



Smartphone as a security token



Group 18

David Gonçalves

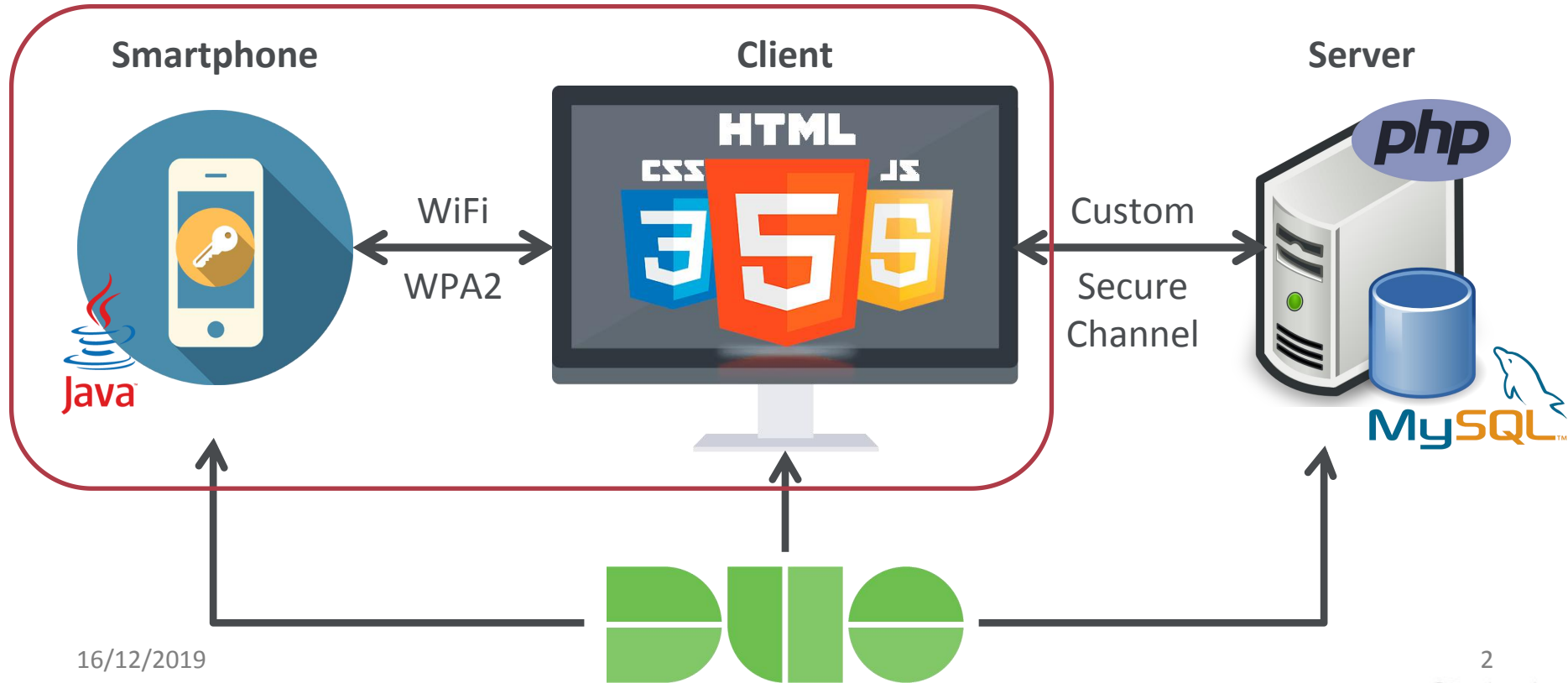
Leonardo Troeira

Francisco do Carmo

16/12/2019

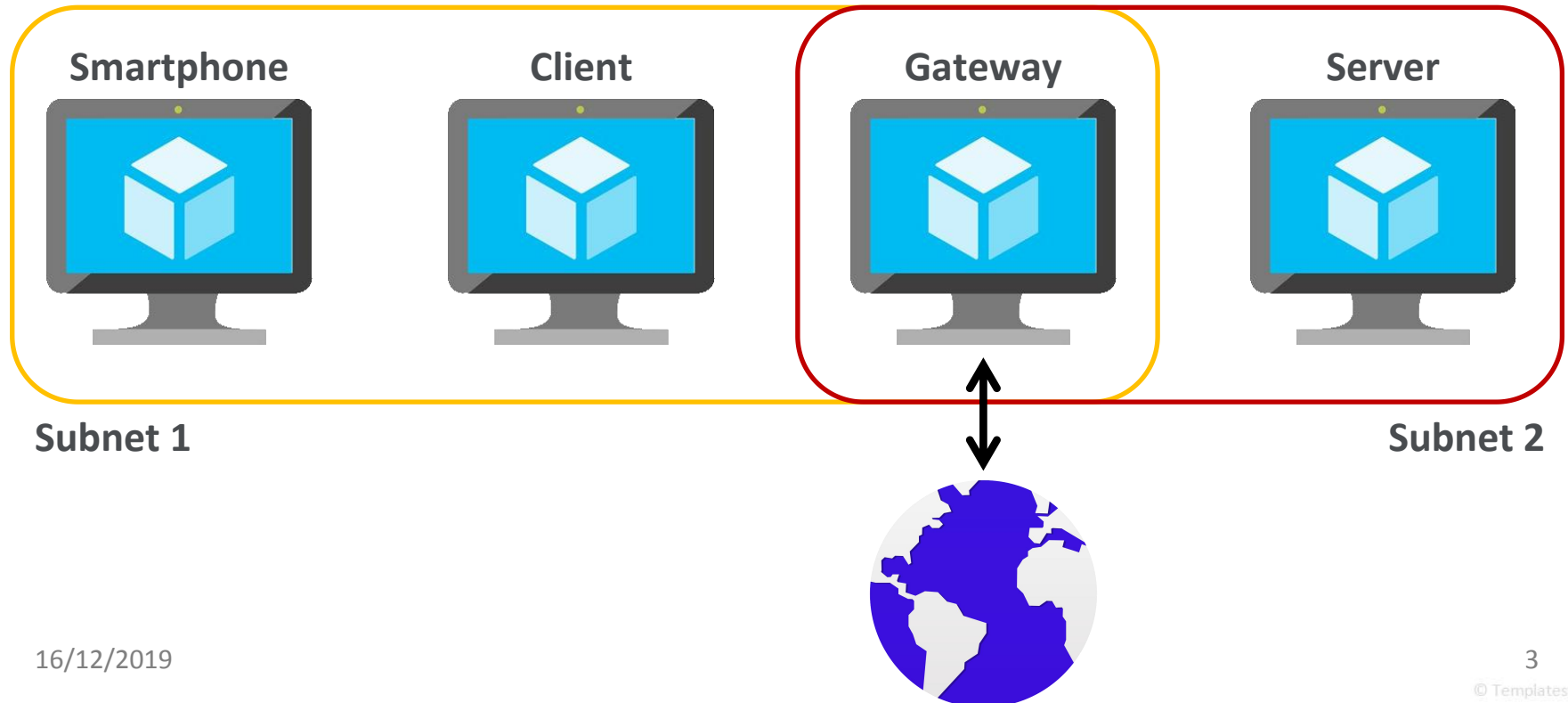


System architecture





System architecture - VMs





Solution

- Password strength, XSS, SQLI
- 2FA
- Custom Secure Channel
- Proximity



Password strength, XSS, SQLI

- Not the focus of our project, but still:
 - we use PHP's **password_hash** which uses **bcrypt**.
 - we require a minimum of 10 characters with lowercase, uppercase, digit and symbol.
 - we escape HTML characters directly or indirectly provided by a user when output to the screen.
 - we use SQL prepared statements.

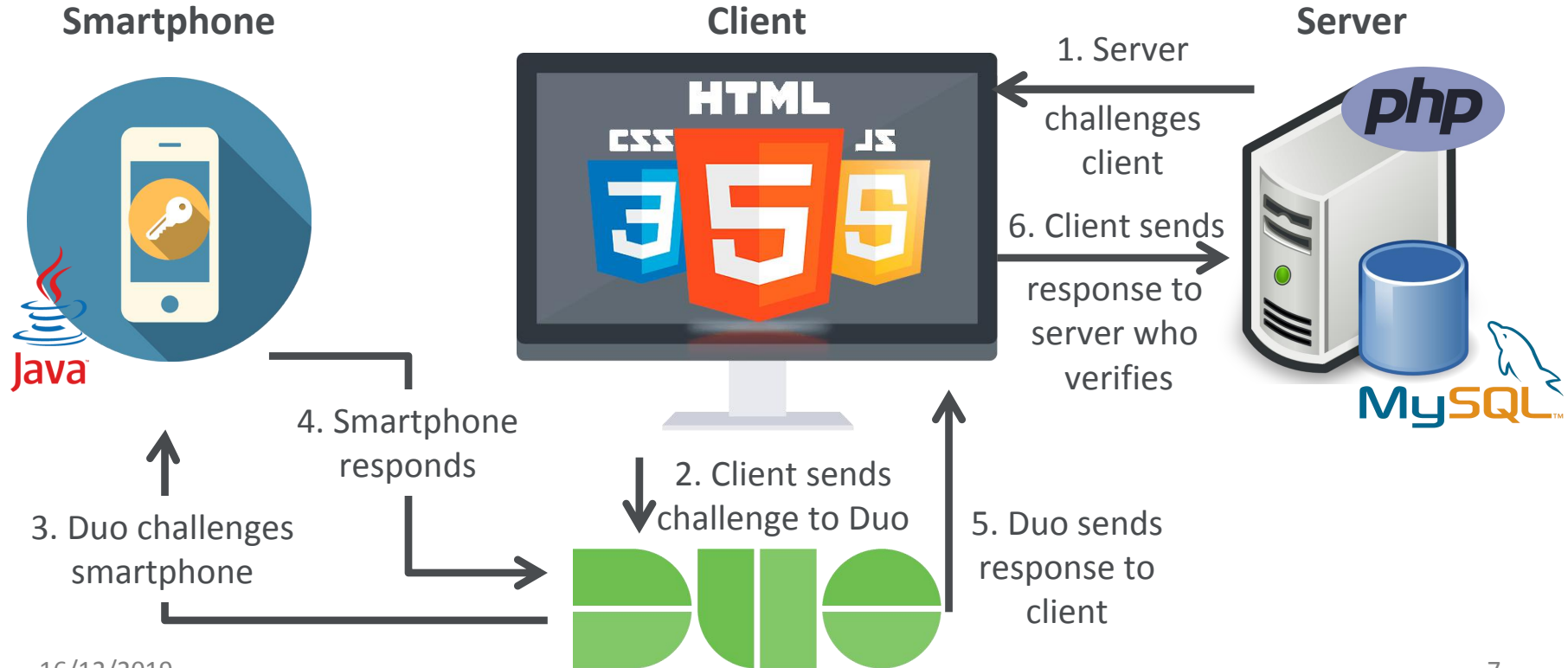


Solution

- Password strength, XSS, SQLI
- 2FA
- Custom Secure Channel
- Proximity



Two-factor authentication





Solution

- Password strength, XSS, SQLI
- 2FA
- Custom Secure Channel
- Proximity



Diffie-Hellman



Private = 5



$(6^5) \text{ MOD } 13$
 $(7776) \text{ MOD } 13$
Public = 2



$(9^5) \text{ MOD } 13$
 $(59049) \text{ MOD } 13$
Shared Secret = 3

Agree upon two numbers:

P Prime Number **13**

G Generator of P **6**

P is large, e.g. 2048 bits

Randomly generate a Private Key

G is a primitive root modulo P

Calculate Public Key:

$(G^{\text{Private}}) \text{ MOD } P$

Exchange Public Keys

Calculate the Shared Secret
 $(\text{Shared Public}^{\text{Private}}) \text{ MOD } P$



Private = 4



$(6^4) \text{ MOD } 13$
 $(1296) \text{ MOD } 13$
Public = 9



$(2^4) \text{ MOD } 13$
 $(16) \text{ MOD } 13$
Shared Secret = 3

Private is
large and
random

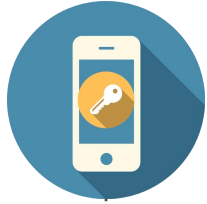


Ephemeral Diffie-Hellman (DHE)

- Grants **perfect forward secrecy** by computing new private and public values for each session
 - If a session key is discovered only that session is compromised.
 - If a long-term private key is compromised, past sessions are not compromised.



Ephemeral Diffie-Hellman with RSA (DHE-RSA)



Client acts as middle man
due to the lack of JS crypto
and smartphone proximity



* clicks Register button *

login to smartphone

request DH

1. Smartphone verifies if RSA pub.'s certificate is signed by a CA. *(not implemented)*
2. Verifies DH pub. key signature
2. Smartphone computes its own DH pub. and shared secret.

P, G, n° bits of P, server DH pub. key
(plain and signed), RSA pub. key

P, G, n° bits of P, server DH pub. key
(plain and signed), RSA pub. key

Smartphone's DH pub. key (plain and
signed)

Smartphone's DH pub. key (plain and
signed)

Request encrypt [username, password]

Ek[username, password, RSA public
key]

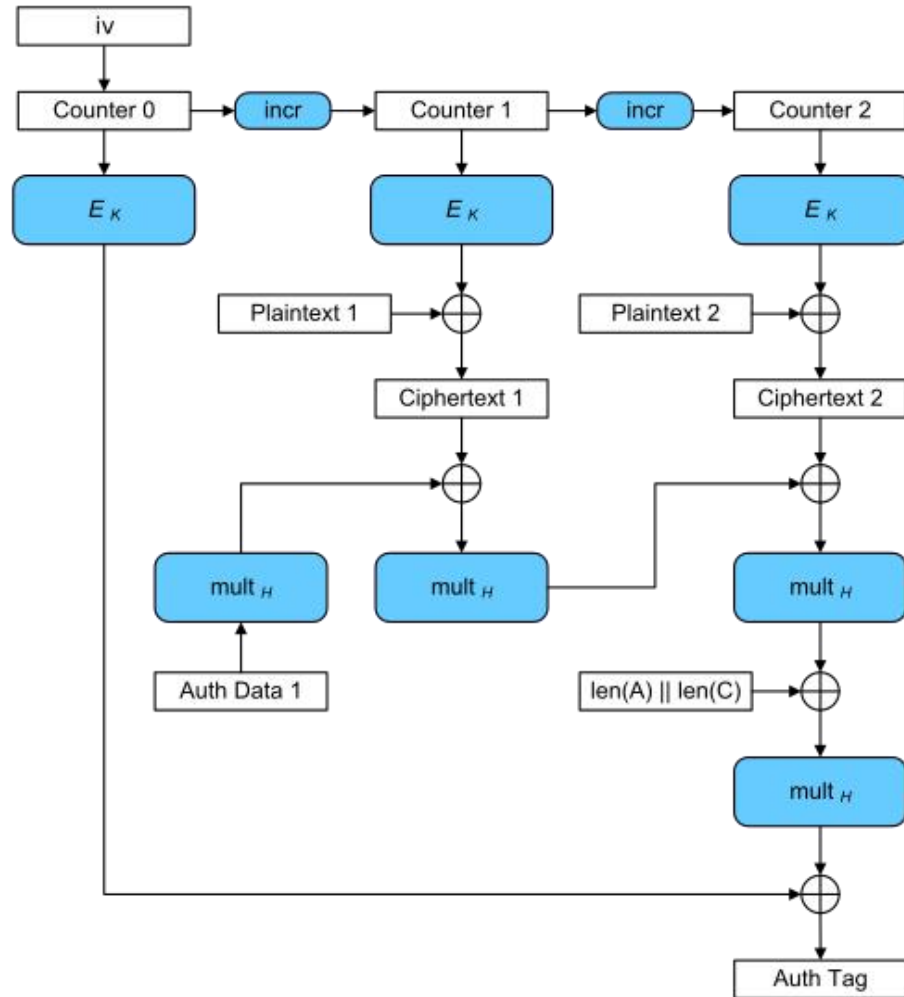
Ek[username, password, RSA public
key]

1. Server verifies DH pub. key signature except on registration
2. Server computes shared secret.



AES Galois/Counter Mode (AES-GCM)

- DH only finds a shared secret that can be used as a symmetric key, it is not an encryption algorithm.
- GCM is an authentication encryption mode of operation, it is composed by two separate functions: one for encryption (AES-CTR) and one for integrity and authentication (GMAC).





Solution

- Password strength, XSS, SQLI
- 2FA
- Custom Secure Channel
- Proximity



Proximity

- The client can only communicate with the server with the smartphone close by.
 - Smartphone contains RSA private key and all cryptographic methods.
- Client application logs out as soon as connection with the smartphone is lost.
- Encrypted .txt files are decrypted to volatile memory only (garbage collected as soon as smartphone connection is lost). *(not implemented)*
- Decrypted binary files are encrypted again as soon as smartphone connection is lost. *(not implemented)*



Conclusion

We implemented:

- Duo 2FA
 - But did not associate the user's smartphone with Duo at register.
- DHE-RSA-AES256-GCM
 - But did not implement freshness
 - Did not verify the server's RSA public key authenticity
 - Did not encrypt HTTP headers (including the session cookie)
- Proximity
 - But did not encrypt / decrypt files