WOJSKOWA AKADEMIA TECHNICZNA

im. Jarosława Dąbrowskiego

WYDZIAŁ CYBERNETYKI

Ataki algebraiczne na szyfry blokowe z wykorzystaniem wyżarzania kwantowego

ROZPRAWA DOKTORSKA

Autor:

kpt. mgr inż. Elżbieta BUREK

Promotor pracy:

dr hab. Marek KOJDECKI

Promotor pomocniczy:

dr inż. Michał WROŃSKI

Warszawa 2023

Spis treści

WPROWADZENIE

Zainteresowanie ideą komputerów kwantowych wzbudził Feynman w 1981 roku w wykładzie na MIT, gdzie przedstawił następujący dylemat: klasyczne komputery nie są w stanie skutecznie symulować ewolucji układów kwantowych. Dlatego też w swojej pracy [30] wysunął hipotezę, że komputer kwantowy byłby w stanie wykonać obliczenia szybciej, niż klasyczna maszyna deterministyczna. Komputery kwantowe są odpowiedzią na proste pytania. Jeśli dynamika kwantowa jest trudna do symulacji, to co by się stało, gdyby zbudowano sprzęt, którego podstawowe operacje miałyby efekty kwantowe? Czy możliwe byłoby symulowanie układów oddziałujących cząstek za pomocą systemu, który wykorzystuje dokładnie te same prawa, które rządzą nimi w sposób naturalny?

Kilka lat później, w 1985 roku, Deutsch w pracy [23] opracował uniwersalny model obliczeń oraz przedstawił opis uniwersalnego komputera kwantowego, który może symulować dowolne obliczenia maszyny Turinga przy koszcie nie większym niż wielomianowy. Podstawą obliczeń kwantowych jest przechowywanie informacji w kwantowych stanach materii oraz realizowanie obliczeń za pomocą bramek kwantowych. Komputer kwantowy nie jest superkomputerem, który może zrobić wszystko szybciej lub rozwiązać każdy problem. Klasa problemów, które komputer kwantowy może rozwiązać efektywniej, niż komputer klasyczny, nazywana jest BQP (*ang. Bounded-error Quantum Polynomial*). Problemy tej klasy można rozwiązać za pomocą komputera kwantowego w czasie wielomianowym, z prawdopodobieństwem błędu co najwyżej $\frac{1}{3}$.

Algorytm, opracowany w celu rozwiązania problemu, który wydawał się trudny dla komputerów klasycznych, został zaproponowany przez Petera Shora w 1994 r. dla problemu faktoryzacji. Algorytm Shora może rozłożyć liczbę n-bitową w czasie wielomianowym na komputerze kwantowym w modelu bramkowym. Rozwiązanie problemu faktoryzacji niesie ze sobą możliwość złamania wielu systemów kryptograficznych z kluczem publicznym, w tym algorytmu RSA i kryptografii krzywych elip-

tycznych, które stanowią podstawę bezpieczeństwa handlu elektronicznego. Kolejnym algorytmem, ważnym dla dziedziny kryptografii, jest algorytm Grovera. W 1996 r. Lov Grover zaprezentował w pracy [36] swój algorytm w kwantowym modelu obliczeń, który może przeszukiwać nieuporządkowaną bazę danych o rozmiarze n w czasie $O\left(\sqrt{n}\right)$. Dla porównania, każdy klasyczny algorytm w celu przeszkujania nieuporządkowanej bazy potrzebuje wykonać średnio $\frac{n}{2}$ kroków, a problem jest złożoności $O(n)$. Stanowi to przyspieszenie kwantowe rzędu pierwiastka z n.

Od tego czasu opracowano szybkie i wydajne algorytmy dla komputerów kwantowych, przeznaczone do rozwiązywania problemów matematycznych, które są trudne lub niewykonalne dla klasycznych komputerów. Jeśli kiedykolwiek zostaną zbudowane komputery kwantowe na dużą skalę, będą w stanie złamać wiele obecnie używanych systemów kryptograficznych z kluczem publicznym. To poważnie zagroziłoby poufności i integralności komunikacji cyfrowej w Internecie i poza nim. Dlatego celem kryptografii postkwantowej jest opracowanie systemów kryptograficznych, które są bezpieczne zarówno, gdy używa się komputerów kwantowych, jak i klasycznych oraz mogą współpracować z istniejącymi protokołami i sieciami komunikacyjnymi. W przeszłości wdrożenie nowoczesnej infrastruktury klucza publicznego zajęło prawie dwie dekady. Dlatego niezależnie od tego, czy jesteśmy w stanie oszacować dokładny czas nadejścia ery komputerów kwantowych, już teraz należy zacząć przygotowywać systemy bezpieczeństwa informacji, aby były odporne na ataki przy użyciu komputerów kwantowych.

W odniesieniu do zagrożeń, jakie komputery kwantowe stwarzają dla algorytmów klucza publicznego, Daniel Bernstein w 2009 w [4] przekonuje, że większość obecnych algorytmów kryptografii symetrycznej i funkcji skrótu jest uważana za względnie odporne na ataki przy użyciu komputerów kwantowych. Nawet algorytm Grovera, mający zastosowania w tych systemach, nie jest tak szybki jak algorytm Shora, a kryptolodzy mogą łatwo zrekompensować zagrożenie, wybierając nieco większe rozmiary kluczy. Powszechnie uważa się, że wszystkie te systemy będą musiały podwoić rozmiar klucza, aby przetrwać ataki za pomocą komputerów kwantowych. Z reguły dla 128-bitowego poziomu bezpieczeństwa systemu opartego na kluczu symetrycznym,

można bezpiecznie używać kluczy o długości 256 bitów.

Motywacją do badań przedstawionych w niniejszej rozprawie było pytanie: czy istnieje lepszy atak na szyfry blokowe wykorzystujący komputery kwantowe, który jest efektywniejszy niż algorytm Grovera oraz który należy wziąć pod uwagę w analizie bezpieczeństwa szyfrów? Jest to ryzyko znane w kryptografii. Właśnie dlatego społeczność inwestuje ogromne ilości czasu i energii w kryptoanalizę. Czasami kryptoanalitycy znajdują atak, który pokazuje, że system jest bezużyteczny, a czasami znajdują ataki, które nie są tak niszczycielskie, ale wymuszają zastosowanie większych rozmiarów kluczy.

Postęp w budowie komputerów kwantowych jest ogromny i w czasie rozpoczęcia przedstawionych w tej rozprawie badań, największy komputer kwantowy w modelu bramkowym firmy Google posiadał 72 działające kubity [35], natomiast obecnie procesor IBM Osprey ma 433 kubity. Jednak to wciąż za mało, aby złamać algorytmy wykorzystywane obecnie w bezpieczeństwie informacji. Dlatego też w ciągu ostatnich kilku lat komputer kwantowy opracowany na modelu adiabatycznych obliczeń kwantowych zyskał dużą popularność. Proces szukania rozwiązania na komputerze tego typu nazywa się wyżarzaniem kwantowym i jest stosowany w komputerach zbudowanych przez firmę D-Wave. Największym z nich jest D-Wave Advantage, który ma 5 760 fizycznych kubitów i może rozwiązywać gęste problemy w formie minimalizacji bez ograniczeń binarnych funkcjonałów kwadratowych (QUBO), składające się maksymalnie z 20 000 zmiennych binarnych oraz problemy rzadkie z maksymalnie 1 000 000 zmiennych binarnych. Wykorzystanie wyżarzania kwantowego pozwoliło na sukces w rozwiązywaniu problemu faktoryzacji. Przez jakiś czas najlepszy wynik faktoryzacji kwantowej został uzyskany na komputerze D-Wave. Wykorzystując transformację rozkładu na czynniki całkowite do problemu QUBO, Dridi i Alghassi [25] byli w stanie rozłożyć na czynniki liczbę 200 099, co później zostało pobite przez Jiang i in. [41] oraz Wang i in. [61], który rozłożył na czynniki 20-bitową liczbę 1 028 171. Takie przekształcenie problemu faktoryzacji do problemu QUBO wymaga około $\frac{n^2}{4}$ kubitów logicznych. Oznacza to, że rozkład na czynniki liczby całkowitej algorytmu RSA o długości 3 072 bitów przy użyciu wyżarzania kwantowego wymaga rozwią-

zania problemu QUBO składającego się z około 2 360 000 zmiennych. Uważa się, że taki problem RSA ma podobny poziom bezpieczeństwa (128 bitów) jak AES128. Co więcej, możliwe jest również przekształcenie problemu logarytmu dyskretnego do problemu QUBO [62], gdzie potrzebnych jest około $2n^2$ kubitów logicznych. Oznacza to, że obliczenie logarytmu dyskretnego w grupie multiplikatywnej z modułem o długości 3 072 bitów, przy użyciu wyżarzania kwantowego, wymaga rozwiązania problemu QUBO składającego się z około 18 875 000 zmiennych. Uważa się, że taki problem logarytmu dyskretnego ma również podobny poziom bezpieczeństwa jak AES128. Przykłady te sprawiają, że wyżarzanie kwantowe może znaleźć zastosowanie również w kryptoanalizie algorytmów kryptografii symetrycznej. Dlatego jako narzędzie ataku na szyfry blokowe, proponowanego w niniejszej rozprawie, wybrano właśnie komputer kwantowy D-Wave.

Pomimo że systemy kryptografii symetrycznej uważa się za względnie bezpieczne w kontekście ataków z wykorzystaniem komputerów kwantowych, zaproponowano kilka algorytmów dla komputerów modelu bramkowego. Proponowane ataki kwantowe na szyfry symetryczne są implementacją ataków klasycznych, takich jak kryptoanaliza różnicowa i liniowa, nierzadko połączonych z wykorzystaniem algorytmu Grovera. Przykłady takich ataków można znaleźć w pracach [47], [66], [44], [64], [63], [56] oraz [65]. Podobnie zaproponowano implementację ataków algebraicznych na szyfry symetryczne dla komputerów ogólnego przeznaczenia, które przedstawiono w pracach [16], [33] oraz [38]. Chociaż zaproponowane ataki mają duże znaczenie w kryptoanalizie kwantowej szyfrów symetrycznych, to nie można tego interpretować jako upadku kryptografii, w taki sam sposób jak algorytmu Shora dla kryptografii z kluczem publicznym.

W przypadku wykorzystania komputera kwantowego z wyżarzaniem kwantowym do kryptoanalizy szyfrów symetrycznych, w 2022 roku pojawiła się praca [54], w której zaproponowano metodę sprowadzenia binarnych równań liniowych do kwadratowego problemu optymalizacyjnego (QUBO), rozwiązywanego za pomocą wyżarzacza kwantowego, np. D-Wave. Przedstawiona tam metoda wykorzystuje koniunktywną postać normalną funkcji boolowskich i różni się znacząco od metody transfor-

macji do problemu QUBO, zaprezentowanej w niniejszej rozprawie. Uzyskane przez autorów pracy [54] rozmiary problemów optymalizacyjnych dla standardu AES są następujące: dla wersji AES128 uzyskali 90 000 zmiennych binarnych, dla wersji AES192 uzyskali $2 \cdot 100\,000$ zmiennych, a dla wersji AES256 uzyskali $2 \cdot 125\,000$ zmiennych. Problemy te są znacznie większe, niż problemy QUBO wyznaczone za pomocą prezentowanej w tej rozprawie metody.

Celem postawionym w niniejszej rozprawie jest dostosowanie modelu transformacji układu równań do postaci problemu optymalizacyjnego, rozwiązywanego za pomocą wyżarzania kwantowego, do wykorzystania w atakach algebraicznych na szyfry blokowe, osiągając rozmiary problemów mniejsze, niż dotychczas opublikowano. Aby osiągnąć postawiony cel, należy zrealizować następujące zadania:

1. Przeanalizowanie implementacji procesu wyżarzania kwantowego w komputerze firmy D-Wave.

2. Dostosowanie modelu transformacji układu równań wielomianowych do problemu optymalizacyjnego w postaci QUBO, w celu wykorzystania go w atakach algebraicznych na szyfry blokowe przy użyciu komputera firmy D-Wave oraz scharakteryzowanie efektywnego układu równań wielomianowych, umożliwiającego uzyskanie możliwie najmniejszego problemu QUBO.

3. Przeanalizowanie struktury wybranych szyfrów blokowych, w celu znalezienia układu efektywnego.

4. Dla małych instancji analizowanych szyfrów, przeprowadzenie proponowanego ataku na dostępnych zasobach komputera D-Wave.

Rozprawa została podzielona na siedem rozdziałów, z których dwa pierwsze rozdziały są wprowadzeniem teoretycznym, a pozostałe zawierają opis prac badawczych przeprowadzonych przez autora.
Struktura niniejszej rozprawy jest następująca:

- **Rozdział 1** stanowi ogólne wprowadzenie do zagadnienia ataków algebraicznych, gdzie omówiono ich ideę oraz przedstawiono główne narzędzia wykorzy-

stywane do ich realizacji.

- **Rozdział 2** poświęcony jest obliczeniom kwantowym oraz komputerowi kwantowemu D-Wave. Pierwsza część tego rozdziału jest ogólnym wprowadzeniem do obliczeń kwantowych, ze szczególnym naciskiem na model adiabatycznych obliczeń kwantowych oraz wyżarzanie kwantowe. Druga część stanowi charakterystykę komputera kwantowego D-Wave, gdzie przedstawiono sposób implementacji oraz realizacji procesu wyżarzania kwantowego na tym komputerze, jak również szczegółowo omówiono proces rozwiązywania problemów optymalizacyjnych, za pomocą komputera D-Wave.

- **Rozdział 3** jest podstawą niniejszej rozprawy doktorskiej. W szczegółowy sposób omówiono w nim proponowany model transformacji układu równań wielomianowych, opisujących dowolny szyfr blokowy, do problemu optymalizacyjnego w postaci QUBO. Omówiono poszczególne kroki transformacji wraz z zastosowanymi metodami ich realizacji. Podsumowaniem rozdziału jest opis wymagań, jakie powinien spełniać układ, opisujący szyfr blokowy, w celu otrzymania problemu w postaci QUBO o jak najmniejszym rozmiarze.

- **Rozdział 4** przedstawia analizę struktury standardu AES – szyfru blokowego typu SPN, w kontekście przedstawienia go za pomocą układu równań, który przetransformowany do problemu QUBO da jak najmniejszy rozmiar problemu optymalizacyjnego. Duży nacisk postawiono na znalezienie układu efektywnego dla skrzynki podstawieniowej.

- **Rozdział 5** prezentuje analizę przedstawienia szyfru blokowego typu ARX, którym jest szyfr Speck, za pomocą takiego układu równań, dla którego otrzyma się jak najmniejszy problem QUBO. Podczas analizy zwrócono uwagę na zakres rundy oraz na strukturę algebraiczną, nad którą przedstawiono równania, opisujące działanie szyfru Speck.

- **Rozdział 6** opisuje poszukiwania układu równań, opisującego szyfr blokowy o strukturze Feistela, umożliwiającego otrzymanie jak najmniejszego problemu

QUBO. Jako przykładowy szyfr wybrano szyfr Simon.

- **Rozdział 7** stanowi podsumowanie niniejszej rozprawy.

1 ATAKI ALGEBRAICZNE

Claude Shannon, w 1949 roku powiedział, że złamanie dobrego szyfru powinno wymagać „tyle pracy, co rozwiązanie układu równań z dużą liczbą niewiadomych" [58]. Faktycznie przez około pięćdziesięciu lat kryptografia symetryczna koncentrowała się głównie na statystycznych aspektach bezpieczeństwa i omijała naturalne, algebraiczne podejście do problemu. Stało się tak prawdopodobnie dlatego, że rozwiązanie dowolnego układu równań wielomianowych o wielu zmiennych jest problemem NP-trudnym. Założono więc, że rozwiązanie takiego układu nie będzie szybsze niż pełne przeszukiwanie przestrzeni kluczy. Jednak nie wszystkie przypadki problemów NP-trudnych same muszą być NP-trudne. Możliwe jest przedstawienie szyfru za pomocą układu równań wielomianowych o wielu zmiennych w taki sposób, aby złożoność rozwiązania tego układu była mniejsza, niż złożoność wykładnicza.

W przeciwieństwie do metod statystycznych, kryptoanaliza algebraiczna wykorzystuje wewnętrzną strukturę szyfru. Ataki algebraiczne składają się z dwóch etapów. Pierwszym etapem jest przedstawienie procesu szyfrowania w postaci układu równań wielomianowych, najczęściej nad ciałem $GF(2)$, ale czasami również nad pierścieniami. Drugim etapem jest rozwiązanie skonstruowanego układu i odzyskanie tajnego klucza.

Większość szyfrów blokowych może być przedstawiona jako układ równań wielomianowych nad ciałem $GF(2)$. Jeśli w takim układzie wszystkie równania są co najwyżej kwadratowe, wtedy rozwiązaniem tego układu jest rozwiązanie problemu MQ (*ang. Multivariate Quadratic problem*) który, jak pokazał Shamir w [45], jest problemem NP-trudnym, ale jego złożoność maleje, gdy układ jest nadokreślony. Następnie w 2000 roku, Nicolas Courtois, Alexander Klimov, Jacques Patarin oraz Adi Shamir, na konferencji Eurocrypt 2000, zaprezentowali nowe algorytmy do rozwiązywania nadokreślonych układów równań wielomianowych o wielu zmiennych, które zostały przedstawione w [18]. Również w późniejszych latach zaproponowano szereg algorytmów do rozwiązywania układów nieliniowych równań wielomianowych o wielu zmiennych. Różnią się one przede wszystkim stopniem złożoności oraz możliwością

ich wykorzystania. Najpopularniejsze algorytmy obecnie stosowane w kryptoanalizie algebraicznej opisane są poniżej.

- Algorytmy Baz Gröbnera. Baza Gröbnera to zbiór, zwykle nieliniowych, wielomianów wielu zmiennych o własnościach, które pozwalają na proste, algorytmiczne podejście do wielu podstawowych problemów, takich jak między innymi rozwiązywanie algebraicznych układów równań. Przykładowa baza Gröbnera, składająca się z trzech wielomianów trzech zmiennych ma następującą postać:

$$\{x^3 - x^2 + 2, y - 2x^2 + 1, z - 3x + 5\}.$$

  Jedną z praktycznych właściwości baz Gröbnera jest to, że układ równań, w podstawowej formie Gröbnera:

$$\begin{cases} x^3 - x^2 + 2 = 0, \\ y - 2x^2 + 1 = 0, \\ z - 3x + 5 = 0. \end{cases}$$

  można łatwo rozwiązać. Najpierw znajdowane są trzy możliwe rozwiązania x pierwszego równania, a następnie każde z otrzymanych rozwiązań podstawiane jest do drugiego i trzeciego równania, co pozwala określić zmienne y i z. Podstawowym spostrzeżeniem i wkładem teorii baz Gröbnera jest to, że każdy układ wielomianowy, nieważne jak skomplikowany, może zostać przekształcony – za pomocą algorytmu – w formę bazy Gröbnera. Algorytmami, które realizują takie przekształcenie są:

  - algorytm Buchbergera, opracowany w 1965, którego podstawą jest algorytmicznie sformułowane weryfikowalne kryterium, sprawdzające czy zbiór wielomianów tworzy bazę Gröbnera; sam algorytm umożliwia obliczanie bazy Gröbnera z danego ideału;
  - algorytm F_4, który po raz pierwszy został opisany przez autora Jeana-Charlesa Faugere'a w pracy [28]; wprowadza strategię redukcji dla algorytmów bazy Gröbnera, która opiera się na połączeniu baz Gröbnera

z algebrą liniową i umożliwia redukcję jednocześnie kilku S-wielomianów, zamiast jednego na raz;

– algorytm F_5, który jest usprawnieniem algorytmu F_4, opracowany oraz przedstawiony również przez Jeana-Charlesa Faugere'a w pracy [29].

- Linearyzacja. Metoda linearyzacji jest powszechnie znaną metodą rozwiązywania dużych układów równań wielomianowych o wielu zmiennych. Niech będzie dany układ m równań kwadratowych, n zmiennych. Istnieje $\binom{n}{2}$ możliwych jednomianów kwadratowych oraz n możliwych jednomianów liniowych, czyli łącznie $\frac{n^2+n}{2}$. Metoda linearyzacji zakłada, że wszystkie jednomiany w układzie są niezależnymi zmiennymi. Stąd rozwiązywanie układu rozpoczyna się od zamiany nazw wszystkich jednomianów na nowe, w wyniku czego każdy jednomian jest nową, unikatową zmienną, a rozwiązywany układ jest układem liniowym. Wykonując eliminację Gaussa, otrzymujemy rozwiązanie. Ponieważ proces linearyzacji niszczy informacje o powiązaniach pomiędzy zmiennymi, można otrzymać kilka rozwiązań, które należy zweryfikować. Prawdą jest, że jeśli istnieje rozwiązanie pierwotnego układu równań wielomianowych, to będzie ono również rozwiązaniem jego linearyzacji (po obliczeniu poprawnych wartości nowych zmiennych). Jednak implikacja odwrotna jest fałszywa. Wiele rozwiązań liniowego układu równań może nie być rozwiązaniem oryginalnego układu. W przypadku kryptoanalizy prawdą jest, że rozwiązanie istnieje, ponieważ wiadomość została zaszyfrowana jakimś kluczem oraz wysłana. Układy liniowe nad ciałem $GF(2)$ mają $2^{n'-r}$ rozwiązań albo żadnego, gdzie n' jest liczbą zmiennych po linearyzacji, a r jest rzędem macierzy współczynników układu równań. Ponieważ linearyzacja nie niszczy żadnych rozwiązań oraz istnieje co najmniej jedno rozwiązanie układu nieliniowego, to układ liniowy ma dokładnie $2^{n'-r}$ rozwiązań. Jeśli rząd macierzy współczynników układu równań niewiele różni się od liczby niewiadomych, to istnieje tylko kilka rozwiązań (jedno, gdy obie liczby są równe). Wtedy w celu znalezienia poprawnego rozwiązania pierwotnego układu, można sprawdzić wszystkie otrzymane rozwiązania. Nie jest

możliwe, aby $r > n'$, ponieważ rząd macierzy nie może być większy od liczby kolumn. Gdy liczba równań wynosi $m = \frac{n^2+n}{2}$ lub niewiele więcej, możliwe jest wykonanie linearyzacji, w wyniku której otrzymamy tylko kilka kandydujących rozwiązań.

- Algorytm XL (*ang. eXtended Linearization*) został po raz pierwszy przedstawiony w [18] przez Nicolasa T. Courtois.

  Zacznijmy od przykładu. Niech dany będzie układ czterech równań kwadratowych o czterech zmiennych. Spośród wszystkich 10 jednomianów, 6 jest kwadratowych, a 4 liniowe. Ponieważ są tylko 4 równania, znacznie mniej niż 10, to linearyzacja wydaje się nieskuteczna. Załóżmy, że wygenerowano nowe, liniowo niezależne równania trzeciego stopnia. Liczba wszystkich możliwych wielomianów sześciennych wynosi: $\binom{n}{3} + \binom{n}{2} + \binom{n}{1} = \frac{n^3}{6} + \frac{5}{6}n$. Jednak w rozpatrywanym przypadku istnieją tylko $\binom{4}{3} = 4$ dodatkowe jednomiany sześcienne, więc wymagane byłoby 14, a nie tylko 10 równań. Algorytm XL generuje dodatkowe równania (w tym przypadku zakończymy z 20, a nie z 4 równaniami).

  Na wejściu algorytm XL przyjmuje układ m równań wielomianowych n zmiennych stopnia d oraz parametr D, najczęściej $D = d + 1$. Na początku algorytmu tworzona jest lista L wszystkich jednomianów stopnia co najwyżej $D - d$. Następnie wykonywane jest mnożenie wszystkich równań przez wszystkie jednomiany z listy L. W wyniku otrzymano $m|L|$ równań stopnia co najwyżej D. Kolejnym krokiem algorytmu jest linearyzacja układu, gdzie każdy jednomian zastępowany jest nową zmienną, aby otrzymać układ liniowy. Ostatnim krokiem jest rozwiązanie układu liniowego.

  Algorytm XL działa tylko dla układów określonych ($n = m$) oraz nadokreślonych ($n \leq m$). Efektywność działania algorytmu zależy od wielkości parametru D. Im wyższa wartość parametru D, tym większa jest liczba równań nowego układu, ale też większy układ do rozwiązania, na przykład za pomocą eliminacji Gaussa. Okazuje się jednak, że im bardziej nadokreślony układ równań, tym efektywniej działa algorytm XL.

Według pracy [18] złożoność algorytmu XL jest wielomianowa, gdy liczba równań m układu wynosi co najmniej ϵn^2, dla $0 < \epsilon \leq \frac{1}{2}$, a podwykładnicza, gdy m przekracza n.

Istnieją modyfikacje algorytmu XL, a do najbardziej znanych należą:

- FXL (*ang. Fixed XL*) [18] – na początku ustalana jest pewna liczba zmiennych układu z konkretnymi wartościami, a następnie stosowany jest algorytm XL do mniejszego układu;
- XSL (*ang. eXtended Sparse Linearization*) [19] – z pierwotnego układu równań wybierany jest pewien podzbiór równań oraz pewien podzbiór jednomianów, nowe równania generowane są poprzez przemnożenie wybranych równań przez iloczyn pewnej liczby wybranych jednomianów oraz poprzez przemnożenie wybranych równań przez pojedyncze zmienne, w taki sposób, aby nowe równania nie zawierały nowych jednomianów. Nowe równania generowane są tak długo, aż będzie możliwość zastosowania linearyzacji.

- Algorytm ElimLin zaprezentowali Nicolas T. Courtois oraz G.V. Bard w pracy [20]. Niech dany będzie układ wielomianów kwadratowych nad dowolnym ciałem. W przypadku układów równań stopnia większego niż dwa, należy wykonać dowolną redukcję stopnia, sprowadzając układ do układu kwadratowego.

  Pierwszym krokiem algorytmu ElimLin jest wykonanie linearyzacji układu równań, w taki sposób, aby po zapisaniu układu w postaci macierzowej wszystkie wyrazy odpowiadające jednomianom kwadratowym znajdowały się w skrajnych lewych kolumnach, a wyrazy odpowiadające jednomianom liniowym – w skrajnych prawych. Następnie wykonuje się eliminację Gaussa, w celu uzyskania zredukowanej postaci. Gdy podczas eliminacji Gaussa nie znaleziono równań liniowych, algorytm kończy działanie. W przeciwnym razie z każdego równania należy wybrać pojedynczą zmienną tak, aby każda z nich była unikatowa. Równania z unikatowymi zmiennymi należy tak przekształcić, aby unikatowa zmienna znalazła się po jednej stronie równości, a pozostałe wyrażenia po dru-

giej. Następnie w całym układzie zmienną unikatową zastępuje się jej definicją, a samą definicję zapamiętuje się w celu odtworzenia tej zmiennej po rozwiązaniu układu. Po wykonaniu wszystkich podstawień, ponownie wykonuje się eliminację Gaussa oraz, w przypadku znalezienia równań liniowych, kolejne kroki. Kolejne iteracje algorytmu wykonuje się dopóki nie zostaną znalezione równania liniowe lub nie zostaną wyeliminowane wszystkie zmienne.

- SAT-Solwery. Użycie SAT solwerów do rozwiązywania równań wielomianowych nad ciałami skończonymi zostało po raz pierwszy zaprezentowane przez G.V. Barda w [2]. Nie jest to nowy algorytm, ale wykorzystanie istniejących rozwiązań z dziedziny rachunku zdań logicznych. Idea polega na przekształceniu problemu rozwiązania nieliniowego układu równań nad ciałem $GF(2)$, opisującego szyfr, do problemu spełnialności SAT (*ang. Satisfiability Problem*). Problem spełnialności polega na znalezieniu takiego przyporządkowania wartości do zmiennych logicznych, aby dana formuła rachunku zdań logicznych była prawdziwa. Do rozwiązania problemu spełnialności wykorzystywane są narzędzia (programy lub biblioteki) zwane SAT-Solwerami. Aby wykorzystać SAT-Solwery do kryptoanalizy, nieliniowy układ równań należy przedstawić za pomocą koniunktywnej postaci normalnej CNF (*ang. Conjuctive Normal Form*). Transformacja do postaci CNF wykonywana jest w dwóch krokach. Najpierw nieliniowy układ wielomianów należy przekształcić do układu liniowego, a następnie otrzymany układ liniowy należy przedstawić za pomocą koniunktywnej postaci normalnej. Na koniec uruchamiany jest SAT-Solwer. Jeżeli jego działanie zakończy się sukcesem, to oznacza, że oryginalny układ równań został rozwiązany i szyfr został złamany.

Obecne narzędzia, wykorzystywane do ataków algebraicznych, nie umożliwiają przeprowadzenia ataku w czasie krótszym, niż atak pełnego przeszukiwania.

2 Wyżarzanie kwantowe jako metoda rozwiązywania problemów optymalizacyjnych

W niniejszym rozdziale w ogólnym zarysie przedstawia się teorię stojącą za komputerem kwantowym, wykorzystującym wyżarzanie kwantowe. Głównymi źródłami wykorzystanymi do opracowania tego rozdziału są prace [39], [49] oraz udostępniona dokumentacja komputera kwantowego D-Wave.

2.1 Podstawy obliczeń kwantowych

Najbardziej oczywistą różnicą między obliczeniami klasycznymi a kwantowymi jest pojęcie stanu. Stan obliczeń klasycznych najczęściej reprezentowany jest poprzez rejestr bitowy i utożsamiany jest z wartościami jego bitów. Klasyczny bit definiowany jest jako najmniejsza możliwa jednostka informacji, utożsamiana z wartościami binarnymi takimi, jak 0 albo 1, *Tak* albo *Nie*, *Prawda* albo *Fałsz*. Fizyczna realizacja klasycznego bitu polega na przypisaniu wartości binarnych do dwóch różnych stanów układu fizycznego.

W przypadku mechaniki kwantowej, która umożliwia jedynie wysunięcie spostrzeżeń dotyczących statystyki danego układu, stwierdzenie: *układ kwantowy znajduje się w danym stanie* oznacza, że układ kwantowy jest elementem układu statystycznego, którego obserwowalne statystyki można obliczyć za pomocą wybranego obiektu matematycznego, reprezentującego ten stan. Stan układu kwantowego, zgodnie z notacją Diraca, oznaczany jest jako *"ket"*, np. $|\psi\rangle$ i przedstawiany jest w postaci wektora kolumnowego, będącego elementem n-wymiarowej przestrzeni Hilberta $\mathbb{H}_n$. Każdemu elementowi przestrzeni Hilberta odpowiada element z przestrzeni dualnej $\mathbb{H}_n^*$, oznaczany jako *"bra"*, np. $\langle\psi|$ i przedstawiany w postaci wektora wierszowego. Iloczyn tych dwóch wektorów, oznaczany jako *braket* daje w wyniku liczbę zespoloną, równą iloczynowi skalarnemu $\langle\phi|\psi\rangle \in \mathbb{C}$. Aby opisać stan za pomocą obiektu matematycznego, którym jest macierz w przestrzeni Hilberta, najpierw należy wybrać mierzalną wielkość fizyczną układu kwantowego, w której można wyróżnić dwa rozróżnialne stany. Te rozróżnialne stany będą odpowiadać wartościom binarnym, przed-

stawianym za pomocą dwóch wektorów bazowych w dwuwymiarowej przestrzeni stanów. Ogólnie, przestrzeń stanów dla mikroskopowego obiektu jest nieskończenie wymiarową przestrzenią Hilberta. Dlatego też fizyczna realizacja dwuwymiarowej przestrzeni stanów polega na zdefiniowaniu dwuwymiarowych przestrzeni własnych, odpowiednio dobranej obserwowalnej wielkości fizycznej. Przykładami takich układów kwantowych i obserwowanych wielkości fizycznych z dwuwymiarowymi przestrzeniami własnymi są:

- elektrony i ich spin – ignorując położenie oraz pęd elektronu, mierzy się tylko jego stan spinowy, gdzie wartości binarne można przyporządkować do:
  - wektorów

    $$|0\rangle = |\uparrow_z\rangle,$$

    $$|1\rangle = |\downarrow_z\rangle,$$

    jako bazy ortonormalnej, która składa się ze stanów własnych operatora σ^z;
  - wektorów

    $$|+\rangle = |\uparrow_x\rangle = \frac{1}{\sqrt{2}}\left(|\uparrow_z\rangle + |\downarrow_z\rangle\right),$$

    $$|-\rangle = |\downarrow_x\rangle = \frac{1}{\sqrt{2}}\left(|\uparrow_z\rangle - |\downarrow_z\rangle\right),$$

    jako bazy ortonormalnej, która składa się ze stanów własnych operatora σ^x;
  - wektorów

    $$|+i\rangle = |\uparrow_y\rangle = \frac{1}{\sqrt{2}}\left(|\uparrow_z\rangle + i|\downarrow_z\rangle\right),$$

    $$|-i\rangle = |\downarrow_y\rangle = \frac{1}{\sqrt{2}}\left(i|\uparrow_z\rangle + |\downarrow_z\rangle\right),$$

    jako bazy ortonormalnej, która składa się ze stanów własnych operatora σ^y;
- fotony i ich polaryzacja – gdzie dla fotonów o zadanym kierunku propagacji, polaryzacja opisana jest tak zwanym wektorem polaryzacji czyli dwuwymiarowym wektorem o elementach ze zbioru liczb zespolonych, gdzie wartości binarne można przyporządkować do:
  - wektorów

$$|0\rangle = |H\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

który reprezentuje polaryzację poziomą oraz

$$|1\rangle = |V\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

który reprezentuje polaryzację pionową, tworzących bazę ortonormalną, która składa się ze stanów własnych operatora $\sigma^z = |H\rangle\langle H| - |V\rangle\langle V|$, gdzie ortogonalne odwzorowania $|H\rangle\langle H|$ i $|V\rangle\langle V|$ nazywane są poziomym i pionowym polaryzatorem;

– wektorów

$$|+\rangle = \frac{1}{\sqrt{2}}(|H\rangle + |V\rangle),$$
$$|-\rangle = \frac{1}{\sqrt{2}}(|H\rangle - |V\rangle),$$

tworzących bazę ortonormalną, składającą się ze stanów własnych polaryzatorów $|+\rangle\langle +|$ i $|-\rangle\langle -|$, obróconych o pół kąta prostego względem poziomego i pionowego polaryzatora;

– wektorów

$$|R\rangle = \frac{1}{\sqrt{2}}(|H\rangle + i|V\rangle),$$
$$|L\rangle = \frac{1}{\sqrt{2}}(i|H\rangle + |V\rangle),$$

tworzących bazę ortonormalną, składającą się ze stanów własnych lewo- i prawoskrętnych polaryzatorów kołowych.

Wspomniane wyżej operatory σ^x, σ^y i σ^z to tak zwane macierze Pauliego, które są obiektami matematycznymi, reprezentującymi obserwacje spinu elektronu wzdłuż danej osi trójwymiarowego układu współrzędnych. Macierze Pauliego to dwuwymiarowe macierze o elementach ze zbioru liczb całkowitych, oznaczone przez σ_j, gdzie $j \in \{x, y, x\}$ lub $\{1, 2, 3\}$, o następującej postaci:

$$\sigma^x = \sigma_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \sigma^y = \sigma_2 = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \sigma^z = \sigma_3 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \quad (1)$$

Dodatkowo wyróżnia się również macierz jednostkową $\sigma_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

Klasyczny bit może być zatem reprezentowany przez ortonormalną bazę w dwuwymiarowej przestrzeni Hilberta. Odpowiednikiem bitu klasycznego jest bit kwantowy, zwyczajowo nazywany kubitem. Wybór bazy dla reprezentacji kubitu jest dowolny, o ile można dobrać dla niej odpowiednią wielkość fizyczną, reprezentowaną przez macierz, której wektory własne są wektorami wybranej bazy. Wybierzmy zatem reprezentację bitu za pomocą wektorów $|0\rangle$ i $|1\rangle$, wraz z pomiarem stanu spinu elektronu, wzdłuż osi **z**. Klasycznym wartościom 0 i 1 przyporządkujmy wektor $|0\rangle$ dla wartości 0, oznaczający zwrot w stronę dodatnich wartości osi **z** oraz wektor $|1\rangle$ dla wartości 1, oznaczający zwrot w stronę wartości ujemnych. Następnie wektory bazowe przedstawmy w następujących postaciach:

$$|0\rangle = |\uparrow_z\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

$$|1\rangle = |\downarrow_z\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Wyznaczmy wartości własne macierzy σ^z dla wektora $|0\rangle$:

$$\sigma^z|0\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = +1|0\rangle,$$

oraz dla wektora $|1\rangle$:

$$\sigma^z|1\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} = -1|1\rangle.$$

Otrzymano wartość własną równą $+1$ dla wektora $|0\rangle$ oraz wartość własną równą -1 dla wektora $|1\rangle$.

Zatem, kubit jest układem mechaniki kwantowej, opisanym za pomocą dwuwymiarowej przestrzeni Hilberta. Informacja przechowywana w kubicie zawarta jest w stanie, w którym się znajduje. Przykładową reprezentacją kubitu jest baza ortonormalna $\{|0\rangle, |1\rangle\}$ wraz z macierzą σ^z, reprezentującą pomiar spinu elektronu wzdłuż

osi **z**. Macierz σ^z posiada znormalizowany wektor własny $|0\rangle$ z wartością własną $+1$ oraz znormalizowany wektor własny $|1\rangle$ z wartością własną -1. Jeśli następnie elektron zostanie odizolowany od jakichkolwiek oddziaływań, tzn. jeśli jego stan zostanie utrzymamy, to dokonując pomiaru otrzymujemy jedną z dwóch wartości własnych macierzy σ^z, względem ortonormalnej bazy $\{|0\rangle, |1\rangle\}$. Jeśli w wyniku pomiaru otrzymamy wartość $+1$, to kubit znajduje się w stanie $|0\rangle$, co odpowiada wartości 0 bitu klasycznego i analogicznie, otrzymanie z pomiaru wartości -1 oznacza to, że kubit znajduje się w stanie $|1\rangle$, co odpowiada klasycznej wartości 1. W dalszej części tego rozdziału domyślnie kubit będzie reprezentowany w dwuwymiarowej przestrzeni Hilberta z ortonormalną bazą $\{|0\rangle, |1\rangle\}$, której wektory nazywane będą stanami bazowymi.

W obliczeniach kwantowych wykorzystywane są dwie podstawowe własności kubitów: superpozycja oraz splątanie.

Dla układów dwuwymiarowych, oprócz stanów bazowych, mechanika kwantowa dopuszcza również stany postaci $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, gdzie $\alpha, \beta \in \mathbb{C}$ oraz $|\alpha|^2 + |\beta|^2 = 1$. Jest to zasada superpozycji mechaniki kwantowej, która mówi, że każda znormalizowana kombinacja liniowa stanów jest ponownie stanem. Takie liniowe kombinacje stanów nie mają odpowiednika w świecie klasycznych bitów. Dlatego też zakres informacji, które można przechowywać w dwuwymiarowym układzie kwantowym (kubicie), jest znacznie większy niż ten, który można przechowywać w klasycznym bicie. Stan $|\psi\rangle$, jest stanem superpozycji, co oznacza, że kubit może znajdować się jednocześnie w stanach 0 i 1, z pewnymi prawdopodobieństwami. Liczby α i β nazywane są amplitudami stanów bazowych, a stan superpozycji można przedstawić w postaci macierzowej:

$$|\psi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}. \tag{2}$$

Kluczowym faktem mechaniki kwantowej jest to, że stany superpozycji nie mogą być bezpośrednio obserwowane. Pomiar stanu kubitu względem danej bazy można wyobrazić sobie jako natychmiastowe zapadanie się kubitu do jednego ze stanów bazowych, co nazwano redukcją stanu kwantowego. Stan kubitu może redukować się

do stanu $|0\rangle$ z prawdopodobieństwem $|\alpha|^2$ – wtedy obserwowany jest klasyczny stan 0, albo do stanu $|1\rangle$ z prawdopodobieństwem $|\beta|^2$ – wtedy obserwowany jest klasyczny stan 1. Prawdopodobieństwa zaobserwowanych stanów spełniają zależność:

$$|\alpha|^2 + |\beta|^2 = 1, \tag{3}$$

gdzie $|\alpha| = |x + iy| = \sqrt{x^2 + y^2}$. Na podstawie zależności (3) można znaleźć wartości $\gamma, \delta, \theta \in \mathbb{R}$, za pomocą których można przedstawić liczby α i β w następującej postaci: $\alpha = e^{i\gamma} \cos\left(\frac{\theta}{2}\right)$ oraz $\beta = e^{i\delta} \sin\left(\frac{\theta}{2}\right)$. Wtedy stan superpozycji $|\psi\rangle$ można przedstawić za pomocą równania (4).

$$|\psi\rangle = e^{i\gamma} \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\delta} \sin\left(\frac{\theta}{2}\right)|1\rangle \tag{4}$$

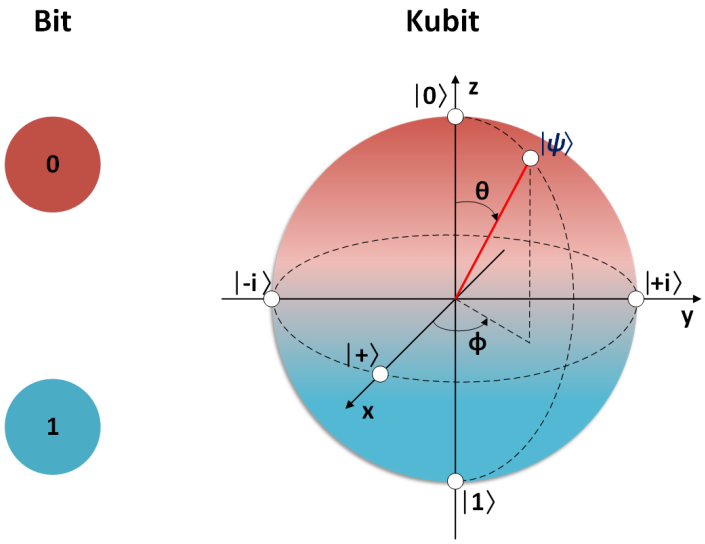
Ponieważ stany $|\psi\rangle$ i $e^{i\omega}|\psi\rangle$, dla dowolnego $\omega \in \mathbb{R}$, są fizycznie nierozróżnialne i opisują ten sam stan układu kwantowego, nazywane są stanami równoważnymi, a ω nazywana jest fazą globalną. Przykładowo, dla stanu $|\psi\rangle$ z równania (4) stanem równoważnym może być stan następującej postaci:

$$e^{-i\frac{\gamma+\delta}{2}}|\psi\rangle = e^{i\frac{\gamma-\delta}{2}} \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\frac{\delta-\gamma}{2}} \sin\left(\frac{\theta}{2}\right)|1\rangle \tag{5}$$

Teraz, jeśli przyjmiemy, że $\phi = \delta - \gamma$, to po podstawieniu tej zależności do równania (5) otrzymujemy stan superpozycji (6), w którym amplitudy stanów bazowych zależą od dwóch kątów ϕ i θ.

$$|\psi(\theta, \phi)\rangle = e^{-i\frac{\phi}{2}} \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\frac{\phi}{2}} \sin\left(\frac{\theta}{2}\right)|1\rangle, \tag{6}$$

gdzie kąty θ i ϕ odpowiadają kątowi biegunowemu i azymutalnemu współrzędnych sferycznych gdzie $0 \leq \theta \leq \pi$ i $0 \leq \phi \leq 2\pi$. Z równania (6) wynika, że superpozycję kubitu można przedstawić jako sferę o promieniu 1 w trójwymiarowej przestrzeni zespolonej. Taka reprezentacja kubitu nosi nazwę sfery Blocha. Na rysunku 1, w lewej części przedstawiono możliwe stany klasycznych bitów, a w prawej części zaprezentowano superpozycję kubitu. Dodatkowo, na rysunku za pomocą symbolu $\circ$ oznaczono przykładowe stany, oznaczone jako $|0\rangle, |1\rangle, |+\rangle, |+i\rangle, |-i\rangle, |\psi\rangle$, które są opisane za pomocą równania (6), przy odpowiednich wartościach kątów. Cała sfera reprezentuje



Rysunek 1: Możliwe wartości stanów, po lewej stronie klasycznych, po prawej – kwantowych bitów.

wszystkie możliwe stany kubitu, a spośród nich wszystkich tylko dwa stany, na biegunach wzdłuż osi **z** (dla $\theta = 0$ i $\theta = \pi$), odpowiadają klasycznym bitom.

Niech teraz pewien rejestr kwantowy składa się z dwóch kubitów $|\psi_1\rangle = [\alpha_1, \beta_1]^T$ oraz $|\psi_2\rangle = [\alpha_2, \beta_2]^T$. Stan rejestru dwóch kubitów jest czterowymiarową przestrzenią Hilberta $H_4 = H_2 \otimes H_2$, którą można przedstawić w następującej postaci:

$$|\psi\rangle = |\psi_1\psi_2\rangle = [\alpha_1, \beta_1]^T \otimes [\alpha_2, \beta_2]^T = [\alpha_1\alpha_2,\ \alpha_1\beta_2,\ \beta_1\alpha_2,\ \beta_1\beta_2]^T, \tag{7}$$

gdzie operator $\otimes$ jest iloczynem tensorowym. Przyjąwszy dla czterowymiarowej przestrzeni Hilberta następujące stany bazowe:

$$\begin{aligned} |00\rangle &= \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, |01\rangle = \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \\ |10\rangle &= \begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, |11\rangle = \begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}, \end{aligned} \tag{8}$$

stan dowolnego rejestru dwóch kubitów można przedstawić jako wektor:

$$|\psi\rangle = |\psi_1\psi_2\rangle = \gamma_1|00\rangle + \gamma_2|01\rangle + \gamma_3|10\rangle + \gamma_4|11\rangle = [\gamma_1, \gamma_2, \gamma_3, \gamma_4]^T, \tag{9}$$

gdzie $\gamma_1 = \alpha_1\alpha_2$, $\gamma_2 = \alpha_1\beta_2$, $\gamma_3 = \beta_1\alpha_2$, $\gamma_4 = \beta_1\beta_2$ oraz $|\gamma_1|^2 + |\gamma_2|^2 + |\gamma_3|^2 + |\gamma_4|^2 = 1$. Wartości $|\gamma_1|^2$, $|\gamma_2|^2$, $|\gamma_3|^2$ oraz $|\gamma_4|^2$ reprezentują prawdopodobieństwa z jakimi stan rejestru dwóch kubitów wynosi, odpowiednio, 00, 01, 10 oraz 11. W przypadku odczytu stanu jednego z dwóch kubitów rejestru, pierwszy kubit będzie znajdował się w stanie 0 z prawdopodobieństwem $|\gamma_1|^2 + |\gamma_2|^2$, a w stanie 1 – z prawdopodobieństwem $|\gamma_3|^2 + |\gamma_4|^2$, podczas gdy drugi kubit będzie znajdował się w stanie 0 z prawdopodobieństwem $|\gamma_1|^2 + |\gamma_3|^2$, a w stanie 1 – z prawdopodobieństwem $|\gamma_2|^2 + |\gamma_4|^2$.

Splątanie jest drugą własnością kubitów, odróżniającą obliczenia kwantowe od klasycznych. Splątane kubity tworzą jeden układ, co oznacza, że wartości ich amplitud stanów bazowych są powiązane, a odczyt stanu jednego z kubitów umożliwia poznanie stanu drugiego bez jego pomiaru, nawet jeśli kubity są od siebie nieskończenie oddalone. Niech $|\psi\rangle \in H_4$ będzie stanem rejestru dwóch kubitów. Jeżeli stan $|\psi\rangle$ można przedstawić w postaci iloczynu tensorowego dwóch stanów $|\psi_1\rangle, |\psi_2\rangle \in H_2$ takich, że $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$, to stan $|\psi\rangle$ jest stanem rozkładalnym. W przeciwnym razie stan $|\psi\rangle$ jest stanem nierozkładalnym, a więc stanem splątanym. Najbardziej znanym przykładem zjawiska splątania jest stan Bella, który opisuje splątanie w następujący sposób:

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \tag{10}$$

Stan Bella jest oczywiście stanem nierozkładalnym. Aby to pokazać, załóżmy coś przeciwnego – że stan Bella jest rozkładalny, a w związku z tym istnieją liczby zespolone α_1, α_2, β_1 oraz β_2 takie, że

$$\begin{aligned} &\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \big(\alpha_1|0\rangle + \alpha_2|1\rangle\big)\big(\beta_1|0\rangle + \beta_2|1\rangle\big) = \\ &= \alpha_1\beta_1|00\rangle + \alpha_1\beta_2|01\rangle + \alpha_2\beta_1|10\rangle + \alpha_2\beta_2|11\rangle. \end{aligned} \tag{11}$$

Wówczas spełniony musiałby być następujący układ równań:

$$\begin{cases} \alpha_1\beta_1 = \frac{1}{\sqrt{2}}, \\ \alpha_1\beta_2 = 0, \\ \alpha_2\beta_1 = 0, \\ \alpha_2\beta_2 = \frac{1}{\sqrt{2}}. \end{cases} \tag{12}$$

Ponieważ układ (12) jest układem sprzecznym, stan Bella: $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ jest stanem nierozkładalnym. Na podstawie układu (12) można zauważyć, że w stanie Bella prawdopodobieństwo zaobserwowania stanów $|01\rangle$ lub $|10\rangle$ wynosi zero, a prawdopodobieństwo zaobserwowania stanów $|00\rangle$ lub $|11\rangle$ wynosi $\frac{1}{2}$. Dodatkowo, odczyt pojedynczego kubitu daje w wyniku 0 lub 1 z jednakowym prawdopodobieństwem równym $|\frac{1}{\sqrt{2}}|^2 + |0|^2 = |0|^2 + |\frac{1}{\sqrt{2}}|^2 = \frac{1}{2}$.

Niech teraz Q oznacza rejestr kwantowy, składający się z n kubitów. Stan superpozycji rejestru Q można przedstawić jako wektor $|\psi\rangle$ o długości $N = 2^n$. Deutsch, w swojej publikacji [23] pokazał, że w celu zrealizowania dowolnej klasycznej funkcji f, zdefiniowanej na n bitach, można zbudować układ kwantowy składający się z odwracalnych bramek kwantowych. Niech C_f oznacza układ kwantowy zastosowany dla rejestru Q. Wynikiem jest nowy stan superpozycji, $C_f|\psi\rangle \rightarrow |\psi'\rangle$. Układ C_f można przedstawić jako macierz o wymiarach $N \times N$. Przykładowo, kontrolowana bramka NOT, oznaczana jako C_{cnot}, na wejściu przyjmuje dwa kubity, a wyjście zależy od pierwszego kubitu (kontrolnego). Jeśli pierwszy kubit ma stan 1, to drugi kubit (docelowy) zmienia swój stan, a jeśli pierwszy kubit ma stan 0, to stan drugiego kubitu zostaje niezmieniony. Stąd bramka C_{cnot} realizuje następujące przejścia:

$$\begin{aligned} C_{cnot}|00\rangle &\rightarrow |00\rangle, \\ C_{cnot}|01\rangle &\rightarrow |01\rangle, \\ C_{cnot}|10\rangle &\rightarrow |11\rangle, \\ C_{cnot}|11\rangle &\rightarrow |10\rangle. \end{aligned}$$

Bramkę C_{cnot} można przedstawić również w postaci następującej macierzy:

$$C_{cnot} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Model obliczeń kwantowych z bramką kwantową został opracowany wokół koncepcji rejestru Q, składającego się z n kubitów oraz wykorzystywanego w układach kwantowych, realizujących obliczenia szeregowo i równolegle. Proces obliczeń obejmuje etap inicjalizacji rejestru Q pewnym znanym stanem, etap transformacji stanów przy wykorzystaniu układów kwantowych oraz ostatni etap, polegający na pomiarze stanu rejestru Q. Pomiar wywołuje redukcję stanu kwantowego i otrzymany stan traktowany jest jako wynik obliczeń. Analiza otrzymanego wyniku wymaga znalezienia ograniczenia, zarówno wymaganego czasu obliczeń, jak i prawdopodobieństwa, że wynik jest poprawny. Ze względu na własność superpozycji kubitów, rejestr Q może jednocześnie przechowywać wszystkie 2^n stanów, a ze względu na własność splątania kubitów, układ kwantowy może przetwarzać wszystkie n kubitów, czyli wszystkie 2^n stanów, w stałym czasie. W przeciwieństwie do tego, układ w klasycznym modelu obliczeniowym, gdzie pojedynczy rejestr może przechowywać tylko jeden bit, musiałby zaktualizować 2^n rejestrów, aby osiągnąć taki sam efekt procesu przetwarzania stanu klasycznego rejestru.

Niech teraz rejestr kwantowy Q, składający się z n kubitów, będzie elementem układu dynamicznego, który ewoluuje w czasie, zgodnie z działającymi na niego siłami zewnętrznymi oraz wewnętrznymi. Ponadto, niech stan superpozycji rejestru Q, zmieniającego się w czasie t, oznaczony będzie jako $|\Psi_t\rangle$ oraz zdefiniowany zgodnie z równaniem (9). Siły działające na układ w dowolnym momencie t można scharakteryzować za pomocą zmiennego w czasie hamiltonianu $\mathcal{H}_t$, który w ujęciu matematycznym jest macierzą hermitowską wymiaru $N \times N$. Operator $\mathcal{H}_t$ umożliwia wyznaczenie obserwowalnej energii układu kwantowego, która jest wartością rzeczywistą. Stąd wartość oczekiwana hamiltonianu $\mathcal{H}_t$ odpowiada wartości oczekiwanej energii układu w stanie $|\Psi_{t'}\rangle$. Zbiór wszystkich możliwych wartości energii danego układu nazwano widmem energii, którego moc wynosi co najwyżej N. Jeżeli co naj-

Rysunek 2: Modele komputerów kwantowych. *Źródło: [1]*

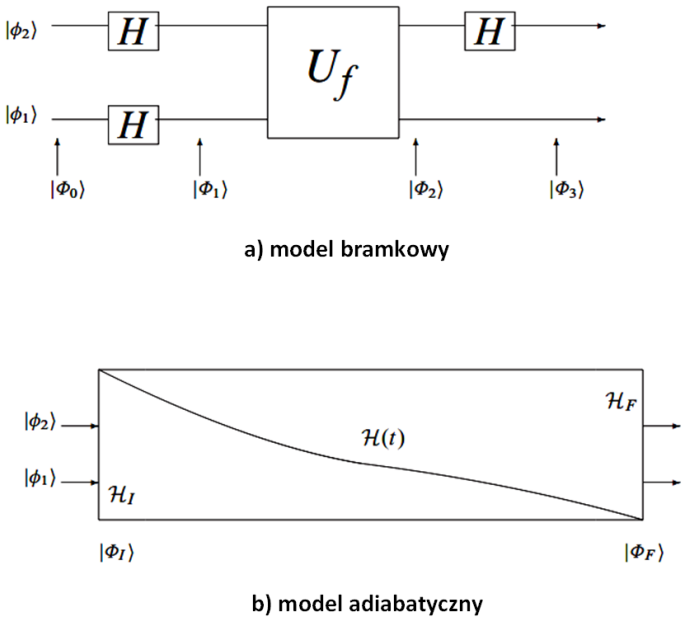
mniej dwóm stanom kwantowym odpowiada ta sama wartość energii, to stany te są nazwane stanami zdegenerowanymi. Stan podstawowy s_g (*ang. ground state*), jest stanem obserwowalnym, w którym stany bazowe osiągają minimalną energię. Stan niebędący stanem podstawowym nazwano stanem wzbudzonym (*ang. excited state*). Pierwszy stan wzbudzony ma minimalną energię spośród wszystkich stanów wzbudzonych układu (drugą pod względem energii w układzie). W tym kontekście, oznaczenie hamiltonianu można interpretować jako:

- operator zmieniający stan układu: $\mathcal{H}_t|\psi_t\rangle \rightarrow |\psi_{t'}\rangle$,
- stan własny $\mathcal{H}_t$, odpowiadający obserwowalnym stanom układu, których wartości własne odpowiadają energiom tych stanów własnych. Stąd, jeśli $|\psi_t\rangle$ jest stanem własnym, to można zmierzyć jego energię λ jako $\mathcal{H}_t|\psi_t\rangle \rightarrow \lambda|\psi_t\rangle$.

2.2 Modele komputerów kwantowych

Obecnie istnieją dwa główne modele komputerów kwantowych: komputery kwantowe ogólnego przeznaczenia oraz komputery kwantowe szczególnego przeznaczenia (wykorzystujące wyżarzanie kwantowe). Najbardziej znanym modelem obliczeń, na którym oparto konstrukcję komputera kwantowego ogólnego przeznaczenia, jest model bramkowy. Jednak istnieje kilka modeli obliczeń kwantowych, które okazały się równoważne z modelem bramkowym i wszystkie one znane są jako komputer kwantowy ogólnego przeznaczenia lub, inaczej jako uniwersalny komputer kwantowy. Modele te przedstawiono na rysunku 2.

Komputer kwantowy ogólnego przeznaczenia, to urządzenie zdolne do wyko-



Rysunek 3: Schemat algorytmu (a) w modelu bramkowym (b) w modelu adiabatycznym. *Źródło: [49]*

nywania obliczeń za pomocą układów, składających się z uniwersalnych zestawów bramek kwantowych. Uniwersalny komputer kwantowy jest czasami określany jako kwantowa maszyna Turinga. Aby model został sklasyfikowany jako uniwersalny komputer kwantowy, należy wykazać, że istnieje wielomianowe odwzorowanie czasu obliczeń i zasobów z modelu bramkowego do danego modelu. Takie odwzorowanie istnieje dla modelu topologicznego, modelu bazy pomiarowej oraz modelu adiabatycznego.

Komputer kwantowy, wykorzystujący wyżarzanie kwantowe, nie jest uniwersalnym komputerem kwantowym, ale jest powiązany z jednym z jego modeli – modelem adiabatycznym, który opiera się na wyznaczaniu stanu układu o najniższej energii.

Jedną z różnic pomiędzy modelem bramkowym, a modelem adiabatycznym, jest ich dyskretna i analogowa natura, co powoduje bardzo odmienny schemat i działanie algorytmów, co pokazano na rysunku 3. Schemat a) przedstawia uproszczony schemat algorytmu Deutsch-Josza, jednego z pierwszych efektywnych algorytmów, zaprojektowanych dla modelu bramkowego. Schemat ten przypomina klasyczny schemat układu z bramkami logicznymi, gdzie poziome linie podążają w czasie za dwoma kubitami, a prostokąty reprezentują układy kwantowe, które zmieniają stany kubitów.

Oznaczenia wzdłuż dolnej linii reprezentują stany w dyskretnych momentach procesu, przed i po zastosowaniu układów kwantowych.

Schemat b) przedstawia proces obliczeń w modelu adiabatycznym, który ma postać analogową. Nie występują w nim bramki ani dyskretne kroki czasowe. Zamiast tego, stany kubitów ewoluują stopniowo, zgodnie z pewnymi siłami reprezentowanymi przez hamiltoniany. Algorytm wykonuje przejście od początkowego hamiltonianu $\mathcal{H}_I$ do końcowego hamiltonianu $\mathcal{H}_F$, a stany kubitów są odczytywane na końcu procesu.

2.3 Adiabatyczne obliczenia kwantowe

Teoria obliczeń adiabatycznych została sformułowana przez Borna i Focka w pracy [9], w celu opisania pewnych własności procesów cząstek kwantowych, które ewoluują zgodnie z równaniem Schrödingera:

$$i\hbar\frac{d}{dt}|\Psi_t\rangle = \mathcal{H}(t)|\Psi_t\rangle, \tag{13}$$

gdzie i jest jednostką urojoną oraz $\hbar = \frac{h}{2\pi}$, gdzie h jest stałą Plancka, a $\mathcal{H}(t)$ oznacza algorytm realizujący obliczenia w modelu adiabatycznym. Równanie (13) przedstawia proporcjonalną zależność pomiędzy chwilową prędkością zmian procesu (pochodną cząstkową wektora stanu), a energią w procesie.

Algorytmy adiabatycznych obliczeń kwantowych zaprojektowano do rozwiązywania problemów optymalizacyjnych sformułowanych następująco:

Dana jest funkcja celu $f : \mathbb{D}^n \to \mathbb{R}$ *zdefiniowana dla n zmiennych* $x = x_1 ... x_n$ *z dziedziny dyskretnej* $\mathbb{D}$. *Znajdź wartości zmiennych x minimalizujących funkcję* $f(x)$.

Algorytm adiabatycznych obliczeń kwantowych, rozwiązujący pewien problem optymalizacyjny P, z funkcją celu $f(x)$, realizuje obliczenia na n-kubitowym rejestrze Q, znajdującym się w stanie superpozycji $|\Psi_t\rangle$ w czasie t. Algorytm opisywany jest za pomocą zmieniającego się w czasie hamiltonianu $\mathcal{H}(t)$, określonego przez trzy komponenty:

- hamiltonian początkowy $\mathcal{H}_I$, który należy wybrać tak, aby stan podstawowy układu był łatwy do znalezienia;
- hamiltonian końcowy $\mathcal{H}_F$, który koduje funkcję celu w taki sposób, żeby stan podstawowy rejestru Q był stanem własnym hamiltonianu końcowego o minimalnej wartości własnej, co oznacza, że stan podstawowy s_g rejestru Q będzie odpowiadał optymalnemu rozwiązaniu problemu P;
- ścieżka ewolucji adiabatycznej, opisana jako funkcja $s(t) = \frac{t}{t_f}$, która rośnie od 0 do 1, w czasie $t : 0 \to t_f$, dla pewnego upływu czasu t_f.

Hamiltonian $\mathcal{H}(t)$ generuje stopniowe przejście od hamiltonianu początkowego $\mathcal{H}_I$ do hamiltonianu końcowego $\mathcal{H}_F$, zgodnie z zależnością:

$$\mathcal{H}(t) = (1 - s(t))\,\mathcal{H}_I + s(t)\mathcal{H}_F. \tag{14}$$

Pojęciem przerwy energetycznej, oznaczonej przez $\delta(s)$, zdefiniowano różnicę pomiędzy wartościami własnymi stanu podstawowego, a pierwszego stanu wzbudzonego, w dowolnym momencie $s(t)$ ewolucji adiabatycznej. Przez δ_m oznaczono minimalną przerwę energetyczną, gdzie $\delta_m = \min_s(\delta(s))$. Niech $|\Psi_g\rangle$ oznacza stan podstawowy końcowego Hamiltonianu. Zgodnie z twierdzeniem adiabatycznym:

- jeśli rejestr kwantowy Q jest w stanie podstawowym w momencie $s = 0$ oraz
- jeśli minimalna przerwa energetyczna δ_m jest większa niż 0 w dowolnym momencie $s(t)$ ewolucji adiabatycznej oraz
- jeśli proces ewoluuje wolno,

to proces zakończy się w stanie podstawowym $|\Psi_{t_f} = \Psi_g\rangle$ z prawdopodobieństwem co najmniej $1 - \left(\frac{C(1)}{t_f}\right)^2$ [57], gdzie t_f jest czasem przejścia oraz

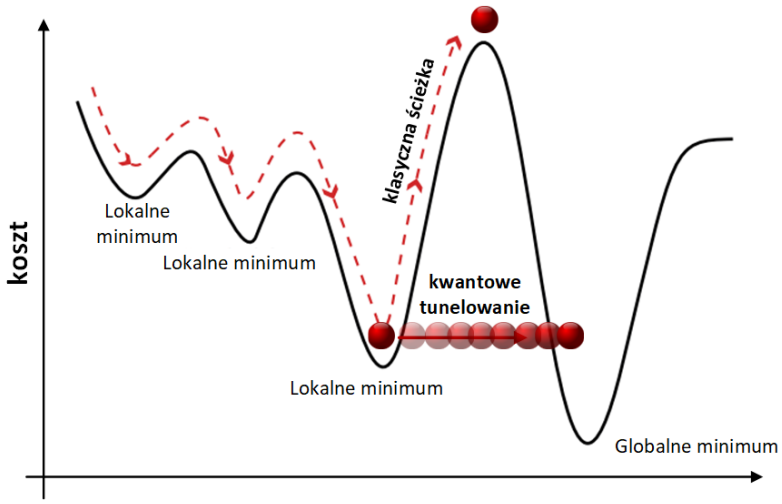
$$C(1) = \frac{\|\dot{\mathcal{H}}(t_f)\|}{\delta(1)^2} + \frac{\|\dot{\mathcal{H}}(0)\|}{\delta(0)^2} + \int_0^1 \left(\frac{\|\ddot{\mathcal{H}}(u)\|}{\delta(u)^2} + 10\frac{\|\dot{\mathcal{H}}(u)\|^2}{\delta(u)^3} \right) du,$$

gdzie $\|\cdot\|$ oznacza normę w przestrzeni Hilberta, wyznaczaną jako pierwiastek kwadratowy z iloczynu skalarnego oraz $\dot{\mathcal{H}}(t)$ i $\ddot{\mathcal{H}}(t)$ oznaczają pochodną, odpowiednio, pierwszego i drugiego rzędu.

2.4 Wyżarzanie kwantowe

Wyżarzanie kwantowe (*QA, ang. Quantum annealing*) jest heurystycznym podejściem do rozwiązywania problemów optymalizacji kombinatorycznej. Idea włączenia wyżarzania kwantowego do heurystycznego modelu optymalizacji, pojawiła się w pracach Finnila [31] oraz Kadowaki i Nishimori [43]. Algorytm wyżarzania kwantowego ma na celu rozwiązanie, najczęściej NP-trudnego, problemu optymalizacji kombinatorycznej.

Ponieważ wyżarzanie kwantowe jest odmianą przeszukiwania heurystycznego, zacznijmy od przybliżenia tego podejścia do rozwiązywania problemów optymalizacyjnych. Celem metody przeszukiwania heurystycznego jest przede wszystkim uniknięcie szukania na ślepo. Główną ideą jest wykorzystanie wszelkich informacji, które mogą poprawić efektywność procesu przeszukiwania. Niech dany będzie problem minimalizacji P, zdefiniowany dla n zmiennych binarnych $x_1 ... x_n$ oraz dla funkcji celu $f(x)$, która każdemu rozwiązaniu $x \in \mathbb{B}^n = \{0,1\}^n$ przypisuje jego koszt. Na początku zakładamy, że każde $x \in \mathbb{B}^n$ jest prawdopodobnym rozwiązaniem. Przestrzeń rozwiązań $\mathbb{B}^n$ można przedstawiać jako graf, w którym węzły reprezentują rozwiązania, a krawędzie definiowane są przez reguły sąsiedztwa. Funkcja celu przedstawiana jest za pomocą wykresu, nazywanego krajobrazem (*ang. landscape*), w którym szczyty odpowiadają rozwiązaniom o wysokim, a doliny – rozwiązaniom o niskim koszcie. Na rysunku 4 pokazano przykładowy wykres funkcji celu. Algorytm przeszukiwania rozpoczyna się w pewnym początkowym węźle w przestrzeni, a następnie, iterując, przechodzi od węzła do sąsiedniego węzła, kierując się w kierunku regionów o niskich kosztach, aż do zakończenia procesu, zdefiniowanego regułą zatrzymania. Wybór zasad dobrego sąsiedztwa, które pomogą odnieść sukces w poszukiwaniach, nie jest łatwy. Węzły przestrzeni rozwiązań muszą być połączone, a ich zasięg powinien mieć małą średnicę, jednak zbyt dużo połączeń może spowodować, że algorytm nigdy nie zapuści się daleko od węzła początkowego. Dlatego preferowane są *gładkie* wykresy funkcji celu, aby optymalne rozwiązania miały sąsiadów z małą różnicą kosztu, co tworzy szeroką "dolinę"wokół globalnego optimum, która jest łatwiejsza do zlokalizowania, niż wąska "przepaść". Zazwyczaj preferowane są reguły sąsiedztwa,



Rysunek 4: Przykładowy wykres funkcji celu pewnego problemu minimalizacji, wraz z dwoma podejściami szukania minimum, klasycznym i kwantowym. *Źródło: [49]*

które łączą rozwiązania o niewielkich przyrostowych różnicach, np. małej odległości Hamminga. Z drugiej jednak strony, wykres z dużą liczbą sąsiadów o niemal równej wartości jest trudny do przemierzenia, ponieważ algorytm nie może stwierdzić, czy robi postępy. Jeśli wykres jest gładki i ogólnie maleje w kierunku optymalnego rozwiązania, dobrze sprawdza się zachłanne przeszukiwanie, które zawsze prowadzi do sąsiada o najniższych kosztach. Ale jeśli wykres zawiera wiele lokalnych minimów, zachłanne przeszukiwanie może utknąć, ciągle iterować i nigdy nie znaleźć optymalnego rozwiązania. Dlatego też zaproponowano ogromną liczbę metod pozwalających na wyprowadzenie poszukiwań z lokalnych minimów, w kierunku obszarów rozwiązań z niskim kosztem. Metody te ogólnie nazywane są metodami przeszukiwania heurystycznego. Przykłady tych metod można znaleźć w pracach [27] lub [50].

Wyżarzanie kwantowe można postrzegać jako rodzaj heurystycznego przeszukiwania, które przebiega po wykresie funkcji celu, szukając nisko położonych obszarów. Aby zbadać krajobraz problemu optymalizacyjnego, metoda wyżarzania kwantowego wykorzystuje fluktuacje kwantowe, które polegają na zmianie ilości energii w punkcie przestrzeni, w bardzo krótkich odstępach czasu, co jest konsekwencją zasady nieoznaczoności sformułowanej przez Heisenberga. Podczas szukania minimum w problemie minimalizacji, *stan bieżący* lub potencjalne rozwiązanie zastępowane jest przez wy-

brany losowo *stan sąsiedni*, który ma mniejszy koszt. Aby umożliwić wyjście z lokalnych minimów, wyżarzanie kwantowe wykorzystuje efekt kwantowego tunelowania, który wykorzystując dualizm materii, pozwala przejść przez międzystanowe bariery potencjału, zamiast próbować je pokonać wspinając się po nich przyrostowo. Pozwala to algorytmowi poruszać się szybciej i dalej w całej przestrzeni rozwiązań, na wczesnym etapie procesu. Zjawisko tunelowania kwantowego, w kontekście szukania minimum problemu optymalizacyjnego, pokazano na rysunku 4. Decyzja o kierunku kroku wykonywanego w kolejnych iteracjach, opiera się na współczynniku pola poprzecznego, nazywanego również współczynnikiem tunelowania i oznaczanym literą Γ. Współczynnik ten określa szerokość tunelowania kwantowego, czyli promień obejmujący sąsiednie stany do zbadania. Na początku promień tunelowania jest na tyle szeroki, że sąsiedztwo obejmuje całą przestrzeń przeszukiwania, ale w miarę upływu czasu jest on stopniowo zmniejszany.

Jak już wcześniej wspomniano, funkcja celu $f(x)$ może być reprezentowana przez końcowy hamiltonian $\mathcal{H}_F$, który jest $N \times N$ wymiarową macierzą hermitowską. Wyżarzanie kwantowe wprowadza hamiltonian pola poprzecznego, zwany także hamiltonianem nieuporządkowanym $\mathcal{H}_D$. Jest on skalowany przez współczynnik tunelowania $\Gamma(t)$, który inicjalizowany jest dużą wartością, a następnie stopniowo redukowany w czasie do 0. Stąd, hamiltonian układu, zależny od czasu, dla wyżarzania kwantowego zdefiniowany jest następująco:

$$\mathcal{H}(t) = \Gamma(t)\mathcal{H}_D + \mathcal{H}_F. \tag{15}$$

Hamiltonian $\mathcal{H}_D$ wprowadza do procesu wyżarzania energię kinetyczną, w postaci kwantowych fluktuacji przestrzeni rozwiązań. Zmniejszanie współczynnika $\Gamma(t)$ przybliża układ do hamiltonianu końcowego $\mathcal{H}_F$, jednocześnie tłumiąc fluktuacje kwantowe. Podobieństwo równania (15) do algorytmu adiabatycznych obliczeń kwantowych jest oczywiste. Hamiltoniany mają ten sam cel. Ortogonalny hamiltonian $\mathcal{H}_D$ wyżarzania kwantowego wprowadza nieporządek, aby umożliwić poszukiwaniom heurystycznym ucieczkę od lokalnych minimów, co jest analogiczne do początkowego hamiltonianu $\mathcal{H}_I$, zdefiniowanego w celu zapewnienia, że początkowe stany super-

pozycji są jednakowo prawdopodobne. Parametr skalujący $\Gamma(t)$ tworzy określony typ ścieżki adiabatycznej. Pod tym względem algorytmy wyżarzania kwantowego są rodzajami algorytmów adiabatycznych obliczeń kwantowych. Istnieją jednak pewne różnice, wynikające z niemożliwości fizycznej realizacji rygorystycznych wymagań modelu adiabatycznych obliczeń kwantowych. Przykładowo, podczas rzeczywistej ewolucji stanu kwantowego mogą występować dodatkowe zachowania dynamiczne, co może spowodować, że układ nie będzie znajdował się w chwilowym stanie własnym. Ponadto, gdy dynamika ewolucji czasowej, generowanej przez hamiltonian (15), jest niewystarczająco wolna, stan kwantowy będzie mieszał się z pobliskimi stanami własnymi energii, co zmniejsza prawdopodobieństwo zaobserwowania poprawnego wyniku. Ostatnią przytoczoną różnicą jest to, że w adiabatycznych obliczeniach kwantowych wyznacza się granicę prawdopodobieństwa zakończenia procesu w stanie podstawowym, zakładając, że układ startuje ze stanu podstawowego. Natomiast w algorytmach wyżarzania kwantowego analizuje się prawdopodobieństwo zbieżności do rozwiązania w pewnym zakresie optymalnego, zaczynając od dowolnego stanu.

2.5 Model Isinga oraz problem QUBO

Model Isinga to model matematyczny używany do badania własności układów fizycznych, które ewoluują w czasie. Model składa się ze zbioru n zmiennych dyskretnych, które reprezentują cząsteczki. Cząsteczki ułożone są zwykle w d-wymiarową sieć, co umożliwia każdej z nich interakcję z sąsiadami. Sieć cząsteczek można reprezentować za pomocą grafu $G = (V, E)$, gdzie zbiór wierzchołków V jest zbiorem cząsteczek, a zbiór krawędzi E reprezentuje interakcje pomiędzy cząsteczkami. Każda cząstka może znajdować się w jednym z dwóch stanów, zwanych spinami, reprezentowanym przez wartość ± 1. Konfiguracja spinowa $s = s_1 ... s_n$ jest przypisaniem wartości spinu do cząstek. Energia danej konfiguracji spinowej jest równa energii własnej hamiltonianu, zdefiniowanego w następujący sposób:

$$H(s) = \sum_i h_i s_i + \sum_{i<j} J_{ij} s_i s_j, \tag{16}$$

gdzie przez h_i określono siły zewnętrzne przyłożone do poszczególnych cząstek, a przez J_{ij} określono siły interakcji pomiędzy sąsiadami w siatce. Jeśli cząstki o indeksach i oraz j nie sąsiadują ze sobą, to współczynnik $J_{ij} = 0$. Znaki wag wpływają na odpowiednie wartości spinów: składnik z ujemnym współczynnikiem h_i zmniejsza energię układu, gdy $s_i = 1$, a składnik z dodatnim h_i – gdy $s_i = -1$; składnik z dodatnim współczynnikiem J_{ij} zmniejsza energię układu gdy $s_i \neq s_j$, a składnik z ujemnym współczynnikiem J_{ij} gdy $s_i = s_j$. Wykorzystując model Isinga do rozwiązania problemów optymalizacyjnych, dąży się do znalezienia stanu podstawowego, czyli takiej konfiguracji spinowej, która minimalizuje funkcję (16). W [40] pokazano, że problem ten jest problemem NP-trudnym, jeśli graf G jest niepłaski, czyli gdy sieć cząstek jest wymiaru co najmniej trzeciego.

Załóżmy, że mamy do dyspozycji zbiór n kubitów rejestru $Q = q_1 ... q_n$, rozłożonych w węzłach d-wymiarowej siatki $G = (V, E)$, gdzie każdy kubit q_i jest zdolny do interakcji z sąsiadami w sieci. Stan superpozycji rejestru Q jest opisany za pomocą wektora $|\Psi\rangle = |\psi_1 ... \psi_n\rangle$. Hamiltonian problemu w postaci modelu Isinga dla algorytmu wyżarzania kwantowego zdefiniowany jest następująco:

$$\mathcal{H}_F = \mathcal{H}_{Ising} = \sum_{i \in V} h_i {\sigma_i}^z + \sum_{(i,j) \in E} J_{ij} {\sigma_i}^z {\sigma_j}^z, \tag{17}$$

gdzie notacja ${\sigma_i}^z$ oznacza zastosowanie macierzy Pauliego σ^z do i-tego kubitu. Aby wyznaczyć energię układu reprezentującego problem w postaci modelu Isinga, należy wyznaczyć energię własną hamiltonianu (17) w stanie $|\Psi\rangle$:

$$\mathcal{H}_{Ising} |\Psi\rangle = \left(\sum_{i \in V} h_i {\sigma_i}^z + \sum_{(i,j) \in E} J_{ij} {\sigma_i}^z {\sigma_j}^z \right) |\Psi\rangle. \tag{18}$$

Ponieważ wartość własna macierzy Pauliego σ^z ma wartość $+1$, gdy stan kubitu wynosi $|0\rangle$ oraz -1 dla kubitu w stanie $|1\rangle$, przyjąwszy, że kubit w stanie $|0\rangle$ reprezentuje cząstkę ze spinem $+1$, a kubit w stanie $|1\rangle$ – cząstkę ze spinem -1, energię układu można wyznaczyć następująco:

$$\mathcal{E}_{Ising} = \sum_{i \in V} h_i (-1)^{\psi_i} + \sum_{(i,j) \in E} J_{ij} (-1)^{\psi_i} (-1)^{\psi_j} = \sum_{i \in V} h_i s_i + \sum_{(i,j) \in E} J_{ij} s_i s_j. \tag{19}$$

Model Isinga, zdefiniowany na spinach $s_i = \pm 1$, jest ściśle powiązany z problemem, dotyczącym zmiennych binarnych $x_i \in \{0, 1\}$, nazwanym problemem QUBO (*ang. Quadratic Unconstrained Binary Optimization*) oraz zdefiniowanym w następujący sposób:

Dana jest macierz wag Q_{ij} o wymiarach $N \times N$ oraz elementach ze zbioru liczb rzeczywistych. Znajdź takie przypisanie wartości binarnych do N zmiennych x_i, dla $i \in \{1, ..., N\}$, które minimalizuje następującą funkcję:

$$f_{qubo}(x) = \sum_i Q_{i,i} x_i + \sum_{i<j} Q_{i,j} x_i x_j = x^T \mathbf{Q} x. \tag{20}$$

Transformacja pomiędzy problemem w postaci modelu Isinga oraz problemem w postaci QUBO jest prosta i polega na wykorzystaniu zależności: $s_i = 1 - 2x_i$ oraz odpowiednim dostosowaniu wag. Wiele problemów optymalizacyjnych można w naturalny sposób sformułować w postaci problemu QUBO. Przykładowo, do takich problemów należą:

- problem podziału zbioru liczb – powszechna wersja tego problemu polega na podzieleniu zbioru liczb na dwa podzbiory, tak aby sumy podzbiorów były jak najbliżej siebie;
- problem podziału grafu (Max Cut) – jest jednym z najbardziej znanych problemów optymalizacji kombinatorycznej, opisanym w następujący sposób: dla danego grafu nieskierowanego $G = (V, E)$ ze zbiorem wierzchołków V i zbiorem krawędzi E, należy podzielić zbiór V na dwa podzbiory w taki sposób, aby liczba krawędzi pomiędzy tymi dwoma podzbiorami była jak największa.

Istnieje wiele innych problemów, które nie wydają się być związane z problemami w postaci QUBO, ale można je przeformułować do tego modelu. W pracy [34] przedstawiono sposoby pozwalające na formułowanie różnych problemów w postaci problemu QUBO. Jedną z opisanych tam metod jest transformacja do modelu QUBO z zastosowaniem kary.

Problem optymalizacyjny w postaci QUBO, zgodnie z jego definicją, nie zawiera żadnych ograniczeń poza tym, aby zmienne były binarne. Jednak zdecydowana

liczba problemów obejmuje dodatkowe ograniczenia, które muszą być spełnione, aby znaleźć prawidłowe rozwiązanie. Wiele z tych problemów można skutecznie przeformułować do postaci QUBO, wprowadzając kary kwadratowe, reprezentujące te dodatkowe ograniczenia i dodając je do funkcji celu. Wprowadzone kary powinny być tak dobrane, aby wpływ pierwotnych ograniczeń na proces szukania prawidłowego rozwiązania można było osiągnąć poprzez naturalne działanie narzędzia, szukającego optymalnego rozwiązania. Oznacza to, że kary powinny być tak sformułowane, aby ich wartość była równa zeru dla prawidłowych rozwiązań oraz równa jakiejś niezerowej wartości dodatniej dla problemów minimalizacji, albo ujemnej dla problemów maksymalizacji. Funkcja celu, po dodaniu kary nazywana jest rozszerzoną funkcją celu i dla niej szuka się rozwiązania optymalnego, które będzie jednocześnie rozwiązaniem optymalnym pierwotnej funkcji celu, bo jeżeli warunki kary zostaną sprowadzone do zera, rozszerzona funkcja celu stanie się pierwotną funkcją. Należy pamiętać, żeby w wyrażeniu reprezentującym kary wszystkie zmienne były zmiennymi binarnymi oraz żeby współczynnik tego wyrażenia był na tyle dużą wartością, aby zapewniał, że wszystkie błędne rozwiązania zostaną odrzucone. W celu zilustrowania idei wprowadzania kary, rozważmy problem, przedstawiony w pracy [34], następującej postaci:

$$\begin{gathered} min\ y = f(x) \\ \textit{pod warunkiem, że:} \\ x_1 + x_2 \leq 1, \end{gathered}$$

gdzie x_1 oraz x_2 są zmiennymi binarnymi. Ograniczenie $x_1 + x_2 \leq 1$ pozwala na przypisanie jednej ze zmiennych wartości 1, albo obu zmiennym wartości 0. Wyklucza ono rozwiązanie, gdy obie zmienne mają wartość 1. Kwadratowa kara odpowiadająca powyższemu ograniczeniu to: Px_1x_2, gdzie P jest liczbą dodatnią. Po wprowadzeniu kary, otrzymujemy rozszerzoną funkcję celu następującej postaci:

$$min\ y = f(x) + Px_1x_2, \tag{21}$$

która ma to samo optymalne rozwiązanie, co pierwotny problem z ograniczeniem. Je-

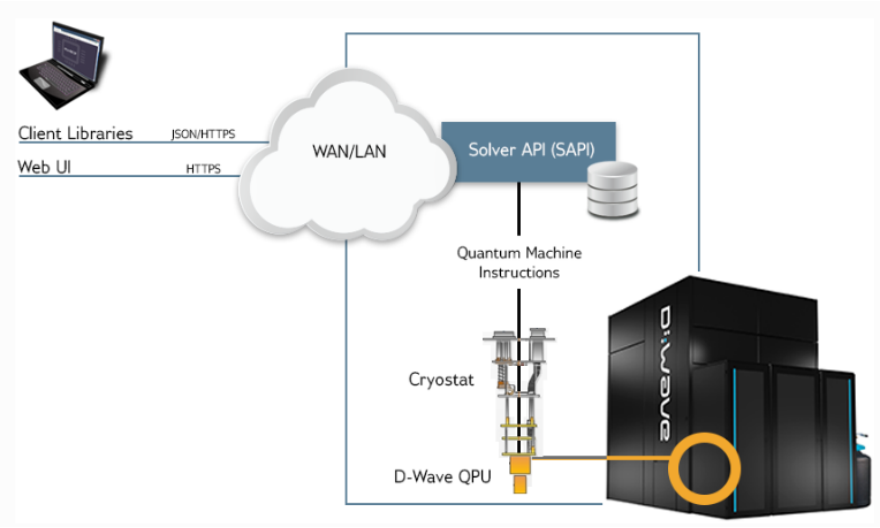
śli funkcja celu $f(x)$ jest liniowa lub kwadratowa, to problem (21) ma postać problemu w postaci QUBO. W rozpatrywanym przykładzie każde narzędzie, próbujące zminimalizować funkcję y, będzie pomijało rozwiązania, w których obie zmienne x_1 oraz x_2 są równe 1, ponieważ w tych przypadkach do wartości funkcji celu $f(x)$ zostanie dodana duża wartość dodatnia, równa P.

2.6 Komputer kwantowy D-Wave

Model komputera D-Wave został zaprojektowany zgodnie z adiabatycznym modelem obliczeń kwantowych. Nie są to jednak pełne procesory w obecnym znaczeniu tego słowa, ale raczej inteligentne akceleratory pamięci, zaprojektowane do rozwiązywania NP-trudnych problemów optymalizacyjnych, przy użyciu wyżarzania kwantowego. Platforma D-Wave składa się z dwóch głównych komponentów:

- układu sprzętowego, który rozwiązuje problemy zdefiniowane w postaci modelu Isinga, za pomocą fizycznej realizacji algorytmu wyżarzania kwantowego;
- konwencjonalnego serwera firmy Intel z interfejsem użytkownika, połączonego z układem sprzętowym za pomocą linii sterujących oraz podsystemów wejścia-/wyjścia.

Użytkownik, chcący skorzystać z komputera kwantowego D-Wave za pośrednictwem internetowego interfejsu użytkownika (UI) oraz narzędzi typu open source, komunikuje się z interfejsem SAPI (Solver API). Komponenty interfejsu SAPI odpowiedzialne są za interakcję z użytkownikiem, jego uwierzytelnienie oraz planowanie pracy. Z drugiej strony, interfejs SAPI łączy się z serwerami, które wysyłają zdefiniowane problemy do i zwracają wyniki z jednostek QPU (*ang. quantum processing unit*). Na rysunku 5 przedstawiono elementy środowiska programistycznego komputera kwantowego D-Wave. W celu wykorzystania komputera kwantowego D-Wave do rozwiązania danego problemu, należy przedstawić go w postaci problemu optymalizacyjnego. Matematyczną reprezentacją optymalizowanego problemu jest funkcja celu, której najmniejsza wartość reprezentuje najlepsze rozwiązanie oraz która wyraża energię

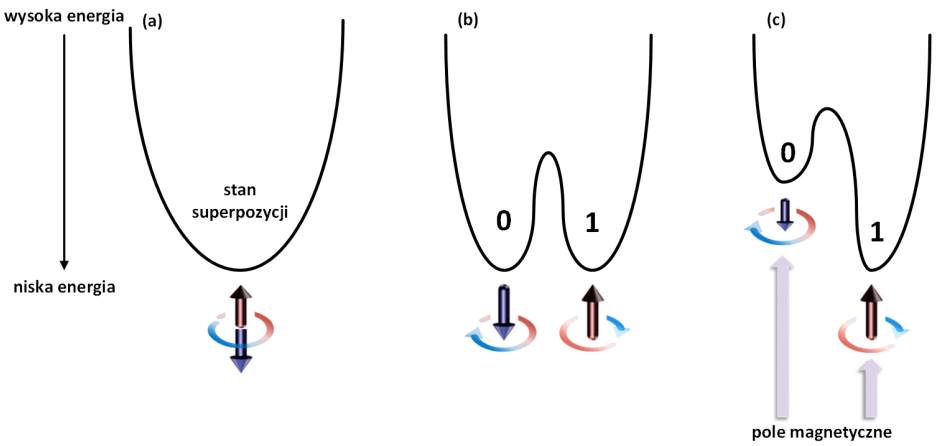


Rysunek 5: Elementy środowiska programistycznego komputera D-Wave. *Źródło: [1]*

systemu. Dla solwera QPU, energia jest funkcją zmiennych binarnych reprezentujących jego kubity. W niektórych przypadkach każdy stan o niskiej energii jest akceptowalnym rozwiązaniem pierwotnego problemu, jednak są problemy, dla których dopuszczalne jest tylko rozwiązanie optymalne, będące globalną minimalną energią w przestrzeni rozwiązań.

Aby dany problem wyrazić jako funkcję celu i przesłać go do komputera D-Wave w celu jego rozwiązania, należy funkcję celu przedstawić za pomocą modelu kwadratowego, akceptowalnego przez oprogramowanie komputera D-Wave. Dopuszczalne są następujące modele kwadratowe:

- Binarny Model Kwadratowy (*BQM, ang. Binary Quadratic Model*), który reprezentuje problemy bez ograniczeń, składające się ze zmiennych binarnych; model ten jest zwykle wykorzystywany w aplikacjach, które optymalizują decyzje przyjmujące rozwiązanie prawda/fałsz;
- Ograniczone Modele Kwadratowe (*CQM, ang. Constrained Quadratic Models*), które reprezentują problemy z ograniczeniami, składające się ze zmiennych całkowitych i/lub binarnych;
- Dyskretne Modele Kwadratowe (*DQM, ang. Discrete Quadratic Models*), które reprezentują problemy bez ograniczeń, składające się ze zmiennych dyskretnych.

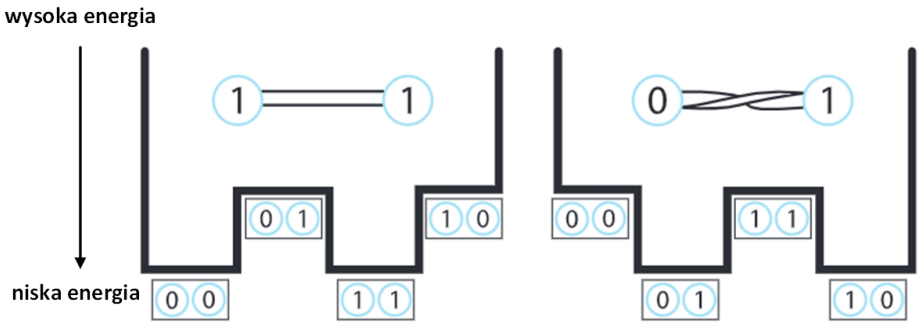


Rysunek 6: Diagram energii w procesie wyżarzania kwantowego. *Źródło: [1]*

2.6.1 Realizacja procesu wyżarzania kwantowego w komputerze D-Wave

Fizykę procesu wyżarzania kwantowego można przedstawić za pomocą diagramu energii (rysunek 6). Diagram zmienia się w czasie. Na początku kubit znajduje się w stanie superpozycji (a), co na diagramie przedstawione jest jako jedna dolina, z jednym minimum. Rozpoczyna się proces wyżarzania kwantowego, bariera zostaje podniesiona, a to zamienia diagram energii w tak zwany potencjał podwójnej studni (b), gdzie najniższy punkt lewej doliny odpowiada stanowi 0, a najniższy punkt prawej doliny odpowiada stanowi 1. Pod koniec wyżarzania kubit znajduje się w jednej z tych dolin. Prawdopodobieństwo, że kubit znajdzie się w obu studniach, jest takie samo i wynosi 50%. Można jednak kontrolować prawdopodobieństwo, z jakim kubit znajdzie się w stanie 0 albo 1, przykładając do kubitu zewnętrzne pole magnetyczne (c), które przechyla potencjał podwójnej studni, zwiększając prawdopodobieństwo, że kubit trafi do niższej studni. Programowalna wielkość, która kontroluje zewnętrzne pole magnetyczne, nazywana jest odchyleniem (*ang. bias*).

Prawdziwy potencjał stosowania kubitów związany jest z własnością splątania, które w komputerze D-Wave realizowane jest za pomocą urządzenia zwanego sprzęgaczem (*ang. coupler*). Sprzęgacz może sprawić, że dwa kubity znajdują się w tym samym stanie lub w stanach przeciwnych. Podobnie jak w przypadku odchylenia, wagi korelacji splątanych kubitów można zaprogramować, ustawiając siłę sprzęże-



Rysunek 7: Zastosowanie sprzęgacza do splątania kubitów. *Źródło: [1]*

nia. Kiedy dwa kubity są splątane, można je traktować jako pojedynczy obiekt z czterema możliwymi stanami. Na rysunku 7 przedstawiono zastosowanie sprzęgacza do splątania kubitów. W lewej części tego rysunku pokazano sytuację, gdy dwa kubity są splątane tak, aby przyjmowały ten sam stan, wtedy sprzęgacz zmniejsza energię, gdy kubity znajdują się w jednakowych stanach, w porównaniu do wartości energii, gdy kubity znajdują się w różnych stanach. W prawej części natomiast przedstawiono sytuację odwrotną, czyli gdy kubity są splątane. Wtedy energia dla stanów przeciwnych jest mniejsza niż dla stanów jednakowych. Względna energia każdego z czterech stanów zależy od odchyleń kubitów i sprzężenia między nimi. Podczas wyżarzania kwantowego stany kubitów są potencjalnie zdelokalizowane w diagramie, zanim ostatecznie pod koniec wyżarzania przyjmą jeden z czterech stanów.

Podsumowując, proces wyżarzania kwantowego zaczyna się od konfiguracji kubitów, z których każdy znajduje się w stanie superpozycji 0 i 1. Nie są one jeszcze splątane. Po rozpoczęciu wyżarzania kwantowego stosowane są sprzęgacze i odpowiednie odchylenia, w wyniku czego kubity zostają splątane. W tym momencie system jest w splątanym stanie wielu możliwych rozwiązań. Pod koniec wyżarzania każdy kubit jest w stanie klasycznym, który reprezentuje stan minimalnej lub bardzo niskiej energii rozwiązywanego problemu. Wszystko to dzieje się w komputerze kwantowym D-Wave w ciągu kilku mikrosekund.

Jak już wcześniej omówiono, klasyczny hamiltonian to matematyczny opis danego układu fizycznego, który za pomocą energii własnej zwraca energię stanu układu. Dla większości przypadków, znalezienie minimalnego stanu energetycznego jest pro-

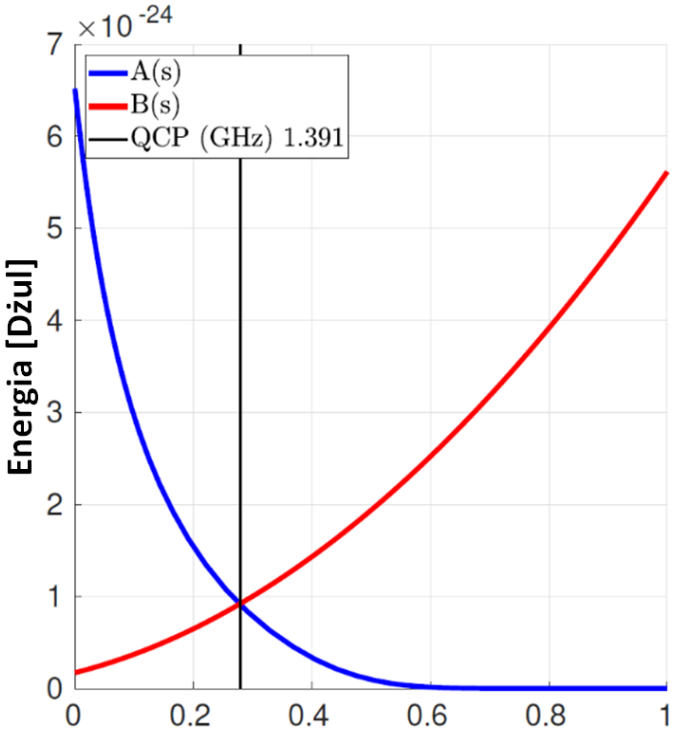
blemem NP-trudnym, którego klasyczne komputery nie mogą efektywnie rozwiązać. W przypadku układu kwantowego, hamiltonian jest funkcją, która odwzorowuje stany własne na energię. Dopiero gdy układ znajduje się w stanie własnym hamiltonianu, jego energia jest dobrze zdefiniowana i nazywana energią własną. Kiedy system znajduje się w jakimkolwiek innym stanie, jego energia jest niepewna. Zbiór stanów własnych o zdefiniowanych energiach własnych tworzy widmo własne. Dla komputera kwantowego D-Wave, hamiltonian układu ma następującą postać:

$$\mathcal{H}(s) = \underbrace{-\frac{A(s)}{2}\left(\sum_i \sigma_i^x\right)}_{\mathcal{H}_I} + \underbrace{\frac{B(s)}{2}\left(\sum_i h_i\sigma_i^z + \sum_{i<j} J_{i,j}\sigma_i^z\sigma_j^z\right)}_{\mathcal{H}_F}, \tag{22}$$

gdzie:

- σ_i^x, σ_i^z są macierzami Pauliego działającymi na i-tym kubicie,
- h_i to wartość energii odchylenia działającej na i-ty kubit,
- $J_{i,j}$ to wartość energii splątania pomiędzy i-tym oraz j-tym kubitem,
- $\mathcal{H}_I$ to hamiltonian początkowy (hamiltonian tunelowania $\mathcal{H}_D$) — którego stan o najniższej energii występuje wtedy, gdy wszystkie kubity są w stanie superpozycji 0 i 1;
- $\mathcal{H}_F$ to hamiltonian końcowy (hamiltonian problemu $\mathcal{H}_{Ising}$) — którego stan o najniższej energii końcowego jest rozwiązaniem problemu.

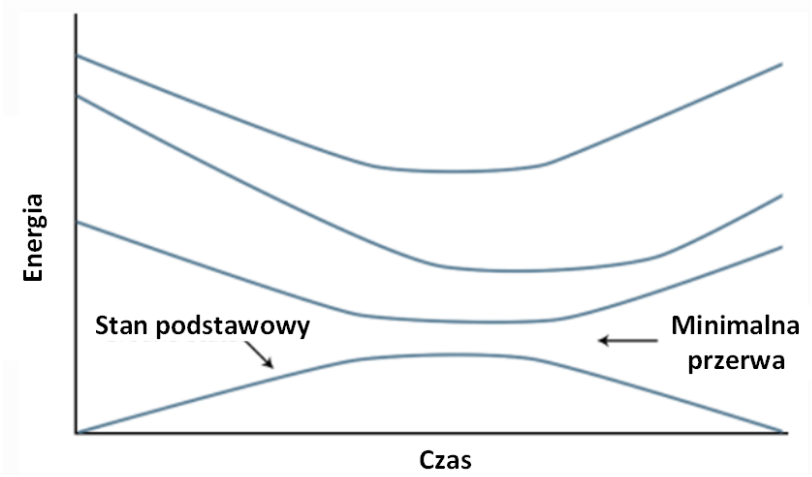
Współczynniki $A(s)$ oraz $B(s)$ realizują ścieżkę ewolucji adiabatycznej. Wartości tych współczynników, w zależności od parametru $s = \frac{t}{t_f}$, z zakresu od 0 do 1, gdzie t to dany moment procesu wyżarzania, a t_f to całkowity czas procesu wyżarzania, przedstawione zostały na rysunku 8. Krzywa $A(s)$ reprezentuje energię tunelowania, a krzywa $B(s)$ reprezentuje energię hamiltonianu problemu. Wartości obu funkcji wyrażone są w dżulach. Krytyczny punkt kwantowy (QCP) to punkt w wyżarzaniu, w którym amplitudy $A(s)$ i $B(s)$ są sobie równe. Przedstawione dane są reprezentatywne dla systemów D-Wave Advantage.



Rysunek 8: Wykres parametrów hamiltonianu układu podczas procesu wyżarzania. *Źródło: [1]*

Na początku, w momencie $t = 0$, układ znajduje się w stanie własnym hamiltonianu początkowego o najniższej energii. Wtedy wartość funkcji $A(0)$ jest dużo większa niż wartość funkcji $B(0)$: $A(0) \gg B(0)$, co prowadzi do kwantowego stanu podstawowego układu, w którym każdy spin znajduje się w zdelokalizowanej kombinacji swoich stanów klasycznych $s_i = \pm 1$. W miarę postępu procesu wyżarzania zwiększa się wpływ hamiltonianu problemu, który zawiera wartości odchyleń i wagi splątania, natomiast zmniejsza się wpływ hamiltonianu początkowego. Proces ten realizowany jest poprzez zmniejszanie wartości parametru A oraz zwiększanie parametru B, do momentu $t = t_f$. Pod koniec procesu wyżarzania, gdy $t = t_f$ oraz $A(1) \ll B(1)$, układ znajduje się w stanie własnym hamiltonianu problemu, którego energia własna jest wartością funkcji celu rozwiązywanego problemu, a macierze σ_i^z można zastąpić przez klasyczne zmienne spinowe $s_i = \pm 1$. W tym momencie układ opisywany jest przez klasyczny układ spinowy modelu Isinga (19), a stan kubitów reprezentuje rozwiązanie problemu.

Na rysunku 9 przedstawiono przykładowy wykres energii własnych w funkcji



Rysunek 9: Przykładowe widmo energii własnych. *Źródło: [1]*

czasu. Na dole pokazano stan podstawowy, a wszystkie stany wzbudzone znajdują się powyżej. W momencie rozpoczęcia procesu wyżarzania, układ znajduje się w stanie podstawowym, od którego, w znaczącej odległości, znajduje się pierwszy stan wzbudzony. W miarę postępu procesu wyżarzania, gdy coraz większy wpływ na hamiltonian układu ma hamiltonian problemu, stany wzbudzone mogą zbliżać się do stanu podstawowego. Wraz ze zbliżaniem się stanów wzbudzonych do stanu podstawowego, rośnie prawdopodobieństwo, że układ przeskoczy ze stanu podstawowego do jednego ze stanów wzbudzonych. Podczas wyżarzania występuje moment, w którym pierwszy stan wzbudzony zbliża się do stanu podstawowego, a następnie ponownie się oddala. Minimalna odległość między stanem podstawowym a pierwszym stanem wzbudzonym w dowolnym punkcie wyżarzania nazywana jest przerwą minimalną. Jednym z czynników, które powodują, że układ przeskoczy ze stanu podstawowego do stanu wzbudzonego, jest fluktuacja termiczna, która istnieje w dowolnym układzie fizycznym. Innym czynnikiem jest zbyt szybkie uruchamianie procesu wyżarzania. Idealnym modelem tego procesu jest, wcześniej już wspomniany, proces adiabatyczny, w którym nie występuje ingerencja z zewnętrznych źródeł energii, hamiltonian układu ewoluuje wystarczająco powoli, a podczas całego procesu układ znajduje się w stanie podstawowym. Ponieważ żadne obliczenia w świecie rzeczywistym nie mogą przebiegać w doskonałej izolacji, w przypadku niektórych problemów prawdopodobieństwo pozostania w stanie podstawowym może być niewielkie. Oczywiście, dla

każdego określonego problemu istnieje inny hamiltonian oraz inne odpowiadające mu widmo energii własnych. W kontekście procesu wyżarzania kwantowego, im mniejsza jest minimalna przerwa energetyczna, tym problem jest trudniejszy do rozwiązania.

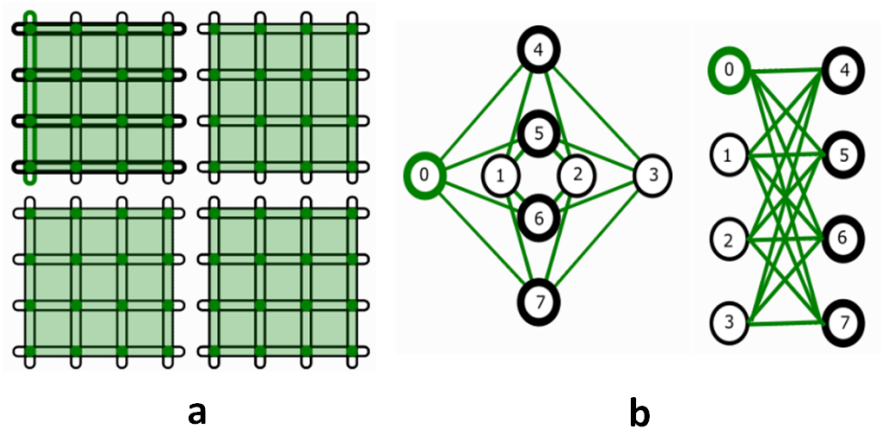
2.6.2 Topologia komputera D-Wave

Jednostka przetwarzania kwantowego (QPU) systemu D-Wave to sieć kubitów, połączonych za pomocą sprzęgaczy. Nie jest to jednak połączenie pełne. Do tej pory, opracowano trzy topologie, według których połączono kubity:

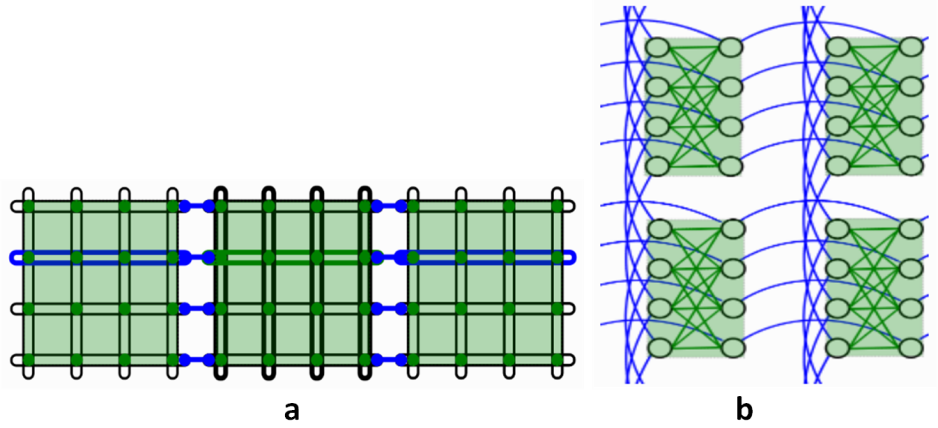
- Chimera – zastosowana w systemach D-Wave 2000Q i wcześniejszych generacjach jednostek QPU;
- Pegasus – zastosowana w systemach D-Wave Advantage,
- Zephyr – przeznaczona dla jednostek QPU nowej generacji, które w momencie pisania niniejszej rozprawy są dopiero opracowywane.

W przypadku jednostek QPU z topologią Chimera, kubity na grafie przedstawiane są jako poziome i pionowe pętle (rysunek 10), pomiędzy którymi występują sprzęgacze. W topologii Chimera wyróżnia się dwa rodzaje sprzęgaczy.

Sprzęgacze wewnętrzne łączą pary kubitów o przeciwnej orientacji (pętle prostopadłe do siebie). Na rysunku 10a sprzęgacze wewnętrzne oznaczono zielonymi kółkami. Przykładowo, lewy skrajny pionowy kubit, oznaczony pogrubioną linią koloru zielonego, jest wewnętrznie sprzęgnięty z czterema poziomymi kubitami, oznaczonymi czarną, pogrubioną linią. Komórką elementarną w topologii Chimera jest powtarzająca się struktura czterech poziomych kubitów, połączonych z czterema pionowymi kubitami, oznaczana jako $K_{4,4}$ oraz przedstawiana na grafie dwudzielnym, za pomocą krzyża lub kolumny (rysunek 10b). W obu sposobach przedstawiania komórki elementarnej Chimery znajdują się dwa zestawy czterech kubitów. Każdy kubit jest połączony ze wszystkimi kubitami z drugiego zestawu, ale z żadnym z własnego. Na przykład, na rysunku 10b, zielony kubit oznaczony jako 0 łączy się z pogrubionymi na czarno kubitami od 4 do 7.

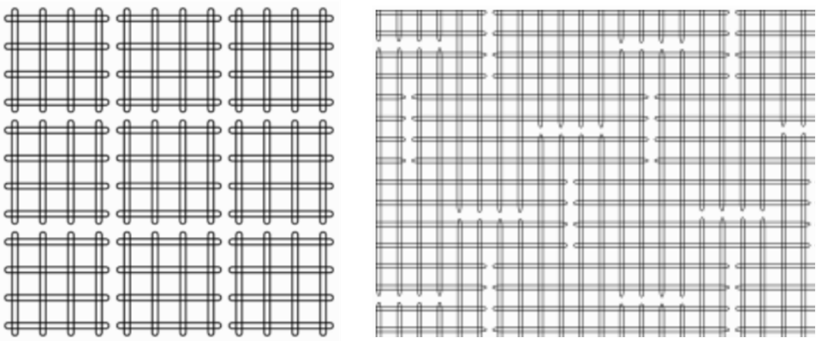


Rysunek 10: Komórki elementarne topologii Chimera z wewnętrznymi sprzęgaczami. *Źródło: [1]*

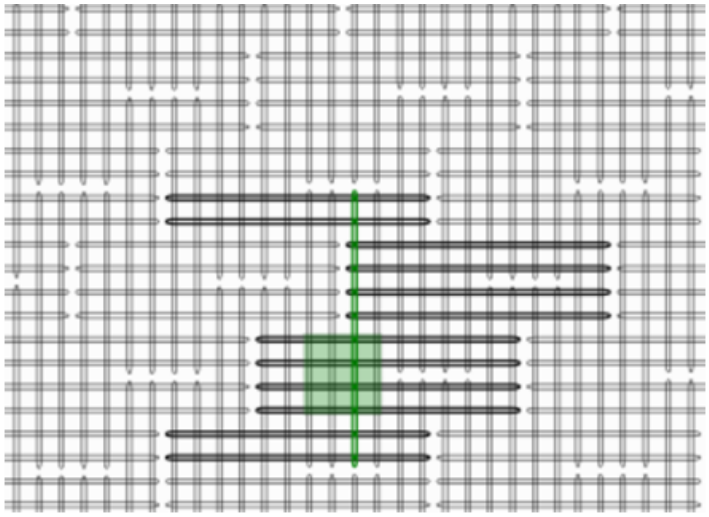


Rysunek 11: Komórki elementarne topologii Chimera z zewnętrznymi sprzęgaczami. *Źródło: [1]*

Sprzęgacze zewnętrzne, łączą pary współliniowych kubitów, tzn. pary kubitów równoległych w tym samym wierszu lub kolumnie. Na rysunku 11a sprzęgacze zewnętrzne pokazano jako połączone niebieskie kółka, które łączą kubity poziome z sąsiednimi kubitami poziomymi. Przykładowo, kubit poziomy oznaczony pogrubioną zieloną liniową, w środkowej komórce, sprzęgnięty został z dwoma kubitami poziomymi, sąsiednich komórek elementarnych, oznaczonymi pogrubioną niebieską linią. Oczywiście, ten sam kubit, jest również sprzężony z kubitami we własnej komórce elementarnej, oznaczonymi pogrubioną czarną linią, za pomocą sprzęgaczy wewnętrznych. Komórki elementarne $K_{4,4}$, utworzone przez sprzęgacze wewnętrzne połączone są sprzęgaczami zewnętrznymi, tworząc sieć – topologię Chimera. Na rysunku 11b

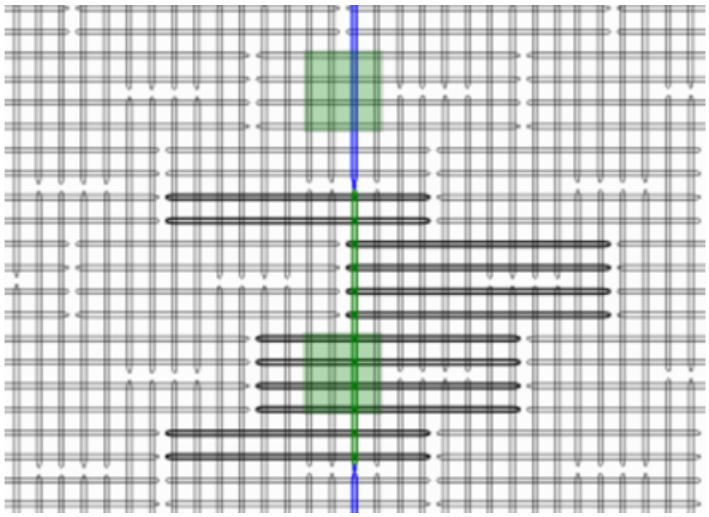


Rysunek 12: Wyrównanie kubitów w topologii Chimera (lewo) oraz w topologii Pegasus (prawo). *Źródło: [1]*



Rysunek 13: Kubity połączone wewnętrznymi sprzęgaczami w topologii Pegasus. *Źródło: [1]*

pokazano cztery komórki elementarne, połączone zewnętrznymi sprzęgaczami (linia niebieska), które tworzą część większego wykresu Chimery. W topologi Chimera długość nominalna połączeń kubitów wynosi 4, co oznacza, że każdy kubit jest połączony z 4 prostopadłymi kubitami, za pomocą wewnętrznych sprzęgaczy. Dodatkowo, połączenia kubitów mają stopień 6, co oznacza, że każdy kubit jest połączony z 6 różnymi kubitami. W przypadku jednostek QPU z topologią Pegasus, kubity zorientowane są poziomo lub pionowo, analogicznie do topologii Chimera, ale grupami przesunięte są względem siebie. Na rysunku 12 po lewej stronie przedstawiono wyrównanie kubitów w topologii Chimera, natomiast po prawej – w topologii Pegasus. W topologii Pegasus sprzęgacze dzielone są na wewnętrzne, zewnętrzne i nieparzyste. Sprzęgacze wewnętrzne łączą pary ortogonalnych kubitów. Każdy kubit jest połączony przez



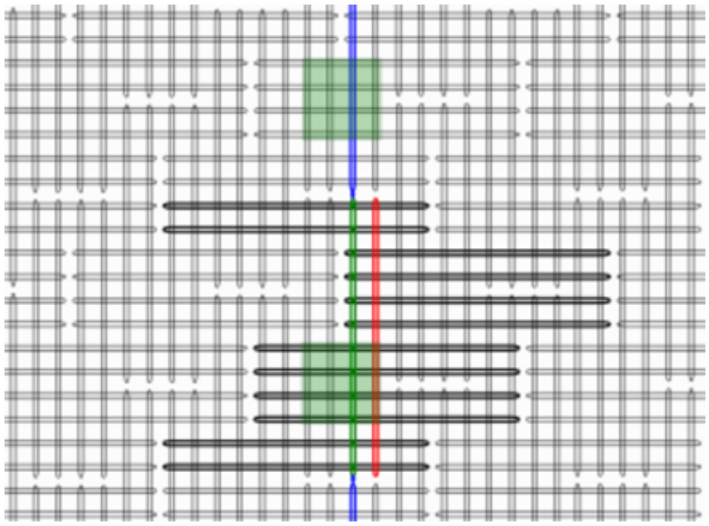
Rysunek 14: Kubity połączone zewnętrznymi sprzęgaczami w topologii Pegasus. *Źródło: [1]*

wewnętrzne sprzężenie z 12 innymi kubitami. Na rysunku 13 wyróżniono kubity połączone sprzęgaczem wewnętrznym. Pionowy kubit, oznaczony kolorem zielonym, sprzęgnięty jest z 12 kubitami poziomymi, oznaczonymi pogrubioną linią koloru czarnego. Zielony kwadrat reprezentuje strukturę komórki elementarnej Chimery $K_{4,4}$.

Sprzęgacze zewnętrzne łączą podobnie wyrównane sąsiednie kubity, pionowe z sąsiednimi kubitami pionowymi, a poziome z sąsiednimi kubitami poziomymi. Na rysunku 14 pionowy kubit, oznaczony kolorem zielonym, sprzęgnięty został z dwoma sąsiednimi kubitami pionowymi, oznaczonymi kolorem niebieskim. Jednocześnie na czarno oznaczone są kubity sprzęgnięte wewnętrznie.

Sprzęgacze nieparzyste łączą podobnie wyrównane pary kubitów. Przykładowo, na rysunku 15, zielony, pionowy kubit został sprzęgnięty z czerwonym, pionowym kubitem za pomocą sprzęgacza nieparzystego.

Rysunek 16 przedstawia dwa widoki sprzężenia kubitów, za pomocą trzech rodzajów sprzęgaczy, w topologii Pegasus. W górnej części rysunku, dwa kubity, oznaczone kolorem czerwonym oraz numerem 1, połączone są sprzęgaczem nieparzystym w parę. Para ta jest sprzęgnięta wewnętrznie z pionowymi parami kubitów o numerach od 3 do 8, przy czym każda para kubitów jest sprzęgnięta sprzęgaczem nieparzystym. Dodatkowo, czerwona para kubitów jest sprzęgnięta zewnętrznie z poziomymi parami kubitów o numerach 2 i 9. Dolna część rysunku 16 przedstawia te same połączenia ku-

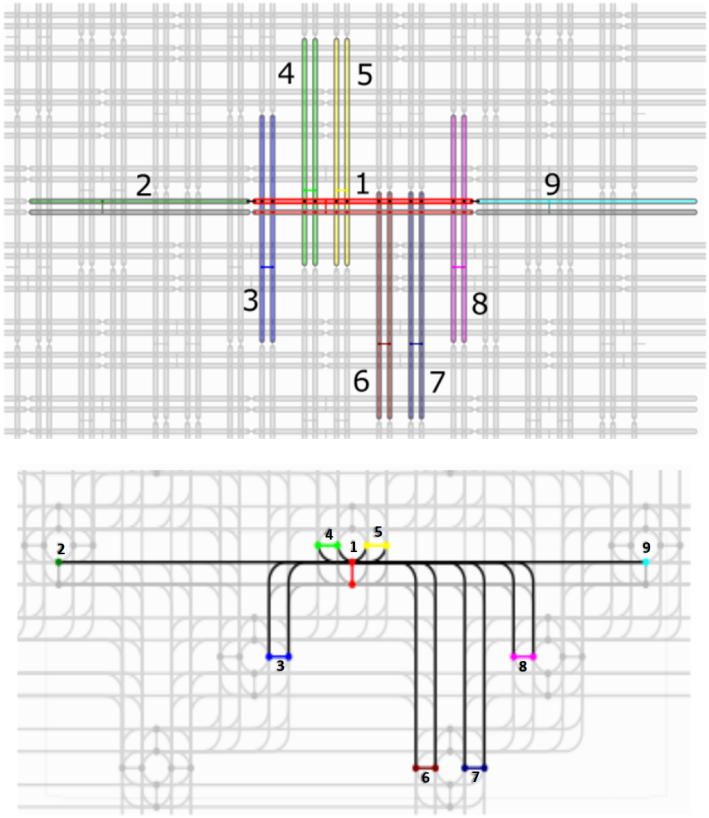


Rysunek 15: Para kubitów (zielony i czerwony) połączona nieparzystym sprzęgaczem w topologii Pegasus. *Źródło: [1]*

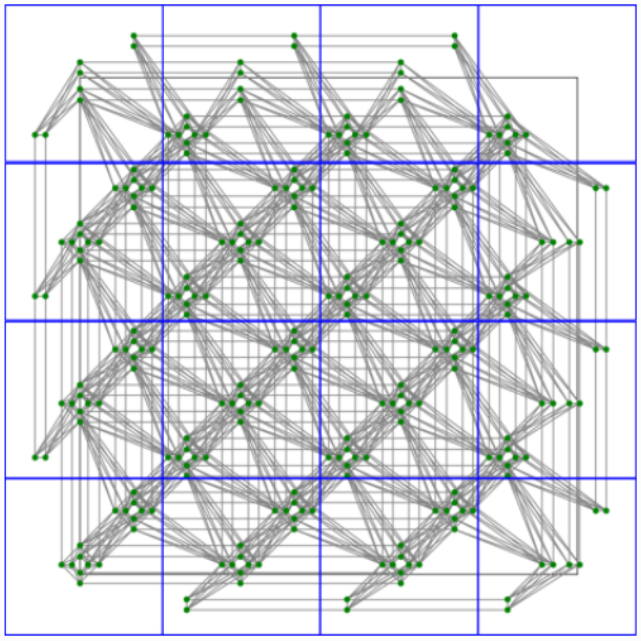
bitów w grafie dwudzielnym. Kubity przedstawione są jako kropki, a sprzęgacze jako linie. Kubit oznaczony na czerwono i numerem 1, połączony jest sprzęgaczem nieparzystym z czerwonym kubitem występującym bezpośrednio pod nim. Ta para kubitów jest wewnętrznie sprzężona z pionowymi kubitami o numerach od 3 do 8 oraz zewnętrznie sprzężona z poziomymi kubitami 2 i 9. Pegasus zawiera podgrafy K_4 i $K_{6,6}$, łączenia kubitów stopnia 15 – każdy kubit jest połączony z 15 różnymi kubitami oraz połączania kubitów mają nominalną długość równą 12 – każdy kubit jest połączony z 12 prostopadłymi kubitami, za pomocą sprzęgaczy wewnętrznych.

Komórka elementarna Pegasusa zawiera dwadzieścia cztery kubity, przy czym każdy kubit jest połączony z jednym podobnie wyrównanym kubitem w komórce i dwoma podobnie wyrównanymi kubitami w sąsiednich komórkach, co pokazano na rysunku 17, gdzie kubity oznaczono zielonymi kropkami, a sprzęrzenia pomiędzy nimi – szarymi liniami. Jednostki QPU systemu D-Wave Advantage to siatka 16×16 komórek elementarnych.

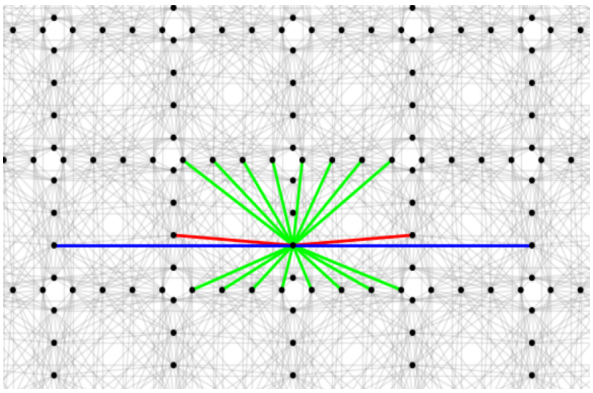
W przypadku topologii Zephyr, kubity są zorientowane pionowo lub poziomo, jak w przypadku topologii Chimera i Pegasus oraz są przesunięte i połączone trzema typami sprzęgaczy, jak w topologii Pegasus, ale w topologii Zephyr osiągana jest większa długość nominalna, wynosząca 16 oraz większy stopień, wynoszący 20. Na rysunku 18 przedstawiono fragment topologii Zephyr, gdzie za pomocą czarnych kropek



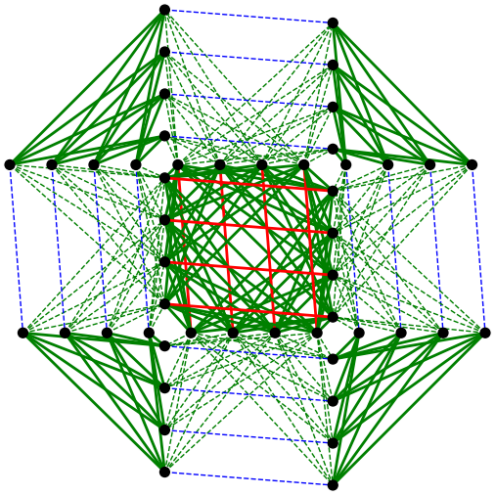
Rysunek 16: Kubity połączone trzema rodzajami sprzęgaczy w topologii Pegasus. *Źródło: [1]*



Rysunek 17: Komórki elementarne w topologii Pegasus w siatce 4×4. *Źródło: [1]*



Rysunek 18: Kubity połączone trzema rodzajami sprzęgaczy w topologii Zephyr. *Źródło: [1]*

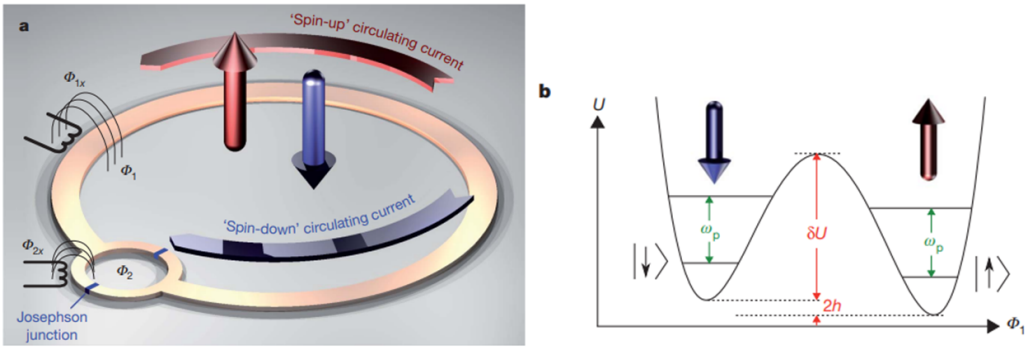


Rysunek 19: Komórki elementarne w topologii Zephyr. *Źródło: [1]*

oznaczono kubity. Pojedynczy kubit ma szesnaście wewnętrznych sprzęgaczy (zielone linie), łączących go z kubitami ortogonalnymi oraz dwa zewnętrzne (niebieskie linie) i dwa nieparzyste sprzęgacze (czerwone linie) łączące go z podobnie wyrównanymi kubitami. Na rysunku 19 przedstawiono siatkę komórek elementarnych w topologii Zephyr, gdzie kubity reprezentowane są jako czarne kropki, linie ciągłe reprezentują sprzęgacze należące do jednej komórki elementarnej, a linie przerywane reprezentują sprzęgacze należące do innych komórek elementarnych. Sprzęgacze wewnętrzne oznaczone są kolorem zielonym, zewnętrzne – niebieskim, a nieparzyste – czerwonym. W topologii Zephyr można wyróżnić podgrafy K_4 oraz $K_{8,8}$.

2.6.3 Implementacja procesu wyżarzania kwantowego w komputerze D-Wave

Aby wdrożyć procesor wykorzystujący wyżarzanie kwantowe do rozwiązywania problemów trudnych obliczeniowo, potrzebny jest programowalny system spinów kwantowych, w którym można kontrolować poszczególne spiny oraz ich sprzężenia, przeprowadzać proces wyżarzania kwantowego oraz na koniec procesu określać stan każdego spinu.



Rysunek 20: Nadprzewodzący kubit strumieniowy. *Źródło: [42]*

Jedna z możliwych implementacji sztucznego układu spinowego modelu Isinga obejmuje nadprzewodnikowe, strumieniowe kubity, zwane również kubitami prądu stałego. Fizycznie, realizowane są one jako pętle wielkości mikrometra, wykonane z nadprzewodzącego metalu oraz przerwane przez szereg złącz Josephsona. Przykładem takiego kubitu jest nadprzewodzący kubit strumieniowy rf-SQUID (*ang. superconducting quantum interference device*). Jednostki QPU, będące głównym elementem systemu D-Wave, zbudowane są z sieci połączonych kubitów rf-SQUID, działających w temperaturze około 12 mK. Kubitami strumieniowymi można manipulować, stosując strumień magnetyczny za pośrednictwem prądów wzdłuż indukcyjnie sprzężonych linii sterujących. Można to osiągnąć za pomocą analogowej linii sterującej, po jednej dla danego urządzenia, sterowanej przez źródła prądu w temperaturze pokojowej i kierowanej, poprzez odpowiednie filtrowanie, do docelowego urządzenia na chipie. Na rysunku 20 przedstawiono uproszczony schemat nadprzewodzącego kubitu strumieniowego. Po lewej stronie (20a) przedstawiono dwie pętle nadprzewodzące, z których każda poddana jest działaniu zewnętrznego strumienia od-

chylenia, odpowiednio Φ_{1x} lub Φ_{2x}. Dynamikę urządzenia można modelować jako kwantowo – mechaniczny potencjał podwójnej studni względem strumienia Φ_1 (rysunek 20b). Wysokość bariery, δU, kontrolowana jest przez strumień Φ_{2x}, podczas gdy różnica energii pomiędzy dwoma minimami, oznaczona jako $2h$, kontrolowana jest przez strumień Φ_{1x}. Dwa najniższe stany energetyczne układu, odpowiadające prądowi krążącemu w kierunku zgodnym lub przeciwnym do ruchu wskazówek zegara w pętli 1, oznaczone są jako $|\uparrow\rangle$ i $|\downarrow\rangle$, ze strumieniem zlokalizowanym odpowiednio w lewej lub prawej studni. Jeśli weźmiemy pod uwagę tylko te dwa stany, dynamika kubitu rf-SQUID jest równoważna z dynamiką spinu w modelu Isinga. Kubity rf-SQUID są sprzęgane ze sobą za pomocą programowalnych elementów sprzęgających, które zapewniają energię sprzężenia, którą można przestrajać w sposób ciągły, pomiędzy sprzężeniem ferromagnetycznym, a antyferromagnetycznym. Każdy kubit strumieniowy rf-SQUID ma porty indukcyjne, połączone z różnymi przetwornikami cyfrowo-analogowymi oraz złączami Josephsona (CCJJ). Fizyczna realizacja hamiltonianu, zwracającego energię takiej sieci sprzężonych kubitów rf-SQUID, to w przybliżeniu:

$$\begin{aligned} H = &-\frac{1}{2}\sum_i \left[\Delta_q(\Phi_{CCJJ}(s))\sigma_i^x - 2h_i \left| I_p(\Phi_{CCJJ}(s))\right| \Phi_i^x(s)\sigma_i^z\right] + \\ &+ \sum_{i>j} J_{i,j} M_{AFM} I_p(\Phi_{CCJJ}(s))^2 \sigma_i^z \sigma_j^z, \end{aligned} \tag{23}$$

gdzie:

- Δ_q jest różnicą energii pomiędzy dwoma stanami własnymi kubitu rf-SQUID bez przyłożonego zewnętrznego pola magnetycznego (punkt degeneracji), gdzie stany własne są równe $\frac{|0\rangle \pm |1\rangle}{\sqrt{2}}$;
- I_p reprezentuje wielkość prądu płynącego w pętli rf-SQUID;
- M_{AFM} to maksymalna indukcyjność wzajemna, wygenerowana przez sprzęgacze pomiędzy kubitami;
- $\Phi_i^x(s)$ to strumień zewnętrznego pola magnetycznego, oddziałujący na kubity;

- $\Phi_{CCJJ}(s)$ to strumień zewnętrznego pola magnetycznego, oddziałujący na wszystkie kubity ze złączem Josephsona, w celu zmiany kształtu energii potencjalnej kubitu rf-SQUID.

W celu przekształcenia równania (23) do równania (22) zakłada się, że $\Phi_i^x(s) = M_{AFM}\left|I_p(s)\right|$, co oznacza, że strumień $\Phi_i^x(s)$ zmienia się tak, aby utrzymać na stałym poziomie względny stosunek energii pomiędzy składnikami h i J. W szczególności, strumień przyłożony do kubitu w celu zaimplementowania stałej wartości h, wzrasta wraz z postępem procesu wyżarzania. Teraz odwzorowanie do hamiltonianu modelu Isinga zachodzi zgodnie z zależnościami:

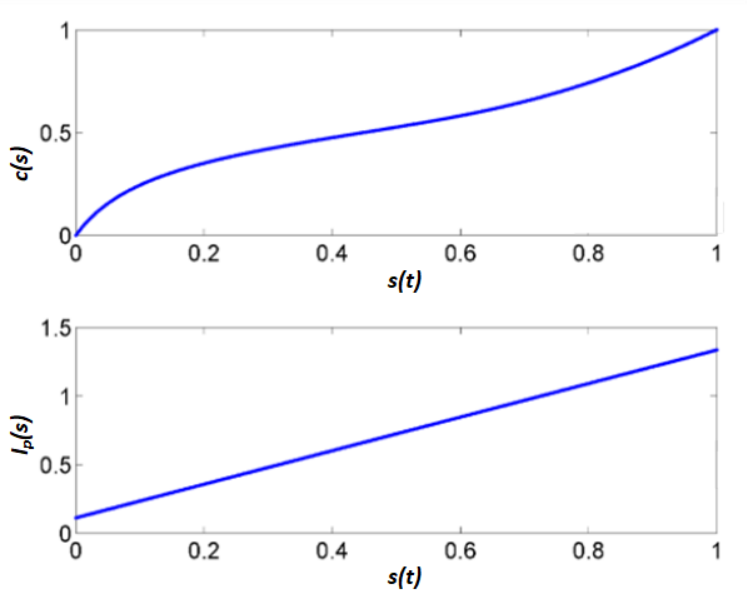
$$A(s) = \Delta_q\left(\Phi_{CCJJ}(s)\right),$$

$$B(s) = 2M_{AFM}\left|I_p\left(\Phi_{CCJJ}(s)\right)\right|^2.$$

Zależność pomiędzy $\Delta_q(\Phi_{CCJJ})$ oraz $I_p(\Phi_{CCJJ})$ jest ustalona przez fizyczne parametry kubitu rf-SQUID. Zmianami funkcji $A(s)$ i $B(s)$ podczas procesu wyżarzania kwantowego steruje pojedyncze, globalne oraz zależne od czasu odchylenie $c(s)$. Dla dowolnej wartości funkcji $s(t)$, stosunek $\frac{A(s)}{B(s)}$ jest stały. Typowe wartości funkcji $A(s)$ i $B(s)$ pokazano już wcześniej na rysunku 8, a odchylenie $c(s)$ można przedstawić za pomocą następującej zależności:

$$c(s) = \frac{\Phi_{CCJJ}(s) - \Phi_{CCJJ}^{\text{początkowy}}(s)}{\Phi_{CCJJ}^{\text{końcowy}}(s) - \Phi_{CCJJ}^{\text{początkowy}}(s)}, \tag{24}$$

gdzie $\Phi_{CCJJ}^{\text{początkowy}}$ i $\Phi_{CCJJ}^{\text{końcowy}}$ są wartościami Φ_{CCJJ}, odpowiednio dla $s = 0$ oraz $s = 1$, dla których $c(0) = 0$ oraz $c(1) = 1$. Sygnał $c(s)$ jest dostarczany przez zewnętrzne źródło prądu, w temperaturze pokojowej. Sygnał odchylenia $c(s)$ w funkcji s nie jest liniowy, ale wybrany tak, aby natężenie prądu płynącego w pętli $I_p(s)$ rosło liniowo w czasie, co pokazano na rysunku 21. Współczynnik hamiltonianu problemu $B(s) = 2M_{AFM}I_p(s)^2$ rośnie kwadratowo w czasie. Współczynnik hamiltonianu początkowego $A(s)$ opisuje dynamikę kubitów w czasie. W miarę postępu procesu wyżarzania, wartości funkcji $A(s)$ maleją, co oznacza, że dynamika układu spinowego



Rysunek 21: Wykres sygnału odchylenia c oraz I_p w funkcji $s(t)$. *Źródło: [1]*

modelu Isinga staje się wolniejsza i sieć zostaje zamrożona, tzn. że stan spinu nie zmienia się znacząco w miarę ewolucji hamiltonianu układu.

2.6.4 Rozwiązywanie problemów za pomocą komputera D-Wave

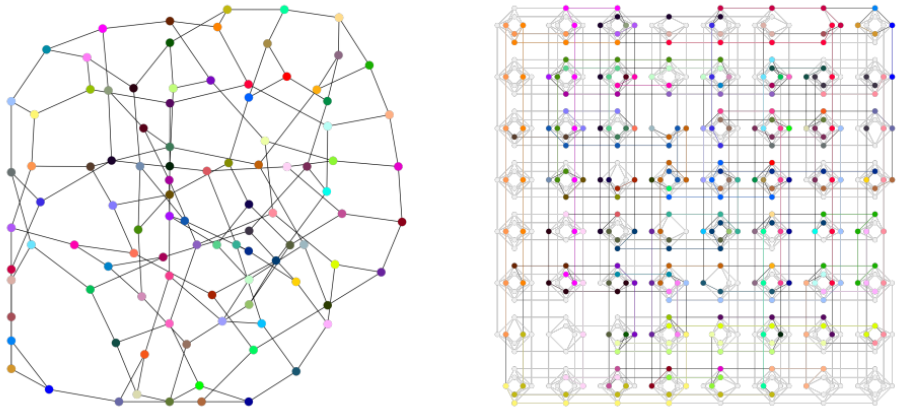
Kroki procesu rozwiązywania problemów optymalizacyjnych za pomocą komputera D-Wave są następujące.

1. Sformułowanie problemu. Dany problem należy przedstawić w postaci problemu optymalizacji kombinatorycznej, definiując cel oraz jego ograniczenia. Istnieją różne sposoby modelowania problemów, a wybór modelu może wpłynąć na wydajność rozwiązania. Należy przede wszystkim zdecydować, za pomocą jakiego rodzaju zmiennych najlepiej sformułować problem oraz jakiego stopnia będą uzyskane wyrażenia matematyczne definiujące problem.

2. Przekształcenie problemu do postaci akceptowalnej przez komputer D-Wave. Sformułowany problem optymalizacyjny należy odwzorować do problemu w postaci modelu Isinga lub problemu w postaci QUBO, należącego do kategorii modeli BQM, bądź do problemu z kategorii DQM lub CQM. Dla komputera kwantowego D-Wave, natywną instancją problemu, jest problem zdefiniowany w mo-

delu Isinga, zgodnie z równaniem (16). Problem w takiej postaci jest bezpośrednio implementowany w układzie sprzętowym komputera D-Wave. Oznacza to, że wagi h_i oraz J_{ij} odpowiadają sygnałom elektronicznym, realizującymi odchylenia. Jeśli jednak zdefiniowany problem nie jest w postaci natywnej, to system D-Wave najpierw transformuje go do modelu Isinga. Tak więc, instancja dowolnego problemu P może być odwzorowana na instancję M modelu Isinga w taki sposób, że optymalne rozwiązanie instancji M będzie również optymalnym rozwiązaniem instancji P, z nie więcej niż wielomianowym nakładem pracy. Dodatkowo, ze względu na ograniczone zasoby komputera D-Wave, transformacja powinna rozszerzać problem P w niewielkim stopniu. Podczas przekształcenia należy wziąć pod uwagę następujące aspekty:

- otrzymany problem po przekształceniu musi być sformułowany tak, aby był kompatybilny z ograniczeniami fizycznego systemu komputera D-Wave i jak najbardziej odporny na jego błędy kontrolne;
- ponieważ problem można przeformułować na różne sposoby, to niektóre z otrzymanych postaci mogą być efektywniejsze niż inne, jednak jeśli otrzymana postać problemu nie zapewnia prawidłowych rozwiązań (z odpowiednio skonfigurowanym solwerem QPU), należy rozważyć sformułowanie problemu w innej postaci; przykładowo przekształcenie, które zwiększa liczbę zmiennych, może być bardziej efektywne podczas osadzania w grafie sprzętowym;
- należy również zwrócić uwagę na skalę odchyleń, czyli różnicę między najmniejszymi i największymi odchyleniami w modelu BQM, co może wpływać na wydajność procesu rozwiązywania; współczynniki liniowe otrzymanego problemu w modelu BQM przekładają się na odchylenia kubitów, a współczynniki kwadratowe na siły sprzężenia; przykładowo, niech najmniejsze odchylenie wynosi $h_1 = -0,032$, a największe $h_2 = 3405$, podczas skalowania odchyleń w dół, do zakresu obsługiwanego przez QPU, małe odchylenie jest w rzeczywistości zerowe.



Rysunek 22: Osadzenie grafu modelu Isinga do grafu układu sprzętowego w topologii Chimera. *Źródło: [49]*

3. Dekompozycja. Proces polegający na rozłożeniu dużego problemu na części, które wymagają liczby kubitów nie większej, niż dostępnych w QPU.

4. Osadzenie w jednostce QPU. Rozwiązanie problemu w systemie D-Wave wymaga odwzorowania, zwanego osadzeniem, grafu modelu Isinga do grafu topologii danej jednostki QPU. Niech $G = (V_g, E_g)$ oznacza strukturę połączeń modelu Isinga, gdzie wierzchołki zbioru V_g odpowiadają zmiennym s_i problemu, a krawędzie zbioru $E_g = (v_i, v_j)$ odpowiadają wagom J_{ij}, o ile istnieją ($J_{ij} \neq 0$). Ponadto, niech $H = (V_h, E_h)$ reprezentuje graf układu sprzętowego danej topologii. Problemem jest znalezienie jak najmniejszego osadzenia grafu G w grafie H, co pokazano na rysunku 22, dla topologii Chimera. Dowolny graf H' jest grafem mniejszym, skonstruowanym z grafu H, poprzez zwinięcie dwóch sąsiednich wierzchołków w jeden lub poprzez usunięcie krawędzi.

   Topologia QPU nie jest w pełni połączona, dlatego większe problemy często wymagają utworzenia łańcuchów, co oznacza, że jedna zmienna logiczna jest reprezentowana przez pewną liczbę fizycznych kubitów, połączonych za pomocą łańcucha. Aby łańcuch kubitów reprezentował zmienną logiczną, wszystkie jego kubity składowe muszą zwracać tę samą wartość dla danego odczytu. Osiąga się to poprzez ustawienie silnego sprzężenia krawędzi łączących te kubity. Aby kubity w łańcuchu mogły zwracać identyczne wartości, siła sprzężenia ich krawę-

dzi musi być większa, w porównaniu do sprzężenia z innymi kubitami. Ustawienie zbyt małej siły łańcucha powoduje jego zerwanie, a w konsekwencji otrzymanie błędnych rozwiązań. Z drugiej strony, ustawienie zbyt dużej siły łańcucha zniekształca problem. Dodatkowo wszystkie łańcuchy w danym problemie powinny być krótkie oraz o zbliżonej długości.

5. Konfiguracja solwera QPU. Przygotowując problem, w celu przesłania go do systemu D-Wave, należy wziąć pod uwagę analogową naturę komputerów kwantowych. W celu zwiększenia prawdopodobieństwa znalezienia dobrych rozwiązań, można zdefiniować następujące parametry:

   - liczba cykli odczytu i czas wyżarzania – zwiększenie czasu wyżarzania oraz liczby odczytów zwiększa prawdopodobieństwo poprawnego rozwiązania problemu, jednak zwiększenie liczby odczytów powyżej pewnej wartości, zależnej od rozwiązywanego problemu, nie poprawia zwracanych wyników; optymalna kombinacja czasu wyżarzania i liczby odczytów w celu uzyskania najlepszego rozwiązania w ustalonym reżimie czasowym jest zależna od problemu i trzeba ją dobierać eksperymentalnie;

   - transformacja z odwróceniem spinów – sprzężenie $J_{i,j}$ dodaje niewielkie odchylenie do kubitów i oraz j powodowane wyciekiem, co ma znaczenie w przypadku kubitów połączonych łańcuchem, ponieważ dodatkowe kubity są do pewnego stopnia odchylone w jednym lub drugim kierunku z powodu niedoskonałości jednostki QPU; zastosowanie transformacji z odwróceniem spinów może poprawić wyniki, zmniejszając wpływ niezamierzonych błędów systematycznych; transformacja ta nie zmienia problemu w modelu Isinga, ale sprowadza się do reinterpretacji spinu w górę jako spinu w dół i na odwrót, dla pewnej liczby spinów; zmiana zbyt małej liczby spinów pozostawia większość błędów bez zmian, z drugiej strony, zmiana zbyt wielu spinów oznacza, że większość sprzęgaczy łączy transformowane spiny, a więc wartości $J_{i,j}$ nie zmieniają znaku, co powoduje, że niektóre błędy systematyczne związane ze sprzęgaczami pozostają nieusunięte; ta

transformacja zwiększa całkowity czas rozwiązywania problemu;

- postprocessing – dostępne narzędzia postprocessingu zapewniają pewną poprawę otrzymanych rozwiązań, na przykład naprawienie przerwanych łańcuchów lub zastosowanie zaimplementowanego algorytmu wyszukiwania zachłannego dla binarnych modeli kwadratowych;

- niedokładność odchylenia – problemy modelu Isinga z parametrami (h_i, $J_{i,j}$) o wysokiej precyzji stanowią wyzwanie dla komputerów kwantowych ze względu na skończoną precyzję, dostępną dla tych parametrów; problem może posiadać najniższe stany energetyczne, które są wrażliwe na małe zmiany wartości h lub J, a poprawność rozwiązania zależy od niewielkich różnic, w obszarach niskoenergetycznych przestrzeni rozwiązań; precyzję można zwiększyć podczas osadzania, kosztem dodatkowych kubitów, np.: zmienną s_i, z największą wartością współczynnika h_i, można przedstawić za pomocą łańcucha n kubitów, a w celu zwiększenia precyzji, wagę h_i należy podzielić przez n; precyzję można również zwiększyć przez uproszczenie problemu, np. jeśli podczas przetwarzań wstępnych w czasie wielomianowym znaleziono wartości zmiennych niepustego podzbioru, które zawsze przyjmują tę samą wartość w stanie podstawowym, to można wyeliminować takie zmienne z problemu;

- pauza i hartowanie – w przypadku niektórych problemów korzystne jest wprowadzenie pauzy lub hartowania w pewnym momencie procesu wyżarzania; prawdopodobieństwo uzyskania rozwiązań ze stanu podstawowego zależy od tego, kiedy w procesie wyżarzania następuje hartowanie, przy czym późniejsze wywołanie hartowania zwiększa prawdopodobieństwo uzyskania rozwiązań ze stanu podstawowego;

- przesunięcie procesu wyżarzania w czasie – może poprawić wyniki w przypadku problemów o nieregularnej dynamice lub problemów o różnej długości łańcuchów; dłuższe łańcuchy mogą zamarznąć wcześniej niż krótsze, co oznacza, że przed zakończeniem procesu wyżarzania niektóre zmienne

zachowują się jak ustalone stałe, podczas gdy inne zmienne pozostają nieustalone, jeśli jednak w krótszych łańcuchach przyspieszy się proces wyżarzania kubitów, to zamarzną one wcześniej, synchronizując trajektorię wyżarzania krótszych łańcuchów z trajektorią dłuższych.

6. Przeprowadzenie obliczeń. Kiedy komputer kwantowy D-Wave rozwiązuje problem, wykorzystuje zjawiska kwantowe (superpozycję i tunelowanie), w celu jednoczesnego sprawdzenia możliwych rozwiązań i znalezienia zestawu najlepszych. Proces ten składa się z następujących etapów:

   - programowanie/inicjalizacja: wagi h_i, J_{ij} lokowane są w kubitach, a kubity rozmieszczane są w stanach superpozycji zgodnie z hamiltonianem $\mathcal{H}_I$, który skalowany jest przez współczynnik $A(0)$; proces programowania podnosi temperaturę chipu, więc obejmuje również czas oczekiwania na jego schłodzenie;
   - wyżarzanie: zachodzi ewolucja układu, w której siły działające na kubity zmieniają się zgodnie ze zmianami funkcji $A(s)$ i $B(s)$, przedstawionymi na rysunku 8;
   - odczyt: proces wyżarzania zakończył się, więc kubity mają klasyczne stany spinów, zgodnie z Hamiltonianem $\mathcal{H}_{Ising}$, który jest skalowany przez $B(1)$; następuje odczyt wartości kubitów, w celu uzyskania rozwiązania problemu; ten etap obejmuje również krótki czas oczekiwania na schłodzenie układu, w celu ponownego próbkowania;
   - próbkowanie: ponieważ układ działa w systemie otwartym, zawsze istnieje dodatnie (czasem znaczące) prawdopodobieństwo, że obliczenia nie zostaną zakończone w stanie podstawowym, dlatego wymagane jest wielokrotne powtórzenie etapów wyżarzania i odczytu.

Większość rzeczywistych problemów nie jest łatwa do rozwiązania. Sformułowanie problemu może być trudne, a znalezienie optymalnego rozwiązania problemu może wymagać wielokrotnych zmian w konfiguracji solwera. W celu zobrazowania wyżej

opisanego procesu rozwiązywania problemu za pomocą komputera kwantowego D-Wave, przedstawiony zostanie prosty przykład, udostępniony w materiałach opisujących system D-Wave.

Dane jest następujące równanie: $25x_1 + 20x_2 + 30x_3 = 50$, gdzie zmienne x_1, x_2 oraz x_3 są zmiennymi binarnymi. Należy znaleźć rozwiązanie tego równania.

Zgodnie z pierwszym krokiem, formułujemy nasz problem w postaci problemu optymalizacyjnego. Problem rozwiązania równania można przekształcić do problemu optymalizacji, przenosząc wszystkie argumenty i stałe na jedną stronę równości i podnosząc ją do kwadratu, otrzymując: $min(50 - 25x_1 - 20x_2 - 30x_3)^2 = min(2500 - 1875x_1 - 1600x_2 - 2100x_3 + 1000x_1x_2 + 1500x_1x_3 + 1200x_2x_3)$. Otrzymane wyrażenie jest spełnione dla takich wartości zmiennych x_1, x_2 i x_3, dla których wartość wyrażenia jest równa zero.

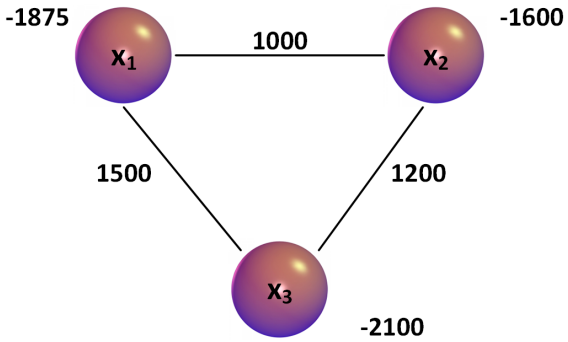
Następnie przekształcamy nasz problem do postaci akceptowalnej przez komputer kwantowy D-Wave. Ponieważ w problemie występują zmienne binarne, to najlepszym modelem dla tego problemu będzie problem w postaci QUBO. Ponieważ w problemie QUBO nie uwzględnia się stałej, to funkcja celu rozwiązywanego problemu w postaci QUBO, zgodnie z równaniem (20), jest następująca:

$$\begin{aligned} f_{qubo}(x_1, x_2, x_3) &= \sum_i \mathbf{Q}_{i,i} x_i + \sum_{i<j} \mathbf{Q}_{i,j} x_i x_j = \\ &= -1875x_1 - 1600x_2 - 2100x_3 + 1000x_1x_2 + 1500x_1x_3 + 1200x_2x_3, \end{aligned} \tag{25}$$

gdzie:

$$\mathbf{Q} = \begin{bmatrix} -1875 & 1000 & 1500 \\ 0 & -1600 & 1200 \\ 0 & 0 & -2100 \end{bmatrix} \tag{26}$$

Globalne minimum funkcji $f_{qubo}(x_1, x_2, x_3)$ odczytane zostanie ze stanu podstawowego, którego wartość energii będzie wynosić -2500. Narysujmy graf dla otrzymanej postaci QUBO naszego problemu. Wierzchołki tego grafu odpowiadają zmiennym binarnym, których wagi są równe odpowiednim współczynnikom liniowym. W funkcji celu występują wszystkie możliwe jednomiany stopnia dwa, a więc każda para wierzchołków zostanie połączona krawędzią, której wagi odpowiadają odpowiednim

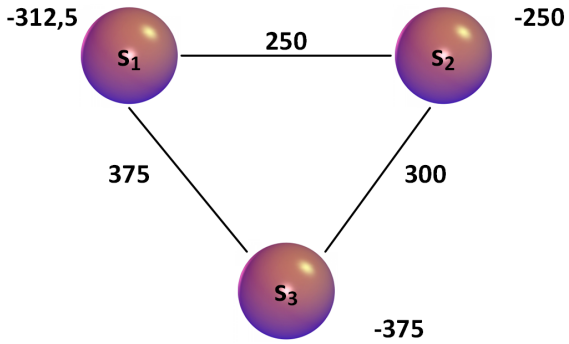


Rysunek 23: Graf przykładowego problemu przedstawionego w postaci QUBO. *Źródło: na podstawie [1]*

współczynnikom kwadratowym. Otrzymany graf przedstawiono na rysunku 23.

Krok trzeci – dekompozycja, zostanie pominięty. Podczas badań przeprowadzonych w ramach niniejszej rozprawy, krok ten został pozostawiony do wykonania przez oprogramowanie systemu D-Wave.

W kroku czwartym, problem w postaci współczynników macierzy (26) przesyłany jest do systemu D-Wave i uruchamiany na jednostce QPU. Po przesłaniu problemu następuje osadzenie grafu problemu 23 w chipie QPU. Ponieważ nasz problem przedstawiono w postaci QUBO, gdzie zmienne x_i przyjmują wartości $\{0,1\}$, to oprogramowanie systemu D-Wave musi przekształcić go, zgodnie z zależnością $x_i = \frac{s_i+1}{2}$, do modelu Isinga, w którym zmienne s_i przyjmują wartości $\{-1,+1\}$. Po przekształceniu otrzymujemy funkcję celu problemu w modelu Isinga, postaci: $-312,5s_1 - 250s_2 - 375s_3 + 250s_1s_2 + 375s_1s_3 + 300s_2s_3$ oraz stałą równą $-1862,5$, która nie jest uwzględniana. Również problem w postaci modelu Isinga można przedstawić za pomocą grafu, który dla rozwiązywanego problemu zaprezentowano na rysunku 24. Następnie problem przedstawiony w modelu Isinga osadzany jest w fizycznym chipie jednostki QPU, co oznacza, że zmiennej logicznej s_i przypisywany jest jej indywidualny kubit, który ją reprezentuje. Jednak, jak to wcześniej opisano, czasem zmienna logiczna reprezentowana jest przez łańcuch kubitów. Wykorzystując interfejs użytkownika D-Wave Leap, można zobaczyć, jak rozwiązywany problem został osadzony w jednostce QPU danego solwera. Na rysunku 25 przedstawiono osadzenie rozwiązywanego problemu w chipie jednostki D-Wave 2000Q, z topologią Chimera, gdzie pro-



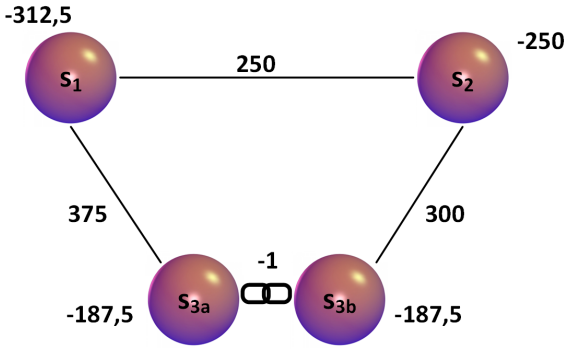
Rysunek 24: Graf przykładowego problemu przedstawionego w modelu Isinga. *Źródło: na podstawie [1]*

Rysunek 25: Osadzenie grafu modelu Isinga w chipie jednostki D-Wave 2000Q. *Źródło: [1]*

blem z trzema zmiennymi, zobrazowany w lewej części rysunku, został osadzony na czterech kubitach, co przedstawia prawa część tego rysunku. Dla porównania, ten sam problem osadzono w chipie jednostki D-Wave Advantage, z topologią Pegasus, gdzie system wykorzystał trzy kubity, co przedstawiono na rysunku 26. W dalszych rozważaniach wykorzystano osadzenie problemu w starszym chipie D-Wave 2000Q, aby zademonstrować koncepcje łańcuchów kubitów. Uzyskany wynik osadzenia można również przedstawić za pomocą grafu, co pokazano na rysunku 27. Zmienna logiczna s_3 reprezentowana jest przez dwa kubity, które oznaczymy s_{3a} oraz s_{3b}. Te dwa kubity tworzą łańcuch długości dwa. Dla łańcucha przyjmujemy wagę o wartości -1.

Następnie wagi są skalowane do zakresów akceptowalnych przez fizyczny chip QPU, czyli od -2 do $+2$ dla wag wierzchołków oraz od -1 do $+1$ dla wag krawędzi. Oznacza to, że jeśli problem w modelu Isinga posiada duże wartości wag, to zostaną one przeskalowane do znacznie mniejszego zakresu. W naszym przypadku wszystkie wagi dzielimy przez wartość 375 – maksymalna waga w problemie. Na rysunku 28

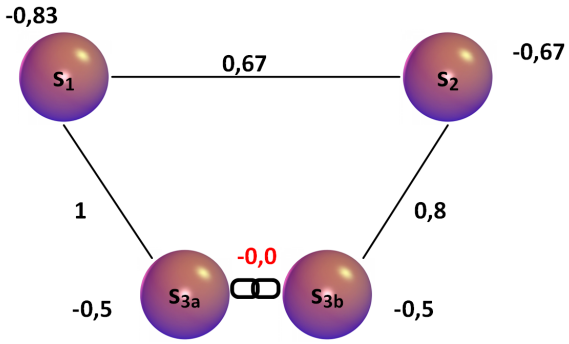
Rysunek 26: Osadzenie grafu modelu Isinga w chipie jednostki D-Wave Advantage. *Źródło: [1]*



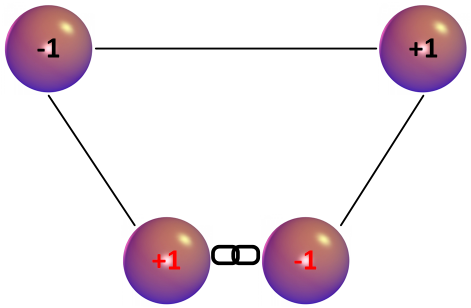
Rysunek 27: Graf przedstawiający osadzenie problemu w chipie jednostki D-Wave 2000Q. *Źródło: na podstawie [1]*

przedstawiono graf problemu po przeskalowaniu, który to graf został zaimplementowany w fizycznym chipie QPU. Należy zwrócić uwagę, że waga krawędzi pomiędzy s_{3a} oraz s_{3b}, a więc siła łańcucha, wynosi 0, a dokładnie $-0,002(6)$, ponieważ wagi pozostałych krawędzi były znacznie większe.

Jedno z rozwiązań, które możemy uzyskać po zakończeniu procesu wyżarzania, przedstawiono na rysunku 29. W tym rozwiązaniu kubity s_2 i s_{3a} zwróciły wartość $+1$, natomiast kubity s_1 i s_{3b} zwróciły wartość -1. Należy pamiętać jednak, że kubity s_{3a} oraz s_{3b} reprezentują jedną zmienną logiczną, dlatego muszą zwracać tę samą wartość. Jeśli zwracają różne wartości, oznacza to, że łańcuch został przerwany oraz, że ustalono zbyt małą wartość jego siły. Jeśli zwrócone z jednostki QPU wyniki zawierają przerwany łańcuch, to dla takiej zmiennej logicznej narzędzia oprogramowania systemu D-Wave wybierają jedną z wartości binarnych jako rozwiązanie, ale niekoniecznie poprawne. Dlatego należy uaktualnić siłę przerwanego łańcucha i ponownie przesłać problem do systemu D-Wave.



Rysunek 28: Graf przedstawiający osadzenie przeskalowanego problemu w chipie jednostki D-Wave 2000Q. *Źródło: na podstawie [1]*



Rysunek 29: Graf przedstawiający jedno z rozwiązań. *Źródło: na podstawie [1]*

W tym momencie, zwracane rozwiązania tłumaczone są z powrotem na zmienne binarne $\{0,1\}$, które należą do modelu QUBO. Oznacza to, że każda wartość -1 jest zastępowana przez 0, a każda wartość 1 pozostaje bez zmian. Dla rozwiązywanego problemu, zwrócone minimum globalne ma postać: $x_1 = 0$, $x_2 = 1$ oraz $x_3 = 1$, co jest prawidłowym rozwiązaniem.

2.7 Złożoność obliczeniowa rozwiązywania problemów za pomocą komputera D-Wave

Oszacowanie złożoności obliczeniowej rozwiązania problemu QUBO za pomocą wyżarzania kwantowego wciąż wymaga wielu badań. Uważa się jednak, że taka złożoność zależy głównie od liczby zmiennych. Dokładna złożoność czasowa rozwiązania problemu QUBO za pomocą wyżarzania kwantowego nie została jeszcze obliczona. Jednak w [51], wykorzystując heurystykę oszacowano, że oczekiwany czas rozwiązania problemu QUBO składającego się z N zmiennych binarnych jest równy $O\left(e^{\sqrt{N}}\right)$.

3 Model transformacji układu równań wielomianowych o wielu zmiennych do problemu QUBO

Jak już wspomniano w rozdziale 1, w kryptoanalizie algebraicznej szyfr blokowy jest opisywany relacjami wielomianowymi nad jakimś ciałem, najczęściej nad ciałem $GF(2)$ lub pierścieniem liczb całkowitych. Wyznaczone wielomiany, opisujące atakowany szyfr, odzwierciedlają zachowanie tego szyfru i wiążą bity znanej pary tekst jawny – szyfrogram z bitami tajnego klucza. Rozwiązanie tak skonstruowanego układu równań wielomianowych powinno zapewnić odzyskanie tajnego klucza. Największym problemem kryptoanalizy algebraicznej jest krok drugi, a więc rozwiązanie układu równań wielomianowych. W niniejszym rozdziale przedstawiony zostanie model transformacji problemu rozwiązania układu równań wielomianowych o wielu zmiennych, opisującego dany szyfr, do problemu znalezienia globalnego rozwiązania problemu optymalizacyjnego w postaci QUBO. Jako narzędzie wykorzystane do rozwiązania tego problemu zastosowany zostanie komputer kwantowy D-Wave.

3.1 Przedstawienie szyfru blokowego za pomocą układu równań wielomianowych o wielu zmiennych

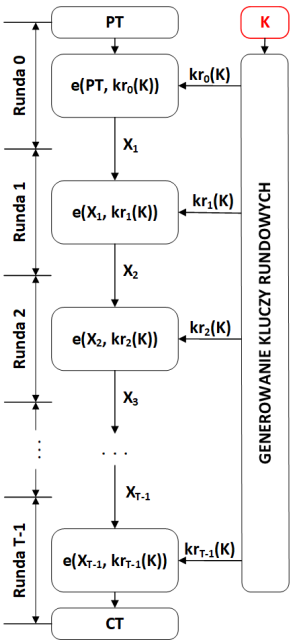
Szyfr blokowy to funkcja, parametryzowana za pomocą klucza, przekształcająca blok tekstu jawnego na blok szyfrogramu. Dowolny szyfr blokowy, o długości bloku równej n i długości klucza równej m, można przedstawić jako układ $(\mathcal{P}, \mathcal{C}, \mathcal{K}, E, D)$, spełniający następujące warunki:

- $\mathcal{P}$ – to skończony zbiór możliwych tekstów jawnych $PT = \{0,1\}^n$,
- $\mathcal{C}$ – to skończony zbiór możliwych szyfrogramów $CT = \{0,1\}^n$,
- $\mathcal{K}$ – to przestrzeń kluczy, czyli skończony zbiór możliwych kluczy $K = \{0,1\}^m$,
- E, D – są zbiorami funkcji, zdefiniowanymi jako $E : \mathcal{P} \times \mathcal{K} \to \mathcal{C}$ oraz $D : \mathcal{C} \times \mathcal{K} \to \mathcal{P}$.

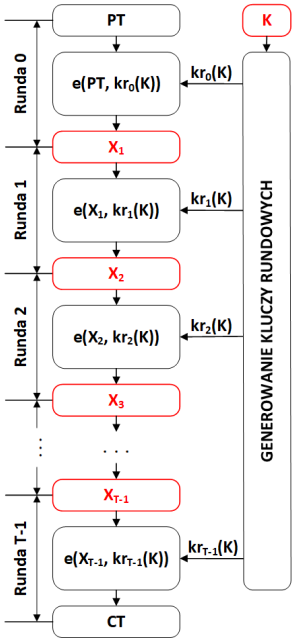
Ponadto, dla każdego klucza $K \in \mathcal{K}$ funkcja $E(PT, K)$ jest przekształceniem odwracalnym z $\{0,1\}^n$ na $\{0,1\}^n$, zwanym algorytmem szyfrowania i oznaczanym jako $E_K(PT)$. Dla ustalonego klucza, funkcja ta jest bijekcją zbioru tekstów jawnych w zbiór szyfrogramów. Odpowiadającym mu przekształceniem odwrotnym jest algorytm deszyfrowania, oznaczany jako $D_K(CT)$, gdzie zachodzi: $D_K\left(E_K(PT)\right) = PT$, dla każdego tekstu jawnego PT.

Każdy szyfr blokowy można przedstawić jako równania wielomianowych o wielu zmiennych, reprezentujące ustalone ograniczenia. Uzyskane rozwiązanie musi spełniać wszystkie ograniczenia i może istnieć zero, jedno lub więcej niż jedno rozwiązanie. Ograniczenia to równania modelujące działanie szyfru oraz relacje pomiędzy elementami znanymi i niewiadomymi. Znanymi elementami jest μ par tekstu jawnego $PT_1, ..., PT_\mu$ i odpowiadającego mu szyfrogramu $CT_1, ..., CT_\mu$, tak, że zachodzi: $E_K(PT_i) = CT_i$ dla wszystkich $i \in \{1, ..., \mu\}$. Niewiadomymi są bity szukanego klucza K. Niemal zawsze istnieją dodatkowe ograniczenia i niewiadome. Jeśli wygenerowane równania prawidłowo reprezentują działanie danego szyfru, to ponieważ wiadomości zostały zaszyfrowane, wiemy, że musiał zostać użyty jakiś klucz, a więc przynajmniej jeden klucz spełnia wszystkie ograniczenia. Stąd wiemy również, że istnieje albo jedno, albo więcej niż jedno rozwiązanie. Ogólnie, kryptoanaliza algebraiczna wymaga zapisania wystarczającej liczby ograniczeń, aby zredukować liczbę możliwych kluczy do jednego oraz tylu, aby system można było rozwiązać w rozsądnym czasie. W szczególności dąży się do tego, aby cały proces przebiegał szybciej niż atak pełnego przeszukiwania. Podczas rozwiązywania układów równań wielomianowych kluczowymi miarami trudności, dla obecnie wykorzystywanych narzędzi, są liczba równań, liczba zmiennych oraz maksymalny stopień wielomianów w równaniach w układzie. W przypadku wyżarzacza kwantowego D-Wave należy wziąć pod uwagę również inne własności generowanych układów, co zostanie omówione w dalszej części tego rozdziału.

W niniejszej rozprawie analizowane będą iterowane szyfry blokowe, które polegają na wielokrotnym wykonaniu tych samych przekształceń, tworzących rundę. Schemat T-rundowego, iterowanego szyfru blokowego przedstawiono na rysunku 30,



(a) T-rundowy, iterowany szyfr blokowy.



(b) T-rundowy, iterowany szyfr blokowy, ze stanami pośrednimi.

Rysunek 30: Podział T-rundowego, iterowany szyfru blokowego za pomocą stanów pośrednich.

gdzie $e\left(X_i, kr_i(K)\right)$ jest funkcją rundy, X_i jest wyjściem z rundy poprzedniej, $kr_i(K)$ jest kluczem rundowym, wyznaczonym na podstawie klucza głównego K, a kolorem czerwonym oznaczono elementy niewiadome. Na początku niewiadomymi są tylko zmienne binarne, reprezentujące klucz główny K, co przedstawiono na rysunku 30a. Jednak, ponieważ funkcja rundy $e\left(X_i, kr_i(K)\right)$ musi zawierać operację nieliniową, która zapewnia bezpieczeństwo szyfru przed atakami (m.in. przed kryptoanalizą liniową i różnicową), to przedstawienie szyfru za pomocą równań, postaci:

$$e\left(...e\left(e\left(e\left(PT, kr_0(K)\right), kr_1(K)\right), kr_2(K)\right), ..., kr_{T-1}(K)\right) = CT, \qquad (27)$$

będących złożeniem kolejnych funkcji rundy, jest nieefektywne, ponieważ operacja nieliniowa sukcesywnie będzie zwiększać stopień tych równań. Przykładowo, dla algorytmu AES128, stopień równań postaci (27) może wynosić 128, ponieważ tyle jest zmiennych niewiadomych, reprezentujących bity klucza.

Można jednak skonstruować drugi układ równań, którego rozwiązanie jest również rozwiązaniem pierwszego układu, tak aby stopień wszystkich równań w drugim

układzie był znacznie mniejszy. To, o ile konkretnie mniejszy, zależy od struktury danego szyfru. Dla algorytmu AES128 można skonstruować układ, składający się z równań stopnia co najwyżej dwa. Można to osiągnąć poprzez zdefiniowanie po każdej rundzie tak zwanego stanu pośredniego, określonego za pomocą dodatkowych zmiennych binarnych. Podejście to przedstawiono na rysunku 30b, gdzie stany pośrednie pokazano kolorem czerwonym i oznaczono jako X_i. Zmienne stanów pośrednich są kolejnymi niewiadomymi w rozwiązywanym układzie równań wielomianowych, a same równania reprezentują tylko pojedynczą rundę. Należy zauważyć, że podejście to jest od samego początku powszechnie wykorzystywane w kryptoanalizie algebraicznej. Tak więc, T-rundowy, iterowany szyfr blokowy z rysunku 30b można przedstawić za pomocą układu równań następującej postaci:

$$\begin{cases} e\left(PT, kr_0(K)\right) - X_1 = 0, \\ e\left(X_1, kr_1(K)\right) - X_2 = 0, \\ \vdots \\ e\left(X_{T-1}, kr_{T-1}(K)\right) - CT = 0. \end{cases} \tag{28}$$

Dane równanie układu (28) jest spełnione wtedy i tylko wtedy, gdy $e\left(X_i, kr_i(K)\right) = X_{i+1}$. Oznacza to, że wszystkie równania są spełnione wtedy i tylko wtedy, gdy $e\left(PT, kr_0(K)\right) = X_1$, $e\left(X_1, kr_1(K)\right) = X_2$, $e\left(X_2, kr_2(K)\right) = X_3, \ldots$, $e\left(X_{T-1}, kr_{T-1}(K)\right) = CT$. Wykonując kolejno podstawienia, ostatecznie otrzymujemy, że równania są spełnione tylko wtedy, gdy $e\left(...e\left(e\left(e\left(PT, kr_0(K)\right), kr_1(K)\right), kr_2(K)\right), \ldots, kr_{T-1}(K)\right) = CT$, czyli gdy spełnione jest równanie (27). Dlatego też, zdefiniowanie stanów pośrednich nie wprowadza żadnych fałszywych rozwiązań oraz nie wprowadza nieoznaczoności, a ostatecznie obniża stopień równań. Stąd, istnieje dokładnie jedno rozwiązanie, spełniające układ postaci (28), które zawiera szukane wartości $K, X_1, X_2, ..., X_{T-1}$.

Oczywiście, aby powiązać klucz główny K, z kluczami rundowymi kr_i, algorytm generowania kluczy rundowych również należy opisać za pomocą równań wielomianowych. Jednak struktura algorytmu generowania kluczy rundowych zależy od konkretnego szyfru. Dlatego, w kolejnych rozdziałach, będzie ona analizowana dla

przykładowych szyfrów.

Jak już wcześniej wspomniano, kryptoanaliza algebraiczna wymaga zdefiniowania tylu równań, aby można było jednoznacznie odzyskać klucz główny. W związku z tym, jeśli długość bloku wejściowego n jest większa lub równa długości klucza m, to dla każdej pary tekst jawny – szyfrogram istnieje jeden właściwy klucz. Sytuacja jest inna, jeśli długość bloku wejściowego jest mniejsza niż długość klucza. W takim przypadku, gdy znane będą tylko pojedyncze pary tekst jawny – szyfrogram, dla każdej takiej pary rozwiązaniem będzie co najwyżej 2^{m-n} kluczy, ale tylko jeden klucz będzie tym właściwym, gdy wzięte zostałyby pod uwagę inne pary. W celu znalezienia, z dużym prawdopodobieństwem, odpowiedniego klucza, wymaganych jest $\lceil \frac{m}{n} \rceil$ par tekst jawny – szyfrogram. Dlatego też, dla wariantów danego szyfru, dla których długość bloku wejściowego jest mniejsza niż długość klucza, równania algorytmu szyfrowania wygenerowane zostaną dla $\lceil \frac{m}{n} \rceil$ różnych par tekst jawny – szyfrogram, tworząc $\lceil \frac{m}{n} \rceil$ układów postaci (28). Należy zwrócić uwagę, że w takim przypadku liczba zmiennych binarnych, wymaganych do utworzenia docelowego układu, opisującego cały szyfr, jest większa.

Głębsza analiza przedstawienia danego szyfru blokowego za pomocą układu równań wielomianowych o wielu zmiennych zostanie przedstawiona w następnych rozdziałach, dla konkretnych szyfrów.

Załóżmy, że pewien szyfr został opisany za pomocą układu, składającego się z s równań dla wielomianów m' zmiennych, stopnia co najwyżej d, nad ciałem $GF(q)$. Otrzymujemy następujący problem:

Znaleźć wektor $(x_0, x_1, \ldots, x_{m'-1}) \in \mathbb{B}^{m'}$ *spełniający układ następującej postaci:*

$$\begin{cases} f_0(x_0, x_1, \ldots, x_{m'-1}) = 0, \\ f_1(x_0, x_1, \ldots, x_{m'-1}) = 0, \\ \vdots \\ f_{s-1}(x_0, x_1, \ldots, x_{m'-1}) = 0, \end{cases} \tag{29}$$

gdzie $f_i(x_0, x_1, \ldots, x_{m'-1})$, *dla* $i = \overline{0, s-1}$ *są wielomianami.*

3.2 Przekształcenie układu równań wielomianowych o wielu zmiennych do układu funkcji pseudo-Boolowskich

Niech $\mathbb{Z}$ oznacza zbiór liczb całkowitych, $\mathbb{B} = \{0,1\}$ oraz niech $V = \{0,1,\ldots,m'-1\}$ oznacza zbiór indeksów zmiennych binarnych. Odwzorowania postaci $f' : \mathbb{B}^n \to \mathbb{Z}$ nazywane są funkcjami pseudo-Boolowskimi. W [37] udowodniono, że każda funkcja pseudo-Boolowska może być zapisana jako wielomian, którego zmienne są liniowe (ponieważ zmienne są binarne to zachodzi $x^2 = x$) oraz który, po redukcji wyrazów podobnych, jest jednoznacznie określony z dokładnością do sum i iloczynów. Wtedy funkcja pseudo-Boolowska jest postaci:

$$f'(x_0, x_1, \ldots, x_{m'-1}) = \sum_{S \subseteq V} a_S \prod_{j \in S} x_j, \tag{30}$$

gdzie S jest danym podzbiorem zbioru indeksów zmiennych V, a $a_S \in \mathbb{Z}$ jest współczynnikiem określonego podzbioru zmiennych, wyznaczonego przez S.

Na tej podstawie, układ (29) przekształcany jest do układu funkcji pseudo-Boolowskich, następującej postaci:

$$\begin{cases} f'_0(x_0, x_1, \ldots, x_{m'-1}) = qk_0, \\ f'_1(x_0, x_1, \ldots, x_{m'-1}) = qk_1, \\ \vdots \\ f'_{s-1}(x_0, x_1, \ldots, x_{m'-1}) = qk_{s-1}, \end{cases} \tag{31}$$

gdzie $k_i \in \mathbb{Z}$ jest wielokrotnością liczby elementów ciała $GF(q)$ dla i-tego równania, a każde równanie jest postaci (30).

3.3 Linearyzacja układu funkcji pseudo-Boolowskich

Możliwe są dwa różne sposoby wykonania transformacji układu nieliniowych równań wielomianowych do problemu optymalizacyjnego w postaci QUBO. Podczas transformacji można wykorzystać linearyzację, albo kwadratyzację. W przypadku ogólnym

problem linearyzacji wielomianów wielu zmiennych jest problemem NP-trudnym, jednak w niniejszej rozprawie zdecydowano się zastosować linearyzację, ponieważ dla wybranych do analizy szyfrów blokowych, problem linearyzacji okazał się problemem o złożoności wielomianowej, co zostanie pokazane w następnych rozdziałach. Istnieje także drugi powód, dla którego zdecydowano się na zastosowanie linearyzacji; zostanie on przedstawiony w dalszej części tego rozdziału.

Do wykonania linearyzacji układu (31) zastosowano metodę redukcji zaproponowaną przez Rosenberga w pracy [55], która polega na zastąpieniu każdego jednomianu kwadratowego nową, pomocniczą zmienną binarną. Przykładowo, niech w równaniach układu występuje jednomian postaci $x_1x_2x_3$, który wymaga linearyzacji. W tym celu wprowadzamy następujące nowe zmienne pomocnicze x_4 i x_5 oraz wykonujemy następujące podstawienia: najpierw $x_4 = x_1x_2$, otrzymując jednomian kwadratowy x_3x_4, a następnie $x_5 = x_3x_4$. Ostatecznie, zamiast jednomianu $x_1x_2x_3$, otrzymujemy zmienną x_5. Jednak, aby to podstawienie było spełnione, należy wprowadzić dodatkowe ograniczenia. Zatem $x_1x_2x_3 = x_5$ wtedy i tylko wtedy, gdy spełnione są równości $x_4 = x_1x_2$ i $x_5 = x_3x_4$. W docelowym problemie QUBO nie można stawiać warunków, zatem, aby podstawienie było spełnione, dodatkowe ograniczenia przedstawiane są w postaci kary i dodawane do, wyznaczonej w następnym kroku, funkcji celu, tworząc rozszerzoną funkcję celu. Wykorzystywać będziemy postać kary zaproponowaną przez Rosenberga, która dla dowolnego podstawienia $x_k = x_ix_j$ wynosi $x_ix_j - 2x_k(x_i + x_j) + 3x_k$. Gwarantuje ona, że jeśli $x_ix_j = x_k$, to wartość kary wynosi zero; w przeciwnym przypadku, gdy $x_ix_j \neq x_k$, wartość kary jest dodatnią liczbą całkowitą, obrazuje to tabela 1. Ponieważ przekształcenia te prowadzą do docelowego problemu optymalizacyjnego w postaci problemu QUBO, w którym szuka się minimum funkcji celu, każde błędne podstawienie powoduje dodanie do funkcji celu dodatniej wartości i w konsekwencji powoduje odrzucenie takiego rozwiązania. Tak więc proces linearyzacji nie wprowadza fałszywych rozwiązań i o ile wszystkie ograniczenia są spełnione, otrzymujemy jednoznaczne rozwiązanie układu (29). Należy jednak zastanowić się nad wagą kary. Wartość rozszerzonej funkcji celu dla błędnego rozwiązania powinna być większa, niż jakikolwiek inny punkt w przestrzeni

rozwiązań funkcji celu, aby mieć pewność, że takie rozwiązanie zostanie odrzucone. Wielkość wagi kary, dla proponowanej w niniejszej rozprawie metody transformacji, zostanie omówiona w dalszej części tego rozdziału.

Tabela 1: Wyznaczenie wartości kary dla wszystkich możliwych wartości zmiennych podstawienia $x_k = x_i x_j$.

x_i	x_j	x_k	*Wartość kary*
0	0	0	0
0	0	1	3
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Koszt każdej wykonanej redukcji to jedna dodatkowa zmienna binarna. Jeśli redukowany jednomian ma stopień d, to dla takiego jednomianu należy wykonać $d-1$ podstawień. Jednak zaletą jest to, że dane podstawienie można wykonać w każdym równaniu całego układu, jeśli występuje w nim redukowany jednomian kwadratowy. Największym problemem jest ustalenie kolejności redukowanych jednomianów. W niniejszej rozprawie wielomiany przedstawiono w odwrotnym porządku stopniowo – leksykograficznym. Kolejne podstawienia podczas linearyzacji wykonano zgodnie z kolejnością pojawiania się jednomianów kwadratowych w danym równaniu.

Załóżmy, że do wykonania linearyzacji układu (31) wymaganych jest t dodatkowych zmiennych binarnych. W związku z tym otrzymany układ można przedstawić w następującej postaci:

$$\begin{cases} f_{0_{lin}}(x_0, x_1, \ldots, x_{m'-1}, x_{m'}, x_{m'+1}, \ldots, x_{m'+t-1}) = qk_0, \\ f_{1_{lin}}(x_0, x_1, \ldots, x_{m'-1}, x_{m'}, x_{m'+1}, \ldots, x_{m'+t-1}) = qk_1, \\ \vdots \\ f_{s-1_{lin}}(x_0, x_1, \ldots, x_{m'-1}, x_{m'}, x_{m'+1}, \ldots, x_{m'+t-1}) = qk_{s-1}. \end{cases} \tag{32}$$

Wyznaczona kara po wszystkich podstawieniach, oznaczona jako $\boldsymbol{Pen}_{lin}$, mnożona

jest przez wagę M i dodawana jest do funkcji celu problemu QUBO.

3.4 Przekształcenie zlinearyzowanego układu do problemu QUBO

W pracach [53], [8], [14] oraz [13] zaprezentowano wykorzystanie wyżarzania kwantowego do rozwiązania liniowego problemu najmniejszych kwadratów. Na tej podstawie, w celu przekształcenia zlinearyzowanego układu równań wielomianowych (32) do problemu QUBO, zastosowano metodę najmniejszych kwadratów. Funkcję celu docelowego problemu QUBO zdefiniowano jako sumę kwadratów reszt, dla reszt postaci $f_{i_{lin}}(x_0, x_1, \ldots, x_{m'-1}, x_{m'}, x_{m'+1}, \ldots, x_{m'+t-1}) - qk_i$ oraz dla $i = \overline{0, s-1}$.

Aby wyznaczyć funkcję celu problemu QUBO, należy najpierw przedstawić wartości wielokrotności k_i za pomocą dodatkowych zmiennych binarnych. W tym celu zakłada się, że wielomian $f_{i_{lin}}$ przyjmuje maksymalną wartość $f_{i_{max}}$. Ponieważ współczynniki wielomianów w pierwotnym układzie należały do ciała $GF(q)$, a dotychczasowe przekształcenia nie wpływały na ich znak, ani na ich wartość, wielomian przyjmuje wartość maksymalną wtedy i tylko wtedy, gdy wszystkie jego zmienne binarne przyjmują wartość 1. Wtedy maksymalna wartość wielomianu jest równa sumie współczynników wszystkich jego jednomianów. Ponieważ chcemy otrzymać rozwiązanie bazowe, musi być spełniona nierówność $qk_i \leq f_{i_{max}}$. Na tej podstawie można wyznaczyć maksymalną możliwą wartość wielokrotności k_i, z równości: $k_{i_{max}} = \lfloor \frac{f_{i_{max}}}{q} \rfloor$. Wartość ta determinuje liczbę pomocniczych zmiennych binarnych, za pomocą których przedstawia się rzeczywistą wartość k_i, w następujący sposób:

$$k_i = \sum_{j=0}^{bl(k_{i_{max}})-2} 2^j x_j + (k_{i_{max}} - 2^{bl(k_{i_{max}})-1} + 1) \cdot x_{bl(k_{i_{max}})-1}, \tag{33}$$

gdzie $bl(k_{i_{max}})$ oznacza długość bitową liczby całkowitej $k_{i_{max}}$. Jak można zauważyć na podstawie równania (33), wszystkie wartości wielokrotności k_i przedstawione zostają za pomocą wielomianów liniowych, które oznaczmy jako k'_i.

Niech r oznacza liczbę dodatkowych zmiennych binarnych, potrzebnych do przedstawienia maksymalnych wartości wielokrotności k_i w układzie (32), dla $i = \overline{0, s-1}$.

Po zdefiniowaniu wielomianów k'_i dla odpowiadających im wartości k_i oraz przeniesieniu ich na lewą stronę, otrzymujemy następujący układ:

$$\begin{cases} f_{0_{lin}}(x_0, x_1, \ldots, x_{m'+t-1}) - qk'_0(x_{m'+t}, x_{m'+t+1}, \ldots, x_{m'+t+r-1}) = 0, \\ f_{1_{lin}}(x_0, x_1, \ldots, x_{m'+t-1}) - qk'_1(x_{m'+t}, x_{m'+t+1}, \ldots, x_{m'+t+r-1}) = 0, \\ \vdots \\ f_{s-1_{lin}}(x_0, x_1, \ldots, x_{m'+t-1}) - qk'_{s-1}(x_{m'+t}, x_{m'+t+1}, \ldots, x_{m'+t+r-1}) = 0. \end{cases} \tag{34}$$

Ponieważ nie wiemy jaka jest rzeczywista wartość wielomianów $f_{i_{lin}}(x_0, \ldots, x_{m'+t-1})$, wprowadzamy do układu nieoznaczoność dla bitów $(x_{m'+t}, \ldots, x_{m'+t+r-1})$, reprezentujących wartości k_i. Wynika ona z tego, że jeśli rzeczywista wartość k_i jest co najmniej dwa razy mniejsza od wartości maksymalnej $k_{i_{max}}$, to rzeczywista wartość może być przedstawiona, za pomocą dostępnych zmiennych binarnych, na kilka sposobów. Przykładowo, niech $k_{i_{max}} = 6$. Wtedy potrzebne są trzy dodatkowe zmienne binarne, a wartość k_i przedstawiana jest jako wielomian $k'_i(x_j, x_{j+1}, x_{j+2}) = x_j + 2x_{j+1} + 3x_{j+2}$. Jednak, jeśli rzeczywista wartość $k_i = 3$, możliwe są dwa prawidłowe rozwiązania: $x_j = 1$, $x_{j+1} = 1$, $x_{j+2} = 0$ oraz $x_j = 0$, $x_{j+1} = 0$, $x_{j+2} = 1$. Konkretna wartość wielokrotności k_i jest bez znaczenia dla rozwiązania pierwotnego układu (29), o ile równania układu (34) są spełnione.

Ponieważ zmienne $(x_0, x_1, \ldots, x_{m'+t-1})$ wielomianów $f_{i_{lin}}$ są jednoznacznie określone pod warunkiem, że spełnione są ograniczenia związane z linearyzacją, dla każdego wielomianu $f_{i_{lin}}$ istnieje dokładnie jedna wartość całkowita k_i. Z drugiej strony, załóżmy, że jedno z równań $f_{i_{lin}}(x_0, \ldots, x_{m'+t-1}) - qk'_i(x_{m'+t}, \ldots, x_{m'+t+r-1}) = 0$ jest spełnione dla błędnej wartości k_i. W konsekwencji zmienne ze zbioru $\{x_0, x_1, \ldots \ldots, x_{m'+t-1}\}$ wielomianu $f_{i_{lin}}$ przyjmą błędne wartości, a więc otrzymamy sprzeczność, ponieważ ich wartość jest jednoznacznie określona i pewien podzbiór równań z pozostałych równań układu (34) nie zostanie spełniony. Stąd wniosek, że rozwiązanie układu (34) jest również rozwiązaniem układu pierwotnego (29).

Jako funkcję celu problemu w postaci QUBO, bierze się sumę kwadratów wie-

lomianów układu (34), postaci:

$$\sum_{i=0}^{s-1}\left(f_{i_{lin}}\left(x_0,x_1,\ldots,x_{m'+t-1}\right)-qk_i'\left(x_{m'+t},x_{m'+t+1},\ldots,x_{m'+t+r-1}\right)\right)^2. \tag{35}$$

Aby pokazać, że rzeczywiście otrzymywany jest problem w postaci QUBO, jako N oznaczmy liczbę wszystkich zmiennych binarnych w układzie (34), czyli $N = m'+t+r$ oraz przedstawmy układ (34) w następującej postaci macierzowej:

$$\mathbf{A}\mathbf{x}+\mathbf{c}=0 \tag{36}$$

gdzie:

- $\mathbf{A}$ jest macierzą współczynników wielomianów $f_{i_{lin}}(x_0,x_1,\ldots,x_{m'+t-1})+$ $-qk_i'(x_{m'+t},x_{m'+t+1},\ldots,x_{N-1})$, o wymiarze $s\times N$,
- $\mathbf{x}$ jest N-wymiarowym wektorem zmiennych binarnych $x_0,x_1,\ldots,x_{N-1}$, występujących w wielomianach $f_{i_{lin}}(x_0,\ldots,x_{m'+t-1})-qk_i'(x_{m'+t},x_{m'+t+1},\ldots,x_{N-1})$,
- $\mathbf{c}$ jest s-wymiarowym wektorem stałych, występujących w wielomianach $f_{i_{lin}}(x_0,x_1,\ldots,x_{m'+t-1})-qk_i'(x_{m'+t},x_{m'+t+1},\ldots,x_{N-1})$.

Równanie (36) można przedstawić w następującej postaci:

$$\mathbf{A}x+\mathbf{c}=\begin{pmatrix}A_{0,0}&A_{0,1}&\ldots&A_{0,N-1}\\A_{1,0}&A_{1,1}&\ldots&A_{1,N-1}\\\vdots&\vdots&\ddots&\vdots\\A_{s-1,0}&A_{s-1,1}&\ldots&A_{s-1,N-1}\end{pmatrix}\times\begin{pmatrix}x_0\\x_1\\\vdots\\x_{N-1}\end{pmatrix}+\begin{pmatrix}c_0\\c_1\\\vdots\\c_{s-1}\end{pmatrix}=0, \tag{37}$$

co jest równoważne:

$$\mathbf{A}x+\mathbf{c}=\begin{pmatrix}A_{0,0}x_0+A_{0,1}x_1+\ldots+A_{0,N-1}x_{N-1}+c_0\\A_{1,0}x_0+A_{1,1}x_1+\ldots+A_{1,N-1}x_{N-1}+c_1\\\vdots\\A_{s-1,0}x_0+A_{s-1,1}x_1+\ldots+A_{s-1,N-1}x_{N-1}+c_{s-1}\end{pmatrix}=0. \tag{38}$$

Wyznaczając euklidesową (drugą) normę wektora reszt (równanie (38)), otrzymu-

jemy:

$$\begin{aligned}
&\|\mathbf{A}x+\mathbf{c}\|_2^2 = \sum_{i=0}^{s-1} |A_{i,0}x_0 + A_{i,1}x_1 + \ldots + A_{i,N-1}x_{N-1} + c_i|^2 = \\
&= \sum_{i=0}^{s-1} \left(A_{i,0}^2 x_0^2 + 2A_{i,0}c_0 x_0 + A_{i,1}^2 x_1^2 + \ldots + A_{i,N-1}^2 x_{N-1}^2 + 2A_{i,N-1}c_{N-1}x_{N-1} \right) + \\
&+ \sum_{i=0}^{s-1} \left(2A_{i,0}A_{i,1}x_0x_1 + 2A_{i,0}A_{i,2}x_0x_2 + \ldots + 2A_{i,N-2}A_{i,N-1}x_{N-2}x_{N-1} \right) + \sum_{i=0}^{s-1} c_i^2 = \\
&= \sum_{i=0}^{s-1} \sum_{j=0}^{N-1} A_{i,j}\left(A_{i,j}+2c_i\right)x_j + \sum_{i=0}^{s-1} 2 \sum_{j<k} A_{i,j}A_{i,k}x_jx_k + \sum_{i=0}^{s-1} c_i^2 = \\
&= \sum_j B_{j,j}x_j + \sum_{j<k} B_{j,k}x_jx_k + C = x^T\mathbf{B}x + C,
\end{aligned} \tag{39}$$

gdzie:

$$B_{j,j} = \sum_{i=0}^{s-1} A_{i,j}\left(A_{i,j}+2c_i\right), \tag{40}$$

$$B_{j,k} = 2\sum_{i=0}^{s-1} A_{i,j}A_{i,k} \tag{41}$$

oraz

$$C = \sum_{i=0}^{s-1} c_i^2. \tag{42}$$

W celu znalezienia rozwiązania układu (34), należy znaleźć rozwiązanie globalne następującego problemu optymalizacyjnego:

$$\min_{x\in\{0,1\}^N} y = x^T\mathbf{B}x + C, \tag{43}$$

w którym warunkami brzegowymi są podstawienia nowych zmiennych binarnych, uzyskanych w procesie linearyzacji.

Wyznaczona podczas linearyzacji kara $M \cdot Pen_{lin}$, jest zapisaniem warunków brzegowych problemu (43), w postaci wielomianu stopnia dwa, który można przedstawić w postaci macierzowej:

$$x^T\mathbf{D}x. \tag{44}$$

Rozwiązanie optymalne problemu z warunkami jest takie samo, jak rozwiązanie bezwarunkowego problemu z karą. Stąd problem (43) można zapisać w następującej postaci bezwarunkowej:

$$\min_{x\in\{0,1\}^N} y = x^T\mathbf{B}x + x^T\mathbf{D}x + C = x^T\mathbf{Q}x + C. \tag{45}$$

Ponieważ zmienne x_i są zmiennymi binarnymi, otrzymano problem optymalizacyjny w postaci QUBO:

$$\min_{x\in\{0,1\}^N} y = x^T\mathbf{Q}x. \tag{46}$$

Stała C nie jest składnikiem postaci QUBO.

Celem proponowanej metody, opartej na atakach algebraicznych, jest odzyskanie poprawnych wartości bitów klucza głównego K, które są rozwiązaniem pierwotnego układu (29). W celu ich odzyskania, należy znaleźć globalne minimum problemu optymalizacyjnego w postaci QUBO (46), czyli rozwiązanie ze stanu podstawowego, o minimalnej energii własnej układu kwantowego.

Wróćmy teraz do ustalenia wartości wagi kary M oraz wyjaśnienia, dlaczego wybrano linearyzację, a nie kwadratyzację. Transformacja układu nieliniowych równań wielomianowych do problemu optymalizacyjnego w postaci QUBO, wykorzystująca kwadratyzację, wykonywana jest w następujących krokach:

1. przekształcenie układu równań wielomianowych o wielu zmiennych, opisujących dany szyfr, do układu funkcji pseudo-Boolowskich;

2. wyznaczenie wartości wielokrotności k_i oraz wyznaczenie sumy kwadratów reszt, w celu otrzymania problemu optymalizacyjnego z wielomianami stopnia większego od 2;

3. wykonanie kwadratyzacji, na przykład metodą Rosenberga, w celu otrzymania kwadratowego problemu optymalizacyjnego w postaci QUBO.

Jeśli najpierw wyznaczona zostanie suma kwadratów reszt, a dopiero później przeprowadzona redukcja stopnia wielomianu, to kolejne podstawienia niszczą związki pomiędzy zmiennymi w danym jednomianie. W konsekwencji, pomimo że suma kwa-

dratów ma zawsze wartość większą lub równą zero, po wprowadzeniu nowych zmiennych podczas redukcji, dla jakiegoś rozwiązania z błędnym podstawieniem wartość funkcji celu może być ujemna. Stąd celem wprowadzanej kary jest zapewnienie, aby funkcja celu nie przyjęła wartości ujemnych dla błędnych rozwiązań. Przedstawmy to na bardzo prostym przykładzie. Dany jest układ dwóch równań wielomianowych nad ciałem $GF(2)$, postaci:

$$\begin{cases} f_0(x_0, x_1) = x_0x_1 + x_0 + 1 = 0, \\ f_1(x_0, x_1) = x_0 + x_1 + 1 = 0. \end{cases} \tag{47}$$

Po przekształceniu tego układu do układu funkcji pseudo-Boolowskich otrzymujemy następujący układ:

$$\begin{cases} x_0x_1 + x_0 + 1 = 2k_0, \\ x_0 + x_1 + 1 = 2k_1, \end{cases} \tag{48}$$

Następnie wyznaczamy maksymalne możliwe wartości wielomianów, które dla obu wielomianów wynoszą 3. Stąd, każdą z wielokrotności k_0 oraz k_1, można przedstawić za pomocą jednej dodatkowej zmiennej binarnej. Wprowadziwszy dodatkową zmienną binarną x_2, reprezentującą wartość wielokrotności k_0, oraz zmienną x_3 – dla wielokrotności k_1, otrzymujemy układ postaci:

$$\begin{cases} x_0x_1 + x_0 + 1 - 2x_2 = 0, \\ x_0 + x_1 + 1 - 2x_3 = 0. \end{cases} \tag{49}$$

Wyznaczając funkcję celu problemu optymalizacyjnego, jako sumę kwadratów wielomianów układu (49), otrzymujemy wielomian trzeciego stopnia, postaci: $f(x_0, x_1, x_2, x_3) = (x_0x_1 + x_0 + 1 - 2x_2)^2 + (x_0 + x_1 + 1 - 2x_3)^2 = -4x_0x_1x_2 + 7x_0x_1 - 4x_0x_2 - 4x_0x_3 - 4x_1x_3 + 6x_0 + 3x_1 + 2$.

Następnie wykonywana jest kwadratyzacja. Ponieważ w wielomianie $f(x_0, x_1, x_2, x_3)$ występuje tylko jeden jednomian stopnia większego od 2, potrzebne jest tylko jedno podstawienie postaci: $x_1x_2 = x_4$, po którym otrzymujemy wielomian $f(x_0, x_1, x_2, x_3, x_4) = -4x_0x_4 + 7x_0x_1 - 4x_0x_2 - 4x_0x_3 - 4x_1x_3 + 6x_0 + 3x_1 + 2$ oraz karę

$Pen_{kwad} = x_1x_2 - 2(x_1 + x_2) + 3x_4$. Łatwo sprawdzić, że dla błędnego rozwiązania: $x_0 = 1$, $x_1 = 0$, $x_2 = 1$, $x_3 = 1$ oraz $x_4 = 1$, w którym $x_1x_2 \neq x_4$, wartość wielomianu $f(x_0, x_1, x_2, x_3, x_4) = -4$, a wartość kary $Pen_{kwad} = 1$; wtedy wartość funkcji celu problemu optymalizacyjnego wynosi -3 i jest to globalne minimum tego problemu. Dlatego też, aby mieć pewność, że dla żadnego błędnego rozwiązania funkcja celu nie przyjmie wartości ujemnych, w algorytmie redukcji Rosenberga, przedstawionym w [10], waga kary powinna być podwojoną sumą modułów współczynników wszystkich jednomianów w redukowanym wielomianie, powiększoną o jeden: $M = 1 + 2\sum_{S \subseteq V} |c_S|$. W analizowanym wyżej przykładzie, waga kary powinna wynosić $M = 69$.

Jeśli natomiast wykonywana jest transformacja z linearyzacją, w której funkcja celu kwadratowego problemu optymalizacyjnego jest sumą sumy kwadratów i nieujemnej wartości kary, to nie istnieje takie rozwiązanie, dla którego funkcja celu przyjęłaby wartość ujemną. Dlatego w tym przypadku wystarczy, aby waga kary była nieujemna.

Aby odpowiedzieć na pytanie gdzie jest prawa granica przedziału, do którego powinna należeć waga kary, musimy wziąć pod uwagę docelowe narzędzie, które zostanie wykorzystane do rozwiązania problemu QUBO, a więc wyżarzacz kwantowy D-Wave. W podrozdziale 2.6.4, przedstawiającym etapy procesu rozwiązywania problemów optymalizacyjnych, za pomocą komputera D-Wave, opisano przekształcenie rozwiązywanego problemu do postaci akceptowalnej przez komputer D-Wave, gdzie zwrócono uwagę na skalowanie odchyleń. Po przekształceniu problemu w postaci QUBO do modelu Isinga, za pomocą odwzorowania $s_i = 1 - 2x_i$, otrzymane wagi odchyleń, wyznaczone dla elementów macierzy $\mathbf{Q}$ problemu (46), skalowane są do zakresów od -2 do $+2$ dla wag wierzchołków – współczynników jednomianów liniowych oraz od -1 do $+1$ dla wag krawędzi – współczynników jednomianów kwadratowych macierzy $\mathbf{Q}$. W związku z tym zastosowanie kwadratyzacji i odpowiedniej wagi kary spowodowałoby, że różnica odchyleń byłaby na tyle duża, że wagi pierwotnego problemu stałyby się mniej znaczące, niż wagi ograniczeń kwadratyzacji. Jest to kolejny powód, dla którego w niniejszej rozprawie zastosowano transformację z li-

nearyzacją. Aby zachować ważność równań pierwotnego problemu oraz ograniczeń linearyzacji na podobnym poziomie, proponuje się dobrać taki współczynnik kary M, aby największy współczynnik wielomianu kary, po wszystkich podstawieniach, był nie większy niż największy współczynnik wielomianu sumy kwadratów reszt.

3.5 Przykład transformacji układu nieliniowych równań wielomianowych do problemu QUBO

W celu lepszego przedstawienia opisanej w tym rozdziale transformacji, rozważmy następujący przykład. Niech dany będzie następujący układ równań wielomianowych zmiennych boolowskich x_0, x_1, x_2, stopnia dwa, nad ciałem $GF(2)$:

$$\begin{cases} f_0(x_0,x_1,x_2) = x_0x_1 + x_2 + 1 = 0, \\ f_1(x_0,x_1,x_2) = x_1x_2 + x_0 = 0, \\ f_2(x_0,x_1,x_2) = x_0 + x_1 + x_2 + 1 = 0. \end{cases} \tag{50}$$

Krok 1: Przekształcamy układ równań wielomianowych o wielu zmiennych do układu funkcji pseudo-Boolowskich:

$$\begin{cases} f'_0(x_0,x_1,x_2) = x_0x_1 + x_2 + 1 = 2k_0, \\ f'_1(x_0,x_1,x_2) = x_1x_2 + x_0 = 2k_1, \\ f'_2(x_0,x_1,x_2) = x_0 + x_1 + x_2 + 1 = 2k_2. \end{cases} \tag{51}$$

Krok 2: Linearyzujemy układ funkcji pseudo-Boolowskich. Układ (51) zawiera dwa różne jednomiany kwadratowe, dlatego też za jednomian kwadratowy x_0x_1 podstawiamy zmienną x_3 i wyznaczamy karę $Pen_{lin_1} = x_0x_1 - 2x_0x_3 - 2x_1x_3 + 3x_3$ oraz za jednomian kwadratowy x_1x_2 podstawiamy zmienną x_4 i wyznaczamy karę $Pen_{lin_2} = x_1x_2 - 2x_1x_4 - 2x_2x_4 + 3x_4$. Po linearyzacji otrzymujemy następujący układ:

$$\begin{cases} f_{0_{lin}}(x_0,x_1,x_2,x_3,x_4) = x_3 + x_2 + 1 = 2k_0, \\ f_{1_{lin}}(x_0,x_1,x_2,x_3,x_4) = x_4 + x_0 = 2k_1, \\ f_{2_{lin}}(x_0,x_1,x_2,x_3,x_4) = x_0 + x_1 + x_2 + 1 = 2k_2. \end{cases} \tag{52}$$

Krok 3: Przekształcamy zlinearyzowany układ do problemu postaci QUBO. Wyznaczamy wartości k_0, k_1 oraz k_2 na podstawie maksymalnych wartości wielomianów $f_{i_{lin}}$, co przedstawia tabela 2.

Tabela 2: Wyznaczanie wartości wielokrotności k_i dla analizowanego przykładu.

i	$f_{i_{lin}}$	$f_{i_{max}}$	$k_{i_{max}}$	k'_i
0	x_3+x_2+1	3	1	x_5
1	x_4+x_0	2	1	x_6
2	$x_0+x_1+x_2+1$	4	2	x_7+x_8

Podstawiając wyznaczone wielomiany, reprezentujące wartość wielokrotności k_i, do układu (52), otrzymujemy:

$$\begin{cases} f_{0_{lin}}(x_0,x_1,x_2,x_3,x_4)-2k'_0(x_5,x_6,x_7,x_8)=x_3+x_2+1-2x_5=0, \\ f_{1_{lin}}(x_0,x_1,x_2,x_3,x_4)-2k'_1(x_5,x_6,x_7,x_8)=x_4+x_0-2x_6=0, \\ f_{2_{lin}}(x_0,x_1,x_2,x_3,x_4)-2k'_2(x_5,x_6,x_7,x_8)=x_0+x_1+x_2+1-2x_7-2x_8=0. \end{cases} \tag{53}$$

Ostatni krok to wyznaczenie sumy kwadratów reszt, której wielomian kwadratowy dla przedstawianego przykładu jest postaci:
$Sum = \left(f_{0_{lin}}(x_0,x_1,x_2,x_3,x_4)-2k'_0(x_5,x_6,x_7,x_8)\right)^2 + \left(f_{1_{lin}}(x_0,x_1,x_2,x_3,x_4)+\right.$
$\left.-2k'_1(x_5,x_6,x_7,x_8)\right)^2 + \left(f_{2_{lin}}(x_0,x_1,x_2,x_3,x_4)-2k'_2(x_5,x_6,x_7,x_8)\right)^2 = 2x_0x_1+$
$+2x_0x_2+2x_1x_2+2x_2x_3+2x_0x_4-4x_2x_5-4x_3x_5-4x_0x_6-4x_4x_6-4x_0x_7-4x_1x_7-$
$4x_2x_7-4x_0x_8-4x_1x_8-4x_2x_8+8x_7x_8+4x_0+3x_1+6x_2+3x_3+x_4+4x_6+2.$
Teraz wielomian kary ma następującą postać: $Pen_{lin} = M \cdot \left(Pen_{lin_1}+Pen_{lin_2}\right) =$
$x_0x_1+x_1x_2-2x_0x_3-2x_1x_3-2x_1x_4-2x_2x_4+3x_3+3x_4$. Maksymalny współczynnik wielomianu sumy kwadratów wynosi 8, a maksymalny współczynnik wielomianu kary wynosi 3. Dlatego dla wagi kary M przyjęto wartość 2, stąd $Pen_{lin} = 2x_0x_1+$
$2x_1x_2-4x_0x_3-4x_1x_3-4x_1x_4-4x_2x_4+6x_3+6x_4$. Ostatecznie, otrzymano docelowy problem w postaci QUBO: $y=4x_0x_1+2x_0x_2+4x_1x_2-4x_0x_3-4x_1x_3+2x_2x_3+$
$2x_0x_4-4x_1x_4-4x_2x_4-4x_2x_5-4x_3x_5-4x_0x_6-4x_4x_6-4x_0x_7-4x_1x_7-4x_2x_7-$
$4x_0x_8-4x_1x_8-4x_2x_8+8x_7x_8+4x_0+3x_1+6x_2+9x_3+7x_4+4x_6$ oraz stałą $C=2$.

Minimalna energia otrzymanego problemu, która jest równa -2 (minimalna energia jest wartością przeciwną do stałej C), przyjmowana jest dla poprawnego rozwiązania problemu $\min_{x\in\{0,1\}^N} y$, dla $N = 9$. Istnieją dwa rozwiązania o minimalnej energii: $x_0 = 0$, $x_1 = 0$, $x_2 = 1$, $x_3 = 0$, $x_4 = 0$, $x_5 = 1$, $x_6 = 0$, $x_7 = 1$, $x_8 = 0$ oraz $x_0 = 0$, $x_1 = 0$, $x_2 = 1$, $x_3 = 0$, $x_4 = 0$, $x_5 = 1$, $x_6 = 0$, $x_7 = 0$, $x_8 = 1$. Można zauważyć, że oba rozwiązania zawierają to samo rozwiązanie układu (53), ponieważ dla obu rozwiązań, $x_7 = 1, x_8 = 0$ oraz $x_7 = 0, x_8 = 1$, wartość zmiennej $k_2 = x_7 + x_8 = 1$. Poszukiwanym rozwiązaniem wyjściowego układu (50) jest: $x_0 = 0, x_1 = 0, x_2 = 1$.

Wykorzystując przedstawioną wcześniej notację macierzową, zgodnie z równaniem (36), mamy:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & -2 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & -2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & -2 & -2 \end{bmatrix}$$

oraz

$$\mathbf{c} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}.$$

Zgodnie z równaniami (45) oraz (46), macierze **B**, **D** oraz **Q**, dla prezentowanego przykładu mają następującą postać:

$$\mathbf{B} = \begin{bmatrix} 4 & 2 & 2 & 0 & 2 & 0 & -4 & -4 & -4 \\ 0 & 3 & 2 & 0 & 0 & 0 & 0 & -4 & -4 \\ 0 & 0 & 6 & 2 & 0 & -4 & 0 & -4 & -4 \\ 0 & 0 & 0 & 3 & 0 & -4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{D} = \begin{bmatrix} 0 & 2 & 0 & -4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -4 & -4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{Q} = \begin{bmatrix} 4 & 4 & 2 & -4 & 2 & 0 & -4 & -4 & -4 \\ 0 & 3 & 4 & -4 & -4 & 0 & 0 & -4 & -4 \\ 0 & 0 & 6 & 2 & -4 & -4 & 0 & -4 & -4 \\ 0 & 0 & 0 & 9 & 0 & -4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 7 & 0 & -4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Stała C jest równa 2.

3.6 Wymagania dla układu równań wielomianowych o wielu zmiennych opisujących dany szyfr

Do tej pory, podczas rozwiązywania układów równań nad ciałami skończonymi, kluczowymi miarami trudności były liczba równań, liczba zmiennych oraz maksymalny stopień równań w układzie. Jednak w proponowanej metodzie miary te są niewystarczające, aby mówić o trudności rozwiązywanego problemu. Ważniejsza od liczby równań i ich maksymalnego stopnia jest ich postać oraz liczba jednomianów, z których się składają, i to zarówno jednomianów nieliniowych, jak i wszystkich jednomianów

w danym równaniu. Wszystko to wpływa na rozmiar docelowego problemu QUBO. Z drugiej strony, rozmiar ostatecznego problemu w postaci QUBO to nie wszystko, ponieważ należy również wziąć pod uwagę wymagania dotyczące charakteru funkcji celu, z którą będzie musiał poradzić sobie wyżarzacz kwantowy D-Wave.

Dlatego też, biorąc pod uwagę zaproponowany w niniejszej rozprawie model transformacji układu równań wielomianów wielu zmiennych, opisujących dany szyfr, do problemu QUBO oraz przedstawiony w poprzednim rozdziale opis procesu szukania rozwiązań za pomocą wyżarzacza kwantowego D-Wave, zaproponowano wymagania, które należy wziąć pod uwagę podczas generowania układu równań reprezentującego dany szyfr.

- Ze względu na fizyczną strukturę chipu QPU komputera kwantowego D-Wave:
  - struktura algebraiczna, nad którą zostaną wygenerowane równania wielomianowe powinna być tak dobrana, aby macierz $\mathbf{Q}$ otrzymanego problemu QUBO była rzadka, co spowoduje, że liczba oraz długość łańcuchów, utworzonych podczas osadzania grafu, będzie jak najmniejsza;
  - struktura algebraiczna oraz waga kary powinny być tak dobrane, aby różnica wag modelu Isinga, czyli różnica pomiędzy maksymalnym a minimalnym współczynnikiem w macierzy $\mathbf{Q}$, była tak mała jak to możliwe, żeby wartości po skalowaniu wag nie stały się nieznaczące.
- Ze względu na model transformacji układu równań wielomianowych, opisujących szyfr, do problemu QUBO:
  - liczba różnych jednomianów stopnia 2 powinna być jak najmniejsza, aby podczas linearyzacji wykorzystać jak najmniejszą liczbę dodatkowych zmiennych binarnych;
  - liczba bitów dla wartości k_i danego równania rośnie logarytmicznie, zatem jeśli to możliwe, powinny być stosowane dłuższe równania wielomianowe, zamiast dodatkowych równań w układzie, co powinno zmniejszyć całkowitą liczbę dodatkowych zmiennych binarnych dla przedstawienia wartości wielokrotności k_i wszystkich równań w układzie.

4 Analiza szyfru blokowego typu SPN na przykładzie standardu AES

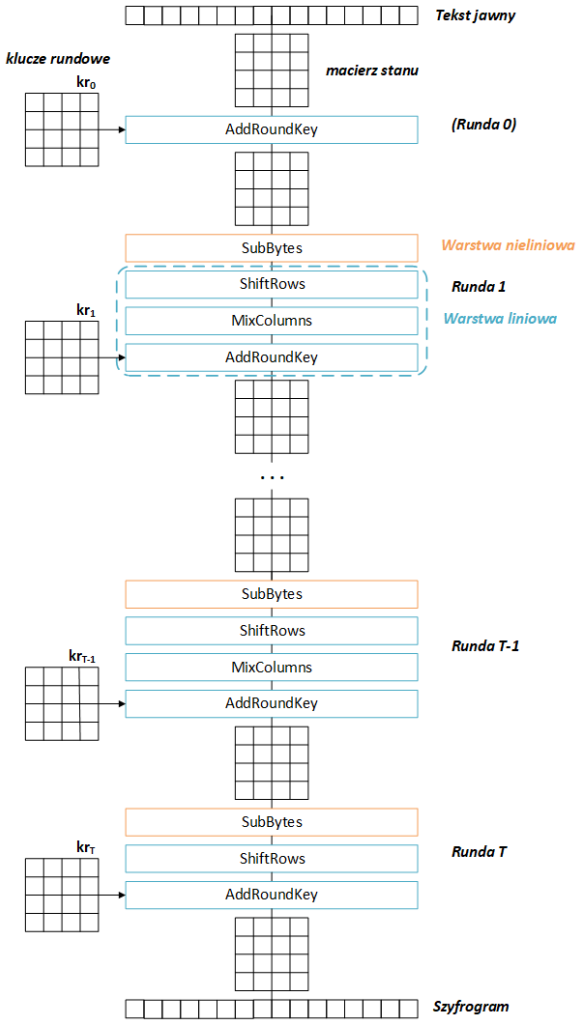
W 2001 r. szyfr blokowy Rijndael, przedstawiony w pracy [22], został wybrany w konkursie, ogłoszonym przez Narodowy Instytut Standardów i Technologii (NIST) Stanów Zjednoczonych jako Advanced Encryption Standard (AES) i zaproponowany jako standard w raporcie [32]. Szyfr ten został zaprojektowany tak, aby oprzeć się dobrze znanym technikom ataku na szyfry blokowe, a przede wszystkim kryptoanalizie liniowej i różnicowej, co pokazano w pracy [21]. Struktura Rijndaela jest, w przeciwieństwie do wielu innych szyfrów blokowych, takich jak DES, czysto algebraiczna. Dlatego też w kolejnych latach AES konsekwentnie był przedstawiany jako układ równań wielomianowych o wielu zmiennych nad ciałem $GF(2)$ [19] oraz nad $GF(2^8)$ [52], w celu znalezienia takiego układu, aby złożoność ataku algebraicznego była mniejsza, niż atak pełnego przeszukiwania.

Shannon zasugerował użycie do projektowania szyfrów dwóch różnych typów funkcji, jednej w celu uzyskania wymieszania (konfuzji), a drugiej w celu uzyskania rozproszenia (dyfuzji). Konfuzję zapewnia tak zwana warstwa podstawieniowa, a dyfuzję zapewnia warstwa dyfuzji. Połączenie tych dwóch typów operacji, wraz z wymieszaniem z kluczami rundowymi, pochodzącymi z klucza głównego, stanowi rundę sieci podstawieniowo – przestawieniowej (*SPN, ang. Substitution – Permutation Network*). Pojedyncze wystąpienie tych funkcji nie zapewnia wysokiego stopnia wymieszania i rozproszenia, ale, po odpowiednio wielokrotnym ich połączeniu w łańcuch, powstały szyfr iterowany może osiągnąć wysoki poziom wymieszania i rozproszenia, aż do osiągnięcia efektu lawinowości. Podejście to pozwala konstruować rundy, składające się z prostszych operacji, które mogą być analizowane osobno.

4.1 Algorytm szyfrowania standardu AES

Algorytm szyfrowania standardu AES jest kombinacją warstwy liniowej oraz nieliniowej. Na rysunku 31 przedstawiono ogólny schemat algorytmu szyfrowania standardu AES, gdzie kolorem niebieskim oznaczono funkcje liniowe, a kolorem pomarańczo-

wym – funkcję nieliniową. Wejściem do algorytmu szyfrowania jest 128-bitowy blok tekstu jawnego, który zgodnie ze specyfikacją [32] przedstawiany jest jako macierz kwadratowa o wymiarach 4×4 bajty. Macierz ta, jako stan pośredni, przetwarzana jest w każdym kroku algorytmu, by po ostatnim kroku zostać przedstawiona jako 128-bitowy szyfrogram. Szyfr składa się z T rund, gdzie liczba rund zależy od długości klucza i wynosi 10 rund dla klucza 128-bitowego, 12 rund dla klucza 192-bitowego oraz 14 rund dla klucza 256-bitowego. $T-1$ pierwszych rund składa się z czterech funkcji: SubBytes, ShiftRows, MixColumns oraz AddRoundKey. Ostatnia runda składa się tylko z trzech funkcji: SubBytes, ShiftRows oraz AddRoundKey. Dodatkowo przed pierwszą rundą, wykonywana jest funkcja AddRoundKey, nazywana rundą zerową lub rundą wstępną.



Rysunek 31: Ogólny schemat budowy algorytmu szyfrowania standardu AES. *Źródło: na podstawie [60]*

Funkcja SubBytes (funkcja podstawieniowa) jest jedyną nieliniową operacją szyfru AES, odwzorowującą dowolny element ciała $GF(2^8)$ na nową wartość. Odwzorowanie danego bajtu polega na wyznaczeniu jego inwersji w ciele $GF(2^8)$, przy czym 0 jest odwzorowywane na samo siebie, następnie przemnożeniu otrzymanej odwrotności przez ustaloną macierz binarną, rozmiaru 8×8, i dodaniu bajtu o wartości $0x63$. Funkcja ta może być reprezentowana jako tabela, odwzorowująca wartości stanów na inne wartości. Taka tabela nazywana jest skrzynką podstawieniową (S-box lub Sbox) i w przypadku AESa wejściem do skrzynki oraz jej wyjściem jest ośmiobitowy wektor. W tej reprezentacji, funkcja SubBytes polega na podstawieniu za każdy bajt z macierzy stanu, odpowiedniego bajtu ze skrzynki podstawieniowej. Cała warstwa nieliniowa w algorytmie szyfrowania standardu AES składa się z szesnastu instancji skrzynki podstawieniowej.

Funkcja ShiftRows jest permutacją, a więc funkcją liniową. Polega ona na cyklicznym przesunięciu w lewo bajtów wierszy macierzy stanu. Pierwszy wiersz nie jest przesuwany, drugi wiersz przesuwany jest o jeden bajt, trzeci wiersz – o dwa bajty, czwarty wiersz przesuwany jest o trzy bajty w lewo.

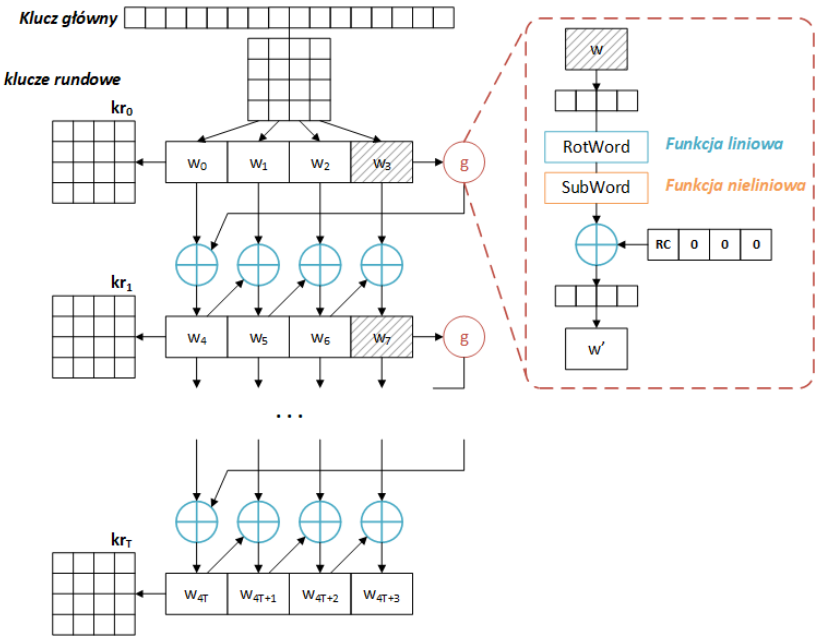
Funkcja MixColumns jest również funkcją liniową, przetwarzającą osobno każdą kolumnę macierzy stanu. Realizowana jest jako mnożenie macierzy wartości stałych przez kolejne kolumny macierzy stanu, w wyniku czego każdy bajt kolumny macierzy stanu odwzorowywany jest na nową wartość, która jest funkcją wszystkich czterech bajtów kolumny.

Funkcja AddRoundKey jest również funkcją liniową, realizowaną jako bitowa operacja xor danej macierzy stanu z macierzą reprezentującą dany klucz rundowy.

Te trzy funkcje liniowe w algorytmie szyfrowania standardu AES tworzą warstwę liniową, odpowiadającą za dyfuzję i wymieszanie z kluczem.

4.2 Algorytm generowania kluczy rundowych szyfru AES

Algorytm generowania kluczy rundowych szyfru AES, podobnie jak algorytm szyfrowania, składa się z funkcji liniowych oraz nieliniowych. Długość klucza głównego szyfru AES wynosi 128, 192 lub 256 bitów. Klucz główny, analogicznie do tekstu



Rysunek 32: Ogólny schemat budowy algorytmu szyfrowania standardu AES128. *Źródło: na podstawie [60]*

jawnego, przestawiany jest jako macierz bajtów, o wymiarze $4 \times N_k$, gdzie N_k zależy od długości klucza głównego i wynosi 4, 6 lub 8. Macierz ta jest przetwarzana, w celu wygenerowania $T+1$ 128-bitowych kluczy rundowych, również przedstawianych w postaci macierzy o wymiarze 4×4 bajty.

W przypadku klucza długości 128 bitów pierwszy klucz rundowy jest kopią pierwszych czterech czterobajtowych słów klucza głównego. Słowa kolejnych kluczy rundowych wyznaczane są na podstawie słów poprzedniego klucza rundowego poprzez wykonanie bitowej operacji xor $w_{j-1} \oplus w_{j-4}$ dla danego słowa w_j, przy czym co czwarte słowo w_{j-1}, gdy $j(mod\ 4) = 0$, przed wykonaniem operacji xor przetwarzane jest najpierw przez funkcję g, która składa się z funkcji RotWord, SubWord oraz operacji xor z wartością stałą. Ogólny schemat algorytmu generowania kluczy rundowych szyfru AES128 przedstawiono na rysunku 32.

Funkcja RotWord jest funkcją liniową, która polega na cyklicznym przesunięciu przetwarzanego słowa w o jeden bajt w lewo.

Funkcja SubWord jest funkcją nieliniową, która za każdy bajt słowa wejściowego podstawia nowy bajt, zgodnie ze skrzynką podstawieniową, tą samą co w algorytmie szyfrowania.

W ostatnim kroku funkcji g wykonywana jest bitowa operacja xor z wartością stałą $(RC[i], 0, 0, 0)$, której trzy najmniej znaczące bajty mają wartość 0, a wartość najbardziej znaczącego bajtu zależy od numeru rundy, oznaczonego jako i, w taki sposób, że dla $i = 1$ stała $RC[1] = 1$, a dla kolejnych rund $RC[i] = 2 \cdot RC[i-1]$, gdzie operacja mnożenia wykonywana jest nad ciałem $GF(2^8)$.

W przypadku standardu AES z kluczem długości 192 bity, algorytm generowania kluczy różni się od wyżej opisanego tym, że pierwszy klucz rundowy kr_0 oraz dwa pierwsze 32-bitowe słowa drugiego klucza rundowego kr_1 są kopią klucza głównego. Dodatkowo, za pomocą funkcji g przetwarzane jest słowo w_{j-1}, gdy $j(mod\ 6) = 0$.

W przypadku wersji szyfru AES z długością klucza równą 256 bitów, algorytm generowania kluczy rundowych różni się jeszcze bardziej od wersji AES128, niż wersja AES192. Poza analogicznymi różnicami takimi, że kopią klucza głównego są dwa pierwsze klucze rundowe (kr_0 i kr_1) oraz za pomocą funkcji g przetwarzane jest słowo w_{j-1}, gdy $j(mod\ 8) = 0$, dodatkowo jeszcze, gdy $j(mod\ 8) = 4$, słowo w_{j-1} przetwarzane jest za pomocą samej funkcji SubWord i dopiero xorowane ze słowem w_{j-8}, dając w wyniku słowo w_j.

4.3 Analiza przedstawienia szyfru AES za pomocą układu równań wielomianowych nad ciałem $GF(2)$

Niniejszy rozdział przedstawia analizę konstruowania układu równań wielomianowych, spełniającego zaproponowane w podrozdziale 3.6 wymagania oraz opisującego szyfr AES. Strukturą algebraiczną, nad którą zostaną wygenerowane równania, jest ciało $GF(2)$.

Ponieważ konstrukcja rundy szyfru AES pozwala analizować każdą jej operację z osobna, przeprowadzona zostanie osobna analiza wyznaczania równań dla warstwy nieliniowej algorytmu szyfrowania oraz osobna dla warstwy liniowej.

4.3.1 Wyznaczanie równań opisujących skrzynkę podstawieniową standardu AES

Niech teraz s oznacza liczbę bitów wektorów wejściowych i wyjściowych dowolnej, odwracalnej skrzynki podstawieniowej. Wtedy taką skrzynkę podstawieniową można przedstawić jako bijekcję: $Sbox : \mathbb{F}_{2^s} \rightarrow \mathbb{F}_{2^s}$. Następnie niech $(x_0, \ldots, x_{s-1})$ oznacza wektor wejściowy do skrzynki podstawieniowej, gdzie bit x_0 jest najbardziej znaczącym bitem wartości wejściowej oraz, analogicznie, niech $(y_0, \ldots, y_{s-1})$ oznacza wektor wyjściowy. Skrzynkę podstawieniową można przedstawić za pomocą dwóch typów równości:

- jawnych (*ang. explicit*), postaci: $y_j = g_j(x_0, x_1, ..., x_{s-1})$, dla $j = \overline{0, s-1}$; ponieważ działania wykonywane są w ciele binarnym, równania te są w algebraicznej postaci normalnej,
- uwikłanych (*ang. implicit*), postaci: $f(x_0, ..., x_{s-1}, y_0, ..., y_{s-1}) = 0$, gdzie $Sbox(x_0, ..., x_{s-1}) = y_0, ..., y_{s-1} \Rightarrow f(x_0, ..., x_{s-1}, y_0, ..., y_{s-1}) = 0$.

Jeśli skrzynka podstawieniowa jest zaprojektowana poprawnie, to każde równanie bitu wyjściowego w algebraicznej postaci normalnej ma wysoki stopień algebraiczny, maksymalnie równy $s-1$. Dlatego też przedstawienie warstwy nieliniowej za pomocą tego typu równań jest nieefektywne w kontekście ich dalszej linearyzacji. Przykładowo, dla skrzynki podstawieniowej szyfru AES, gdzie $s = 8$, stopień równań, reprezentujących bity wektora wyjściowego, wynosi 7. Co więcej, równania te składają się z od 106 do 141 jednomianów nieliniowych. Sama operacja linearyzacji tych ośmiu równań wymagałaby 246 dodatkowych zmiennych binarnych, co przekłada się na 3 936 dodatkowych zmiennych binarnych dla warstwy nieliniowej pojedynczej rundy algorytmu szyfrowania oraz na 39 360 dodatkowych zmiennych binarnych dla całego algorytmu szyfrowania.

W pracy [19] Courtois oraz Pieprzyk pokazali, jak zbudować nadokreślony układ równań stopnia 2, m.in. dla szyfru Rijndael. Metoda tam przedstawiona pozwala konstruować równania niejawne, zawierające tylko jednomiany z wcześniej zdefiniowanego zbioru akceptowalnych jednomianów.

Niech dany będzie zbiór jednomianów stopnia co najwyżej dwa, których zmienne należą do zbioru $\{x_0, x_1, ..., x_{s-1}, y_0, y_1, ..., y_{s-1}\}$. Dla skrzynki podstawieniowej o rozmiarze s, liczba wszystkich możliwych wejść do skrzynki wynosi 2^s, liczba wszystkich możliwych jednomianów wynosi 2^{2s}, liczba wszystkich możliwych jednomianów stopnia dwa wynosi $\binom{2s}{2}$ oraz liczba wszystkich możliwych jednomianów stopnia co najwyżej dwa wynosi $\binom{2s}{2} + 2s + 1$. W celu wyznaczenia równań wielomianowych stopnia co najwyżej 2, opisujących skrzynkę podstawieniową, konstruowana jest macierz o wymiarze $\left(\binom{2s}{2} + 2s + 1\right) \times 2^s$, w której każdy wiersz zawiera wartości danego jednomianu dla wszystkich możliwych wejść. Następnie wykonywana jest eliminacja Gaussa, podczas której zapamiętywane są operacje na wierszach. Co najmniej $\binom{2s}{2} + 2s + 1 - 2^s$ wierszy zostanie wyzerowanych, co oznacza, że istnieje co najmniej $\binom{2s}{2} + 2s + 1 - 2^s$ równań kwadratowych spełniających skrzynkę podstawieniową. W ten sposób otrzymywane są wszystkie równania kwadratowe, spełniające daną skrzynkę podstawieniową.

Dla przykładu, wygenerujmy równania wielomianowe stopnia co najwyżej 2, opisujące skrzynkę podstawieniową szyfru S-AES (*ang. Simplified AES*), której rozmiar wynosi $s = 4$. Skrzynka ta jest zdefiniowana za pomocą następującej permutacji:

$$S = (9, 4, 10, 11, 13, 1, 8, 5, 6, 2, 0, 3, 12, 14, 15, 7).$$

Można również tę skrzynkę przedstawić jako tabelę prawdy dla wszystkich bitów wyjściowych, w zależności od wartości bitów wejściowych, co przedstawiono w tabeli 3. Liczba wszystkich możliwych wektorów wejściowych do skrzynki podstawieniowej algorytmu S-AES wynosi 16, liczba wszystkich możliwych jednomianów wynosi 256, liczba wszystkich możliwych jednomianów stopnia dwa wynosi 28, a liczba wszystkich możliwych jednomianów stopnia co najwyżej dwa wynosi 37. Na podstawie tabeli prawdy konstruowana jest macierz, o wymiarach 37×16, gdzie pierwszy wiersz jest wierszem samych jedynek, reprezentującym jednomian stopnia zero, wiersze dla pojedynczych zmiennych są odpowiednimi kolumnami z tabeli 3, a pozostałe wiersze, czyli wartości jednomianów kwadratowych, wyznaczane są na podstawie wartości zmiennych, z których składa się dany jednomian. Macierz ta została przedstawiona w tabeli 4.

Tabela 3: Tabela prawdy dla skrzynki podstawieniowej szyfru S-AES.

x_0	x_1	x_2	x_3	y_0	y_1	y_2	y_3
0	0	0	0	1	0	0	1
0	0	0	1	0	1	0	0
0	0	1	0	1	0	1	0
0	0	1	1	1	0	1	1
0	1	0	0	1	1	0	1
0	1	0	1	0	0	0	1
0	1	1	0	1	0	0	0
0	1	1	1	0	1	0	1
1	0	0	0	0	1	1	0
1	0	0	1	0	0	1	0
1	0	1	0	0	0	0	0
1	0	1	1	0	0	1	1
1	1	0	0	1	1	0	0
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	1	1	1

Przykładowo, na podstawie tabeli 3 wiemy, że dla wektora wejściowego $(0,0,1,1)$, wektor wyjściowy ma wartość $(1,0,1,1)$ (wiersz oznaczony kolorem zielonym). Oznacza to, że bity x_2, x_3, y_0, y_2 i y_3 mają wartość 1, a pozostałe bity mają wartość 0. Ta para wektorów wejścia/wyjścia jest piątą kolumną w tabeli 4, co oznaczono kolorem zielonym. Na podstawie wartości tych bitów uzupełniana jest pozostała część kolumny, dla jednomianów kwadratowych. Przykładowo, wartość jednomianu $x_2y_3 = 1$, bo obie zmienne w tej kolumnie mają wartość 1 (bity oznaczone kolorem czerwonym), a wartość jednomianu $x_1y_2 = 0$, bo zmienna x_1 ma wartość 0, a zmienna y_2 ma wartość 1 (bity oznaczone kolorem niebieskim).

W wyniku wykonania eliminacji Gaussa, co najmniej $\binom{8}{2} + 8 + 1 - 16 = 21$ wierszy zostanie wyzerowanych. Kolejne kroki eliminacji Gaussa wyznaczają kolejne równania spełniające skrzynkę podstawieniową. Przykładowo, wiersz dla jednomianu x_3y_2 zostaje wyzerowany poprzez dodanie do siebie wierszy następujących jednomianów: x_0y_0, x_0, x_3, y_0 oraz 1 (wiersze oznaczone kolorem szarym). Oznacza to, że skrzynkę podstawieniową szyfru S-AES zawsze spełnia równanie: $x_3y_2 + x_0y_0 + x_0 + x_3 + y_0 + 1 = 0$.

Tabela 4: Wartości jednomianów stopnia co najwyżej dwa dla wszystkich możliwych wejść skrzynki podstawieniowej szyfru S-AES.

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
x_3	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
x_2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
y_3	1	0	0	1	1	1	0	1	0	0	0	1	0	0	1	1
y_2	0	0	1	1	0	0	0	0	1	1	0	1	0	1	1	1
y_1	0	1	0	0	1	0	0	1	1	0	0	0	1	1	1	1
y_0	1	0	1	1	1	0	1	0	0	0	0	0	1	1	1	0
x_3x_2	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
x_3x_1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1
x_3x_0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1
x_2x_1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1
x_2x_0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1
x_1x_0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
x_3y_3	0	0	0	1	0	1	0	1	0	0	0	1	0	0	0	1
x_3y_2	0	0	0	1	0	0	0	0	0	1	0	1	0	1	0	1
x_3y_1	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	1
x_3y_0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
x_2y_3	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1
x_2y_2	0	0	1	1	0	0	0	0	0	0	0	1	0	0	1	1
x_2y_1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1
x_2y_0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	1	0
x_1y_3	0	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1
x_1y_2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
x_1y_1	0	0	0	0	1	0	0	1	0	0	0	0	1	1	1	1
x_1y_0	0	0	0	0	1	0	1	0	0	0	0	0	1	1	1	0
x_0y_3	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1
x_0y_2	0	0	0	0	0	0	0	0	1	1	0	1	0	1	1	1
x_0y_1	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1
x_0y_0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
y_3y_2	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	1
y_3y_1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	1
y_3y_0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0
y_2y_1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1
y_2y_0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1	0
y_1y_0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1	0

W wyniku przeprowadzenia eliminacji Gaussa otrzymano 21 wielomianów kwadratowych, spełniających skrzynkę podstawieniową szyfru S-AES. Otrzymany układ

wielomianów ma następującą postać:

$$\begin{cases}
f_0 = x_0x_1 + x_0x_3 + x_0y_2 + x_0 + x_1x_2 + x_1 + y_0 + y_1 + 1, \\
f_1 = x_0x_1 + x_0x_2 + x_0y_0 + x_0y_1 + x_0y_2 + x_0 + x_1 + x_2 + y_0 + y_1 + y_2 + 1, \\
f_2 = x_0x_1 + x_0x_2 + x_0x_3 + x_0y_0 + x_0y_1 + x_0y_3 + x_0 + x_1x_3 + x_1y_0 + x_1, \\
f_3 = x_0x_1 + x_0x_2 + x_0x_3 + x_0y_0 + x_0y_3 + x_1x_3 + x_1y_1 + y_0 + y_1 + 1, \\
f_4 = x_0x_1 + x_0x_2 + x_0y_2 + x_0y_3 + x_0 + x_1y_2, \\
f_5 = x_0x_3 + x_0y_1 + x_0y_2 + x_0 + x_1x_3 + x_1y_3 + x_2 + x_3 + y_0 + y_1 + y_3, \\
f_6 = x_0x_1 + x_0y_0 + x_0y_2 + x_0y_3 + x_0 + x_1x_3 + x_2x_3 + x_2 + y_0 + y_3, \\
f_7 = x_0x_1 + x_0x_2 + x_0x_3 + x_0y_2 + x_1x_3 + x_2y_0 + x_3 + y_3 + 1, \\
f_8 = x_0x_3 + x_0y_0 + x_0y_2 + x_1x_3 + x_2y_1 + x_2 + x_3 + y_3 + 1, \\
f_9 = x_0x_1 + x_0x_2 + x_0y_0 + x_0y_1 + x_0y_3 + x_0 + x_1 + x_2y_2 + x_2 + y_0 + y_1 + 1, \\
f_{10} = x_0x_3 + x_0y_0 + x_0y_1 + x_0y_3 + x_0 + x_1x_3 + x_2y_3 + x_2 + y_0 + y_3, \\
f_{11} = x_0x_1 + x_0x_2 + x_0y_1 + x_0y_3 + x_3y_0 + x_3 + y_0 + 1, \\
f_{12} = x_0y_0 + x_0y_1 + x_0 + x_2 + x_3y_1 + x_3 + y_0 + y_3, \\
f_{13} = x_0y_0 + x_0 + x_3y_2 + x_3 + y_0 + 1, \\
f_{14} = x_0x_1 + x_0x_2 + x_0x_3 + x_0y_0 + x_0y_2 + x_1x_3 + x_3y_3 + x_3 + y_0 + 1, \\
f_{15} = x_0x_2 + x_0x_3 + x_0y_1 + x_0y_3 + x_2 + x_3 + y_0y_1 + y_0 + y_1 + y_3, \\
f_{16} = x_0x_1 + x_0y_1 + x_0y_2 + x_0y_3 + x_1 + x_2 + y_0y_2 + y_0 + y_1 + 1, \\
f_{17} = x_0x_1 + x_0x_2 + x_0y_2 + x_0 + x_1x_3 + y_0y_3 + y_3, \\
f_{18} = x_0x_2 + x_0y_1 + x_0y_2 + x_0y_3 + x_0 + y_1y_2, \\
f_{19} = x_0x_2 + x_0y_1 + x_0y_2 + x_0y_3 + x_1x_3 + y_0 + y_1y_3 + y_1 + 1, \\
f_{20} = x_0x_3 + x_0y_0 + x_0y_3 + x_0 + x_3 + y_0 + y_2y_3 + 1.
\end{cases} \tag{54}$$

W przypadku skrzynki podstawieniowej szyfru AES, gdzie $s = 8$, liczba wszystkich możliwych wektorów wejściowych do skrzynki wynosi 256, liczba wszystkich możliwych jednomianów stopnia 2 wynosi $\binom{16}{2} = 120$, a liczba wszystkich możliwych jednomianów stopnia co najwyżej dwa wynosi $\binom{16}{2} + 16 + 1 = 137$. W celu znalezienia równań wielomianów kwadratowych spełniających skrzynkę podstawieniową szyfru AES skonstruowano macierz o wymiarach 137×256. Liczba wierszy jest mniejsza niż liczba kolumn, więc poszukiwane równania mogą nie istnieć, bo $\binom{16}{2} + 16 + 1 - 256 = -119 < 0$. Jednak w wyniku wykonania eliminacji Gaussa otrzymano 39 równań wielomianów kwadratowych, zawsze spełniających skrzynkę podstawieniową szyfru AES. Równania wielomianów kwadratowych powstałego w ten spo-

sób układu mają 120 różnych jednomianów kwadratowych i każde równanie składa się z od 20 do 50 jednomianów. W całym układzie występuje 1337 wszystkich jednomianów. Sama linearyzacja takiego układu, opisującego pojedynczą skrzynkę podstawieniową szyfru AES, wymaga 120 dodatkowych zmiennych binarnych, co przekłada się na 1 920 dodatkowych zmiennych dla całej warstwy nieliniowej pojedynczej rundy oraz na 19 200 dla warstwy nieliniowej algorytmu szyfrowania standardu AES128.

Ponieważ zastosowana metoda wyznaczania równań kwadratowych, spełniających skrzynkę podstawieniową została opracowana do konstruowania układów nadokreślonych, pojawia się wątpliwość, czy wszystkie równania są konieczne, aby jednoznacznie wyznaczyć skrzynkę podstawieniową. Przy założeniu, że układ taki musi składać się z co najmniej 8 równań, $s = 8$, to do sprawdzenia pozostaje $2^{39} - \sum_{i=0}^{7} \binom{39}{i} \approx 5,5 \cdot 10^{11}$ możliwych układów. Problem znalezienia efektywnego układu opisującego skrzynkę podstawieniową można przedstawić jako problem optymalizacyjny, gdzie przestrzenią rozwiązań jest zbiór wszystkich układów równań kwadratowych jednoznacznie definiujących skrzynkę podstawieniową, a funkcja celu przyporządkowuje każdemu układowi liczbę dodatkowych zmiennych binarnych, potrzebnych do wykonania jego transformacji do problemu QUBO. Ponieważ wyznaczono równania kwadratowe, proces linearyzacji polega na zastąpieniu każdego jednomianu kwadratowego nową zmienną binarną. Stąd liczba dodatkowych zmiennych binarnych wymaganych podczas linearyzacji jest równa liczbie różnych jednomianów kwadratowych w układzie. W tym przypadku proces linearyzacji nie jest problemem NP-trudnym. Dodatkowo, ponieważ współczynniki jednomianów w wyznaczonych równaniach mają wartość jeden, maksymalna wartość danego wielomianu jest równa liczbie wszystkich jego jednomianów. Stąd liczba dodatkowych zmiennych binarnych, potrzebnych do wyznaczenia wartości wielokrotności k_i każdego równania, zależy od liczby wszystkich jednomianów, z których się składa. W związku z tym, rozwiązaniem minimalnym tak zdefiniowanego problemu jest układ posiadający następujące własności:

- ze względu na późniejszą linearyzację, liczba różnych jednomianów kwadratowych w układzie jest możliwie najmniejsza,

- ze względu na późniejsze wyznaczanie wartości wielokrotności k_i, układ składa się w większości z bardzo krótkich równań i kilku dłuższych, przy czym idealna sytuacja występuje wtedy, gdy układ składa się z jednego równania, posiadającego dużą liczbę jednomianów, i z pozostałych równań, posiadających po dwa jednomiany każdy.

Zastanówmy się, kiedy układ jednoznacznie wyznacza skrzynkę podstawieniową. Niech F oznacza zbiór wszystkich równań wielomianów kwadratowych, spełniających skrzynkę podstawieniową, uzyskanych podczas wykonywania eliminacji Gaussa oraz niech t oznacza liczność zbioru F. Elementami zbioru F są równania kwadratowe postaci $f_i(x_0, ..., x_{s-1}, y_0, ..., y_{s-1}) = 0$, gdzie $i = \overline{0, t-1}$. Skrzynka podstawieniowa jest jednoznacznie określona przez układ równań, o ile dla każdego możliwego wejścia wszystkie równania są spełnione dokładnie dla jednego wyjścia. Można to przedstawić za pomocą dwuwymiarowych tabel, takich jak tabela 5, która zawiera wartości wielomianów f_i skrzynki podstawieniowej szyfru S-AES dla wejścia równego zero, czyli gdy $x_0 = 0$, $x_1 = 0$, $x_2 = 0$ oraz $x_3 = 0$. Kolorem szarym oznaczono jedyną kolumnę, która zawiera same zera. Kolumna ta odpowiada wyjściu równemu 9 ($y_0 = 1$, $y_1 = 0$, $y_2 = 0$, $y_3 = 1$), co oznacza, że dla wejścia równego 0 układ jest spełniony tylko dla wyjścia równego 9. W analogiczny sposób, tworząc pozostałe 15 tabel, można zauważyć, że w każdej z nich będzie istniała dokładnie jedna kolumna samych zer. Stąd można wyciągnąć następujący wniosek.

Wniosek: *Jeśli w każdej z 2^s tabel, zawierających wartości pewnego podzbioru wielomianów ze zbioru F, dla danego wejścia występuje tylko jedna kolumna samych zer, gdzie numer kolumny zer zgadza się z permutacją definiującą skrzynkę podstawieniową, to układ jednoznacznie wyznacza skrzynkę podstawieniową.*

Niech F' oznacza pewien podzbiór wielomianów, tworzących układ jednoznacznie wyznaczający skrzynkę podstawieniową, gdzie $F' \subseteq F$ oraz niech t' oznacza liczność zbioru F'. W celu znalezienia elementów zbioru F' dla zadanej wartości t' wykonano pełne przeszukanie, polegające na tym, że dla każdego z $\binom{t}{t'}$ podzbiorów zbioru F sprawdzono, czy nowy układ jednoznacznie określa skrzynkę podstawieniową. Innymi słowy, czy jeśli z każdej z 2^s tabel wybierzemy t' danych wierszy, to czy nadal

każda tabela będzie posiadała dokładnie jedną kolumnę samych zer. Jeśli dla wszystkich możliwych wejść wszystkie wielomiany danego podzbioru przyjmują wartość 0 tylko dla jednego wyjścia, to, o ile wystąpiły wszystkie możliwe wektory wyjściowe i każdy tylko raz, układ składający się z wielomianów danego podzbioru jednoznacznie wyznacza skrzynkę podstawieniową.

Tabela 5: Tabela wartości wszystkich równań wielomianów, spełniających skrzynkę podstawieniową szyfru S-AES, dla zerowego wejścia.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
f_0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1
f_1	1	1	0	0	0	0	1	1	0	0	1	1	1	1	0	0
f_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f_3	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1
f_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f_5	0	1	0	1	1	0	1	0	1	0	1	0	0	1	0	1
f_6	0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	0
f_7	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
f_8	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
f_9	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1
f_{10}	0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	0
f_{11}	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
f_{12}	0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	0
f_{13}	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
f_{14}	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
f_{15}	0	1	0	1	1	0	1	0	1	0	1	0	1	0	1	0
f_{16}	1	1	1	1	0	0	0	0	0	0	1	1	1	1	0	0
f_{17}	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0
f_{18}	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1
f_{19}	1	1	1	1	0	1	0	1	0	0	0	0	1	0	1	0
f_{20}	1	1	1	0	1	1	1	0	0	0	0	1	0	0	0	1

Jak już pokazano wcześniej, przestrzeń przeszukiwania jest ogromna. Jednak, ponieważ oczekuje się, że liczba wielomianów efektywnego układu t' będzie bliska rozmiarowi skrzynki: $t' \approx s$, maksymalną liczbę wielomianów w układzie ustalono na 14. W tabeli 6 przedstawiono liczbę wszystkich układów jednoznacznie wyznaczających skrzynkę podstawieniową szyfru AES dla zadanej liczby równań (wielomianów) w układzie. Układy te stanowią podzbiór zbioru rozwiązań problemu znalezienia efektywnego układu spełniającego skrzynkę podstawieniową szyfru AES. W dalszych podrozdziałach, w których opisano opracowane metody poszukiwania układu efektywnego, układy z tabeli 6 będą nazywane układami początkowymi.

Tabela 6: Liczba wszystkich układów spełniających skrzynkę podstawieniową szyfru AES, dla zadanej liczby równań w układzie.

Liczba równań w układzie	*Liczba układów*
< 12	0
12	1 052
13	2 690 682
14	227 550 310

4.3.2 Poszukiwanie efektywnego układu spełniającego skrzynkę podstawieniową szyfru AES - Metoda I

W pierwszej metodzie, proces poszukiwania efektywnego układu zrealizowano w dwóch etapach. W etapie pierwszym skupiono się na liczbie różnych jednomianów kwadratowych, która jest równa liczbie dodatkowych zmiennych binarnych potrzebnych w procesie linearyzacji. Dlatego też, spośród wszystkich układów początkowych (tabela 6), dla danej liczby równań w układzie, wybrano tylko te układy, które posiadają najmniejszą liczbę różnych jednomianów kwadratowych. W tabeli 7, dla danej liczby równań, przedstawiono liczności otrzymanych zbiorów układów, mających najmniejszą liczbę różnych jednomianów stopnia 2.

Tabela 7: Liczba układów spełniających skrzynkę podstawieniową szyfru AES, posiadających najmniejszą liczbę różnych jednomianów kwadratowych.

Liczba równań w układzie	*Liczba układów*	*Liczba różnych jednomianów kwadratowych*	*Liczba wszystkich jednomianów w układzie*
12	1	64	347
13	667	54	od 352 do 406
14	3	54	415, 416, 417

Spośród otrzymanych układów, według tabeli 7, do drugiego etapu poszukiwań wybrano układy składające się z 13 równań, ponieważ mają o 10 różnych jednomianów kwadratowych mniej, niż układy składające się z 12 równań, więc na pewno podczas linearyzacji wymaganych będzie o 10 mniej dodatkowych zmiennych binarnych. Liczba wszystkich jednomianów w układzie, która jest większa dla układów 13 rów-

nań, niż dla układów 12 równań jest mniej istotna, ponieważ jednomiany te rozkładają się na wszystkie równania układu i wpływają na ich długość, a co za tym idzie wpływają na liczbę dodatkowych zmiennych binarnych, potrzebnych przy wyznaczaniu wielokrotności k_i, a liczba ta rośnie logarytmicznie.

W drugim etapie skupiono się na liczbie wszystkich jednomianów w poszczególnych równaniach danego układu, która wpływa na liczbę dodatkowych zmiennych binarnych potrzebnych do wyznaczenia wartości wielokrotności k_i. Etap drugi polega na sprawdzeniu, czy możliwe jest zastąpienie danego równania w układzie równaniem o mniejszej liczbie jednomianów, powstałym poprzez wykonanie operacji xor na pewnej liczbie równań tego układu. Niech F' oznacza zbiór równań wielomianów analizowanego układu oraz niech G oznacza podzbiór wielomianów zbioru F' taki, że $|G| = r$, $r \in \{2, \ldots, |F'|-1\}$ oraz $G = \{g_i(x_0, \ldots, x_{s-1}, y_0, \ldots, y_{s-1}) : g_i \in F' \wedge (i \neq j \Rightarrow g_i \neq g_j)\}$, dla $i = \overline{0, r-1}$. Jeżeli liczba jednomianów wielomianu $h = g_0 \oplus g_1 \oplus \cdots \oplus g_{r-1}$ jest mniejsza, niż liczba jednomianów któregokolwiek z wielomianów $g_0, \ldots g_{r-1}$, to ten wielomian zastępowany jest wielomianem h. Podstawienie to zachowuje jednoznaczność wyznaczania skrzynki podstawieniowej. Załóżmy, że dla danej pary wejścia/wyjścia wszystkie wielomiany g_i, dla $i = \overline{0, r-1}$, mają wartość 0. Wtedy podstawienie $h = 0$ za którykolwiek z wielomianów g_i nie wpływa na spełnialność układu, ponieważ zależy ona od pozostałych wielomianów zbioru F'. Jeśli co najmniej jeden wielomian g_i ma wartość 1, to niezależnie od wartości pozostałych wielomianów zbioru F', dla tej pary wejścia/wyjścia układ nie powinien być spełniony. W związku z tym rozpatrzmy dwa przypadki:

1. Jeśli wielomian h ma wartość 1, tzn. że nieparzysta liczba wielomianów g_i ze zbioru G ma wartość 1, to zastąpienie któregokolwiek z wielomianów $g_i = 1$ albo $g_i = 0$ wielomianem $h = 1$ nie zmienia decyzji o odrzuceniu pary. Co najwyżej zwiększy się o jeden liczba wielomianów niespełniających skrzynkę.

2. Jeśli wielomian h ma wartość 0, co znaczy że parzysta liczba wielomianów g_i ze zbioru G ma wartość 1, to zastąpienie któregokolwiek z wielomianów $g_i = 1$ albo $g_i = 0$ wielomianem $h = 0$ nie zmienia decyzji o odrzuceniu pary, ponieważ ist-

nieje jeszcze co najmniej jeden wielomian niespełniający skrzynki podstawieniowej dla tej pary wejścia/wyjścia.

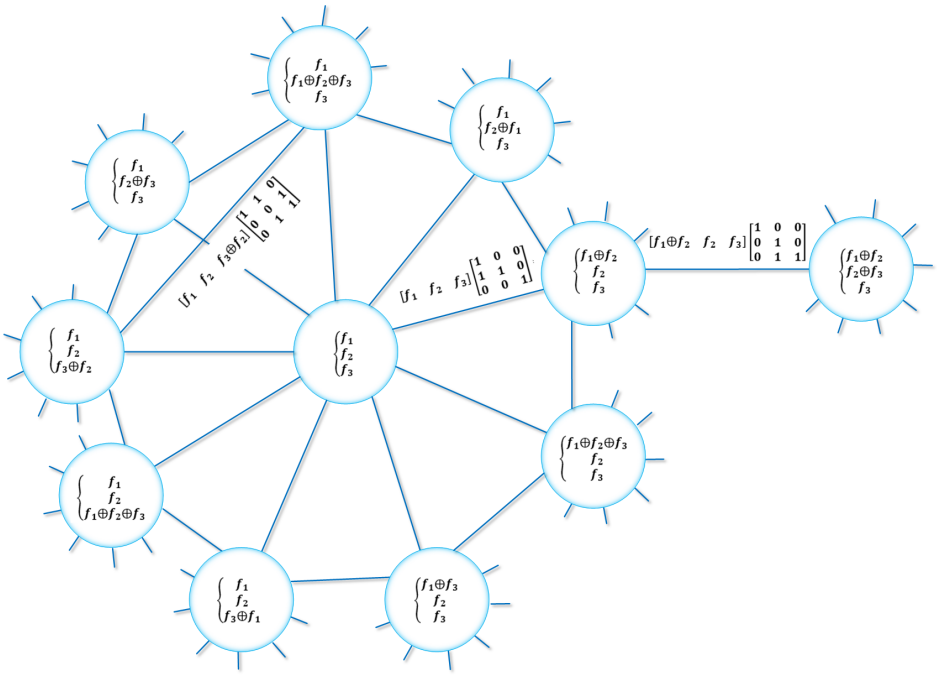
Co więcej, operacja ta nie zmienia liczby równań układu oraz nie zmienia liczby różnych jednomianów kwadratowych w układzie; zmienia jednak liczbę jednomianów w poszczególnych równaniach, a w konsekwencji liczbę wszystkich jednomianów w całym układzie. Operację tę przeprowadzono dla wszystkich 667 układów trzynastu równań, za każdym razem xorując wszystkie kombinacje dwóch, trzech, czterech oraz pięciu wielomianów. Proces powtarzano, dopóki występował zysk redukcji, czyli dopóki wynikiem operacji xor był wielomian, składający się z liczby jednomianów mniejszej, niż liczba jednomianów któregokolwiek z xorowanych wielomianów. Spośród otrzymanych układów równań kwadratowych, wybrano układ z najmniejszą liczbą wszystkich jednomianów. Wybrany układ przed zastosowaniem redukcji składał się z 378 jednomianów, natomiast po redukcji miał 268 wszystkich jednomianów. W kontekście dalszych przekształceń do problemu optymalizacyjnego w postaci QUBO, układ ten wymagał 109 dodatkowych zmiennych binarnych, natomiast po redukcji wymagał 106 dodatkowych zmiennych.

Przeprowadzono redukcję również dla układu składającego się z 12 równań, uzyskując zmniejszenie liczby wszystkich jednomianów w układzie z 347 do 277. Ostatecznie, liczba dodatkowych zmiennych binarnych zmniejszyła się ze 115 do 113, co potwierdza wcześniej wyciągnięty wniosek, że układ ten jest nieefektywny, w kontekście transformacji do problemu QUBO.

Omówiona metoda poszukiwania efektywnego układu, spełniającego skrzynkę podstawieniową szyfru AES, została zaprezentowana w pracy [12], wraz z uzyskanymi na jej podstawie wynikami.

4.3.3 Poszukiwanie efektywnego układu spełniającego skrzynkę podstawieniową szyfru AES – Metoda II

W pierwszej metodzie podczas wyboru spośród układów początkowych tylko tych układów, które posiadają najmniejszą liczbę różnych jednomianów kwadratowych, zawężana jest przestrzeń przeszukiwania. Dlatego opracowano kolejną metodę, która



Rysunek 33: Fragment grafu przedstawiający proces poszukiwania, dla układu składającego się z trzech równań.

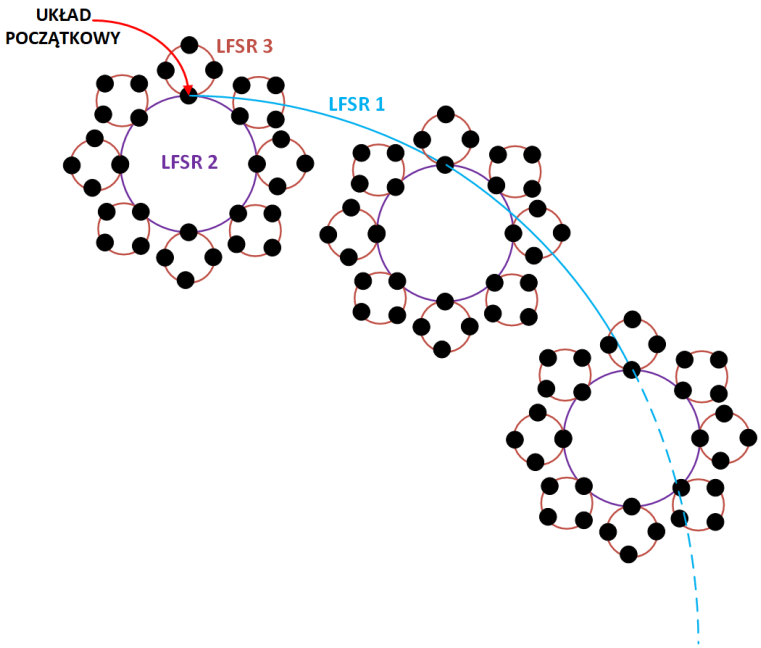
działa na przestrzeni wszystkich układów początkowych. Druga metoda jest analogią do pełnego przeszukiwania i polega na xorowaniu pewnej liczby równań danego układu początkowego, spełniającego skrzynkę podstawieniową szyfru AES, niezależnie od liczby różnych jednomianów kwadratowych. Za miarę efektywności danego układu przyjęto liczbę zmiennych binarnych, potrzebnych do przedstawienia wartości wielokrotności k_i dla wszystkich równań w analizowanym układzie. Przestrzeń takiego przeszukiwania dla każdego początkowego układu można wyobrazić sobie jako kompletny graf, którego wierzchołki to układy, powstałe po wykonaniu operacji xor, na pewnej liczbie równań układu początkowego. Przedstawiając układ jako wektor, gdzie elementami wektora są równania układu, pojedynczą operację xorowania równań danego układu można utożsamić z pomnożeniem wektora przez odwracalną macierz binarną. Stąd liczba wszystkich możliwych układów, wygenerowanych na podstawie danego układu początkowego za pomocą operacji xor, jest równa liczbie wszystkich odwracalnych macierzy binarnych podzielonej przez liczbę permutacji równań układu. Dlatego, jeśli układ składa się z t' równań, to rozmiar przestrzeni przeszukiwania, czyli liczba wierzchołków w grafie, wynosi $\frac{\prod_{i=0}^{t'-1}\left(2^{t'}-2^{i}\right)}{t'!}$. Na rysunku 33

przedstawiono fragment grafu, będącego przestrzenią przeszukiwania dla dowolnego układu składającego się z trzech równań. Pełny graf składa się z 28 wierzchołków. Na tym rysunku przedstawiono również przykładowe przejścia pomiędzy wierzchołkami, realizowane jako mnożenie wektora przez odwracalną macierz binarną. W przypadku skrzynki podstawieniowej szyfru AES, dla każdego z 1 052 początkowych układów dwunastu równań istnieje $1,3 \times 10^{34} \approx 2^{114}$ układów pochodnych (wierzchołków w grafie), natomiast dla każdego z 2 690 682 początkowych układów trzynastu równań istnieje $3,5 \times 10^{40} \approx 2^{135}$ układów pochodnych. Oczywiście, nie ma możliwości przeszukania tak dużych przestrzeni, dlatego przyjęto pewne ograniczenia. Ponieważ operację xorowania równań można zrealizować jako mnożenie wektora przez odwracalną macierz binarną, w celu uzyskania efektywnej implementacji zastosowano rejestry przesuwne z liniowym sprzężeniem zwrotnym LFSR (*ang. Linear Feedback Shift Register*), gdzie wektor, reprezentujący układ równań, jest stanem LFSRa, a przejście do kolejnych stanów jest realizowane jako mnożenie kolejnego stanu przez tę samą macierz. Jeśli w rejestrze zostanie zastosowany wielomian pierwotny, to rejestr ten posiada maksymalny okres, równy 2^r, gdzie r jest długością rejestru. Dla układów 13 równań, jeden LFSR pozwoli na sprawdzenie tylko 8 192 układów, dlatego zastosowano trzy rejestry przesuwne z wielomianami pierwotnymi, których postać przedstawiono w tabeli 8.

Tabela 8: Postać wielomianów pierwotnych dla rejestrów przesuwnych z liniowym sprzężeniem zwrotnym, zastosowanych w drugiej metodzie przeszukiwań.

Liczba równań układu	*LFSR 1*	*LFSR 2*	*LFSR 3*
12	$x^{12}+x^{11}+x^{10}+x^2+1$	$x^{12}+x^{11}+x^{10}+x^4+1$	$x^{12}+x^{11}+x^8+x^6+1$
13	$x^{13}+x^{12}+x^2+x+1$	$x^{13}+x^{12}+x^4+x^2+1$	$x^{13}+x^{12}+x^{11}+x^8+1$

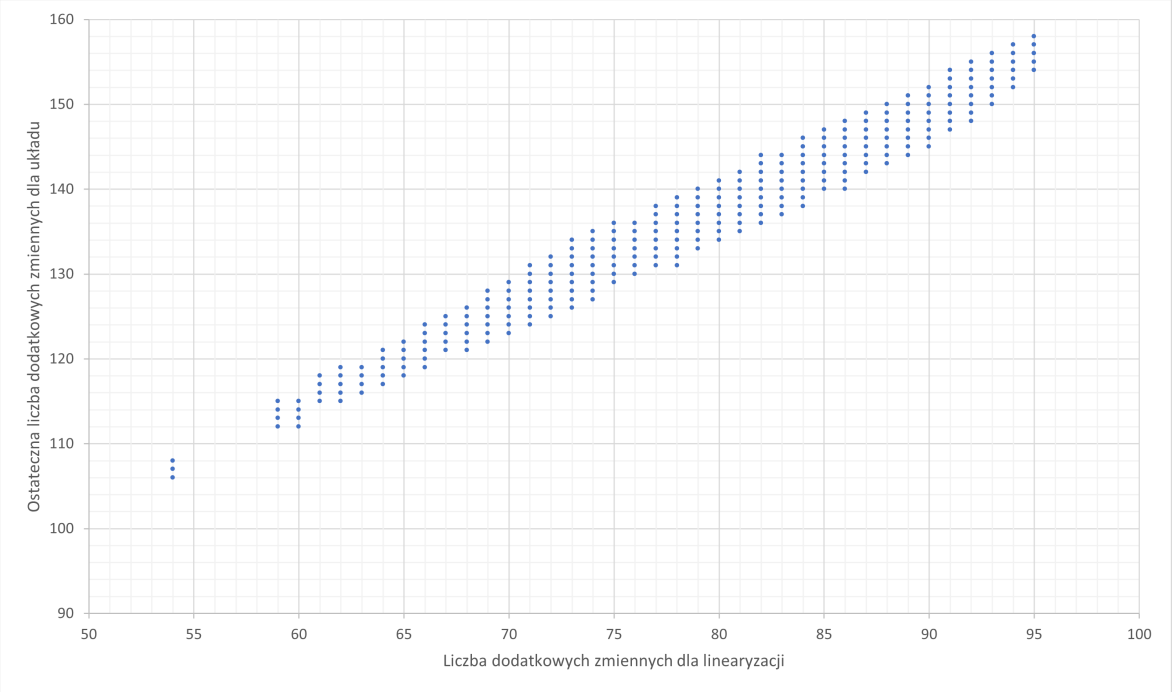
Idea opracowanej metody procesu poszukiwania efektywnego układu została przedstawiona za pomocą schematu na rysunku 34. Proces poszukiwań rozpoczyna się w danym układzie początkowym i przechodzi do następnych układów, zgodnie z przejściami do następnych stanów danego LFSRa, aż wykona jego pełny cykl. Cykl zewnętrzny przetwarza wektor równań zgodnie z wielomianem charakterystycznym dla pierwszego rejestru (LFSR 1). Każdy stan pierwszego rejestru przetwarzany jest



Rysunek 34: Ogólny schemat realizowania procesu szukania efektywnego układu, spełniającego skrzynkę podstawieniową szyfru AES, na podstawie metody drugiej.

zgodnie z wielomianem charakterystycznym drugiego rejestru przesuwnego (LFSR 2) oraz analogicznie, każdy stan drugiego rejestru przetwarzany jest zgodnie z wielomianem charakterystycznym trzeciego rejestru przesuwnego (LFSR 3). W każdym stanie trzeciego rejestru, wykonywane jest sprawdzenie, czy powstały układ równań wymaga mniejszej liczby zmiennych binarnych dla przedstawienia wielokrotności k_i każdego jego równania, niż dotychczas znaleziony najlepszy układ. Ponieważ przestrzeń przeszukiwania jest bardzo duża, przyjęto horyzont czasowy, równy 20 minut dla danego układu początkowego. Aby zmieścić się w przyjętym horyzoncie czasowym, wykonano 1 280 kroków pierwszego rejestru oraz 8 192 kroki, czyli pełen okres, dla rejestrów drugiego i trzeciego, sprawdzając tym samym $8,6 \times 10^{10}$ układów pochodnych dla danego układu początkowego.

Przeszukiwanie przestrzeni za pomocą przedstawionej metody zastosowano dla wszystkich 1 052 układów składających się z dwunastu równań, jednak nie znaleziono układu lepszego, niż układ znaleziony za pomocą pierwszej metody. W przypadku układów 13 równań, sprawdzono 281 544 z 2 690 682 układów początkowych, czyli około jednej dziesiątej, w losowej kolejności. Własności otrzymanych układów

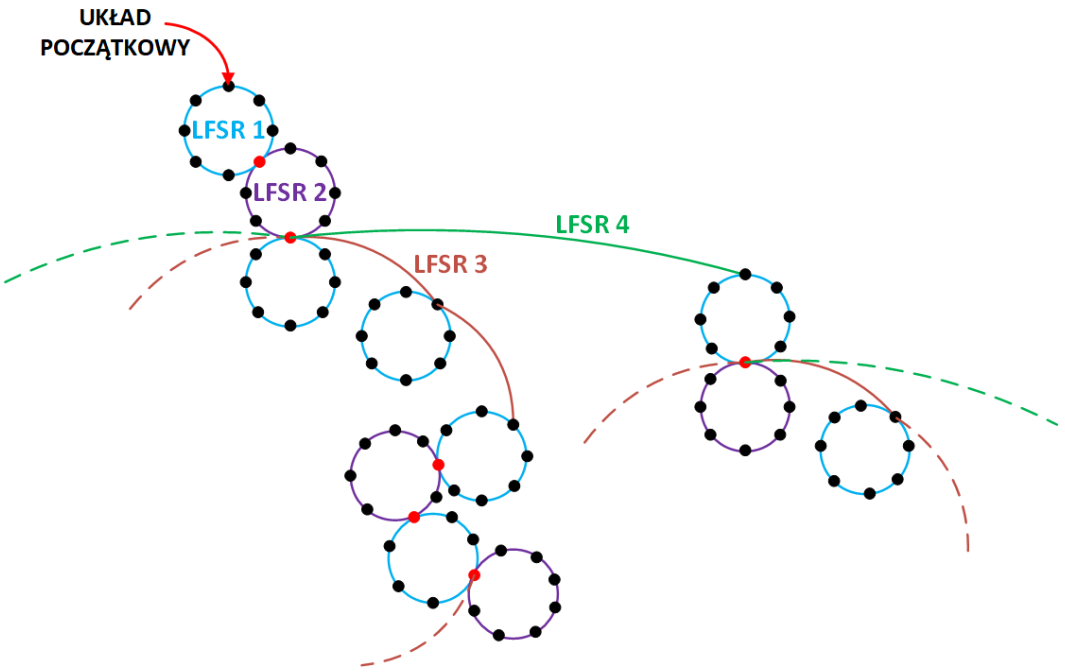


Rysunek 35: Ostateczna liczba dodatkowych zmiennych dla całego układu, w zależności od liczby zmiennych dla linearyzacji, dla układów znalezionych drugą metodą.

przedstawiono na rysunku 35, gdzie pokazano ostateczną liczbę dodatkowych zmiennych binarnych, wymaganych do wykonania transformacji otrzymanego układu do problemu QUBO, w zależności od liczby dodatkowych zmiennych binarnych potrzebnych do przeprowadzenia linearyzacji. Równania otrzymanych układów mają od 54 do 95 różnych jednomianów kwadratowych i wymagają od 106 do 158 dodatkowych zmiennych binarnych do przeprowadzenia transformacji do problemu QUBO. Jak pokazuje wykres na rysunku 35, liczba ta rośnie wraz ze wzrostem liczby różnych jednomianów kwadratowych w układzie. Z otrzymanych wartości liczby dodatkowych zmiennych binarnych potrzebnych do przedstawienia wartości wielokrotności k_i dla poszczególnych układów początkowych wynika, że opracowana metoda poprawia tę wartość co najwyżej o 6 zmiennych. Warto zauważyć, że nie występują układy spełniające skrzynkę podstawieniową szyfru AES, posiadające od 55 do 58 różnych jednomianów kwadratowych.

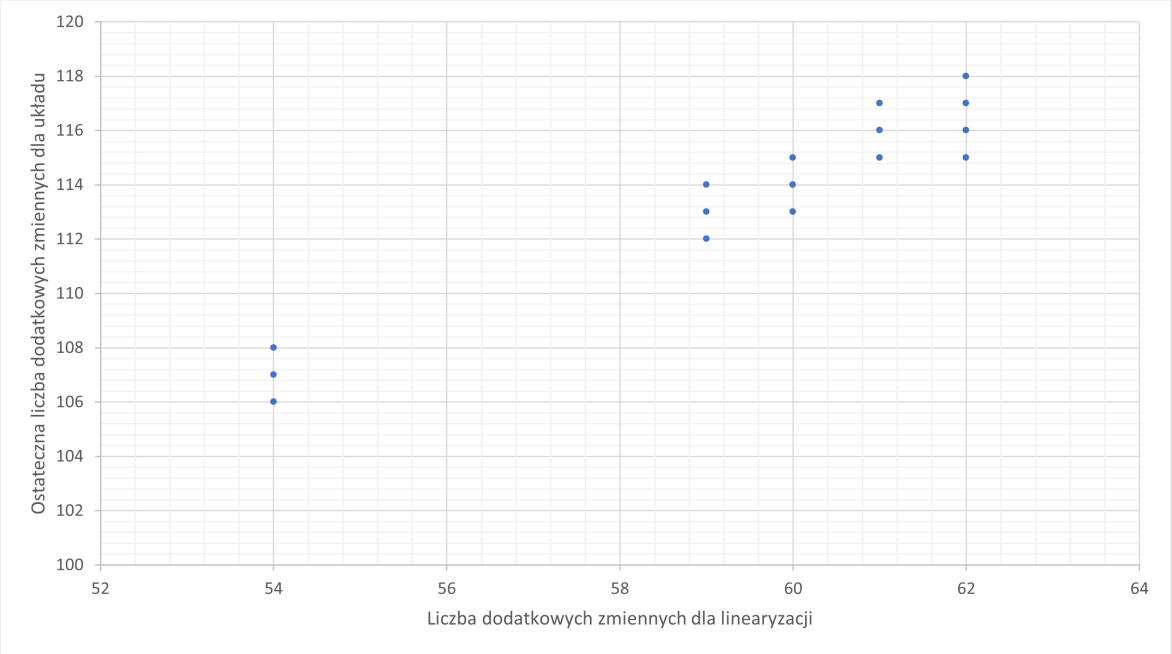
4.3.4 Poszukiwanie efektywnego układu spełniającego skrzynkę podstawieniową szyfru AES – Metoda III

Ponieważ proces poszukiwań za pomocą metody drugiej nie przyniósł zadowalających rezultatów i przeszukana przestrzeń była mała, opracowano kolejną metodę, w której spróbowano znaleźć wielkość, wyznaczającą kierunek chodzenia po grafie, czyli wierzchołek w cyklu danego LFSRa, który jest stanem początkowym kolejnego LFSRa. Dlatego też skupiono się na liczbie wszystkich jednomianów w równaniach danego układu. Zgodnie z przedstawioną wcześniej definicją funkcji celu problemu poszukiwania efektywnego układu, pożądane jest aby różnica pomiędzy liczbami wszystkich jednomianów w równaniach była jak największa. Dlatego przyjęto, że kierunek przechodzenia do kolejnego wierzchołka grafu będzie wyznaczać wariancja tej liczby, a za miarę efektywności danego układu pozostawiono liczbę zmiennych binarnych, potrzebnych do przedstawienia wartości wielokrotności k_i wszystkich równań przetwarzanego układu. Zdecydowano się również na zmianę schematu przeszukiwania przestrzeni układów, który pokazano na rysunku 36. Proces szukania efektywnego układu w trzeciej metodzie zrealizowano za pomocą czterech rejestrów przesuwnych z liniowym sprzężeniem zwrotnym, z których trzy pierwsze to rejestry z metody poprzedniej, natomiast czwarty rejestr zdefiniowano poprzez wielomian charakterystyczny, będący wielomianem pierwotnym postaci $x^{13} + x^{12} + x^4 + x^3 + 1$. Proces przebiega następująco: wektor równań układu początkowego przetwarzany jest najpierw za pomocą rejestru pierwszego, dla którego wykonywanych jest 8 192 kroków (pełen cykl). W każdym stanie tego LFSRa sprawdzana jest wariancja liczby wszystkich jednomianów w równaniach układu oraz liczba zmiennych binarnych dla wartości wielokrotności k_i. Jeśli otrzymana wariancja jest większa niż dotychczas zapamiętana wariancja maksymalna, to zostaje ona zaktualizowana. Na rysunku 36 układy z maksymalną wariancją w danym cyklu rejestru oznaczono czerwonymi punktami. Analogicznie, jeśli wyznaczona liczba zmiennych binarnych dla wartości k_i jest mniejsza niż dotychczas zapamiętana wartość minimalna, to jest ona aktualizowana oraz zapamiętywany jest nowy układ efektywny. Jeśli proces przeszukiwania wykonał pełny cykl rejestru i wrócił do stanu początkowego, a podczas cyklu przetwarzania został znaleziony nowy



Rysunek 36: Ogólny schemat realizowania procesu szukania efektywnego układu, spełniającego skrzynkę podstawieniową szyfru AES, na podstawie metody trzeciej.

układ z maksymalną wariancją, to jest on stanem początkowym drugiego rejestru. Przetwarzanie za pomocą drugiego rejestru jest identyczne jak w przypadku pierwszego. Jeśli podczas wykonywania kroków rejestru drugiego znaleziono nowy układ z maksymalną wariancją, to jest on stanem początkowym ponownie rejestru pierwszego. Przetwarzanie kolejnych układów, na zmianę za pomocą rejestru pierwszego i drugiego, realizowane jest do momentu, aż któryś z tych rejestrów wróci do stanu początkowego, bez znalezienia nowego układu o maksymalnej wariancji w swoim cyklu. Wtedy stosowany jest rejestr trzeci, aby wyjść z tego stanu początkowego, przechodząc do innego wierzchołka grafu oraz rozpoczynając przetwarzanie układu ponownie za pomocą naprzemiennie rejestru pierwszego i drugiego. Jeśli natomiast rejestr trzeci wykona 8 192 kroki (pełen cykl), to wyjście ze stanu początkowego i przejście do innego wierzchołka grafu realizowane jest za pomocą rejestru czwartego. Następnie powtarzana jest procedura przetwarzania układu trzema rejestrami. Szukanie efektywnego układu kończy się, gdy za pomocą rejestru czwartego zostanie wykonany pełen cykl. Biorąc pod uwagę wyniki uzyskane drugą metodą przedstawione na rysunku 35, z których wynika, że ostateczna liczba zmiennych binarnych, wymaganych podczas



Rysunek 37: Ostateczna liczba dodatkowych zmiennych dla całego układu, w zależności od liczby zmiennych dla linearyzacji, dla układów znalezionych trzecią metodą.

transformacji danego układu do problemu QUBO, rośnie wraz ze wzrostem liczby różnych jednomianów kwadratowych, zdecydowano, że poszukiwanie układu efektywnego za pomocą metody trzeciej zostanie przeprowadzone dla układów zawierających od 54 do 62 różnych jednomianów kwadratowych. Wyniki otrzymane trzecią metodą przedstawiono na rysunku 37, jako wykres zależności liczby zmiennych binarnych, wymaganych dla wartości wielokrotności k_i, od liczby zmiennych wymaganych dla linearyzacji. W wyniku zastosowanej metody nie udało się znaleźć układu efektywniejszego, niż w pierwszej metodzie. Najlepszy układ wymaga 106 dodatkowych zmiennych binarnych dla transformacji do problemu QUBO, a sama metoda przeszukiwania przestrzeni umożliwia redukcję liczby zmiennych dla wielokrotności k_i o co najwyżej 6 zmiennych.

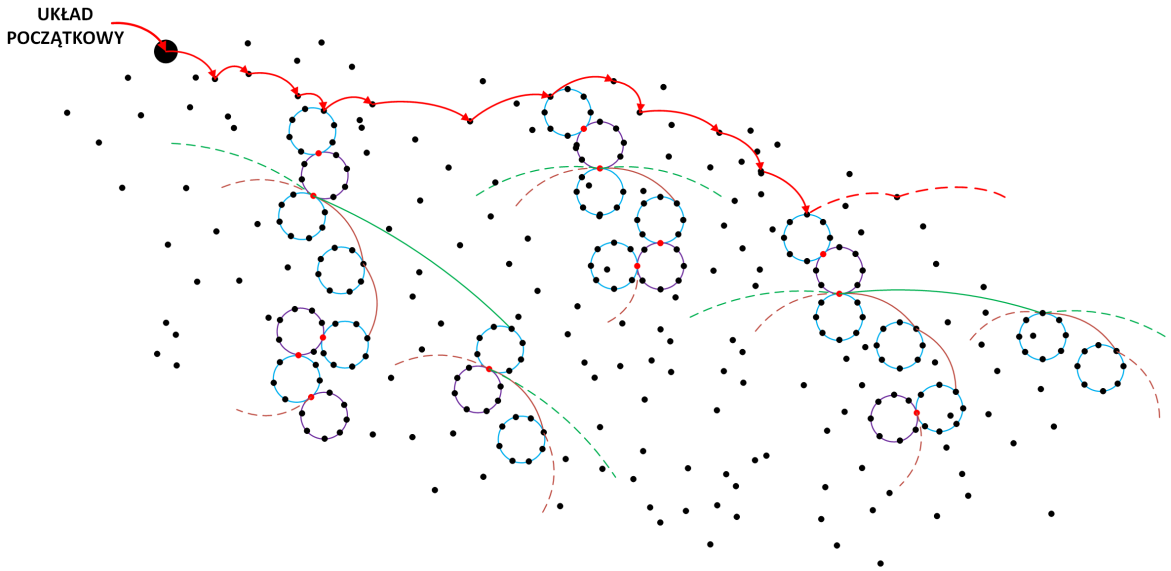
4.3.5 Poszukiwanie efektywnego układu spełniającego skrzynkę podstawieniową szyfru AES – Metoda IV

Ponieważ za pomocą poprzednich metod sprawdzono wszystkie obiecujące układy, bez zadowalającego efektu, w ostatniej opracowanej metodzie poszukiwań układu

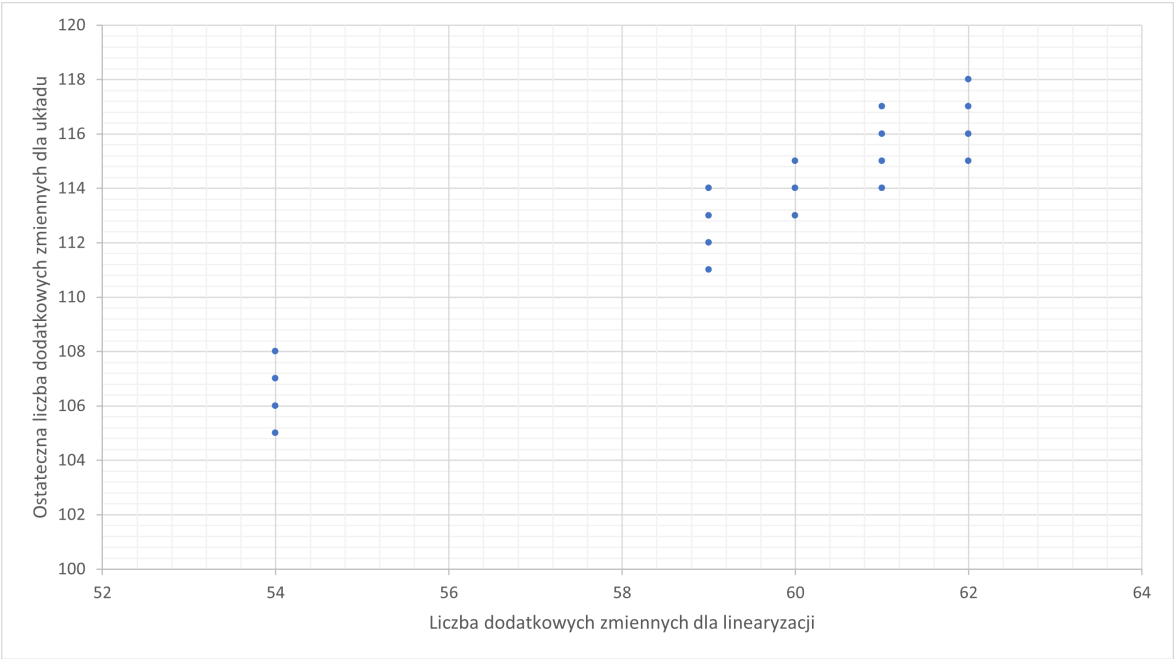
efektywnego skupiono się na zwiększeniu liczby sprawdzonych układów. W tym celu wygenerowano losową, odwracalną macierz binarną, wymiaru 13×13, postaci:

$$\begin{bmatrix}
1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0
\end{bmatrix}$$

Czwarta metoda przeszukiwania polega na wykonaniu najpierw pewnej liczby mnożeń danego początkowego układu przez powyższą macierz, a następnie wykonaniu przeszukiwania trzecią metodą. Schemat procesu poszukiwania efektywnego układu czwartą metodą przedstawiono na rysunku 38. Tak samo jak poprzednio, czwartą metodę zastosowano do układów zawierających od 54 do 62 różnych jednomianów kwadratowych, z liczbą początkowych mnożeń przez powyższą macierz równą od 1 do 1 335. Dzięki tej metodzie, udało się znaleźć trzy efektywne układy, posiadające 54 różne jednomiany kwadratowe, dla których ostateczna liczba dodatkowych zmiennych binarnych, potrzebnych do wykonania transformacji do problemu QUBO, wynosi 105. Otrzymane wyniki przedstawiono na wykresie, na rysunku 39. Opracowana metoda szukania układu efektywnego pozwala zredukować o 14 liczbę zmiennych binarnych dla wartości wielokrotności k_i.



Rysunek 38: Ogólny schemat realizowania procesu szukania efektywnego układu, spełniającego skrzynkę podstawieniową szyfru AES, na podstawie metody czwartej.



Rysunek 39: Ostateczna liczba dodatkowych zmiennych dla całego układu, w zależności od liczby zmiennych dla linearyzacji, dla układów znalezionych czwartą metodą.

4.3.6 Wyznaczanie równań opisujących warstwę nieliniową pojedynczej rundy algorytmu szyfrowania standardu AES

Spośród znalezionych trzech układów efektywnych wybrano jeden jako układ opisujący skrzynkę podstawieniową szyfru AES. Wielomiany wybranego układu są następującej postaci:

$$
\begin{aligned}
&x_0y_6 + x_0y_7 + x_1y_1 + x_1y_7 + x_2y_1 + x_2y_2 + x_2y_3 + x_2y_4 + x_2y_6 + x_3y_0 + x_3y_1 + x_3y_4 + \\
&+x_3y_7 + x_4y_2 + x_4y_3 + x_4y_4 + x_4y_5 + x_4y_6 + x_5y_0 + x_5y_1 + x_6y_1 + x_6y_2 + x_6y_3 + x_6y_5 + \\
&+x_6y_6 + x_7y_2 + x_7y_3 + x_7y_5 + x_1 + x_3 + y_5, \\
&x_0y_0 + x_0y_5 + x_0y_7 + x_1y_5 + x_2y_1 + x_2y_2 + x_2y_5 + x_3y_1 + x_3y_4 + x_3y_5 + x_4y_1 + x_4y_3 + \\
&+x_4y_5 + x_4y_7 + x_5y_1 + x_5y_3 + x_5y_6 + x_5y_7 + x_6y_1 + x_6y_2 + x_6y_4 + x_6y_5 + x_7y_2 + x_7y_3 + \\
&+x_7y_5 + x_1 + x_2 + x_3 + x_7 + y_3 + y_4, \\
&x_0y_5 + x_0y_6 + x_1y_1 + x_1y_5 + x_2y_0 + x_2y_1 + x_2y_3 + x_2y_4 + x_3y_0 + x_4y_0 + x_4y_1 + x_4y_3 + \\
&+x_4y_4 + x_4y_7 + x_5y_0 + x_5y_1 + x_5y_3 + x_5y_4 + x_5y_5 + x_5y_7 + x_6y_1 + x_6y_2 + x_7y_2 + x_7y_3 + \\
&+x_7y_5 + x_0 + x_1 + x_2 + y_0 + y_6 + 1, \\
&x_0y_3 + x_0y_6 + x_0y_7 + x_1y_1 + x_1y_5 + x_2y_0 + x_2y_1 + x_2y_4 + x_2y_6 + x_2y_7 + x_3y_0 + x_3y_2 + \\
&+x_3y_4 + x_3y_6 + x_4y_3 + x_4y_7 + x_5y_0 + x_5y_4 + x_6y_1 + x_6y_4 + x_7y_2 + x_7y_3 + x_7y_5 + x_7y_7 + \\
&+x_0 + x_1 + x_2 + x_6 + y_3 + y_4 + y_5, \\
&x_0y_0 + x_0y_2 + x_0y_3 + x_0y_4 + x_0y_5 + x_0y_7 + x_1y_1 + x_1y_5 + x_2y_3 + x_2y_4 + x_2y_5 + x_2y_6 + \\
&+x_2y_7 + x_3y_1 + x_3y_7 + x_4y_1 + x_4y_5 + x_5y_1 + x_5y_6 + x_5y_7 + x_6y_2 + x_6y_3 + x_6y_6 + x_7y_5 + \\
&+x_3 + x_4 + x_7 + y_0 + y_5 + y_7 + 1, \\
&x_1y_1 + x_2y_0 + x_2y_3 + x_2y_7 + x_3y_2 + x_3y_5 + x_3y_7 + x_4y_0 + x_4y_2 + x_4y_4 + x_4y_7 + x_5y_1 + \\
&+x_5y_2 + x_5y_4 + x_5y_5 + x_5y_6 + x_6y_0 + x_6y_1 + x_6y_4 + x_6y_5 + x_7y_1 + x_7y_2 + x_7y_3 + \\
&+x_7y_5 + x_1 + x_6 + y_2 + y_5 + y_7 \\
&x_0y_3 + x_0y_4 + x_0y_5 + x_1y_1 + x_1y_5 + x_2y_4 + x_2y_6 + x_3y_0 + x_3y_1 + x_3y_6 + x_3y_7 + x_4y_0 + \\
&+x_4y_1 + x_4y_3 + x_4y_6 + x_4y_7 + x_5y_2 + x_5y_3 + x_5y_6 + x_6y_2 + x_6y_5 + x_7y_1 + x_7y_7 + x_0 + \\
&+x_4 + x_5 + y_0 + y_4 + y_6 + y_7, \\
&x_0y_0 + x_0y_4 + x_0y_6 + x_1y_1 + x_1y_5 + x_2y_0 + x_2y_1 + x_2y_2 + x_2y_3 + x_2y_4 + x_2y_5 + x_2y_6 + \\
&+x_2y_7 + x_3y_0 + x_3y_5 + x_4y_7 + x_5y_0 + x_5y_6 + x_6y_5 + x_7y_2 + x_7y_7 + x_0 + x_5 + x_6 + x_7 + \\
&+y_3 + y_4 + y_5 + y_6 + 1,
\end{aligned}
$$

$$x_0y_7 + x_1y_1 + x_1y_5 + x_2y_2 + x_2y_4 + x_2y_5 + x_2y_7 + x_3y_3 + x_3y_4 + x_4y_2 + x_4y_4 + x_4y_5 +$$
$$+x_5y_0 + x_5y_1 + x_5y_4 + x_6y_0 + x_6y_2 + x_6y_3 + x_7y_2 + x_7y_5 + x_0 + x_1 + x_2 + x_3 + x_6 +$$
$$+x_7 + y_0 + y_2 + y_3 + 1,$$
$$x_0y_0 + x_0y_2 + x_0y_3 + x_0y_5 + x_0y_6 + x_2y_0 + x_2y_1 + x_2y_4 + x_2y_6 + x_3y_1 + x_3y_3 + x_3y_6 +$$
$$+x_3y_7 + x_4y_0 + x_4y_1 + x_4y_6 + x_5y_1 + x_5y_2 + x_5y_4 + x_5y_6 + x_5y_7 + x_6y_2 + x_6y_5 + x_7y_3 +$$
$$+x_7y_7 + x_5 + x_7,$$
$$x_0y_0 + x_0y_2 + x_0y_3 + x_0y_5 + x_1y_1 + x_1y_7 + x_2y_2 + x_2y_7 + x_3y_2 + x_3y_4 + x_3y_5 + x_3y_7 +$$
$$+x_4y_4 + x_4y_5 + x_5y_1 + x_5y_4 + x_5y_6 + x_6y_1 + x_6y_3 + x_7y_7 + x_2 + x_3 + x_6 + y_0 + y_1 +$$
$$+y_5 + y_7,$$
$$x_0y_2 + x_0y_3 + x_0y_6 + x_1y_1 + x_1y_5 + x_1y_7 + x_3y_2 + x_3y_5 + x_3y_7 + x_4y_0 + x_4y_1 + x_4y_2 +$$
$$+x_4y_3 + x_4y_5 + x_5y_0 + x_5y_1 + x_5y_2 + x_5y_3 + x_5y_6 + x_5y_7 + x_7y_5 + x_2 + x_3 + x_4 + y_0 +$$
$$+y_6 + 1,$$
$$x_0y_4 + x_0y_6 + x_1y_5 + x_2y_1 + x_4y_0 + x_5y_0 + x_5y_1 + x_5y_3 + x_5y_6 + x_6y_6 + x_7y_1 + x_7y_2 +$$
$$+x_7y_3 + x_2 + y_5.$$

Ponieważ warstwa nieliniowa algorytmu szyfrowania standardu AES składa się z 16 skrzynek podstawieniowych, całą warstwę można przedstawić jako układ składający się z 208 równań kwadratowych. Liczba dodatkowych zmiennych binarnych potrzebnych do zlinearyzowania tego układu wynosi 864, a liczba zmiennych potrzebnych do przedstawienia wielokrotności k_i wszystkich jego równań wynosi 816. Ostatecznie, podczas transformacji układu równań, opisującego warstwę nieliniową pojedynczej rundy algorytmu szyfrowania standardu AES, do problemu optymalizacyjnego w postaci QUBO, wymaganych jest 1 680 dodatkowych zmiennych binarnych.

4.3.7 Wyznaczanie równań opisujących warstwę liniową pojedynczej rundy algorytmu szyfrowania standardu AES

Warstwa liniowa pojedynczej rundy algorytmu szyfrowania standardu AES, z wyjątkiem rundy zerowej oraz ostatniej, składa się z trzech operacji, kolejno: ShiftRows, MixColumns oraz AddRoundKey. Chcąc opisać tę warstwę za pomocą równań wielomianowych nad ciałem $GF(2)$, należy wyznaczyć równanie dla każdego bitu stanu

pośredniego. Oczywiście, ponieważ jest to warstwa, składająca się z operacji liniowych, otrzymane równania również będą liniowe.

Długość bloku przetwarzania dla szyfru AES wynosi 128 bitów i jest ona stała dla wszystkich jego trzech wariantów. Przedstawmy ten blok w postaci $(b_0, b_1, \ldots, b_{127})$, gdzie b_j, dla $j = \overline{0,127}$, oznacza j-ty bit bloku. Stan pośredni jest macierzą o wymiarach 4×4 bajty, gdzie w kolejne kolumny wstawiane są odpowiednie bity bloku, w następujący sposób:

$$\left[\begin{array}{c|c|c|c} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ \hline s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ \hline s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ \hline s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{array}\right] = \left[\begin{array}{c|c|c|c} b_0, b_1, \ldots b_7 & b_{32}, b_{33}, \ldots b_{39} & b_{64}, b_{65}, \ldots b_{71} & b_{96}, b_{97}, \ldots b_{103} \\ \hline b_8, b_9, \ldots b_{15} & b_{40}, b_{41}, \ldots b_{47} & b_{72}, b_{73}, \ldots b_{79} & b_{104}, b_{105}, \ldots b_{111} \\ \hline b_{16}, b_{17}, \ldots b_{23} & b_{48}, b_{49}, \ldots b_{55} & b_{80}, b_{81}, \ldots b_{87} & b_{112}, b_{113}, \ldots b_{119} \\ \hline b_{24}, b_{25}, \ldots b_{31} & b_{56}, b_{57}, \ldots b_{63} & b_{88}, b_{89}, \ldots b_{95} & b_{120}, b_{121}, \ldots b_{127} \end{array}\right] \tag{55}$$

W szyfrze AES wszystkie operacje wykonywane są na bajtach $s_{r,c}$ macierzy stanu, gdzie r jest numerem wiersza, a c jest numerem kolumny tej macierzy. Działania arytmetyczne, dodawanie i mnożenie, wykonywane są w rozszerzonym ciele binarnym $GF(2^8)$ z wielomianem nierozkładalnym $m(x) = x^8 + x^4 + x^3 + x + 1$.

Dla danej macierzy stanu, operacja ShiftRows cyklicznie przesuwa bajty wiersza $r = 1$, $r = 2$ oraz $r = 3$ o odpowiednio 1, 2 i 3 bajty w lewo. Wykonując operację ShiftRows na macierzy stanu (55), otrzymuje się następującą macierz:

$$\left[\begin{array}{c|c|c|c} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ \hline s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ \hline s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ \hline s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{array}\right] = \left[\begin{array}{c|c|c|c} b_0, b_1, \ldots b_7 & b_{32}, b_{33}, \ldots b_{39} & b_{64}, b_{65}, \ldots b_{71} & b_{96}, b_{97}, \ldots b_{103} \\ \hline b_{40}, b_{41}, \ldots b_{47} & b_{72}, b_{73}, \ldots b_{79} & b_{104}, b_{105}, \ldots b_{111} & b_8, b_9, \ldots b_{15} \\ \hline b_{80}, b_{81}, \ldots b_{87} & b_{112}, b_{113}, \ldots b_{119} & b_{16}, b_{17}, \ldots b_{23} & b_{48}, b_{49}, \ldots b_{55} \\ \hline b_{120}, b_{121}, \ldots b_{127} & b_{24}, b_{25}, \ldots b_{31} & b_{56}, b_{57}, \ldots b_{63} & b_{88}, b_{89}, \ldots b_{95} \end{array}\right] \tag{56}$$

Opisując tę operację za pomocą równań nad ciałem $GF(2)$, każdemu bitowi bajtu $s'_{r',c'}$ macierzy wyjściowej, przypisuje się odpowiedni bit bajtu $s_{r,c}$ wejściowej macierzy stanu, gdzie r', c' to współrzędne bajtu po wykonaniu operacji ShiftRows.

Podczas wykonywania operacji MicColumns, każdy bajt kolumny macierzy stanu odwzorowywany jest na nową wartość, która jest funkcją wszystkich czterech bajtów kolumny macierzy wejściowej. Operację tę można zdefiniować jako mnożenie macie-

rzy wartości stałych przez macierz stanu, w następującej postaci:

$$\left[\begin{array}{c|c|c|c} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ \hline s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ \hline s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ \hline s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{array}\right] = \left[\begin{array}{c|c|c|c} 2 & 3 & 1 & 1 \\ \hline 1 & 2 & 3 & 1 \\ \hline 1 & 1 & 2 & 3 \\ \hline 3 & 1 & 1 & 2 \end{array}\right] \times \left[\begin{array}{c|c|c|c} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ \hline s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ \hline s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ \hline s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{array}\right] \tag{57}$$

Zgodnie z operacją mnożenia macierzy, każdy element $s'_{r,c}$ macierzy wynikowej jest sumą iloczynów elementów jednego wiersza i jednej kolumny, co można przedstawić za pomocą następujących równań:

$$\begin{aligned} s'_{0,c} &= \left(2 \cdot s_{0,c}\right) \oplus \left(3 \cdot s_{1,c}\right) \oplus s_{2,c} \oplus s_{3,c}, \\ s'_{1,c} &= s_{0,c} \oplus \left(2 \cdot s_{1,c}\right) \oplus \left(3 \cdot s_{2,c}\right) \oplus s_{3,c}, \\ s'_{2,c} &= s_{0,c} \oplus s_{1,c} \oplus \left(2 \cdot s_{2,c}\right) \oplus \left(3 \cdot s_{3,c}\right), \\ s'_{3,c} &= \left(3 \cdot s_{0,c}\right) \oplus s_{1,c} \oplus s_{2,c} \oplus \left(2 \cdot s_{3,c}\right), \end{aligned} \tag{58}$$

gdzie operacje arytmetyczne dodawania i mnożenia wykonywane są w ciele $GF(2^8)$. Przedstawmy każdy bajt macierzy stanu, o współrzędnych (r,c), w postaci elementu ciała $GF(2^8)$, w następujący sposób:

$$s_{r,c} = b_j x^7 + b_{j+1} x^6 + b_{j+2} x^5 + b_{j+3} x^4 + b_{j+4} x^3 + b_{j+5} x^2 + b_{j+6} x + b_{j+7}, \tag{59}$$

gdzie $j = 8(r+4c)$. Następnie zdefiniujmy mnożenie dowolnego bajtu macierzy stanu przez stałe 2 i 3. Ponieważ każdą n-bitową liczbę można jednoznacznie przedstawić jako wielomian nad ciałem $GF(2^n)$, liczbie 2 odpowiada wielomian x w ciele $GF(2^8)$. Mnożąc dowolny bajt $s_{r,c}$ przez wielomian x, otrzymujemy:

$$2 \cdot s_{r,c} = b_j x^8 + b_{j+1} x^7 + b_{j+2} x^6 + b_{j+3} x^5 + b_{j+4} x^4 + b_{j+5} x^3 + b_{j+6} x^2 + b_{j+7} x.$$

Ponieważ $x^8 \ mod(x^8 + x^4 + x^3 + x + 1) = x^4 + x^3 + x + 1$, po redukcji modularnej dostajemy następujące równanie:

$$\begin{aligned} 2 \cdot s_{r,c} = {} & b_{j+1} x^7 + b_{j+2} x^6 + b_{j+3} x^5 + \left(b_{j+4} + b_j\right) x^4 + \left(b_{j+5} + b_j\right) x^3 + b_{j+6} x^2 + \\ & + \left(b_{j+7} + b_j\right) x + b_j. \end{aligned} \tag{60}$$

Analogicznie wyznaczamy postać wielomianu, będącego iloczynem dowolnego bajtu

macierzy stanu oraz wartości 3. Otrzymany wielomian ma następującą postać:

$$3 \cdot s_{r,c} = \left(b_j + b_{j+1}\right) x^7 + \left(b_{j+1} + b_{j+2}\right) x^6 + \left(b_{j+2} + b_{j+3}\right) x^5 + \left(b_j + b_{j+3} + b_{j+4}\right) x^4 + \\ + \left(b_j + b_{j+4} + b_{j+5}\right) x^3 + \left(b_{j+5} + b_{j+6}\right) x^2 + \left(b_j + b_{j+6} + b_{j+7}\right) x + \left(b_j + b_{j+7}\right). \tag{61}$$

Podstawiając równania (59), (60) oraz (61) do równań (58), przedstawiających operację MixColumns, otrzymujemy wielomian, który definiuje dany bajt $s'_{r,c}$ wynikowej macierzy stanu w zależności od konkretnych bitów stanu wejściowego. Przykładowo, bajt zerowego wiersza i kolumny c wyznaczany jest na podstawie następującego wielomianu:

$$s'_{0,c} = \left(b_{32c+1} + b_{32c+8} + b_{32c+8+1} + b_{32c+16} + b_{32c+24}\right) x^7 + \left(b_{32c+2} + b_{32c+8+1} + \right. \\ \left. + b_{32c+8+2} + b_{32c+16+1} + b_{32c+24+1}\right) x^6 + \left(b_{32c+3} + b_{32c+8+2} + b_{32c+8+3} + b_{32c+16+2} + \right. \\ \left. + b_{32c+24+2}\right) x^5 + \left(b_{32c+4} + b_{32c} + b_{32c+8} + b_{32c+8+3} + b_{32c+8+4} + b_{32c+24+3} + \right. \\ \left. + b_{32c+16+3}\right) x^4 + \left(b_{32c+5} + b_{32c} + b_{32c+8} + b_{32c+8+4} + b_{32c+8+5} + b_{32c+16+4} + \right. \\ \left. + b_{32c+24+4}\right) x^3 + \left(b_{32c+6} + b_{32c+8+5} + b_{32c+8+6} + b_{32c+16+5} + b_{32c+24+5}\right) x^2 + \\ + \left(b_{32c+7} + b_{32c} + b_{32c+8} + b_{32c+8+6} + b_{32c+8+7} + b_{32c+16+6} + b_{32c+24+6}\right) x + \\ + b_{32c} + \left(b_{32c+8} + b_{32c+8+7}\right) + b_{32c+16+7} + b_{32c+24+7}. \tag{62}$$

Na podstawie wielomianu (62) można wyznaczyć równania liniowe nad ciałem binarnym, reprezentujące bity bajtu $s'_{0,c}$. Równania te mają następującą postać:

$$\begin{aligned}
b'_{32c} &= b_{32c+1} + b_{32c+8} + b_{32c+8+1} + b_{32c+16} + b_{32c+24}, \\
b'_{32c+1} &= b_{32c+2} + b_{32c+8+1} + b_{32c+8+2} + b_{32c+16+1} + b_{32c+24+1}, \\
b'_{32c+2} &= b_{32c+3} + b_{32c+8+2} + b_{32c+8+3} + b_{32c+16+2} + b_{32c+24+2}, \\
b'_{32c+3} &= b_{32c+4} + b_{32c} + b_{32c+8} + b_{32c+8+3} + b_{32c+8+4} + b_{32c+16+3} + b_{32c+24+3}, \\
b'_{32c+4} &= b_{32c+5} + b_{32c} + b_{32c+8} + b_{32c+8+4} + b_{32c+8+5} + b_{32c+16+4} + b_{32c+24+4}, \\
b'_{32c+5} &= b_{32c+6} + b_{32c+8+5} + b_{32c+8+6} + b_{32c+16+5} + b_{32c+24+5}, \\
b'_{32c+6} &= b_{32c+7} + b_{32c} + b_{32c+8} + b_{32c+8+6} + b_{32c+8+7} + b_{32c+16+6} + b_{32c+24+6}, \\
b'_{32c+7} &= b_{32c} + b_{32c+8} + b_{32c+8+7} + b_{32c+16+7} + b_{32c+24+7}.
\end{aligned} \tag{63}$$

Analogicznie wyznaczane są pozostałe bajty, w wyniku czego otrzymujemy równania

definiujące bajt $s'_{1,c}$:

$$
\begin{aligned}
b'_{32c+8} &= b_{32c} + b_{32c+8+1} + b_{32c+16} + b_{32c+16+1} + b_{32c+24},\\
b'_{32c+8+1} &= b_{32c+1} + b_{32c+8+2} + b_{32c+16+1} + b_{32c+16+2} + b_{32c+24+1},\\
b'_{32c+8+2} &= b_{32c+2} + b_{32c+8+3} + b_{32c+16+2} + b_{32c+16+3} + b_{32c+24+2},\\
b'_{32c+8+3} &= b_{32c+3} + b_{32c+8+4} + b_{32c+8} + b_{32c+16} + b_{32c+16+3} + b_{32c+16+4} + b_{32c+24+3},\\
b'_{32c+8+4} &= b_{32c+4} + b_{32c+8+5} + b_{32c+8} + b_{32c+16} + b_{32c+16+4} + b_{32c+16+5} + b_{32c+24+4},\\
b'_{32c+8+5} &= b_{32c+5} + b_{32c+8+6} + b_{32c+16+5} + b_{32c+16+6} + b_{32c+24+5},\\
b'_{32c+8+6} &= b_{32c+6} + b_{32c+8+7} + b_{32c+8} + b_{32c+16} + b_{32c+16+6} + b_{32c+16+7} + b_{32c+24+6},\\
b'_{32c+8+7} &= b_{32c+7} + b_{32c+8} + b_{32c+16} + b_{32c+16+7} + b_{32c+24+7},
\end{aligned}
\tag{64}
$$

następnie równania dla bajtu $s'_{2,c}$:

$$
\begin{aligned}
b'_{32c+16} &= b_{32c} + b_{32c+8} + b_{32c+16+1} + b_{32c+24} + b_{32c+24+1},\\
b'_{32c+16+1} &= b_{32c+1} + b_{32c+8+1} + b_{32c+16+2} + b_{32c+24+1} + b_{32c+24+2},\\
b'_{32c+16+2} &= b_{32c+2} + b_{32c+8+2} + b_{32c+16+3} + b_{32c+24+2} + b_{32c+24+3},\\
b'_{32c+16+3} &= b_{32c+3} + b_{32c+8+3} + b_{32c+16+4} + b_{32c+16} + b_{32c+24} + b_{32c+24+3} + b_{32c+24+4},\\
b'_{32c+16+4} &= b_{32c+4} + b_{32c+8+4} + b_{32c+16+5} + b_{32c+16} + b_{32c+24} + b_{32c+24+4} + b_{32c+24+5},\\
b'_{32c+16+5} &= b_{32c+5} + b_{32c+8+5} + b_{32c+16+6} + b_{32c+24+5} + b_{32c+24+6},\\
b'_{32c+16+6} &= b_{32c+6} + b_{32c+8+6} + b_{32c+16+7} + b_{32c+16} + b_{32c+24} + b_{32c+24+6} + b_{32c+24+7},\\
b'_{32c+16+7} &= b_{32c+7} + b_{32c+8+7} + b_{32c+16} + b_{32c+24} + b_{32c+24+7}
\end{aligned}
\tag{65}
$$

oraz równania dla bajtu $s'_{3,c}$:

$$\begin{aligned}
b'_{32c+24} &= b_{32c} + b_{32c+1} + b_{32c+8} + b_{32c+16} + b_{32c+24+1},\\
b'_{32c+24+1} &= b_{32c+1} + b_{32c+2} + b_{32c+8+1} + b_{32c+16+1} + b_{32c+24+2},\\
b'_{32c+24+2} &= b_{32c+2} + b_{32c+3} + b_{32c+8+2} + b_{32c+16+2} + b_{32c+24+3},\\
b'_{32c+24+3} &= b_{32c} + b_{32c+3} + b_{32c+4} + b_{32c+8+3} + b_{32c+16+3} + b_{32c+24+4} + b_{32c+24},\\
b'_{32c+24+4} &= b_{32c} + b_{32c+4} + b_{32c+5} + b_{32c+8+4} + b_{32c+16+4} + b_{32c+24+5} + b_{32c+24},\\
b'_{32c+24+5} &= b_{32c+5} + b_{32c+6} + b_{32c+8+5} + b_{32c+16+5} + b_{32c+24+6},\\
b'_{32c+24+6} &= b_{32c} + b_{32c+6} + b_{32c+7} + b_{32c+8+6} + b_{32c+16+6} + b_{32c+24+7} + b_{32c+24},\\
b'_{32c+24+7} &= b_{32c} + b_{32c+7} + b_{32c+8+7} + b_{32c+16+7} + b_{32c+24}.
\end{aligned} \tag{66}$$

Aby opisać przetwarzanie macierzy stanów przez operację MixColumns, należy wyznaczyć równania (63), (64), (65) oraz (66) dla $c = \overline{0,3}$.

Operacja AddRoundKey polega na wykonaniu bitowej operacji xor na odpowiadających sobie bajtach macierzy stanu oraz macierzy klucza rundowego. Operację tę można przedstawić w następującej postaci:

$$\begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix} = \begin{bmatrix} b_0, b_1, \dots b_7 & b_{32}, b_{33}, \dots b_{39} & b_{64}, b_{65}, \dots b_{71} & b_{96}, b_{97}, \dots b_{103} \\ b_8, b_9, \dots b_{15} & b_{40}, b_{41}, \dots b_{47} & b_{72}, b_{73}, \dots b_{79} & b_{104}, b_{105}, \dots b_{111} \\ b_{16}, b_{17}, \dots b_{23} & b_{48}, b_{49}, \dots b_{55} & b_{80}, b_{81}, \dots b_{87} & b_{112}, b_{113}, \dots b_{119} \\ b_{24}, b_{25}, \dots b_{31} & b_{56}, b_{57}, \dots b_{63} & b_{88}, b_{89}, \dots b_{95} & b_{120}, b_{121}, \dots b_{127} \end{bmatrix} \oplus$$
$$\oplus \begin{bmatrix} kr_0, kr_1, \dots kr_7 & kr_{32}, kr_{33}, \dots kr_{39} & kr_{64}, kr_{65}, \dots kr_{71} & kr_{96}, kr_{97}, \dots kr_{103} \\ kr_8, kr_9, \dots kr_{15} & kr_{40}, kr_{41}, \dots kr_{47} & kr_{72}, kr_{73}, \dots kr_{79} & kr_{104}, kr_{105}, \dots kr_{111} \\ kr_{16}, kr_{17}, \dots kr_{23} & kr_{48}, kr_{49}, \dots kr_{55} & kr_{80}, kr_{81}, \dots kr_{87} & kr_{112}, kr_{113}, \dots kr_{119} \\ kr_{24}, kr_{25}, \dots kr_{31} & kr_{56}, kr_{57}, \dots kr_{63} & kr_{88}, kr_{89}, \dots kr_{95} & kr_{120}, kr_{121}, \dots kr_{127} \end{bmatrix} \tag{67}$$

Ponieważ równania opisujące szyfr AES wyznaczane są nad ciałem binarnym, każdy bit macierzy stanu po operacji AddRoundKey zdefiniowany jest jako suma odpowiadającego mu bitu macierzy wejściowej oraz odpowiadającego mu bitu klucza rundowego.

W celu przedstawienia całej warstwy liniowej za pomocą równań nad ciałem $GF(2)$ należy dokonać złożenia operacji ShiftRows, MixColumns oraz AddRound-

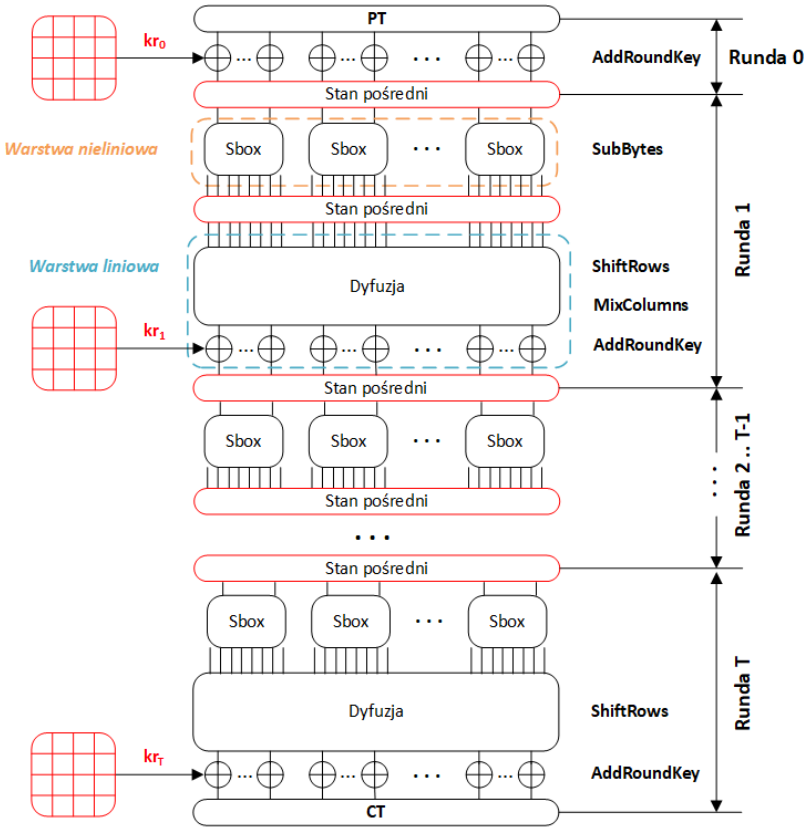
Key. Złożenie to można przedstawić za pomocą równań (58) operacji MixColumns, w których operacja ShiftRows powoduje zmianę indeksów bitów, a operacja AddRoundKey powoduje dodanie do każdego równania odpowiedniego bajtu klucza rundowego. Ostatecznie, równania reprezentujące warstwę liniową można przedstawić następująco:

$$\begin{aligned}
s'_{0,c} &= \left(2 \cdot s_{0,c}\right) \oplus \left(3 \cdot s_{1,(c+1)mod\ 4}\right) \oplus s_{2,(c+2)mod\ 4} \oplus s_{3,(c+3)mod\ 4} \oplus kr_{0,c}, \\
s'_{1,c} &= s_{0,c} \oplus \left(2 \cdot s_{1,(c+1)mod\ 4}\right) \oplus \left(3 \cdot s_{2,(c+2)mod\ 4}\right) \oplus s_{3,(c+3)mod\ 4} \oplus kr_{1,c}, \\
s'_{2,c} &= s_{0,c} \oplus s_{1,(c+1)mod\ 4} \oplus \left(2 \cdot s_{2,(c+2)mod\ 4}\right) \oplus \left(3 \cdot s_{3,(c+3)mod\ 4}\right) \oplus kr_{2,c}, \\
s'_{3,c} &= \left(3 \cdot s_{0,c}\right) \oplus s_{1,(c+1)mod\ 4} \oplus s_{2,(c+2)mod\ 4} \oplus \left(2 \cdot s_{3,(c+3)mod\ 4}\right) \oplus kr_{3,c},
\end{aligned} \tag{68}$$

dla $c = \overline{0,3}$. Stąd całą warstwę liniową można zapisać za pomocą układu, składającego się ze 128 liniowych równań wielomianowych o wielu zmiennych. Ponieważ są to równania liniowe, nie wymagają dodatkowych zmiennych binarnych podczas linearyzacji. Ze względu na liczbę wszystkich jednomianów w równaniach, układ ten składa się z 80 równań, zawierających 7 jednomianów oraz z 48 równań zawierających 9 jednomianów. Podczas transformacji tego układu do problemu optymalizacyjnego w postaci QUBO wymaganych będzie 304 dodatkowych zmiennych binarnych, do przedstawienia wartości wielokrotności k_i dla wszystkich równań.

4.3.8 Wyznaczanie równań, nad ciałem $GF(2)$, opisujących algorytm szyfrowania standardu AES

W celu przedstawienia algorytmu szyfrowania standardu AES za pomocą równań wielomianowych nad ciałem binarnym, należy wprowadzić stany pośrednie i zdefiniować je za pomocą zmiennych binarnych. Aby otrzymać równania stopnia co najwyżej 2, stany pośrednie zostały wprowadzone pomiędzy każdą rundę oraz pomiędzy warstwami liniową i nieliniową w każdej rundzie, z wyjątkiem rundy ostatniej. Sposób wprowadzenia stanów pośrednich został przedstawiony na rysunku 40, gdzie oznaczono je kolorem czerwonym. Dodatkowo kolorem czerwonym oznaczono również klucze rundowe, ponieważ także one zostały zdefiniowane za pomocą zmiennych binarnych. Zatem liczba zmiennych binarnych wymaganych do utworzenia układu rów-

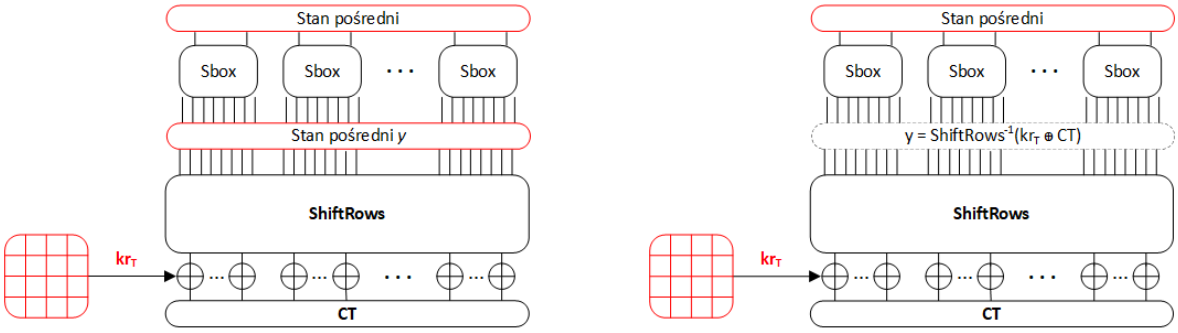


Rysunek 40: Podział algorytmu szyfrowania standardu AES za pomocą stanów pośrednich.

nań, co najwyżej kwadratowych nad ciałem binarnym, opisującego algorytm szyfrowania standardu AES, wynosi $128(2T-1)$ dla stanów pośrednich oraz $128(T+1)$ dla kluczy rundowych; razem otrzymujemy $384T$, gdzie T to liczba rund dla danej wersji szyfru. Podczas ataku znany jest tekst jawny i odpowiadający mu szyfrogram, oznaczone na rysunku 40 jako, odpowiednio, PT oraz CT. Wyznaczmy teraz postać równań dla rundy zerowej oraz ostatniej. Runda zerowa składa się tylko z operacji AddRoundKey, więc równania opisujące tę rundę są liniowe, postaci:

$$PT_j + kr_{0_j} + x_j = 0, \tag{69}$$

dla $j = \overline{0,127}$, gdzie j oznacza j-ty bit: znanego tekstu jawnego PT_j, klucza rundowego dla rundy zerowej kr_{0_j} oraz stanu pośredniego x_j. Czyli runda zerowa zostaje przedstawiona za pomocą 128-miu równań liniowych, nie wymagających dodatkowych zmiennych podczas linearyzacji oraz składających się, w zależności od wartości bitu PT_j, z dwóch albo trzech jednomianów, co daje jedną dodatkową zmienną



(a) Ostatnia runda ze stanem pośrednim pomiędzy warstwami.

(b) Ostatnia runda bez stanu pośredniego pomiędzy warstwami.

Rysunek 41: Analiza przedstawienia ostatniej rundy szyfru AES za pomocą efektywnego układu równań.

binarną dla każdego równania, do przedstawienia wartości wielokrotności k_i.

W przypadku rundy ostatniej, która w warstwie liniowej składa się tylko z operacji ShiftRows i AddRoundKey, typowym dla wcześniejszych rozważań podejściem byłoby wprowadzenie stanu pośredniego po warstwie nieliniowej, co pokazano na rysunku 41a, a następnie zapisanie osobnych układów dla warstwy liniowej i nieliniowej. Wtedy równania dla warstwy liniowej mają postać:

$$y_{j'} + kr_{T_j} + CT_j = 0, \tag{70}$$

dla $j' = \overline{0,127}$, gdzie j' oznacza indeks bitu j po wykonaniu operacji ShiftRows, $y_{j'}$ oznacza j'-ty bit stanu pośredniego pomiędzy warstwą liniową i nieliniową, a CT_j to j-ty bit znanego szyfrogramu. Aby opisać całą ostatnią rundę, w tym podejściu, wymaganych jest 256 zmiennych binarnych dla dwóch stanów pośrednich, 864 zmienne binarne dla linearyzacji równań warstwy nieliniowej i 816 zmiennych dla przedstawienia wartości wielokrotności k_i równań warstwy nieliniowej oraz 128 zmiennych, dla tej samej operacji, dla warstwy liniowej. Ostatecznie, aby przetransformować układ równań opisujących ostatnią rundę, potrzebnych jest 2 064 dodatkowych zmiennych binarnych.

W równaniach (70) bity stanu y to bity wyjścia ze skrzynek podstawieniowych i jednocześnie bity występujące w równaniach wielomianowych opisujących skrzynkę podstawieniową. Można te bity wyznaczyć za pomocą odwrotnej operacji ShiftRows

i podstawić je do równań wielomianowych warstwy nieliniowej. Podejście to przedstawia rysunek 41b, gdzie ShiftRows^{-1} oznacza operację odwrotną do ShiftRows. Wyznaczenie w ten sposób równań ostatniej rundy pozwala zaoszczędzić 128 zmiennych binarnych, przez zrezygnowanie ze stanu pośredniego y, ale jednocześnie zwiększa liczbę jednomianów w równaniach. W znalezionym układzie efektywnym reprezentującym skrzynkę podstawieniową, nie występują jednomiany kwadratowe postaci $y_i y_j$, ale występują jednomiany kwadratowe postaci $x_i y_j$. Dlatego, jeśli w równaniu skrzynki podstawieniowej występuje jednomian kwadratowy $x_i y_j$, to podstawiając $y_j = kr_{T_{j'}} + CT_{j'}$, otrzymujemy:

$$x_i \left(kr_{T_{j'}} + CT_{j'} \right) = x_i kr_{T_{j'}} + x_i CT_{j'}. \tag{71}$$

Czyli każdy jednomian kwadratowy zamieni się na jednomian kwadratowy i co najwyżej jednomian liniowy, w zależności od wartości bitu $CT_{j'}$. Operacja ta nie zmienia liczby różnych jednomianów kwadratowych w układzie, ale zmienia liczbę jednomianów w równaniach układu. W związku z tym liczba dodatkowych zmiennych binarnych dla przedstawienia wartości wielokrotności k_i każdego równania z 51 wzrasta do co najwyżej 64 dla równań pojedynczej skrzynki podstawieniowej. Stąd, aby opisać całą ostatnią rundę w tym podejściu, potrzebnych jest 128 zmiennych binarnych dla wejściowego stanu pośredniego, 864 zmienne dla linearyzacji oraz 1 024 zmienne dla przedstawienia wartości k_i wszystkich równań. Ostateczna liczba dodatkowych zmiennych binarnych, potrzebnych do przedstawienia układu opisującego ostatnią rundę w postaci problemu QUBO, wynosi 2 016.

Podsumowując, cały algorytm szyfrowania T-rundowego szyfru AES, można przedstawić za pomocą układu równań wielomianów stopnia co najwyżej 2, składającego się z:

- 128 równań, postaci (69),
- $208(T-1)$ równań kwadratowych, opisujących warstwy nieliniowe,
- $128(T-1)$ równań liniowych, opisujących warstwy liniowe oraz
- 208 równań kwadratowych, opisujących ostatnią rundę.

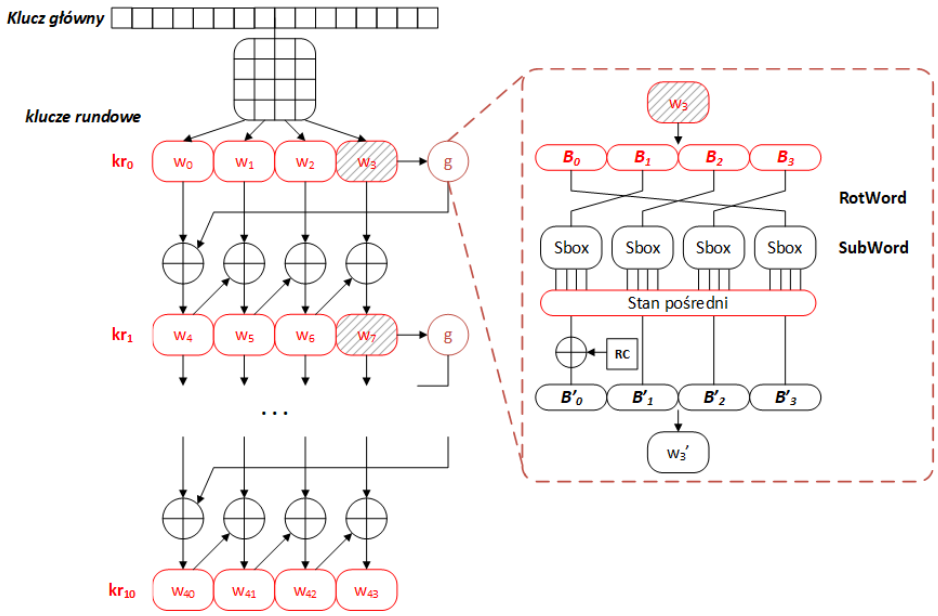
Liczba dodatkowych zmiennych binarnych wymaganych podczas linearyzacji takiego układu wynosi $864T$, a liczba dodatkowych zmiennych binarnych, potrzebnych do przedstawienia wartości wielokrotności k_i wszystkich równań, wynosi 1 152+ 1 120$(T-1)$.

4.3.9 Wyznaczanie równań, nad ciałem $GF(2)$, opisujących algorytm generowania kluczy rundowych szyfru AES

Ponieważ algorytm generowania kluczy rundowych szyfru AES, składa się, tak samo jak algorytm szyfrowania, z warstwy liniowej i nieliniowej, również jego struktura zostanie podzielona za pomocą stanów pośrednich, aby uzyskać równania stopnia co najwyżej dwa. Algorytm generowania kluczy różni się w szczegółach pomiędzy trzema wersjami szyfru AES, toteż zostanie on przeanalizowany osobno dla każdej wersji.

Sposób podzielenia algorytmu generowania kluczy rundowych standardu AES128, przedstawiono na rysunku 42. Za pomocą zmiennych binarnych przedstawia się wszystkie klucze rundowe, przy czym klucz kr_0 jest szukanym kluczem głównym K oraz dla każdej iteracji algorytmu wprowadzany jest 32-bitowy stan pośredni po operacji SubWord, w funkcji g. Dlatego też, aby opisać działanie funkcji g za pomocą równań nad ciałem binarnym, najpierw trzeba opisać relację pomiędzy słowem w_{j-1}, gdy $j(mod\ 4) = 0$ a stanem pośrednim, a następnie relację pomiędzy stanem pośrednim, a słowem w_j. Jak już wcześniej opisano, funkcja g składa się z operacji RotWord, SubWord oraz funkcji xor ze stałą $(RC[i], 0, 0, 0)$. Funkcja RotWord jest funkcją liniową, która nie wpływa ani na stopień wyznaczanych równań, ani na liczbę jednomianów w równaniach, ale zmienia kolejność bajtów, co dla opisujących ją równań oznacza zmianę indeksów zmiennych binarnych. Nieliniowa funkcja SubWord wykorzystuje tę samą skrzynkę podstawieniową co algorytm szyfrowania, dlatego też do jej opisu wykorzystany zostanie znaleziony efektywny układ równań. Dodatkowo na rysunku 42 pominięto operację xor trzech najmniej znaczących bajtów stanu pośredniego, ponieważ operacja xor z wartością 0 nie zmienia wartości wejściowej.

Układ równań nad ciałem binarnym, opisujący pojedynczą iterację algorytmu generowania kluczy rundowych, która generuje cztery słowa w_j, składa się z:



Rysunek 42: Podział algorytmu generowania kluczy rundowych szyfru AES128 za pomocą stanów pośrednich.

- $4 \cdot 13$ równań kwadratowych, opisujących funkcje RotWord oraz SubWord,
- 32 równań opisujących zależność pomiędzy stanem pośrednim, a wyznaczanym słowem w_j, gdy $j(mod\ 4) = 0$, wśród których jest:
  - 8 równań liniowych postaci:

$$w_{j_b} + w_{j-4_b} + x_b + RC_b = 0, \tag{72}$$

    dla $b = \overline{0,7}$ oraz

  - 24 równania liniowe postaci:

$$w_{j_b} + w_{j-4_b} + x_b = 0, \tag{73}$$

    dla $b = \overline{8,31}$, gdzie b oznacza numer bitu danego słowa, x_b oznacza b-ty bit stanu pośredniego oraz RC_b oznacza b-ty bit stałej RC,

- $3 \cdot 32$ równania liniowe, opisujące wyznaczanie słowa w_j, gdy $j\ mod(4) \neq 0$, równania te mają następującą postać:

$$w_{j_b} + w_{j-4_b} + w_{j-1_b} = 0, \tag{74}$$

dla $b = \overline{0,31}$.

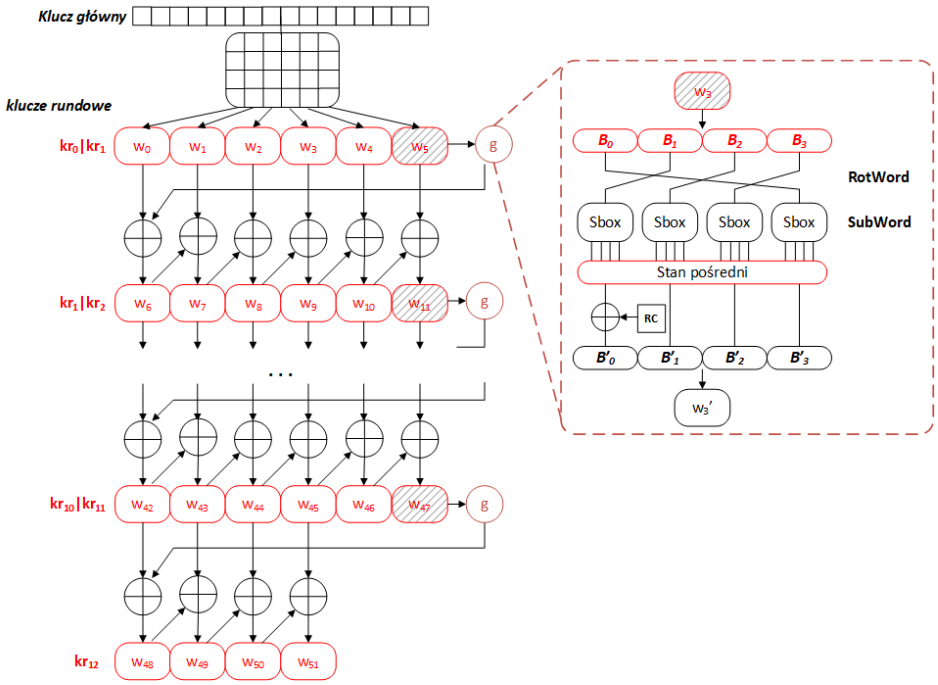
Dla szyfru AES128, którego liczba rund wynosi 10, wymagane jest wykonanie dziesięciu iteracji algorytmu generowania kluczy rundowych, stąd wyżej opisany układ należy wygenerować 10 razy.

Zatem w celu wygenerowania układu równań, opisującego algorytm generowania kluczy rundowych standardu AES128, potrzebnych jest 1 728 zmiennych binarnych dla kluczy rundowych i stanów pośrednich. Wygenerowanych zostanie 1 800 równań, w tym 520 równań kwadratowych oraz 1 280 równań liniowych. Następnie, w celu tranformacji takiego układu do problemu optymalizacyjnego w postaci QUBO, potrzebnych jest dodatkowo 2 160 zmiennych dla linearyzacji oraz co najwyżej 3 400 zmiennych, w zależności od wartości bitu $\boldsymbol{RC}_j$, dla zdefiniowania wartości k_i wszystkich równań.

W analogiczny sposób został wyznaczony układu równań, opisujący algorytm generowania kluczy rundowych, dla wersji szyfru AES192. Na rysunku 43 zaprezentowano podział struktury algorytmu generowania kluczy szyfru AES192, którego liczba rund wynosi 12. W przypadku tej wersji szyfru, funkcja g wykonywana jest dla słowa w_j, gdy $j(mod\ 6) = 0$, dlatego wykonywana jest 8 razy, co zmniejsza liczbę równań kwadratowych w układzie. Sama funkcja g definiowana jest przez takie same równania, jak w przypadku poprzedniej wersji. Dla pojedynczej iteracji, generującej sześć słów w_j, zmienia się liczba liniowych równań (74), opisujących generowanie słów w_j, gdy $j(mod\ 6) \neq 0$, która tutaj wynosi $5 \cdot 32$.

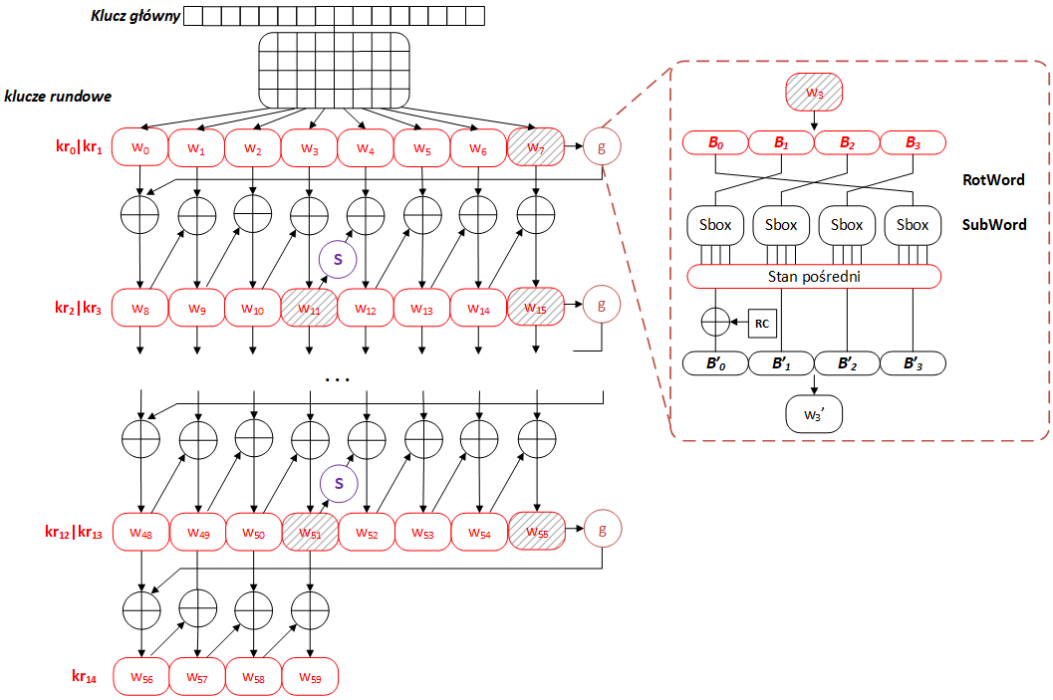
Aby utworzyć układ równań stopnia co najwyżej 2 nad ciałem binarnym, opisujący algorytm generowania kluczy rundowych dla wersji AES192, potrzebnych jest 1 664 zmiennych binarnych, do zdefiniowania 13-tu kluczy rundowych oraz 256 zmiennych dla stanów pośrednich funkcji g. Układ ten składać się będzie z 1 888 równań, w tym 416 kwadratowych i 1 472 liniowych. Następnie, podczas transformacji do problemu QUBO, konieczne będzie zdefiniowanie kolejnych 1 728 zmiennych do przeprowadzenia linearyzacji oraz co najwyżej 3 168 zmiennych do przedstawienia wartości wielokrotności k_i równań.

Ostatnia wersja szyfru AES wymaga klucza głównego długości 256 bitów, który



Rysunek 43: Podział algorytmu generowania kluczy rundowych szyfru AES192 za pomocą stanów pośrednich.

jest jednocześnie dwoma pierwszymi kluczami rundowymi. Na ich podstawie generowanych jest 13 pozostałych kluczy rundowych. Tak samo jak w wersjach poprzednich, również w tej wersji zdefiniowano stan pośredni po warstwie nieliniowej funkcji g, co przedstawiono na rysunku 44. Konstrukcja funkcji g jest identyczna jak w poprzednich wersjach, ale tutaj jest wykonywana 7 razy, gdy $j(mod\ 8) = 0$. Jednak algorytm generowania kluczy szyfru AES256 posiada drugą funkcję, która na rysunku 44 została oznaczona jako S i polega ona na wykonaniu funkcji SubWord dla słowa w_{j-1}, gdy $j(mod\ 8) = 4$. Zatem funkcja ta wykonywana jest 6 razy. Ponieważ funkcja S wykorzystuje operację warstwy nieliniowej, opisywana jest za pomocą równań kwadratowych, definiujących skrzynkę podstawieniową, co zwiększa liczbę równań kwadratowych w układzie. W celu wyznaczenia układu równań, opisującego algorytm generowania kluczy szyfru AES256, należy zdefiniować 1 920 zmiennych dla kluczy rundowych oraz 416 zmiennych dla stanów pośrednich funkcji g oraz S. Wygenerowany układ będzie się składać z 2 148 równań, z których 676 to równania kwadratowe i pozostałe 1 472 to równania liniowe. Podczas transformacji takiego układu do problemu QUBO, linearyzacja będzie wymagać 2 808 dodatkowych zmiennych binarnych, a zdefiniowa-



Rysunek 44: Podział algorytmu generowania kluczy rundowych szyfru AES256 za pomocą stanów pośrednich.

nie wartości k_i wszystkich równań będzie wymagać co najwyżej 4 180 dodatkowych zmiennych.

4.3.10 Konstrukcja układu równań nad ciałem $GF(2)$, opisującego szyfr AES

Aby utworzyć układ równań opisujących cały szyfr AES, należy połączyć w jeden układ równania opisujące algorytm szyfrowania z równaniami opisującymi algorytm generowania kluczy rundowych. Dodatkowo, jak już wspomniano wcześniej, dla wersji szyfru, w których długość klucza jest większa niż długość bloku wejściowego, równania opisujące algorytm szyfrowania należy wygenerować dwa razy, dla dwóch różnych par tekst jawny – szyfrogram. Dlatego też dla szyfru AES128 wygenerowano układ opisujący algorytm szyfrowania dla jednej pary, natomiast dla szyfrów AES192 oraz AES256 dla dwóch różnych par. Następnie otrzymane układy, po połączeniu z odpowiednim układem algorytmu generowania kluczy, przetransformowano do problemu optymalizacyjnego w formie QUBO, zgodnie z zaproponowaną metodą. Uzyskane rozmiary problemu QUBO zaprezentowano w tabeli 9, gdzie dla każdego wa-

riantu szyfru AES, wybrano trzy różne układy definiujące skrzynkę podstawieniową.

Tabela 9: Rozmiar problemu QUBO dla wariantów i wybranych układów dla skrzynki podstawieniowej szyfru AES, opisanego za pomocą równań nad $GF(2)$.

	Układ równań dla SBoxa	*Liczba rund*	*Liczba par*	*Liczba równań*	*Liczba zmiennych binarnych*			
					początkowy układ	*linearyzacja*	*wartości k*	*problem QUBO*
AES128	39 równań	10	1	10 360	4 160	24 000	40 440	68 600
	metoda I/etap 1			5 160		10 800	15 208	30 168
	metoda IV			5 160		10 800	14 568	**29 528**
AES192	39 równań	12	2	20 768	4 864	49 920	83 848	138 632
	metoda I/etap 1			9 952		22 464	31 592	58 920
	metoda IV			9 952		22 464	30 056	**57 384**
AES256	39 równań	14	2	24 556	5 632	60 000	100 099	165 731
	metoda I/etap 1			11 556		27 000	37 427	70 059
	metoda IV			11 556		27 000	35 555	**68 187**

Wiersz *39 równań* przedstawia wyniki transformacji do problemu QUBO układu, w którym skrzynka podstawieniowa jest definiowana za pomocą wszystkich 39 równań kwadratowych, uzyskanych po eliminacji Gaussa. Wiersz *metoda I/etap 1* dotyczy przypadku, gdy skrzynka podstawieniowa zdefiniowana jest za pomocą jednego z 667 układów, posiadających najmniejszą liczbę różnych jednomianów kwadratowych. Układy te zostały znalezione w pierwszym etapie pierwszej metody poszukiwań układu efektywnego dla skrzynki podstawieniowej. Natomiast wiersz *metoda IV* przedstawia wyniki dla układu wygenerowanego dla szyfru AES, w którym skrzynka podstawieniowa zdefiniowana została za pomocą układu efektywnego, znalezionego w metodzie IV.

Łatwo zauważyć, że najgorsze wyniki uzyskano dla przypadku, gdy skrzynkę podstawieniową opisywał układ nadokreślony. Znalezienie układu z minimalną liczbą różnych jednomianów kwadratowych razem z małą liczbą równań pozwoliło uzyskać problem ponad dwa razy mniejszy. Kolejne metody szukania efektywnego układu, poprzez kolejne xorowania równań, doprowadziły do znalezienia układu, który pozwala zmniejszyć ostateczny problem QUBO o ok. 2,5% w stosunku do układu z metody I i o ok. 58%, w stosunku do układu nadokreślonego. Najlepsze uzyskane wyniki zostały oznaczone w tabeli 9 pogrubioną czcionką. Warto również tutaj zauważyć, że w wyniku zastosowania przedstawionej metody transformacji oraz poszukiwań efek-

tywnego układu opisującego skrzynkę podstawieniową, udało się osiągnąć o około 70% mniejszy rozmiar docelowego problemu optymalizacyjnego w formie QUBO, dla każdego wariantu szyfru AES, niż dotychczas opublikowane wyniki, przedstawione w pracy [54].

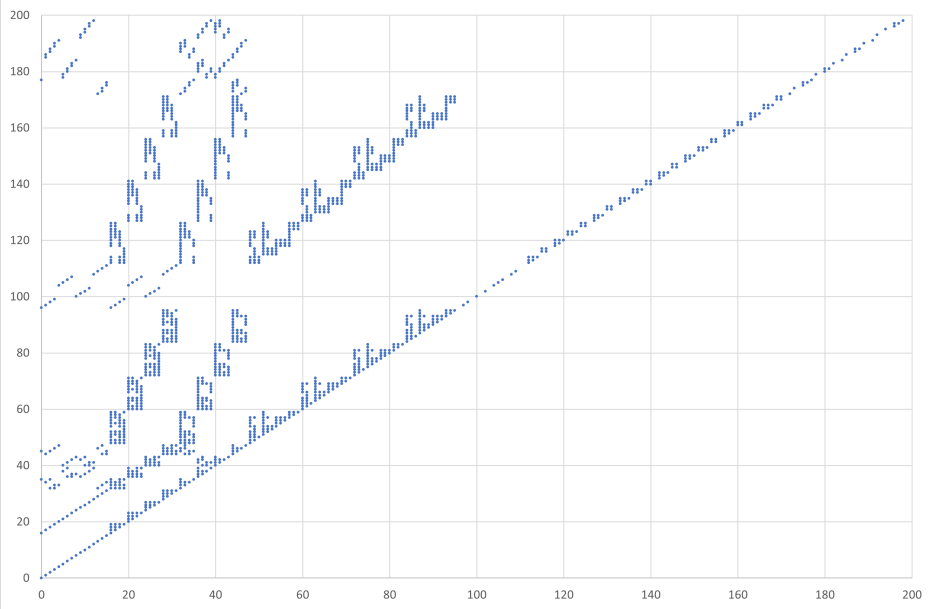
4.4 Atak algebraiczny z wykorzystaniem wyżarzania kwantowego na szyfr AES

Przyjmując, zgodnie z pracą [51], że złożoność rozwiązania problemu optymalizacyjnego w postaci QUBO, składającego się z N zmiennych binarnych, wynosi $O\left(e^{\sqrt{N}}\right)$ ze współczynnikiem równym 1, dla każdego wariantu szyfru AES (na podstawie wyników w tabeli 9) wyznaczono liczbę rund, dla której proponowany atak byłby lepszy niż atak pełnego przeszukiwania. Otrzymane wyniki przedstawiono w tabeli 10, gdzie pokazano liczbę zaatakowanych rund (w nawiasie także jako odsetek standardowej liczby rund szyfru) oraz w osobnej kolumnie pokazano również wyniki dla najlepszego ataku klasycznego. Najlepszy atak klasyczny z odzyskaniem klucza na pełne wersje standardu AES przedstawiono w pracy [7]. Jest to pierwszy atak ze złożonością lepszą niż atak pełnego przeszukiwania, jednak nadal jest on obliczeniowo niewykonalny. Inne ataki na standard AES można przykładowo znaleźć w pracach [5], [6] lub [48].

Tabela 10: Liczba zaatakowanych rund dla najlepszego ataku klasycznego oraz proponowanego ataku, dla poszczególnych wariantów standardu AES.

Wariant standardu AES	*Liczba rund szyfru*	*Liczba zaatakowanych rund dla najlepszego ataku klasycznego*	*Liczba zaatakowanych rund dla proponowanego ataku*
AES128	10	10 (100%)	2 (20%)
AES192	12	12 (100%)	3 (25%)
AES256	14	14 (100%)	6 (43%)

W ramach przeprowadzonych badań najlepszy wynik osiągnięto dla szyfru AES256, dla którego proponowany atak jest lepszy od ataku pełnego przeszukiwania dla 6 z 14 rund, co stanowi 43%. Niestety, dla każdej wersji standardu AES, liczba zaatakowanych rund dla najlepszego ataku klasycznego jest większa niż liczba rund



Rysunek 45: Struktura macierzy Q problemu QUBO wygenerowanego dla jednorundowego szyfru S-AES, opisanego nad $GF(2)$.

dla proponowanego ataku.

W celu sprawdzenia poprawności konstruowania układu oraz transformacji do problemu optymalizacyjnego, przeprowadzono atak na mniejszą instancję szyfru AES, przy użyciu komputera kwantowego D-Wave. Aby była realna możliwość uzyskania rozwiązania, przy dostępnych zasobach zdecydowano się wygenerować problem w postaci QUBO, o rozmiarze ok. 200 zmiennych binarnych. Taki rozmiar problemu uzyskano dla jednej rundy szyfru S-AES o następujących parametrach:

- długość bloku: 16 bitów,
- długość klucza: 16 bitów,
- wielkość skrzynki podstawieniowej: 4×4 bity,
- liczba skrzynek podstawieniowych: 4,
- liczba macierzy MDS: 2,
- liczba rund: $T = 1$.

Rozmiar otrzymanego problemu QUBO wyniósł 199 zmiennych binarnych. Dla wielomianu kary przyjęto wagę równą 1 000. Struktura macierzy Q wygenerowanego

problemu została przedstawiona na rysunku 45. Macierz ma elementy o wartościach od 2 do 3 010, które po skalowaniu w wyżarzaczu, przyjmą wartości z zakresu od $6,6 \cdot 10^{-4}$ do 1. Dodatkowo, liczba wszystkich elementów w górno-trójkątnej macierzy Q wynosi 19 900, z których 1 482 jest niezerowych, co stanowi $7,4\%$, a więc macierz ta jest rzadka.

Oczekiwana wartość energii minimalnej wynosi -26. Czas wyżarzania jakim dysponowano to 20 minut, po którym otrzymano prawidłową minimalną energię oraz odzyskano prawidłowy klucz główny.

5 Analiza szyfru blokowego typu ARX na przykładzie szyfru Speck

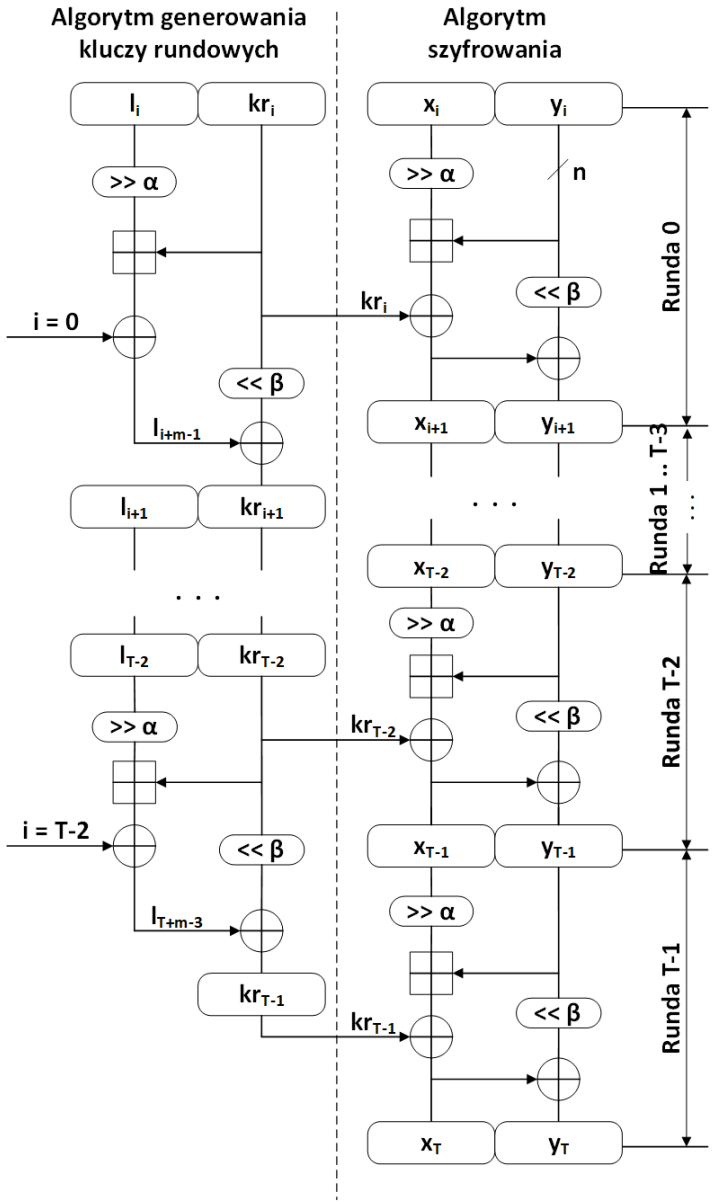
Szyfry typu ARX to rodzaj szyfrów blokowych, które nie zawierają skrzynek podstawieniowych, a warstwa konfuzji w tych szyfrach realizowana jest za pomocą funkcji nieliniowych, takich jak dodawanie modularne lub mnożenie modularne. Jednym z szyfrów tego typu jest szyfr Speck, który należy do rodziny tzw. lekkich szyfrów blokowych, zaprezentowanych w pracy [3] jako wysoko zoptymalizowane szyfry blokowe, przeznaczone dla implementacji programowych lub sprzętowych. W niniejszej rozprawie szyfr ten zostanie przeanalizowany pod względem możliwości przekształcenia go do problemu QUBO.

Tabela 11: Parametry szyfru Speck2n/mn. *Źródło: [3]*

Rozmiar bloku ($2n$)	*Rozmiar klucza* (mn)	*Rozmiar słowa* (n)	*Liczba słów klucza* (m)	*Liczba bitów dla rotacji w prawo* (α)	*Liczba bitów dla rotacji w lewo* (β)	*Liczba rund* (T)
32	64	16	4	7	2	22
48	72 96	24	3 4	8	3	22 23
64	96 128	32	3 4	8	3	26 27
96	96 144	48	2 3	8	3	28 29
128	128 192 256	64	2 3 4	8	3	32 33 34

Niech Speck2n/mn oznacza instancję szyfru Speck, gdzie $2n$ oznacza długość bloku wejściowego, n oznacza długość słowa, a mn jest długością klucza głównego. Ogólna struktura szyfru Speck2n/mn została przedstawiona na rysunku 46, gdzie T oznacza liczbę rund. W zależności od instancji, liczba rund zawiera się w przedziale od 22 do 34. Parametry poszczególnych instancji szyfru Speck2n/mn przedstawiono w tabeli 11. Szyfr wykorzystuje następujące operacje na n-bitowych słowach:

- bitowa operacja xor, oznaczona jako $\oplus$,



Rysunek 46: Ogólny schemat budowy szyfru Speck. *Źródło: na podstawie [3]*

- dodawanie modulo 2^n, oznaczone jako $\boxplus$,

- rotacje, w prawą stronę o α bitów i w lewą stronę o β bitów, oznaczone odpowiednio jako $\gg \alpha$ oraz $\ll \beta$.

Operacją nieliniową jest operacja dodawania modulo 2^n, a pozostałe operacje są operacjami liniowymi.

5.1 Algorytm szyfrowania Speck

Funkcja rundy algorytmu szyfrowania Speck$2n/mn$ jest odwzorowaniem $R : \mathbb{Z}_{2^n} \times \mathbb{Z}_{2^n} \to \mathbb{Z}_{2^n} \times \mathbb{Z}_{2^n}$, zdefiniowanym następująco:

$$R\left(x_{i+1}, y_{i+1}\right) = \left(\left(\left(x_i \gg \alpha\right) \boxplus y_i\right) \oplus kr_i, \left(y_i \ll \beta\right) \oplus \left(\left(x_i \gg \alpha\right) \boxplus y_i\right) \oplus kr_i\right), \quad (75)$$

gdzie kr_i jest kluczem rundowym, a i jest numerem rundy.

Blok wejściowy, długości $2n$ bitów, danej rundy dzielony jest na dwa n-bitowe słowa, gdzie n najmniej znaczących bitów tworzy prawe słowo y_i, a n najbardziej znaczących bitów bloku tworzy lewe słowo x_i. Na lewym słowie najpierw wykonywana jest rotacja w prawo o α bitów. Następnie do wyniku dodawane jest prawe słowo y_i, a na otrzymanej sumie wykonywana jest operacja bitowa xor z kluczem rundowym kr_i. W ten sposób otrzymywane jest lewe słowo x_{i+1} stanu pośredniego. Jednocześnie, na prawym słowie y_i bloku wejściowego wykonywana jest rotacja w lewo o β bitów, a następnie wykonywana jest bitowa operacja xor ze słowem x_{i+1}, czyli wynikiem przetwarzania lewego słowa. W ten sposób otrzymywane jest prawe słowo y_{i+1} stanu pośredniego.

5.2 Algorytm generowania kluczy rundowych szyfru Speck

Klucz główny K, o długości mn, w algorytmie generowania kluczy rundowych szyfru Speck dzielony jest na m n-bitowych słów, gdzie n najmniej znaczących bitów tworzy klucz pierwszej rundy, a kolejnych n-bitów tworzy kolejne słowa l_i w następujący sposób: $K = [l_{m-2}, ..., l_0, kr_0]$, gdzie $l_i, kr_0 \in \mathbb{Z}_{2^n}$. Sam algorytm generowania kluczy rundowych wykorzystuje funkcję rundy R, przedstawioną równaniem (75), w której argument x_i zastąpiony jest słowem l_i, argument y_i zastąpiony jest kluczem rundowym kr_i, a klucz rundowy kr_i zastąpiony jest numerem rundy i, co pokazano za pomocą równania (76).

$$R\left(l_{i+m-1}, kr_{i+1}\right) = \left(\left(\left(l_i \gg \alpha\right) \boxplus kr_i\right) \oplus i, \left(kr_i \ll \beta\right) \oplus \left(\left(l_i \gg \alpha\right) \boxplus kr_i\right) \oplus i\right). \quad (76)$$

5.3 Analiza przedstawienia szyfru Speck za pomocą układu równań wielomianowych o wielu zmiennych nad $\mathbb{Z}_{2^n}$

Naturalną strukturą algebraiczną, nad którą można przedstawić równania opisujące szyfr Speck, jest pierścień $\mathbb{Z}_{2^n}$. W niniejszym rozdziale przeanalizowane zostaną sposoby opisu szyfru Speck, w celu znalezienia takiego, który pozwoli uzyskać możliwie najmniejszy docelowy problem QUBO.

Załóżmy, że bity kluczy rundowych są przedstawione za pomocą zmiennych binarnych, a równania wielomianów wielu zmiennych generowane są nad pierścieniem $\mathbb{Z}_{2^n}$, po jednym równaniu dla lewego i prawego słowa stanu pośredniego każdej rundy szyfru. Operacje szyfru Speck$2n/mn$ realizowane są na n-bitowych słowach w następujący sposób. Niech a i b oznaczają n-bitowe słowa postaci:

$$a = 2^{n-1}a_{n-1} + 2^{n-2}a_{n-2} + \cdots + 2a_1 + a_0 = \sum_{j=0}^{n-1} 2^j a_j,$$

$$b = 2^{n-1}b_{n-1} + 2^{n-2}b_{n-2} + \cdots + 2b_1 + b_0 = \sum_{j=0}^{n-1} 2^j b_j.$$

gdzie a_j i b_j są bitami słów. Rotacja słowa a w prawo o α bitów oraz rotacja słowa b w lewo o β bitów wykonywana jest zgodnie z równaniami, odpowiednio (77) i (78).

$$\begin{aligned} a \gg \alpha &= 2^{n-1}a_{(n-1+\alpha) \bmod n} + 2^{n-2}a_{(n-2+\alpha) \bmod n} + \cdots + 2a_{\alpha+1} + a_\alpha = \\ &= \sum_{j=0}^{n-1} 2^j a_{(j+\alpha) \bmod n}, \end{aligned} \tag{77}$$

$$\begin{aligned} b \ll \beta &= 2^{n-1}b_{(n-1-\beta) \bmod n} + 2^{n-2}b_{(n-2-\beta) \bmod n} + \cdots + 2b_{n-\beta+1} + b_{n-\beta} = \\ &= \sum_{j=0}^{n-1} 2^j b_{(j-\beta) \bmod n}. \end{aligned} \tag{78}$$

Bitową operację xor dwóch bitów a_j i b_j w pierścieniu, można przedstawić jako $a_j \oplus b_j = a_j + b_j - 2a_j b_j$. Dlatego też bitowa operacja xor dwóch n-bitowych słów a i b realizowana jest zgodnie z równaniem (79). Jak można łatwo zauważyć, stopień wielomianu będący wynikiem operacji xor jest sumą stopnia wielomianu słowa a i stopnia

wielomianu słowa b:

$$\begin{aligned} a \oplus b = 2^{n-1}\left(a_{n-1}+b_{n-1}-2a_{n-1}b_{n-1}\right)+2^{n-2}\left(a_{n-2}+b_{n-2}-2a_{n-2}b_{n-2}\right)+ \\ +\cdots+2\left(a_1+b_1-2a_1b_1\right)+a_0+b_0-2a_0b_0=\sum_{j=0}^{n-1}2^j\left(a_j+b_j-2a_jb_j\right). \end{aligned} \tag{79}$$

W celu znalezienia układu równań wielomianowych, opisujących szyfr Speck, który po przekształceniach da jak najmniejszy problem QUBO, przeanalizowano dwa miejsca przecięcia ścieżki przetwarzania, definiujące stan pośredni, wyznaczający zakres rundy. Na rysunku 47 przedstawiono dwa schematy i-tej rundy algorytmu szyfrowania szyfru Speck$2n/mn$, gdzie słowa x_i i y_i są wejściem do i-tej rundy, kr_i jest kluczem rundowym, a słowa x_{i+1} i y_{i+1} są wyjściem z i-tej rundy. Przedstawmy te słowa za pomocą następujących sum bitów:

$$x_i = 2^{n-1}x_{i_{n-1}}+2^{n-2}x_{i_{n-2}}+\cdots+2x_{i_1}+x_{i_0}=\textstyle\sum_{j=0}^{n-1}2^j x_{i_j},$$

$$y_i = 2^{n-1}y_{i_{n-1}}+2^{n-2}y_{i_{n-2}}+\cdots+2y_{i_1}+y_{i_0}=\textstyle\sum_{j=0}^{n-1}2^j y_{i_j},$$

$$kr_i = 2^{n-1}kr_{i_{n-1}}+2^{n-2}kr_{i_{n-2}}+\cdots+2kr_{i_1}+kr_{i_0}=\textstyle\sum_{j=0}^{n-1}2^j kr_{i_j},$$

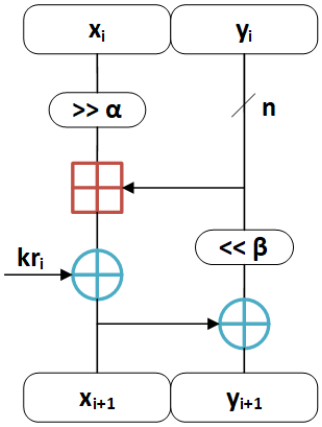
$$x_{i+1} = 2^{n-1}x_{i+1_{n-1}}+2^{n-2}x_{i+1_{n-2}}+\cdots+2x_{i+1_1}+x_{i+1_0}=\textstyle\sum_{j=0}^{n-1}2^j x_{i+1_j},$$

$$y_{i+1} = 2^{n-1}y_{i+1_{n-1}}+2^{n-2}y_{i+1_{n-2}}+\cdots+2y_{i+1_1}+y_{i+1_0}=\textstyle\sum_{j=0}^{n-1}2^j y_{i+1_j},$$

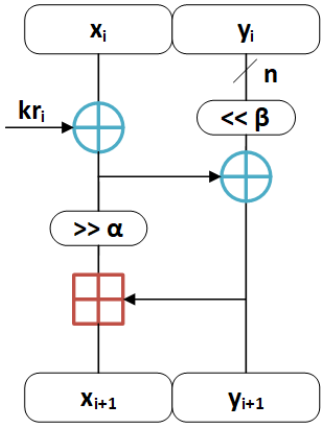
gdzie x_{i_j} oznacza j-ty bit słowa x_i.

5.3.1 Wyznaczanie równań nad pierścieniem $\mathbb{Z}_{2^n}$, opisujących algorytm szyfrowania według dokumentacji szyfru Speck

Rozważmy jedną rundę algorytmu szyfrowania szyfru Speck$2n/mn$, której zakres jest zgodny z dokumentacją, co przedstawia rysunek 47a. W tym podejściu operacja xor wykonywana jest po operacji dodawania modulo 2^n. Ponieważ operacja xor jest operacją bitową, w celu jej wykonania każdy bit otrzymanej sumy musi zostać zdefiniowany za pomocą wielomianu uwzględniającego bity przeniesienia. Przedstawmy operację



(a) Jedna runda szyfru Speck$2n/mn$ według dokumentacji.



(b) Jedna runda szyfru Speck$2n/mn$ według zaproponowanego przesunięcia.

Rysunek 47: Analizowane zakresy jednej rundy algorytmu szyfrowania szyfru Speck$2n/mn$.

dodawania dwóch n-bitowych słów a i b za pomocą równania:

$$(a+b)\ mod\ 2^n = 2^{n-1}s_{n-1} + 2^{n-2}s_{n-2} + \cdots + 2s_1 + s_0 = \sum_{j=0}^{n-1} 2^j s_j, \tag{80}$$

w którym każdy bit s_j sumy jest reprezentowany przez wielomian. W celu wyznaczenia poszczególnych wielomianów s_j, wykorzystano algorytm 1, opisujący działanie pełnego n-bitowego sumatora, w którym operacja xor wykonywana jest zgodnie z równaniem (79). Jak wynika z tego algorytmu, j-ty bit s_j sumy zależy od wszystkich mniej

Algorytm 1: Algorytm pełnego, n-bitowego sumatora.

Input : $a = \sum_{j=0}^{n-1} 2^j a_j$, $b = \sum_{j=0}^{n-1} 2^j b_j$
Output: $(a+b)\ mod\ 2^n = \sum_{j=0}^{n-1} 2^j s_j$

```
for j = 0 up to n − 1 do
    if (j = 0) then
        s_0 ← a_0 ⊕ b_0
        c_0 ← a_0 b_0
    else
        s_j ← c_{j−1} ⊕ a_j ⊕ b_j
        c_j ← a_j b_j + c_{j−1}(a_j ⊕ b_j)
return Σ_{j=0}^{n−1} 2^j s_j
```

znaczących bitów obu składników, czyli od bitów od a_0 do a_j oraz od b_0 do b_j. Przy-

kładowo, drugi bit s_2 sumy można przedstawić za pomocą następującego wielomianu:

$$s_2 = a_2 + b_2 + a_1 b_1 - 2a_2 b_2 + a_0 a_1 b_0 + a_0 b_0 b_1 - 2a_1 a_2 b_1 - 2a_1 b_1 b_2 - 2a_0 a_1 a_2 b_0 + \\ -2a_0 a_2 b_0 b_1 - 2a_0 a_1 b_0 b_2 - 2a_0 b_0 b_1 b_2 - 2a_0 a_1 b_0 b_1 + 4a_1 a_2 b_1 b_2 + 4a_0 a_1 a_2 b_0 b_1 + \\ +4a_0 a_1 a_2 b_0 b_2 + 4a_0 a_1 b_0 b_1 b_2 + 4a_0 a_2 b_0 b_1 b_2 - 8a_0 a_1 a_2 b_0 b_1 b_2,$$

którego stopień wynosi 6. Wielomian ten składa się z jednego jednomianu stopnia 6, czterech jednomianów stopnia 5, sześciu jednomianów stopnia 4, czterech jednomianów stopnia 3, dwóch jednomianów stopnia 2 oraz dwóch jednomianów stopnia 1. Ogólnie, stopień wielomianu dla j-tego bitu sumy wynosi $2(j+1)$, stąd dla bloku wejściowego o długości $2n$, stopień wielomianu reprezentującego najbardziej znaczący bit sumy wynosi $2n$.

Przejdźmy teraz do opisania pojedynczej rundy szyfru Speck$2n/mn$ za pomocą równań, uwzględniając omówiony wyżej sposób realizacji poszczególnych operacji. Lewe słowo pojedynczej rundy algorytmu szyfrowania można przedstawić za pomocą równania (81) lub równoważnie za pomocą równania (82), gdzie dane słowa przedstawiono według wcześniej zdefiniowanych sum bitów.

$$x_{i+1} = \left(\left(x_i \gg \alpha\right) + y_i\right) \ mod\ 2^n \oplus kr_i, \tag{81}$$

$$\sum_{j=0}^{n-1} 2^j x_{i+1_j} = \left(\left(\sum_{j=0}^{n-1} 2^j x_{i_j} \gg \alpha\right) + \sum_{j=0}^{n-1} 2^j y_{i_j}\right) \ mod\ 2^n \oplus \sum_{j=0}^{n-1} 2^j kr_{i_j}. \tag{82}$$

Zgodnie z funkcją algorytmu szyfrowania, pierwszą operacją wykonywaną na lewym słowie jest operacja rotacji w prawo o α bitów, wykonana zgodnie z równaniem (77). W wyniku tej operacji otrzymujemy równanie:

$$\sum_{j=0}^{n-1} 2^j x_{i+1_j} = \left(\left(\sum_{j=0}^{n-1} 2^j x_{i_{(j+\alpha)\ mod\ n}}\right) + \sum_{j=0}^{n-1} 2^j y_{i_j}\right) \ mod\ 2^n \oplus \sum_{j=0}^{n-1} 2^j kr_{i_j}. \tag{83}$$

Następnie wykonywane jest dodawanie modulo 2^n, którego wynik przedstawmy zgodnie z równaniem (80). Po tej operacji, równanie opisujące lewe słowo można przedstawić za pomocą następującego równania:

$$\sum_{j=0}^{n-1} 2^j x_{i+1_j} = \sum_{j=0}^{n-1} 2^j s_{i_j} \oplus \sum_{j=0}^{n-1} 2^j kr_{i_j}, \tag{84}$$

gdzie bity s_{i_j} są bitami sumy słów $\sum_{j=0}^{n-1} 2^j x_{i_{(j+\alpha) \bmod n}}$ i $\sum_{j=0}^{n-1} 2^j y_{i_j}$. Ostatnią operacją wykonywaną na lewym słowie jest bitowa operacja xor, po przeprowadzeniu której otrzymujemy równanie:

$$\sum_{j=0}^{n-1} 2^j x_{i+1_j} = \sum_{j=0}^{n-1} 2^j \left(s_{i_j} + kr_{i_j} - 2 s_{i_j} kr_{i_j} \right). \tag{85}$$

Przenosząc lewą stronę równania (85) na prawo i przyrównując do zera, otrzymujemy finalne równanie (86), opisujące lewą ścieżkę przetwarzania bloku wejściowego dla pojedynczej rundy algorytmu szyfrowania szyfru Speck$2n/mn$.

$$\sum_{j=0}^{n-1} 2^j \left(s_{i_j} + kr_{i_j} - 2 s_{i_j} kr_{i_j} - x_{i+1_j} \right) = 0. \tag{86}$$

Analogiczny opis przekształceń można przedstawić dla prawej ścieżki przetwarzania bloku wejściowego pojedynczej rundy algorytmu szyfrowania szyfru Speck$2n/mn$. Prawe słowo każdego stanu pośredniego można opisać za pomocą równania (87) lub równoważnie za pomocą równania (88).

$$y_{i+1} = \left(y_i \ll \beta \right) \oplus x_{i+1}, \tag{87}$$

$$\sum_{j=0}^{n-1} 2^j y_{i+1_j} = \left(\sum_{j=0}^{n-1} 2^j y_{i_j} \ll \beta \right) \oplus \sum_{j=0}^{n-1} 2^j x_{i+1_j}. \tag{88}$$

Przekształcając równanie (88) zgodnie z równaniem (78) dla rotacji i równaniem (79) dla operacji xor oraz przenosząc lewą stronę równania na prawo i przyrównując do zera, otrzymujemy następujące równanie, opisujące prawą ścieżkę przetwarzania:

$$\sum_{j=0}^{n-1} 2^j \left(y_{i_{(j-\beta) \bmod n}} + x_{i+1_j} - 2 y_{i_{(j-\beta) \bmod n}} x_{i+1_j} - y_{i+1_j} \right) = 0. \tag{89}$$

Przeanalizujmy otrzymane równania w kontekście kolejnych ich przekształceń do problemu QUBO, a przede wszystkim w kontekście ich linearyzacji. Zacznijmy od równania (86) dla lewego słowa. Jak pokazano wcześniej, stopień wielomianu reprezentującego najbardziej znaczący bit $s_{i_{n-1}}$ sumy wynosi $2n$, a po wykonaniu operacji xor z kluczem rundowym, który zgodnie z założeniem reprezentowany jest przez zmienne binarne, stopień otrzymanego równania (86) wyniesie $2n+1$. Aby wyznaczyć

liczbę dodatkowych zmiennych binarnych wymaganych podczas procesu linearyzacji, przeanalizujmy postać wielomianów, reprezentujących bity sumy modulo 2^n. Wykorzystując wcześniej pokazaną postać wielomianu dla drugiego bitu sumy (s_2), możemy wyznaczyć postać składnika sumy w równaniu (86), dla $j = 2$. Składnik ten ma postać $4(s_{i_2} + kr_{i_2} - 2s_{i_2}kr_{i_2} - x_{i+1_2})$, gdzie:

$$
\begin{aligned}
& s_{i_2} = x_{i_{(2+\alpha)}} + y_{i_2} + x_{i_{(1+\alpha)}}y_{i_1} - 2x_{i_{(2+\alpha)}}y_{i_2} + x_{i_\alpha}x_{i_{(1+\alpha)}}y_{i_0} + x_{i_\alpha}y_{i_0}y_{i_1} - 2x_{i_{(1+\alpha)}}x_{i_{(2+\alpha)}}y_{i_1} + \\
& -2x_{i_{(1+\alpha)}}y_{i_1}y_{i_2} - 2x_{i_\alpha}x_{i_{(1+\alpha)}}x_{i_{(2+\alpha)}}y_{i_0} - 2x_{i_\alpha}x_{i_{(2+\alpha)}}y_{i_0}y_{i_1} - 2x_{i_\alpha}x_{i_{(1+\alpha)}}y_{i_0}y_{i_2} + \\
& -2x_{i_\alpha}y_{i_0}y_{i_1}y_{i_2} - 2x_{i_\alpha}x_{i_{(1+\alpha)}}y_{i_0}y_{i_1} + 4x_{i_{(1+\alpha)}}x_{i_{(2+\alpha)}}y_{i_1}y_{i_2} + 4x_{i_\alpha}x_{i_{(1+\alpha)}}x_{i_{(2+\alpha)}}y_{i_0}y_{i_1} + \\
& +4x_{i_\alpha}x_{i_{(1+\alpha)}}x_{i_{(2+\alpha)}}y_{i_0}y_{i_2} + 4x_{i_\alpha}x_{i_{(1+\alpha)}}y_{i_0}y_{i_1}y_{i_2} + 4x_{i_\alpha}x_{i_{(2+\alpha)}}y_{i_0}y_{i_1}y_{i_2} + \\
& -8x_{i_\alpha}x_{i_{(1+\alpha)}}x_{i_{(2+\alpha)}}y_{i_0}y_{i_1}y_{i_2}.
\end{aligned}
$$

Po wykonaniu operacji mnożenia wielomianu s_{i_2} przez zmienną binarną kr_{i_2}, analizowany składnik jest wielomianem stopnia 7, składającym się z jednego jednomianu stopnia 7, pięciu różnych jednomianów stopnia 6, dziesięciu różnych jednomianów stopnia 5, dziesięciu różnych jednomianów stopnia 4, sześciu różnych jednomianów stopnia 3, czterech różnych jednomianów stopnia 2 oraz czterech różnych jednomianów stopnia 1. Ponieważ dla jednomianu stopnia d podczas linearyzacji potrzebnych jest $d-1$ dodatkowych zmiennych binarnych, tylko dla tego składnika podczas linearyzacji wymaganych jest 110 dodatkowych zmiennych binarnych. Spróbujmy wyznaczyć górną granicę liczby zmiennych binarnych potrzebnych do linearyzacji równania lewego słowa. Niech z^d oznacza jednomian stopnia d. Wtedy liczbę różnych jednomianów danego stopnia w powyższym wielomianie, reprezentującym składnik $kr_{i_2}s_{i_2}$, można przedstawić jako $z^7 + 5z^6 + 10z^5 + 10z^4 + 6z^3 + 4z^2 + 4z$, gdzie współczynnik przy zmiennej z^d oznacza liczbę różnych jednomianów stopnia d. Aby wyznaczyć liczbę różnych jednomianów j-tego wielomianu równania (86), przeanalizujmy algorytm 1, w którym w każdym kroku wyznaczane są wielomiany c_j oraz s_j. Liczbę różnych jednomianów stopnia d w wielomianie $c_j = a_jb_j + c_{j-1}(a_j \oplus b_j) = a_jb_j + c_{j-1}(a_j + b_j - 2a_jb_j)$ można przedstawić jako $z^2 + c_{j-1}(2z + z^2)$. Podobnie, dla wielomianu $s_j = c_{j-1} \oplus a_j \oplus b_j = c_{j-1} \oplus (a_j + b_j - 2a_jb_j) = c_{j-1} + (a_j + b_j - 2a_jb_j) - 2c_{j-1}(a_j + b_j - 2a_jb_j)$ liczbę jego różnych jednomianów stopnia d przedstawia wielomian $c_{j-1} + 2z + z^2 + c_{j-1}(2z + z^2)$. Zatem dla j-tego bitu równania (86) i-tej rundy,

reprezentowanego przez wielomian $s_{i_j} + kr_{i_j} - 2s_{i_j}kr_{i_j} - x_{i+1_j}$, liczbę jego różnych jednomianów stopnia d można przedstawić jako $2z + c_{j-1} + 2z + z^2 + c_{j-1}(2z + z^2) + z(c_{j-1} + 2z + z^2 + c_{j-1}(2z + z^2)) = z^3 + 3z^2 + 4z + c_{j-1}(1 + 2z + z^2) + c_{j-1}(z + 2z + z^2)$. Ostatecznie, dla n bitowego słowa szyfru Speck, liczba różnych jednomianów danego stopnia w równaniu (86), opisujących lewe słowo pojedynczej rundy, wynosi co najwyżej tyle, ile współczynnik przy zmiennej z danego stopnia w wielomianie:

$$\sum_{j=0}^{n-1} \left(z^3 + 3z^2 + 4z + \left(1 + 2z + z^2\right) c_{j-1} + \left(z + 2z^2 + z^3\right) c_{j-1}\right),$$

gdzie $c_j = z^2 + (2z + z^2)c_{j-1}$ oraz $c_j = 0$ dla $j < 0$. Dla każdego jednomianu z^d podczas linearyzacji potrzebnych jest $d - 1$ zmiennych binarnych, stąd można oszacować że, przykładowo, dla $n = 16$, liczba zmiennych binarnych wynosi co najwyżej $1,7 \cdot 10^9$.

W przypadku równania (89) prawego słowa, jego stopień jest stały i wynosi 2. Wielomian opisujący liczbę jednomianów danego stopnia tego równania ma postać $\sum_{j=0}^{n-1}(3z + z^2) = 3nz + nz^2$, co oznacza, że równanie to składa się z co najwyżej $3n$ różnych jednomianów stopnia 1 oraz n różnych jednomianów stopnia 2. Stąd liczba zmiennych binarnych wymaganych podczas linearyzacji wynosi co najwyżej n.

5.3.2 Wyznaczanie równań nad pierścieniem $\mathbb{Z}_{2^n}$, opisujących algorytm szyfrowania z przesuniętym zakresem rundy szyfru Speck

Teraz rozważmy jedną rundę algorytmu szyfrowania szyfru Speck$2n/mn$, której zakres został przesunięty, co pokazano na rysunku 47b. W tym podejściu operacja dodawania modulo 2^n wykonywana jest po bitowej operacji xor, natomiast równania wielomianowe o wielu zmiennych, tak samo jak w podejściu poprzednim, generowane są nad pierścieniem $\mathbb{Z}_{2^n}$, po jednym równaniu dla lewego i prawego słowa stanu pośredniego. Operacje rotacji oraz bitowa operacja xor szyfru Speck$2n/mn$ wykonywana jest identycznie jak w poprzednim podejściu, natomiast operacja dodawania modularnego realizowana jest w pierścieniu $\mathbb{Z}_{2^n}$ jako dodawanie arytmetyczne, przedstawione równaniem (90). Suma ta przyjmuje maksymalną wartość dla składników $a = b = 2^n - 1$ i wynosi $2^{n+1} - 2$. Dlatego redukcja modularna polega na odjęciu, co najwyżej raz,

modułu 2^n od sumy $a+b$.

$$\begin{aligned}(a+b)\ mod\ 2^n &= 2^{n-1}(a_{n-1}+b_{n-1})+2^{n-2}(a_{n-2}+b_{n-2})+\\ &+\cdots+2(a_1+b_1)+a_0+b_0-c\cdot 2^n=\sum_{j=0}^{n-1}2^j(a_j+b_j)-c\cdot 2^n,\end{aligned} \tag{90}$$

gdzie c jest bitem przeniesienia sumy.

W tym podejściu równanie reprezentujące lewe słowo jednej rundy algorytmu szyfrowania można przedstawić za pomocą równania:

$$x_{i+1}=\left(\left(\left(x_i\oplus kr_i\right)\gg\alpha\right)+y_{i+1}\right)\ mod\ 2^n. \tag{91}$$

Wykonując operację dodawania modularnego zgodnie z równaniem (90) oraz przenosząc lewą stronę równania (91) na prawo i przyrównując do zera, otrzymujemy następujące równanie, opisujące lewą ścieżkę przetwarzania algorytmu szyfrowania:

$$\left(\left(\left(x_i\oplus kr_i\right)\gg\alpha\right)+y_{i+1}\right)-x_{i+1}-c\cdot 2^n=0. \tag{92}$$

Ponieważ podczas dalszej transformacji do problemu QUBO równanie (92) zostanie przekształcone do równania ze zmiennymi binarnymi i współczynnikami całkowitymi, które powinno przystawać do $0(mod\ 2^n)$, składnik $-c\cdot 2^n$ można pominąć, otrzymując równanie (93) lub równoważne równanie (94), gdzie słowa przedstawiono za pomocą sum bitów.

$$\left(\left(\left(x_i\oplus kr_i\right)\gg\alpha\right)+y_{i+1}\right)-x_{i+1}=0. \tag{93}$$

$$\left(\left(\left(\sum_{j=0}^{n-1}2^j x_{i_j}\oplus\sum_{j=0}^{n-1}2^j kr_{i_j}\right)\gg\alpha\right)+\sum_{j=0}^{n-1}2^j y_{i+1_j}\right)-\sum_{j=0}^{n-1}2^j x_{i+1_j}=0. \tag{94}$$

Po wykonaniu operacji xor zgodnie z równaniem (79) oraz po wykonaniu rotacji w prawo (równanie (77)) otrzymujemy finalne równanie dla lewej ścieżki przetwarzania pojedynczej rundy algorytmu szyfrowania Speck, następującej postaci:

$$\sum_{j=0}^{n-1}2^j\left(x_{i_{(j+\alpha)\ mod\ n}}+kr_{i_{(j+\alpha)\ mod\ n}}-2x_{i_{(j+\alpha)\ mod\ n}}kr_{i_{(j+\alpha)\ mod\ n}}+y_{i+1_j}-x_{i+1_j}\right)=0. \tag{95}$$

Podobnie, równanie reprezentujące prawe słowo jednej rundy algorytmu szyfro-

wania można przedstawić za pomocą równania (96) lub za pomocą bitów, jako równanie (97):

$$y_{i+1} = ((y_i \ll \beta) \oplus x_i \oplus kr_i), \tag{96}$$

$$\sum_{j=0}^{n-1} 2^j y_{i+1_j} = \left(\sum_{j=0}^{n-1} 2^j y_{i_j} \ll \beta \right) \oplus \sum_{j=0}^{n-1} 2^j x_{i_j} \oplus \sum_{j=0}^{n-1} 2^j kr_{i_j}. \tag{97}$$

Po wykonaniu rotacji w lewo, operacji xor oraz przeniesieniu lewej strony na prawą i przyrównaniu do zera, otrzymujemy równanie, opisujące prawą ścieżkę przetwarzania pojedynczej rundy algorytmu szyfrowania, postaci:

$$\begin{aligned} &\sum_{j=0}^{n-1} 2^j \Big(y_{i_{(j-\beta) \bmod n}} + x_{i_j} - 2y_{i_{(j-\beta) \bmod n}} x_{i_j} + kr_{i_j} - 2kr_{i_j} y_{i_{(j-\beta) \bmod n}} + \\ &-2kr_{i_j} x_{i_j} + 4kr_{i_j} x_{i_j} y_{i_{(j-\beta) \bmod n}} - y_{i+1_j} \Big) = 0. \end{aligned} \tag{98}$$

Ponownie przeanalizujmy otrzymane równania dla proponowanego podejścia w kontekście ich linearyzacji. Na początku należy podkreślić, że stopień otrzymanych równań jest stały i niezależny od długości bloku wejściowego. Stopień równania (95) dla lewej ścieżki przetwarzania pojedynczej rundy wynosi 2. Maksymalną liczbę jednomianów danego stopnia tego równania można przedstawić za pomocą wielomianu $\sum_{j=0}^{n-1}(4z + z^2) = 4nz + nz^2$, czyli równanie (95) składa się z co najwyżej $4n$ różnych jednomianów stopnia 1 oraz n różnych jednomianów stopnia 2. Łatwo zauważyć, że liczba zmiennych binarnych wymaganych podczas linearyzacji wynosi co najwyżej n. Podobną analizę przeprowadźmy dla prawej ścieżki przetwarzania, dla której stopień równania (98) wynosi 3. Wielomian opisujący maksymalną liczbę jednomianów w tym równaniu ma postać $\sum_{j=0}^{n-1}(4z + 3z^2 + z^3) = 4nz + 3nz^2 + nz^3$, a więc równanie to składa się z co najwyżej $4n$ różnych jednomianów stopnia 1, $3n$ różnych jednomianów stopnia 2 oraz n różnych jednomianów stopnia 3. Jednak jednomian stopnia trzy $kr_{i_j} x_{i_j} y_{i_{(j-\beta) \bmod n}}$, to iloczyn jednomianu kwadratowego $kr_{i_j} x_{i_j}$ oraz zmiennej $y_{i_{(j-\beta) \bmod n}}$, więc podczas linearyzacji w obu jednomianach można podstawić tę samą nową zmienną, wykorzystując tym samym nie trzy, a dwie dodatkowe zmienne binarne. Dlatego w procesie linearyzacji wymaganych jest co najwyżej $4n$ dodatkowych zmiennych binarnych.

Podsumowując powyższą analizę wyznaczania równań dla jednej rundy algorytmu szyfrowania Speck, oszacowano liczbę zmiennych binarnych potrzebnych w procesie linearyzacji dla poszczególnych długości słowa zgodnych z parametrami szyfru. Otrzymane wyniki zaprezentowano w tabeli 12 jako sumę liczby potrzebnych zmiennych dla lewego i prawego słowa. Można z niej wywnioskować, że dla zaproponowanego zakresu rundy, ostateczny problem QUBO będzie miał znacznie mniejszy rozmiar.

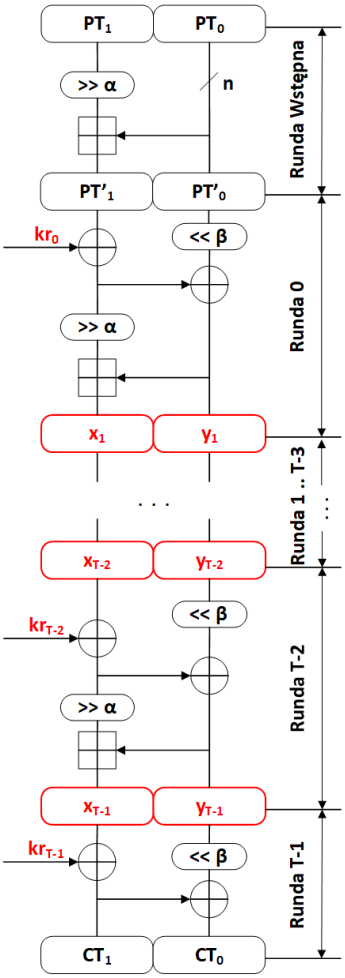
Tabela 12: Liczba zmiennych binarnych wymaganych podczas linearyzacji równań pojedynczej rundy, dla analizowanych zakresów rundy szyfru Speck2n/mn.

Rozmiar słowa (n)	*Liczba zmiennych dla linearyzacji rundy*	
	zakres rundy według dokumentacji szyfru	*zakres rundy według zaproponowanego podejścia*
16	$1,7 \cdot 10^9 + 16$	$16 + 64$
24	$1,7 \cdot 10^{13} + 24$	$24 + 96$
32	$1,5 \cdot 10^{17} + 32$	$32 + 128$
48	$9,9 \cdot 10^{24} + 48$	$48 + 192$
64	$5,8 \cdot 10^{32} + 64$	$64 + 256$

Rozszerzmy przeprowadzoną analizę zaproponowanego podejścia z przesunięciem zakresu rundy na cały algorytm szyfrowania. Na rysunku 48 przedstawiono schemat algorytmu szyfrowania szyfru Speck2n/mn, dla zaproponowanego zakresu rundy. Słowa PT_1 i PT_0 oraz CT_1 i CT_0 to n-bitowe słowa, odpowiednio, tekstu jawnego i szyfrogramu, które podczas ataku są znane. Kolorem czerwonym oznaczono stany pośrednie oraz klucze rundowe, dla których potrzebne są dodatkowe zmienne binarne podczas wyznaczania równań. Liczba dodatkowych zmiennych binarnych jest sumą liczby zmiennych dla stanów pośrednich (słowa od x_1 do x_{T-1} oraz od y_1 do y_{T-1}), wynoszącej $2n(T-1)$, oraz liczby zmiennych binarnych dla kluczy rundowych (słowa od kr_1 do kr_{T-1}), wynoszącej $n(T-1)$.

Ponieważ w rundzie, oznaczonej na rysunku 48 jako *RundaWstępna*, nie jest dodawany klucz rundowy, bity słów PT'_1 i PT'_0 są znane i można je wyznaczyć za pomocą następujących zależności:

$$PT'_1 = \left(\left(PT_1 \gg \alpha\right) + PT_0\right) \ mod \ 2^n$$



Rysunek 48: Algorytm szyfrowania szyfru Speck2n/mn z zaproponowanym przesunięciem.

dla lewego słowa oraz

$$PT'_0 = PT_0$$

dla prawego słowa stanu pośredniego po rundzie wstępnej.

Uwzględniając znajomość bitów słów PT'_1 i PT'_0 oraz CT_1 i CT_0 w równaniach (95) i (98), opisujących, odpowiednio, lewą i prawą ścieżkę przetwarzania pojedynczej rundy, otrzymujemy dla pierwszej rundy (*Runda 0*) dla lewego słowa równanie liniowe (99) oraz dla prawego słowa również równanie liniowe (100). Linearyzacja tych równań nie wymaga dodatkowych zmiennych binarnych.

$$\sum_{j=0}^{n-1} 2^j \left(PT'_{1_{(j+\alpha) \bmod n}} + kr_{0_{(j+\alpha) \bmod n}} - 2PT'_{1_{(j+\alpha) \bmod n}} kr_{0_{(j+\alpha) \bmod n}} + y_{1_j} - x_{1_j} \right) = 0, \tag{99}$$

$$\sum_{j=0}^{n-1} 2^j \Big(PT'_{0_{(j-\beta) \bmod n}} + PT'_{1_j} - 2PT'_{0_{(j-\beta) \bmod n}} PT'_{1_j} + kr_{0_j} - 2kr_{0_j} PT'_{0_{(j-\beta) \bmod n}} + \\ -2kr_{0_j} PT'_{1_j} + 4kr_{0_j} PT'_{0_{(j-\beta) \bmod n}} PT'_{1_j} - y_{1_j} \Big) = 0. \tag{100}$$

Ostatnia runda (*Runda T-1*), w wyniku przesunięcia stanów pośrednich, została skrócona do dwóch operacji xor oraz rotacji. Stąd ostatnią rundę można przedstawić za pomocą równania (101) stopnia 2 dla lewego słowa oraz równania liniowego (102), dla prawego słowa. W równaniu (101) występuje n różnych jednomianów kwadratowych, więc podczas linearyzacji wymaganych jest co najwyżej n dodatkowych zmiennych binarnych.

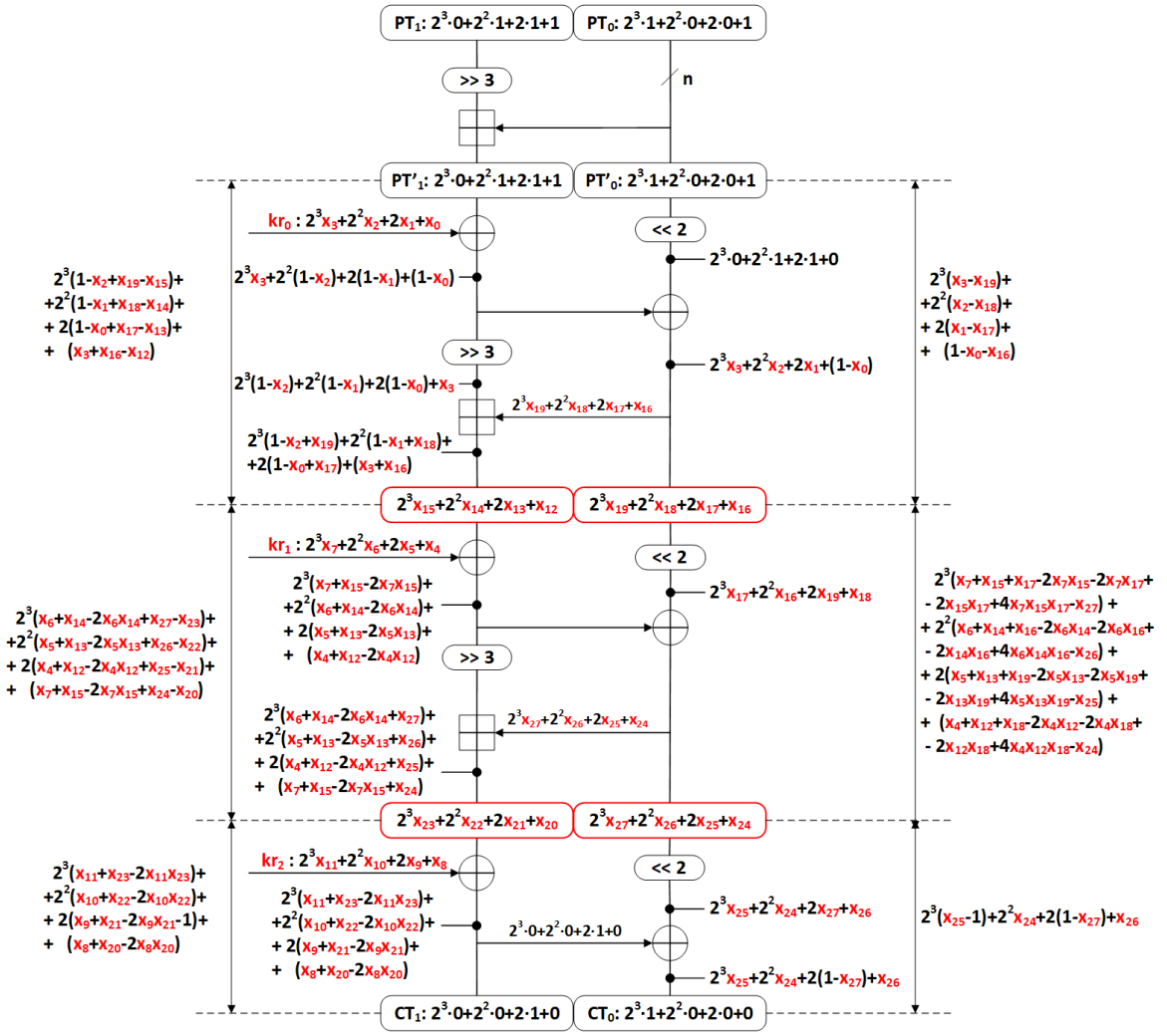
$$\sum_{j=0}^{n-1} 2^j \Big(x_{T-1_j} + kr_{T-1_j} - 2x_{T-1_j} kr_{T-1_j} - CT_{1_j} \Big) = 0, \tag{101}$$

$$\sum_{j=0}^{n-1} 2^j \Big(y_{T-1_{(j-\beta) \bmod n}} + CT_{1_j} - 2y_{T-1_{(j-\beta) \bmod n}} CT_{1_j} - CT_{0_j} \Big) = 0. \tag{102}$$

Dla pozostałych $T-2$ rund, lewa i prawa ścieżka przetwarzania jest przedstawiana za pomocą, odpowiednio, równania (95) i równania (98).

W celu zobrazowania wyznaczania równań nad pierścieniem $\mathbb{Z}_{2^n}$ dla algorytmu szyfrowania Speck, na rysunku 49 przedstawiono opis małej instancji szyfru, dla następujących parametrów:

- długość bloku: $2n = 8$ bitów,
- długość słowa: $n = 4$ bity,
- długość klucza: $mn = 8$ bitów,
- liczba słów klucza: $m = 2$,
- liczba bitów dla rotacji w prawo: $\alpha = 3$,
- liczba bitów dla rotacji w lewo: $\beta = 2$,
- liczba rund: $T = 3$,
- tekst jawny: $PT_1 = 7$, $PT_0 = 9$,



Rysunek 49: Przykład wyznaczania równań nad $\mathbb{Z}_{2^n}$ dla algorytmu szyfrowania małej instancji szyfru Speck8/8.

- szyfrogram: $CT_1 = 2$, $CT_0 = 8$.

Podsumowując, algorytm szyfrowania T-rundowego szyfru Speck2n/mn, przy założeniu, że bity kluczy rundowych reprezentowane są przez zmienne binarne, może zostać opisany za pomocą układu, składającego się z:

- jednego równania liniowego (99),
- jednego równania liniowego (100),
- $T-2$ równań stopnia 2 (95),
- $T-2$ równań stopnia 3 (98),

- jednego równania stopnia 2 (101) oraz
- jednego równania liniowego (102).

Liczba dodatkowych zmiennych binarnych, wymaganych podczas procesu linearyzacji, wynosi co najwyżej $(T-2)(n+5n)+n$.

5.3.3 Wyznaczanie równań nad pierścieniem $\mathbb{Z}_{2^n}$, opisujących algorytm generowania kluczy rundowych szyfru Speck

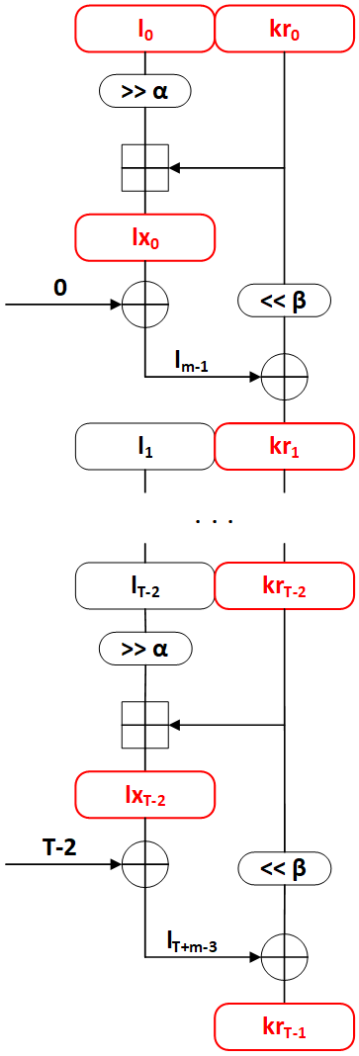
Zgodnie z dokumentacją szyfru Speck, algorytm generowania kluczy rundowych, w rundzie i wyznacza klucz dla rundy $i+1$-wszej, według równania (103). Jednocześnie, w tej samej rundzie, wyznaczane jest również $i+m-1$-wsze słowo l, zgodnie z równaniem (104). Liczba rund algorytmu generowania kluczy jest o jeden mniejsza, niż liczba rund algorytmu szyfrowania, dla danej instancji szyfru Speck$2n/mn$.

$$kr_{i+1} = \left(kr_i \ll \beta\right) \oplus l_{i+m-1}, \tag{103}$$

$$l_{i+m-1} = \left(\left(l_i \gg \alpha\right) + kr_i\right) \ mod\ 2^n \oplus i. \tag{104}$$

Ponieważ algorytm generowania kluczy rundowych wykorzystuje funkcję rundy, dla opisujących go równań można wykorzystać analizę, przeprowadzoną dla algorytmu szyfrowania. Jednak w algorytmie generowania kluczy, jeśli parametr $m \neq 2$, to nie występuje ciągłość w przetwarzaniu lewego słowa l_i, tzn. wejściem dla i-tej rundy jest słowo l_i, a wyjściem l_{i+m-1}. Dlatego też nie jest możliwe identyczne przesunięcie zakresu rundy jak dla algorytmu szyfrowania. Można jednak zmienić miejsce wprowadzenia stanu pośredniego dla lewej ścieżki przetwarzania, tzn. można wprowadzić dodatkowe stany pośrednie po operacji dodawania modulo 2^n. Wtedy stany pośrednie przedstawiane są jako zmienne binarne i przecinają tym samym lewą ścieżkę przetwarzania. Natomiast bity słów l_i, dla $i = \overline{0, T+m-3}$, przedstawiane są jako wielomiany. Na rysunku 50 przedstawiono proponowany podział algorytmu generowania kluczy rundowych, gdzie kolorem czerwonym oznaczono klucze rundowe oraz dodatkowe stany pośrednie, oznaczone jako lx_i.

Rozważmy najpierw przypadek, gdzie bity kluczy rundowych reprezentowane



Rysunek 50: Proponowany podział algorytmu generowania kluczy rundowych szyfru Speck$2n/mn$.

są jako zmienne binarne. W takim podejściu liczba zmiennych binarnych potrzebnych do wyznaczenia równań wielomianowych, opisujących algorytm generowania kluczy, wynosi $mn+2(T-1)n$, gdzie mn to liczba zmiennych dla klucza głównego $K = [l_{m-2},...,l_0,kr_0]$, $(T-1)n$ to liczba zmiennych dla kluczy rundowych oraz $(T-1)n$ to liczba zmiennych dla stanów pośrednich lx_i. Każda runda algorytmu generowania kluczy szyfru Speck reprezentowana jest za pomocą dwóch równań wielomianowych o wielu zmiennych nad pierścieniem $\mathbb{Z}_{2^n}$, po jednym równaniu dla każdego stanu pośredniego lx_i i każdego klucza rundowego kr_i.

Rozpatrzmy najpierw lewą ścieżkę przetwarzania. Słowo lx_i, dla $i = \overline{0,T-2}$, definiowane jest za pomocą równania (105), które można przekształcić, tak samo jak

w algorytmie szyfrowania, do równania (106):

$$lx_i = \left(\left(l_i \gg \alpha\right) + kr_i\right) \ mod\ 2^n, \tag{105}$$

$$\left(\left(l_i \gg \alpha\right) + kr_i\right) - lx_i = 0. \tag{106}$$

Przedstawiając poszczególne słowa jako sumy bitów, analogicznie do równań algorytmu szyfrowania, równanie (106) można przedstawić w postaci:

$$\left(\left(\sum_{j=0}^{n-1} 2^j l_{i_j} \gg \alpha\right) + \sum_{j=0}^{n-1} 2^j kr_{i_j}\right) - \sum_{j=0}^{n-1} 2^j lx_{i_j} = 0, \tag{107}$$

gdzie słowa $\sum_{j=0}^{n-1} 2^j l_{i_j}$, dla $i = \overline{0, m-2}$, są słowami klucza głównego, przedstawionymi za pomocą zmiennych binarnych, a dla $i = \overline{m-1, T-2}$ są wielomianami, wyznaczanymi zgodnie z równaniem:

$$\sum_{j=0}^{n-1} 2^j l_{i+m-1_j} = \sum_{j=0}^{n-1} 2^j lx_{i_j} \oplus \sum_{j=0}^{n-1} 2^j i_j, \tag{108}$$

dla $i = \overline{0, T-2}$, gdzie bit i_j jest j-tym bitem rundy numer i. Po wykonaniu operacji xor, zgodnie z równaniem (79), dla słowa $\sum_{j=0}^{n-1} 2^j l_{i+m-1_j}$ otrzymujemy równanie:

$$\sum_{j=0}^{n-1} 2^j l_{i+m-1_j} = \sum_{j=0}^{n-1} 2^j \left(lx_{i_j} + i_j - 2lx_{i_j} i_j\right) = \sum_{j=0}^{n-1} 2^j \left(lx_{i_j}(1-2i_j) + i_j\right), \tag{109}$$

które jest równaniem liniowym, ponieważ bity i_j są znane. Co więcej, jeśli j-ty bit i_j numeru rundy jest równy 0, to j-ty bit słowa l_{i+m-1} jest równy j-tej zmiennej słowa lx_i: $l_{i+m-1_j} = lx_{i_j}$, w przeciwnym przypadku, j-ty bit słowa l_{i+m-1} jest równy $1 - lx_{i_j}$: $l_{i+m-1_j} = 1 - lx_{i_j}$.

Wracając do równania (107), po wykonaniu rotacji w prawo otrzymujemy finalne równanie dla lewej ścieżki przetwarzania algorytmu generowania kluczy rundowych, następującej postaci:

$$\sum_{j=0}^{n-1} 2^j \left(l_{i_{(j+\alpha)\ mod\ n}} + kr_{i_j} - lx_{i_j}\right) = 0. \tag{110}$$

Podobnie przeanalizujmy prawą ścieżkę przetwarzania, gdzie klucz rundowy kr_{i+1}

można zdefiniować za pomocą równania (111) lub równoważnie za pomocą równania (112):

$$kr_{i+1} = \left(\left(lx_i \oplus i\right) \oplus \left(kr_i \ll \beta\right)\right), \tag{111}$$

$$\sum_{j=0}^{n-1} 2^j kr_{i+1_j} = \sum_{j=0}^{n-1} 2^j lx_{i_j} \oplus \sum_{j=0}^{n-1} 2^j i_j \oplus \left(\sum_{j=0}^{n-1} 2^j kr_{i_j} \ll \beta\right). \tag{112}$$

Po wykonaniu operacji xor i rotacji w lewo oraz przeniesieniu lewej strony na prawą i przyrównaniu do zera, otrzymujemy ostateczne równanie dla kolejnych kluczy rundowych, postaci:

$$\begin{aligned}&\sum_{j=0}^{n-1} 2^j \Big(lx_{i_j} + i_j - 2i_j lx_{i_j} + kr_{i_{(j-\beta) \bmod n}} - 2i_j kr_{i_{(j-\beta) \bmod n}} + \\ &+(4i_j - 2) lx_{i_j} kr_{i_{(j-\beta) \bmod n}} - kr_{i+1_j}\Big) = 0.\end{aligned} \tag{113}$$

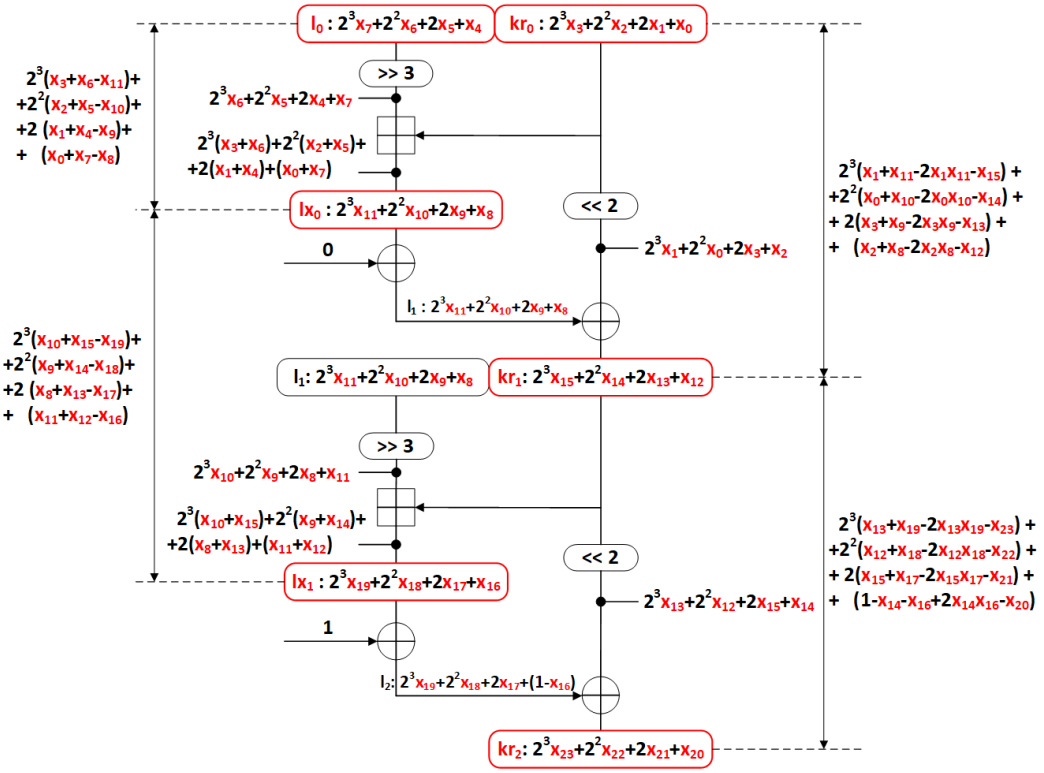
Analizując postacie otrzymanych równań w kontekście późniejszej linearyzacji widzimy, że równanie (110) jest równaniem liniowym i nie wymaga dodatkowych zmiennych binarnych podczas linearyzacji, natomiast równanie (113) składa się z n różnych jednomianów stopnia 2, dlatego też podczas linearyzacji potrzebnych jest co najwyżej n dodatkowych zmiennych binarnych.

Dla tej samej małej instancji szyfru Speck8/8, co dla algorytmu szyfrowania, na rysunku 51 przedstawiono opis algorytmu generowania kluczy rundowych za pomocą równań nad $\mathbb{Z}_{2^n}$.

5.3.4 Konstrukcja układu równań nad pierścieniem $\mathbb{Z}_{2^n}$, opisującego szyfr Speck

W celu utworzenia układu równań wielomianowych o wielu zmiennych opisujących cały szyfr Speck, należy połączyć równania algorytmu szyfrowania z równaniami algorytmu generowania kluczy rundowych. Liczba zmiennych binarnych, potrzebnych do utworzenia takiego układu, opisującego T-rundowy szyfr Speck, wynosi $4(T-1)n+mn$, na którą składa się:

- mn zmiennych dla klucza głównego,



Rysunek 51: Przykład wyznaczania równań nad $\mathbb{Z}_{2^n}$ dla algorytmu generowania kluczy rundowych małej instancji szyfru Speck8/8.

- $(T-1)n$ zmiennych dla kluczy rundowych, z wyjątkiem kr_0,
- $(T-1)n$ zmiennych dla stanów pośrednich algorytmu generowania kluczy,
- $2(T-1)n$ zmiennych dla stanów pośrednich algorytmu szyfrowania.

Układ ten składa się z:

- jednego równania liniowego (99),
- jednego równania liniowego (100),
- $T-2$ równań stopnia 2 (95),
- $T-2$ równań stopnia 3 (98),
- jednego równania stopnia 2 (101),
- jednego równania liniowego (102),

dla algorytmu szyfrowania oraz

- $T-1$ równań liniowych (110),
- $T-1$ równań stopnia 2 (113),

dla algorytmu generowania kluczy rundowych. Podczas linearyzacji takiego układu wymaganych jest co najwyżej $(6(T-2)+2)n$ dodatkowych zmiennych binarnych. Tak jak w przypadku standardu AES, również dla szyfru Speck, opisanego za pomocą równań nad $\mathbb{Z}_{2^n}$, proces linearyzacji nie jest problemem NP-trudnym.

Należy również zauważyć, że jest wiele różnych wariantów szyfru Speck, co przedstawiono już wcześniej w tabeli 11. Każdy wariant ma różny rozmiar bloku wejściowego i różną długość klucza. Dlatego też dla wariantów, dla których długość bloku wejściowego jest mniejsza niż długość klucza, równania algorytmu szyfrowania wyznaczone zostały dla dwóch różnych par tekst jawny – szyfrogram, tworząc dwa układy opisujące algorytm szyfrowania. W tych przypadkach liczba zmiennych binarnych, wymaganych do utworzenia układu dla algorytmu szyfrowania, jest dwa razy większa i wynosi $4(T-1)n$. Wyznaczone dwa układy równań dla algorytmu szyfrowania zostały połączone z równaniami algorytmu generowania kluczy, tworząc ostateczny układ, który przekształcono do równoważnego problemu QUBO. W tabeli 13 przedstawiono liczbę zmiennych binarnych równoważnego problemu QUBO dla każdego wariantu szyfru Speck$2n/mn$.

Tabela 13: Rozmiar problemu QUBO dla wariantów szyfru Speck$2n/mn$, opisanego za pomocą równań nad $\mathbb{Z}_{2^n}$.

Wariant szyfru Speck2n/mn	*Liczba rund*	*Liczba par*	*Liczba równań*	*Liczba zmiennych binarnych*			
				początkowy układ	*linearyzacja*	*wartości k*	*problem QUBO*
Speck32/64	22	2	130	2 080	2 745	750	5 575
Speck48/72	22	2	130	3 096	4 209	829	8 134
Speck48/96	23	2	136	3 264	4 416	830	8 510
Speck64/96	26	2	154	4 896	6 789	1 042	12 727
Speck64/128	27	2	160	5 120	7 068	1 083	13 271
Speck96/96	28	1	110	5 280	6 204	766	12 250
Speck96/144	29	2	172	8 208	11 562	1 224	20 994
Speck128/128	32	1	126	8 064	9 576	971	18 611
Speck128/192	33	2	196	12 480	17 766	1 525	31 771
Speck128/256	34	2	202	12 928	18 333	1 572	32 833

Przedstawiona w niniejszej rozprawie analiza opisania szyfru Speck$2n/mn$ za

pomocą układu równań wielomianowych została zaprezentowana na międzynarodowej konferencji *International Conference on Computational Science ICCS 2022* oraz opublikowana w artykule [11].

5.4 Analiza przedstawienia szyfru Speck za pomocą układu równań wielomianowych o wielu zmiennych nad ciałem $GF(2)$

Ponieważ spośród trzech podstawowych operacji szyfru Speck dwie są operacjami bitowymi, drugą strukturą algebraiczną, nad którą zostaną wyznaczone równania opisujące ten szyfr, jest ciało binarne. Ponadto, skalowanie elementów macierzy Q problemu QUBO w wyżarzaczu D-Wave również przyczynia się do wyboru ciała binarnego, ponieważ elementy tego ciała w macierzy Q będą znacznie mniejsze, niż dla pierścienia $\mathbb{Z}_{2^n}$, jednak kosztem większej liczby zmiennych z powodu większej liczby równań oraz realizacji operacji dodawania modularnego nad $GF(2)$. Celem niniejszego rozdziału jest oszacowanie tego kosztu.

5.4.1 Wyznaczanie równań nad ciałem $GF(2)$, opisujących algorytm szyfrowania Speck

Załóżmy, że bity kluczy rundowych przedstawione są za pomocą zmiennych binarnych, a równania wielomianowe o wielu zmiennych wyznaczane są nad ciałem binarnym, dla każdego bitu słowa stanu pośredniego każdej rundy szyfru. Niech teraz n-bitowe słowa a i b przedstawione będą jako ciąg bitów, postaci:

$$a = \left(a_{n-1}, a_{n-2}, \ldots, a_1, a_0\right),$$

$$b = \left(b_{n-1}, b_{n-2}, \ldots, b_1, b_0\right),$$

gdzie a_j i b_j są j-tymi bitami słów. Rotacja słowa a w prawo o α bitów oraz rotacja słowa b w lewo o β bitów wykonywana jest jako cykliczne przesunięcie bitów w ciągu, zgodnie z następującymi równaniami:

$$a \gg \alpha = \left(a_{(n-1+\alpha) \bmod n}, a_{(n-2+\alpha) \bmod n}, \ldots, a_{\alpha+1}, a_\alpha\right), \tag{114}$$

$$b \ll \beta = \left(b_{(n-1-\beta) \bmod n}, b_{(n-2-\beta) \bmod n}, \ldots, b_{n-\beta+1}, b_{n-\beta}\right). \tag{115}$$

Bitową operację xor dwóch bitów a_j i b_j w ciele binarnym można wykonać jako dodawanie arytmetyczne w tym ciele: $a_j \oplus b_j = a_j + b_j$, stąd bitowa operacja xor dwóch n-bitowych słów a i b realizowana jest zgodnie z równaniem:

$$a \oplus b = \left(a_{n-1} + b_{n-1}, a_{n-2} + b_{n-2}, \ldots, a_1 + b_1, a_0 + b_0\right). \tag{116}$$

Ponieważ chcemy wyznaczyć równania nad ciałem $GF(2)$, operacja dodawania modulo 2^n musi być realizowana za pomocą pełnego n-bitowego sumatora, przedstawionego za pomocą algorytmu 1, a samą sumę dwóch n-bitowych słów a i b przedstawmy w postaci równania:

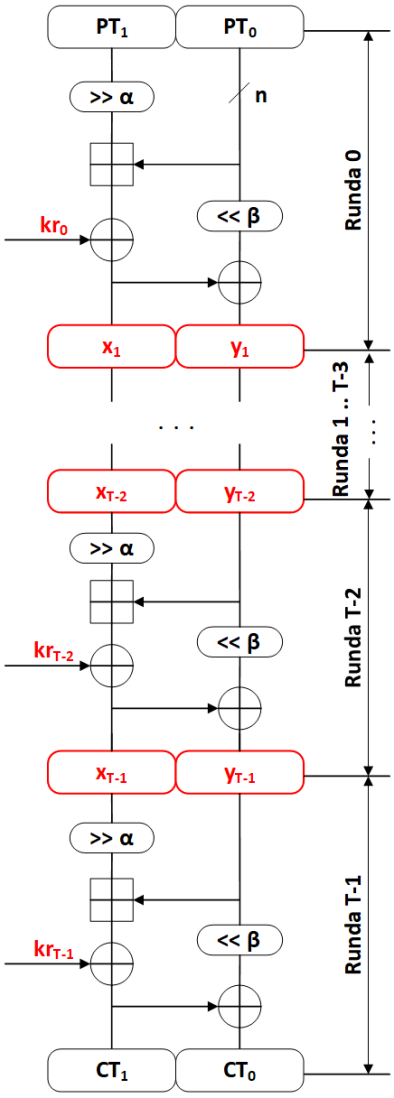
$$(a+b) \bmod 2^n = \left(s_{n-1}, s_{n-2}, \ldots, s_1, s_0\right), \tag{117}$$

gdzie s_j oznacza j-ty bit sumy i reprezentuje odpowiedni wielomian. Ponieważ operacja xor realizowana jest jako dodawanie odpowiadających sobie bitów (116), nie zwiększa ona stopnia wielomianu. Dlatego stopnie wielomianów s_j sumy (117) są mniejsze, niż w poprzednim rozdziale, gdzie operacja xor realizowana była nad pierścieniem $\mathbb{Z}_{2^n}$. Dla porównania, wielomian drugiego bitu sumy s_2, który nad $\mathbb{Z}_{2^n}$ miał stopień 7, teraz nad $GF(2)$ ma stopień 3 oraz następującą postać:

$$s_2 = a_2 + b_2 + a_1 b_1 + a_0 a_1 b_0 + a_0 b_0 b_1.$$

Wielomian ten składa się z dwóch różnych jednomianów stopnia 3, jednego jednomianu stopnia 2 oraz dwóch różnych jednomianów stopnia 1. Ogólnie, stopień wielomianu dla j-tego bitu sumy wynosi $j+1$, stąd dla bloku wejściowego o długości $2n$ stopień wielomianu reprezentującego najbardziej znaczący bit sumy wynosi n.

W celu wygenerowania równań nad ciałem $GF(2)$, opisujących algorytm szyfrowania Speck, wykorzystany zostanie zakres rundy zgodny z dokumentacją. Zatem najpierw wykonywana jest operacja dodawania modularnego, zgodnie z algorytmem n bitowego sumatora, którego danymi wejściowymi są ciągi pojedynczych bitów, a następnie wykonywana jest operacja xor. W wykorzystywanej strukturze algebraicznej zastosowanie przesuniętego zakresu rundy powodowałoby, że dodawanie modularne



Rysunek 52: Algorytm szyfrowania Speck$2n/mn$ według dokumentacji.

wykonywane byłoby na ciągach sum bitów, co zwiększyłoby liczbę różnych nieliniowych jednomianów. Na rysunku 52 przedstawiono schemat algorytmu szyfrowania Speck$2n/mn$, na którym kolorem czerwonym oznaczono wprowadzone stany pośrednie oraz klucze rundowe, dla których wymagane jest zdefiniowanie zmiennych binarnych. Liczba zmiennych dla stanów pośrednich wynosi $2n(T-1)$, natomiast dla kluczy rundowych wynosi $n(T-1)$. Słowa PT_1 i PT_0 oraz CT_1 i CT_0, na rysunku 52, oznaczają, odpowiednio, znany tekst jawny i odpowiadający mu szyfrogram, a n-bitowe słowa x_i, y_i, x_{i+1}, y_{i+1} oraz kr_i zdefiniowane są następująco:

$$x_i = \left(x_{i_{n-1}}, x_{i_{n-2}}, \ldots, x_{i_1}, x_{i_0}\right),$$

$$y_i = \left(y_{i_{n-1}}, y_{i_{n-2}}, \ldots, y_{i_1}, y_{i_0} \right),$$

$$kr_i = \left(kr_{i_{n-1}}, kr_{i_{n-2}}, \ldots, kr_{i_1}, kr_{i_0} \right),$$

$$x_{i+1} = \left(x_{i+1_{n-1}}, x_{i+1_{n-2}}, \ldots, x_{i+1_1}, x_{i+1_0} \right),$$

$$y_{i+1} = \left(y_{i+1_{n-1}}, y_{i+1_{n-2}}, \ldots, y_{i+1_1}, y_{i+1_0} \right),$$

gdzie x_{i_j} oznacza j-ty bit słowa x_i.

Rozważmy najpierw postać równań dla dowolnej rundy. Lewą ścieżkę przetwarzania dowolnej rundy algorytmu szyfrowania, realizowaną zgodnie z równaniem (81), można przedstawić za pomocą układu n równań następującej postaci:

$$s_{i_j} + kr_{i_j} + x_{i+1_j} = 0, \tag{118}$$

dla $j = \overline{0, n-1}$, w którym s_{i_j} jest wielomianem reprezentującym j-ty bit sumy s_i, wyznaczonej w i-tej rundzie, zgodnie z równaniem:

$$s_i = \left(\left(x_i \gg \alpha \right) + y_i \right) \bmod 2^n = \left(s_{i_{n-1}}, s_{i_{n-2}}, \ldots, s_{i_1}, s_{i_0} \right).$$

Prawa ścieżka przetwarzania dowolnej rundy algorytmu szyfrowania, definiowana równaniem (87), składa się tylko z dwóch operacji bitowych: rotacji w lewo oraz operacji xor. Można je przedstawić nad ciałem binarnym za pomocą układu n równań postaci:

$$y_{i_{(j-\beta) \bmod n}} + x_{i+1_j} + y_{i+1_j} = 0, \tag{119}$$

dla $j = \overline{0, n-1}$.

Przeanalizujmy otrzymane równania wielomianów, w kontekście ich linearyzacji. Zacznijmy od równań postaci (119) dla prawej ścieżki. Każde z tych n równań jest oczywiście równaniem liniowym, zatem proces linearyzacji nie wymaga dla nich dodatkowych zmiennych binarnych. Teraz rozważmy równania postaci (118) dla lewej ścieżki, których stopień zależy od stopnia wielomianu s_{i_j}. Oznacza to, że proces linearyzacji układu równań, opisujących pojedynczą rundę algorytmu szyfrowania Speck, będzie dotyczyć tylko wielomianów sumy. Spróbujmy oszacować górną oraz dolną

granicę liczby dodatkowych zmiennych binarnych potrzebnych do linearyzacji sumy, w zależności od długości słowa n.

Dla przykładu niech dane będą dwa 4-bitowe słowa $a = (a_3, a_2, a_1, a_0)$ oraz $b = (b_3, b_2, b_1, b_0)$. Wielomiany kolejnych bitów sumy $s = (a+b) \ mod \ 2^4 = (s_3, s_2, s_1, s_0)$ mają następującą postać:

$$
\begin{aligned}
s_0 &= a_0 + b_0, \\
s_1 &= a_0 b_0 + a_1 + b_1, \\
s_2 &= a_1 b_1 + a_0 b_0 a_1 + a_0 b_0 b_1 + a_2 + b_2, \\
s_3 &= a_2 b_2 + a_1 b_1 a_2 + a_1 b_1 b_2 + a_0 b_0 a_1 a_2 + a_0 b_0 a_1 b_2 + a_0 b_0 b_1 a_2 + a_0 b_0 b_1 b_2 + a_3 + b_3.
\end{aligned}
\tag{120}
$$

Aby wyznaczyć maksymalną liczbę zmiennych binarnych dla procesu linearyzacji zakładamy, że dla każdego jednomianu stopnia d potrzebnych jest $d-1$ zmiennych binarnych. Zatem dla powyższego przykładu wymagane są 23 zmienne. Analogicznie jak w poprzednim rozdziale, niech z^d oznacza jednomian stopnia d. Stąd liczbę różnych jednomianów stopnia d w wielomianach (120) można przedstawić jako $2z$ dla s_0, $2z + z^2$ dla s_1, $2z + z^2 + 2z^3$ dla s_2 oraz $2z + z^2 + 2z^3 + 4z^4$ dla s_3. Aby wyznaczyć liczbę różnych jednomianów stopnia d dla dowolnego wielomianu bitu s_j sumy, rozważmy operacje algorytmu 1 pełnego sumatora nad ciałem binarnym. Tak więc wielomian $c_j = a_j b_j + c_{j-1}(a_j \oplus b_j)$ nad ciałem binarnym wyznaczany jest jako $c_j = a_j b_j + c_{j-1}(a_j + b_j)$. Liczbę jego różnych jednomianów można przedstawić w postaci $z^2 + 2zc_{j-1}$. Następnie, wielomian $s_j = c_{j-1} \oplus a_j \oplus b_j$ nad ciałem binarnym można zapisać jako $s_j = c_{j-1} + a_j + b_j$, a liczbę jego różnych jednomianów stopnia d w postaci $c_{j-1} + 2z$. Ogólnie dla równania (118), liczbę jednomianów stopnia d, dla j-tego bitu, można wyznaczyć za pomocą wielomianu $c_{j-1} + 4z$, gdzie $c_j = z^2 + 2zc_{j-1}$ oraz $c_j = 0$, dla $j < 0$.

Aby wyznaczyć minimalną liczbę dodatkowych zmiennych binarnych, zakładamy wykonanie optymalnej linearyzacji (co jest problemem NP trudnym), która polega na takim podstawieniu nowej zmiennej za jednomian kwadratowy, aby to podstawienie można było wykonać możliwie jak największą liczbę razy w całym układzie.

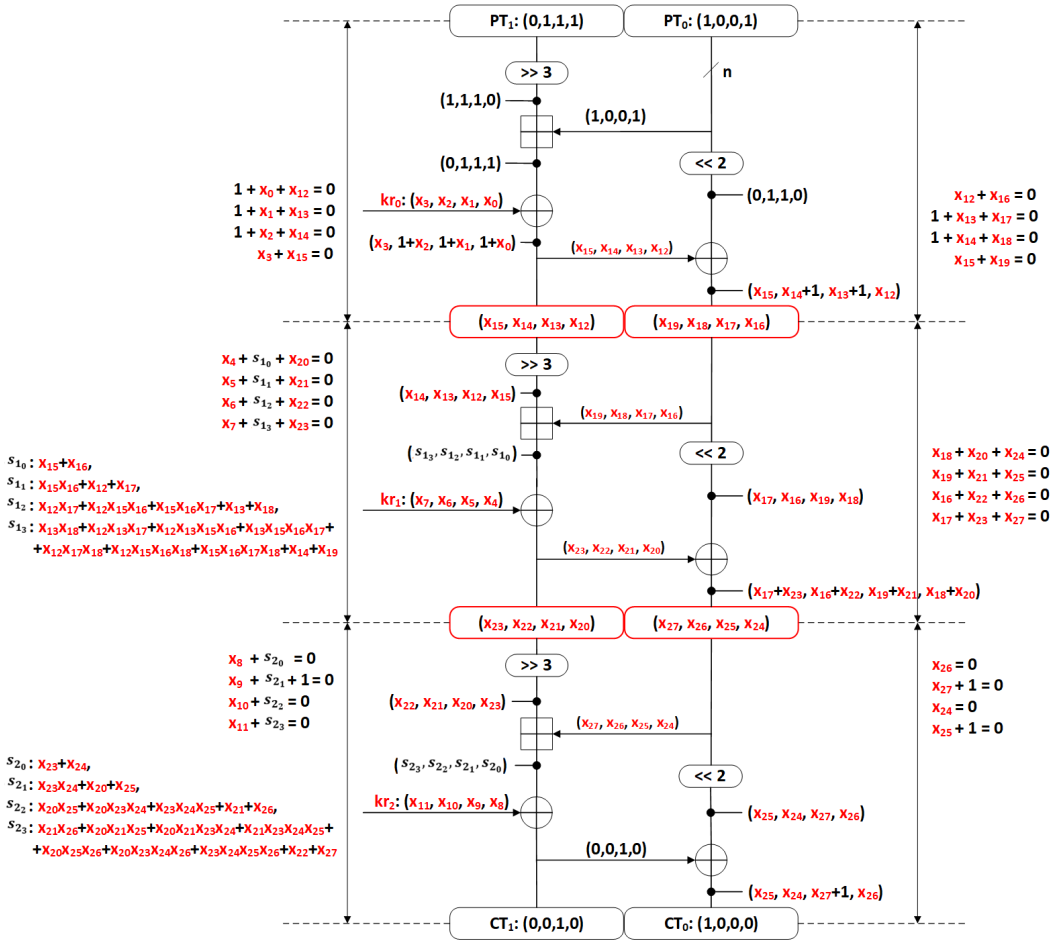
Analizując przedstawiony przykład równań (120), można zauważyć, że kolejne jednomiany stopnia co najmniej dwa wymagają jednej dodatkowej zmiennej, ponieważ są iloczynami jednomianów występujących wcześniej oraz bitu a_j lub b_j, co można przedstawić następująco:

$$
\begin{aligned}
&s_0 = a_0 + b_0,\\
&s_1 = \underbrace{a_0 b_0}_{x_1} + a_1 + b_1 = x_1 + a_1 + b_1,\\
&s_2 = \underbrace{a_1 b_1}_{x_2} + \underbrace{a_0 b_0}_{x_1} a_1 + \underbrace{a_0 b_0}_{x_1} b_1 + a_2 + b_2 = x_2 + \underbrace{x_1 a_1}_{x_3} + \underbrace{x_1 b_1}_{x_4} + a_2 + b_2 =\\
&= x_2 + x_3 + x_4 + a_2 + b_2,\\
&s_3 = \underbrace{a_2 b_2}_{x_5} + \underbrace{a_1 b_1}_{x_2} a_2 + \underbrace{a_1 b_1}_{x_2} b_2 + \underbrace{a_0 b_0}_{x_1} a_1 a_2 + \underbrace{a_0 b_0}_{x_1} a_1 b_2 + \underbrace{a_0 b_0}_{x_1} b_1 a_2 +\\
&+ \underbrace{a_0 b_0}_{x_1} b_1 b_2 + a_3 + b_3 = x_5 + \underbrace{x_2 a_2}_{x_6} + \underbrace{x_2 b_2}_{x_7} + \underbrace{x_1 a_1}_{x_3} a_2 + \underbrace{x_1 a_1}_{x_3} b_2 +\\
&+ \underbrace{x_1 b_1}_{x_4} a_2 + \underbrace{x_1 b_1}_{x_4} b_2 + a_3 + b_3 = x_5 + x_6 + x_7 + \underbrace{x_3 a_2}_{x_8} + \underbrace{x_3 b_2}_{x_9} + \underbrace{x_4 a_2}_{x_{10}} +\\
&+ \underbrace{x_4 b_2}_{x_{11}} + a_3 + b_3 = x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + a_3 + b_3.
\end{aligned}
\tag{121}
$$

Dla rozważanego przykładu, minimalna liczba dodatkowych zmiennych binarnych dla linearyzacji wynosi 11. Jak już wcześniej zauważono, każdy nieliniowy jednomian jest iloczynem bitu a_j lub b_j oraz nieliniowego jednomianu, wcześniej zastąpionego przez nową zmienną binarną, dlatego w ogólnym przypadku liczba dodatkowych zmiennych binarnych wymaganych podczas linearyzacji jest równa liczbie różnych, nieliniowych jednomianów w całym układzie. Zatem operacja linearyzacji nie jest problemem NP-trudnym w tym przypadku. W tabeli 14 przedstawiono liczbę zmiennych binarnych dla linearyzacji wielomianów, reprezentujących bity sumy modulo 2^n, dwóch n-bitowych słów.

Tabela 14: Minimalna liczba zmiennych binarnych dla linearyzacji wielomianów, reprezentujących bity sumy modularnej dwóch n-bitowych słów.

Rozmiar słowa (n)	*Minimalna liczba zmiennych binarnych*
16	65 519
24	$1,68 \cdot 10^7$
32	$4,29 \cdot 10^9$
48	$2,81 \cdot 10^{14}$
64	$1,84 \cdot 10^{19}$



Rysunek 53: Przykład wyznaczania równań nad $GF(2)$ dla algorytmu szyfrowania małej instancji szyfru Speck8/8.

Na podstawie wartości z tabeli 14 można już wywnioskować, że koszt opisania szyfru Speck nad ciałem binarnym jest bardzo duży. Należy również zwrócić uwagę, że koszt linearyzacji równań pojedynczej rundy nad ciałem binarnym jest o kilka rzędów wielkości mniejszy, niż koszt linearyzacji równań pojedynczej rundy nad pierścieniem $\mathbb{Z}_{2^n}$, o zakresie zgodnym z dokumentacją (tabela 12). Jest to konsekwencją

realizacji operacji xor w ciele binarnym jako dodawania, które nie zwiększa stopnia wielomianu, a co za tym idzie, stopień sumy jest dwa razy mniejszy w ciele binarnym niż w pierścieniu $\mathbb{Z}_{2^n}$.

Podsumowując, cały algorytm szyfrowania można przedstawić jako układ $2Tn$ równań wielomianowych nad $GF(2)$, z których Tn równań są równaniami liniowymi, a stopień pozostałych Tn równań zależy od wielomianów sumy modularnej.

Tak samo jak w poprzednim rozdziale, dla zobrazowania wyznaczania równań nad $GF(2)$, opisujących szyfr Speck, na rysunku 53 przedstawiono opis małej instancji szyfru Speck$8/8$ nad ciałem binarnym.

5.4.2 Wyznaczanie równań nad ciałem $GF(2)$, opisujących algorytm generowania kluczy rundowych szyfru Speck

W analogiczny sposób, można nad ciałem binarnym przedstawić równania opisujące algorytm generowania kluczy rundowych. Na rysunku 54 przedstawiono schemat algorytmu generowania kluczy rundowych szyfru Speck, gdzie zakres rundy jest zgodny z dokumentacją szyfru. Ponadto, kolorem czerwonym oznaczono klucze rundowe oraz słowa l_i, dla których wymagane jest zdefiniowanie dodatkowych zmiennych binarnych, w celu wyznaczenia równań. Liczba dodatkowych zmiennych jest sumą liczby zmiennych dla kluczy rundowych, wynoszącej $(T-1)n$ oraz liczby zmiennych dla słów l_i, wynoszącej $(T-1)n$.

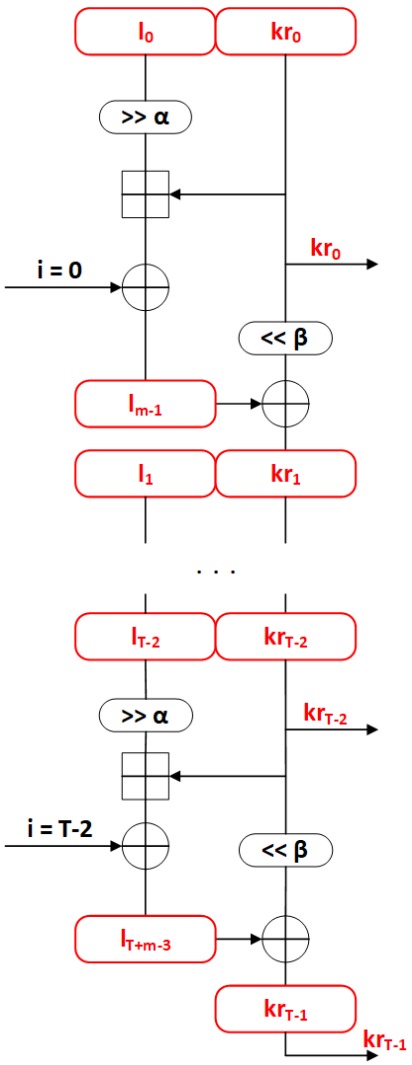
Lewa ścieżka przetwarzania generuje n-bitowe słowa l_i, zgodnie z równaniem (104), które można przedstawić za pomocą układu n równań następującej postaci:

$$s_{i_j} + i_j + l_{i+m-1_j} = 0, \tag{122}$$

dla $j = \overline{0, n-1}$, gdzie i_j oznacza znany, j-ty bit numeru rundy. W tych równaniach, tak samo jak w algorytmie szyfrowania, zmienna s_{i_j} reprezentuje wielomian j-tego bitu sumy i-tej rundy, postaci:

$$s_i = \left(\left(l_i \gg \alpha\right) + kr_i\right) \ mod \ 2^n.$$

Natomiast prawa ścieżka przetwarzania generuje n-bitowe klucze rundowe kr_i, we-



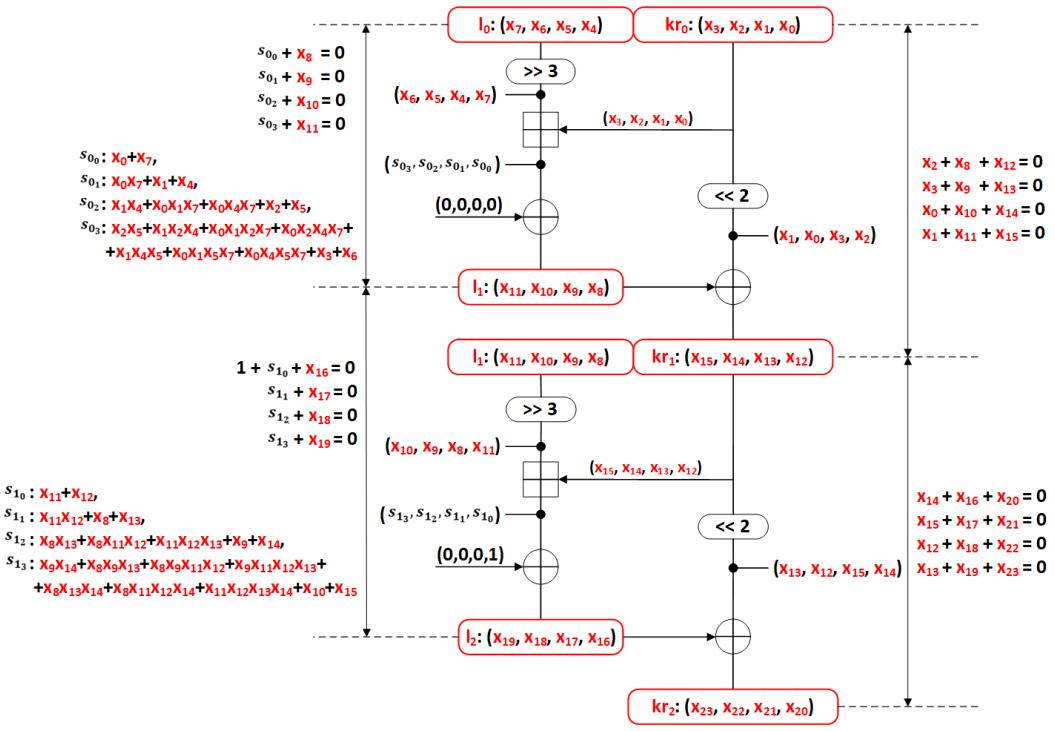
Rysunek 54: Podział zgodny z dokumentacją algorytmu generowania kluczy rundowych szyfru Speck$2n/mn$.

dług równania (103) zgodnego z dokumentacją, które można przedstawić jako układ n równań postaci:

$$kr_{i_{(j-\beta) \bmod n}} + l_{i+m-1_j} + kr_{i+1_j} = 0, \tag{123}$$

dla $j = \overline{0, n-1}$.

Ponieważ funkcja rundy algorytmu szyfrowania Speck jest identyczna z funkcją rundy algorytmu generowania kluczy rundowych, analiza otrzymanych równań w kontekście ich linearyzacji jest identyczna jak w przypadku algorytmu szyfrowania. Zatem cały algorytm generowania kluczy rundowych można opisać za pomocą $2(T-1)n$ równań wielomianowych nad ciałem binarnym, z których $(T-1)n$ równań opisujących prawą ścieżkę przetwarzania jest równaniami liniowymi, nie wymagają-



Rysunek 55: Przykład wyznaczania równań nad $GF(2)$ dla algorytmu generowania kluczy rundowych małej instancji szyfru Speck8/8.

cymi linearyzacji. W pozostałych $(T-1)n$ równaniach dla lewej ścieżki przetwarzania proces linearyzacji przeprowadzony jest tylko dla równań, będących wynikiem sumy modularnej. Koszt linearyzacji wyniku sumy został już oszacowany podczas analizowania algorytmu szyfrowania i jest równy liczbie różnych, nieliniowych jednomianów w całym układzie.

Na rysunku 55 zaprezentowano opis algorytmu generowania kluczy rundowych szyfru Speck8/8 za pomocą równań wielomianowych nad ciałem binarnym.

5.4.3 Konstrukcja układu równań nad ciałem $GF(2)$, opisującego szyfr Speck

Chcąc opisać cały szyfr Speck2n/mn za pomocą układu równań nad ciałem binarnym należy połączyć układy równań algorytmu szyfrowania oraz algorytmu generowania kluczy rundowych. Oczywiście, ze względu na znajomość podczas ataku bitów tekstu jawnego oraz odpowiadającego mu szyfrogramu, równania opisujące przetwarzanie

lewego słowa w pierwszej rundzie są równaniami liniowymi następującej postaci:

$$\left(\left(\left(PT_1 \gg \alpha\right) + PT_0\right) \ mod\ 2^n\right)_j + kr_{0_j} + x_{1_j} = 0, \tag{124}$$

dla $j = \overline{0, n-1}$, a dla prawego słowa, równania liniowe mają postać:

$$PT_{0_{(j-\beta)\ mod\ n}} + x_{1_j} + y_{1_j} = 0, \tag{125}$$

dla $j = \overline{0, n-1}$. Podobnie dla ostatniej rundy, lewe słowo można opisać za pomocą n równań postaci:

$$s_{T-1_j} + kr_{T-1_j} + CT_{1_j} = 0, \tag{126}$$

dla $j = \overline{0, n-1}$, gdzie s_{T-1_j} jest j-tym bitem sumy słów:

$$s_{T-1} = ((x_{T-1} \gg \alpha) + y_{T-1})\ mod\ 2^n,$$

a prawe słowo można przedstawić za pomocą n równań:

$$y_{T-1_{(j-\beta)\ mod\ n}} + CT_{1_j} + CT_{0_j} = 0, \tag{127}$$

dla $j = \overline{0, n-1}$.

Liczba zmiennych binarnych dla układu opisującego T-rundowy szyfr Speck$2n/mn$ jest równa $mn + (T-1)n + (T-m+1)n + 2(T-1)n$, gdy długość klucza jest równa długości bloku wejściowego oraz $mn + (T-1)n + (T-m+1)n + 4(T-1)n$, gdy długość klucza jest większa niż długość bloku wejściowego, gdzie:

- mn – liczba zmiennych binarnych dla klucza głównego,
- $(T-1)n$ – liczba zmiennych binarnych dla kluczy rundowych kr_i,
- $(T-m+1)n$ – liczba zmiennych binarnych dla słów l_i,
- $2(T-1)n$ – liczba zmiennych binarnych dla stanów pośrednich algorytmów szyfrowania, dla jednej pary tekst jawny – szyfrogram.

Układ opisujący T-rundowy szyfr Speck$2n/mn$ nad ciałem binarnym składa się z:

- n równań liniowych, postaci (124),

- n równań liniowych, postaci (125),
- $(T-2)n$ równań stopnia od 1 do n, postaci (118),
- $(T-2)n$ równań liniowych, postaci (119),
- n równań stopnia od 1 do n, postaci (126),
- n równań liniowych, postaci (127),
- $(T-1)n$ równań stopnia od 1 do n, postaci (122),
- $(T-1)n$ równań liniowych, postaci (123).

W tabeli 15 przedstawiono oszacowane wielkości problemu QUBO dla poszczególnych wariantów szyfru Speck2n/mn, przy założeniu, że wykonano optymalną linearyzację, zgodnie z przykładem (121). Jak można wywnioskować z przedstawionych w nich wartości, koszt przedstawienia szyfru Speck za pomocą równań nad ciałem binarnym jest zbyt duży, ponieważ problemu tego rozmiaru nie można nawet uruchomić na obecnym wyżarzaczu kwantowym D-Wave.

Tabela 15: Rozmiar problemu QUBO dla wariantów szyfru Speck2n/mn, opisanego za pomocą równań nad $GF(2)$.

Wariant szyfru Speck2n/mn	*Liczba rund*	*Liczba par*	*Liczba równań*	*Liczba zmiennych binarnych*			
				początkowy układ	*linearyzacja*	*wartości k*	*problem QUBO*
Speck32/64	22	2	1 376	2 048	$4,13 \cdot 10^6$	8 789	$4,14 \cdot 10^6$
Speck48/72	22	2	2 064	3 072	$1,06 \cdot 10^9$	18 633	$1,06 \cdot 10^9$
Speck48/96	23	2	2 160	3 216	$1,11 \cdot 10^9$	19 518	$1,11 \cdot 10^9$
Speck64/96	26	2	3 264	4 864	$3,22 \cdot 10^{11}$	39 089	$3,22 \cdot 10^{11}$
Speck64/128	27	2	3 392	5 056	$3,35 \cdot 10^{11}$	40 650	$3,35 \cdot 10^{11}$
Speck96/96	28	1	5 280	5 280	$1,52 \cdot 10^{16}$	62 418	$1,52 \cdot 10^{16}$
Speck96/144	29	2	5 472	8 160	$2,36 \cdot 10^{16}$	97 788	$2,36 \cdot 10^{16}$
Speck128/128	32	1	8 064	8 064	$1,14 \cdot 10^{21}$	127 226	$1,14 \cdot 10^{21}$
Speck128/192	33	2	8 320	12 416	$1,77 \cdot 10^{21}$	198 048	$1,77 \cdot 10^{21}$
Speck128/256	34	2	8 576	12 800	$1,83 \cdot 10^{21}$	204 233	$1,83 \cdot 10^{21}$

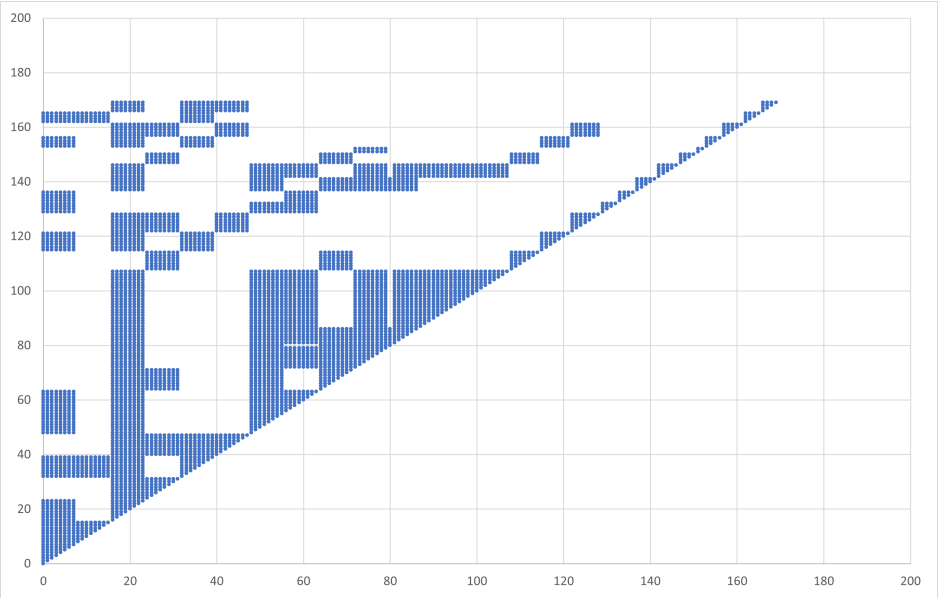
5.5 Atak algebraiczny z wykorzystaniem wyżarzania kwantowego na szyfr Speck

Zakładając, że $O\left(e^{\sqrt{N}}\right)$ ze współczynnikiem równym 1 to złożoność rozwiązania problemu QUBO, składającego się z N zmiennych, w tabeli 16 przedstawiono liczbę rund, dla której proponowany atak jest lepszy niż pełne przeszukiwanie, dla każdego wariantu szyfru Speck, opisanego nad $\mathbb{Z}_{2^n}$. W nawiasie pokazano również procentową wielkość zaatakowanej liczby rund. Najlepszy wynik osiągnięto dla wariantu Speck128/256, gdzie proponowany atak jest lepszy od pełnego przeszukiwania dla 32 z 34 rund, co stanowi 94% liczby rund całego szyfru. Jest to wynik lepszy niż najlepszy znany atak klasyczny na ten wariant szyfru, gdzie przy użyciu kryptoanalizy różnicowej [59] możliwe jest złamanie 25 z 34 rund szybciej niż przez pełne przeszukiwanie. Inne ataki klasyczne na zredukowany szyfr Speck można znaleźć w pracach [46], [26] lub [24].

Tabela 16: Liczba zaatakowanych rund dla najlepszego ataku klasycznego oraz proponowanego ataku, dla poszczególnych wariantów szyfru Speck$2n/mn$.

Wariant szyfru Speck2n/mn	*Liczba rund szyfru*	*Liczba zaatakowanych rund dla najlepszego ataku klasycznego*	*Liczba zaatakowanych rund dla proponowanego ataku*
Speck32/64	22	15 (68%)	9 (41%)
Speck48/72	22	16 (73%)	8 (36%)
Speck48/96	23	17 (74%)	13 (56%)
Speck64/96	26	19 (73%)	10 (38%)
Speck64/128	27	20 (74%)	17 (63%)
Speck96/96	28	20 (71%)	12 (43%)
Speck96/144	29	21 (72%)	15 (52%)
Speck128/128	32	23 (72%)	15 (47%)
Speck128/192	33	24 (73%)	19 (58%)
Speck128/256	34	25 (74%)	32 (94%)

W celu sprawdzenia poprawności generowanych układów, wykonano atak na mniejsze wersje szyfru Speck, za pomocą wyżarzacza kwantowego D-Wave. Pierwszy atak został przeprowadzony na problem o wielkości porównywalnej do rozwiązanego problemu QUBO dla algorytmu AES. Aby otrzymać problem QUBO, o rozmiarze



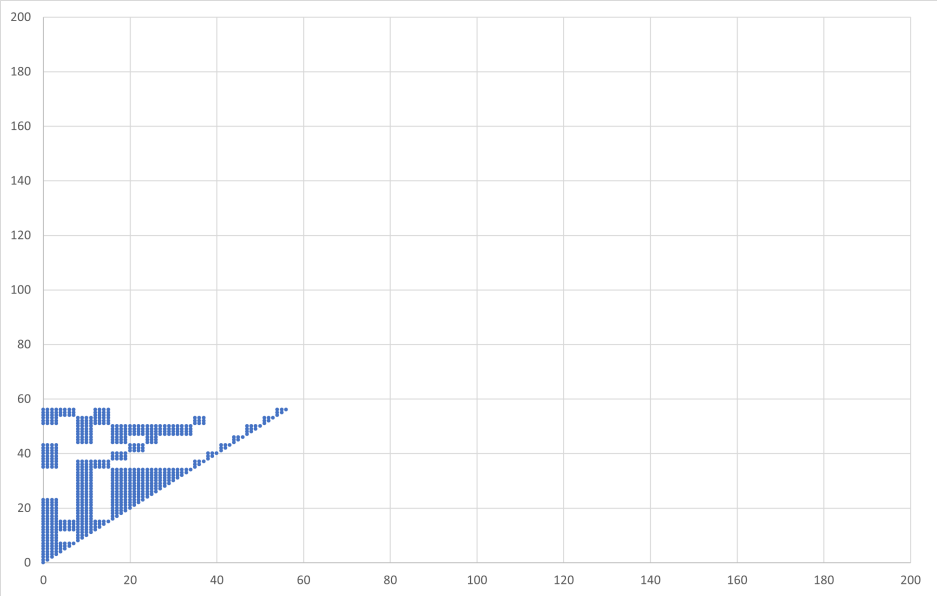
Rysunek 56: Struktura macierzy Q problemu QUBO wygenerowanego dla trzyrundowego szyfru Speck16/16, opisanego nad $\mathbb{Z}_{2^8}$.

około 200 zmiennych binarnych, dla szyfru Speck przyjęto następujące parametry:

- długość bloku: $2n = 16$ bitów,
- długość słowa: $n = 8$ bitów,
- długość klucza: $mn = 16$ bitów,
- liczba słów klucza: $m = 2$,
- liczba bitów dla rotacji w prawo: $\alpha = 4$,
- liczba bitów dla rotacji w lewo: $\beta = 2$,
- liczba rund: $T = 3$.

Dla wielomianu kary przyjęto współczynnik równy 6. W wyniku transformacji układu równań wielomianowych opisujących szyfr Speck16/16 o powyższych parametrach, otrzymano problem QUBO składający się ze 170 zmiennych binarnych. Strukturę macierzy Q wygenerowanego problemu przedstawiono na rysunku 56. Macierz ta składa się z elementów o wartościach od 1 do 14 680 064, co sprawia, że po skalowaniu w wyżarzaczu wartości współczynników są z zakresu od $6,8 \cdot 10^{-8}$ do 1. Dodatkowo

liczba wszystkich elementów w górno-trójkątnej macierzy Q wynosi 14 365, z których 4 902 jest niezerowych, co stanowi 34%. A więc macierz ta jest gęstsza niż macierz rozwiązanego problemu QUBO dla algorytmu AES. Oczekiwana wartość energii minimalnej dla tego problemu wynosi: $-38\,203$. Czas wyżarzania ustalono na 20 minut, w ciągu których otrzymano rozwiązanie o energii minimalnej równej $-37\,530$. Otrzymany wynik wyżarzania nie zgadza się z zastosowanym kluczem głównym.

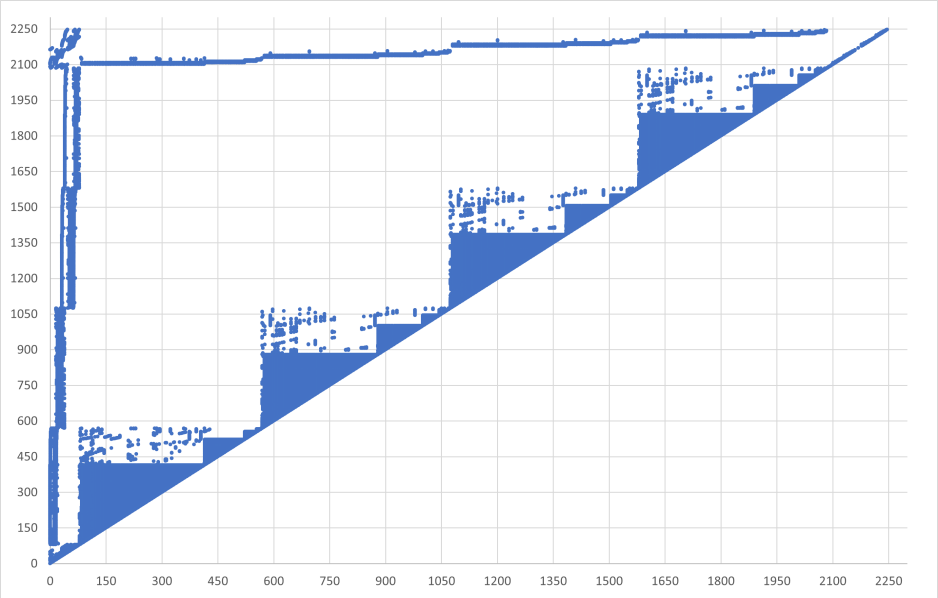


Rysunek 57: Struktura macierzy Q problemu QUBO wygenerowanego dla dwurundowego szyfru Speck8/8, opisanego nad $\mathbb{Z}_{2^4}$.

Ponieważ nie otrzymano prawidłowego rozwiązania, zmniejszono parametry szyfru Speck do następujących:

- długość bloku: $2n = 8$ bitów,
- długość słowa: $n = 4$ bitów,
- długość klucza: $mn = 8$ bitów,
- liczba słów klucza: $m = 2$,
- liczba bitów dla rotacji w prawo: $\alpha = 1$,
- liczba bitów dla rotacji w lewo: $\beta = 2$,
- liczba rund: $T = 2$.

Otrzymano problem QUBO składający się z 56 zmiennych binarnych, którego macierz przedstawiono na rysunku 57. W celu ukazania wielkości tego problemu, na wykresie zachowano maksymalny zakres osi równy 200, jako układ odniesienia. Problem ten został rozwiązany w czasie 5 minut.



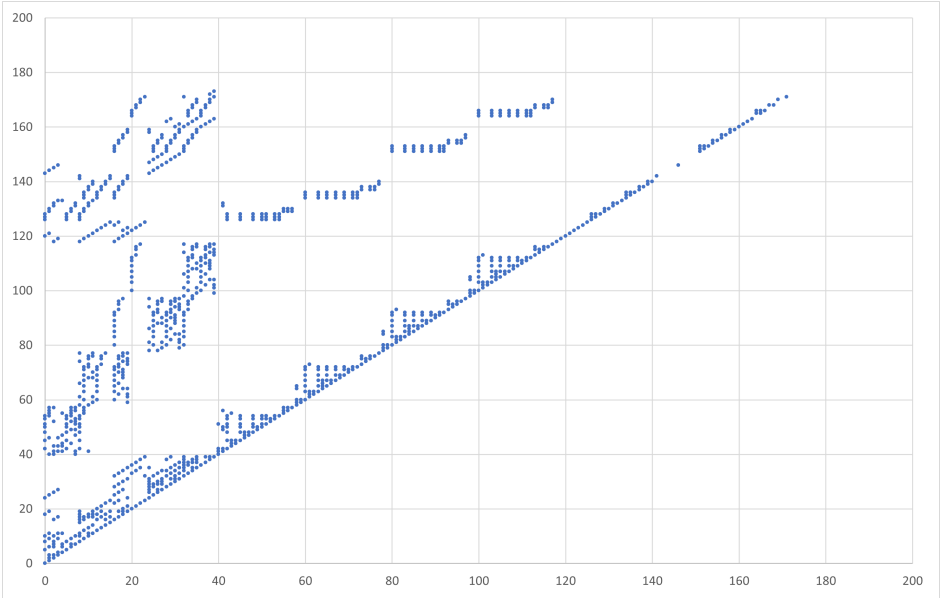
Rysunek 58: Struktura macierzy Q problemu QUBO wygenerowanego dla dwurundowego szyfru Speck16/16, opisanego nad $GF(2)$.

Ponieważ macierz Q problemu QUBO dla szyfru Speck16/16, opisanego nad pierścieniem $\mathbb{Z}_{2^8}$, okazała się gęstsza niż dla szyfru AES oraz skalowane współczynniki okazały się bardzo małe, wygenerowano problem QUBO dla szyfru Speck16/16 o identycznych parametrach jak poprzednio, opisanego za pomocą równań nad ciałem binarnym. Otrzymano problem QUBO składający się z 2 248 zmiennych binarnych, którego strukturę macierzy Q przedstawiono na rysunku 58. Elementy tej macierzy przyjmują wartości z zakresu od 1 do 4 096, a po skalowaniu przyjmują wartości z zakresu od $2{,}4 \cdot 10^{-4}$. Liczba niezerowych elementów macierzy wynosi 60 585 z 2 527 876 wszystkich elementów, co stanowi 2,4%. A więc macierz ta jest znacznie rzadsza niż macierz problemu QUBO dla tego szyfru opisanego nad pierścieniem $\mathbb{Z}_{2^8}$.

Niestety, rozmiar otrzymanego problemu QUBO jest zbyt duży, by uzyskać poprawny wynik w ciągu 20 minut. Dlatego też wygenerowano kolejny problem QUBO,

dla mniejszej instancji szyfru Speck, opisanego nad ciałem binarnym. Instancja ta ma następujące parametry:

- długość bloku: $2n = 8$ bitów,
- długość słowa: $n = 4$ bitów,
- długość klucza: $mn = 8$ bitów,
- liczba słów klucza: $m = 2$,
- liczba bitów dla rotacji w prawo: $\alpha = 3$,
- liczba bitów dla rotacji w lewo: $\beta = 2$,
- liczba rund: $T = 3$.



Rysunek 59: Struktura macierzy Q problemu QUBO wygenerowanego dla trzyrundowego szyfru Speck8/8, opisanego nad $GF(2)$.

Po przetransformowaniu układu równań wielomianowych nad ciałem binarnym do problemu QUBO, otrzymano problem ze 174 zmiennymi binarnymi. Strukturę macierzy Q otrzymanego problemu przedstawiono na rysunku 59. Składa się ona z 1 023 niezerowych współczynników, co stanowi 6,7% elementów macierzy Q. Wartości

otrzymanych współczynników są z zakresu od 1 do 39, co po skalowaniu daje zakres od $0,026$ do 1. Problem ten został rozwiązany w ciągu 20 minut na wyżarzaczu kwantowym D-Wave oraz otrzymano poprawny klucz główny.

6 Analiza szyfru blokowego typu Feistel na przykładzie szyfru Simon

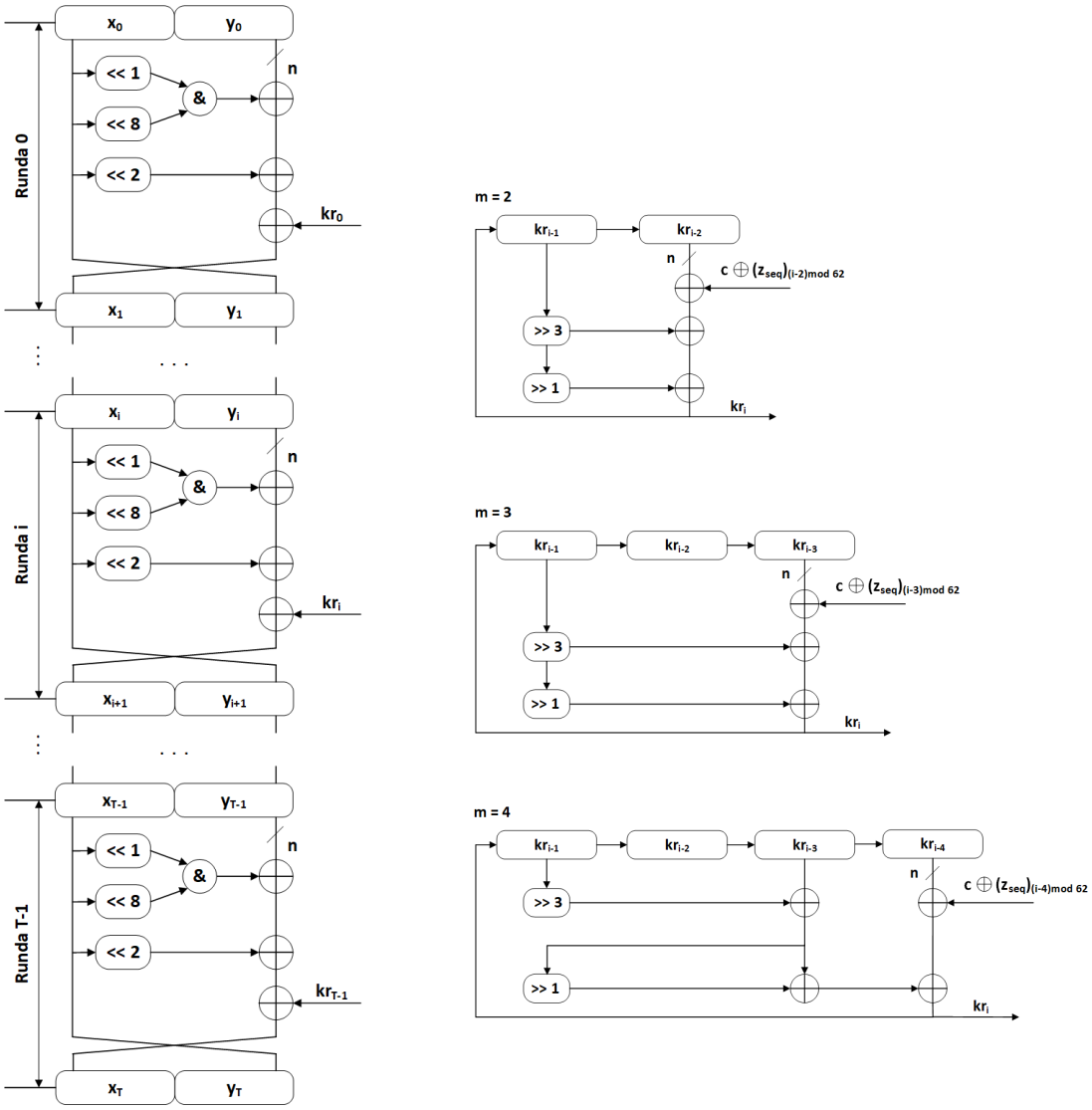
Ostatnim przeanalizowanym w niniejszej rozprawie szyfrem jest szyfr Simon, posiadający strukturę typu Feistel. Szyfr ten został zaprojektowany jako lekki szyfr blokowy oraz zaprezentowany, razem z szyfrem Speck, w pracy [3]. Szyfr Simon, tak samo jak szyfr Speck, jest rodziną szyfrów o różnych parametrach. W tabeli 17 przedstawiono parametry wszystkich wariantów szyfru Simon. Analogicznie do szyfru Speck, oznaczmy poprzez Simon$2n/mn$ instancję szyfru Simon, gdzie $2n$ oznacza długość bloku wejściowego, n – długość słowa oraz mn – długość klucza głównego.

Tabela 17: Parametry szyfru Simon$2n/mn$. *Źródło: [3]*

Rozmiar bloku ($2n$)	*Rozmiar klucza* (mn)	*Rozmiar słowa* (n)	*Liczba słów klucza* (m)	*Stała dla kluczy rundowych* (z_{seq})	*Liczba rund* (T)
32	64	16	4	z_0	32
48	72	24	3	z_0	36
	96		4	z_1	36
64	96	32	3	z_2	42
	128		4	z_3	44
96	96	48	2	z_2	52
	144		3	z_3	54
128	128	64	2	z_2	68
	192		3	z_3	69
	256		4	z_4	72

Na rysunku 60 przedstawiono schemat algorytmu szyfrowania (60a) oraz schemat algorytmu generowania kluczy rundowych (60b) w zależności od wartości parametru m. Szyfr Simon$2n/mn$ wykorzystuje następujące operacje bitowe, realizowane na n-bitowych słowach:

- bitowa operacja xor, oznaczona jako $\oplus$,
- bitowa operacja and, oznaczona jako &,
- rotacje, w prawą i lewą stronę o j bitów, oznaczone odpowiednio jako $\gg j$ oraz $\ll j$.



(a) Schemat algorytmu szyfrowania szyfru Simon$2n/mn$.

(b) Schematy algorytmu generowania kluczy rundowych szyfru Simon$2n/mn$.

Rysunek 60: Ogólny schemat budowy szyfru Simon$2n/mn$. *Źródło: na podstawie [3]*

Operacją nieliniową jest bitowa operacja and, natomiast pozostałe operacje są operacjami liniowymi. Liczba rund oznaczona została literą T i należy ona do przedziału od 32 do 72, w zależności od instancji szyfru Simon.

6.1 Algorytm szyfrowania Simon

Dla danego klucza rundowego $kr \in \mathbb{Z}_{2^n}$ funkcja rundy algorytmu szyfrowania szyfru Simon$2n/mn$ jest odwzorowaniem $R_{kr} : \mathbb{Z}_{2^n} \times \mathbb{Z}_{2^n} \to \mathbb{Z}_{2^n} \times \mathbb{Z}_{2^n}$, zdefiniowanym

w następujący sposób:

$$R_{kr}\left(x_{i+1}, y_{i+1}\right) = \left(y_i \oplus F(x_i) \oplus kr_i, x_i\right), \tag{128}$$

gdzie kr_i jest kluczem rundowym, i jest numerem rundy, a funkcja F zdefiniowana została następująco:

$$F(x_i) = \left(\left(x_i \ll 1\right) \& \left(x_i \ll 8\right)\right) \oplus \left(x_i \ll 2\right).$$

Każda instancja szyfru Simon$2n/mn$ wykorzystuje znaną zasadę przetwarzania danych, zgodnie z klasyczną siecią Feistela, gdzie w danej rundzie przetwarzana jest tylko połowa stanu wejściowego. Jak pokazano na rysunku 60a, zgodnie z ideą Feistela, $2n$-bitowy stan wejściowy do i-tej rundy dzielony jest na dwa n-bitowe słowa, zwane gałęziami, gdzie n najbardziej znaczących bitów tworzy słowo x_i, zwane gałęzią źródłową, a n najmniej znaczących bitów tworzy słowo y_i, zwane gałęzią docelową. Gałąź źródłowa przetwarzana jest za pomocą funkcji F, która w przypadku szyfru Simon polega na równoległym wykonaniu rotacji w lewo o 1, 8 i 2 bity. Na wynikach dwóch pierwszych rotacji wykonywana jest bitowa operacja and, której wynik xorowany jest z wynikiem trzeciej rotacji. W ten sposób otrzymywana jest wartość funkcji $F(x_i)$, która następnie xorowana jest z kluczem rundowym i-tej rundy oraz z gałęzią docelową, tworząc gałąź źródłową x_{i+1} kolejnej rundy. Gałęzią docelową $i+1$-wszej rundy jest nieprzetworzona gałąź źródłowa i-tej rundy: $y_{i+1} = x_i$.

6.2 Algorytm generowania kluczy rundowych szyfru Simon

Klucz główny K szyfru Simon$2n/mn$ dzielony jest na m, n-bitowych kluczy rundowych, w następujący sposób: $K = [kr_{m-1}, ..., kr_0]$, gdzie $kr_i \in \mathbb{Z}_{2^n}$. Na podstawie klucza głównego, przy użyciu rotacji w prawo oraz operacji xor, generowane są pozostałe $T - m$ klucze rundowe. Na rysunku 60b przedstawiono algorytm generowania kluczy rundowych szyfru Simon, dla trzech możliwych wartości parametru m. Aby wyznaczyć klucz i-tej rundy kr_i, potrzebna jest znajomość m wcześniejszych kluczy rundowych. Sposób wyznaczenia i-tego klucza rundowego jest identyczny dla $m = 2$ i $m = 3$:

klucz poprzedni, kr_{i-1}, przesuwany jest cyklicznie w prawo o 3 i o 4 bity, a następnie otrzymane wyniki xorowane są z kluczem rundowym kr_{i-m} i stałą $c \oplus (z_{seq})_{(i-m)mod\ 62}$, czego wynikiem jest klucz rundowy kr_i. W przypadku gdy $m = 4$, klucz kr_{i-1} przesuwany jest cyklicznie w prawo tylko raz, o 3 bity, po czym xorowany jest z kluczem kr_{i-3}. Otrzymany wynik xorowany jest ze swoim cyklicznym przesunięciem w prawo o 1 bit i dopiero xorowany jest z kluczem kr_{i-4} i stałą $c \oplus (z_{seq})_{(i-m)mod\ 62}$, czego wynikiem jest klucz rundowy kr_i, gdzie:

- stała c to n-bitowa liczba, równa $2^n - 4 = 0xfff...fc$,
- stała $(z_{seq})_{(i-m)mod\ 62}$ jest bitem o numerze $(i-m)mod\ 62$, sekwencji z_{seq}.

Wyznaczono pięć okresowych sekwencji: z_0, z_1, z_2, z_3 oraz z_4, o okresach 31 albo 62. Sekwencje z_0 i z_1, postaci:

$$z_0 = (z_0)_0(z_0)_1(z_0)_2... = 11111010001001010110000111001 10...,$$

$$z_1 = (z_1)_0(z_1)_1(z_1)_2... = 10001110111110010011000010110 10...,$$

mają okres 31, ponadto generowane są za pomocą 5-bitowych LFSRów oraz wykorzystywane są dla instancji szyfru Simon$2n/mn$ o długości bloku równej 32 i 48 bitów (patrz tabela 17). Sekwencje z_2, z_3 i z_4, o okresie równym 62, wykorzystywane są dla instancji szyfru Simon$2n/mn$ o blokach długości 64, 96 oraz 128. Generowane są za pomocą bitowej operacji xor sekwencji 01010101..., o okresie 2, z sekwencją:

- z_0, w celu otrzymania sekwencji z_2,
- z_1, w celu otrzymania sekwencji z_3 oraz
- z sekwencją 10000100101100111110001101110 10..., o okresie 31, w celu wygenerowania sekwencji z_4.

W wyniku tego dostajemy:

$$z_2 = (z_2)_0(z_2)_1(z_2)_2... =$$
$$10101111011100000011010010011 00 01010000100011111100101101100 11...,$$

$$z_3 = (z_3)_0(z_3)_1(z_3)_2... =$$
$$11011011101011000110010111100000010010001010011100110100001111...,$$

$$z_4 = (z_4)_0(z_4)_1(z_4)_2... =$$
$$11010001111001101011011000100000010111000011001010010011101111...,$$

gdzie $(z_i)_j$ jest j-tym bitem sekwencji z_i. Celem operacji xor z jednobitową stałą $(z_{seq})_{(i-m)mod\ 62}$, zależną od numeru rundy, jest wyeliminowanie symetrii operacji cyklicznego przesunięcia oraz podatności na atak z przesunięciem.

6.3 Analiza przedstawienia szyfru Simon za pomocą układu równań wielomianowych o wielu zmiennych nad ciałem $GF(2)$

Ze względu na bitowy charakter wszystkich operacji szyfru Simon, naturalną strukturą algebraiczną, nad którą zostaną przedstawione równania wielomianowe opisujące ten szyfr jest ciało binarne $GF(2)$.

6.3.1 Wyznaczanie równań nad ciałem $GF(2)$, opisujących algorytm szyfrowania szyfru Simon

Analogicznie do analizy szyfru Speck załóżmy, że bity kluczy rundowych przedstawione są za pomocą zmiennych binarnych, a równania wielomianowe generowane są nad ciałem binarnym dla każdego bitu stanu pośredniego każdej rundy szyfru Simon. Niech n-bitowe słowa a i b przedstawione będą jako ciąg bitów, postaci:

$$a = \left(a_{n-1}, a_{n-2}, \ldots, a_1, a_0\right),$$

$$b = \left(b_{n-1}, b_{n-2}, \ldots, b_1, b_0\right).$$

gdzie a_j i b_j są bitami słów. Rotacja słowa a w lewo o 1, 2 i 8 bitów realizowana jest jako cykliczne przesunięcie bitów w ciągu, zgodnie z następującymi równaniami:

$$a \ll 1 = \left(a_{n-2}, a_{n-3}, \ldots, a_1, a_0, a_{n-1}\right), \tag{129}$$

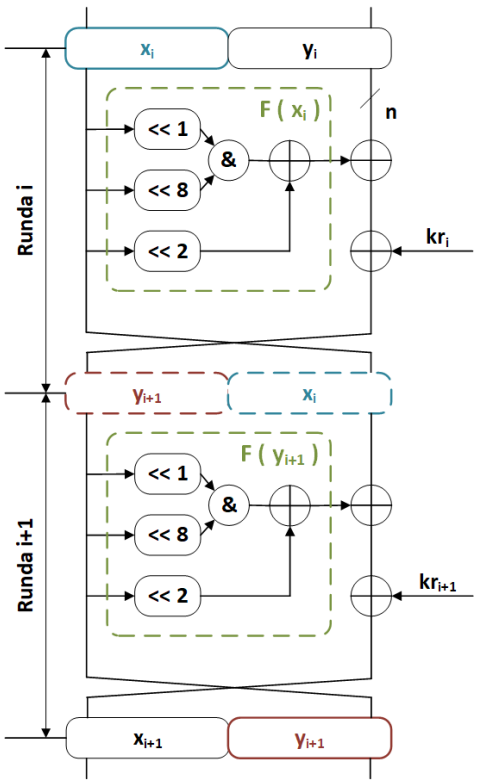
$$a \ll 2 = \left(a_{n-3}, a_{n-4}, \ldots, a_1, a_0, a_{n-1}, a_{n-2}\right), \tag{130}$$

$$a \ll 8 = \left(a_{n-9}, a_{n-10}, \ldots, a_{n-6}, a_{n-7}, a_{n-8}\right). \tag{131}$$

Bitowe operacje xor i and dwóch bitów a_j i b_j można zrealizować jako, odpowiednio, dodawanie i mnożenie w ciele binarnym. Stąd operacja xor/and dla dwóch n-bitowych słów a i b realizowana jest jako dodawanie/mnożenie odpowiadających sobie bitów słów. Równanie przedstawiające operację xor dwóch n-bitowych słów w ciele binarnym, pokazano za pomocą równania (116), podczas analizowania szyfru Speck. Operację and dwóch n-bitowych słów w ciele binarnym można przedstawić w postaci następującego równania:

$$a \,\&\, b = \left(a_{n-1}b_{n-1}, a_{n-2}b_{n-2}, \ldots, a_1b_1, a_0b_0\right). \tag{132}$$

Podczas wyznaczania równań wielomianowych, opisujących algorytm szyfrowania Simon, wykorzystano strukturę Feistela, łącząc parami sąsiednie rundy i uzyskując tym samym o połowę mniejszą liczbę równań. Na rysunku 61 pokazano sposób połączenia dwóch sąsiednich rund za pomocą zmiennych reprezentujących stany pośrednie. Słowa x_i, y_i, x_{i+1} oraz y_{i+1} tworzą stany pośrednie pomiędzy podwójnymi rundami i są reprezentowane poprzez zmienne binarne. Słowo x_i, oznaczone kolorem niebieskim, będące gałęzią źródłową rundy i, zostaje przetworzone przez funkcję F, której operacje na rysunku 61 ograniczono zieloną, przerywaną linią. Jednocześnie, zmienne binarne reprezentujące słowo x_i tworzą gałąź docelową rundy $i+1$, co przedstawiono stanem, oznaczonym niebieską, przerywaną linią. Podobnie słowo y_{i+1}, oznaczone kolorem czerwonym i reprezentowane przez zmienne binarne, jest jednocześnie gałęzią źródłową rundy $i+1$, przedstawioną stanem, oznaczonym przerywaną linią koloru czerwonego oraz jest jednocześnie wynikiem przetwarzania gałęzi docelowej y_i rundy i. Na podstawie rundy i zdefiniowano zmienne binarne słowa y_{i+1} jako $y_{i+1} = F(x_i) \oplus y_i \oplus kr_i$, na podstawie rundy $i+1$ zdefiniowano słowo x_{i+1}, jako $x_{i+1} = F(y_{i+1}) \oplus x_i \oplus kr_{i+1}$. Wykorzystując te zależności, skonstruowano układ równań wielomianowych opisujący podwójną rundę algorytmu szyfrowania Simon$2n/mn$. Równania te wyznaczane są nad ciałem binarnym, po jednym



Rysunek 61: Schemat połączenia dwóch sąsiednich rund szyfru Simon2n/mn.

równaniu dla każdego bitu stanu pośredniego. Przedstawmy słowa stanów pośrednich jako ciągi bitów następującej postaci:

$$x_i = \left(x_{i_{n-1}}, x_{i_{n-2}}, \ldots, x_{i_1}, x_{i_0}\right),$$

$$y_i = \left(y_{i_{n-1}}, y_{i_{n-2}}, \ldots, y_{i_1}, y_{i_0}\right),$$

$$x_{i+1} = \left(x_{i+1_{n-1}}, x_{i+1_{n-2}}, \ldots, x_{i+1_1}, x_{i+1_0}\right),$$

$$y_{i+1} = \left(y_{i+1_{n-1}}, y_{i+1_{n-2}}, \ldots, y_{i+1_1}, y_{i+1_0}\right).$$

Układ równań nad $GF(2)$, opisujący podwójną rundę algorytmu szyfrowania Simon,

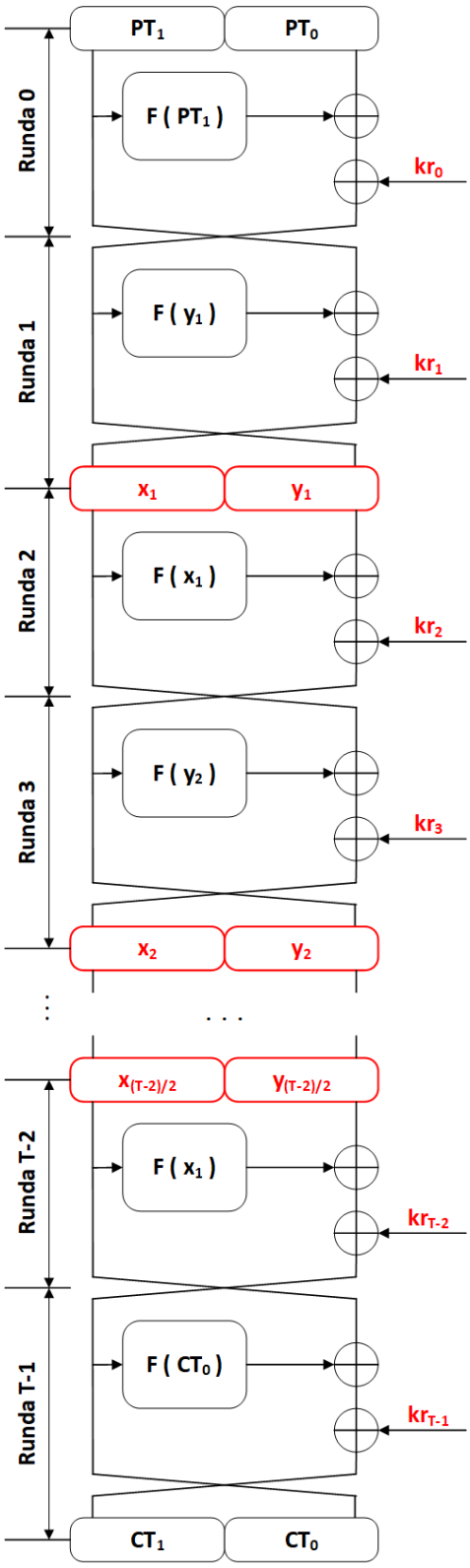
składa się z $2n$ równań i ma następującą postać:

$$\begin{cases} f_1 : y_{i+1_{n-2}} y_{i+1_{n-9}} + y_{i+1_{n-3}} + x_{i_{n-1}} + kr_{i+1_{n-1}} + x_{i+1_{n-1}} = 0, \\ f_2 : y_{i+1_{n-3}} y_{i+1_{n-10}} + y_{i+1_{n-4}} + x_{i_{n-2}} + kr_{i+1_{n-2}} + x_{i+1_{n-2}} = 0, \\ \vdots \\ f_{n-2} : y_{i+1_1} y_{i+1_{n-6}} + y_{i+1_0} + x_{i_2} + kr_{i+1_2} + x_{i+1_2} = 0, \\ f_{n-1} : y_{i+1_0} y_{i+1_{n-7}} + y_{i+1_{n-1}} + x_{i_1} + kr_{i+1_1} + x_{i+1_1} = 0, \\ f_n : y_{i+1_{n-1}} y_{i+1_{n-8}} + y_{i+1_{n-2}} + x_{i_0} + kr_{i+1_0} + x_{i+1_0} = 0, \\ f_{n+1} : x_{i_{n-2}} x_{i_{n-9}} + x_{i_{n-3}} + y_{i_{n-1}} + kr_{i_{n-1}} + y_{i+1_{n-1}} = 0, \\ f_{n+2} : x_{i_{n-3}} x_{i_{n-10}} + x_{i_{n-4}} + y_{i_{n-2}} + kr_{i_{n-2}} + y_{i+1_{n-2}} = 0, \\ \vdots \\ f_{2n-2} : x_{i_1} x_{i_{n-6}} + x_{i_0} + y_{i_2} + kr_{i_2} + y_{i+1_2} = 0, \\ f_{2n-1} : x_{i_0} x_{i_{n-7}} + x_{i_{n-1}} + y_{i_1} + kr_{i_1} + y_{i+1_1} = 0, \\ f_{2n} : x_{i_{n-1}} x_{i_{n-8}} + x_{i_{n-2}} + y_{i_0} + kr_{i_0} + y_{i+1_0} = 0, \end{cases} \tag{133}$$

gdzie i jest numerem pierwszej rundy z pary. Równania od f_1 do f_n opisują wyznaczanie słowa x_{i+1}, a równania od f_{n+1} do f_{2n} opisują wyznaczanie słowa y_{i+1}.

Rozpatrując stopień uzyskanych wielomianów, można zauważyć, że wszystkie równania układu (133) mają stopień 2, przy czym każde równanie posiada dokładnie jeden jednomian kwadratowy. Co za tym idzie, podczas linearyzacji, dla jednej podwójnej rundy algorytmu szyfrowania wymaganych jest $2n$ dodatkowych zmiennych binarnych. Zatem, również w przypadku szyfru Simon, proces linearyzacji nie jest problemem NP-trudnym.

Rozszerzając powyższy opis na cały algorytm szyfrowania, na rysunku 62 kolorem czerwonym oznaczono klucze rundowe oraz stany pośrednie pomiędzy podwójnymi rundami, które reprezentowane są za pomocą zmiennych binarnych. Liczba dodatkowych zmiennych binarnych jest sumą liczby zmiennych dla stanów pośrednich (słowa od x_1 do $x_{(T-2)/2}$ oraz od y_1 do $y_{(T-2)/2}$) wynoszącej $(T-2)n$ oraz liczby zmiennych binarnych dla kluczy rundowych (słowa od kr_m do kr_{T-1}) wynoszącej $(T-m)n$. Na rysunku 62 oznaczono również znane bity tekstu jawnego oraz odpowiadającego mu szyfrogramu jako słowa PT_1, PT_0, CT_1 i CT_0. Przy uwzględnieniu



Rysunek 62: Algorytm szyfrowania szyfru Simon2n/mn z oznaczonymi stanami pośrednimi.

znajomości tych bitów, układ równań opisujący pierwszą parę rund ma następującą postać:

$$\begin{cases} f_1 : y_{1_{n-2}} y_{1_{n-9}} + y_{1_{n-3}} + PT_{1_{n-1}} + kr_{1_{n-1}} + x_{1_{n-1}} = 0, \\ f_2 : y_{1_{n-3}} y_{1_{n-10}} + y_{1_{n-4}} + PT_{1_{n-2}} + kr_{1_{n-2}} + x_{1_{n-2}} = 0, \\ \vdots \\ f_{n-2} : y_{1_1} y_{1_{n-6}} + y_{1_0} + PT_{1_2} + kr_{1_2} + x_{1_2} = 0, \\ f_{n-1} : y_{1_0} y_{1_{n-7}} + y_{1_{n-1}} + PT_{1_1} + kr_{1_1} + x_{1_1} = 0, \\ f_n : y_{1_{n-1}} y_{1_{n-8}} + y_{1_{n-2}} + PT_{1_0} + kr_{1_0} + x_{1_0} = 0, \\ f_{n+1} : PT_{1_{n-2}} PT_{1_{n-9}} + PT_{1_{n-3}} + PT_{0_{n-1}} + kr_{0_{n-1}} + y_{1_{n-1}} = 0, \\ f_{n+2} : PT_{1_{n-3}} PT_{1_{n-10}} + PT_{1_{n-4}} + PT_{0_{n-2}} + kr_{0_{n-2}} + y_{1_{n-2}} = 0, \\ \vdots \\ f_{2n-2} : PT_{1_1} PT_{1_{n-6}} + PT_{1_0} + PT_{0_2} + kr_{0_2} + y_{1_2} = 0, \\ f_{2n-1} : PT_{1_0} PT_{1_{n-7}} + PT_{1_{n-1}} + PT_{0_1} + kr_{0_1} + y_{1_1} = 0, \\ f_{2n} : PT_{1_{n-1}} PT_{1_{n-8}} + PT_{1_{n-2}} + PT_{0_0} + kr_{0_0} + y_{1_0} = 0. \end{cases} \tag{134}$$

W układzie tym n pierwszych równań to równania kwadratowe, z których każde posiada dokładnie jeden jednomian stopnia 2. Natomiast n ostatnich równań to równania liniowe, niewymagające wykonania procesu linearyzacji. Stąd proces linearyzacji równań pierwszej pary rund wymaga zdefiniowania dodatkowych n zmiennych binarnych.

Podobnie ostatnia para rund, ze względu na znajomość bitów szyfrogramu, jest opisana za pomocą układu równań postaci:

$$\begin{cases} f_1 : CT_{0_{n-2}} CT_{0_{n-9}} + CT_{0_{n-3}} + x_{(T-2)/2_{n-1}} + kr_{T-1_{n-1}} + CT_{1_{n-1}} = 0, \\ f_2 : CT_{0_{n-3}} CT_{0_{n-10}} + CT_{0_{n-4}} + x_{(T-2)/2_{n-2}} + kr_{T-1_{n-2}} + CT_{1_{n-2}} = 0, \\ \vdots \\ f_{n-1} : CT_{0_0} CT_{0_{n-7}} + CT_{0_{n-1}} + x_{(T-2)/2_1} + kr_{T-1_1} + CT_{1_1} = 0, \\ f_n : CT_{0_{n-1}} CT_{0_{n-8}} + CT_{0_{n-2}} + x_{(T-2)/2_0} + kr_{T-1_0} + CT_{1_0} = 0, \\ f_{n+1} : x_{(T-2)/2_{n-2}} x_{(T-2)/2_{n-9}} + x_{(T-2)/2_{n-3}} + y_{(T-2)/2_{n-1}} + kr_{T-2_{n-1}} + CT_{0_{n-1}} = 0, \\ f_{n+2} : x_{(T-2)/2_{n-3}} x_{(T-2)/2_{n-10}} + x_{(T-2)/2_{n-4}} + y_{(T-2)/2_{n-2}} + kr_{T-2_{n-2}} + CT_{0_{n-2}} = 0, \\ \vdots \\ f_{2n-1} : x_{(T-2)/2_0} x_{(T-2)/2_{n-7}} + x_{(T-2)/2_{n-1}} + y_{(T-2)/2_1} + kr_{T-2_1} + CT_{0_1} = 0, \\ f_{2n} : x_{(T-2)/2_{n-1}} x_{(T-2)/2_{n-8}} + x_{(T-2)/2_{n-2}} + y_{(T-2)/2_0} + kr_{T-2_0} + CT_{0_0} = 0, \end{cases} \tag{135}$$

z których n pierwszych równań to równania liniowe, natomiast n ostatnich to równania kwadratowe, z jednym jednomianem kwadratowym w każdym równaniu. Podczas linearyzacji równań reprezentujących ostatnią parę rund, wymaganych jest również n dodatkowych zmiennych binarnych.

Każda para, z pozostałych $(T-4)/2$ par rund, reprezentowana jest za pomocą układu równań postaci (133).

6.3.2 Wyznaczanie równań nad ciałem $GF(2)$, opisujących algorytm generowania kluczy rundowych szyfru Simon

Jak już opisano wcześniej, algorytm generowania kluczy rundowych szyfru Simon$2n/mn$ wykorzystuje bitową operację xor oraz rotację w prawo o 1 i 3 bity. Pierwszych m kluczy rundowych zawiera się w kluczu głównym, a następne klucze kr_i, dla $i = \overline{m, T-1}$, generowane są za pomocą algorytmu generowania kluczy rundowych. Wszystkie klucze rundowe przedstawione są za pomocą zmiennych binarnych i nie są wymagane dodatkowe stany pośrednie. Stąd liczba zmiennych, potrzebnych do wygenerowania układu równań wielomianowych opisujących algorytm generowania kluczy rundo-

wych, wynosi $(T-1)n$. Równania opisujące generowanie kluczy rundowych wyznaczane są dla każdego bitu klucza rundowego kr_i, dla $i = \overline{m, T-1}$.

Niech n-bitowy klucz rundowy kr_i przedstawiony będzie za pomocą ciągu bitów postaci:

$$kr_i = \left(kr_{i_{n-1}}, kr_{i_{n-2}}, \ldots, kr_{i_1}, kr_{i_0} \right),$$

gdzie kr_{i_j} jest j-tym bitem klucza rundowego rundy i. Rotacja klucza rundowego w prawo o 1 i 3 bity realizowana jest, analogicznie do algorytmu szyfrowania, jako cykliczne przesunięcie bitów ciągu, co przedstawiono za pomocą następujących równań:

$$kr_i \gg 1 = \left(kr_{i_0}, kr_{i_{n-1}}, kr_{i_{n-2}}, \ldots, kr_{i_2}, kr_{i_1} \right), \tag{136}$$

$$kr_i \gg 3 = \left(kr_{i_2}, kr_{i_1}, kr_{i_0}, kr_{i_{n-1}}, \ldots, kr_{i_4}, kr_{i_3} \right), \tag{137}$$

natomiast bitowa operacja xor realizowana jest zgodnie z równaniem (116). Dany klucz rundowy kr_i wyznaczany jest na podstawie równania:

$$kr_i = kr_{i-m} \oplus \left(kr_{i-1} \gg 3 \right) \oplus \left(kr_{i-1} \gg 4 \right) \oplus c \oplus \left(z_{seq} \right)_{(i-m) mod\ 62}, \tag{138}$$

gdy $m = 2$ i $m = 3$ lub na podstawie równania:

$$kr_i = kr_{i-m} \oplus kr_{i-3} \oplus \left(kr_{i-1} \gg 3 \right) \oplus \left(kr_{i-1} \gg 4 \right) \oplus \left(kr_{i-3} \gg 1 \right) \oplus c \oplus (z_{seq})_{(i-m) mod\ 62}, \tag{139}$$

gdy $m = 4$. Za pomocą powyższych równań, zawierających działania realizowane na n-bitowych słowach, wygenerowano układ n równań wielomianowych, nad ciałem binarnym, dla klucza rundowego kr_i. Otrzymany układ ma następującą postać:

$$\begin{cases} kr_{i-m_{n-1}} + kr_{i-1_2} + kr_{i-1_3} + c_{n-1} + (z_{seq})_{(i-m) mod\ 62_{n-1}} + kr_{i_{n-1}} = 0, \\ kr_{i-m_{n-2}} + kr_{i-1_1} + kr_{i-1_2} + c_{n-2} + (z_{seq})_{(i-m) mod\ 62_{n-2}} + kr_{i_{n-2}} = 0, \\ \vdots \\ kr_{i-m_1} + kr_{i-1_4} + kr_{i-1_5} + c_1 + (z_{seq})_{(i-m) mod\ 62_1} + kr_{i_1} = 0, \\ kr_{i-m_0} + kr_{i-1_3} + kr_{i-1_4} + c_0 + (z_{seq})_{(i-m) mod\ 62_0} + kr_{i_0} = 0, \end{cases} \tag{140}$$

dla $m = 2$ i $m = 3$ lub

$$\begin{cases} kr_{i-m_{n-1}} + kr_{i-3_{n-1}} + kr_{i-1_2} + kr_{i-1_3} + kr_{i-3_0} + c_{n-1} + (z_{seq})_{(i-m) mod\ 62_{n-1}} + kr_{i_{n-1}} = 0, \\ kr_{i-m_{n-2}} + kr_{i-3_{n-2}} + kr_{i-1_1} + kr_{i-1_2} + kr_{i-3_{n-1}} + c_{n-2} + (z_{seq})_{(i-m) mod\ 62_{n-2}} + kr_{i_{n-2}} = 0, \\ \vdots \\ kr_{i-m_1} + kr_{i-3_1} + kr_{i-1_4} + kr_{i-1_5} + kr_{i-3_2} + c_1 + (z_{seq})_{(i-m) mod\ 62_1} + kr_{i_1} = 0, \\ kr_{i-m_0} + kr_{i-3_0} + kr_{i-1_3} + kr_{i-1_4} + kr_{i-3_1} + c_0 + (z_{seq})_{(i-m) mod\ 62_0} + kr_{i_0} = 0, \end{cases} \tag{141}$$

dla $m = 4$. Wszystkie wygenerowane równania są liniowe, dlatego też podczas linearyzacji nie są wymagane dodatkowe zmienne binarne. Liczba równań w układzie opisującym cały algorytm generowania kluczy rundowych wynosi $(T-m)n$.

6.3.3 Konstrukcja układu równań nad ciałem $GF(2)$, opisującego szyfr Simon, gdy bity kluczy rundowych reprezentowane są za pomocą zmiennych binarnych

W pierwszym przypadku konstruowania układu równań, opisującego szyfr Simon, bity kluczy rundowych reprezentowane są jako zmienne binarne. Wtedy liczba zmiennych binarnych, potrzebnych do wygenerowania układu równań dla T-rundowego szyfru Simon$2n/mn$, jest równa $mn+(T-m)n+(T-2)n$, gdy długość klucza jest równa długości bloku wejściowego oraz $mn+(T-m)n+2(T-2)n$, gdy długość klucza jest większa niż długość bloku wejściowego, gdzie:

- mn – liczba zmiennych binarnych dla klucza głównego,
- $(T-m)n$ – liczba zmiennych binarnych dla kluczy rundowych,
- $(T-2)n$ – liczba zmiennych binarnych dla stanów pośrednich algorytmów szyfrowania, dla jednej pary tekst jawny – szyfrogram.

Układ opisujący T-rundowy szyfr Simon$2n/mn$, gdy długość klucza głównego jest równa długości bloku wejściowego, składa się z:

- n równań kwadratowych i n równań liniowych, postaci (134),

- $(T-4)n$ równań stopnia 2, postaci (133) oraz
- n równań liniowych i n równań kwadratowych, postaci (135),

dla algorytmu szyfrowania oraz

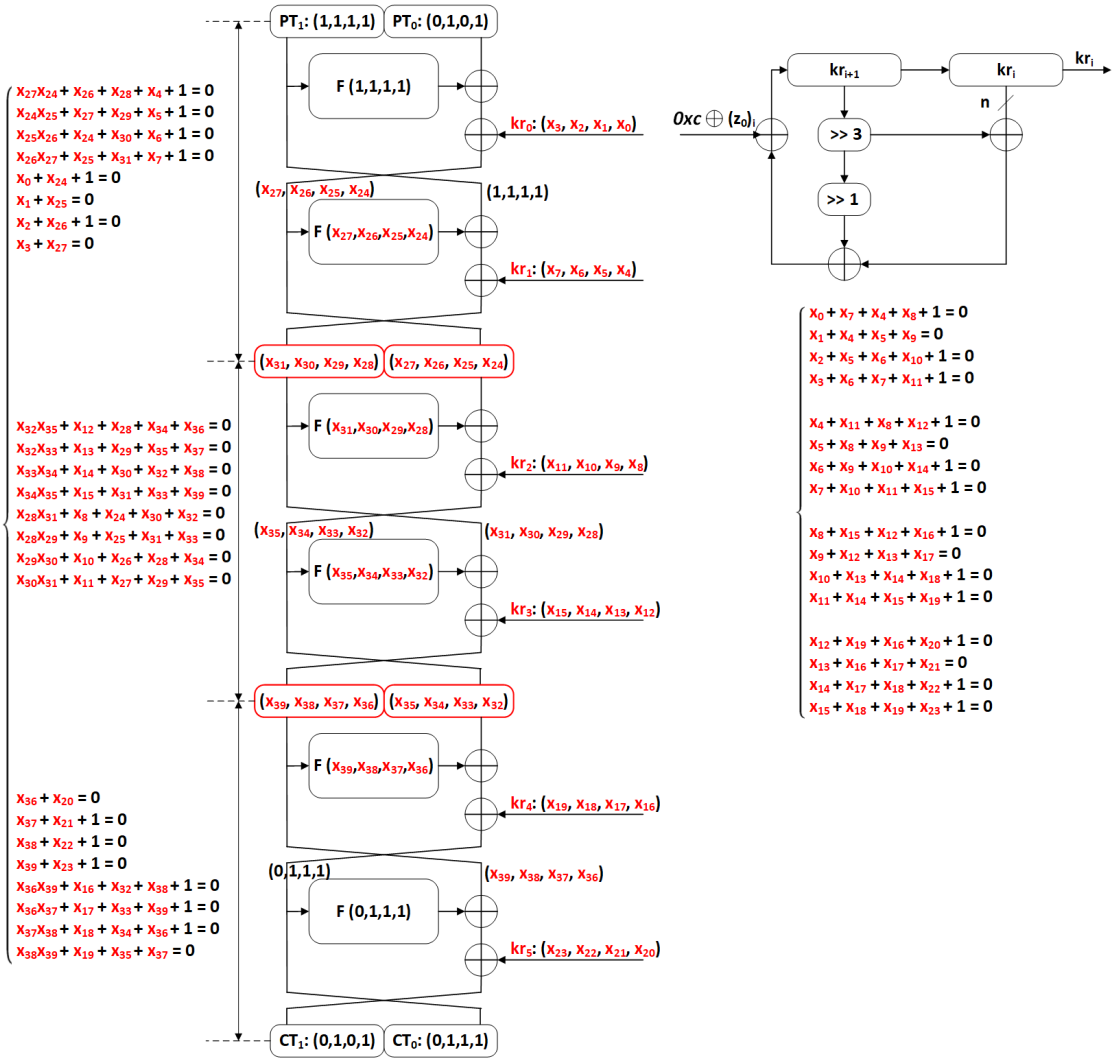
- $(T-m)n$ liniowych, postaci (140) albo (141),

dla algorytmu generowania kluczy rundowych. W przypadku, gdy długość klucza głównego jest większa niż długość bloku wejściowego, równania opisujące algorytm szyfrowania wyznaczane są dla dwóch różnych par tekst jawny – szyfrogram, a ich liczba jest dwa razy większa niż powyżej. Podczas linearyzacji wyznaczonego układu opisującego szyfr wymaganych jest $(T-2)n$ dodatkowych zmiennych binarnych, dla jednej pary tekst jawny – szyfrogram, albo $2(T-2)n$ dla dwóch różnych par.

Rozpatrzmy kolejne przekształcenie podczas transformacji układu do problemu QUBO, czyli przedstawienie, za pomocą zmiennych binarnych, wartości krotności k_i dla każdego równania. Liczbę tę łatwo wyznaczyć analitycznie, ponieważ współczynniki wielomianów nad ciałem binarnym mają wartość 1, zatem maksymalna wartość wielomianu zależy od liczby jednomianów w danym równaniu. W przypadku równań opisujących algorytm szyfrowania, każde z $(T-2)n$ równań kwadratowych składa się z 5, a każde z $2n$ równań liniowych składa się z 3 jednomianów. Stąd dla tych równań wymaganych jest $2(T-1)n$ dodatkowych zmiennych binarnych, aby przedstawić ich wielokrotności k_i. W przypadku algorytmu generowania kluczy rundowych, każde z $(T-m)n$ równań liniowych składa się z 5, dla $m=2$ i $m=3$, albo 7 jednomianów, dla $m=4$. W obu tych przypadkach, liczba dodatkowych zmiennych binarnych jest równa $2(T-m)n$. Oczywiście, jeśli długość klucza jest większa niż długość bloku wejściowego, to liczba dodatkowych zmiennych binarnych, ze względu na wartości krotności k_i, dla algorytmu szyfrowania wynosi co najwyżej $4(T-1)n$.

Jak w przypadku szyfru Speck, wyznaczono małą instancję szyfru Simon, dla której przyjęto następujące parametry:

- długość bloku: $2n=8$ bitów,
- długość słowa: $n=4$ bity,



Rysunek 63: Przykład wyznaczania równań nad $GF(2)$ dla małej instancji szyfru Simon8/8, gdy bity kluczy rundowych reprezentowane są za pomocą zmiennych binarnych.

- długość klucza: $mn = 8$ bitów,
- liczba słów klucza: $m = 2$,
- sekwencja bitów dla generacji kluczy rundowych: z_0,
- liczba rund: $T = 6$,
- tekst jawny: $PT_1 = 0xf$, $PT_0 = 0x5$,
- szyfrogram: $CT_1 = 0x5$, $CT_0 = 0x7$.

Na rysunku 63 przedstawiono opis małej instancji szyfru Simon8/8, za pomocą

równań nad ciałem binarnym przy założeniu, że bity kluczy rundowych reprezentowane są za pomocą zmiennych binarnych.

6.3.4 Konstrukcja układu równań nad ciałem $GF(2)$, opisującego szyfr Simon, gdy bity kluczy rundowych reprezentowane są za pomocą wielomianów

W drugim przypadku konstruowania układu równań, opisującego szyfr Simon, bity kluczy rundowych reprezentowane są jako wielomiany, które następnie xorowane są z odpowiednim stanem w algorytmie szyfrowania. Ponieważ operacja xor realizowana jest jako dodawanie i nie zwiększa stopnia równań tylko liczbę jednomianów, dlatego podejście to jest warte rozpatrzenia. Liczba zmiennych binarnych potrzebnych do wygenerowania układu równań, opisujących cały szyfr, wynosi $mn+(T-2)n$, gdy długość klucza jest równa długości bloku wejściowego oraz $mn+2(T-2)n$, gdy długość klucza jest większa niż długość bloku wejściowego, gdzie:

- mn – liczba zmiennych binarnych dla klucza głównego oraz
- $(T-2)n$ – liczba zmiennych binarnych dla stanów pośrednich algorytmów szyfrowania, dla jednej pary tekst jawny – szyfrogram.

W tym podejściu, w równaniach (138) i (139), opisujących generowanie kolejnych kluczy rundowych, za zmienne kr_{i-1}, kr_{i-3} oraz kr_{i-m} podstawiane są reprezentujące je wielomiany, co skutkuje tym, że dla kolejnych rund wielomiany reprezentujące klucze rundowe kr_i są coraz dłuższe. Układ opisujący T-rundowy szyfr Simon$2n/mn$ składa się z równań (133), (134) oraz (135), opisujących algorytm szyfrowania, gdzie pod zmienne kr_i podstawiane są reprezentujące je wielomiany. Ponieważ bitowa operacja xor nie zwiększa stopnia równania, liczba zmiennych binarnych koniecznych do wykonania procesu linearyzacji jest taka sama jak w poprzednim podejściu. Jednak w tym podejściu liczba jednomianów w danym równaniu nie jest stała i rośnie wraz z kolejną rundą. Przykładowo, w tabeli 18 przedstawiono liczbę jednomianów każdego z n równań reprezentujących i-ty klucz rundowy szyfru Simon$32/64$. Dla każdej liczby jednomianów wyznaczono również liczbę zmiennych binarnych potrzeb-

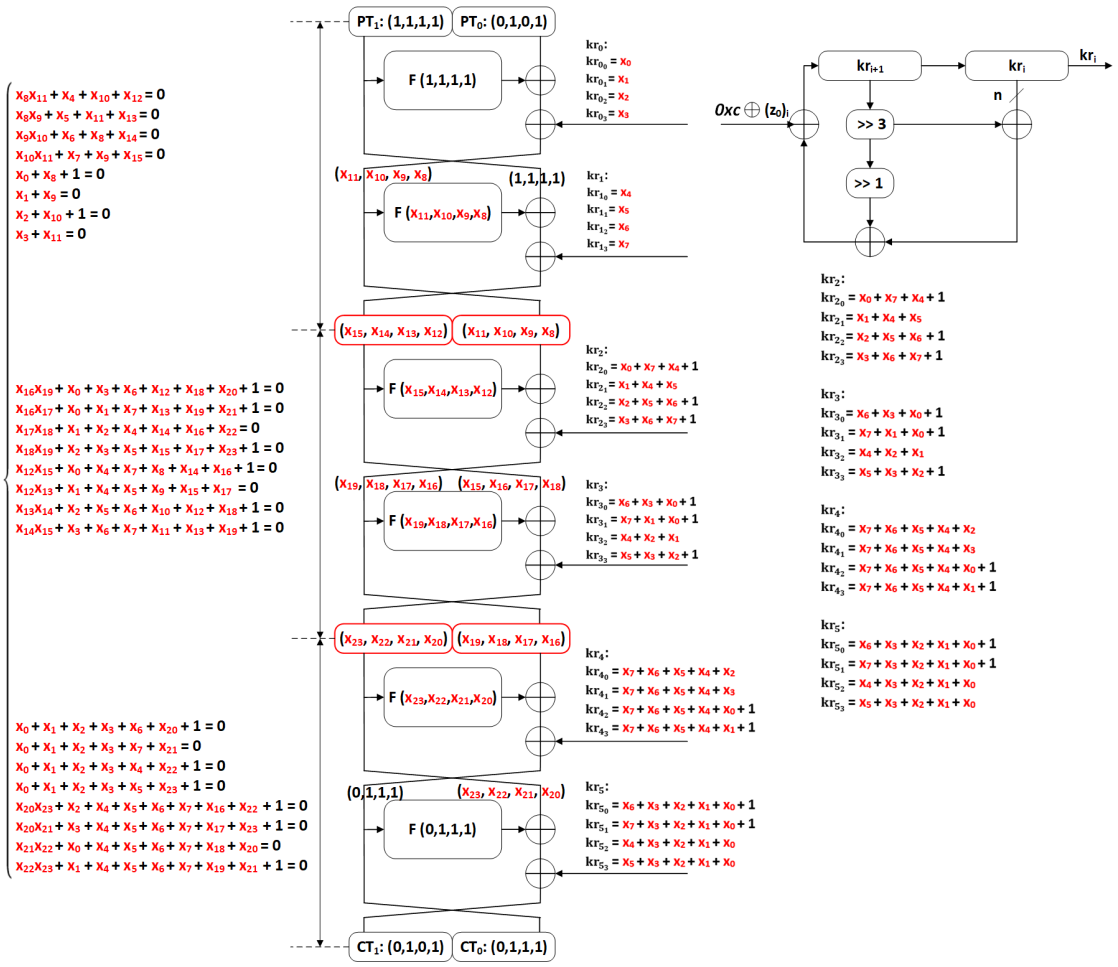
nych do przedstawienia wielokrotności k_i, wielomianów reprezentujących bity kluczy rundowych kolejnych rund. Na podstawie wartości z tabeli 18, można wyznaczyć liczbę potrzebnych zmiennych binarnych, wymaganych do przedstawienia wielokrotności k_i równań opisujących szyfr Simon32/64, w celu przekształcenia ich do problemu QUBO. Liczba ta jest równa 4 240. Oczywiście, jeśli długość klucza głównego jest większa niż długość bloku wejściowego, to opisany układ należy wygenerować dla dwóch różnych par tekst jawny – szyfrogram.

Tabela 18: Liczba jednomianów w wielomianach reprezentujących bity danego klucza rundowego szyfru Simon32/64.

Numer klucza rundowego	*Liczba jednomianów*	*Liczba zmiennych dla wartości k*	*Numer klucza rundowego*	*Liczba jednomianów*	*Liczba zmiennych dla wartości k*
0	1	0	16	29 lub 30	4
1	1	0	17	33 lub 34	5
2	1	0	18	33 lub 34	5
3	1	0	19	29 lub 30	4
4	5 lub 6	2	20	29 lub 30	4
5	9 lub 10	3	21	27 lub 28	4
6	17 lub 18	4	22	25 lub 26	4
7	19 lub 20	4	23	25 lub 26	4
8	21 lub 22	4	24	29 lub 30	4
9	23 lub 24	4	25	27 lub 28	4
10	27 lub 28	4	26	31 lub 32	4 lub 5
11	29 lub 30	4	27	29 lub 30	4
12	27 lub 28	4	28	29 lub 30	4
13	27 lub 28	4	29	27 lub 28	4
14	27 lub 28	4	30	33 lub 34	5
15	29 lub 30	4	31	29 lub 30	4

Dla tej samej instancji szyfru Simon8/8, co w poprzednim rozdziale, na rysunku 64 przedstawiono jej opis za pomocą równań nad ciałem binarnym przy założeniu, że bity kluczy rundowych reprezentowane są za pomocą odpowiednich wielomianów.

Dla każdej instancji szyfru Simon2n/mn wyznaczono rozmiar problemu QUBO, dla obu sposobów reprezentacji bitów kluczy rundowych. Otrzymane wyniki zaprezentowano w tabeli 19, gdzie dla danej instancji górny wiersz zawiera wyniki dla po-



Rysunek 64: Przykład wyznaczania równań nad $GF(2)$ dla małej instancji szyfru Simon8/8, gdy bity kluczy rundowych reprezentowane są za pomocą wielomianów.

dejścia, w którym bity kluczy rundowych reprezentowane są za pomocą zmiennych binarnych, natomiast dolny wiersz zawiera wyniki, gdy bity kluczy rundowych reprezentowane są za pomocą wielomianów.

Jak pokazują otrzymane wyniki, gdy długość klucza jest równa długości bloku wejściowego, reprezentacja bitów kluczy rundowych za pomocą wielomianów jest podejściem efektywniejszym, ponieważ rozmiar otrzymanego problemu QUBO jest mniejszy niż dla podejścia, w którym bity kluczy rundowych reprezentowane są za pomocą osobnych zmiennych binarnych. Dla pozostałych wariantów, gdy klucz główny jest dłuższy niż długość bloku wejściowego, podczas opisywania szyfru Simon za pomocą równań wielomianowych bity kluczy rundowych powinny zostać zdefiniowane jako zmienne binarne, ponieważ tylko algorytm szyfrowania opisywany jest na podsta-

wie dwóch różnych par tekst jawny – szyfrogram, natomiast układ opisujący algorytm generowania kluczy rundowych jest taki sam dla obu par.

Tabela 19: Rozmiar problemu QUBO dla wariantów szyfru Simon2n/mn, opisanego za pomocą równań nad $GF(2)$.

Wariant szyfru Simon2n/mn	*Liczba rund*	*Liczba par*	*Liczba równań*	*Liczba zmiennych binarnych*			
				początkowy układ	*linearyzacja*	*wartości k*	*problem QUBO*
Simon32/64	32	2	1 472	1 472	960	2 880	5 312
			1 024	1 024	960	4 240	6 224
Simon48/72	36	2	2 520	2 496	1 632	4 944	9 072
			1 728	1 704	1 632	6 538	9 874
Simon48/96	36	2	2 496	2 496	1 632	4 896	9 024
			1 728	1 728	1 632	7 680	11 040
Simon64/96	42	2	3 936	3 904	2 560	7 744	14 208
			2 688	2 656	2 560	11 100	16 316
Simon64/128	44	2	4 096	4 096	2 688	8 064	14 848
			2 816	2 816	2 688	13 308	18 812
Simon96/96	52	1	4 896	4 896	2 400	9 696	16 992
			2 496	2 496	2 400	9 966	14 862
Simon96/144	54	2	7 632	7 584	4 992	15 072	27 648
			5 184	5 136	4 992	22 998	33 126
Simon128/128	68	1	8 576	8 576	4 224	17 024	29 824
			4 352	4 352	4 224	18 694	27 270
Simon128/256	72	2	13 568	13 568	8 960	26 880	49 408
			9 216	9 216	8 960	51 128	69 304

Analiza przedstawienia szyfru Simon za pomocą układu równań wielomianowych nad ciałem binarnym oraz jego transformacja do problemu optymalizacyjnego w postaci QUBO została zaprezentowana na konferencji CECC22 (*22nd Central European Conference on Cryptography*).

6.4 Atak algebraiczny z wykorzystaniem wyżarzania kwantowego na szyfr Simon

Ponownie, zakładając że złożoność rozwiązania problemu QUBO, składającego się z N zmiennych binarnych, wynosi $O\left(e^{\sqrt{N}}\right)$ ze współczynnikiem równym 1, oszacowano liczbę rund dla której proponowany atak jest lepszy od ataku pełnego przeszukiwania. Otrzymane wyniki, dla każdego wariantu szyfru Simon2n/mn, przedstawiono w tabeli 20. Niestety, dla żadnej wersji nie udało się uzyskać wyniku lepszego niż naj-

lepszy atak klasyczny, które przedstawiono w pracach [15] oraz [17]. Najlepszy wynik udało się uzyskać dla wersji Simon128/256, dla którego możliwe jest zaatakowanie ponad połowy rund.

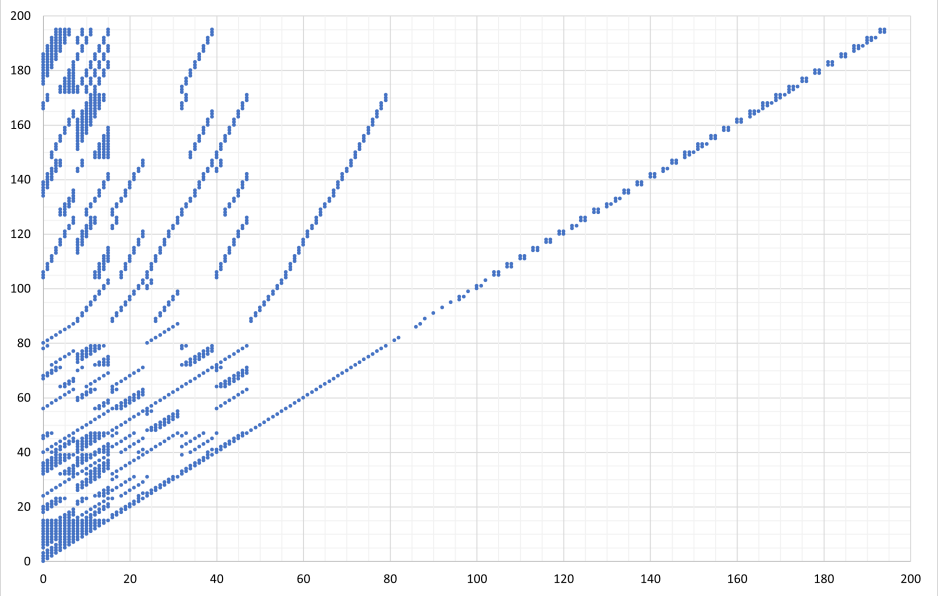
Tabela 20: Liczba zaatakowanych rund dla najlepszego ataku klasycznego oraz proponowanego ataku, dla poszczególnych wariantów szyfru Simon2n/mn.

Wariant szyfru Simon2n/mn	*Liczba rund szyfru*	*Liczba zaatakowanych rund dla najlepszego ataku klasycznego*	*Liczba zaatakowanych rund dla proponowanego ataku*
Simon32/64	32	24 (75%)	13 (41%)
Simon48/72	36	24 (67%)	11 (31%)
Simon48/96	36	25 (69%)	18 (50%)
Simon64/96	42	30 (71%)	14 (33%)
Simon64/128	44	31 (70%)	24 (55%)
Simon96/96	52	37 (71%)	16 (31%)
Simon96/144	54	38 (70%)	20 (37%)
Simon128/128	68	49 (72%)	22 (32%)
Simon128/256	72	53 (74%)	46 (64%)

W celu otrzymania problemu QUBO o rozmiarze około 200 zmiennych binarnych, zaprojektowano mniejszą instancję szyfru Simon2n/mn, o następujących parametrach:

- długość bloku: $2n = 16$ bitów,
- długość słowa: $n = 8$ bitów,
- długość klucza: $mn = 16$ bitów,
- liczba słów klucza: $m = 2$,
- sekwencja bitów dla generacji kluczy rundowych: z_0,
- liczba rund: $T = 6$.

Ustalono wartość współczynnika wielomianu kary równą 6. W wyniku transformacji układu równań wielomianowych opisujących szyfr Simon16/16, o powyższych parametrach, otrzymano problem QUBO składający się ze 196 zmiennych binarnych. Struktura macierzy Q wygenerowanego problemu została przedstawiona na rysunku



Rysunek 65: Struktura macierzy Q problemu QUBO wygenerowanego dla sześciorundowego szyfru Simon16/16, opisanego nad $GF(2)$.

65. Wartości bezwzględne współczynników otrzymanej macierzy Q mieszczą się w zakresie od 2 do 29, które po skalowaniu w wyżarzaczu kwantowym D-Wave przyjęły wartości od $0,069$ do 1. Liczba niezerowych elementów w powyższej górno-trójkątnej macierzy Q wynosi 1 684 z 19 306 możliwych elementów, co stanowi $8,72\%$. Macierz ta jest rzadsza niż macierz Q szyfru Speck16/16, przedstawiona na rysunku 56 oraz niewiele gęstsza niż macierz Q jednorundowego szyfru S-AES, przedstawiona na rysunku 45. Czas wyżarzania problemu QUBO szyfru Simon16/16 ustalono na 20 minut, po którym otrzymano rozwiązanie, zawierające poprawny klucz.

7 PODSUMOWANIE

Głównym celem niniejszej rozprawy było opracowanie ataku, polegającego na przekształceniu układu równań wielomianowych, opisującego szyfr blokowy, do problemu optymalizacyjnego w formie QUBO (binarnego zadania najmniejszych kwadratów bez ograniczeń), możliwego do uruchomienia na komputerze kwantowym D-Wave. W proponowanym ataku wykorzystano ideę ataków algebraicznych, jednak zamiast konstruować układy nadokreślone, skupiono się na znalezieniu układu dającego możliwie jak najmniejszy docelowy problem QUBO. W tym celu przeanalizowano zagadnienie procesu wyżarzania kwantowego, jego implementację oraz proces rozwiązywania problemów optymalizacyjnych za pomocą komputera kwantowego firmy D-Wave. Zwrócono przede wszystkim uwagę na moment osadzania problemu w fizycznej strukturze obliczeniowej jednostki kwantowej, podczas którego skalowane są wartości macierzy, reprezentującej rozwiązywany problem optymalizacyjny oraz tworzone są łańcuchy fizycznych kubitów.

Następnie dostosowano model transformacji układu równań wielomianowych o wielu zmiennych, opisujący dowolny szyfr blokowy, do problemu optymalizacyjnego w postaci QUBO. Poszczególne kroki modelu transformacji znane były wcześniej, jednak w niniejszej rozprawie dobrano takie metody, które zapewniają, że globalne rozwiązanie problemu optymalizacyjnego QUBO jest rozwiązaniem układu opisującego szyfr oraz w jak najmniejszym stopniu zwiększają rozmiar docelowego problemu optymalizacyjnego. Pokazano również, że proponowana metoda transformacji umożliwia jednoznaczne odzyskanie klucza głównego.

Wynikiem powyższych etapów pracy było zdefiniowanie postaci efektywnego układu równań opisującego szyfr blokowy, jaka umożliwia uzyskanie jak najmniejszego problemu optymalizacyjnego. Definicja ta uwzględnia metody zastosowane w modelu transformacji oraz fizyczne ograniczenia zastosowanego narzędzia, rozwiązującego problemy optymalizacyjne.

Główną pracą badawczą, stanowiącą podstawę niniejszej rozprawy, było znalezienie układu efektywnego dla wybranych szyfrów blokowych. Do badań wybrano

trzy najpowszechniejsze struktury szyfrów blokowych: SPN, ARX oraz Feistela.

Pierwszym analizowanym szyfrem blokowym był szyfr typu SPN, na przykładzie obecnego standardu kryptografii symetrycznej – szyfru AES. Podczas analizy jego budowy skupiono się na skrzynce podstawieniowej, która zapewnia odporność na większość znanych ataków. Dlatego zdefiniowano pojęcie układu równań, jednoznacznie wyznaczającego skrzynkę podstawieniową. Następnie, zakładając że zbiór wszystkich tych układów jest przestrzenią rozwiązań, zdefiniowano problem optymalizacyjny, w którym określono funkcję celu wyznaczającą układ efektywny, w kontekście jego transformacji do problemu QUBO. W ramach poszukiwań układu efektywnego dla skrzynki podstawieniowej, opracowano cztery metody przeszukiwania przestrzeni rozwiązań. W wyniku poszukiwań udało się znaleźć układ efektywny, jednoznacznie wyznaczający skrzynkę podstawieniową, który pozwolił zmniejszyć o ok. 58% rozmiar docelowego problemu QUBO. Uzyskane wyniki są również o ok. 70% lepsze, ze względu na rozmiar problemu optymalizacyjnego, niż dotychczas opublikowane wyniki, przedstawione w pracy [54].

Kolejnym przeanalizowanym szyfrem blokowym był szyfr o strukturze ARX, dla której jako przedstawiciela wybrano szyfr Speck. Szyfr ten opisano nad dwoma strukturami algebraicznymi, nad pierścieniem $\mathbb{Z}_{2^n}$ oraz nad ciałem binarnym $GF(2)$. W szyfrze tym skupiono się na operacji dodawania modularnego, dlatego, aby znaleźć efektywny układ równań nad $\mathbb{Z}_{2^n}$ opisujący szyfr Speck$2n/mn$, przesunięto zakres rundy, zmieniając tym samym kolejność wykonywanych operacji. Umożliwiło to uzyskanie problemów optymalizacyjnych w postaci QUBO dla poszczególnych wersji szyfru o kilkanaście rzędów wielkości mniejszych, niż gdyby zachowano zakres rundy zgodny z dokumentacją tego szyfru. Dla wersji szyfru Speck$2n/mn$, z kluczem takiej samej długości jak dla wersji szyfru AES, otrzymane rozmiary problemów optymalizacyjnych dla szyfru Speck są dwa razy mniejsze, niż dla szyfru AES. Szyfr Speck$2n/mn$ przedstawiono również za pomocą układu równań nad ciałem binarnym, jednak rozmiary otrzymanych problemów optymalizacyjnych w postaci QUBO są ok. $6 \cdot 10^{16}$ razy większe, niż dla szyfru Speck opisanego nad $\mathbb{Z}_{2^n}$ oraz ok. $3 \cdot 10^{16}$ razy większe, niż dla szyfru AES.

Ostatnim przeanalizowanym szyfrem był szyfr o strukturze Feistela, którego przykładem jest szyfr Simon. Aby otrzymać efektywny układ równań, opisujący ten szyfr, wykorzystano jego strukturę Feistela, w której w każdej rundzie przetwarzana jest tylko połowa bloku wejściowego. Dlatego sąsiadujące ze sobą rundy połączono w pary, tworząc jedną podwójną rundę. Podczas analizy szyfru Simon skupiono się na reprezentacji bitów kluczy rundowych, które przedstawiono na dwa sposoby: jako zmienne binarne oraz jako wielomiany nad ciałem binarnym. Przedstawienie bitów kluczy rundowych jako wielomianów zmniejsza liczbę równań w układzie, ale zwiększa liczbę wszystkich jednomianów w równaniach układu. Powoduje to, że dla wersji szyfru Simon$2n/mn$ o długości klucza głównego równej długości bloku wejściowego, efektywniejszym podejściem jest przedstawienie bitów kluczy rundowych jako wielomianów, co pociąga za sobą otrzymanie o 10% mniejszego problemu optymalizacyjnego. Dzięki temu, dla wersji szyfru Simon$128/128$ rozmiar otrzymanego problemu QUBO jest prawie równy rozmiarowi problemu dla szyfru AES128 i o 30% większy niż dla szyfru Speck. W przypadku wersji szyfru Simon z kluczem długości 256 bitów, otrzymany problem QUBO jest o ok. 30% mniejszy niż problem QUBO dla szyfru AES256 i o ok. 30% większy niż problem QUBO dla szyfru Speck$128/256$.

Pomimo że złożoność obliczeniowa rozwiązania problemu QUBO za pomocą wyżarzania kwantowego nie została jeszcze w pełni zbadana, istnieją pewne przypuszczenia, że rozwiązanie problemu QUBO z N zmiennymi binarnymi wymaga $O\left(e^{\sqrt{N}}\right)$ operacji elementarnych. Zakładając prawdziwość tej złożoności ze współczynnikiem równym 1, najlepszy rezultat ataku algebraicznego wykorzystującego wyżarzanie kwantowe osiągnięto dla szyfru Speck$128/256$, dla którego atak ten może być skuteczny na 32 z 34 rund, co jest lepszym wynikiem niż najlepszy atak klasyczny. Dla wszystkich pozostałych wersji analizowanych szyfrów, za pomocą proponowanego ataku osiągnięto wyniki gorsze niż dla ataków klasycznych. Mimo to, warto zauważyć, że algorytmy blokowe są znacznie łatwiejsze do złamania za pomocą ataku wykorzystującego wyżarzanie kwantowe niż problem faktoryzacji, czy logarytmu dyskretnego, o podobnym poziomie bezpieczeństwa.

Dla każdego z analizowanych typów szyfrów blokowych, opracowano taką jego

małą instancję, aby docelowy problem QUBO składał się z około 200 zmiennych binarnych. Ustalono również czas przeprowadzenia procesu wyżarzania na 20 minut, jednakowy dla wszystkich wyznaczonych problemów małych instancji. Przyjmując te parametry jako parametry odniesienia, można przeanalizować wpływ struktury macierzy problemu optymalizacyjnego, przesyłanego do komputera D-Wave, na możliwość uzyskania prawidłowego rozwiązania. Problemy QUBO, dla których przeprowadzono zaproponowany atak i odzyskano prawidłowy klucz główny, zostały wyznaczone dla małych instancji analizowanych szyfrów, opisanych nad ciałem binarnym:

- jednorundowego szyfru S-AES, dla którego postawiono problem QUBO ze 199 zmiennymi binarnymi, a liczba jego niezerowych współczynników jednomianów stanowi $7,4\%$ całej macierzy,
- trzyrundowego szyfru Speck$8/8$, dla którego postawiono problem QUBO ze 174 zmiennymi binarnymi, a liczba jego niezerowych współczynników jednomianów stanowi $6,7\%$ całej macierzy,
- sześciorundowego szyfru Simon$16/16$, dla którego postawiono problem QUBO ze 196 zmiennymi binarnymi, a liczba jego niezerowych współczynników jednomianów stanowi $8,72\%$ całej macierzy.

Natomiast dla problemu QUBO, wyznaczonego dla szyfru Speck$16/16$, przedstawionego nad $\mathbb{Z}_{2^n}$, dla którego problem składa się z 170 zmiennych binarnych i którego liczba niezerowych współczynników jednomianów stanowi 34% całej macierzy, nie odzyskano prawidłowego klucza głównego. Dlatego, pomimo że problemy QUBO otrzymane dla szyfru Speck, opisanego nad $\mathbb{Z}_{2^n}$, są znacznie mniejsze, niż problemy QUBO otrzymane dla standardu AES o tej samej wielkości bloku i długości klucza, ze względu na strukturę macierzy Q tych problemów wydaje się, że ataki algebraiczne na szyfr Speck z wykorzystaniem wyżarzania kwantowego nie muszą być skuteczniejsze, niż w przypadku szyfru AES.

Głównym wkładem niniejszej rozprawy jest:

- zdefiniowanie układu jednoznacznie wyznaczającego skrzynkę podstawieniową

oraz zaproponowanie metody poszukiwania dla niej układu efektywnego, w kontekście dalszych przekształceń do problemu QUBO;

- określenie zakresu wartości wagi kary dla linearyzacji, w kontekście zastosowanej metody transformacji układu równań do problemu optymalizacyjnego w postaci QUBO i zastosowanego komputera kwantowego D-Wave do rozwiązania tego problemu oraz pokazanie, że problem linearyzacji układów równań opisujących przeanalizowane szyfry blokowe nie jest problemem NP-trudnym;
- zdefiniowanie wymagań dla układów równań opisujących szyfry blokowe oraz pokazanie sposobu konstruowania układów równań wielomianowych o wielu zmiennych opisujących szyfry blokowe o najpopularniejszych typach struktur tak, aby docelowy problem w postaci QUBO był jak najmniejszy;
- uzyskanie rozmiarów problemów QUBO dla standardu AES znacznie mniejszych niż opublikowanych do tej pory oraz znacznie mniejszych niż dla problemów kryptografii asymetrycznej.

Dalsze prace powinny skupiać się na badaniu złożoności obliczeniowej ataków algebraicznych na algorytmy blokowe, z wykorzystaniem wyżarzania kwantowego, która wydaje się zależeć nie tylko od liczby zmiennych w rozwiązywanym problemie optymalizacyjnym. Należy również zastanowić się nad zastosowaniem przedstawionej metody do innego typu prymitywów kryptografii symetrycznej, takich jak szyfry strumieniowe oraz funkcje skrótu. Dodatkowo można również zastanowić się nad zastosowaniem przedstawionej metody transformacji do innych znanych ataków na szyfry blokowe, jak na przykład do szukania różniczek w kryptoanalizie różnicowej. Dzisiaj wydaje się, że problemy takich rozmiarów jak problem QUBO dla standardu AES128, nie zostaną rozwiązane za pomocą wyżarzania kwantowego w najbliższych latach.

LITERATURA

[1] D-wave systems. Dokumentacja systemu D-Wave dostępna w Internecie: https://www.dwavesys.com, dostęp: 2022-06-10.

[2] G. V. Bard, N. T. Courtois, C. Jefferson. Efficient methods for conversion and solution of sparse systems of low-degree multivariate polynomials over gf(2) via sat-solvers. *Cryptology ePrint Archive, Paper 2007/024*, 2007. https://eprint.iacr.org/2007/024.

[3] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, L. Wingers. The simon and speck families of lightweight block ciphers. *Technical report, Cryptology ePrint Archive, Report./404*, 2013.

[4] D. J. Bernstein. *Introduction to post-quantum cryptography*, strony 1–14. Springer, 2009. doi: 10.1007/978-3-540-88702-7_1.

[5] A. Biryukov, D. Khovratovich. Related-key cryptanalysis of the full aes-192 and aes-256. *Advances in Cryptology – ASIACRYPT 2009. Lecture Notes in Computer Science*, wolumen 5912, strony 1–18. Springer, 2009. doi: 10.1007/978-3-642-10366-7_1.

[6] A. Biryukov, D. Khovratovich, I. Nikolić. Distinguisher and related-key attack on the full aes-256. *Advances in Cryptology - CRYPTO 2009. Lecture Notes in Computer Science*, wolumen 5677, strony 231–249. Springer, 2009. doi: 10.1007/978-3-642-03356-8_14.

[7] A. Bogdanov, D. Khovratovich, C. Rechberger. Biclique cryptanalysis of the full aes. *Advances in Cryptology – ASIACRYPT 2011. Lecture Notes in Computer Science*, wolumen 7073, strony 344–371. Springer, 2011. doi: 10.1007/978-3-642-25385-0_19.

[8] A. Borle, S. J. Lomonaco. Analyzing the quantum annealing approach for solving linear least squares problems. *International Workshop on Algorithms and Computation*, strony 289–301. Springer, 2019.

[9] M. Born, V. Fock. Beweis des adiabatensatzes. *Zeitschrift für Physik*, 51(3):165–180, 1928. doi: 10.1007/BF01343193.

[10] E. Boros, P. L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123(1-3):155–225, 2002. doi: 10.1016/S0166-218X(01)00341-9.

[11] E. Burek, M. Wroński. Quantum annealing and algebraic attack on speck cipher. *Computational Science – ICCS 2022. Lecture Notes in Computer Science*, wolumen 13353, strony 143–149. Springer, 2022. doi: 10.1007/978-3-031-08760-8_12.

[12] E. Burek, M. Wroński, K. Mańk, M. Misztal. Algebraic attacks on block ciphers using quantum annealing. *IEEE Transactions on Emerging Topics in Computing*, 10(2):678–689, 2022. doi: 10.1109/TETC.2022.3143152.

[13] C. C. Chang, A. Gambhir, T. S. Humble, S. Sota. Quantum annealing for systems of polynomial equations. *Scientific reports*, 9(1):1–9, 2019. doi: 10.1038/s41598-019-46729-0.

[14] T. H. Chang, T. C. Lux, S. S. Tipirneni. Least-squares solutions to polynomial systems of equations with quantum annealing. *Quantum Information Processing*, 18(12):1–17, 2019. doi: 10.1007/s11128-019-2489-x.

[15] H. Chen, X. Wang. Improved linear hull attack on round-reduced simon with dynamic key-guessing techniques. *Fast Software Encryption. Lecture Notes in Computer Science*, wolumen 9783, strony 428–449. Springer Berlin Heidelberg, 2016. doi: 10.1007/978-3-662-52993-5_22.

[16] Y.-A. Chen, X.-S. Gao. Quantum algorithm for boolean equation solving and quantum algebraic attack on cryptosystems. *Journal of Systems Science and Complexity*, 35(1):373–412, 2022. doi: 10.1007/s11424-020-0028-6.

[17] Z. Chu, H. Chen, X. Wang, X. Dong, L. Li. Improved integral attacks on simon32 and simon48 with dynamic key-guessing techniques. *Security and Communication Networks*, 2018, 2018. doi: 10.1155/2018/5160237.

[18] N. Courtois, A. Klimov, J. Patarin, A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. *Advances in Cryptology — EUROCRYPT 2000. Lecture Notes in Computer Science*, wolumen 1807, strony 392–407. Springer, 2000. doi: 10.1007/3-540-45539-6_27.

[19] N. Courtois, J. Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. *Advances in Cryptology — ASIACRYPT 2002. Lecture Notes in Computer Science*, wolumen 2501, strony 267–287. Springer, 2002. doi: 10.1007/3-540-36178-2_17.

[20] N. T. Courtois, G. V. Bard. Algebraic cryptanalysis of the data encryption standard. *Cryptography and Coding 2007. Lecture Notes in Computer Science*, wolumen 4887, strony 152–169. Springer, 2007. doi: 10.1007/978-3-540-77272-9_10.

[21] J. Daemen, V. Rijmen. Aes proposal: Rijndael. 1999.

[22] J. Daemen, V. Rijmen. *The design of Rijndael*, wolumen 2. Springer Berlin, Heidelberg, 2002. doi: 10.1007/978-3-662-04722-4.

[23] D. Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, 1985. doi: 10.1098/rspa.1985.0070.

[24] I. Dinur. Improved differential cryptanalysis of round-reduced speck. *Selected Areas in Cryptography – SAC 2014. Lecture Notes in Computer Science*, wolumen 8781, strony 147–164. Springer International Publishing, 2014. doi: 10.1007/978-3-319-13051-4_9.

[25] R. Dridi, H. Alghassi. Prime factorization using quantum annealing and computational algebraic geometry. *Scientific reports*, 7(1):1–10, 2017. doi: 10.1038/srep43048.

[26] A. D. Dwivedi, P. Morawiecki, G. Srivastava. Differential cryptanalysis of round-

reduced speck suitable for internet of things devices. *IEEE Access*, 7:16476–16486, 2019. doi: 10.1109/ACCESS.2019.2894337.

[27] S. Edelkamp, S. Schrodl. *Heuristic search: theory and applications*. Elsevier, 2011.

[28] J. C. Faugère. A new efficient algorithm for computing gröbner bases (f4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, 1999. doi: 10.1016/S0022-4049(99)00005-5.

[29] J. C. Faugère. A new efficient algorithm for computing gröbner bases without reduction to zero (f5). *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, ISSAC '02, strony 75–83. Association for Computing Machinery, 2002. doi: 10.1145/780506.780516.

[30] R. P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6/7):467–488, 1982. doi: 10.1007/BF02650179.

[31] A. B. Finnila, M. Gomez, C. Sebenik, C. Stenson, J. D. Doll. Quantum annealing: A new method for minimizing multidimensional functions. *Chemical Physics Letters*, 219(5-6):343–348, 1994. doi: 10.1016/0009-2614(94)00117-0.

[32] P. FIPS. Fips 197: Specification for the advanced encryption standard (aes). *Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD*, strony 20899–8900, 2001. doi: 10.6028/NIST.FIPS.197.

[33] J. Gao, H. Li, B. Wang, X. Li. Quantum security of aes-128 under hhl algorithm. *Quantum Information and Computation*, 22(3&4):0209–0240, 2022.

[34] F. Glover, G. Kochenberger, Y. Du. A tutorial on formulating and using qubo models. *arXiv preprint arXiv:1811.11538*, 2018. doi: 10.48550/arXiv.1811.11538.

[35] T. Greene. Google reclaims quantum computer crown with 72 qubit processor. *The Next Web [online], https://tnw. to/2FmfU14*, 2018. dostęp: 2022-04-06.

[36] L. K. Grover. A fast quantum mechanical algorithm for database search. *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, strony 212–219, 1996. doi: 10.1145/237814.237866.

[37] P. L. Hammer, S. Rudeanu. Pseudo-boolean programming. *Operations Research*, 17(2):233–261, 1969. doi: 10.1287/opre.17.2.233.

[38] A. W. Harrow, A. Hassidim, S. Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009. doi: 10.1103/PhysRevLett.103.150502.

[39] M. Hirvensalo. *Algorytmy kwantowe*. Wydawnictwa Szkolne i Pedagogiczne, 2004.

[40] S. Istrail. Statistical mechanics, three-dimensionality and np-completeness: I. universality of intracatability for the partition function of the ising model across non-planar surfaces. *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, STOC '00, strony 87–96. Association for Computing Machinery, 2000. doi: 10.1145/335305.335316.

[41] S. Jiang, K. A. Britt, A. J. McCaskey, T. S. Humble, S. Kais. Quantum annealing for prime factorization. *Scientific reports*, 8(1):1–9, 2018. doi: 10.1038/s41598-018-36058-z.

[42] M. W. Johnson, M. H. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, i in. Quantum annealing with manufactured spins. *Nature*, 473(7346):194–198, 2011. doi: 10.1038/nature10012.

[43] T. Kadowaki, H. Nishimori. Quantum annealing in the transverse ising model. *Physical Review E*, 58(5):5355–5363, 1998. doi: 10.1103/PhysRevE.58.5355.

[44] M. Kaplan, G. Leurent, A. Leverrier, M. Naya-Plasencia. Quantum differential and linear cryptanalysis. *IACR Transactions on Symmetric Cryptology*, 2016(1):71—-94, 2016. doi: 10.13154/tosc.v2016.i1.71-94.

[45] A. Kipnis, A. Shamir. Cryptanalysis of the hfe public key cryptosystem by relinearization. *Advances in Cryptology — CRYPTO' 99. Lecture Notes in Computer Science*, wolumen 1666, strony 19–30. Springer, 1999. doi: 10.1007/3-540-48405-1_2.

[46] H. Lee, S. Kim, H. Kang, D. Hong, J. Sung, S. Hong. Calculating the approximate probability of differentials for arx-based cipher using sat solver. *Journal of the Korea Institute of Information Security & Cryptology*, 28(1):15–24, 2018.

[47] H. Li, L. Yang. Quantum differential cryptanalysis to the block ciphers. *Applications and Techniques in Information Security*, strony 44–51. Springer, 2015. doi: 10.1007/978-3-662-48683-2_5.

[48] J. Lu, O. Dunkelman, N. Keller, J. Kim. New impossible differential attacks on aes. *Progress in Cryptology - INDOCRYPT 2008. Lecture Notes in Computer Science*, wolumen 5365, strony 279–293. Springer, 2008. doi: 10.1007/978-3-540-89754-5_22.

[49] C. C. McGeoch. *Adiabatic quantum computation and quantum annealing: Theory and practice*, wolumen 5, strony 1–93. Morgan & Claypool Publishers, 2014. doi: 10.1007/978-3-031-02518-1.

[50] Z. Michalewicz, D. B. Fogel. *How to solve it: modern heuristics*. Springer Berlin, Heidelberg, 2013. doi: 10.1007/978-3-662-07807-5.

[51] S. Mukherjee, B. K. Chakrabarti. Multivariable optimization: Quantum annealing and computation. *The European Physical Journal Special Topics*, 224(1):17–24, 2015. doi: 10.1140/epjst/e2015-02339-y.

[52] S. Murphy, M. J. Robshaw. Essential algebraic structure within the aes. *Advances in Cryptology — CRYPTO 2002. Lecture Notes in Computer Science*, wolumen 2442, strony 1–16. Springer, 2002. doi: 10.1007/3-540-45708-9_1.

[53] D. O'Malley, V. V. Vesselinov. Toq.jl: A high-level programming language for d-wave machines based on julia. *2016 IEEE High Performance Extreme*

Computing Conference (HPEC), strony 1–7. IEEE, 2016. doi: 10.1109/HPEC.2016.7761616.

[54] A. I. Pakhomchik, V. V. Voloshinov, V. M. Vinokur, G. B. Lesovik. Converting of boolean expression to linear equations, inequalities and qubo penalties for cryptanalysis. *Algorithms*, 15(2):33, 2022. doi: 10.3390/a15020033.

[55] I. G. Rosenberg. Reduction of bivalent maximization to the quadratic case. *Cahiers du Centre d'Etudes de Recherche Operationnelle*, 17:71–74, 1975.

[56] T. Santoli, C. Schaffner. Using simon's algorithm to attack symmetric-key cryptographic primitives. *Quantum Information & Computation*, 17(1–2):65–78, 2017. doi: 10.5555/3179483.3179487.

[57] W. Scherer. *Mathematics of quantum computing*. Springer, 2019.

[58] C. E. Shannon. Communication theory of secrecy systems. *The Bell system technical journal*, 28(4):656–715, 1949. doi: 10.1002/j.1538-7305.1949.tb00928.x.

[59] L. Song, Z. Huang, Q. Yang. Automatic differential analysis of arx block ciphers with application to speck and lea. *Information Security and Privacy ACISP 2016. Lecture Notes in Computer Science*, wolumen 9723, strony 379–394. Springer International Publishing, 2016. doi: 10.1007/978-3-319-40367-0_24.

[60] W. Stallings, L. Brown, M. D. Bauer, M. Howard. *Computer security: principles and practice*, wolumen 2. Pearson Upper Saddle River, 2012.

[61] B. Wang, F. Hu, H. Yao, C. Wang. Prime factorization algorithm based on parameter optimization of ising model. *Scientific reports*, 10(1):1–10, 2020. doi: 10.1038/s41598-020-62802-5.

[62] M. Wroński. Practical solving of discrete logarithm problem over prime fields using quantum annealing. *Computational Science – ICCS 2022. Lecture Notes in Computer Science*, wolumen 13353, strony 93–106. Springer, 2022. doi: 10.1007/978-3-031-08760-8_8.

[63] H. Xie, L. Yang. Quantum impossible differential and truncated differential cryptanalysis. *ArXiv*, abs/1712.06997, 2017. doi: 10.48550/arXiv.1712.06997.

[64] H. Xie, L. Yang. Using bernstein–vazirani algorithm to attack block ciphers. *Designs, Codes and Cryptography*, 87(5):1161–1182, 2019. doi: 10.1007/s10623-018-0510-5.

[65] H. Xie, L. Yang. A quantum related-key attack based on the bernstein–vazirani algorithm. *Quantum Information Processing*, 19(8):1–20, 2020. doi: 10.1007/s11128-020-02741-2.

[66] Q. Zhou, S. Lu, Z. Zhang, J. Sun. Quantum differential cryptanalysis. *Quantum Information Processing*, 14(6):2101–2109, 2015. doi: 10.1007/s11128-015-0983-3.