skyrise

// Bezpieczeństwo aplikacji Android

Komunikacja klient - serwer

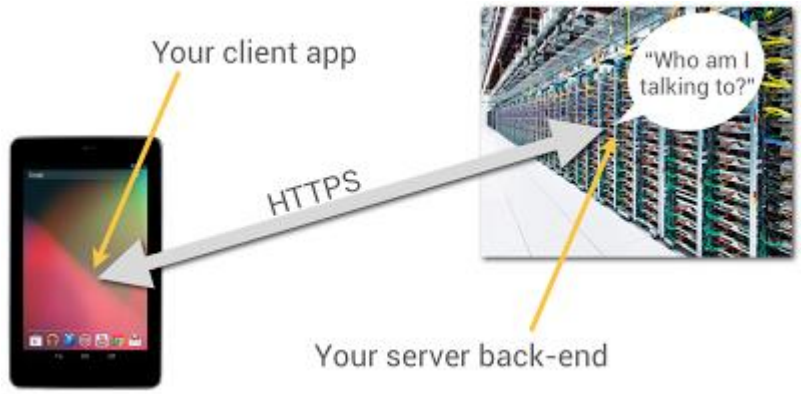
Autentykacja aplikacji

Autentykacja aplikacji

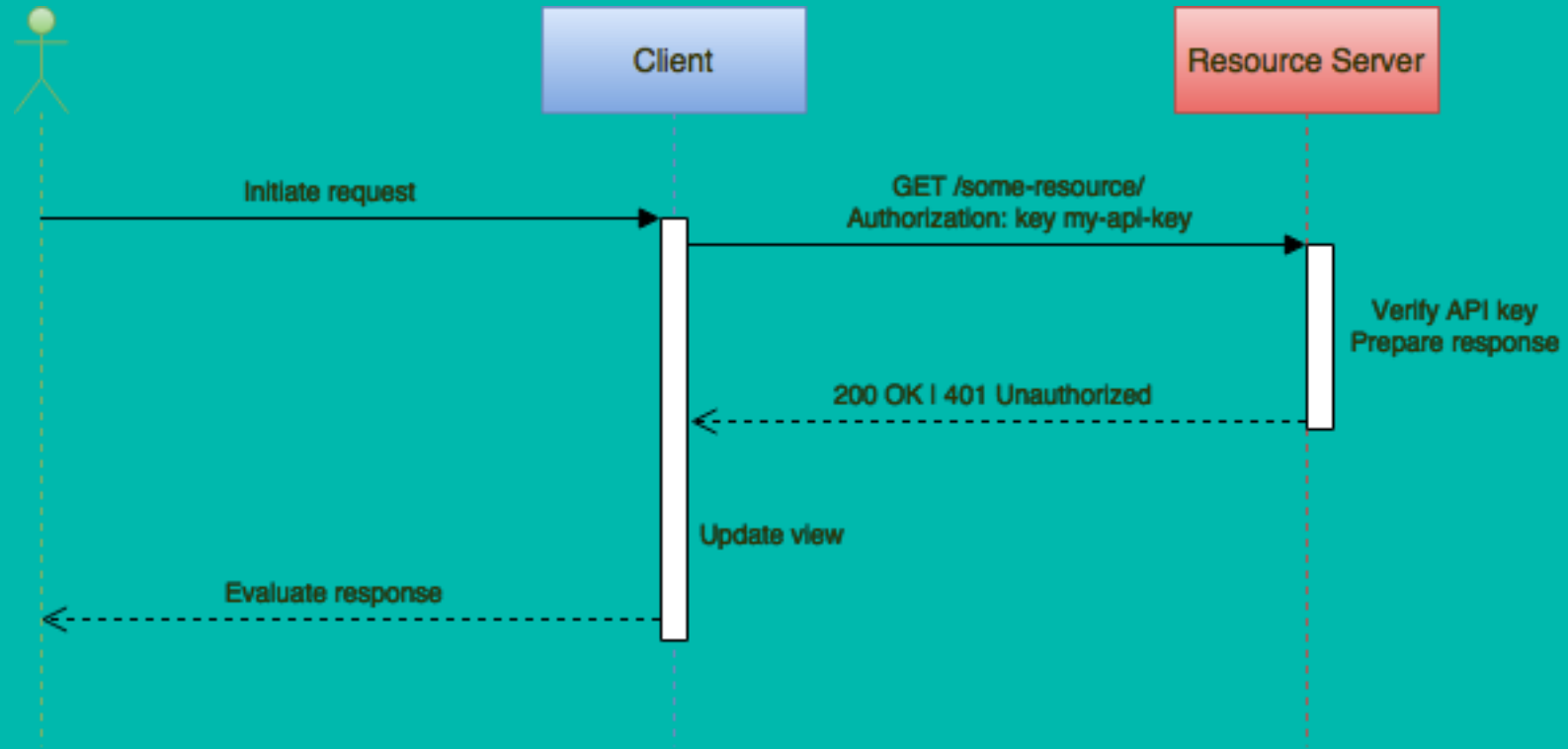
Cel: Ograniczenie zapytań naszego API do zaufanych klientów

Autentykacja aplikacji

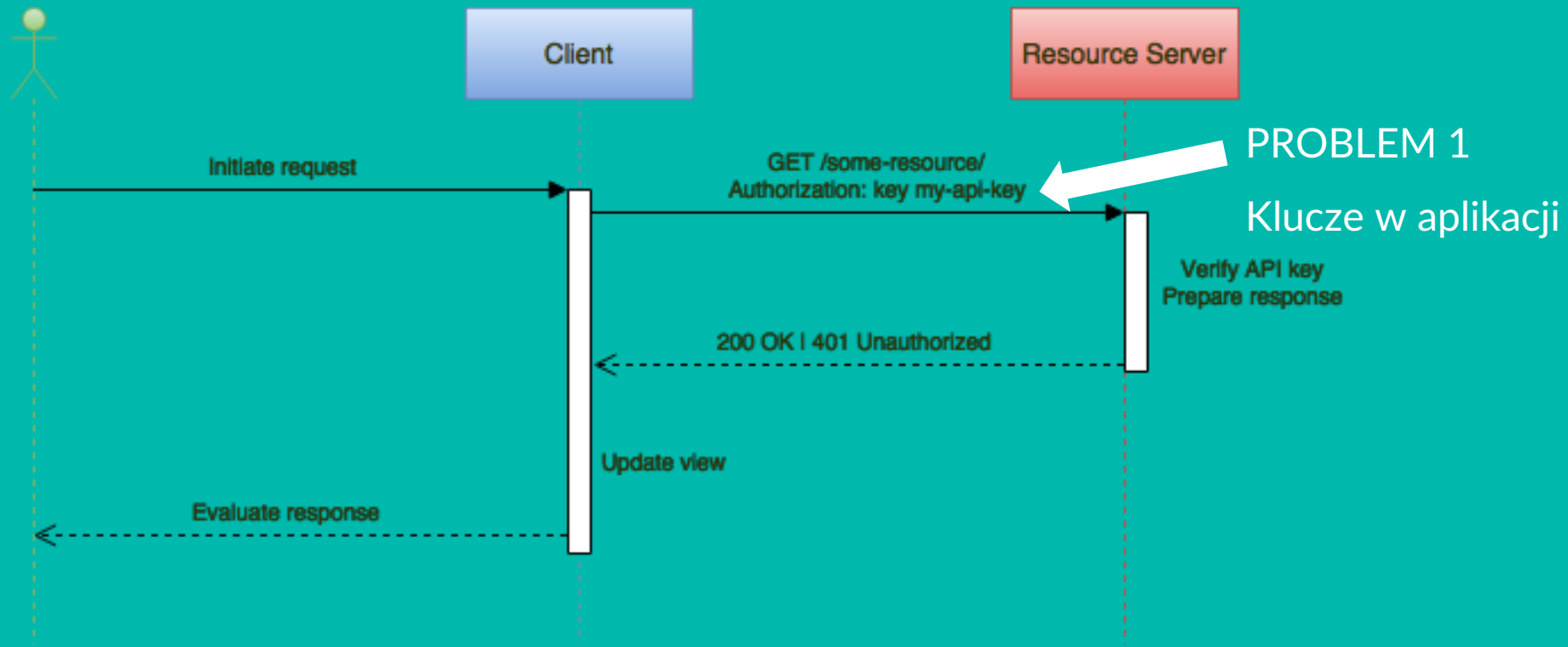
Cel: Ograniczenie zapytań naszego API do zaufanych klientów



Autentykacja tokenem



Autentykacja tokenem



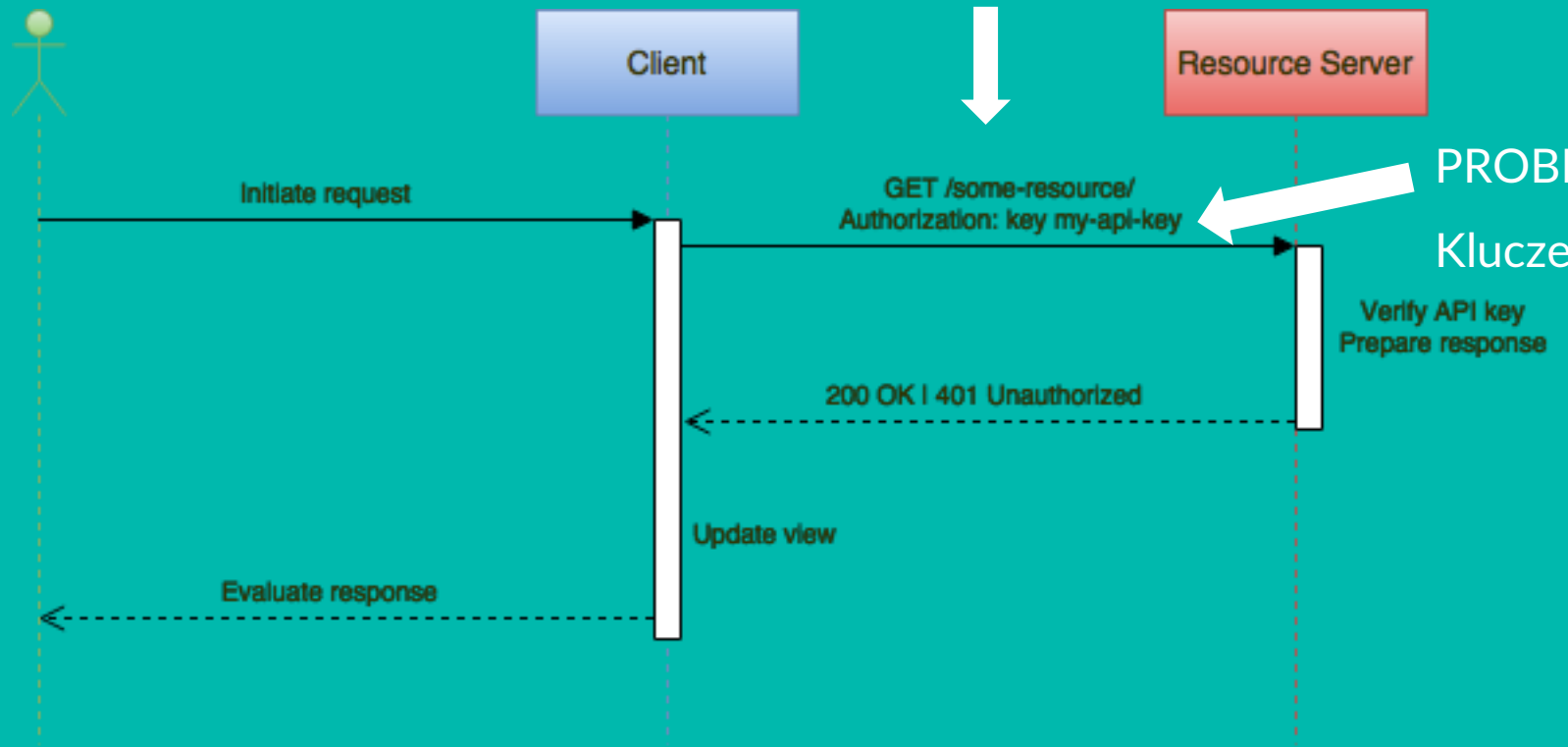
PROBLEM 1

Klucze w aplikacji

Autentykacja tokenem

PROBLEM 2

Man-In-The-Middle Attack



PROBLEM 1

Klucze w aplikacji

Bezpieczeństwo kluczy

Bezpieczeństwo kluczy

- Aplikacja (klient) może zostać pobrana jako plik .apk i zdekompilowana

Bezpieczeństwo kluczy

- Aplikacja (klient) może zostać pobrana jako plik .apk i zdekompilowana
- Ukrywanie klucza utrudnia, ale nie uniemożliwia jego odczytania

Bezpieczeństwo kluczy

- Aplikacja (klient) może zostać pobrana jako plik .apk i zdekompilowana
- Ukrywanie klucza utrudnia, ale nie uniemożliwia jego odczytania

- Przykład (negatywny): Parkanizer

Bezpieczeństwo kluczy

- Aplikacja (klient) może zostać pobrana jako plik .apk i zdekompilowana
- Ukrywanie klucza utrudnia, ale nie uniemożliwia jego odczytania

- Przykład (negatywny): Parkanizer
- Przykład (pozytywny):

Bezpieczeństwo kluczy

- Aplikacja (klient) może zostać pobrana jako plik .apk i zdekompilowana
- Ukrywanie klucza utrudnia, ale nie uniemożliwia jego odczytania

- Przykład (negatywny): Parkanizer
- Przykład (pozytywny): BRAK ☻

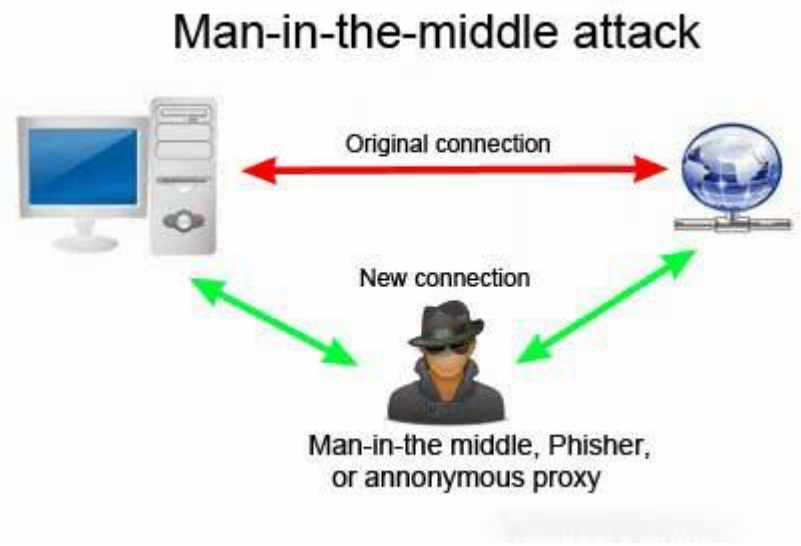
Bezpieczeństwo kluczy

- Aplikacja (klient) może zostać pobrana jako plik .apk i zdekompilowana
- Ukrywanie klucza utrudnia, ale nie uniemożliwia jego odczytania

- Przykład (negatywny): Parkanizer
- Przykład (pozytywny): BRAK ☻

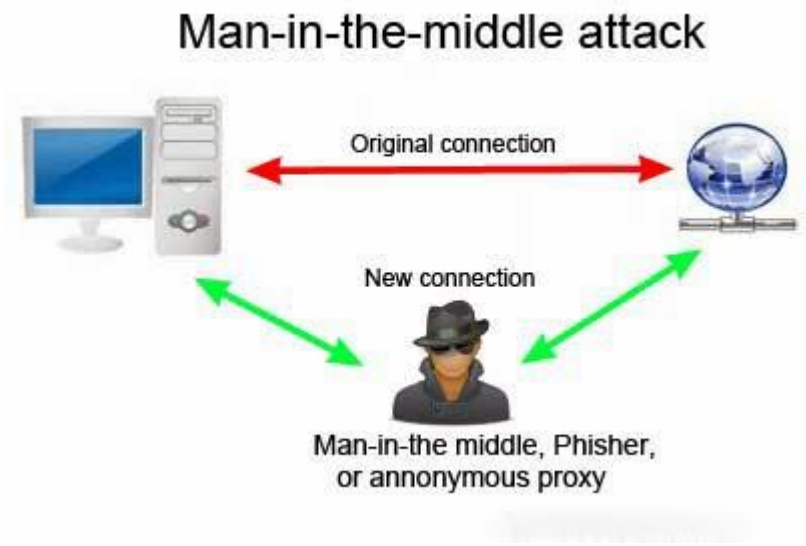
- Przykłady utrudniania: OWASP Crackmes

Man-In-The-Middle Attack



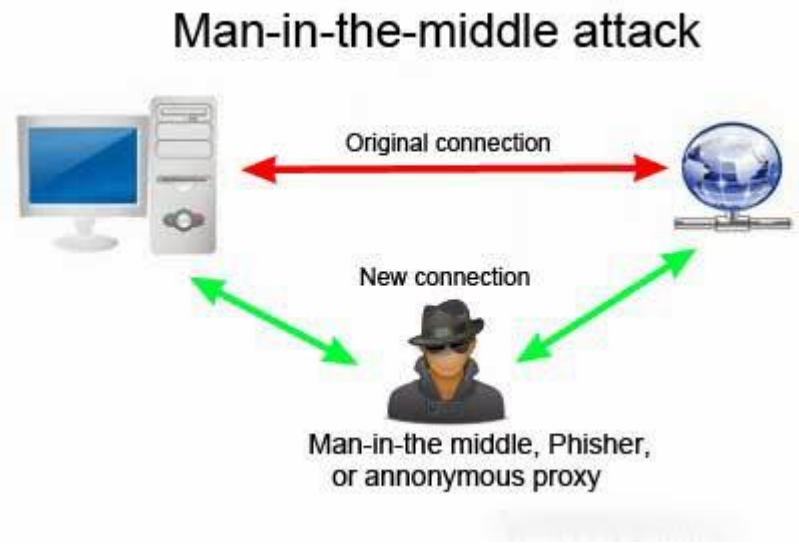
Man-In-The-Middle Attack

- Komunikacja klient – serwer jest podsłuchiwana przez „trzecią stronę”



Man-In-The-Middle Attack

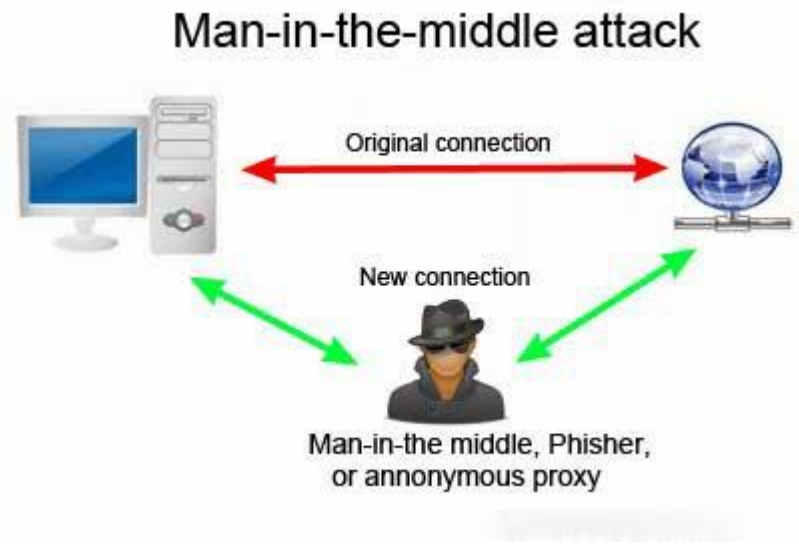
- Komunikacja klient – serwer jest podsłuchiwana przez „trzecią stronę”
- Android: Package Capture



Man-In-The-Middle Attack

- Komunikacja klient – serwer jest podsłuchiwana przez „trzecią stronę”
- Android: Package Capture
- Zabezpieczenie:

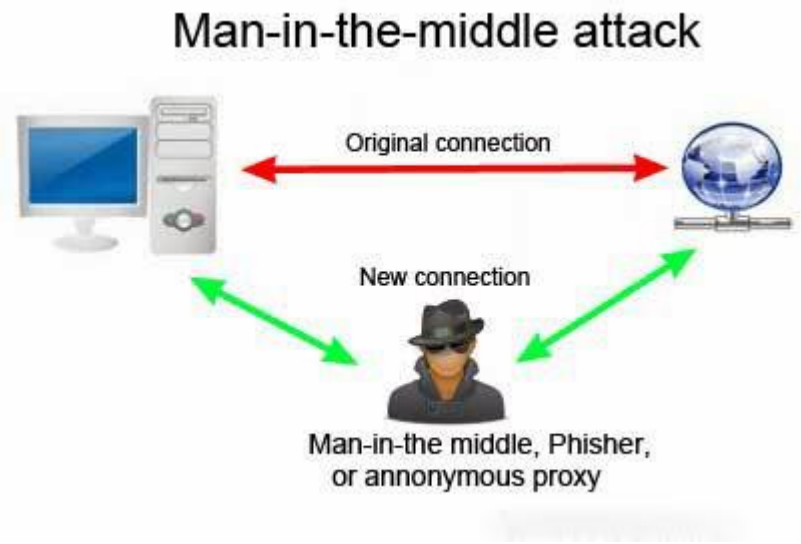
  https



Man-In-The-Middle Attack

- Komunikacja klient – serwer jest podsłuchiwana przez „trzecią stronę”
- Android: Package Capture
- Zabezpieczenie:

  ~~https~~ → możliwość podpięcia własnego certyfikatu

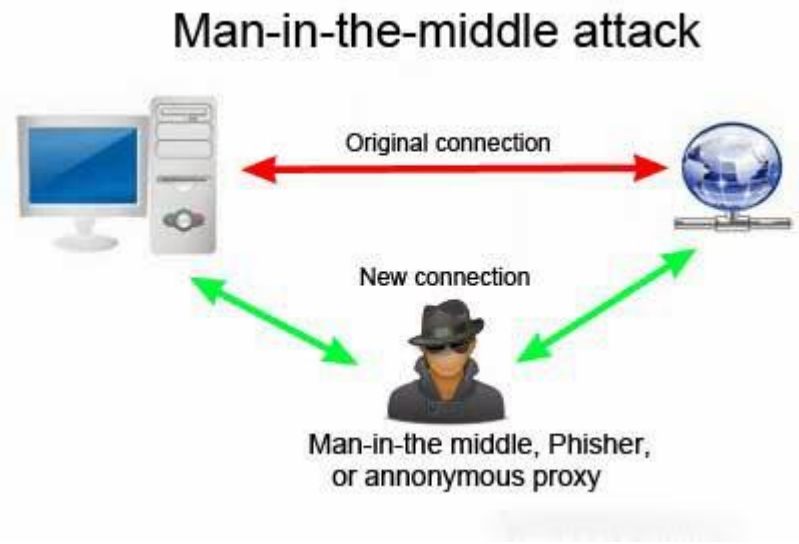


Man-In-The-Middle Attack

- Komunikacja klient – serwer jest podsłuchiwana przez „trzecią stronę”
- Android: Package Capture
- Zabezpieczenie:

  ~~https~~ → możliwość podpięcia własnego certyfikatu

  https + certificate pinning

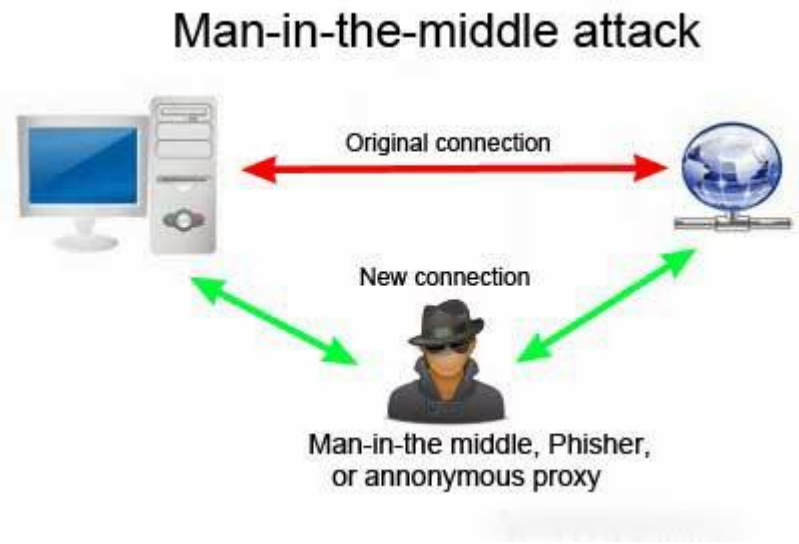


Man-In-The-Middle Attack

- Komunikacja klient – serwer jest podsłuchiwana przez „trzecią stronę”
- Android: Package Capture
- Zabezpieczenie:

  ~~https~~ → możliwość podpięcia własnego certyfikatu

  ~~https + certificate pinning~~ → all-trusting client



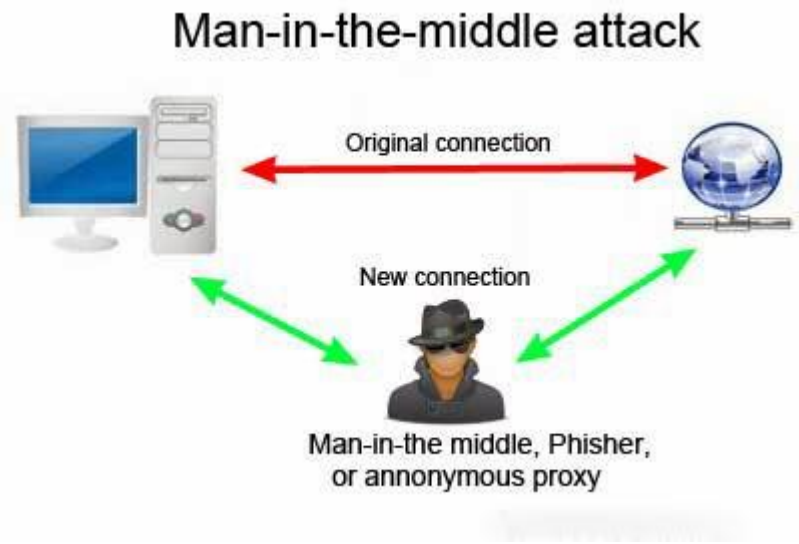
Man-In-The-Middle Attack

- Komunikacja klient – serwer jest podsłuchiwana przez „trzecią stronę”
- Android: Package Capture
- Zabezpieczenie:

  ~~https~~ → możliwość podpięcia własnego certyfikatu

  ~~https + certificate pinning~~ → all-trusting client

  https + certificate pinning + app signing certificate



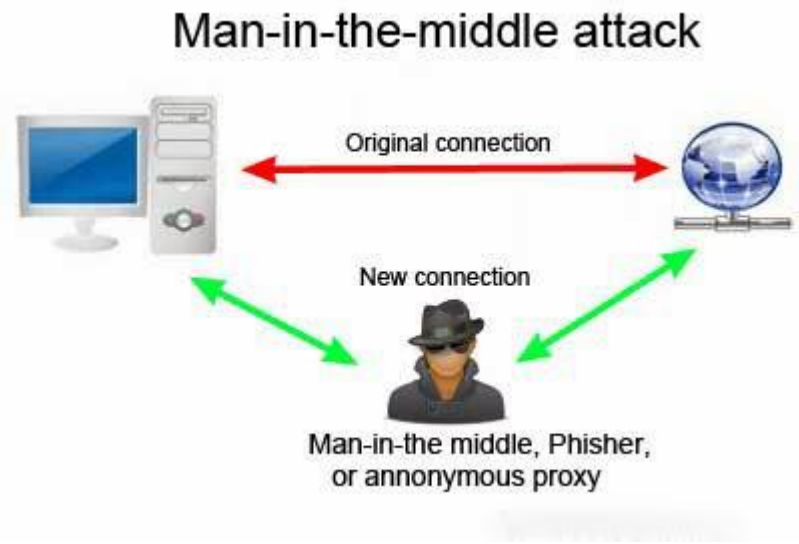
Man-In-The-Middle Attack

- Komunikacja klient – serwer jest podsłuchiwana przez „trzecią stronę”
- Android: Package Capture
- Zabezpieczenie:

  ~~https~~ → możliwość podpięcia własnego certyfikatu

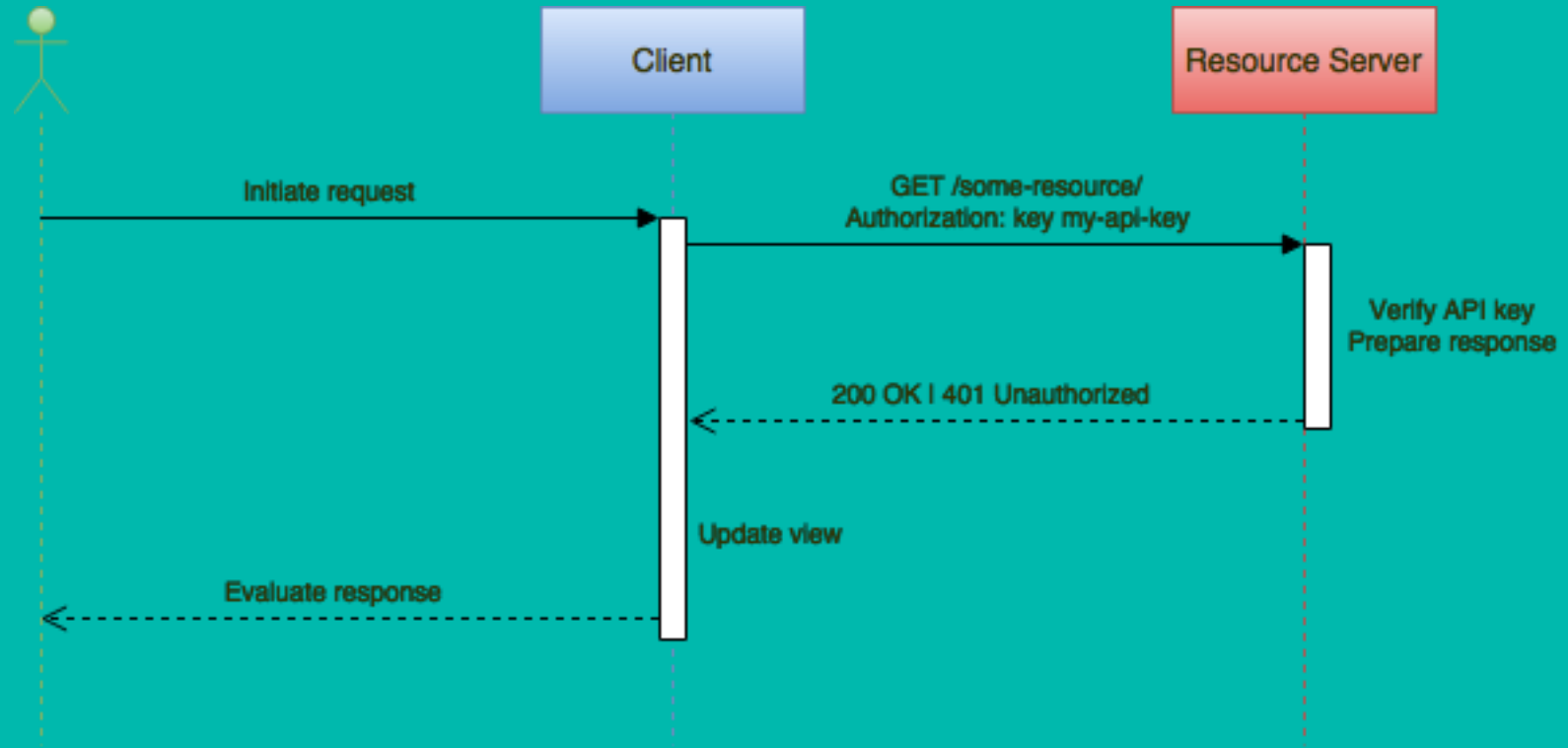
  ~~https + certificate pinning~~ → all-trusting client

  https + certificate pinning + app signing certificate (?)

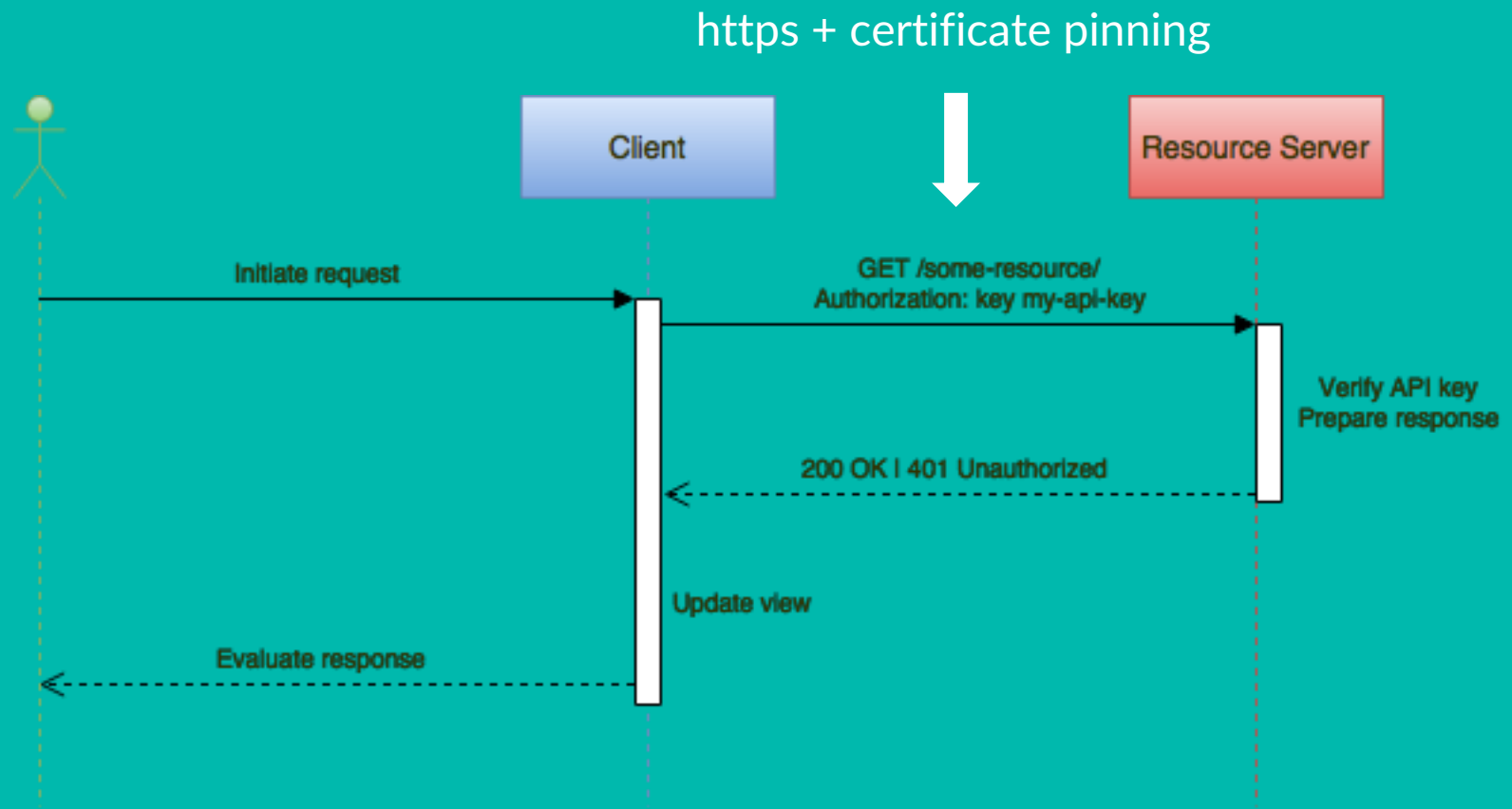


Autentykacja tokenem - wnioski

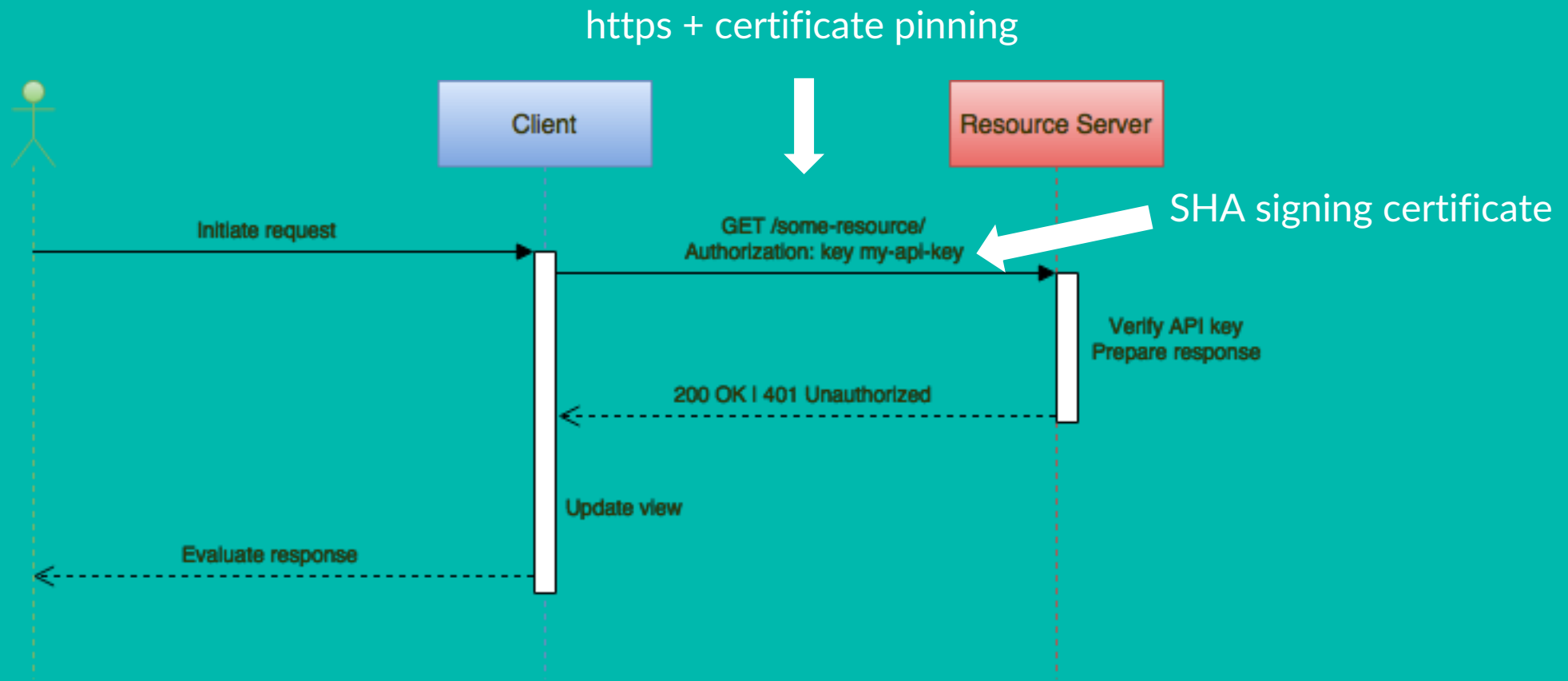
Autentykacja tokenem - wnioski



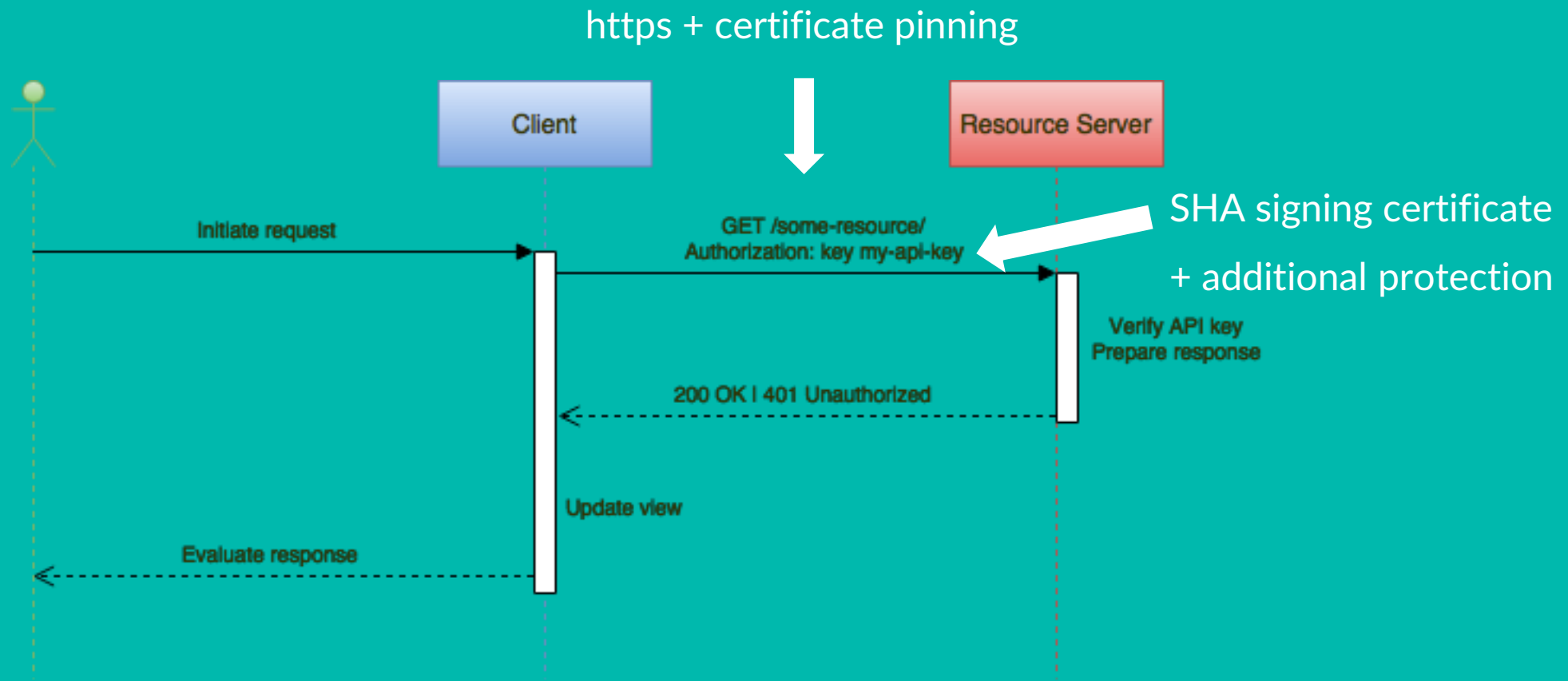
Autentykacja tokenem - wnioski



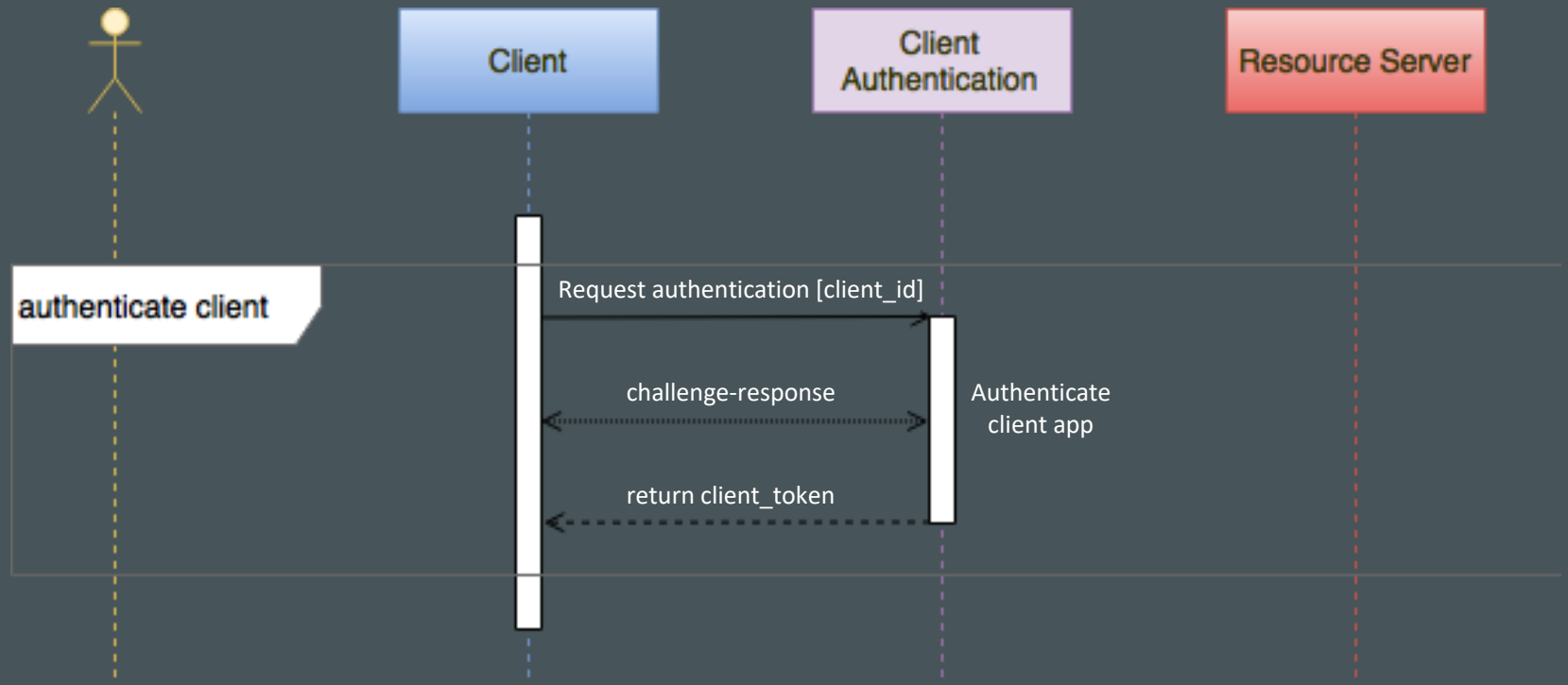
Autentykacja tokenem - wnioski



Autentykacja tokenem - wnioski



Challenge - response



Challenge - response

- Wersja uproszczona:

Challenge - response

- Wersja uproszczona:

  [Client]

Challenge - response

- Wersja uproszczona:

  [Client] → <request-token> → [Server]

Challenge - response

- Wersja uproszczona:

  [Client] → <request-token> → [Server]

  [Server]

Challenge - response

- Wersja uproszczona:

[Client]	→	<request-token>	→	[Server]
[Client]	←	<challenge>	←	[Server]

Challenge - response

- Wersja uproszczona:

  [Client] → <request-token> → [Server]

  [Client] ← <challenge> ← [Server]

  [Client]

Challenge - response

- Wersja uproszczona:

  [Client] → <request-token> → [Server]

  [Client] ← <challenge> ← [Server]

  [Client] <computes-challenge>

Challenge - response

- Wersja uproszczona:

  [Client] → <request-token> → [Server]

  [Client] ← <challenge> ← [Server]

  [Client] <computes-challenge>

  [Client]

Challenge - response

- Wersja uproszczona:

[Client]	→	<request-token>	→	[Server]
[Client]	←	<challenge>	←	[Server]
[Client]		<computes-challenge>		
[Client]	→	<challenge-response>	→	[Server]

Challenge - response

- Wersja uproszczona:

[Client] → <request-token> → [Server]

[Client] ← <challenge> ← [Server]

[Client] <computes-challenge>

[Client] → <challenge-response> → [Server]

[Server]

Challenge - response

- Wersja uproszczona:

[Client] → <request-token> → [Server]

[Client] ← <challenge> ← [Server]

[Client] <computes-challenge>

[Client] → <challenge-response> → [Server]

<verifies-challenge-response> [Server]

Challenge - response

- Wersja uproszczona:

[Client] → <request-token> → [Server]

[Client] ← <challenge> ← [Server]

[Client] <computes-challenge>

[Client] → <challenge-response> → [Server]

<verifies-challenge-response> [Server]

[Server]

Challenge - response

- Wersja uproszczona:

[Client]	→	<request-token>	→	[Server]
[Client]	←	<challenge>	←	[Server]
[Client]		<computes-challenge>		
[Client]	→	<challenge-response>	→	[Server]
		<verifies-challenge-response>		[Server]
		<HTTP 200/401>	←	[Server]

Challenge - response

- Wersja uproszczona:

[Client] → <request-token> → [Server]

[Client] ← <challenge> ← [Server]

[Client] <computes-challenge>

[Client] → <challenge-response> → [Server]

<verifies-challenge-response> [Server]

<HTTP 200/401> ← [Server]

[Server]

Challenge - response

- Wersja uproszczona:

[Client]	→	<request-token>	→	[Server]
[Client]	←	<challenge>	←	[Server]
[Client]		<computes-challenge>		
[Client]	→	<challenge-response>	→	[Server]
		<verifies-challenge-response>		[Server]
		<HTTP 200/401>	←	[Server]
[Client]	←	<client-token>	←	[Server]

Challenge - response

- Wersja rozwinięta: SCRAM (Salted-Challenge-Response-Authentication-Mechanism:

Challenge - response

- Wersja rozwinięta: SCRAM (Salted-Challenge-Response-Authentication-Mechanism

  [Client]

Challenge - response

- Wersja rozwinięta: SCRAM (Salted-Challenge-Response-Authentication-Mechanism

  [Client] first message → Hash(user_id, client_nonce)

Challenge - response

- Wersja rozwinięta: SCRAM (Salted-Challenge-Response-Authentication-Mechanism

  [Client] first message → Hash(user_id, client_nonce)

  [Server]

Challenge - response

- Wersja rozwinięta: SCRAM (Salted-Challenge-Response-Authentication-Mechanism

  [Client] first message → Hash(user_id, client_nonce)

  [Server] first message → Hash(client_nonce+server_nonce, server_salt, iterator)

Challenge - response

- Wersja rozwinięta: SCRAM (Salted-Challenge-Response-Authentication-Mechanism

  [Client] first message → Hash(user_id, client_nonce)

  [Server] first message → Hash(client_nonce+server_nonce, server_salt, iterator)

  [Client]

Challenge - response

- Wersja rozwinięta: SCRAM (Salted-Challenge-Response-Authentication-Mechanism

  [Client] first message → Hash(user_id, client_nonce)

  [Server] first message → Hash(client_nonce+server_nonce, server_salt, iterator)

  [Client] → <computes Hi(client_password, server_salt, iterator) function>

Challenge - response

- Wersja rozwinięta: SCRAM (Salted-Challenge-Response-Authentication-Mechanism

  [Client] first message → Hash(user_id, client_nonce)

  [Server] first message → Hash(client_nonce+server_nonce, server_salt, iterator)

  [Client] → <computes Hi(client_password, server_salt, iterator) function>

  [Client]

Challenge - response

- Wersja rozwinięta: SCRAM (Salted-Challenge-Response-Authentication-Mechanism

  [Client] first message → Hash(user_id, client_nonce)

  [Server] first message → Hash(client_nonce+server_nonce, server_salt, iterator)

  [Client] → <computes Hi(client_password, server_salt, iterator) function>

  [Client] → <computes Server final message>

Challenge - response

- Wersja rozwinięta: SCRAM (Salted-Challenge-Response-Authentication-Mechanism

  [Client] first message → Hash(user_id, client_nonce)

  [Server] first message → Hash(client_nonce+server_nonce, server_salt, iterator)

  [Client] → <computes Hi(client_password, server_salt, iterator) function>

  [Client] → <computes Server final message>

  [Client]

Challenge - response

- Wersja rozwinięta: SCRAM (Salted-Challenge-Response-Authentication-Mechanism

  [Client] first message → Hash(user_id, client_nonce)

  [Server] first message → Hash(client_nonce+server_nonce, server_salt, iterator)

  [Client] → <computes Hi(client_password, server_salt, iterator) function>

  [Client] → <computes Server final message>

  [Client] final message → Hash(client_server_nonce, client_proof(Hi() result))

Challenge - response

- Wersja rozwinięta: SCRAM (Salted-Challenge-Response-Authentication-Mechanism

  [Client] first message → Hash(user_id, client_nonce)

  [Server] first message → Hash(client_nonce+server_nonce, server_salt, iterator)

  [Client] → <computes Hi(client_password, server_salt, iterator) function>

  [Client] → <computes Server final message>

  [Client] final message → Hash(client_server_nonce, client_proof(Hi() result))

  [Server]

Challenge - response

- Wersja rozwinięta: SCRAM (Salted-Challenge-Response-Authentication-Mechanism

  [Client] first message → Hash(user_id, client_nonce)

  [Server] first message → Hash(client_nonce+server_nonce, server_salt, iterator)

  [Client] → <computes Hi(client_password, server_salt, iterator) function>

  [Client] → <computes Server final message>

  [Client] final message → Hash(client_server_nonce, client_proof(Hi() result))

  [Server] → <computes Hi() func + compares with client_proof>

Challenge - response

- Wersja rozwinięta: SCRAM (Salted-Challenge-Response-Authentication-Mechanism

  [Client] first message → Hash(user_id, client_nonce)

  [Server] first message → Hash(client_nonce+server_nonce, server_salt, iterator)

  [Client] → <computes Hi(client_password, server_salt, iterator) function>

  [Client] → <computes Server final message>

  [Client] final message → Hash(client_server_nonce, client_proof(Hi() result))

  [Server] → <computes Hi() func + compares with client_proof> → client_authenticated

Challenge - response

- Wersja rozwinięta: SCRAM (Salted-Challenge-Response-Authentication-Mechanism

  [Client] first message → Hash(user_id, client_nonce)

  [Server] first message → Hash(client_nonce+server_nonce, server_salt, iterator)

  [Client] → <computes Hi(client_password, server_salt, iterator) function>

  [Client] → <computes Server final message>

  [Client] final message → Hash(client_server_nonce, client_proof(Hi() result))

  [Server] → <computes Hi() func + compares with client_proof> → client_authenticated

  [Server]

Challenge - response

- Wersja rozwinięta: SCRAM (Salted-Challenge-Response-Authentication-Mechanism

  [Client] first message → Hash(user_id, client_nonce)

  [Server] first message → Hash(client_nonce+server_nonce, server_salt, iterator)

  [Client] → <computes Hi(client_password, server_salt, iterator) function>

  [Client] → <computes Server final message>

  [Client] final message → Hash(client_server_nonce, client_proof(Hi() result))

  [Server] → <computes Hi() func + compares with client_proof> → client_authenticated

  [Server] final message -> Hash(server_signature[salted_password, func(messages), token]

Challenge - response

- Wersja rozwinięta: SCRAM (Salted-Challenge-Response-Authentication-Mechanism

  [Client] first message → Hash(user_id, client_nonce)

  [Server] first message → Hash(client_nonce+server_nonce, server_salt, iterator)

  [Client] → <computes Hi(client_password, server_salt, iterator) function>

  [Client] → <computes Server final message>

  [Client] final message → Hash(client_server_nonce, client_proof(Hi() result))

  [Server] → <computes Hi() func + compares with client_proof> → client_authenticated

  [Server] final message -> Hash(server_signature[salted_password, func(messages), token]

  [Client]

Challenge - response

- Wersja rozwinięta: SCRAM (Salted-Challenge-Response-Authentication-Mechanism

  [Client] first message → Hash(user_id, client_nonce)

  [Server] first message → Hash(client_nonce+server_nonce, server_salt, iterator)

  [Client] → <computes Hi(client_password, server_salt, iterator) function>

  [Client] → <computes Server final message>

  [Client] final message → Hash(client_server_nonce, client_proof(Hi() result))

  [Server] → <computes Hi() func + compares with client_proof> → client_authenticated

  [Server] final message -> Hash(server_signature[salted_password, func(messages), token]

  [Client] → <compares computed and received server message>

Challenge - response

- Wersja rozwinięta: SCRAM (Salted-Challenge-Response-Authentication-Mechanism

  [Client] first message → Hash(user_id, client_nonce)

  [Server] first message → Hash(client_nonce+server_nonce, server_salt, iterator)

  [Client] → <computes Hi(client_password, server_salt, iterator) function>

  [Client] → <computes Server final message>

  [Client] final message → Hash(client_server_nonce, client_proof(Hi() result))

  [Server] → <computes Hi() func + compares with client_proof> → client_authenticated

  [Server] final message -> Hash(server_signature[salted_password, func(messages), token]

  [Client] → <compares computed and received server message> → server authenticated

Funkcja Hi (PBKDF2)

Funkcja Hi (PBKDF2)

Password-Base-Key-Derivation-Function 2

Funkcja Hi (PBKDF2)

Password-Base-Key-Derivation-Function 2

- Funkcja, która generuje klucz pochodny za pomocą n iteracji

Funkcja Hi (PBKDF2)

Password-Base-Key-Derivation-Function 2

- Funkcja, która generuje klucz pochodny za pomocą n iteracji
- Ogranicza ataki brute-force

Funkcja Hi (PBKDF2)

Password-Base-Key-Derivation-Function 2

- Funkcja, która generuje klucz pochodny za pomocą n iteracji
- Ogranicza ataki brute-force
- Zalecane min. 4000 iteracji (LastPass – 10000 iter. -browser, 100 000 iter. - server

Funkcja Hi (PBKDF2)

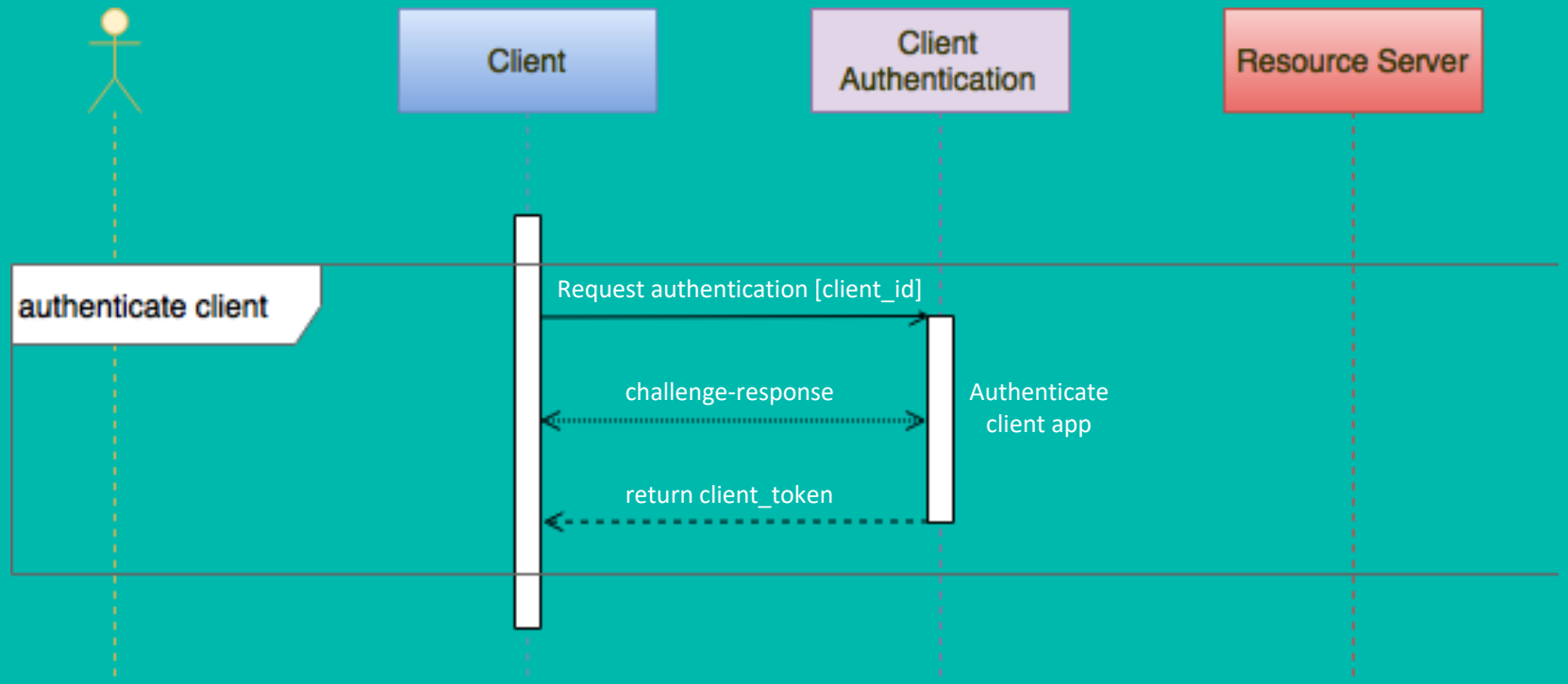
Password-Base-Key-Derivation-Function 2

- Funkcja, która generuje klucz pochodny za pomocą n iteracji
- Ogranicza ataki brute-force
- Zalecane min. 4000 iteracji (LastPass – 10000 iter. -browser, 100 000 iter. - server

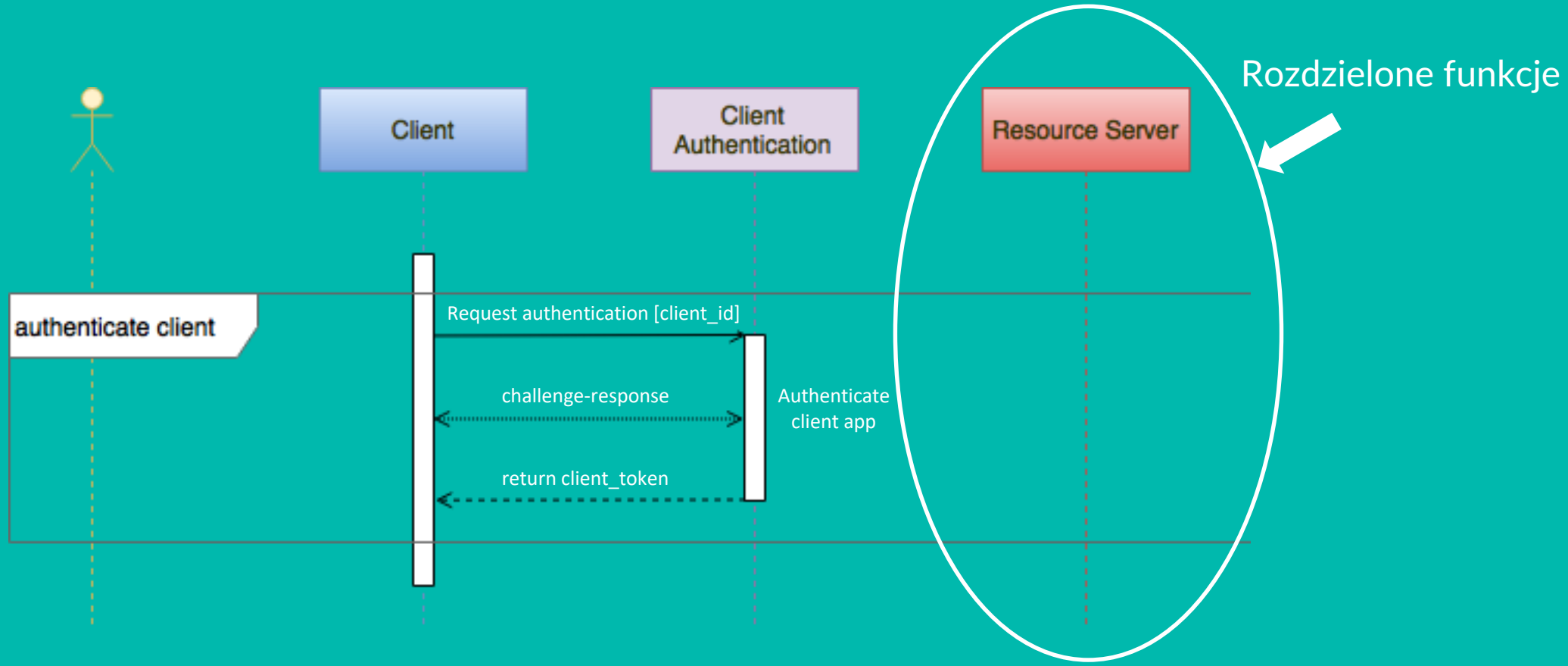
```
Hi(str, salt, i):
                U1   := HMAC(str, salt + INT(1))
                U2   := HMAC(str, U1)
                (...)
                Ui-1 := HMAC(str, Ui-2)
                Ui   := HMAC(str, Ui-1)
Hi := U1 XOR U2 XOR ... XOR Ui
```

Challenge–response - wnioski

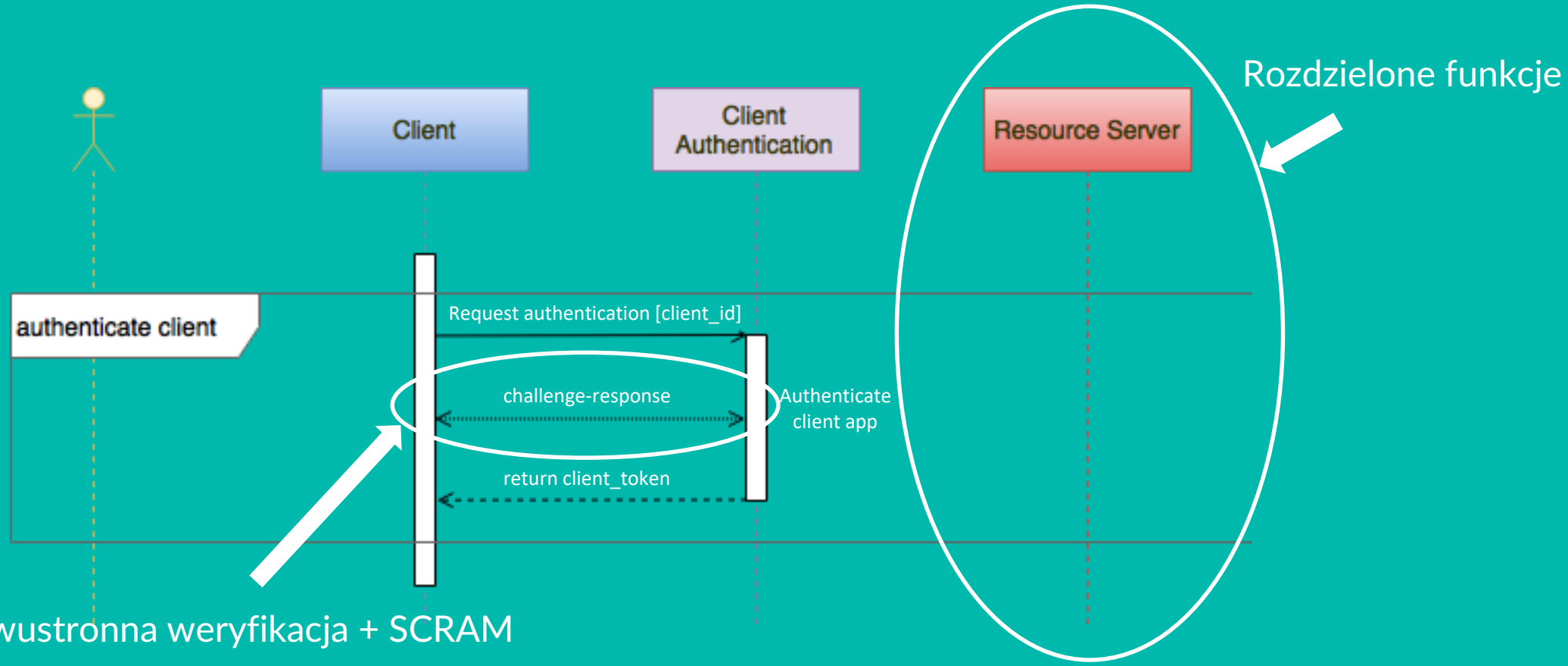
Challenge–response - wnioski



Challenge–response - wnioski



Challenge–response - wnioski



Rozdzielone funkcje

Dwustronna weryfikacja + SCRAM

Session token

Session token

- Po poprawnej weryfikacji aplikacja otrzymuje access_token + refresh_token

Session token

- Po poprawnej weryfikacji aplikacja otrzymuje access_token + refresh_token
- Access_token ma ograniczony czas życia

Session token

- Po poprawnej weryfikacji aplikacja otrzymuje access_token + refresh_token
- Access_token ma ograniczony czas życia
- Do komunikacji używany jest access_token + client_id (dodatkowe zabezpieczenie)

Session token

- Po poprawnej weryfikacji aplikacja otrzymuje access_token + refresh_token
- Access_token ma ograniczony czas życia
- Do komunikacji używany jest access_token + client_id (dodatkowe zabezpieczenie)
- Jeżeli access_token jest bliski przedawnienia, klient odświeża go za pomocą refresh_token'a

Podsumowanie

Podsumowanie

- Komunikacja klient–serwer odbywa się za pomocą https z walidacją certyfikatu

Podsumowanie

- Komunikacja klient–serwer odbywa się za pomocą https z walidacją certyfikatu
- Aplikacja [klient] autentykuje się charakterystycznym dla danego build’u kluczem

Podsumowanie

- Komunikacja klient–serwer odbywa się za pomocą https z walidacją certyfikatu
- Aplikacja [klient] autentykuje się charakterystycznym dla danego build'u kluczem
- Występuje dodatkowa weryfikacja za pomocą dwustronnego challenge-response'a

Podsumowanie

- Komunikacja klient–serwer odbywa się za pomocą https z walidacją certyfikatu
- Aplikacja [klient] autentykuje się charakterystycznym dla danego build'u kluczem
- Występuje dodatkowa weryfikacja za pomocą dwustronnego challenge-response'a
- Po poprawnej autentykacji komunikacja odbywa się za pomocą session_tokena

skyrise

Adrian Defus

Skyrise Sp. z o.o.
Ul. Sobieskiego 2, 40-081 Katowice

www.skyrise.tech

Tel.: +48 32 728 12 50