Zastosowanie kryptografii postkwantowej w sieciach IoT

Tomasz Styn, Michał Pitera, Mateusz Okoń, Patryk Harasik

Akademia Górniczo-Hutnicza, Wydział Informatyki, Elektroniki i Telekomunikacji, Koło Naukowe Cipher, Kraków

Streszczenie: Artykuł porusza kwestie wprowadzenia postkwantowego algorytmu wymiany klucza kryptograficznego w kontekście sieci Internet of Things (IoT) na podstawie protokołu ZigBee. Coraz szersze zastosowanie internetu rzeczy powoduje konieczność zabezpieczenia urządzeń wchodzących w jego skład oraz komunikacji między nimi. Jest to kluczowa kwestia zarówno dla bezpieczeństwa organizacji wdrażających te technologie jako część infrastruktury, jak i użytkowników końcowych stosujących je w swoich domach. Bezpieczeństwo rozwiązania opiera się na implementacji mechanizmu ML-KEM-512 (Kyber) w celu enkapsulacji klucza symetrycznego stosowanego w szyfrowaniu komunikacji za pomocą AES-256 w trybie CTR. Za pomocą demonstracyjnej sieci IoT przeprowadzone zostały pomiary narzutów czasowych w komunikacji i wymianie klucza. Został oceniony poziom bezpieczeństwa zarówno standardowego, jak i proponowanego rozwiązania. Pozyskano wyniki porównania wydajności komunikacji przed wprowadzeniem zmodyfikowanego rozwiązania i po jego wprowadzeniu w celu oceny potencjału jego powszechnego wdrożenia. Zaobserwowana zmienność czasów przesyłu pakietów została powiązana z funkcjonowaniem sprzętu, co obaliło pierwotną hipotezę retransmisji. Badania wskazują na znaczące obciążenie zasobów sieci i urządzeń, a to pokazuje, że rozwiązanie w tej postaci znajdzie zastosowanie między kluczowymi węzłami o większych zasobach obliczeniowych lub w przypadku, gdy bezpieczeństwo przesyłanej informacji ma większy priorytet od prędkości jej transferu.

Słowa kluczowe: cyberbezpieczeństwo, kryptografia postkwantowa, internet rzeczy, ZigBee

APPLYING POST-QUANTUM CRYPTOGRAPHY IN IoT NETWORKS

Abstract: The advent of powerful quantum computing brings risks to the security of cryptographic systems as we know them today. Continuous growth of the already widespread IoT sphere emphasizes the need for securing devices it consists of, and the communication between them. This paper describes the process of implementing ML-KEM-512 (Kyber) – a post-quantum key encapsulation mechanism in the ZigBee protocol. Furthermore, an encryption layer utilizing AES-256 is introduced in a way that is compatible with existing infrastructure, as we feel that this aspect is the key to rapid and widespread adoption among consumers and businesses. Through the use of a demonstrative IoT network based on ZigBee, efficiency measurements were taken before and after implementing the proposed solution. Their analysis was carried out to judge the solution's viability. Transmission time fluctuations were found likely to be connected with hardware, rather than network retransmissions, as was initially theorized. The security level of the paper's proposed solution was evaluated in the context of the existing standard. Results show a significant increase in network and device resource usage, which leads us to believe the solution can find adoption between crucial IoT network nodes, where security of transmitted information takes priority over the transfer's speed.

Keywords: cybersecurity, post-quantum cryptography, Internet of Things, ZigBee

https://doi.org/10.7494/978-83-68219-80-7_11

1. Wstęp

W związku z wieloma korzyściami płynącymi z opracowywanych technologii komputerów kwantowych rozwój tej dziedziny nieustannie postępuje. Właśnie z tego powodu należy przyjrzeć się również powstałym zagrożeniom.

Najpoważniejszym ze względu na cyberbezpieczeństwo zagrożeniem jest możliwość realizacji algorytmu Shora (Shor 1997). Algorytm ten może zostać skutecznie wykorzystany do zaatakowania kryptosystemów RSA oraz El Gamal. Gwarantem bezpieczeństwa w przypadku pierwszego kryptosystemu jest złożoność obliczeniowa operacji rozkładu liczb na czynniki pierwsze. W RSA podczas generowania kluczy na jednym z etapów wykorzystywany jest iloczyn dużych liczb pierwszych p i q (Rivest i in. 1978). O ile mnożenie przez siebie dwóch czynników jest operacją stosunkowo prostą do wykonania, o tyle rozłożenie znanego iloczynu na czynniki – nazywane faktoryzacją – wymaga dużych zasobów obliczeniowych. Funkcje mające własność łatwego obliczenia i trudnego odwrócenia nazywane są funkcjami jednokierunkowymi lub zapadkowymi (Paar i Pelzl 2010). W przypadku kryptosystemu El Gamal funkcją jednokierunkową jest rozwiązanie równania $g^r \equiv x(mod\ p)$ dla r (El Gamal 1985).

Oba opisane gwaranty, po wykorzystaniu algorytmu Shora, są poddane redukcji złożoności obliczeniowej z obecnie wykładniczej do wielomianowej. Algorytm Shora pozwala z wysokim prawdopodobieństwem dokonać faktoryzacji iloczynu dużych liczb pierwszych – stosowanego w RSA – oraz, w przypadku kryptosystemu El Gamal, odzyskać z klucza publicznego element r.

Za pomocą algorytmu Shora można rozwiązać powyższe problemy dzięki wykorzystaniu funkcji pomocniczych. W przypadku RSA funkcją tą jest $f(a) = x^a(mod\ n)$, gdzie n jest iloczynem p i q, a x jest losowo wybieraną liczbą względnie pierwszą z n. Dla El Gamal funkcją tą jest $f(a, b) = g^a x^{(-b)}(mod\ p - 1)$. Obydwie funkcje są okresowe, a odnalezienie ich okresów odbywa się za pomocą kwantowej transformaty Fouriera. Znajomość okresu potrzebna jest do przeprowadzenia dalszych obliczeń wyłaniających prawdopodobną liczbę rozwiązującą problem (Shor 1997).

W odpowiedzi na nadchodzące zagrożenia w celu zabezpieczenia dotychczas stosowanych kryptosystemów tworzone i wprowadzane są algorytmy kryptografii postkwantowej, które kosztem wymogu większej mocy obliczeniowej mają za zadanie zapewnić bezpieczeństwo w obliczu możliwości realizacji algorytmu Shora.

W rozwiązaniu przedstawionym w niniejszej publikacji zastosowano mechanizm ML-KEM-512, potocznie zwany Kyber512, który został wyłoniony jako zwycięzca w kwietniu 2024 roku w ogłoszonym w roku 2016 konkursie NIST Post-Quantum Cryptography Standardization. Konkurs został powołany w celu ustandaryzowania kryptosystemów kryptografii postkwantowej oraz zbadania bezpieczeństwa nominowanych do konkursu algorytmów. Rywalizacja została podzielona na cztery rundy, w wyniku czego

wyłoniono łącznie pięć algorytmów w dwóch kategoriach – KEM: CRYSTALS-KYBER, HQC oraz DSA: CRYSTALS-DILITHIUM, FALCON, SPHINCS+.

KEM – *Key Encapsulation Mechanism* – jest podejściem, w którym jedna ze stron komunikacji jest odpowiedzialna za wygenerowanie losowego klucza symetrycznego i przekazanie go drugiej w postaci kapsułki kanałem publicznym. Oznacza to utworzenie szyfrogramu za pomocą klucza publicznego odbiorcy. Istnieje również inne niż zastosowane w implementacji ML-KEM-512 rozwiązanie z rodziny KEM – RSA-KEM. Odmiennym podejściem jest mechanizm KEX, w którym każda ze stron przyczynia się w pewnym stopniu do uzgodnienia klucza symetrycznego. W wyniku jawnej wymiany odpowiednio obliczonych składników klucza kanałem publicznym tworzony jest współdzielony przez obie strony wymiany sekret. Przykładem tego podejścia jest protokół Diffiego–Hellmana.

Rodzina kryptosystemów ML-KEM została ustandaryzowana w dokumencie FIPS 203 (NIST 2024). Należą do niej ML-KEM-512, ML-KEM-768 oraz ML-KEM-1024. Każda z wersji ML-KEM wykorzystuje ten sam algorytm z innymi wartościami parametrów, których rekomendacje również umieszczono w FIPS 203. Wybrany przez autorów ML-KEM-512 zapewnia bezpieczeństwo bitowe na poziomie przynajmniej 128 bitów, a rekomendowanymi wartościami parametrów są $n = 256, q = 3329, k = 2, \eta_1 = 3, \eta_2 = 2, d_1 = 10, d_2 = 4$ (NIST 2024).

Algorytm ML-KEM jest oparty na problemie MLWE (*Module Learning with Errors*), który z kolei jest oparty na problemach kratowych (Avanzi i in. 2021). Wykorzystanie problemów opartych na kratach gwarantuje bezpieczeństwo przed atakami z pomocą komputera kwantowego, ponieważ aktualnie nie istnieje kwantowy algorytm zdolny do przyspieszenia rozwiązywania tego typu problemów.

W ML-KEM klucz symetryczny jest kapsułkowany kluczem publicznym z dodatkiem wektora błędu o niewielkich wartościach. Zadaniem odbiorcy kapsułkowanego klucza jest jego odszyfrowanie. Ze względu na wektor błędu proces ten obarczony jest prawdopodobieństwem błędu, które w przypadku poszczególnych ML-KEM ma następujące wartości: ML-KEM-512 – $2^{-138.8}$%, ML-KEM-768 – $2^{-164.8}$% i ML-KEM-1024 – $2^{-1748.8}$%.

Celem prowadzonych badań jest analiza narzutów czasowych i wydajnościowych rozwiązań postkwantowych w istniejącej infrastrukturze IoT na podstawie powszechnie stosowanego protokołu ZigBee (ZigBee Alliance 2015). Można uznać, że rozwiązanie rozszerzające stosowany już protokół oraz możliwe do wdrożenia w aktualnie stosowanych urządzeniach stanowi kluczowy czynnik związany z jego rozpowszechnieniem.

Wybrany protokół ZigBee umożliwia tworzenie sieci typu *mesh* oraz zapewnia energooszczędność, co czyni go idealnym rozwiązaniem w zastosowaniach, gdzie różnorodne urządzenia zasilane bateriami o niskich pojemnościach mają komunikować

się w zróżnicowanych środowiskach. Jest w domach, w służbie zdrowia, a także w przemyśle. Z powodu wielkoskalowych aplikacji IoT wymaga odpowiedniego, długoterminowego zabezpieczenia.

Z technicznego punktu widzenia ZigBee opiera się na standardzie IEEE 802.15.4, który definiuje dwie najniższe warstwy modelu komunikacyjnego: warstwę fizyczną (PHY) – odpowiedzialną za transmisję danych w medium – oraz warstwę kontroli dostępu do medium (MAC) – odpowiedzialną za kontrolę dostępu do kanału radiowego za pomocą mechanizmu CSMA-CA lub LBT, a także synchronizację i zapewnienie niezawodnego mechanizmu transmisji. ZigBee rozszerza ten model o dodatkowe poziomy – warstwę sieciową (NWK) oraz warstwę aplikacji (APL).

W ramach sieci ZigBee urządzenia mogą pracować w jednym z trzech trybów, które określają ich rolę w topologii sieci:

- **Koordynator** – węzeł centralny lub nadrzędny w stosunku do innych węzłów w sieci, odpowiedzialny za inicjalizację i zarządzanie siecią. Przechowuje listę powiązanych urządzeń, obsługuje procesy asocjacji, dysocjacji oraz skanowania dostępnych urządzeń.
- **Router** – urządzenie pośrednie, odpowiedzialne za przekazywanie pakietów między urządzeniami końcowymi lub między urządzeniem końcowym a koordynatorem.
- **Urządzenie końcowe** – proste urządzenie, odpowiedzialne za monitorowanie i zbieranie danych lub podejmowanie określonych działań na podstawie poleceń użytkownika.

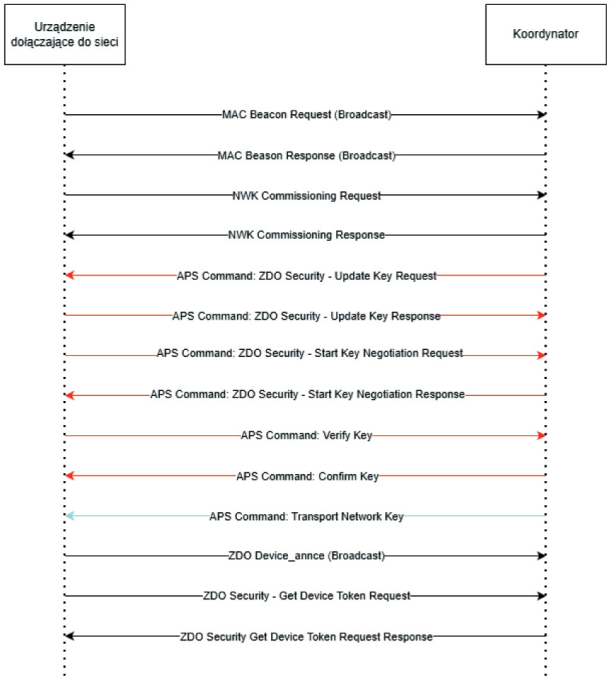
Mimo że standard ZigBee zapewnia szyfrowaną komunikację, zastosowane w nim protokoły uzgadniania klucza – ECDHE oraz SPEKE – nie są odporne na ataki z wykorzystaniem komputerów kwantowych. Do szyfrowania danych ZigBee stosuje algorytm AES-128.

W kwestii bezpieczeństwa protokół ZigBee opiera się na dwóch podstawowych rodzajach kluczy kryptograficznych: *link key* oraz *network key*. Klucze typu *link key* służą do szyfrowania ruchu w warstwie aplikacyjnej (APS), w szczególności komunikacji związanej z transportem klucza m.in. podczas procesu dołączania urządzenia do sieci. Z kolei klucze typu *network key* służą do szyfrowania komunikacji w ramach całej sieci ZigBee.

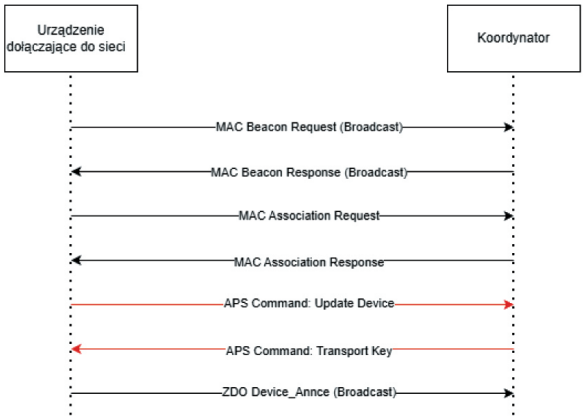
Proces dołączania w trybie *dynamic key negotiation* urządzenia do sieci ZigBee składa się z kilku etapów, rozpoczynających się od wykrycia sieci i uzyskania jej podstawowych parametrów za pomocą zapytania *Beacon Request*. Koordynator odpowiada na zapytanie komunikatem *Beacon Response*, a następnie urządzenie inicjuje ustanawianie połączenia na poziomie warstwy sieciowej dzięki wymianie komunikatów *NWK Commissioning Request*/*Response*. Po zestawieniu połączenia urządzenie i koordynator

przechodzą do fazy uzgadniania klucza – *link key* – na poziomie warstwy aplikacji (APS), w której wykorzystywane są komunikaty *Update Key Request*/*Response*, *Start Key Negotiation Requst*/*Response* oraz proces weryfikacji i potwierdzania klucza. Po pomyślnym zakończeniu negocjacji klucza koordynator przesyła urządzeniu klucz sieciowy (*network key*), który umożliwia szyfrowanie dalszej komunikacji sieciowej. Ostatnim etapem procesu dołączenia jest wymiana komunikatów odpowiedzialnych za finalizację rejestracji urządzenia w sieci.

Dynamic Key Negotiation (rys. 1) to nowszy sposób na wymianę klucza *link key*. W przypadku gdy jedno z urządzeń nie wspiera tego rozwiązania lub urządzenia ZigBee działają w sieci w różnych wersjach specyfikacji, wymiana kluczy następuje według starej statycznej metody (kluczy *pre-configured*). Natomiast gdy klucze preinstalowane nie są ustawione, kluczem domyślnym jest łańcuch *ZigBeeAlliance09*, co wpływa znacząco na zmniejszenie bezpieczeństwa protokołu. Proces dołączania w podanej statycznej metodzie polega na analogicznym uzyskaniu podstawowych parametrów sieci oraz na procesie asocjacji. Po zakończeniu asocjacji koordynator wysyła klucz sieciowy przy użyciu preinstalowanego klucza *link key* oraz wysyła komunikat *Device Announce*, który działa w ten sam sposób jak w przypadku wcześniejszego sposobu *dynamic* (rys. 2).



Rys. 1. Schemat wymiany komunikatów między urządzeniem a koordynatorem w procesie dołączania do sieci ZigBee (na podstawie specyfikacji ZigBee R23) (Connectivity Standards Alliance 2023)



Rys. 2. Schemat wymiany komunikatów między urządzeniem a koordynatorem w procesie dołączania do sieci ZigBee (w przypadku braku obsługi wersji specyfikacji ZigBee R23) (Connectivity Standards Alliance 2023)

2. *State-of-the-art*

Prace łączące kryptografię postkwantową z internetem rzeczy (np. Sajimon i in. 2022) skupiają się z reguły na wymaganiach obliczeniowych nowych algorytmów i ich osiągach w kontekście urządzeń IoT z niską mocą obliczeniową. T.M. Fernández-Caramés przedstawia szerokie spojrzenie na zagadnienie kryptografii postkwantowej w internecie rzeczy, porównując algorytmy z drugiej rundy konkursu PQC NIST i sprawdzając różne konfiguracje na kilku procesorach używanych w urządzenaich IoT (Fernández-Caramés 2020). L. Malina wraz ze współpracownikami przedstawił rezultaty implementacji różnych algorytmów postkwantowych w urządzeniach o niskiej mocy obliczeniowej (Malina i in. 2018). Badania te prowadzone były jednak przed ogłoszeniem dokumentu FIPS 203, który wyłonił CRYSTALS-Kyber i CRYSTALS-Dilithium jako najlepiej sprawdzające się algorytmy szyfrowania i podpisów cyfrowych oraz podał rekomendowane parametry dla ML-KEM. W badaniach skupiono się na aspekcie narzutów czasowych przy implementacji ML-KEM-512 w protokole ZigBee do wymiany klucza oraz w komunikacji sieciowej z szyfrowaniem AES-256 w trybie CTR na podstawie pomiarów wykonanych z pomocą funkcjonującej sieci urządzeń.

3. Metoda

W tym podrozdziale opisano metody badawcze, rozpoczynając od omówienia wykorzystanego sprzętu i oprogramowania, a kończąc na metodologii prowadzenia pomiarów.

3.1. Narzędzia i urządzenia

Do przeprowadzenia doświadczenia wykorzystane zostały narzędzia zigpy-cli (Open Home Foundation b.d.), esp.idf (Espressif Systems b.d.), pyenv, nrfutil, Wireshark i liboqs oraz poniższe urządzenia:

- **Bramka ZigBee 3.0 SONOFF ZBDongle-E**. Urządzenie pełniące rolę koordynatora sieci ZigBee. Umożliwia tworzenie i zarządzanie siecią ZigBee 3.0. Za jego pomocą możliwe było zestawienie sieci oraz komunikacja z urządzeniami końcowymi.
- **SparkFun Thing Plus – ESP32-C6.** Mikrokontroler z obsługą standardu ZigBee 3.0, używany jako urządzenie końcowe (*end device*). W projekcie realizował funkcje nadawcy danych w sieci ZigBee, wykorzystując stos ESP-IDF oraz przykładowe aplikacje ZigBee.
- **nRF52840 Dongle.** Moduł oparty na układzie firmy Nordic Semiconductor, wykorzystywany jako *sniffer* (monitor) ruchu ZigBee. Dzięki odpowiedniemu firmware'owi i integracji z Wiresharkiem umożliwiał przechwytywanie ramek 802.15.4 oraz analizę protokołu ZigBee w czasie rzeczywistym.
- **Komputer z systemem Kali Linux (2025.1).** Służył jako stanowisko monitorujące – zarządzał snifferem nRF52840, konfigurował jego interfejs i przeprowadzał przechwytywanie oraz analizę ruchu w Wiresharku.
- **Komputer z systemem Ubuntu 24.04.2 LTS.** Pełnił rolę terminala kontrolnego w przypadku bramki ZigBee oraz ESP32-C6. Umożliwiał programowanie ESP32-C6 za pomocą ESP-IDF, konfigurację sieci ZigBee przy użyciu biblioteki zigpy-cli, a także zbieranie i przetwarzanie danych pomiarowych.

3.2. Konfiguracja sieci bazowej

Na pierwszym etapie zestawiono standardową sieć ZigBee bez modyfikacji kryptograficznych. Do konfiguracji bramki zastosowano bibliotekę zigpy-cli w wersji 1.1.0, umożliwiającą ręczne zestawienie sieci i odczyt jej parametrów (w tym klucza sieciowego i kanału operacyjnego). Urządzenie końcowe ESP32-C6 zaprogramowano przy użyciu pakietu ESP-IDF oraz oficjalnego repozytorium esp-zigbee-sdk, wykorzystując dostarczone przykładowe implementacje węzłów końcowych ZigBee.

3.3. Monitoring ruchu sieciowego

Do analizy ruchu użyto sniffera nRF52840 oraz oprogramowania Wireshark. *Sniffer* został skonfigurowany z wykorzystaniem narzędzia *nrfutil* i odpowiedniego firmware'u. Następnie zintegrowano go z Wiresharkiem jako urządzenie przechwytujące, umożliwiając dekodowanie ruchu ZigBee przy użyciu klucza sieciowego pozyskanego wcześniej

z konfiguratora. Przechwycony ruch posłużył do weryfikacji poprawności komunikacji oraz stanowił podstawę do pomiarów czasów transmisji i wymiany kluczy.

3.4. Procedura pomiarowa

W każdym cyklu pomiarowym inicjalizowano pełny proces komunikacyjny, obejmujący zestawienie połączenia, wymianę danych i ich deszyfrowanie. W przypadku każdego scenariusza zebrano 1000 niezależnych pomiarów czasu trwania komunikacji (od momentu wysłania pakietu do jego pełnego odbioru i rozszyfrowania). Czas mierzono na poziomie systemu operacyjnego z dokładnością do milisekundy.

3.5. Zmodyfikowana sieć ZigBee z postkwantową wymianą klucza

W odpowiedzi na rosnące wymagania w zakresie bezpieczeństwa komunikacji w sieciach IoT opracowano autorską modyfikację warstwy komunikacyjnej sieci ZigBee, wykorzystującą postkwantowy kryptosystem ML-KEM-512 (Kyber) do wymiany kluczy szyfrujących. Koordynator sieci, oparty na bibliotekach zigpy i zaimplementowany w języku Python, uruchamia sieć ZigBee oraz tworzy lokalną bazę danych zigbee.db, w której przechowywane są informacje o połączonych urządzeniach. Po uruchomieniu sieci następuje okres pozwalający na dołą-czanie nowych urządzeń końcowych. Każde z nich przesyła koordynatorowi swój klucz publiczny ML-KEM-512. Następnie koordynator generuje unikatowy klucz symetryczny (256-bitowy), który szyfruje przy użyciu otrzymanego klucza publicznego i przesyła z powrotem do urządzenia końcowego. Dalsza komunikacja między urządzeniem a koordynatorem odbywa się z dodatkową warstwą szyfrowania za pomocą algorytmu AES w trybie CTR i z użyciem wymienionego klucza symetrycznego. Ze względu na potrzeby badania komunikacja między urządzeniami zawsze była szyfrowana.

3.5.1. Implementacja klastra ZigBee

Protokół ZigBee wspiera tworzenie własnych aplikacji, nazywanych klastrami. Każdy klaster ma wersję przeznaczoną dla klienta i serwera oraz umożliwia komunikację za pomocą zdefiniowanych przez twórcę komend. Zarówno serwer, jak i klient mogą wysyłać komunikaty oraz przechowywać atrybuty – przykładem zastosowania może być domowy system ogrzewania. Piec grzewczy (będący serwerem klastra „ogrzewanie") otrzymuje od klientów (termostatów) żądanie ogrzewania różnych miejsc za pomocą komendy *request_heating*. Za pomocą innego klastra piec może sterować zaworami w celu docieplenia tylko potrzebujących tego pomieszczeń.

Aby zintegrować mechanizm wymiany klucza, utworzono specjalny klaster ZigBee o identyfikatorze 0xfcfc, nazwany *security cluster*. Klaster ten zawiera atrybut identyfi-

kowany przez 0x0000 (key_version), który informuje o wersji klucza symetrycznego, oraz trzy podstawowe komendy:

- **0xFF – send_key** (wykorzystywana przez koordynatora do przesyłania zakodowanego klucza AES; końcowe urządzenie odpowiada tą samą komendą w formie potwierdzenia odbioru (ACK));
- **0xFC – request_key** (używana do zażądania klucza publicznego od urządzenia końcowego w przypadku jego braku po stronie koordynatora);
- **0xFA – send_pub_key** (wysyłana przez klienta w odpowiedzi na żądanie request_key, zawiera klucz publiczny ML-KEM-512).

3.5.2. Ruch w sieci

Proces wymiany zaszyfrowanego klucza symetrycznego, składającego się z 768 bajtów, ze względu na specyfikę działania protokołu ZigBee jest realizowany z wykorzystaniem fragmentaryzacji na 12 pakietów po 64 bajty, które następnie przesyłane są do urządzenia końcowego. Po otrzymaniu wszystkich fragmentów szyfrogramu urządzenie końcowe wysyła potwierdzenie otrzymania całej wiadomości.

Procedura przesyłania klucza publicznego do koordynatora jest rozpoczynana przez koordynator wysłaniem do urządzenia końcowego komendy 0xFC. W odpowiedzi urządzenie końcowe przesyła swój klucz publiczny. Składa się on z 800 bajtów, co wymusza fragmentaryzację – jest przesyłany za pomocą 13 pakietów, pierwsze 12 po 64 bajty oraz ostatni 32-bajtowy. W celu symulowania komunikacji danych urządzenie końcowe (ESP32-C6) generuje pakiety zawierające losowe dane, które następnie przesyła do koordynatora.

3.6. Metodologia testów

W ramach testów wydajnościowych przeanalizowano kluczowe komponenty zmodyfikowanego protokołu komunikacyjnego. Przedmiotem pomiarów były czas generowania i transmisji 64-bajtowego szyfrogramu zawierającego klucz symetryczny oraz opóźnienie związane z procesem jego fragmentacji i przesyłania przez sieć ZigBee. Zostało przeprowadzonych 1000 niezależnych pomiarów czasu transmisji szyfrogramu.

Został również zmierzony narzut czasowy spowodowany przez implementowaną warstwę szyfrowania AES-256 (tryb CTR) w ruchu sieciowym, a także przeprowadzono po 1000 niezależnych pomiarów czasów przesłania 64-bajtowego pakietu ZigBee w sieci bez modyfikacji z użyciem szyfrowania AES-256, co pozwoliło oszacować wpływ warstwy szyfrowania AES na opóźnienie transmisji.

Dodatkowo zmierzono czasy wykonania funkcji odpowiedzialnych za szyfrowanie pakietów za pomocą AES. Testy wykonano na 64-bajtowych ciągach losowych danych.

4. Dyskusja

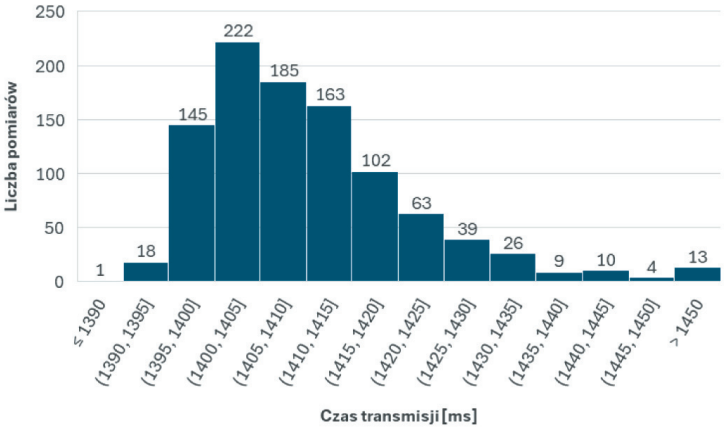
Poniżej omówiono wyniki pomiarów czasu wymiany klucza za pomocą ML-KEM-512 oraz porównano zmierzone czasy przesyłu danych w standardowej sieci ZigBee z wynikami otrzymanymi w sieci implementującej opisane w artykule rozwiązanie szyfrowania ruchu za pomocą AES-256.

4.1. Wydajność wymiany klucza symetrycznego

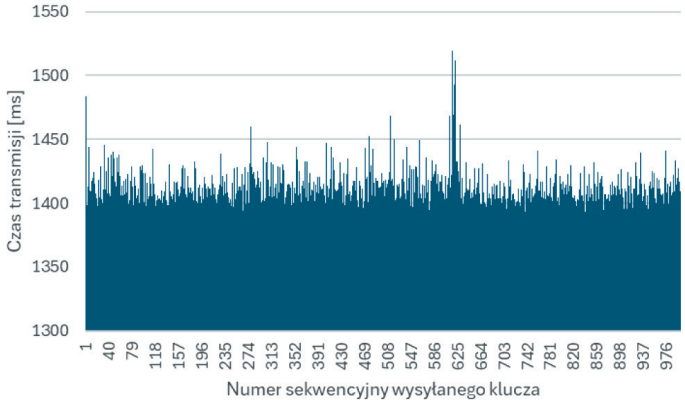
Po przeprowadzeniu wyżej opisanych testów prędkości przesyłania klucza symetrycznego otrzymano wyniki przedstawione na rysunkach 3–5.

Jak pokazano na rysunku 3, najwięcej pomiarów (222) mieści się w przedziale [1400, 1405) ms. Większość pomiarów skupia się w zakresie od 1395 ms do 1420 ms. Rozkład danych jest prawostronnie skośny, występuje wyraźny „ogon" po prawej stronie – są też pomiary z czasami dochodzącymi nawet do ponad 1450 ms, ale nieliczne. Sugeruje to obecność sporadycznych opóźnień, które mogą wynikać z obciążenia sieci, przeciążeń sprzętowych lub zakłóceń transmisji. Wartości skrajnie niskie (<1390 ms) i skrajnie wysokie (>1450 ms) są rzadkie (łącznie 14 pomiarów). Wskazuje to na relatywną stabilność działania protokołu (zdarzają się przypadki wyjątkowe). Rysunek 4 przedstawia graficznie czasy wymiany kolejnych kluczy. Obserwowany w okolicy 625 pomiaru wzrost czasu wymiany jest artefaktem działania sieci w rzeczywistym środowisku.

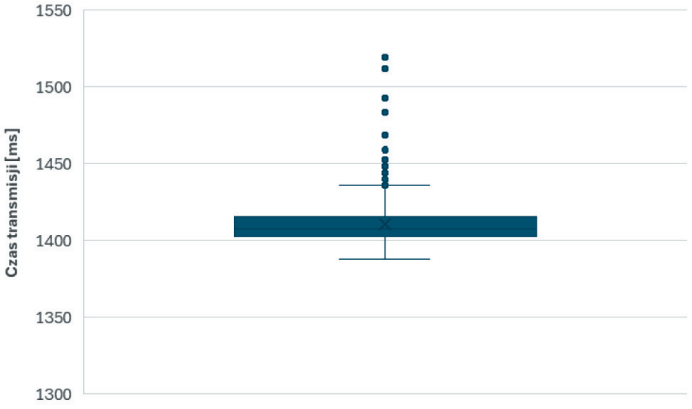
Na podstawie 1000 pomiarów czasu transmisji klucza symetrycznego można stwierdzić, że średnia wartość wyniosła 1410,51 ms, natomiast mediana była nieco niższa i wyniosła 1407,65 ms. Odchylenie standardowe próbki na poziomie 12,97 ms wskazuje na niewielką zmienność wyników, a niski błąd standardowy średniej wynosi 0,41% (rys. 5).



Rys. 3. Rozkład czasów transmisji klucza symetrycznego



Rys. 4. Graficzna reprezentacja czasów przesłania kolejnych kluczy

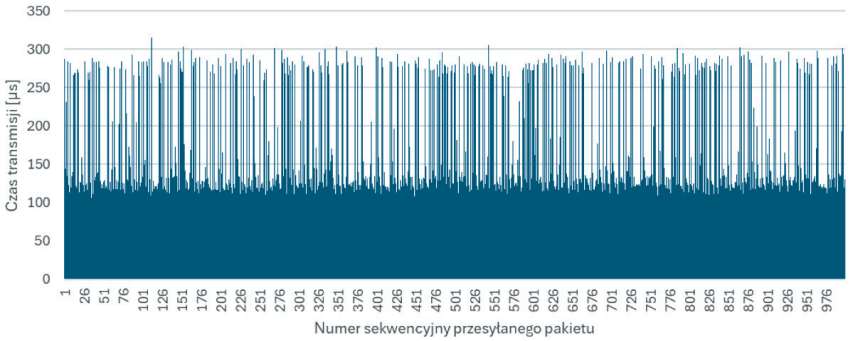


Rys. 5. Wykres pudełkowy z wąsami czasów transmisji klucza symetrycznego

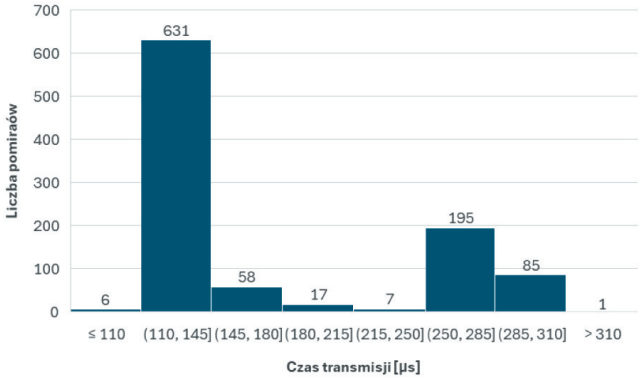
4.2. Wyniki pomiarów w sieci podstawowej

Pomiary czasu transmisji pakietów w bazowej sieci oraz w sieci korzystającej z proponowanego rozwiązania zostały przeprowadzone w seriach 1000 przesyłanych pakietów o jednakowej długości 64 bajtów i losowej zawartości. Między transmisją kolejnych pakietów został wprowadzony odstęp 100 ms, który nie jest uwzględniany w pomiarze. Rysunek 6 ilustruje graficzną reprezentację czasu transmisji w sieci bazowej na podstawie 1000 pomiarów. Widoczne są dwie przeważające wartości – około 125 µs i 275 µs. Tę obserwację potwierdza histogram przedstawiony na rysunku 7.

Można przypuszczać, że obserwowany podział jest skutkiem konieczności retransmisji niektórych pakietów, co powoduje około dwukrotny wzrost czasu przesyłu. Wydłużenie czasu transmisji wydaje się związane z koniecznością odczekania na potwierdzenie odbioru pakietu ze strony odbiorcy oraz podjęciem decyzji o retransmisji.

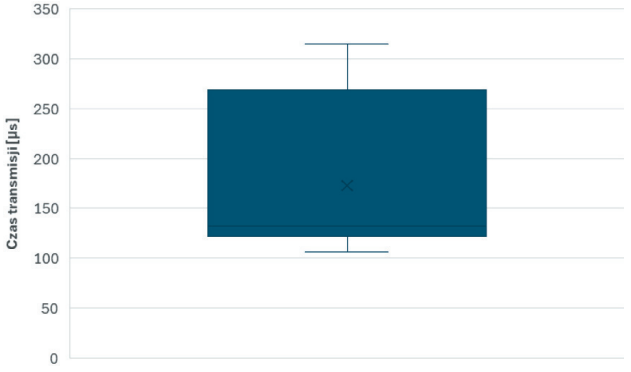


Rys. 6. Graficzna reprezentacja czasów przesłania pakietów z serii w sieci, która nie korzysta z szyfrowania



Rys. 7. Histogram czasów przesyłu pakietu bez dodatkowej warstwy szyfrowania

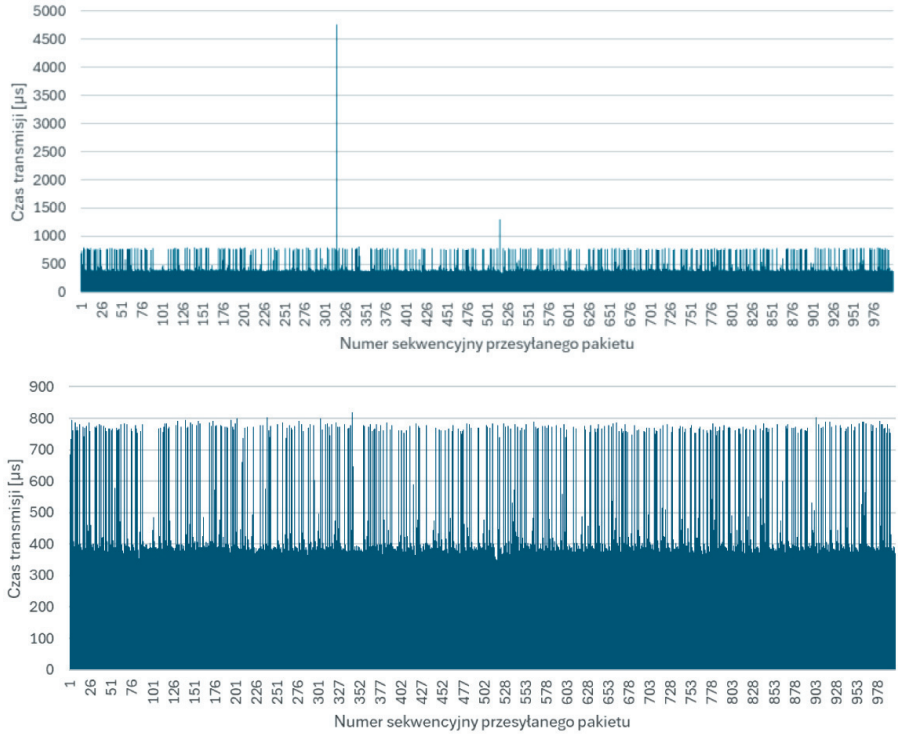
Maksymalnym obserwowanym czasem przesyłu było 315 µs, natomiast najkrótszy czas wynosił 106 µs. Średni czas wynosił 172,69 µs, a mediana czasu przesyłania ma wartość 132 µs – te dane ilustruje rysunek 8.



Rys. 8. Wykres pudełkowy z wąsami czasów przesyłu pakietu bez użycia szyfrowania AES-256

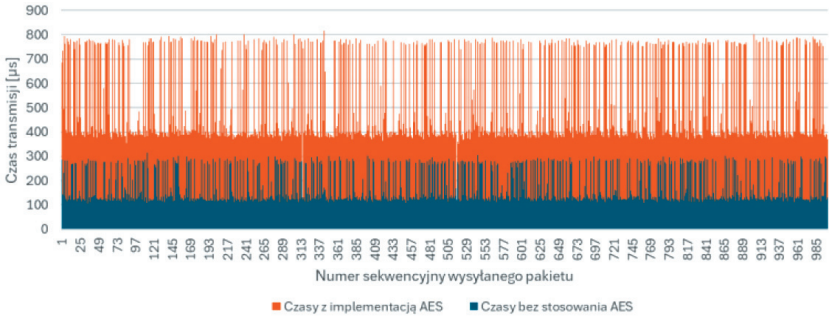
4.3. Wyniki pomiarów w przypadku sieci implementującej szyfrowanie AES-256

Wyniki pomiarów czasu przesyłu kolejnych pakietów w sieci implementującej szyfrowanie za pomocą AES ilustruje rysunek 9. W pomiarach pojawiły się dwa wyniki znacznie odbiegające od reszty – uznane zostały one za wyjątki. Tak samo jak na rysunku 6, również na rysunku 9 można zaobserwować dwie wyraźne serie wartości czasów.

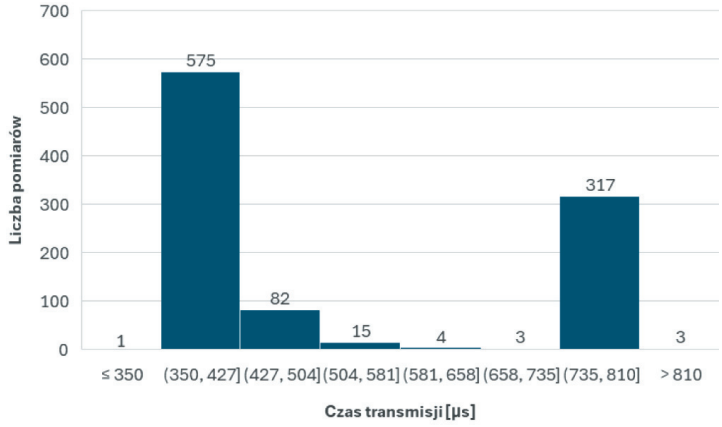


Rys. 9. Graficzna reprezentacja czasów przesłania pakietów z serii w sieci, która implementuje szyfrowanie za pomocą AES. Górny wykres przedstawia dane bez modyfikacji, natomiast na dolnym wykresie ilustrowane są czasy bez uwzględnienia dwóch odstających pomiarów

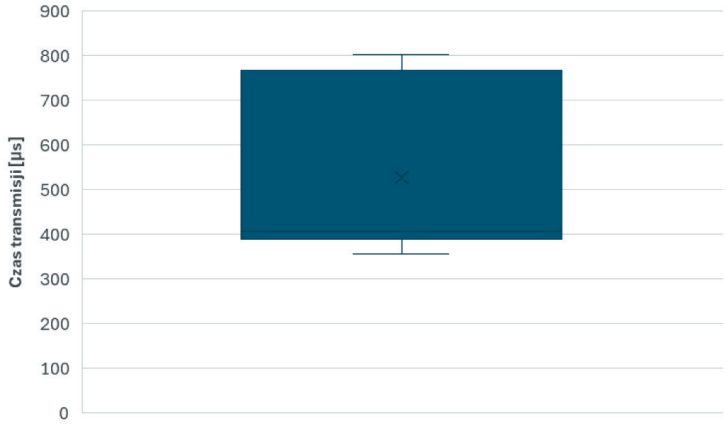
Rysunek 10 ilustruje porównanie serii pomiarów w przypadku bazowej sieci oraz po implementacji szyfrowania AES. Histogram przedstawiony na rysunku 11 również odzwierciedla występujące w poprzednich pomiarach dwie wyraźne serie. Najdłuższy wynik pomiaru liczył 818 µs, najkrótszy natomiast wyniósł 349 µs – jest to wartość wyższa od maksymalnego odnotowanego czasu w sieci bazowej. Średni czas przesyłu z szyfrowaniem wynosił w przybliżeniu 522,36 µs, a wartość mediany dla tej serii to 406 µs. Te dane przedstawia rysunek 12.



Rys. 10. Wykres porównujący czasy transmisji bez szyfrowania (niebieski) oraz po implementacji AES (pomarańczowy). Odrzucone zostały pomiary drastycznie różniące się od pozostałych



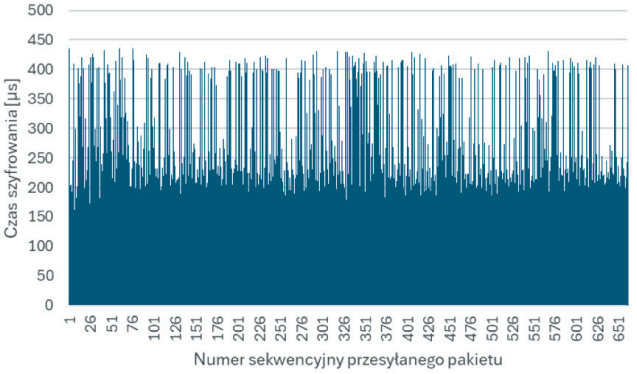
Rys. 11. Histogram czasów przesyłu pakietu z użyciem dodatkowej warstwy szyfrowania



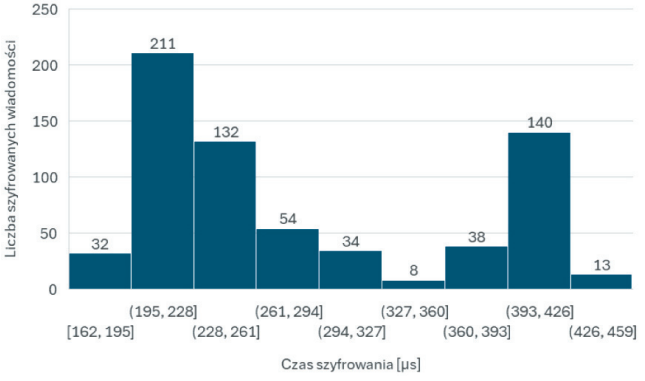
Rys. 12. Wykres pudełkowy z wąsami czasów przesyłu pakietu z użyciem dodatkowej warstwy szyfrowania, nieuwzględniający pomiarów drastycznie różniących się od pozostałych

4.4. Wyniki pomiarów czasów szyfrowania AES

W poszukiwaniu przyczyn zmienności czasów transmisji zostało przeprowadzonych 660 pomiarów czasów wykonania funkcji szyfrującej. Wyniki pomiarów są ilustrowane na rysunku 13. Ponownie obserwowany jest podział na dwie serie czasów. Wartości wyższej serii stanowią mniej więcej dwukrotność tych z serii niższej. Ten trend odzwierciedlony jest na histogramie przedstawionym na rysunku 14. Podobieństwo tych wyników do pomiarów transmisji prowadzi do wniosku, że zmienności rezultatów pomiarów są powodowane funkcjonowaniem sprzętu.



Rys. 13. Wykres czasów szyfrowania kolejnych pakietów



Rys. 14. Histogram czasów szyfrowania pakietów

4.5. Interpretacja wyników

Wyniki pomiarów czasów przesłania szyfrogramu klucza AES sugerują znaczący wpływ wdrożenia wymiany klucza na czas inicjalizacji infrastruktury. Pokazują one narzut czasowy rzędu około 1,4 s w przypadku każdego urządzenia będącego częścią sieci, który znacząco ogranicza skalowalność proponowanego w artykule rozwiązania.

Analiza wyników przesyłu pakietów zarówno w sieci bez szyfrowania, jak i w sieci implementującej szyfrowanie wykazuje obecność dwóch dominujących przedziałów czasowych, odpowiednio: od 110 µs do 145 µs i od 250 µs do 285 µs w sieci bez szyfrowania oraz od 350 µs do 427 µs i od 735 µs do 810 µs w sieci z dodatkową warstwą szyfrującą komunikację. Ten trend jest również widoczny w pomiarach czasów samego szyfrowania AES – odpowiednio od 195 µs do 261 µs oraz od 393 µs do 426 µs. Na podstawie widocznej zależności między pomiarami można wnioskować, że zmienność czasów transmisji jest zależna od funkcjonowania sprzętu użytego w badaniach – konkretnie ESP32-C6. Prawdopodobnie ten fakt związany jest z ograniczonymi zasobami pamięciowymi oraz obliczeniowymi urządzenia lub mechanizmami zarządzania tymi zasobami. Analiza funkcjonowania rozwiązania w konfiguracji różniącej się od prezentowanej w artykule może pomóc wyjaśnić tę obserwację. Porównując czasy z głównych grup widocznych na histogramach z rysunków 7 i 11, można zobaczyć, że przedział czasu obejmujący większość niższych wyników w sieci bez szyfrowania – (110,145] µs – drastycznie wzrasta przy implementacji AES – do (350,427] µs. Ten przedział czasu jest znacznie wyższy od maksymalnego wyniku pomiarów wykonanych w przypadku bazowej sieci, który wynosił 315 µs. Druga grupa, o dłuższych czasach transmisji, obserwowana na histogramach, przeniosła się z przedziału (250,310] do przedziału (735,810] w przypadku sieci implementującej przedstawiane rozwiązanie. Natomiast wzrost mierzonego czasu w przypadku grupy z niższymi wartościami sięga około 261 µs, czyli około 205%. W wyższej grupie ta zmiana wynosiła 492,5 µs, czyli w przybliżeniu 176%. Mediany czasów wynosiły 132 µs w serii niestosującej szyfrowania oraz 406 µs w serii pomiarów z wykorzystaniem opisanego rozwiązania. Oznacza to różnicę wynoszącą 274 µs, co przekłada się na 207% wzrostu czasu przesłania pakietu.

5. Wnioski

Bezpieczeństwo szyfrowania uzależnione jest od skutecznej wymiany klucza symetrycznego. W zaproponowanym rozwiązaniu proces ten realizowany jest z wykorzystaniem algorytmu ML-KEM-512 odpornego na ataki i komputerów kwantowych.

Poziom bezpieczeństwa standardowego ZigBee
a poziom bezpieczeństwa proponowanego rozwiązania

Rozwiązania kryptograficzne w standardowym protokole ZigBee wykorzystują protokół ECDHE, oparty na krzywej Curve25519 i zapewniający 128 bitów bezpieczeństwa. W przypadku ataków za pomocą komputera kwantowego bezpieczeństwo bitowe kombinacji ECDH + AES256 praktycznie spada do zera, ze względu na całkowite złamanie protokołu ECDH algorytmem Shora. Proponowane rozwiązanie imple-

mentuje algorytm ML-KEM-512 o 128 bitach bezpieczeństwa (NIST 2024) oraz o udowodnionej odporności na znane techniki kryptoanalizy z użyciem komputera kwantowego. Jest on stosowany podczas wymiany klucza AES-256 gwarantującego 256 bitów bezpieczeństwa bez udziału komputera kwantowego. Algorytm Grovera pozwala jednak na redukcję bezpieczeństwa do 128 bitów (Grover 1996). Oznacza to, że jako całość proponowane rozwiązanie zapewnia bezpieczeństwo na poziomie 128 bitów. Uzyskany poziom jest akceptowany przez aktualne standardy.

Wprowadzenie takiego mechanizmu wiąże się jednak z dodatkowymi wymaganiami dotyczącymi infrastruktury systemu. W szczególności istotnym czynnikiem ograniczającym jest rozmiar generowanego szyfrogramu, który w przypadku ML-KEM-512 wynosi 768 bajtów. Może to stanowić istotne obciążenie dla sieci o ograniczonej przepustowości, zwłaszcza w środowiskach o dużej liczbie urządzeń końcowych. Czas inicjalizacji rośnie w tym przypadku liniowo wraz z liczbą urządzeń, ponieważ koordynator musi przeprowadzić indywidualną wymianę klucza z każdym z nich.

Zastosowanie wzmocnionego szyfrowania z wykorzystaniem algorytmu AES-256, pomimo dostępności sprzętowych akceleratorów wspierających jego działanie, wiąże się ze wzrostem zapotrzebowania na zasoby systemowe. Może to negatywnie wpłynąć na działanie systemu w przypadku urządzeń o ograniczonej mocy obliczeniowej, powodując spadek efektywności komunikacji.

Z przeprowadzonych analiz wynika, że zaproponowane rozwiązanie nie jest optymalne w zastosowaniach wymagających obsługi dużej liczby urządzeń końcowych generujących intensywny ruch sieciowy. Może ono natomiast być przydatne w systemach o ograniczonej skali lub w sieciach złożonych z urządzeń dysponujących odpowiednią mocą obliczeniową. Rozwiązanie to może być szczególnie skutecznie stosowane w komunikacji pomiędzy wybranymi węzłami, w których przypadku zapewnienie wysokiego poziomu bezpieczeństwa transmisji danych ma kluczowe znaczenie. Ze względu na narzuty związane z inicjalizacją sieci koordynatorem powinno być urządzenie o wysokiej dostępności. Pozwoli to ograniczyć występowanie zdarzeń powodujących, że cała sieć musi zostać zbudowana od nowa.

Literatura

Avanzi R., Bos J., Ducas L., Kiltz E., Lepoint T., Lyubashevsky V., Schanck J.M. et al., 2021, *CRYSTALS-Kyber. Algorithm specifications and supporting documentation (version 3.02)*. https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf [dostęp: 13 maja 2025].

Connectivity Standards Alliance, 2023, *ZigBee Specification – Revision 23*. https://csa-iot.org/wp-content/uploads/2023/04/05-3474-23-csg-zigbee-specification-compressed.pdf [dostęp: 13 maja 2025].

El Gamal, T., 1985, *A public key cryptosystem and a signature scheme based on discrete logarithms*, [w:] Blakley G.R., Chaum D. (eds.), *Advances in cryptology. Proceedings of CRYPTO 84*, Lecture Notes in Computer Science, vol. 196, s. 10–18, Springer, Berlin–Heidelberg–New York–Tokyo.

Espressif Systems, *esp-idf*, b.d. https://github.com/espressif/esp-idf [dostęp: 11 maja 2025].

Fernández-Caramés T.M., 2020, *From pre-quantum to post-quantum IoT security: A survey on quantum-resistant cryptosystems for the Internet of Things,* IEEE Internet of Things Journal, vol. 7, no. 7, s. 6457–6480. https://doi.org/10.1109/JIOT.2019.2958788.

Grover L.K., 1996, *A fast quantum mechanical algorithm for database search*, [w:] *STOC '96. Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, Association for Computing Machinery, New York, USA, s. 212–219. https://doi.org/10.1145/237814.237866.

Malina L., Popelova L., Dzurenda P., Hajny J., Martinasek Z., 2018, *On feasibility of post-quantum cryptography on small devices,* IFAC-PapersOnLine, vol. 51, s. 462–467. https://doi.org/10.1016/j.ifacol.2018.07.104.

NIST (National Institute of Standards and Technology), 2024, *Module-Lattice-Based Key-Encapsulation Mechanism Standard*, FIPS 203. Federal Information Processing Standards Publication. https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.203.pdf [dostęp: 13 maja 2025].

Open Home Foundation, b.d., *zigpy.* https://github.com/zigpy/zigpy [dostęp: 11.05.2025].

Paar C., Pelzl J., 2010, *The RSA cryptosystem*, [w:] Paar C., Pelzl J., *Understanding cryptography: A Textbook for students and practitioners*, Springer, Heidelberg–Dordrecht–London–New York.

Rivest R.L., Shamir A., Adleman L., 1978, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM, vol. 21, no. 2, s. 120–126. https://doi.org/10.1145/359340.359342.

Sajimon P.C., Jain K., Krishnan P., 2022, *Analysis of post-quantum cryptography for Internet of Things*, [w:] *2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS)*, Madurai, Indie, s. 387–394. https://doi.org/10.1109/ICICCS53718.2022.9787987.

Shor P.W., 1997, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM Journal on Computing, vol. 26, no. 5, s. 1484–1509. https://doi.org/10.48550/arXiv.quant-ph/9508027.

ZigBee Alliance, 2015, *ZigBee Specification*. https://zigbeealliance.org/wp-content/uploads/2019/11/docs-05-3474-21-0csg-zigbee-specification.pdf [dostęp: 3.12.20125].