

# Slajd 1 - Cel

## Tekst do przeczytania podczas prezentacji

Celem tej części projektu jest zabezpieczenie dwóch różnych rodzajów danych. Pierwszym są deskryptory, czyli krótkie, losowe wartości o długości 128 bitów, a więc 16 bajtów. Drugim jest ciągły strumień multimedialny MPEG Transport Stream.

W przypadku deskryptorów istotne jest ograniczenie narzutu, ponieważ sam zabezpieczany element jest bardzo mały. W przypadku strumienia TS najważniejsze są natomiast przepustowość i niskie opóźnienie. Projekt musi więc zapewnić poufność i integralność danych, ale jednocześnie nie może znacząco spowolnić transmisji.

**Przejście:** Z tego powodu rozdzielamy cięższe operacje postkwantowe od szybkiego szyfrowania właściwych danych.

## Slajd 2 - Zastosowane algorytmy

### Tekst do przeczytania podczas prezentacji

W projekcie wykorzystujemy trzy główne algorytmy, ale każdy z nich pełni inną funkcję.

ML-KEM służy do bezpiecznego ustanowienia wspólnego sekretu między pojazdem a RSU. Pojazd wykonuje enkapsulację przy użyciu publicznego klucza RSU, a RSU wykonuje dekapulację swoim kluczem prywatnym. W efekcie obie strony uzyskują ten sam sekret, z którego później wyprowadzane są klucze sesyjne.

ML-DSA jest natomiast algorytmem podpisu cyfrowego. Nie służy do uzgadniania sekretu. Jego zadaniem jest potwierdzenie tożsamości nadawcy oraz sprawdzenie, czy podpisany komunikat nie został zmieniony.

Najprościej można więc powiedzieć, że ML-KEM tworzy wspólny sekret, natomiast ML-DSA potwierdza, z kim się komunikujemy.

Po ustanowieniu sesji przechodzimy na AES-256-GCM. Jest to szybki szyfr symetryczny, w którym obie strony korzystają z tego samego klucza. Klucz ma długość 256 bitów, natomiast tryb GCM zapewnia jednocześnie poufność, integralność i autentyczność danych.

AES-GCM tworzy również tag uwierzytelniający. Jeżeli ktoś zmieni zaszyfrowane dane albo chroniony nagłówek, weryfikacja tagu zakończy się błędem i rekord zostanie odrzucony.

Ważną zasadą AES-GCM jest zakaz ponownego wykorzystania tej samej pary klucz-nonce. Dlatego w naszym projekcie nonce jest tworzony między innymi na podstawie rosnącego numeru sekwencyjnego.

Algorytmy postkwantowe stosujemy tylko przy ustanawianiu zaufania, a nie dla każdego pakietu TS. Dzięki temu ograniczamy obciążenie procesora, opóźnienie i rozmiar przesyłanych danych.

**Przejdźcie:** Po uzyskaniu wspólnego sekretu nie tworzymy jednak jednego klucza do całej komunikacji.

## Slajd 3 - Separacja kluczy

### Tekst do przeczytania podczas prezentacji

Z głównego sekretu sesji wyprowadzamy kilka niezależnych kluczy operacyjnych. Osobne klucze stosujemy dla deskryptorów, strumienia TS, komunikatów sterujących oraz procesu rotacji.

Dodatkowo rozdzielamy kierunek nadawania i odbioru. Oznacza to, że pojazd używa innego klucza do wysyłania danych i innego do ich odbierania.

Taka separacja ogranicza skutki ewentualnego ujawnienia jednego klucza. Przykładowo kompromitacja klucza używanego do strumienia wideo nie musi oznaczać przejęcia komunikatów sterujących ani deskryptorów.

Oddzielne klucze dla TX i RX zapobiegają także sytuacji, w której w obu kierunkach pojawiłby się ten sam numer sekwencyjny i doszłoby do ponownego użycia pary klucz-nonce.

Wartości nonce salt są dodatkowym elementem wykorzystywanym do tworzenia unikalnych wartości nonce dla kolejnych rekordów AES-GCM.

**Przejście:** Na kolejnym slajdzie pokazany jest proces, w którym powstaje główny sekret wykorzystywany do wyprowadzenia tych kluczy.

## Slajd 4 - Diagram przebiegu sesji

### Tekst do przeczytania podczas prezentacji

Proces rozpoczyna się od komunikatu beacon wysyłanego przez RSU. Zawiera on między innymi identyfikator RSU, losowy nonce, certyfikat ML-DSA, publiczny klucz ML-KEM oraz numer obowiązującej polityki kryptograficznej.

Pojazd najpierw weryfikuje certyfikat RSU i sprawdza, czy nie został on unieważniony. Następnie wykonuje enkapsulację ML-KEM. W tym momencie powstaje wspólny sekret oraz ciphertext, który może zostać przesłany do RSU.

Pojazd przygotowuje odpowiedź zawierającą ciphertext ML-KEM, identyfikator poświadczenia, nonce, timestamp oraz podpis ML-DSA. Podpis obejmuje cały istotny transkrypt komunikacji, dzięki czemu chroni również przed podmianą parametrów lub powtórzeniem starego komunikatu.

RSU najpierw sprawdza podpis i aktualność wiadomości. Dopiero później wykonuje dekapsulację ML-KEM i odzyskuje ten sam sekret, który został utworzony po stronie pojazdu.

ML-KEM odpowiada więc za uzyskanie wspólnego sekretu, natomiast ML-DSA potwierdza autentyczność stron i komunikatów. Po pomyślnej weryfikacji strony wyprowadzają klucze operacyjne i przechodzą na szyfrowanie AES-256-GCM.

Od tego momentu dane bieżące, takie jak deskryptory i strumień TS, są zabezpieczane szybko symetrycznie. Operacje postkwantowe są ponownie wykonywane dopiero przy ponownym onboardingu, zmianie strefy zaufania albo pełnej rotacji sesji.