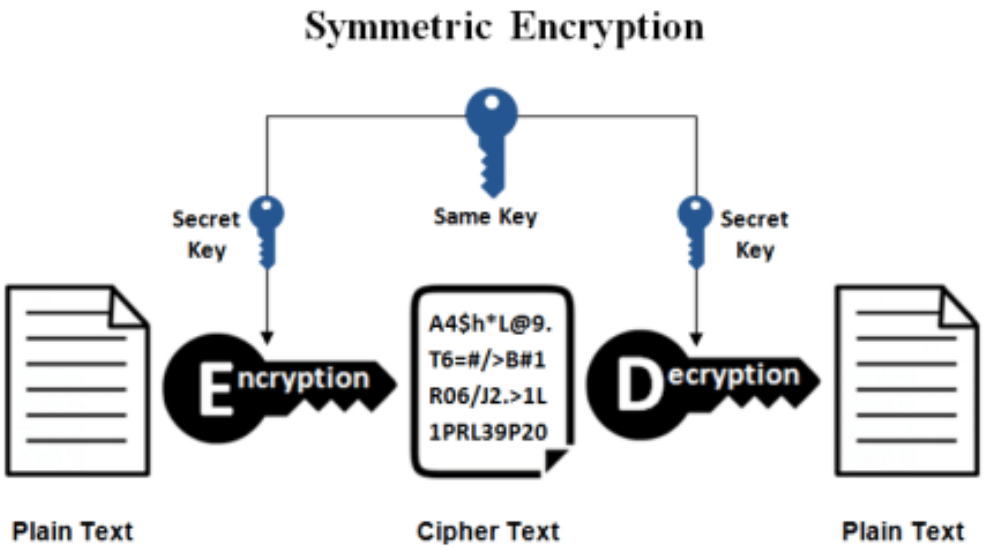
Server: What is hash(time+QRcode2)

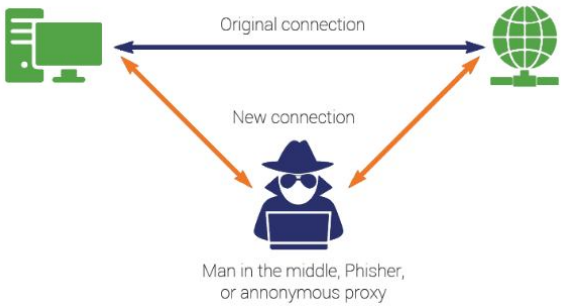
Client: Computes hash(time+QRcode2)

Needs accurate time

Used by: Gmail,Amazon, Facebook,Insta, Microsoft, Github,Twilio

Communication over insecure channel
Exchanging password is unsafe

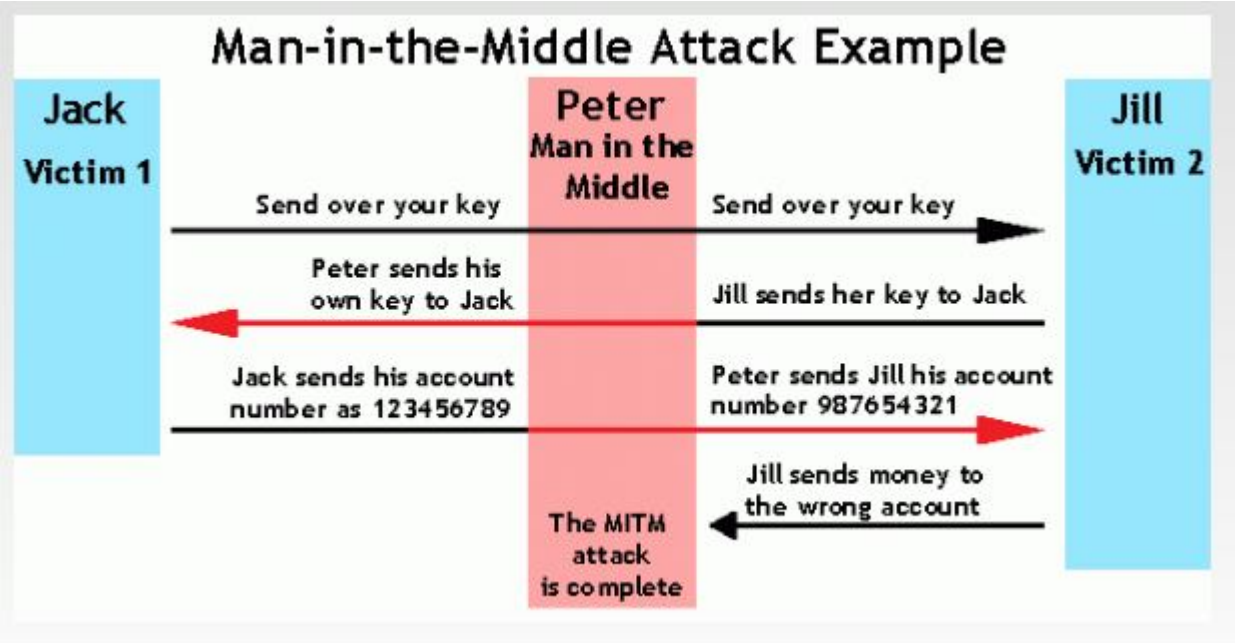




MITM

Problem – how to exchange keys?

MITM attack



Login Protocol -1

Example

- Alan: Do you know Eve's phone?
- Bob: Yes.
- Alan: Prove it.
- Bob: 650-555-1234

Is this a good protocol?

Problems

- Now Bob knows the number
- Can be used only once

Protocol 2

Example

- Alan: Do you know Eve's phone?
- Bob: Yes.
- Alan: Prove it.
- Bob: 650-…

Is this a good protocol?

Problems

- Bob knows part of the number each time
- Can be used only a few times

Protocol 3

Example

- Alan: Do you know Eve’s phone?
- Bob: Yes.
- Alan: Prove it, what is last digit?
- Bob: 4

Is this a good protocol?

Protocol 4

Example

- Alan: Do you know Eve’s phone?
- Bob: Yes.
- Alan: Prove it, what is the sum of digits?
- Bob: 36

Is this a good protocol?

Problem

- Number can be computed after many rounds.

One Way Function

Hashing/fingerprinting, given the knowledge (account number) you can compute the sum (fingerprint) but you can't get the account number from the sum.

Protocol 5 – one way hashing

Example

- Alan: Do you know Eve's phone?
- Bob: Yes.
- Alan: Prove it, what is the sha2 hash digits?
- Bob: 2dd619305603f60f68bc…

Is this a good protocol?

Protocol 6 – hash + challenge response

Example

- Alan: Do you know Eve's phone?
- Bob: Yes.
- Alan: Prove it, what is the first 3 hex digits of the sha2 hash digits?
- Bob: 0x2dd

Is this a good protocol?

Pros/Cons

- Can be used only once
- Hash to number can be computed using “Rainbow tables” (map: strings -> hashes).
- Server has to store your secret key to verify your answer.

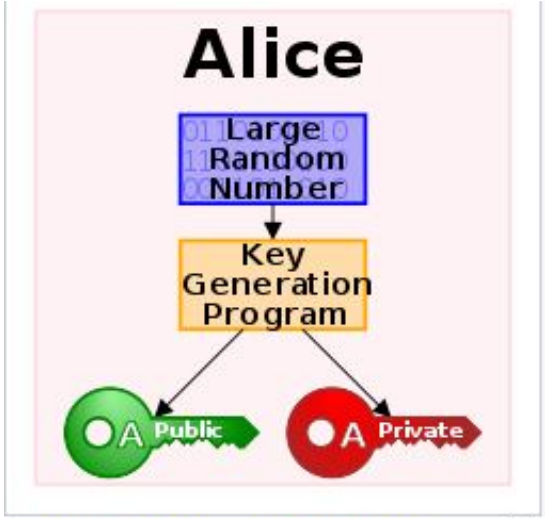
Zero knowledge proofs

Zero-knowledge proof or *Zero-knowledge protocol* is a method by which one party (the *prover*) can prove to another party (the *verifier*) that a given statement is true, without conveying any information apart from the fact that the statement is indeed true

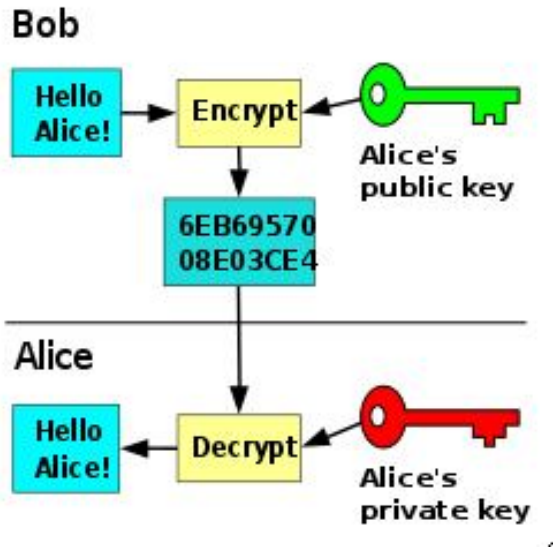
Pros/Cons

- Some math required
- Protocol can be used forever with no loss of private information.
- Bob can prove he knows the number without revealing anything (0 knowledge)
- Used by ssh login – server challenges the user to prove she has the private-password, by sending an encrypted random number (challenge response).

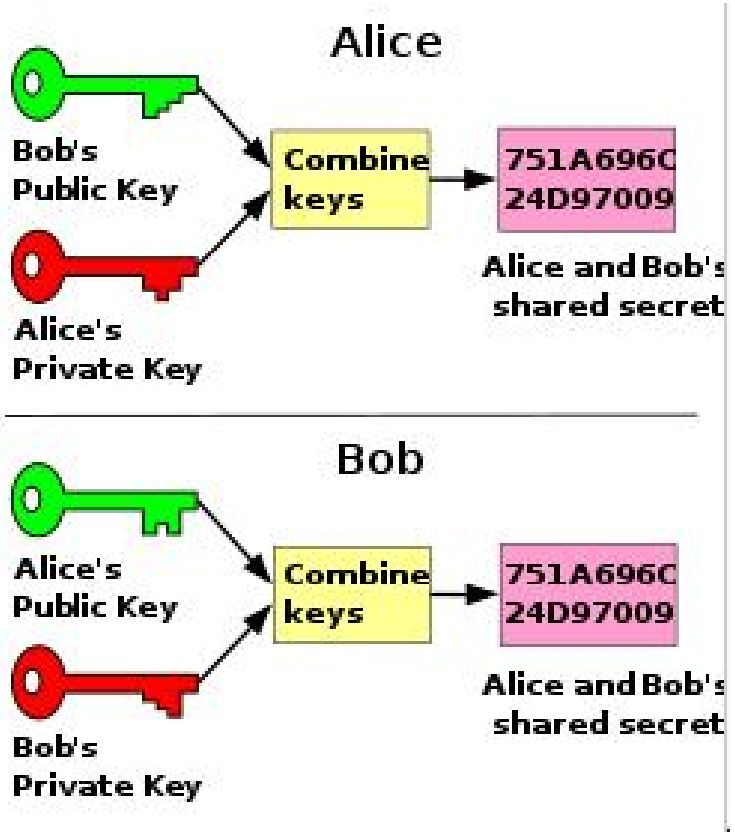
Public Key



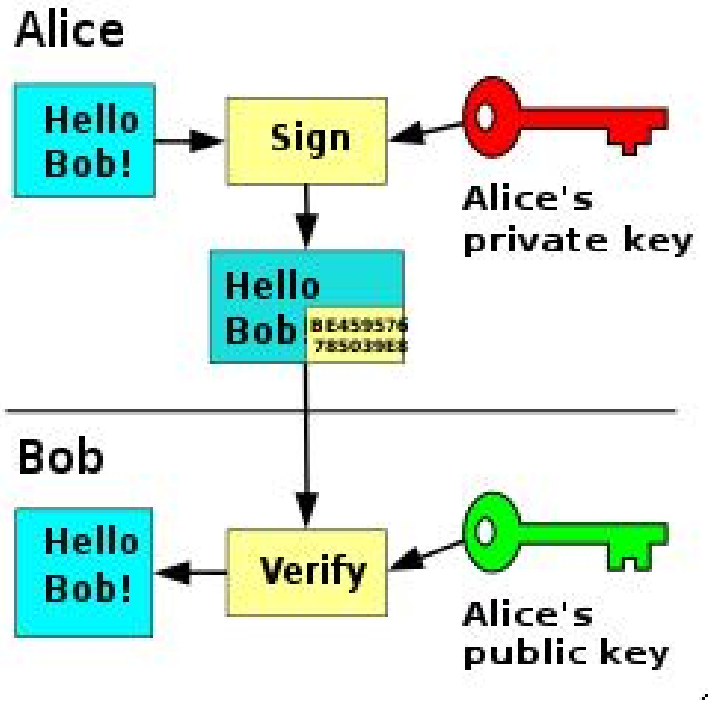
An unpredictable (typically large and random) number is used to begin generation of an acceptable pair of keys suitable for use by an asymmetric key algorithm.



Shared Secret



Signing

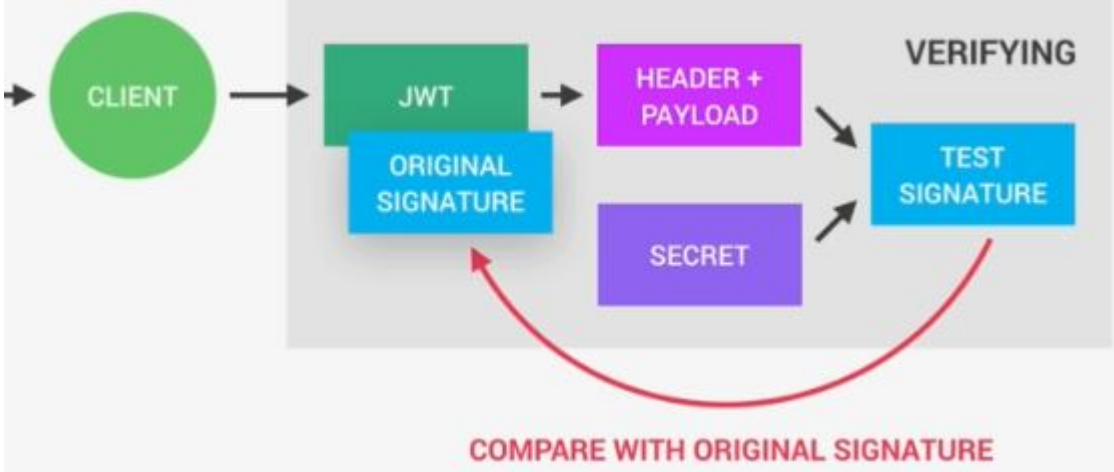

Alice
Hello Bob!
Sign
Alice's private key
Hello Bob
Bob
Hello Bob!
Verify
Alice's public key

Oauth Tokens

- Example of a single use auth token with limited validity.

- **OAuth** is an open standard for access delegation, commonly used as a way for Internet users to grant websites or applications access to their information on other websites but without giving them the passwords

JWT (signed data)



Cookies

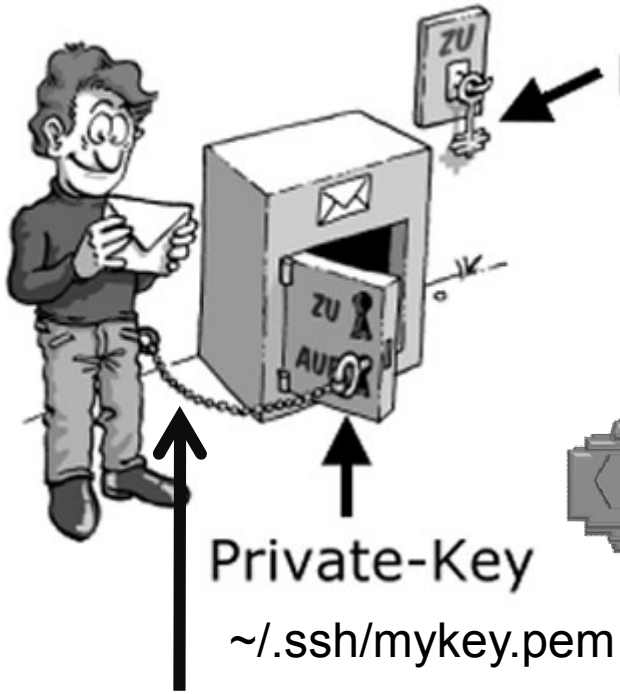
- **Cookie** is a temporary id, sent by server to the client (browser) to identify the user.
- Secure - https only
- httpOnly - not visible to javascript.
- It is has expiry date.
- Valid only on issuing domain (e.g. gmail.com cookie won’t work on youtube.com)

Issues

- **XSS**: 3rd party Javascript injection into your website.
- **SQL Injection**: data contains sql code.
- Weakness in code (not validating user inputs).
- Weakness in servers.

Public key usage in ssh, gpg, ssl

cloud: github, aws, google, fb, microsoft, cc, sims



Public-Key

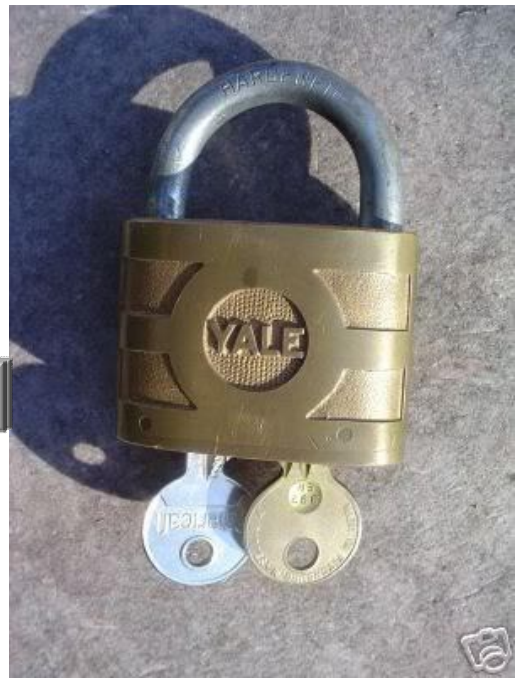
~/.ssh/mykey.pub

Private-Key

~/.ssh/mykey.pem

Passphrase

pass:xyz



Application - SSH

- [Zero knowledge proofs] You can use it billion times without divulging any private info.
- You never send your password to the server
- Server sends a challenge - a random number encrypted with your public key
- Only you can open the challenge with your private key and send the answer back to the server and prove you have the private key.
- Server lets you login.

ssh agent

- Your agent keeps your private key, so you cant lose it accidently.
- You never hold the raw private-key, not even on saved on disk.
- When challenged, you give the challenge to the agent, the agent gives the answer to you, which you give to the login server.

Smart cards - CC and SIMs

- The SIM card contains a private key or more commonly a symmetric key called the "Ki", and the card is designed to never divulge this key to the outside world.
- Reading the key from the impossible without destroying the original card and data.
- Impractical to clone a SIM card.

Server setup

```
# Create a ssh key
$ ssh-keygen   -f   mykey.pem

# Add a passphrase to your key, if you don't have one.
# Never share this private-key/passphrase with anyone
$  ssh-keygen -p -f mykey.pem
Passphrase: "secret pass phrase"

# Let the server know your login public key (no passwd reqd)
$ ssh-keygen -y -f  mykey.pem >>
    server:~your/.ssh/authorized_keys
```

Client setup Linux

$ cat ~/.ssh/config

```
host myserver
     user            ubuntu
     hostname        54.198.45.19
     IdentityFile    ~/.ssh/mykey.pem
```

$ eval $(ssh-agent -s) # Start agent

$ ssh-add ~/.ssh/mykey.pem # Give it your key

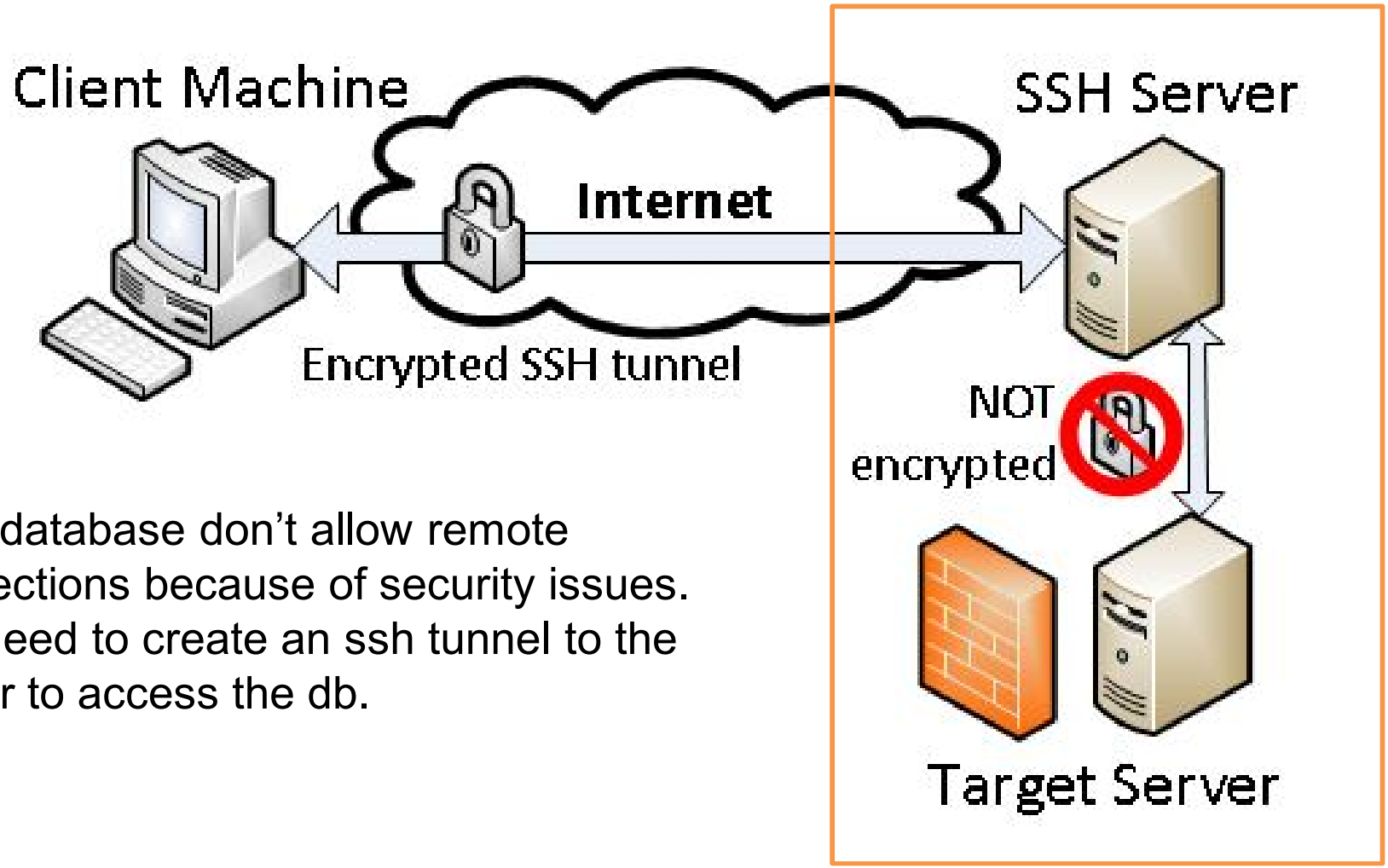
password for mykey.pem: ****

$ ssh myserver # Login to server

Client setup on Windows

```
# Start agent with your key
> pagent ~/.ssh/mykey.pem   # Key to agent
  password for mykey.pem:****
> putty –load myserver      # Login to server
```

Example use: ssh tunnel (and vpn)



Most database don't allow remote connections because of security issues. You need to create an ssh tunnel to the server to access the db.

Public Key Crypto
Diffie Hellman Merkle 1976

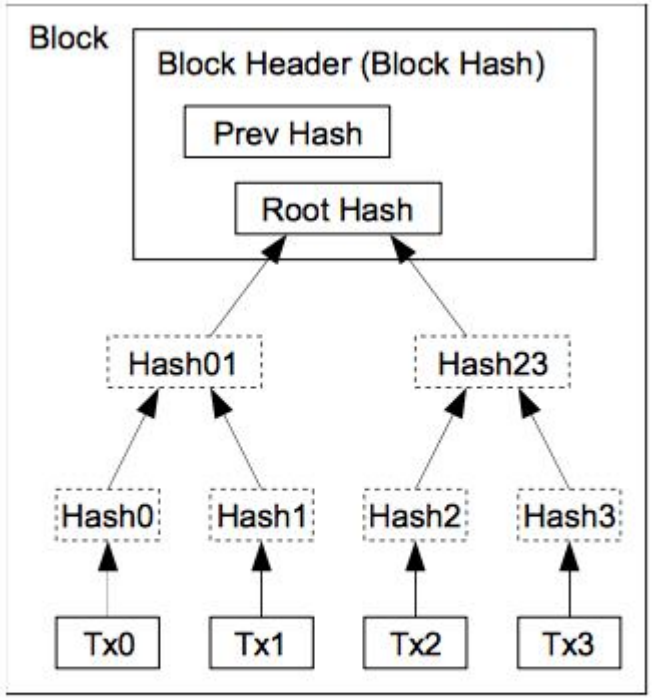
Merkle puzzles idea: easy to check, hard to solve without hint.

Implemented in RSA using
large (100 digit) random numbers,
Idea: Numbers are easy to multiply, but hard to factor.

Merkle enrolled in CS244 (Undergrad Computer Security) at UC Berkeley in 1974. Merkle submitted a proposal for what would eventually become known as *Public Key Cryptography* -- which Prof rejected. he dropped the course, but kept working on the idea.

Application – Block Chain

- Merkle Trees, distributed hashes of data



Transactions Hashed in a Merkle Tree

Merkle Trees in Bitcoin

References

- Wikipedia: Zero knowledge proof, RSA, Public Key, Key Exchange, Diffie Hellman, Felton Bitcoin, Merkle tree, gpg, ssh, gauth
- *Applied Cryptography* by Bruce Schneier.

For details, search above terms in Google.

Questions?

Thank you.