WOJSKOWA AKADEMIA TECHNICZNA
im. Jarosława Dąbrowskiego

ROZPRAWA DOKTORSKA

Synteza i optymalizacja kwantowych układów logicznych reprezentujących prymitywy kryptograficzne

Autor
mgr inż. Adam Jagielski

Promotor
dr hab. inż. Andrzej Paszkiewicz

Promotor Pomocniczy
dr inż. Krzysztof Kanciak

Warszawa 2025

Promotorom, Nauczycielom i Wszystkim, którzy mi pomogli

– za pomoc.

Rodzicom, Przyjaciołom i Bliskim

– za wsparcie.

Hektolitrom kawy i słuchawkom wyciszającym

– za wszystko pozostałe.

Nie panikuj!

Douglas Addams, 1979, *"Autostopem przez Galaktykę"*

Spis treści

Oznaczenia

`AD`	Stopień algebraiczny *(z j. ang. algebraic degree)*
`AES`	Zaaw. standard szyfrowania *(z j. ang. Adv. Encryption Standard)*
`ANF`	Algebraiczna postać normalna *(z j. ang. algebraic normal form)*
`ARX`	Zestaw operacji *(z j. ang. Add, Rotate, XOR)*
$\mathbb{B}$	Algebra Boole'a
`BFS`	Algorytm przeszukiwania grafu wszerz *(z j. ang. breadth first search)*
$\mathbb{C}$	Ciało liczb zespolonych
`CNF`	Koniunkcyjna postać normalna *(z j. ang. conjunctive normal form)*
`DDT`	Tabela rozkładu różniczek *(z j. ang. differential distribution table)*
`DFS`	Algorytm przeszukiwania grafu w głąb *(z j. ang. depth first search)*
`HD`	Odległość Hamminga
`HW`	Waga Hamminga
`IoT`	Internet rzeczy *(z j. ang. internet of things)*
`LAT`	Tabela przybliżeń liniowych *(z j. ang. linear approximations table)*
`LWC`	Kryptografia lekka *(z j. ang. lightweight cryptography)*
`MCT`	Zestaw bramek *(z j. ang. multiple-control Toffoli)*
`NIST`	*National Institute of Standards and Technology*
`NCT`	Zestaw bramek `NOT`, `CNOT`, `Toffoli`
`ND`	Stopień nieliniowośc *(z j. ang. nonlinearity degree)*
`SAC`	Ścisłe Kryterium Lawinowe *(z j. ang. strict avalanche criterion)*
`SAT`	Spełnialność, problem spełnialności *(z j. ang. satisfiability)*
`SAT-solver`	Program rozwiązujący problem spełnialności

Wprowadzenie

Rosnąca potrzeba miniaturyzacji urządzeń i tworzenia silnie rozproszonych systemów stawia wiele wyzwań w zakresie bezpieczeństwa informacyjnego. Wszędzie tam, gdzie urządzenie ma ograniczone zasoby, takie jak moc obliczeniowa czy żywotność baterii, kluczowe jest uwzględnienie kosztów i złożoności zastosowanych środków bezpieczeństwa, czasem nawet kosztem ich długoterminowej niezawodności.

Środowiska o ograniczonych zasobach, takie jak Internet Rzeczy *(z j. ang. IoT – Internet of Things)* stoją przed trudnym problemem odnalezienia konsensusu pomiędzy kosztami rozwiązań a ich bezpieczeństwem. Problem ten stał się szczególnie naglący w widmie nowego zagrożenia, jakim stał się komputer kwantowy opisany po raz pierwszy w pracach Tomasso Toffoliego *"Reversible Computing"* (1980) [1], Richarda Feynmana: *"Simulating Physics with Computers"* (1982) [2], *"Quantum Mechanical Computers"* (1986) [3] oraz serii jego wykładów *"Lectures on Computation"* [4].

Problem ten został spostrzeżony przez instytucje wiodące w zakresie światowego bezpieczeństwa informacyjnego. W roku 2015 Narodowy Instytut Standaryzacji i Technologii w USA *(z j. ang. NIST – National Institute of Standards and Technology)* przeprowadził pierwsze warsztaty dotyczące tematu standaryzacji kryptografii lekkiej [5] *(z j. ang. LWC – lightweight cryptography)*, której celem jest zapewnienie bezpieczeństwa w środowiskach o ograniczonych zasobach.

Dwa lata później, w marcu 2017, NIST opublikował raport [6], który przedstawił przegląd stanu wiedzy na temat kryptografii lekkiej, a także plany NIST na przebieg procesu standaryzacji rozwiązań z tej kategorii. W szczególności w raporcie została podjęta decyzja o wybraniu nowego standardu w procesie otwartego konkursu oraz określone zostały ogólne wymagania dla rozwiązań i kryteria, jakimi powinni się kierować interesariusze konkursu przy projektowaniu i ocenie rozwiązań. Ponadto, w raporcie określono, że w ramach procesu poszukiwane będą algorytmy szyfrujące, kryptograficzne funkcje skrótu oraz algorytmy szyfrowania z uwierzytelnianiem.

Po upływie kolejnego roku, w maju 2018 opublikowane zostały wymagania dla zgłoszeń [7]. Wraz z publikacją został otwarty konkurs standaryzacyjny, do którego zgłoszonych zostało 57 kandydatów w różnych kategoriach. Raport z pierwszej rundy [8] został opublikowany w październiku 2019. Po zakończeniu pierwszej rundy pula kandydatów została ograniczona z 57 do 32. Po kolejnych dwóch latach, w lipcu 2021 ogłoszony został raport z drugiej rundy [9], w której wybranych zostało 10 finalistów:

- `Ascon` [10];
- `Elephant` [11];
- `GIFT-COFB` [12];
- `Grain` [13];
- `ISAP` [14];
- `PHOTON-Beetle` [15];
- `Romulus` [16];
- `SPARKLE` [17];
- `TinyJAMBU` [18];
- `Xoodyak` [19].

Pomimo, że oryginalne wymagania nie brały pod uwagę potrzeby oceny zgłoszeń pod względem bezpieczeństwa na ataki z użyciem komputera kwantowego, to wszelkie badania przeprowadzone w tej dziedzinie zostały wzięte pod uwagę w rozstrzygnięciu konkursu i ostatecznie zostały zamieszczone w raporcie z przebiegu ostatniej rundy konkursu [20].

Rezultatem zakończenia konkursu na nowy standard kryptografii lekkiej był wybór algorytmu szyfrowania z uwierzytelnianiem oraz kryptograficznej funkcji skrótu `Ascon`. Procedura formalizacji standardu jest w trakcie realizacji i spisywany jest aktualnie szkic nowego dokumentu standaryzującego [21].

Ostatnie lata pokazują, że ocena bezpieczeństwa kryptografii symetrycznej przed atakami z użyciem komputera kwantowego staje się coraz bardziej istotna. Pierwsze analizy pojawiły się już w 2015 roku, zaczynając od zbadania właściwości standardu szyfru `AES`. Artykuł M. Grassla [22] zawiera pierwsze badanie, w którym została opracowana nowa metoda badania szyfrów poprzez sprowadzenie ich algorytmu do postaci układu logiki odwracalnej i oszacowania jego parametrów. Taka analiza pozwala przewidywać, jakie zasoby powinien posiadać komputer kwantowy, aby zrealizować atak z powodzeniem. W kolejnych latach pojawiły się nowe prace S. Jaquesa (2020) [23] i B. Langenberga (2020) [24], które zredukowały rozmiar takiej implementacji szyfru `AES`.

Badaniom poddane zostały inne szyfry szczególnego zainteresowania:

- Bliźniacze algorytmy `SIMON` i `SPECK` [25] w pracach G. Leandera (2017) [26], R. Ananda (2020) [27], K. Janga (2020) [28] oraz B. Karthikeyana (2020) [29];
- Szyfr `KATAN` [30] - w pracy R. Mostafizar (2022) [31];
- Szyfr `ChaCha20` [32] - w pracy B. Bhagwan (2021) [33].

Podobne analizy zostały przeprowadzone również dla szyfrów, które nie należą do ogólnie pojętego nurtu wschodniego krajów Europy i USA, który dyktowany jest przez instytucje takie jak `NATO, NIST` czy `ETSI`. W ostatnich latach przeprowadzono badania nad standardowymi szyframi następujących państw:

- Korea Południowa - szyfry `LEA, HIGHT, CHAM, PIPO` - w pracy K. Janga (2020) [34] oraz G. Songa (2021) [35];
- Japonia - szyfr `Camellia` - w pracy L. Yanjun (2021) [36];
- Rosja - szyfry `Gost, Kuznechik, MAGMA` - w pracach D. Denisenko (2019) [37] oraz G. Marshalko (2019) [38];
- Chiny - szyfry `ShangMi 3, ShangMi 4` - w pracy P. Jong Hwan (2022) [39].

Pokazuje to, że zainteresowanie takimi badaniami obecne jest nie tylko dla szyfrów używanych powszechnie, ale też dla szyfrów opracowanych na rzecz rządów państw o dużym znaczeniu na arenie międzynarodowej. Badania takie mogą mieć wpływ na bezpieczeństwo informacyjne wielu państw, a co za tym idzie również na sytuację geopolityczną Świata.

Praca R. Ananda [40] przedstawia ogólną metodykę opracowywania takich ataków i badań dla szyfrów opartych o rejestry ze sprzężeniem zwrotnym `FSR`. Przedstawia to pewien poziom dojrzałości tej dziedziny z racji pojawiania się bardziej ogólnych podejść do zagadnienia, które zostały wytworzone na podstawie szczególnych przypadków badań dla pojedynczych szyfrów.

Poza atakami opartymi o wykorzystanie algorytmu Grovera zaczynają ujawniać się inne podejścia do próby złamania szyfrów z użyciem komputera kwantowego. Obiecujące wyniki wykorzystania wyżarzania kwantowego zostały przedstawione w pracy E. Burek (2022) [41].

Zagrożenie wynikające z rozwoju prac nad komputerem kwantowym zostało również zaobserwowane dla kryptografii asymetrycznej. Algorytm Shora[42], oparty o kwantową transformację Fouriera, jest w stanie efektywnie odnajdywać okresy dys-

kretnych funkcji, co za tym idzie efektywnie rozwiązywać problem logarytmu dyskretnego. Korzystając z redukcji problemu faktoryzacji liczb do problemu logarytmu dyskretnego E. Bacha [43], możliwe jest również efektywne rozkładanie liczb całkowitych na czynniki pierwsze.

Skutkiem tego, wszelkie asymetryczne systemy kryptograficzne oparte o te dwa problemy są istotnie zagrożone w perspektywie postępów technologii kwantowych, a w tym, w szczególności:

- `RSA` – *Rivest-Shamir-Adleman* [44];
- `DSA` – *Digital Signature Algorithm* [45];
- `ECDSA` – *Elliptic Curve Digital Signature Algorithm*[46];
- `DH` – *Diffie-Hellman* [47];
- `ECDH` – *Elliptic Curve Diffie-Hellman* [48].

Próba zażegnania tego niebezpieczeństwa została rozpoczęta w roku 2017 poprzez ogłoszenie przez NIST nowego konkursu na standard kryptografii postkwantowej *(z j. ang. NIST Post-Quantum Cryptography – PQC)* [49].

Do marca 2025 po 4 rundach konkursowych wybrane zostało łącznie pięć standardowych kryptosystemów postkwantowych [50]. Trzy algorytmy podpisu cyfrowego:

- `CRYSTALS-DILITHIUM` [51];
- `FALCON` [52];
- `Sphincs+` [53].

Oraz dwa algorytmy enkapsulacji kluczy:

- `CRYSTALS-KYBER` [54];
- `HQC` [55].

Cel pracy

Celem niniejszej pracy jest poszerzenie stanu wiedzy na temat badania bezpieczeństwa symetrycznych prymitywów kryptograficznych przed atakami z użyciem komputera kwantowego.

Problemy poruszone w pracy

Praca porusza zagadnienia dotyczące badania logiki kwantowej i odwracalnej. W szczególności reprezentowania symetrycznych prymitywów kryptograficznych jako układy odwracalne. Badania skupione zostały na poszukiwaniu efektywnych metod modelowania szyfrów jako układy i optymalizacji tych układów. Dalszy ciąg badań został poświęcony poszukiwaniom możliwości wnioskowania o bezpieczeństwie kwantowym szyfrów przy użyciu utworzonego układu. W ramach pracy zostały przeprowadzone i opisane wyniki następujących badań:

- Synteza optymalnych układów logiki odwracalnej reprezentujących skrzynki podstawieniowe wybranych szyfrów i kryptograficznych funkcji skrótu;
- Synteza kompletnej przestrzeni szablonów identyczności;
- Badanie korelacji pomiędzy klasycznymi właściwościami bezpieczeństwa skrzynek podstawieniowych, a optymalnym rozmiarem implementacji;
- Badanie bezpieczeństwa kwantowego szyfrów z rodziny `SPARKLE`;
- Badanie bezpieczeństwa kwantowego standardu szyfrowania `Ascon`.

Wyniki uzyskane w pracy

Dla wszystkich wskazanych problemów udało się znaleźć nowe wyniki, które są całkowicie odtwarzalne, deterministyczne i niezależne od wybranych próbek badawczych lub źródeł losowości:

- Rozdział 5 – Po raz pierwszy znaleziono optymalne implementacje wybranych skrzynek podstawieniowych jako układy logiki kwantowej i odwracalnej, a także dowiedziono ich optymalności, w tym dla nowego standardu szyfrowania lekkiego `Ascon`;
- Rozdział 6 – Po raz pierwszy wyznaczono kompletną przestrzeń szablonów identyczności oraz reguł podstawieniowych służących do optymalizacji układów logiki kwantowej i odwracalnej;
- Rozdział 7 – Po raz pierwszy sformułowano problem istnienia korelacji pomiędzy klasycznymi właściwościami bezpieczeństwa skrzynek podstawieniowych, a optymalnymi rozmiarami implementacji oraz niezaprzeczalnie zaobserwowano istnienie takiej korelacji;
- Rozdział 8 – Po raz pierwszy dokonano całościowej oceny bezpieczeństwa kwantowego szyfrów z rodziny `SPARKLE`;
- Rozdział 9 – Po raz pierwszy dokonano całościowej oceny bezpieczeństwa kwantowego standardu szyfrowania `Ascon`.

Rozdział 1

Pojęcia podstawowe

Definicja 1. *Algebra Boole'a* $\mathbb{B} = \{0, 1, \neg, \wedge, \vee\}$ *– struktura algebraiczna złożona z dwóch elementów –* 0 *i* 1*, jednoargumentowego działania negacji oraz dwuargumentowych działań koniunkcji i alternatywy. Działania spełniają aksjomaty algebry Boole'a opisane tabelami poniżej:*

x	$\neg x$
0	1
1	0

x	y	$x \wedge y$	$x \vee y$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Tabela 1.1: Tabele prawdy podstawowych działań algebry Boole'a

Definicja 2. *Alternatywa wykluczająca* $\oplus$ *– dwuargumentowe działanie wewnętrzne nad algebrą Boole'a opisane równaniem:*

$$a \oplus b \ = (a \wedge \neg b) \vee (\neg a \ \wedge b). \tag{1.1}$$

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

Tabela 1.2: Tabela prawdy alternatywy wykluczającej

Na potrzeby pracy, działanie koniunkcji będzie nazywane też mnożeniem, a alternatywy wykluczającej dodawaniem. Wyniki tych działań nazywane też będą odpowiednio iloczynem oraz sumą. Mnożenie elementów będzie również notowane domniemanym (pominiętym) symbolem działania, np. $a \wedge b \ = ab$.

Twierdzenie 1. *Struktura algebraiczna złożona ze zbioru elementów algebry Boole'a* $\{0,1\}$ *z dołączonymi działaniami dodawania i mnożenia* $(\oplus, \wedge)$ *jest izomorficzna z ciałem Galois rzędu 2 –* $GF(2)$.

Definicja 3. $\mathbb{B}^n$ *- Przestrzeń wektorowa złożona z n-elementowych krotek o współrzędnych z algebry Boole'a, z działaniem dodawania wektorów* $(u \oplus v)$ *oraz mnożenia przez skalar* $(k * v)$*:*

$$\mathbb{B}^n = \{[v_{n-1}, v_{n-2}, \ldots, v_1, v_0] : \forall(v_i \in \mathbb{B})\}. \tag{1.2}$$

$$[u_{n-1}, \ldots, u_0] \oplus [v_{n-1}, \ldots, v_0] = [u_{n-1} \oplus v_{n-1}, \ldots, u_0 \oplus v_0]. \tag{1.3}$$

$$k * [u_{n-1}, \ldots, u_0] = [ku_{n-1}, \ldots, ku_0]. \tag{1.4}$$

Struktura spełnia aksjomaty przestrzeni wektorowej, ponieważ jest przestrzenią współrzędnych nad strukturą izomorficzną z ciałem $GF(2)$.

Przestrzeń wektorowa $\mathbb{B}^n$ zawiera dokładnie 2^n elementów – $|\mathbb{B}^n| = 2^n$. Dla uproszczenia, wektory z tej przestrzeni będą oznaczane liczbami naturalnymi ze zbioru $\{0, 1, ..., 2^n - 1\}$, wyznaczonymi ze wzoru:

$$v \ = \sum_{i=0}^{n-1} v_i * 2^i. \tag{1.5}$$

Na przykład:

$$\mathbb{B}^4 \ni (1,0,1,0) = 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0 = 6. \tag{1.6}$$

Definicja 4. *Funkcja boolowska - Dowolna funkcja* f *z n-wymiarowej przestrzeni* $\mathbb{B}^n$ *do algebry Boole'a* $\mathbb{B}$*:*

$$f : \mathbb{B}^n \to \mathbb{B}. \tag{1.7}$$

Na potrzeby pracy, dla uproszczenia, funkcje pomiędzy przestrzeniami wektorowymi postaci $f : \mathbb{B}^n \to \mathbb{B}^m$, również będą nazywane funkcjami boolowskimi.

Definicja 5. *Postać* `ANF`*, Formuła* `ANF` *– Algebraiczna postać normalna (z j. ang. algebraic normal form). Postać funkcji f Boolowskiej jako wielomian nad ciałem Galois $GF(2)$:*

$$\begin{aligned} f &= \bigoplus_i t_i \\ t_i &= \bigwedge_p x_p \end{aligned} \tag{1.8}$$

Każdy term t_i jest iloczynem podzbioru argumentów funkcji f.

Definicja 6. *Postać* `CNF`*, Formuła* `CNF` *– Koniunkcyjna postać normalna (z j. ang. conjunctive normal form). Postać funkcji f Boolowskiej jako koniunkcja klauzul:*

$$\begin{aligned} f &= \bigwedge_i c_i \\ c_i &= \bigvee_p x_p \vee \bigvee_r \neg x_r \end{aligned} \tag{1.9}$$

Każda klauzula c_i jest alternatywą literałów, czyli zmiennych x_p i zanegowanych zmiennych x_n.

Definicja 7. $\mathbb{C}$ *– Ciało liczb zespolonych.*

$$\mathbb{C} = \{a + bi : a, b \in \mathbb{R}, i^2 = -1\} \tag{1.10}$$

Definicja 8. *Szyfr symetryczny – Piątka uporządkowana* $(\mathbb{K}, \mathbb{M}, \mathbb{C}, \mathbb{E}, \mathbb{D})$

- $\mathbb{K} = \{K\}$ – przestrzeń kluczy;
- $\mathbb{M} = \{M\}$ – przestrzeń wiadomości;
- $\mathbb{C} = \{C\}$ – przestrzeń szyfrogramów;
- $\mathbb{E} : \mathbb{K} \times \mathbb{M} \to \mathbb{C}$ – funkcja szyfrująca;
- $\mathbb{D} : \mathbb{K} \times \mathbb{C} \to \mathbb{M}$ – f. deszyfrująca, odwrotna do $\mathbb{E}$ przy ustalonym kluczu:

$$\forall_{K \in \mathbb{K}, M \in \mathbb{M}} D(K, E(K, M)) = M. \tag{1.11}$$

Dodatkowo, dla znanego szyfrogramu $C = E(M, K)$, wyznaczenie M lub K powinno być trudne obliczeniowo, a dla znanego tekstu jawnego i szyfrogramu $M, C = E(M, K)$, wyznaczenie klucza k powinno być trudne obliczeniowo.

Definicja 9. *Szyfr AEAD (z j. ang. Authenticated Encryption with Associated Data) – Krotka uporządkowana* $(\mathbb{K}, \mathbb{M}, \mathbb{C}, \mathbb{T}, \mathbb{A}, \mathbb{E}, \mathbb{D})$

- $\mathbb{K}, \mathbb{M}, \mathbb{C}$ – Jak w przypadku szyfru symetrycznego;
- $\mathbb{A} - \{A\}$ – Przestrzeń jawnych danych dodatkowych, nazywanych też nagłówkami wiadomości;
- $\mathbb{T} = \{T\}$ – Przestrzeń tagów uwierzytelniania;
- $E : \mathbb{K} \times \mathbb{A} \times \mathbb{M} \to \mathbb{C} \times \mathbb{T}$ – funkcja szyfrująca i uwierzytelniająca;
- $D : \mathbb{K} \times \mathbb{A} \times (\mathbb{C} \times \mathbb{T}) \to \mathbb{M} \cup \{\perp\}$ – funkcja deszyfrująca i weryfikująca, odwrotna do E przy ustalonym kluczu i wartości danych dodatkowych:

$$\forall_{K \in \mathbb{K}, A \in \mathbb{A}, M \in \mathbb{M}} D(K, A, E(K, A, M)) = M. \tag{1.12}$$

  Funkcja D zwraca symbol $\perp$ jeżeli otrzymany szyfrogram C i tag T nie został uzyskany przy użyciu klucza K.

Analogicznie, znajomość jawnie przesyłanych informacji takich jak tag uwierzytelnienia, szyfrogram, dane dodatkowe nie powinna umożliwiać odtworzenia klucza lub zaszyfrowanej wiadomości. Dodatkowo utworzenie szyfrogramu i tagu uwierzytelniania bez znajomości klucza K, takich że mogą zostać pozytywnie zweryfikowane tym kluczem musi być trudne obliczeniowo.

Szyfry AEAD mogą wymagać, aby elementy zbioru A zawierały losowo wybraną wartość jednorazową. Celem dodania takiej wartości losowej jest uniemożliwienie ataków powtórzeniowych gdzie wykorzystany jest fakt, iż wysłanie tej samej wiadomości zaszyfrowanej tym samym kluczem daje ten sam szyfrogram co może zostać zaobserwowane i wykorzystane przez atakującego.

Rozdział 2

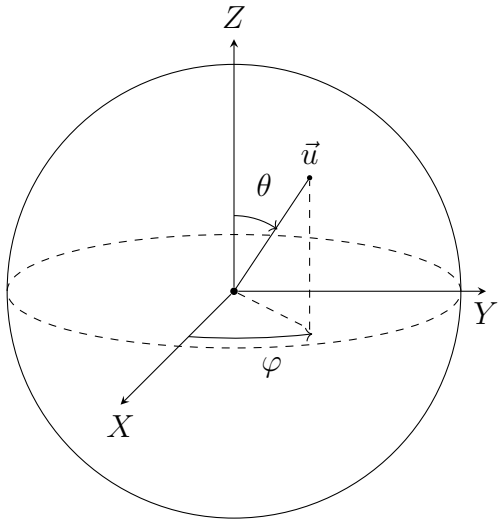
Model obliczeń kwantowych

Poniższy rozdział wprowadza uproszczone aksjomaty modelu obliczeń kwantowych. Przedstawiony model jest wystarczający do zrozumienia notacji używanej w pracy i podstaw obliczeń kwantowych. Model obliczeń oparty jest o oryginalne prace Felixa Blocha, [56], Clauda Cohen-Tannoudji [57] oraz Ramamurtiego Shankara [58], które po raz pierwszy wprowadzają reprezentację stanu qubitów jako wektorów zespolonych w przestrzeni Hilberta, a operacji na stanach jako macierzy unitarnych.

2.1 Qubit

Definicja 10. *Qubit – podstawowy nośnik informacji w kwantowym modelu obliczeniowym. Odpowiednik bitu z klasycznego modelu obliczeniowego.*

W przeciwieństwie do klasycznego bitu, którego zbiór stanów to dwa elementy 0 i 1, zbiór stanów qubitu opisany jest Sferą Blocha [56]. Pojedynczy qubit może być interpretowany jako wektor jednostkowy, stałej długości i w stałym punkcie zaczepienia, osadzony w trójwymiarowej przestrzeni euklidesowej.



Rysunek 2.1: Sfera blocha

Do opisania stanu qubitu w takiej interpretacji potrzebne są dwa parametry kątowe – θ oraz ϕ. Pierwszy to kąt odchylenia wektora od górnej półosi pionowej, zwyczajowo oznaczanej Z i mieści się w przedziale $\theta \in [0, \pi]$. Drugi to kąt odchylenia od jednej z półosi poziomych, zwyczajowo oznaczanej X i mieści się w przedziale $\phi \in [0, 2\pi]$. Długość wektora nie jest parametrem stanu qubitu, ponieważ jest stała oraz równa 1.

Tak opisany model qubitu można również przedstawić przy pomocy struktury algebraicznej $\mathbb{H}_1$, skonstruowanej następująco:

1. $\mathbb{C}^2 = \{v = [a, b]^T : a, b \in \mathbb{C}\}$ - Zbiór dwuelementowych wektorów kolumnowych o współczynnikach zespolonych;
2. $\overline{\mathbb{C}^2} = \{v = [a, b]^T : v \in \mathbb{C}^2 \wedge |a|^2 + |b^2| = 1\}$ - Zbiór wektorów unitarnych z $\mathbb{C}^2$;
3. $\sim$ - Relacja równoważności fazy globalnej, taka że:

$$u \sim v \iff \exists_{\phi \in \mathbb{R}}(u = e^{i\phi}v) \tag{2.1}$$

4. $[v] = \{x : x \sim v\}$ - Klasa abstrakcji wektora v pod relacją $\sim$;
5. $\mathbb{H}_1 = \overline{\mathbb{C}^2}/\sim = \{[v] : v \in \overline{\mathbb{C}^2}\}$ - Struktura ilorazowa powstała z podziału $\overline{\mathbb{C}^2}$ na klasy abstrakcji relacji $\sim$.

Wszystkie wektory z tej przestrzeni będą zapisywane w notacji *bra-ket*, w szczególności, zgodnie z konwencją oznaczenie *ket* $|v\rangle$, będzie używane do zapisu wektorów kolumnowych.

W zbiorze $\mathbb{H}_1$ wyróżnione są dwa normalne i ortogonalne wektory stanu, które tworzą bazę zwaną też bazą kanoniczną $\mathcal{B}_1$.

$$\mathcal{B}_1 = \left\{ |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\} \tag{2.2}$$

Dowolny wektor opisany kątami θ i ϕ, można zapisać jako kombinację liniową wektorów tej bazy:

$$\begin{gathered}\left|\psi_{(\theta,\phi)}\right\rangle = \cos\frac{\theta}{2}\left|0\right\rangle + e^{i\phi}\sin\frac{\theta}{2}\left|1\right\rangle = \alpha_0\left|0\right\rangle + \alpha_1\left|1\right\rangle = \begin{bmatrix}\alpha_0\\ \alpha_1\end{bmatrix} \\ \alpha_0 \in \mathbb{R}, \alpha_1 \in \mathbb{C}, |\alpha_0|^2 + |\alpha_1|^2 = 1.\end{gathered} \tag{2.3}$$

Należy pamiętać, że stan qubitu w rzeczywistości jest reprezentowany przez klasę abstrakcji wektorów, powyżej został opisany jeden z jej przedstawicieli. Wszystkie elementy tej klasy to wektory postaci:

$$\begin{aligned}\left|\psi_{(\theta,\phi)}\right\rangle &= e^{i\lambda}\left(\cos\frac{\theta}{2}\left|0\right\rangle + e^{i\phi}\sin\frac{\theta}{2}\left|1\right\rangle\right) = \\ &= e^{i\lambda}\alpha_0\left|0\right\rangle + e^{i\lambda}\alpha_1\left|1\right\rangle = \alpha_0'\left|0\right\rangle + \alpha_1'\left|1\right\rangle \\ \alpha_0', \alpha_1' &\in \mathbb{C}, |\alpha_0'|^2 + |\alpha_1'|^2 = 1.\end{aligned} \tag{2.4}$$

Gdzie λ jest dowolnym kątem, nazywanym też fazą globalną. Łatwo zauważyć, że wszystkie takie wektory są równoważne w relacji opisanej równaniem 2.1.

2.2 Operacje na qubicie

Jedynymi operacjami, które można realizować na pojedynczym qubicie, są bezwarunkowe rotacje opisane w pracy A. Barenco [59]. Aby wykonać rotację, należy wybrać oś rotacji oraz jej kąt. Niedozwolone są rotacje, których oś lub kąt jest jakkolwiek zależny od stanu qubitu.

Rotacje wzdłuż osi X, Y, Z o kąt δ można w tym modelu przedstawić macierzami:

$$\begin{aligned}R_x(\delta) &= \begin{bmatrix}\cos\frac{\delta}{2} & -i\sin\frac{\delta}{2}\\ -i\sin\frac{\delta}{2} & \cos\frac{\delta}{2}\end{bmatrix} \\ R_y(\delta) &= \begin{bmatrix}\cos\frac{\delta}{2} & -\sin\frac{\delta}{2}\\ \sin\frac{\delta}{2} & \cos\frac{\delta}{2}\end{bmatrix} \\ R_z(\delta) &= \begin{bmatrix}e^{-i\frac{\delta}{2}} & 0\\ 0 & e^{i\frac{\delta}{2}}\end{bmatrix}\end{aligned} \tag{2.5}$$

Rotację na stanie zadanym w postaci wektora wykonuje się poprzez pomnożenie tego wektora przez macierz rotacji. Rotacje wzdłuż innych osi są również dozwolone. Wszystkie rotacje pojedynczego qubitu możliwe są do zaprezentowania macierzami unitarnymi [60].

Na potrzeby pracy zostaną wyszczególnione cztery operacje:

1. **N** - Negacja - podobnie jak klasyczny odpowiednik zamienia wartość bitu na przeciwny, tak negację można albo interpretować jako zamianę wektorów bazowych albo zamianę współczynników przy wektorach bazowych. Negację realizuje się poprzez rotację wzdłuż osi X o kąt $\delta = \pi$.

$$\begin{aligned} N\left|\psi\right\rangle &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_0 \end{bmatrix}, \\ N\left|0\right\rangle &= \left|1\right\rangle, \quad N\left|1\right\rangle = \left|0\right\rangle, \\ N^2\left|\psi\right\rangle &= \left|\psi\right\rangle. \end{aligned} \tag{2.6}$$

2. **H** - Bramka Hadamarda - Rotacja o kąt $\delta = \pi$ wzdłuż osi skośnej pomiędzy osiami X i Z, przechylonej o $\frac{\pi}{4}(45°)$.

$$\begin{aligned} H\left|\psi\right\rangle &= \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}}(\alpha_0 + \alpha_1) \\ \frac{1}{\sqrt{2}}(\alpha_0 - \alpha_1) \end{bmatrix} \\ H\left|0\right\rangle &= \frac{1}{\sqrt{2}}\left|0\right\rangle + \frac{1}{\sqrt{2}}\left|1\right\rangle, \quad H\left|1\right\rangle = \frac{1}{\sqrt{2}}\left|0\right\rangle - \frac{1}{\sqrt{2}}\left|1\right\rangle, \\ H^2\left|\psi\right\rangle &= \left|\psi\right\rangle. \end{aligned} \tag{2.7}$$

3. **Z** - Rotacja z - Rotacja wzdłuż osi Z o kąt $\delta = \pi$.

$$\begin{aligned} Z\left|\psi\right\rangle &= \begin{bmatrix} e^{\frac{-i\pi}{2}} & 0 \\ 0 & e^{\frac{i\pi}{2}} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \begin{bmatrix} e^{\frac{-i\pi}{2}}\alpha_0 \\ e^{\frac{i\pi}{2}}\alpha_1 \end{bmatrix} = e^{\frac{-i\pi}{2}} \begin{bmatrix} \alpha_0 \\ e^{i\pi}\alpha_1 \end{bmatrix} \equiv \begin{bmatrix} \alpha_0 \\ -\alpha_1 \end{bmatrix} \\ Z\left|0\right\rangle &= \left|0\right\rangle \quad Z\left|1\right\rangle = -\left|1\right\rangle \\ Z^2\left|\psi\right\rangle &= \left|\psi\right\rangle. \end{aligned} \tag{2.8}$$

4. **I** - Identyczność, operacja nie zmieniająca stanu qubitu.

$$I|\psi\rangle = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = |\psi\rangle \tag{2.9}$$
$$I|0\rangle = |0\rangle \quad I|1\rangle = |1\rangle$$

2.3 Pomiar qubitu

Definicja 11. *Pomiar qubitu – niedeterministyczny i nieodwracalny proces uzyskiwania informacji o stanie qubitu, wiążący się z jego nieodwracalnym zaburzeniem.*

Do obu stanów bazowych przypisuje się klasyczne wartości bitowe, kolejno 0 i 1. dla dowolnego wektora stanu qubitu $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, pomiar stanu przekształca go na jeden z wektorów bazowych z prawdopodobieństwem $P(|x\rangle) = |\alpha_x|^2$. Wynikiem pomiaru jest jedna z przypisanych wartości bitowych, odpowiadająca wektorowi bazowemu, do którego stan został przekształcony [61].

2.4 Rejestr kwantowy

Definicja 12. *Rejestr kwantowy – Złożony nośnik danych w kwantowym modelu obliczeniowym powstały ze złożenia wielu qubitów.*

Stan rejestru złożonego z n qubitów opisuje się wektorem o 2^n współczynnikach. Łączny stan rejestru złożonego z wielu qubitów można wyznaczyć korzystając z iloczynu Kroneckera. Przykładowo, stan rejestru złożonego z dwóch qubitów $|\psi\rangle$ i $|\phi\rangle$, można opisać wektorem:

$$|\psi\phi\rangle = |\psi\rangle \otimes |\phi\rangle = \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} \otimes \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \alpha_0\beta_0 \\ \alpha_0\beta_1 \\ \alpha_1\beta_0 \\ \alpha_1\beta_1 \end{bmatrix} \tag{2.10}$$

Stan rejestru złożonego z wielu qubitów $[|\psi_{n-1}\rangle, \ldots, |\psi_1\rangle, |\psi_0\rangle]$ równy jest:

$$|\psi_{n-1} \ldots \psi_1\psi_0\rangle = \bigotimes_{i=0}^{n-1} |\psi_i\rangle = |\psi_{n-1}\rangle \otimes \cdots \otimes |\psi_0\rangle \otimes |\psi_0\rangle \tag{2.11}$$

Zbiór stanów rejestru złożonego z n qubitów to $\mathbb{H}_n$, który jest tworzony analogicznie jak zbiór stanów jednego qubitu, jednakże rozpoczynając od zbioru $\mathbb{C}^{2^n}$.

$$\mathbb{H}_n = \left(\overline{\mathbb{C}^{2^n}} / \sim\right) = \{[v] : v \in \overline{\mathbb{C}^{2^n}}\} \tag{2.12}$$

Dla każdej przestrzeni $\mathbb{H}_n$ można wskazać bazę ortonormalną $\mathcal{B}_n$, która zawiera 2^n ortonormalnych wektorów oznaczonych indeksami naturalnymi z $\{0, 1, \ldots, 2^n - 1\}$.

$$\begin{aligned} \mathcal{B}_n &= \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \cdots, \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \right\} = \\ &= \{|0\rangle, |1\rangle, |2\rangle, |3\rangle, \cdots, |2^n - 1\rangle\}. \end{aligned} \tag{2.13}$$

Należy zwrócić uwagę na to, że oznaczenia indeksami naturalnymi w notacji *ket* mogą się powtarzać pomiędzy bazami. To samo oznaczenie może wskazywać na różne wektory, jednakże pochodzące z różnych baz. W nielicznych przypadkach, gdy rozmiar wektora i rejestru nie będzie wprost wynikał z kontekstu, rozmiar zostanie oznaczony dodatkowym indeksem dolnym, np. $|0_4\rangle \in \mathcal{B}_4$ będzie wektorem długości $16 = 2^4$ oznaczającym stan rejestru złożonego z 4 qubitów.

Dowolną bazę $\mathcal{B}_n$ można utworzyć przez złożenie n baz jednego qubitu $\mathcal{B}_1$, np:

$$
\begin{aligned}
\mathcal{B}_2 &= \mathcal{B}_1 \otimes \mathcal{B}_1 = \\
&= \{|0\rangle, |1\rangle\} \otimes \{|0\rangle, |1\rangle\} = \\
&= \{|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle\} = \\
&= \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\} = \\
&= \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right\} = \\
&= \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\} = \{|0\rangle, |1\rangle, |2\rangle, |3\rangle\}
\end{aligned}
\tag{2.14}
$$

Z takiej konstrukcji bazy $\mathcal{B}_n$, naturalnie wynika dodatkowa notacja jej elementów poprzez ciąg, który odpowiada wektorom bazowym, pojedynczych qubitów, z których został wyznaczony, np. $|01\rangle = |0\rangle \otimes |1\rangle$. Łatwo też zauważyć, że notacja ta jest równoważna z zapisem indeksu naturalnego w postaci binarnej.

2.5 Stany splątane

W przeciwieństwie do klasycznych rejestrów bitowych, zbiór stanów rejestru kwantowego $\mathbb{H}_n$ nie jest iloczynem kartezjańskim zbiorów stanów pojedynczych jego elementów $\mathbb{H}_1^n$. Zbiór stanów zawiera taki iloczyn, ale zawiera też inne elementy.

$$
\begin{gathered}
(\mathbb{H}_1^n \subset \mathbb{H}_n) \wedge (\mathbb{H}_1^n \neq \mathbb{H}_n) \\
\mathbb{H}_1^n \setminus \mathbb{H}_n = \mathbb{S}_{\ltimes} \neq \phi
\end{gathered}
\tag{2.15}
$$

Zbiór $\mathbb{S}_n$ jest zbiorem stanów splątanych rejestru n qubitowego. Przykładami takich stanów dla dwóch qubitów, są *Stany Bella*, a jednym z nich jest stan $|\Phi^+\rangle$:

$$\left|\Phi^+\right\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}}\left|00\right\rangle + 0\left|01\right\rangle + 0\left|10\right\rangle + \frac{1}{\sqrt{2}}\left|11\right\rangle = \frac{1}{\sqrt{2}}\left(\left|00\right\rangle + \left|11\right\rangle\right). \quad (2.16)$$

Można udowodnić, że nie istnieją dwa takie stany qubitów, których iloczyn Kroneckera utworzy wektor wskazanego stanu $|\Phi^+\rangle$. Wszystkie stany rejestru, których nie da się rozłożyć na stany pojedynczych qubitów, nazywane są stanami splątanymi.

2.6 Wybrane operacje na rejestrach kwantowych

Podobnie jak w przypadku pojedynczego qubitu, dla którego operacje opisuje się macierzami unitarnymi rozmiaru 2×2, tak wszystkie operacje na rejestrze rozmiaru n można opisać jako macierze unitarne rozmiaru $2^n \times 2^n$.

Można zauważyć, że korzystając z takiej definicji, nie jest możliwe wykonanie jakiejkolwiek operacji, jeżeli jej rozmiar nie jest zgodny z rozmiarem rejestru. W szczególności nie można wykonać operacji jednoqubitowej na rejestrze złożonym z wielu qubitów. Aby to umożliwić, należy wygenerować odpowiednio większą macierz, która opisuje to działanie na całym rejestrze. Do tego celu można wykorzystać właściwość rozdzielności iloczynu Kroneckera nad mnożeniem macierzy oraz podstawienia operacji identyczności. Poniżej wyznaczony jest przykładowy operator N_1, który reprezentuje negację qubitu o indeksie 1, znajdującego się w dwuelementowym rejestrze.

$$\begin{aligned} N_1(|\psi_1\psi_0\rangle) &= N\left|\psi_1\right\rangle \otimes \left|\psi_0\right\rangle = N\left|\psi_1\right\rangle \otimes I\left|\psi_0\right\rangle = \\ &= (N \otimes I)(\left|\psi_1\right\rangle \otimes \left|\psi_0\right\rangle) = (N \otimes I)\left|\psi_1\psi_0\right\rangle \implies \\ \implies N_1 &= N \otimes I \end{aligned} \quad (2.17)$$

Oczywiście dowód ten zachodzi tylko dla stanów, które nie są splątane, niemniej jednak pamiętając, że wszelkie operacje są bezwarunkowe i zawsze są opisane tymi samymi operatorami niezależnie od stanów rejestru, na których operują, można rozszerzyć to rozumowanie na stany splątane, uzyskując tym wzór na operator N_1.

$$
\begin{aligned}
N_1 &= N \otimes I = \\
&= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes I = \begin{bmatrix} 0*I & 1*I \\ 1*I & 0*I \end{bmatrix} = \\
&= \begin{bmatrix} 0 * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & 1 * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ 1 * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & 0 * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}
\end{aligned}
\tag{2.18}
$$

Można teraz zauważyć, że:

$$
\begin{aligned}
N_1 |00\rangle &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = |10\rangle \\
N_1 |01\rangle &= |11\rangle \,,\; N_1 |10\rangle = |00\rangle \,,\; N_1 |11\rangle = |01\rangle \,.
\end{aligned}
\tag{2.19}
$$

Zatem analogicznie taką negację można również traktować albo jako zmianę jednego symbolu w wektorze bazowym, lub permutację współczynników.

Przykładem operacji, którą można wykonać na niemniej niż dwóch qubitach, jest kontrolowana negacja `CNOT`. Wymaga ona wskazania jednego qubitu kontrolującego *(z j. ang. control qubit)* oraz docelowego *(z j. ang. target qubit)*. Działa ona podobnie jak zwykła negacja, z tą różnicą, że działa wyłącznie dla stanów bazowych, w których qubit kontrolny jest w stanie $|1\rangle$, pozostałe stany nie są modyfikowane. Przykładowo, niech indeks qubitu kontrolnego to 1, a docelowego to 0. Operacja `CNOT` przekształca wtedy wektory bazowe w następujący sposób:

$$
\begin{aligned}
\texttt{CNOT}_{1,0} |00\rangle &= |00\rangle \\
\texttt{CNOT}_{1,0} |01\rangle &= |01\rangle \\
\texttt{CNOT}_{1,0} |10\rangle &= |11\rangle \\
\texttt{CNOT}_{1,0} |11\rangle &= |10\rangle
\end{aligned}
\tag{2.20}
$$

Co można zaprezentować macierzą:

$$\texttt{CNOT}_{1,0} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.21}$$

Dla stanu $|b_1, b_0\rangle$ przekształcenie to można interpretować algebraicznie jako:

$$\texttt{CNOT}_{1,0} |b_1, b_0\rangle = |b_1, b_0 \oplus b_1\rangle \tag{2.22}$$

Drugą operacją, która wymaga wprowadzenia, jest kontrolowana rotacja Z, oznaczana symbolem CZ. Wymaga ona wskazania zbioru qubitów kontrolujących, CZ zamienia współczynniki przy stanach bazowych na przeciwne, ale tylko takich, dla których wszystkie wybrane qubity kontrolujące są w stanie $|1\rangle$. Np. dla rejestru o rozmiarze 3 i indeksach qubitów kontrolujących 0 i 2:

$$\begin{aligned} CZ_{2,0} |000\rangle &= +|000\rangle, \quad CZ_{2,0} |100\rangle = +|100\rangle, \\ CZ_{2,0} |001\rangle &= +|001\rangle, \quad CZ_{2,0} |101\rangle = -|101\rangle, \\ CZ_{2,0} |010\rangle &= +|010\rangle, \quad CZ_{2,0} |110\rangle = +|110\rangle, \\ CZ_{2,0} |011\rangle &= +|011\rangle, \quad CZ_{2,0} |111\rangle = -|111\rangle. \end{aligned} \tag{2.23}$$

2.7 Pomiar rejestru

Pomiar rejestru odbywa się analogicznie do pomiaru qubitu. Pomiar powoduje zaburzenie rejestru i przekształcenie jego stanu w jeden ze stanów bazowych. Podobnie jak wcześniej, prawdopodobieństwo to jest zależne od współczynnika zespolonego stojącego przy tym stanie.

Dla stanu bazowego $|X\rangle$, prawdopodobieństwo, że stan rejestru zostanie do niego zrzutowany wynosi $P(|\psi\rangle \to |X\rangle) = |\alpha_x|^2$. Wynikiem pomiaru rejestru złożonego z n qubitów jest n-bitowe słowo, które reprezentuje dany stan bazowy.

Rozdział 3

Logika kwantowa i odwracalna

W rozdziale 2, wprowadzone zostały aksjomaty kwantowego modelu obliczeniowego. W tym rozdziale uwaga zostanie poświęcona odwracalnym przekształceniom słów bitowych, czyli tak zwanej logice odwracalnej. Przykładem takich przekształceń są operacje `NOT` i `CNOT` operujące na stanach bazowych rejestrów kwantowych. Jak się dalej okazuje, dla wielu algorytmów kwantowych istotnym elementem jest stworzenie układu, którego wyłącznym celem jest algebraiczna manipulacja stanami bazowymi.

3.1 Funkcje odwracalne

Definicja 13. *Funkcja odwracalna - Funkcja boolowska spełniająca dwa warunki:*

$$f : \mathbb{B}^n \to \mathbb{B}^n. \tag{3.1}$$

$$\forall_{x,y \in \mathbb{B}^n} \left(f(x) = f(y) \implies x = y \right). \tag{3.2}$$

Ponieważ f jest funkcją pomiędzy skończonymi, równolicznymi zbiorami, powyższe warunki są wystarczające do stwierdzenia, że taka funkcja f jest bijekcją, a także spełnia aksjomat permutacji.

Definicja 14. *Permutacja - Dowolna bijekcja p ze zbioru S na ten zbiór*

$$p \ : S \to S \tag{3.3}$$

Definicja 15. *Rozmiar funkcji odwracalnej - Liczba wymiarów przestrzeni wektorowej, na jakiej określona jest funkcja.*

$$ord(f) = n : (f : \mathbb{B}^n \to \mathbb{B}^n). \tag{3.4}$$

Konkretne funkcje f rozmiaru n będą zapisywane w uproszczonej notacji permutacji rozmiaru 2^n, korzystając z faktu, iż ich argumenty możemy oznaczać liczbami naturalnymi. na przykład dla zadanej funkcji f:

$$f(x) = \begin{cases} (0,1) = 2 & \text{dla } \ x = (0,0) = 0 \\ (1,1) = 3 & \text{dla } \ x = (0,1) = 1 \\ (1,0) = 1 & \text{dla } \ x = (1,0) = 2 \\ (0,0) = 0 & \text{dla } \ x = (1,1) = 3 \end{cases} \tag{3.5}$$

Będzie krócej zapisana jako $f = [2, 3, 1, 0]$.

Definicja 16. *$\mathcal{F}_n$ – Zbiór wszystkich funkcji odwracalnych rozmiaru n.*

Twierdzenie 2. *$|\mathcal{F}_n| = (2^n)!$ – Liczba funkcji odwracalnych rozmiaru n wynosi dokładnie $(2^n)!$.*

Dowód. Dowolna funkcja odwracalna f jest permutacją na zbiorze $\mathbb{B}^n$. Istnieje dokładnie $k!$ permutacji zbioru o rozmiarze k, rozmiar zbioru $\mathbb{B}^n$ wynosi 2^n, zatem istnieje $(2^n)!$ funkcji odwracalnych o rozmiarze n. □

Twierdzenie 3. *Złożenie dwóch funkcji odwracalnych h rozmiaru n jest funkcją odwracalną rozmiaru n:*

$$\forall_{f,g \in \mathcal{F}_n} (h = f \circ g \ \in \mathcal{F}_n) \tag{3.6}$$

Dowód. Złożenie dwóch permutacji jest permutacją. □

Twierdzenie 4. *Dla każdej funkcji odwracalnej f istnieje taka funkcja f^{-1}, nazywana też funkcją odwrotną, dla której zachodzi:*

$$\forall_{x \in \mathbb{B}^n} \left(f(f^{-1}(x)) = f^{-}1(f(x)) = x \right) \tag{3.7}$$

Dowód. Dla każdej permutacji istnieje permutacja odwrotna, taka że ich złożenie jest identycznością. □

Definicja 17. *Inwolucja – funkcja odwracalna f, taka że jest równa swojej funkcji odwrotnej, tj. $f = f^{-}1$.*

Twierdzenie 5. *Operacje odwracalne rozmiaru n z operacją złożenia tworzą grupę.*

Dowód 1. *Ponieważ funkcje odwracalne są równoważne z permutacjami, to grupa ta będzie izomorficzna z grupą permutacji 2^n elementów – S_{2^n}.*

Definicja 18. *Linie bitowe, linie logiczne (z j. ang. bit lines) – Indeksy współrzędnych wejściowych i wyjściowych funkcji odwracalnej. funkcja odwracalna rozmiaru n posiada n linii logicznych o indeksach ze zbioru* $\{0, 1, \ldots, n-1\}$.

3.2 Logiczne bramki odwracalne

Definicja 19. *Logiczna bramka odwracalna (z j. ang. reversible gate) - Dowolny niepodzielny element realizujący funkcję odwracalną. Argumenty tej funkcji są nazywane wejściami bramki, a wartości jej wyjściami. Rozmiar bramki to liczba wejść oraz wyjść danej bramki.*

3.2.1 Bramki `NCT` i `MCT`

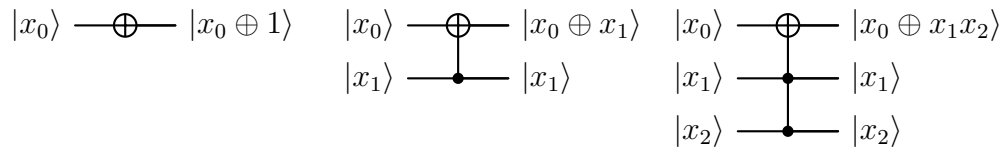
Szczególną rodziną bramek kwantowych, wykorzystywanych do konstrukcji układów logicznych, jest rodzina `NCT` *(z j. ang. NOT, CNOT, Toffoli)*, a także `MCT` *(z j. ang. multiple-control Toffoli)*.

Pierwszy zbiór składa się z trzech bramek:

1. `NOT` – Negacja bezwarunkowa. Bramka jedno-bitowa, zmienia stan bitu, na którym operuje na przeciwny.

2. `CNOT` – Negacja kontrolowana. Bramka dwu-bitowa, operuje na jednym bicie docelowym oraz jednym bicie kontrolnym *(z j. ang. target bit, control bit)*. Negacja zachodzi na bicie docelowym, wtedy i tylko wtedy gdy stan bitu kontrolnego wynosi 1, bit kontrolny zawsze pozostaje bez zmian.

3. `Toffoli` – Bramka Toffoli-ego [1]. Bramka trzy-bitowa, działa analogicznie do bramki `CNOT`, z tą różnicą, że operuje jednocześnie na dwóch bitach kontrolnych. Do negacji na bicie docelowym zachodzi wyłącznie kiedy oba bity kontrolne są w stanie 1. Bity kontrolne pozostają bez zmian.



(a) Bramka `NOT` (b) Bramka `CNOT` (c) Bramka `Toffoli`

Rysunek 3.1: Rodzina bramek `NCT`

Funkcje realizowane przez te bramki mają szczególną postać - dla każdego wejścia tej bramki, maksymalnie jedna współrzędna zostaje zmieniona na wartość przeciwną (zanegowana). Ponadto funkcje te są inwolucjami.

Na rysunku 3.1 bramki kolejno realizują funkcje $f_a = [1, 0]$, $f_b = [0, 1, 3, 2]$, $f_c = [1, 2, 3, 4, 5, 6, 8, 7]$.

Rozszerzeniem rodziny bramek `NCT` jest rodzina `MCT`. Poza trzema wskazanymi wcześniej bramkami, w tym zbiorze zawierają się bramki realizujące negację kontrolowaną o dowolnej liczbie bitów kontrolujących. Podobnie jak w przypadku bramki `CNOT` i `Toffoli`, tutaj też bit docelowy zmienia stan na przeciwny wyłącznie gdy wszystkie bity kontrolne są w stanie 1.

W ogólnym przypadku bramka `MCT` rozmiaru n, posiadająca $n-1$ linii kontrolnych o indeksach $C = \{c_0, c_1, \ldots c_{n-2}\}$ i jedną linię docelową różną od linii kontrolnych $t \notin C$, realizuje funkcję, taką że bit wyjściowy na linii docelowej y_t jest opisany równaniem:

$$y_t = x_t \oplus (x_{c_0} x_{c_1} \ldots x_{c_{n-2}}). \tag{3.8}$$

A pozostałe bity wyjściowe $y_i : i \neq t$:

$$y_i = x_i. \tag{3.9}$$

3.3 Odwracalne układy logiczne

Odwracalnym układem logicznym, o szerokości n i głębokości d będziemy nazywać:

- L - Zbiór n linii bitowych posiadających indeksy $\{0, 1, \ldots, n-1\}$.
- G - Ciąg d bramek rozmiaru $k \leq n$, wraz z ciągiem linii bitowych, na których operuje.

$$G = [g_0, g_1, \ldots, g_{d-1}]. \tag{3.10}$$

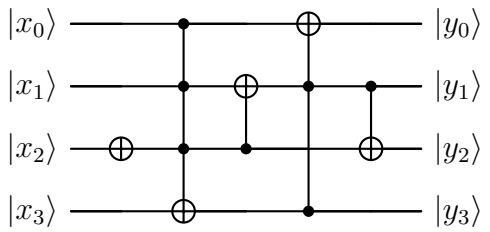
W szczególności, w przypadku układów złożonych z bramek rodziny `NCT` lub `MCT` każdą bramkę można opisać jako krotkę złożoną z kolejno indeksu linii docelowej i uporządkowanych rosnąco indeksów linii kontrolnych:

$$g_i = (t_i, c_{0,i}, c_{1,i}, \ldots, c_{k-2,i}). \tag{3.11}$$

np. bramkę z rysunku 3.1c można zapisać jako $g_0 = (t_0, c_{0,0}, c_{1,0}) = (0, 1, 2)$.

Przykładowy odwracalny układ logiczny o szerokości 4 i głębokości 5.

- $L = \{0, 1, 2, 3\}$
- $G = [g_0 = (2), g_1 = (3, 0, 1, 2), g_2 = (1, 2), g_3 = (0, 1, 3), g_4 = (2, 1)]$



Rysunek 3.2: Przykładowy odwracalny układ logiczny

Należy zwrócić uwagę, że bramka w zależności od tego, na jakich liniach bitowych układu zostanie umieszczona, de facto realizuje różne funkcje odwracalne. Dodatkowo, rozmiar funkcji, którą realizuje, też może się zmienić, jeżeli zostanie ona umieszczona w układzie o większej szerokości niż rozmiar bramki. Np. bramka `CNOT`, na rysunku 3.1b realizuje funkcję $[0, 1, 3, 2]$, tak na rysunku 3.2 realizuje kolejno dwie funkcje:

- $g_2 : [0, 1, 2, 3, 6, 7, 4, 5, 8, 9, 10, 11, 14, 15, 12, 13]$
- $g_4 : [0, 1, 6, 7, 4, 5, 2, 2, 8, 9, 14, 15, 12, 13, 10, 11]$

Jeżeli kolejne bramki g_i realizują kolejne funkcje odwracalne f_i, to cały układ realizuje funkcję:

$$f \ = f_{d-1} \circ f_{d-2} \circ \cdots \circ f_1 \circ f_0 \tag{3.12}$$

Istotną różnicą między częściej spotykanymi (nieodwracalnymi) układami logicznymi jest to, że układ musi posiadać dokładnie tyle samo wejść i wyjść. Dodatkowo, wejścia pierwotne oraz wyjścia z bramek nie mogą zostać rozgałęzione i stać się wejściami do wielu bramek. Układy te również nie mogą być sekwencyjne i są w całości kombinacyjne, nie mogą zatem zawierać sprzężeń zwrotnych.

Twierdzenie 6. *Układ realizujący funkcję odwrotną do zadanego układu złożonego z bramek* `MCT` *można utworzyć przez odwrócenie kolejności jego bramek.*

Dowód. Bramki `MCT` realizują funkcje, które są inwolucjami tj. $f_i \circ f_i = id$. Układ będący złożeniem oryginalnego ciągu bramek i ciągu odwróconego realizuje funkcję opisaną wzorem:

$$\begin{aligned} f_0 \circ f_1 \circ \cdots \circ f_{d-2} \circ (f_{d-1} \circ f_{d-1}) \circ f_{d-2} \circ \cdots \circ f_1 \circ f_0 = \\ f_0 \circ f_1 \circ \cdots \circ f_{d-2} \circ id \ \circ f_{d-2} \circ \cdots \circ f_1 \circ f_0 = \\ f_0 \circ f_1 \circ \cdots \circ (f_{d-2} \circ f_{d-2}) \circ \cdots \circ f_1 \circ f_0 = \\ ... \\ = id \end{aligned} \tag{3.13}$$

Zatem cały układ realizuje funkcję identyczności, a co z tego wynika, dołączony układ złożony z bramek w odwróconej kolejności realizuje funkcję odwrotną. □

3.4 Operacje `ARX`

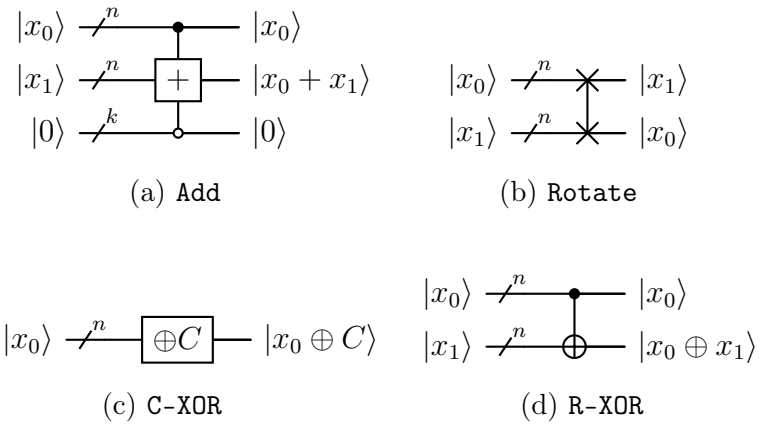
Istotnym zestawem operacji, które często są używane do projektowania szyfrów, są operacje `ARX` – *(z j. ang. Add, Rotate, XOR)*. Operacje te mogą zostać zrealizowane w kwantowym i odwracalnym modelu obliczeniowym.

- `Add` – Dodawanie modulo 2^n – Operacja nieliniowa, przekształca dwa słowa n-bitowe w miejscu zgodnie ze wzorem $(x, y) \rightarrow (x, y \ + x \, mod \, 2^n)$.
- `Rotate` – Przesunięcia i rotacje bitowe – W kwantowym modelu obliczeniowym operacje te mogą być uznane za darmowe. Nie wymagają one użycia żadnych bramek logicznych i mogą zostać wykonane wyłącznie poprzez zmianę indek-

sów qubitów w trakcie etapu przygotowania układu, o ile permutacja indeksów jest znana w trakcie przygotowania układu. Aby skorzystać z tej własności trzeba jednocześnie nadać qubitom indeksy fizyczne i logiczne.

- `XOR` – Alternatywa wykluczająca może zostać użyta w dwóch przypadkach:
  - `C-XOR` – Dodanie wartości stałej c. Operacja przekształca słowo bitowe $(x) \to (x \oplus c)$. Jeżeli dodawana wartość jest stała lub znana w trakcie przygotowywania układu kwantowego, operacja jest realizowana przy użyciu bramek `NOT`.
  - `R-XOR` – Dodanie dwóch rejestrów w miejscu zgodnie z przekształceniem $(x, y) \to (x, x \oplus y)$, w tej sytuacji operacja jest realizowana przy użyciu bramek `CNOT`.

  Niezależnie od przypadku, obie operacje typu `XOR` wymagają użycia liczby bramek równej szerokości rejestrów, na których są wykonywane. Dodatkowo, wszystkie bramki mogą zostać zrealizowane równolegle i w jednej warstwie układu z racji, że operują na rozłącznych zbiorach qubitów;



(a) `Add` (b) `Rotate`

(c) `C-XOR` (d) `R-XOR`

Rysunek 3.3: Bramki reprezentujące operacje `ARX`

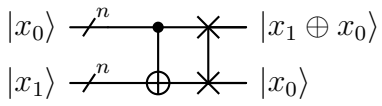
Dla bramki na rysunku 3.3a reprezentującej dodawanie, najniższy rejestr rozpoczynający i kończący w stanie $|0\rangle$ jest rejestrem qubitów pomocniczych. Symbol białej kropki oznacza użycie rejestru jako pomocniczego przez daną bramkę oraz przywrócenie jego stanu do wartości $|0\rangle$.

3.4.1 Zamiana Feistela

Operacja zamiany Feistela, pierwszy raz opisana w patencie szyfru `Lucifer` [62], jest przekształceniem złożonym z dwóch operacji `ARX` - rotacji oraz `C-XOR`. Zamiana Feistela na rejestrze $x = x_0||x_1$ złożonym z $2n$ bitów opisana jest równaniem:

$$FS(x_0||x_1) = (x_1||(x_1 \oplus x_0)) \tag{3.14}$$

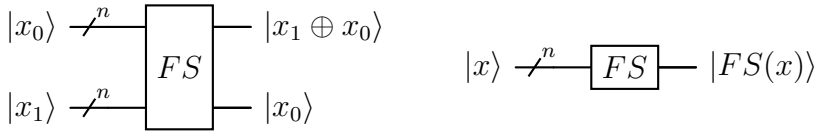
Rysunek 3.4 przedstawia implementację tego przekształcenia.



Rysunek 3.4: Układ implementujący zamianę Feistela

Koszt tego przekształcenia wynosi n bramek `CNOT`, dla rejestru złożonego z dwóch połówek o rozmiarze n.

Blok zamiany Feistela będzie używany w dwóch wariantach, w drugim wariancie należy domniemać, że przekształcenie operuje na połówkach rejestru wejściowego.



Rysunek 3.5: Blok zamiany Feistela

3.5 Metryki układu

Poszczególne układy logiki odwracalnej mogą być mierzone różnymi miarami. Dla układów z rodziny `NCT` i `MCT` najistotniejsze z nich to:

- `IO` – Rozmiar wejścia i wyjścia – Liczba qubitów, na których układ operuje logicznie i są elementem przekształcenia. Jeżeli układ jest parametryczny, to zależnie od argumentów liczba ta może się zmieniać, dlatego też w tym przypadku jest to liczba qubitów, dla których istnieje taki parametr, że układ z tym parametrem operuje na nich. Przykładem jest operacja `C-XOR` na rysunku 3.3c, w tym przypadku rozmiarem wejścia jest rozmiar dodawanego słowa bitowego, a nie jego waga Hamminga, która efektywnie będzie dyktować liczbę przekształconych qubitów.

- `ANC` – Liczba qubitów dodatkowych *(z j. ang. ancillary qubits)* – Liczba qubitów, które logicznie nie są elementem przekształcenia, jednak są wymagane do zrealizowania wybranej implementacji operacji. Stan tych qubitów musi wynosić $|0\rangle$ przed i po zrealizowaniu przekształcenia. Qubity należące do wejścia nie są qubitami dodatkowymi.

- `W` – Szerokość – Liczba wszystkich qubitów, które zmieniają lub mogą zmienić stan. Efektywnie jest to suma rozmiaru wejścia i liczby qubitów dodatkowych:

$$\texttt{W} = \texttt{IO} + \texttt{ANC}$$

- `#G` – Liczba bramek w układzie:
  - `#NOT` – Liczba bramek `NOT`;
  - `#CNOT` – Liczba bramek `CNOT`;
  - `#Toff` – Liczba bramek `Toffoli`;
  - `#Toff-n` – Liczba bramek `Toffoli-n`, czyli poszerzonych bramek Toffoliego z `n` qubitami kontrolnymi;

- `D` – Głębokość układu – Liczba warstw układu, w których znajdują się bramki, które mogą zostać wykonane równolegle. Jeżeli wszystkie bramki są realizowane bez zrównoleglenia, to miara ta jest równa liczbie bramek `W`.

Tabela 3.1 przedstawia przykład metryk dla operacji 3.3c oraz 3.3d.

Operacja	IO	ANC	W	#G	#NOT	#CNOT	#Toff	D
C-XOR	n	0	n	n	n	0	0	1
R-XOR	$2n$	0	n	n	0	n	0	1
Add*	$2n$	$2n$	$4n$	$16n$	$2n$	$4n$	$10n$	
		$-2k$	$-2k$	$-12k$			$-12k$	$4k$
		-1	-1	-13	-2	-5	-6	$+2$

*Przy założeniu, że $n = 2^k, n \in \{8, 16, 32, 64, \dots\}$.

Tabela 3.1: Koszt operacji ARX

Wzory opisujące miary układu dodawania Add zachodzą przy dodawaniu rejestrów o rozmiarach nie mniejszych niż 8 i jednocześnie będącymi potęgami liczby 2. Wzory zostały zaczerpnięte z pracy T. Drapera [63], w której opisana jest implementacja sumatorów w reżimie logiki odwracalnej o logarytmicznej głębokości.

3.6 Qubity pomocnicze

Do tej pory rozmiar funkcji i szerokość układu traktowane były jeden do jednego. Okazuje się jednak, że w pewnych sytuacjach korzystne może być realizowanie funkcji odwracalnej rozmiaru $n \times n$ przez większy układ. Na przykład w celu zmniejszenia liczby bramek potrzebnych do zaimplementowania funkcji.

Pierwszym krokiem jest modyfikacja tablicy prawdy funkcji, dla której układ ma zostać utworzony. Przykładowo, niech zadana będzie funkcja 2×2:

x_1	x_0	y_1	y_0
0	0	0	1
0	1	1	0
1	0	0	0
1	1	1	1

Tabela 3.2: Tablica prawdy przykładowej funkcji odwracalnej 2×2.

Do podanej funkcji należy dodać dodatkowy qubit, na którym układ będzie mógł wykonać operacje pomocnicze. Rozszerzenie zmiennej o dodatkową linię, która nie jest modyfikowana ($y_2 = x_2$) skutkuje otrzymaniem następującej tablicy prawdy:

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

Tabela 3.3: Rozszerzenie przykładowej tablicy prawdy

Niestety, takie rozszerzenie nie zwiększa znacząco przestrzeni rozwiązań, co może skutkować sytuacją, w której mniejszy układ nie zostanie znaleziony. Aby dodatkowo zwiększyć przestrzeń rozwiązań, należy założyć, że qubit pomocniczy znajduje się w znanym stanie i nie jest wymagane, aby dla obu stanów wejściowych qubitu pomocniczego pozostałe dwa wyjścia implementowały żądaną funkcję. Bez straty ogólności można przyjąć, że qubit pomocniczy zawsze znajduje się na początku w stanie 0 i na końcu też znajduje się w tym stanie. Zapewnia to możliwość ponownego użycia go przez inne układy i w przypadku obliczeń kwantowych powoduje, że qubit pomocniczy nie jest splątany z zasadniczą częścią rejestru. Używając stanów obojętnych, tablica prawdy dla tak zmodyfikowanej funkcji wygląda następująco:

x_2	x_1	x_0	y_2	y_1	y_0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	0	1	1
1	0	0	1	**x**	**x**
1	0	1	1	**x**	**x**
1	1	0	1	**x**	**x**
1	1	1	1	**x**	**x**

Tabela 3.4: Tablica prawdy funkcji z bitem pomocniczym

Porównując tabele 3.2 i 3.4, można zauważyć, że w tej drugiej, pod wartości obojętne można wstawić dowolną funkcję odwracalną rozmiaru 2×2. W efekcie, istnieje $2^2! = 24$ funkcji o rozmiarze 3×3, które przy zadanych założeniach implementują żądaną funkcję rozmiaru 2×2. Rozszerzając zbiór dopuszczalnych funkcji, może skutkować znalezieniem mniejszego układu, który implementuje żądaną funkcję.

3.7 Synteza układów

Syntezą układu nazywana będzie dowolna procedura, która pozwala na odnalezienie układu, który realizuje wskazaną funkcję odwracalną. Naturalnie, nasuwają się dwa następujące pytania:

1. Czy zawsze jest możliwe znalezienie takiego układu?
2. Jeżeli jest to możliwe, to jaki jest jego najmniejszy możliwy rozmiar?

Pierwsze pytanie definitywnie zostało rozstrzygnięte dla układów `MCT` w roku 2003 w pracy D. Millera [64]. Praca prezentuje pierwszy algorytm syntezy układów przy użyciu bramek `MCT`, dowiedzione zostało również, że algorytm jest kompletny i dla dowolnej funkcji zwraca układ. Dodatkowo, dla funkcji odwracalnej rozmiaru $n \times n$ układ ten ma zawsze nie więcej niż $(n-1) * 2^n$ bramek.

W przypadku układów `NCT`, problem jest trudniejszy. Każda bramka `MCT`, a w tym `NCT` rozmiaru n mniejszego niż szerokość układu, w którym się znajduje, zawsze realizuje permutację parzystą, co zostało udowodnione w pracy T. Toffoliego [1]. Efektem tego jest brak możliwości syntezy funkcji nieparzystych rozmiaru większego niż 3 przez bramki wyłącznie z rodziny `NCT`.

Okazuje się jednak, że istnieje możliwość ominięcia tego problemu przy użyciu qubitów dodatkowych, które zostały opisane w rozdziale 3.6. W pracy [65] przedstawiono algorytm syntezy dowolnej funkcji odwracalnej przy użyciu maksymalnie jednego qubitu pomocniczego i wyłącznie bramek z rodziny `NCT`.

Konstruktywne algorytmy syntezy układów zostały przedstawione i rozwinięte w następujących pracach:

- M. Asma Taheri (2024) *"Advanced Methods in Quantum and Reversible Circuit Synthesis for Boolean and Multivalued Logic"* [66] - Praca doktorska poświęcona syntezie układów reprezentujących multipleksery i demultipleksery;
- E. Younis, C. Iancu (2022) *"Quantum Circuit Optimization and Transpilation via Parameterized Circuit Instantiation"* [67] - artykuł poświęcony syntezie parametryzowalnych układów kwantowych;
- M. Davis et. al. (2020) *"Towards Optimal Topology Aware Quantum Circuit Synthesis"* [68] - artykuł poświęcony syntezie układów do bramek komputerów kwantowych klasy `NISQ`;

- O. Matteo, M. Mosca (2016) *"Parallelizing quantum circuit synthesis"* [69] - artykuł opisuje procedurę zrównoleglania wybranych algorytmów syntezy układów;
- P. Gupta et. al. (2006) *"An Algorithm for Synthesis of Reversible Logic Circuits"* [70] - artykuł prezentuje polepszone wyniki zawarte w oryginalnej pracy D. Millera [64];
- Prace przeglądowe:
  - G. Yan et. al. (2024) *"Quantum Circuit Synthesis and Compilation Optimization: Overview and Prospects"* [71];
  - M. Seedi, I. Markov (2013) *"Synthesis and optimization of reversible circuits - a survey"* [72].

3.7.1 Optymalizacja

Wiedząc, które funkcje mogą zostać zsyntezowane i w jakich warunkach, kolejnym problemem jest optymalizacja już znalezionych układów.

Najprostszą metodą optymalizacji układów jest poszukiwanie i aplikowanie reguł podstawieniowych. Każda reguła składa się z dwóch układów, które realizują tę samą funkcję, przy czym jeden jest większy od drugiego. Następnie, w układzie, który ma zostać zoptymalizowany, poszukiwane są wystąpienia większego układu z każdej reguły, które następnie zastępowane są na mniejsze. Wybór, które reguły podstawieniowe mają zostać zastosowane oraz w których miejscach odbywa się heurystycznie.

Istotnym etapem tych metod jest faza przygotowania reguł podstawieniowych. Najczęściej używaną metodą odnajdowania podstawień jest poszukiwanie tzw. szablonów identyczności *(z j. ang. identity templates)*.

Definicja 20. *Szablon identyczności - Dowolny układ odwracalny, który realizuje funkcję identyczności.*

Okazuje się, że dowolny taki układ może zostać użyty do wygenerowania wielu nowych reguł. Odbywa się to poprzez przecięcie szablonu na dwie części różnych rozmiarów. Ponieważ cały układ realizuje funkcję identyczności, to pomniejsze części realizują tę samą funkcję z dokładnością do odwrotności, zatem tworzą regułę podstawieniową.

Procedury optymalizacji układów w oparciu o mechanizmy reguł podstawieniowych i szablonów identyczności zostały zbadane i opisane w pracach [64][73][74][75][76].

3.7.2 Synteza optymalna

Najtrudniejszym zadaniem, z kategorii syntezy, jest synteza optymalna. Jej celem jest znajdowanie układów o najmniejszych możliwych rozmiarach i dowodzenie ich optymalności.

W tym celu mogą zostać użyte metody zautomatyzowanego dowodzenia, np. `SAT-solvery` lub `SMT-solvery`. Procedury oparte o metody formalne w ogólnej postaci sprowadzają się do sformułowania pytania "Czy istnieje układ złożony z `G` bramek, który implementuje funkcję f".

Posiadając metodę formułowania takiego zadania w języku odpowiednim dla systemu zautomatyzowanego dowodzenia, można rozpocząć syntezę optymalną, zaczynając od zapytania o układ złożony z jednej bramki, następnie z dwóch itd. W momencie gdy otrzymana zostanie pierwsza twierdząca odpowiedź, pewne jest, że znaleziony układ posiada rozmiar optymalny, ponieważ dla wszystkich poprzednich dowiedziono, że układ o tym rozmiarze nie istnieje.

Podejście to zostało opisane po raz pierwszy w roku 2007 w pracy R. Wille [77]. Praca prezentuje pierwszą metodę modelowania zapytania o układ złożony z `G` bramek `MCT` do problemu spełnialności.

Praktyczne zastosowanie metod syntezy optymalnej osiągalne jest dla funkcji nielinowych rozmiarów nie większych niż 5×5. W przypadku funkcji linowych istnieje możliwość dokonania syntezy optymalnej dla większej liczby wejść. Konstruktywny algorytm syntezy optymalnej odwracalnych funkcji linowych, nieoparty o modelowanie do problemu SAT, został przedstawiony w pracy K. Patela (2008) [78].

W pracy G. Meuli (2018) [79] przedstawiona została metoda syntezy optymalnej kwantowych układów logicznych z użyciem zestawu bramek *Clifford + T*, pokazuje to, że może zostać dokonana nie tylko dla zestawu bramek `NCT` i `MCT`.

3.7.3 Synteza kompletna

W przeciwieństwie do poprzednich problemów dotyczących syntezy, problem syntezy kompletnej polega na znalezieniu optymalnych implementacji całej klasy układów.

Badanie D. Maslova [80] przedstawia wyniki syntezy optymalnej wszystkich funkcji rozmiaru 3×3, w pracy O. Golubitskiego [81] zostały odnalezione optymalne implementacje wszystkich funkcji rozmiaru 4×4. Obie prace przedstawiają rozkład optymalnych rozkładów implementacji dla wszystkich funkcji odwracalnych wskazanych rozmiarów.

Obie prace korzystają z algorytmu grafowego `BFS` do odnalezienia wszystkich implementacji. Zamiast poszukiwać każdej implementacji z osobna, algorytm w oparciu o znajomość układów optymalnych rozmiaru `G` i mniejszych odnajduje układy optymalne rozmiaru `G+1`, następnie `G+2` itd. aż do znalezienia implementacji dla wszystkich funkcji.

3.7.4 Alternatywne metody syntezy

Poza algorytmami konstruktywnymi i opartymi o dowodzenie zautomatyzowane, w ostatnich latach pojawiły się badania na temat wykorzystania szeroko pojętego uczenia maszynowego.

Badanie przeprowadzone w pracy M. Ostaszewskiego (2021) [82]. oraz T. Fosela (2021) [83] pokazują, że możliwe jest wykorzystanie uczenia ze wzmacnianiem *(z j. ang. Reinforcment Learning)* do syntezy i optymalizacji układów. Natomiast w pracy F. Furruttera (2024) [84] użyte zostały modele dyfuzji *(z j. ang. Diffusion Models)* do wytwarzania kwantowych układów logicznych.

Potencjalną, ale jeszcze nieszczególnie rozwiniętą ścieżką rozwoju algorytmów syntezy są algorytmy genetyczne i ewolucyjne. Praca L. Martina (2003) pokazała pierwszy przykład wykorzystania tej rodziny algorytmów.

Rozdział 4

Algorytm Grovera

Definicja 21. *Problem spełnialności - Zadana jest funkcja boolowska* $f : \mathbb{B}^n \to \mathbb{B}$*, o nieznanej strukturze. Mając dostęp do wyroczni – czarnej skrzynki (z j. ang. oracle, black-box), która realizuje tę funkcję, należy znaleźć dowolną wartość wektora (rozwiązanie)* x_0*, taką, że* $f(x_0) = 1$*, lub stwierdzić, że taki wektor nie istnieje w przypadku, gdy istotnie nie ma takiej wartości.*

Na potrzeby pracy przyjęte zostały następujące oznaczenia:

- $N = 2^n = |\mathbb{B}^n|$ – liczba wszystkich wartościowań (argumentów) funkcji f;
- $T \subset \mathbb{B}^n, T = \{x : x \in \mathbb{B}^n \wedge f(x) = 1\}$ – zbiór wszystkich rozwiązań;
- $F \subset \mathbb{B}^n, F = \{x : x \in \mathbb{B}^n \wedge f(x) = 0\}$ – zbiór wszystkich nierozwiązań;
- $t = |T| \leq N$ – liczba rozwiązań;
- $f = |F| \leq N$ – liczba nierozwiązań;
- $r_t = \frac{t}{N}$ – gęstość rozwiązań, stosunek l. rozwiązań do l. wartościowań;
- $r_f = \frac{f}{N}$ – gęstość nierozwiązań.

Z powyższych definicji zachodzą:

- $(F \cup T \ = \mathbb{B}^n) \wedge (F \cap T \ = \phi)$;
- $t + f = N$;
- $r_f + r_t = 1$.

W klasycznym modelu obliczeniowym, aby rozwiązać problem spełnialności, w najgorszym przypadku należy wybrać i sprawdzić $f+1$ różnych wartościowań, aby znaleźć przynajmniej jedno rozwiązanie. w szczególności, jeżeli $t \in \{0, 1\}$, to należy dokonać pełnego przeglądu, czyli dokonać $N = 2^n$ sprawdzeń, aby mieć pewność

odnalezienia rozwiązania lub stwierdzić, że takowe nie istnieją. Oznacza to, że liczba zapytań wyroczni rośnie wykładniczo wraz ze wzrostem liczby zmiennych wejściowych funkcji, nawet przy stałej gęstości rozwiązań. Zadanie to można efektywniej rozwiązać przy użyciu kwantowego systemu obliczeniowego, wykorzystując algorytm Grovera, pierwotnie opisany w pracy Lova K. Grovera [85].

4.1 Etapy algorytmu

Algorytm rozpoczyna się od przygotowania rejestru kwantowego długości n w stanie $|\psi_0\rangle = |0\rangle$. Następnie wprowadza się go w stan jednolitej superpozycji, aplikując przekształcenie Hadamarda na każdy qubit:

$$|\psi_1\rangle = H^{\otimes n} |0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \,. \tag{4.1}$$

Stan ten nazywany jest superpozycją jednolitą *(z j. ang. uniform superposition)*, ponieważ wszystkie stany bazowe mają dokładnie ten sam współczynnik.

Kolejny etap zakłada, że istnieje operator kodowania fazowego funkcji f, który działa w następujący sposób dla stanów bazowych:

$$V_f |x\rangle = \begin{cases} -|x\rangle & \text{dla } \; f(x) = 1 \\ +|x\rangle & \text{dla } \; f(x) = 0 \end{cases} = (-1)^{f(x)} |x\rangle \,. \tag{4.2}$$

Operator ten jest swego rodzaju kwantowym odpowiednikiem wyroczni implementującej funkcję f. Jak się okazuje, operator taki zawsze istnieje i jest możliwy do skonstruowania, a co więcej, zawsze jest możliwe skonstruowanie go z użyciem rodziny bramek `NCT` lub `MCT`. Niemniej jednak, konstrukcja takiego operatora z dostępnych bramek logicznych nie jest problemem prostym. Dodatkową warstwą tego problemu jest potrzeba minimalizacji rozmiaru takiego układu. Metody konstruowania i optymalizacji operatorów kodowania są przedmiotem wielu badań, w szczególności główny obszar badawczy tej pracy jest poświęcony temu zagadnieniu w kontekście analizy prymitywów kryptograficznych.

Do otrzymanego stanu $|\psi_1\rangle$ aplikowany jest operator kodowania fazowego zadanej funkcji boolowskiej f:

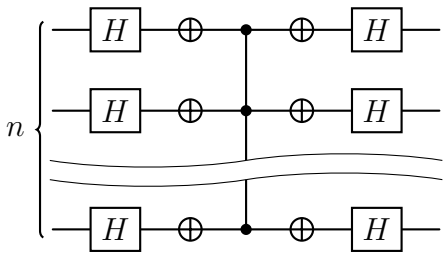
$$\begin{aligned} |\psi_2\rangle = V_f\,|\psi_1\rangle = V_f\left(\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle\right) = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1} V_f\,|x\rangle = \\ = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}\,|x\rangle\,. \end{aligned} \tag{4.3}$$

Otrzymany stan ma szczególną postać – wszystkie stany bazowe mają te same rzeczywiste współczynniki z dokładnością do znaku. **Stany, które są rozwiązaniami mają współczynniki rzeczywiste mniejsze od zera, a pozostałe – rzeczywiste większe od zera.** Kolejnym etapem jest wzmocnienie amplitud stanów będących rozwiązaniami.

Definicja 22. *Operator dyfuzji Grovera G_n – Operator kwantowy realizowany na n qubitach przedstawiony macierzą o rozmiarze $2^n \times 2^n$:*

$$\begin{aligned} G_n &= \begin{bmatrix} \frac{2}{2^n}-1 & \frac{2}{2^n} & \cdots & \frac{2}{2^n} \\ \frac{2}{2^n} & \frac{2}{2^n}-1 & \cdots & \frac{2}{2^n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{2^n} & \frac{2}{2^n} & \cdots & \frac{2}{2^n}-1 \end{bmatrix} = \\ &= \frac{2}{2^n}\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \end{aligned} \tag{4.4}$$

Operator Grovera G_n zaimplementować można następującym układem:



Rysunek 4.1: Implementacja operatora dyfuzji Grovera

Operator ten można zapisać jako $G_n = \frac{2}{2^n}\mathbf{1}_{2^n} - I_{2^n}$, gdzie macierz $\mathbf{1}_{2^n}$ to macierz rozmiaru $2^n \times 2^n$ złożona z samych jedynek. Pomnożenie wektora z bazy kanonicznej przez tę macierz daje następujący wynik:

$$\mathbf{1}_{2^n}|x_0\rangle = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \sum_{x=0}^{2^n-1} |x\rangle = |\beta\rangle \tag{4.5}$$

Jak widać, wynik $|\beta\rangle$ jest niezależny od wyboru wektora bazowego $|x_0\rangle$. Pomnożenie dowolnego wektora $|\psi\rangle$ skutkuje następującym wynikiem:

$$\begin{aligned} \mathbf{1}_{2^n}|\psi\rangle = \mathbf{1}_{2^n}\sum_{x=0}^{2^n-1}\alpha_x|x\rangle = \sum_{x=0}^{2^n-1}\alpha_x\left(\mathbf{1}_{2^n}|x\rangle\right) = \\ = \sum_{x=0}^{2^n-1}\alpha_x|\beta\rangle = \left(\sum_{x=0}^{2^n-1}\alpha_x\right)|\beta\rangle = \sigma|\beta\rangle \end{aligned} \tag{4.6}$$

Gdzie σ jest sumą wszystkich współczynników wektora $|\psi\rangle$. Opierając się na tym, działanie operatora G_n na dowolny stan $|\psi\rangle = \sum_{x=0}^{2^n-1}\alpha_x|x\rangle$ można opisać następującym równaniem:

$$\begin{aligned} G_n\sum_{x=0}^{2^n-1}\alpha_x|i\rangle = \left(\frac{2}{2^n}\mathbf{1}_{2^n} - I_{2^n}\right)\sum_{x=0}^{2^n-1}\alpha_x|x\rangle = \frac{2}{2^n}\mathbf{1}_{2^n}\sum_{x=0}^{2^n-1}a_x|x\rangle - I_{2^n}\sum_{x=0}^{2^n-1}a_x|x\rangle = \\ = \sum_{x=0}^{2^n-1}\frac{2}{2^n}s|x\rangle - \sum_{x=0}^{2^n-1}\alpha_x|x\rangle = \sum_{x=0}^{2^n-1}2\mu|x\rangle - \sum_{x=0}^{2^n-1}a_x|x\rangle = \\ = \sum_{x=0}^{2^n-1}(2\mu - \alpha_x)|x\rangle \end{aligned} \tag{4.7}$$

Gdzie wartość $\mu = \frac{\sigma}{2^n}$ jest średnią arytmetyczną współczynników wektora $|\psi\rangle$. Przyjmując, że funkcja ma kilka rozwiązań, oraz $t << 2^n$, można stwierdzić, że μ jest mocno zbliżone do $\frac{1}{\sqrt{2^n}}$.

Po przekształceniu, wartość współczynników w zależności od wartości funkcji f dla danego wartościowania wynosi:

$$(2\mu - \alpha_x) \approx \begin{cases} 2 * \frac{1}{\sqrt{2^n}} - \frac{1}{\sqrt{2^n}} = \frac{1}{\sqrt{2^n}} & \text{dla } \ f(x) = 0 \\ 2 * \frac{1}{\sqrt{2^n}} + \frac{1}{\sqrt{2^n}} = \frac{3}{\sqrt{2^n}} & \text{dla } \ f(x) = 1 \end{cases} \tag{4.8}$$

Operator ten zatem będzie zwiększał amplitudę dla stanów o ujemnych współczynnikach, czyli rozwiązań. Dla pozostałych wartościowań amplituda ta zmaleje.

Pomimo że wskazana para operatorów pozwala na nieznaczne wzmocnienie amplitudy stanów będących rozwiązaniami, to wzmocnienie to nie jest wystarczająco wysokie. Prawdopodobieństwo pomiaru wskazanego stanu będącego rozwiązaniem wynosi w przybliżeniu:

$$P(x = x_0) \approx \left| \frac{3}{\sqrt{2^n}} \right|^2 = \frac{9}{2^n} \tag{4.9}$$

Natomiast prawdopodobieństwo zmierzenia dowolnego z t rozwiązań jest nie większe niż $\frac{9t}{2^n}$. Ponieważ założono, że t jest znacznie mniejsze od 2^n, to istotnie to prawdopodobieństwo jest lepsze od naiwnego losowania rozwiązań podejściem brutalnym, ale jak widać nadal jest ono pomijalne.

W celu polepszenia tego prawdopodobieństwa, procedurę kodowania i wzmacniania można powtórzyć wielokrotnie. w tym celu należy wyznaczyć funkcję prawdopodobieństwa pomiaru rozwiązania w zależności od liczby powtórzeń procedury.

Zakładając, że oba kroki procedury powtórzono i razy, poniżej zostanie zbadany wpływ wykonania $i + 1$ iteracji.

Bez straty ogólności, można przyjąć, że po i-tej iteracji procedury współczynniki stojące przy rozwiązaniach wynoszą $\epsilon_i \in R^+$, a przy nierozwiązaniach $\lambda_i \in R^+$. Stan rejestru po wykonaniu i iteracji procedury kodowania i wzmacniania wynosi:

$$|\phi_i\rangle = \lambda_i \sum_{x \in F} |x\rangle + \epsilon_i \sum_{x \in T} |x\rangle \,. \tag{4.10}$$

Kodowanie fazowe na tym stanie powoduje odbicie fazy przy nierozwiązaniach:

$$\left|\phi_i^{'}\right\rangle = \lambda_i \sum_{x \in F} |x\rangle - \epsilon_i \sum_{x \in T} |x\rangle \,. \tag{4.11}$$

Średnia arytmetyczna współczynników wynosi:

$$\mu_i = \frac{1}{N}(f\lambda_i - t\epsilon_i) = \frac{f\lambda_i}{N} - \frac{t\epsilon_i}{N} = r_f\lambda_i - r_t\epsilon_i. \tag{4.12}$$

Dokończenie iteracji $i+1$ poprzez wzmacnianie amplitud generuje nowy stan:

$$|\phi_{i+1}\rangle = \lambda_{i+1}\sum_{x\in F}|x\rangle + \epsilon_{i+1}\sum_{x\in T}|x\rangle\,. \tag{4.13}$$

Gdzie:

$$\begin{aligned}\lambda_{i+1} &= 2\mu_i - \lambda_i = 2(r_f\lambda_i - r_t\epsilon_i) - \lambda_i = (2r_f - 1)\lambda_i - 2r_t\epsilon_i = (r_f - r_t)\lambda_i - 2r_t\epsilon_i \\ \epsilon_{i+1} &= 2\mu_i + \epsilon_i = 2(r_f\lambda_i - r_t\epsilon_i) + \epsilon_i = 2r_f\lambda_i + (1 - 2r_t)\epsilon_i = 2r_f\lambda_i + (r_f - r_t)\epsilon_i\end{aligned} \tag{4.14}$$

Lub równoważnie:

$$\begin{bmatrix}\lambda_{r+1}\\ \epsilon_{i+1}\end{bmatrix} = \begin{bmatrix}r_f - r_t & -2r_t\\ 2r_f & r_f - r_t\end{bmatrix}\begin{bmatrix}\lambda_i\\ \epsilon_i\end{bmatrix}. \tag{4.15}$$

Suma amplitud po każdej iteracji i musi wynosić 1, zatem:

$$f\lambda_i^2 + t\epsilon_i^2 = 1. \tag{4.16}$$

Jest to równanie elipsy o środku $(0,0)$ postaci:

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 = 1. \tag{4.17}$$

Dla $a = \sqrt{f}^{-1}$, $b = \sqrt{t}^{-1}$ oraz współrzędnych ortogonalnych $x = \lambda_i$ i $y = \epsilon_i$. Rozmiary obu półosi elipsy a, b są niezależne od iteracji i. Korzystając z równania parametrycznego elipsy, możemy wyznaczyć punkt (λ_i, ϵ_i) dla pewnej wartości parametru - kąta θ_i.

$$\begin{bmatrix}\lambda_i\\ \epsilon_i\end{bmatrix} = \begin{bmatrix}\sqrt{f}^{-1} & 0\\ 0 & \sqrt{t}^{-1}\end{bmatrix}\begin{bmatrix}\cos\theta_i\\ \sin\theta_i\end{bmatrix} \iff \begin{bmatrix}\cos\theta_i\\ \sin\theta_i\end{bmatrix} = \begin{bmatrix}\sqrt{f} & 0\\ 0 & \sqrt{t}\end{bmatrix}\begin{bmatrix}\lambda_i\\ \epsilon_i\end{bmatrix} \tag{4.18}$$

Podstawiając pod równanie 4.18 $i := i + 1$, a następnie równanie rekurencyjne 4.14 zachodzi:

$$\begin{aligned}
\begin{bmatrix} \cos\theta_{i+1} \\ \sin\theta_{i+1} \end{bmatrix} &= \begin{bmatrix} \sqrt{f} & 0 \\ 0 & \sqrt{t} \end{bmatrix} \begin{bmatrix} \lambda_{i+1} \\ \epsilon_{i+1} \end{bmatrix} = \begin{bmatrix} \sqrt{f} & 0 \\ 0 & \sqrt{t} \end{bmatrix} \begin{bmatrix} r_f - r_t & -2r_t \\ 2r_f & r_f - r_t \end{bmatrix} \begin{bmatrix} \epsilon_i \\ \lambda_i \end{bmatrix} = \\
&= \begin{bmatrix} \sqrt{f} & 0 \\ 0 & \sqrt{t} \end{bmatrix} \begin{bmatrix} r_f - r_t & -2r_t \\ 2r_f & r_f - r_t \end{bmatrix} \begin{bmatrix} \sqrt{f}^{-1} & 0 \\ 0 & \sqrt{t}^{-1} \end{bmatrix} \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \end{bmatrix} = \\
&= \begin{bmatrix} r_f - r_t & -2r_t\sqrt{\frac{f}{t}} \\ 2r_f\sqrt{\frac{t}{f}} & r_f - r_t \end{bmatrix} \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \end{bmatrix} = \begin{bmatrix} r_f - r_t & -2r_t\sqrt{\frac{r_f}{r_t}} \\ 2r_f\sqrt{\frac{r_t}{r_f}} & r_f - r_r t \end{bmatrix} \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \end{bmatrix} = \\
&= \begin{bmatrix} r_f - r_t & -2\sqrt{r_f r_t} \\ 2\sqrt{r_f r_t} & r_f - r_t \end{bmatrix} \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \end{bmatrix}
\end{aligned} \tag{4.19}$$

Można zauważyć, że współczynniki tej macierzy na przekątnej są sobie równe, natomiast wartości na przeciwprzekątnej mają tę samą wartość bezwzględną, ale przeciwne znaki. Równanie to można zapisać w postaci:

$$\begin{aligned}
\begin{bmatrix} \cos\theta_{i+1} \\ \sin\theta_{i+1} \end{bmatrix} &= \begin{bmatrix} u & -v \\ v & u \end{bmatrix} \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \end{bmatrix} \\
u &= r_f - r_t \in [-1, 1] \\
v &= 2\sqrt{r_f r_t} \in [0, 1]
\end{aligned} \tag{4.20}$$

Należy dalej zauważyć, że $u^2 + v^2 = 1$

$$\begin{aligned}
u^2 + v^2 = (r_f - r_t)^2 + (2\sqrt{r_f r_t})^2 = r_f^2 - 2r_f r_t + r_t^2 + 4r_f r_t = \\
= r_f^2 + 2r_f r_t + r_t^2 = (r_f + r_t)^2 = 1. \qquad \square
\end{aligned} \tag{4.21}$$

Wartość u i v są zatem sinusem i cosinusem pewnej wartości $\omega \in [0, \pi]$. Co po podstawieniu daje:

$$\begin{bmatrix} \cos\theta_{i+1} \\ \sin\theta_{i+1} \end{bmatrix} = \begin{bmatrix} \cos\omega & -\sin\omega \\ \sin\omega & \cos\omega \end{bmatrix} \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \end{bmatrix} = \begin{bmatrix} \cos\theta_i\cos\omega - \sin\theta_i\sin\omega \\ \cos\theta_i\sin\omega + \sin\theta_i\cos\omega \end{bmatrix} = \\ = \begin{bmatrix} \cos(\theta_i + \omega) \\ \sin(\theta_i + \omega) \end{bmatrix} \tag{4.22}$$

Przyjmując, że przed wykonaniem pierwszej iteracji kąt wynosi θ_0 oraz wiedząc, że ω jest wartością stałą i zależną wyłącznie od gęstości rozwiązań oraz niezależną od liczby iteracji:

$$\begin{bmatrix} \cos\theta_1 \\ \sin\theta_1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_0 + \omega) \\ \sin(\theta_0 + \omega) \end{bmatrix} \\ \begin{bmatrix} \cos\theta_2 \\ \sin\theta_2 \end{bmatrix} = \begin{bmatrix} \cos(\theta_1 + \omega) \\ \sin(\theta_1 + \omega) \end{bmatrix} = \begin{bmatrix} \cos(\theta_0 + 2\omega) \\ \sin(\theta_0 + 2\omega) \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \end{bmatrix} = \begin{bmatrix} \cos(\theta_0 + i\omega) \\ \sin(\theta_0 + i\omega) \end{bmatrix} \tag{4.23}$$

Co pozwala wyznaczyć wartości współczynników rozwiązań i nierozwiązań w zależności od indeksu iteracji i:

$$\begin{bmatrix} \lambda_i \\ \epsilon_i \end{bmatrix} = \begin{bmatrix} \sqrt{f}^{-1}\cos\theta_i \\ \sqrt{t}^{-1}\sin\theta_i \end{bmatrix} = \begin{bmatrix} \sqrt{f}^{-1}\cos(\theta_0 + i\omega) \\ \sqrt{t}^{-1}\sin(\theta_0 + i\omega) \end{bmatrix} \tag{4.24}$$

Podstawiając warunki początkowe algorytmu:

$$\begin{bmatrix} \lambda_0 \\ \epsilon_0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2^n}} \\ \frac{1}{\sqrt{2^n}} \end{bmatrix} = \begin{bmatrix} \sqrt{f}^{-1}\cos\theta_0 \\ \sqrt{t}^{-1}\sin\theta_0 \end{bmatrix} \implies \begin{bmatrix} \cos\theta_0 \\ \sin\theta_0 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{f}}{\sqrt{2^n}} \\ \frac{\sqrt{t}}{\sqrt{2^n}} \end{bmatrix} = \begin{bmatrix} \sqrt{r_f} > 0 \\ \sqrt{r_t} > 0 \end{bmatrix} \implies \\ \implies \theta_0 \in \left(0, \frac{\pi}{2}\right) \tag{4.25}$$

Z definicji ω w równaniu 4.20:

$$\begin{cases} \cos\omega = r_f - r_t = \sqrt{r_f}^2 - \sqrt{r_t}^2 = (\cos\theta_0)^2 - (\sin\theta_0)^2 = \cos 2\theta_0 \\ \sin\omega = 2\sqrt{r_f r_t} = 2\sqrt{r_f}\sqrt{r_t} = 2\sin\theta_0\cos\theta_0 = \sin 2\theta_0 \end{cases} \implies \omega = 2\theta_0 \tag{4.26}$$

Podstawiając $\omega = 2\theta_0$ pod równanie stanu otrzymujemy:

$$\begin{bmatrix} \lambda_i \\ \epsilon_i \end{bmatrix} = \begin{bmatrix} \sqrt{f}^{-1}\cos\left((2i+1)\theta_0\right) \\ \sqrt{t}^{-1}\sin\left((2i+1)\theta_0\right) \end{bmatrix} \tag{4.27}$$

Ostatnim krokiem jest wyznaczenie najmniejszej wartości i, która maksymalizuje prawdopodobieństwo znalezienia rozwiązania. Funkcja prawdopodobieństwa wynosi:

$$P(i) = t * (\epsilon_i)^2 = t * \left(\sqrt{t}^{-1}\sin\left((2i+1)\theta_0\right)\right)^2 = \sin\left((2i+1)\theta_0\right)^2. \tag{4.28}$$

Funkcja ta przyjmuje wartości w $[0, 1]$. Najmniejsza wartość, dla jakiej funkcja sinus przyjmuje wartość 1, to $\frac{\pi}{2}$.

$$(2i+1)\theta_0 = \frac{\pi}{2} \implies i = -\frac{1}{2} + \frac{\pi}{4\theta_0} \approx \left\lfloor \frac{\pi}{4\theta_0} \right\rfloor. \tag{4.29}$$

Dla odpowiednio dużej liczby wartościowań i małej liczby rozwiązań, gęstość rozwiązań jest zbliżona do 0. z definicji θ_0, wiadomo, że $(\sin\theta_0)^2 = r_t \approx 0$. Można zatem przyjąć, że $\theta_0 = \sin\theta_0 = \sqrt{r_t}$

$$\theta_0 \approx \sqrt{r_t} \implies i = \left\lfloor \frac{\pi}{4\sqrt{r_t}} \right\rfloor = \left\lfloor \frac{\pi}{4\sqrt{t}} * 2^{\frac{n}{2}} \right\rfloor \tag{4.30}$$

Wyznaczona wartość i nie jest dokładnym rozwiązaniem układu, jednak dowiedziono, że dla takiego oszacowania wartości i suma amplitud rozwiązań funkcji wynosi nie mniej niż $\frac{1}{4}$, zatem prawdopodobieństwo zmierzenia wartości, która jest rozwiązaniem problemu, nie wynosi mniej niż 25%. Dodatkowo dokładność oszacowania szybko rośnie wraz z rozmiarem badanej funkcji i dla problemów o jakkolwiek istotnych rozmiarach ta niedokładność jest pomijalna, co zostało opisane w podręczniku M. Hirvensalo [86].

Z otrzymanego wzoru na liczbę iteracji można wnioskować o złożoności tego algorytmu. Dla przypomnienia, w przypadku klasycznym, podejście brutalne skutkuje potrzebą zapytania wyroczni ok $N = 2^n$ razy. w przypadku algorytmu Grovera, wymagana liczba wywołań wyroczni wynosi około $2^{\frac{n}{2}} = \sqrt{N}$. Można zatem zaobserwować kwadratową redukcję złożoności obliczeniowej.

Umieszczając to w kontekście bezpieczeństwa kryptograficznego, atak w oparciu o algorytm Grovera efektywnie redukuje rozmiar kluczy o połowę. Dodatkowo należy zauważyć, że algorytm ten nie korzysta ze specjalnych właściwości funkcji, którą ma rozwiązać. w związku z tym algorytm ten może zostać użyty do ataku na dowolny kryptosystem, zarówno symetryczny, jak i asymetryczny, podobnie jak atak brutalny.

4.2 Optymalność Algorytmu

W roku 1999, w pracy [87] Christof Zalka udowodnił, że dla tak zadanego problemu spełnialności, gdzie nie można wnioskować na temat struktury problemu, a istnieje jedynie dostęp do wyroczni kwantowej, która implementuje odpowiednią funkcję, algorytm Grovera jest optymalny i niemożliwe jest uzyskanie większej redukcji złożoności niż otrzymana redukcja kwadratowa.

4.3 Atak na szyfr symetryczny

Proponowany model ataku zakłada, że fragment szyfrogramu (C) i odpowiadającego mu tekstu jawnego (M) jest znany. Atakujący ma też dostęp do kwantowej wyroczni szyfrującej U_E, która pozwala mu dokonać szyfrowania wiadomości superpozycją wszystkich kluczy K z przestrzeni kluczy $\mathbb{K}$.

$$\frac{1}{\sqrt{|\mathbb{K}|}} \sum_{K \in \mathbb{K}} |K\rangle |M\rangle \xrightarrow{U_E} \frac{1}{\sqrt{|\mathbb{K}|}} \sum_{K \in \mathbb{K}} |k\rangle |E_K(M)\rangle \tag{4.31}$$

Gdzie $E_K(m)$ jest szyfrogramem wiadomości M uzyskanym przez wykorzystanie klucza K. Niech K' będzie poszukiwanym kluczem symetrycznym (rozwiązaniem) oraz $(M, C = E_{K'}(M))$ znaną parą tekstu jawnego i szyfrogramu. Wartości C można użyć do warunkowego odbicia fazy tylko tych stanów, które reprezentują szyfrogramy odpowiadające rozwiązaniu K' poprzez odpowiednie wykorzystanie operacji `C-XOR(C)` i warunkowej rotacji CZ.

$$\frac{1}{\sqrt{|\mathbb{K}|}}\sum_{K\in\mathbb{K}}|K\rangle\,|E_K(M)\rangle \xrightarrow{CZ} \frac{1}{\sqrt{|\mathbb{K}|}}\left(\left(\sum_{K\in\mathbb{K}\setminus\{K'\}}|K\rangle\,|E_K(M)\rangle\right)-|K'\rangle\,|C\rangle\right) \tag{4.32}$$

Korzystając z odwrotnej wyroczni szyfrującej $U_E^\dagger$ można odwrócić szyfrowanie i usunąć splątanie rejestrów:

$$\begin{aligned}\frac{1}{\sqrt{|\mathbb{K}|}}\left(\left(\sum_{K\in\mathbb{K}\setminus\{K'\}}|K\rangle\,|E_K(M)\rangle\right)-|K'\rangle\,|C\rangle\right) \xrightarrow{U_E^\dagger} \\ \xrightarrow{U_E^\dagger}\frac{1}{\sqrt{|\mathbb{K}|}}\left(\sum_{K\in\mathbb{K}\setminus\{K'\}}\big(|K\rangle\big)-|K'\rangle\right)|M\rangle\end{aligned} \tag{4.33}$$

Procedura ta pozwala uzyskać odbicie fazy tylko dla wybranego stanu $|K'\rangle$, efektywnie otrzymując operator kodowania fazowego dla funkcji skonstruowanej w ten sposób, że jej rozwiązanie jest równoważne ze znalezieniem szukanego klucza.

4.3.1 Unikalność rozwiązań

Aby skorzystać z ataku w tej formie, należy zapewnić, że istnieje tylko jedno rozwiązanie problemu. Dla każdego szyfru spełniającego ścisłe kryterium lawinowe klucza *(z j. ang. SAC - strict key avalanche criterion)* zdefiniowaną w pracy E. Dawsona [88], możliwe jest, aby dwa lub więcej kluczy mogło generować taki sam szyfrogram z jednej wiadomości. aby temu zapobiec, można użyć więcej niż jednej pary odpowiadających sobie szyfrogramów i wiadomości. Można wyliczyć, że gdy kryterium SAC jest spełnione, rozmiar klucza to $|K|$, a rozmiar bloku wiadomości to $|M|$, najmniejsza liczba par wynosi:

$$\mu > \left\lceil\frac{2|K|}{|M|}\right\rceil \tag{4.34}$$

W najczęściej spotykanym przypadku, kiedy rozmiar bloku jest równy rozmiarowi bloku wiadomości $\mu = 3$.

Niech zbiór $\mathbb{K}^* = \{(K_0, K_1) \in \mathbb{K}^2 : K_0 \neq K_1\}$ – zbiór par kluczy różnych od siebie. Dla pary kluczy (k_0, k_1) prawdopodobieństwo kolizji, tzn. sytuacji, w której szyfrogramy jednego bloku wiadomości, uzyskane z dwóch różnych kluczy, będą równe, wynosi $P_C(|M|, 1) = 2^{-|M|}$ jeżeli szyfr spełnia ścisłe kryterium lawinowe. Jeżeli wiadomość składa się z μ bloków, to prawdopodobieństwo, że wszystkie bloki zostaną zaszyfrowane do tej samej wartości, wynosi $P_C(|M|, \mu) = P_C(|M|, 1)^{\mu} = 2^{-\mu|M|}$. Wiedząc, że moc zbioru $|K^*| = 2^{2|K|} - 2^{|K|} < 2^{2|K|}$, to oczekiwana liczba kolizji $\#C$, czyli par kluczy, które generują ten sam szyfrogram, wynosi:

$$\begin{aligned} \#C(\mu) &= |\mathbb{K}^*| * P(|M|, \mu) \\ \#C(\mu) &< 2^{2|K|} * P(|M|, \mu) \end{aligned} \tag{4.35}$$

Aby mieć pewność, że rozwiązanie jest unikalne, oczekiwana liczba kolizji musi być mniejsza niż 1. Jeżeli ograniczenie górne tej liczby jest mniejsze od 1 to rzeczywista liczba kolizji też będzie mniejsza od 1.

$$\begin{aligned} \#C(\mu) < 2^{2|K|} * P(|M|, \mu) &< 1 \\ 2^{2|K|} * 2^{-\mu|M|} &< 1 \\ 2^{2|K|-\mu|M|} &< 1 \\ 2|K| - \mu|M| &< 0 \\ \mu &> \left\lceil \frac{2|K|}{|M|} \right\rceil . \end{aligned} \tag{4.36}$$

□

Otrzymując w ten sposób najmniejszą liczbę bloków szyfrogramu i tekstu jawnego μ wymaganą do przeprowadzenia ataku.

W przypadku szyfrów z uwierzytelnianiem, tag uwierzytelnienia również może nieść informacje o użytym kluczu, podobnie jak szyfrogram. w związku z tym może zostać użyty do przeprowadzenia ataku i jego optymalizacji. Dla ataku opartego o algorytm Grovera może on być użyty do redukcji rozmiaru wymaganej wiadomości w postaci jawnej, niezbędnej do zapewnienia unikalności rozwiązania.

Analogicznie jak dla bloku wiadomości, prawdopodobieństwo kolizji tagów wynosi $P_C(|T|) = 2^{-|T|}$. Prawdopodobieństwo, że μ bloków wiadomości, każdy długości $|M|$ i tag długości $|T|$ będą jednocześnie kolidować, wynosi:

$$P_C(|M|, \mu, |T|) = P_C(|M|, \mu) * P_C(|T|) = 2^{-\mu|M|} * 2^{-|T|} = 2^{-(\mu|M|+|T|)} \tag{4.37}$$

Postępując podobnie jak w poprzednim przypadku, po wyznaczeniu oczekiwanej liczby kolizji $\#C$ i ograniczeniu jej wartością 1 otrzymano oszacowanie wartości μ – minimalnej liczby bloków:

$$\begin{aligned} \#C(\mu) < 2^{2|K|} * P_C(|M|, \mu, |T|) &< 1 \\ 2^{2|K|} * 2^{-(\mu|M|+|T|)} &< 1 \\ 2^{(2|K|-\mu|M|-|T|)} &< 1 \\ 2|K| - \mu|M| - |T| &< 0 \\ \mu &> \left\lceil \frac{2|K| - |T|}{|M|} \right\rceil . \end{aligned} \tag{4.38}$$

Rozdział 5

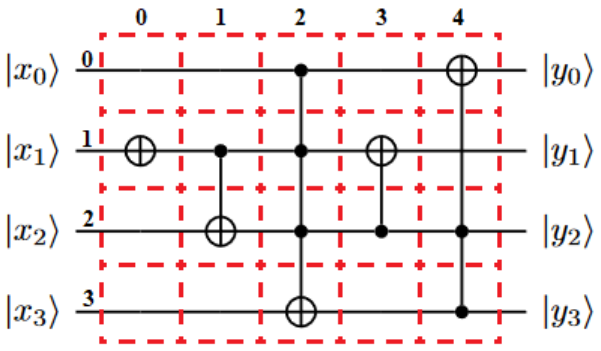
Optymalne układy implementujące skrzynki podstawieniowe

Celem badania jest odnalezienie optymalnych implementacji skrzynek podstawieniowych wybranych szyfrów symetrycznych oraz kryptograficznych funkcji skrótu. w szczególności dla kandydatów, którzy brali udział w konkursie NIST na kryptografię lekką [7].

5.1 Metoda badawcza

5.1.1 Kodowanie układu

Pierwszym krokiem w formułowaniu odpowiedniego problemu spełnialności jest znalezienie jednoznacznego i efektywnego kodowania układu złożonego z bramek `MCT`. w tym celu układ został umieszczony na prostokątnej siatce, tak aby jedna linia logiczna znajdowała się w jednym wierszu, a każda bramka znajdowała się w jednej kolumnie siatki. Wymiary siatki są równe szerokości `W` i głębokości `D` układu. Siatka składa się z `W` $*$ `D` pól. Ponieważ badanie zakłada, że w każdej warstwie znajduje się tylko jedna bramka, to liczba bramek jest równa głębokości układu `#G` $=$ `D`.



Rysunek 5.1: Podział układu na pola siatki

Dla dowolnego układu można zaobserwować, że każde pole musi się znajdować w jednym z trzech stanów:

1. Pole jest puste;
2. Pole zawiera znacznik qubitu kontrolnego;
3. Pole zawiera znacznik qubitu docelowego.

Wynika z tego, że do zakodowania stanu każdego pola niezbędne są dwa bity. Niech $0 \leq i < \mathtt{D}, 0 \leq j < \mathtt{W}$ będzie parą współrzędnych wskazujących jedno z pól – i wskazuje na kolumnę siatki, a j na wiersz. Do zakodowania stanu pola użyte zostaną dwie zmienne $c_{i,j}$ oraz $t_{i,j}$. Pierwsza z nich opisuje, czy pole zawiera qubit kontrolny, a druga, czy zawiera qubit docelowy.

Układ z rysunku 5.1 może zostać zapisany przy użyciu $\mathtt{D} * \mathtt{W} * 2 = 4 * 5 * 2 = 40$ zmiennych boolowskich:

$$\begin{gathered}
c_{0,0} = 0,\ c_{0,1} = 0,\ c_{0,2} = 1,\ c_{0,3} = 0,\ c_{0,4} = 0 \\
c_{1,0} = 0,\ c_{1,1} = 1,\ c_{1,2} = 1,\ c_{1,3} = 0,\ c_{1,4} = 0 \\
c_{2,0} = 0,\ c_{2,1} = 0,\ c_{2,2} = 1,\ c_{2,3} = 1,\ c_{2,4} = 1 \\
c_{3,0} = 0,\ c_{3,1} = 0,\ c_{3,2} = 0,\ c_{3,3} = 0,\ c_{3,4} = 1
\end{gathered} \tag{5.1}$$

$$\begin{gathered}
t_{0,0} = 0,\ t_{0,1} = 0,\ t_{0,2} = 0,\ t_{0,3} = 0,\ t_{0,4} = 1 \\
t_{1,0} = 1,\ t_{1,1} = 0,\ t_{1,2} = 0,\ t_{1,3} = 1,\ t_{1,4} = 0 \\
t_{2,0} = 0,\ t_{2,1} = 1,\ t_{2,2} = 0,\ t_{2,3} = 0,\ t_{2,4} = 0 \\
t_{3,0} = 0,\ t_{3,1} = 0,\ t_{3,2} = 1,\ t_{3,3} = 0,\ t_{3,4} = 0
\end{gathered}$$

Oczywiście, pomiędzy takimi zmiennymi występują pewne ograniczenia:

1. Pole nie może zawierać jednocześnie qubitu docelowego i kontrolnego;
2. W każdej kolumnie może być najwyżej jeden qubit docelowy;
3. Dla układów `NCT`, liczba qubitów kontrolnych w każdej warstwie nie może przekroczyć 2. dla układów `MCT` liczba ta jest dowolna o ile poprzednie ograniczenia są spełnione.

Formalnie ograniczenia te można zapisać jako:

$$\forall_{0\leq i\ <G, 0\leq j\ <W} \left(\neg c_{i,j} \vee \neg t_{i,j}\right) \tag{5.2}$$

$$\forall_{0\leq i\ <G} \left(\sum_{0\leq j\ <W} t_{i,j} = 1\right) \tag{5.3}$$

$$\forall_{0\leq i\ <G} \left(\sum_{0\leq j\ <W} c_{i,j} \leq 2\right) \tag{5.4}$$

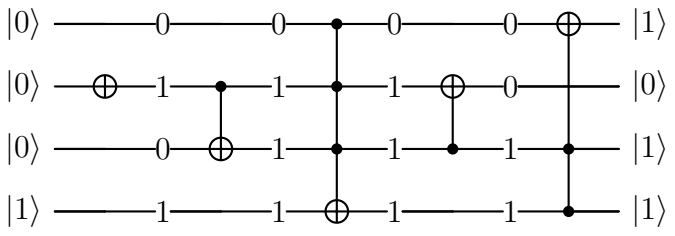
Gdzie ostatnie obostrzenie może zostać pominięte dla układów `MCT`.

Drugie i trzecie ograniczenie są tak zwanymi ograniczeniami kardynalnymi – opisują, ile jednocześnie zmiennych z zadanego zbioru może posiadać wartość 1. Ograniczenia te nie są trywialne do zaimplementowania jako formuła `CNF` obsługiwana przez SAT-solvery. Przekształcenia tych ograniczeń do formuły `CNF` zostały opisane w podręczniku A. Biere [89].

Opisane powyżej ograniczenia są wystarczające do opisania poprawnego układu logiki odwracalnej typu `NCT` lub `MCT`. Ograniczenia te jednak nie są wystarczające, aby opisać żądaną funkcjonalność układu. Próba rozwiązania problemu spełnialności z wyłącznie tak opisanymi ograniczeniami będzie skutkowała zwróceniem wyniku, który reprezentuje poprawny, ale "losowy"układ o zadanej głębokości i szerokości.

5.1.2 Przypisywanie funkcjonalności do układu

Wiedząc jak zakodować poprawny układ o wybranych wymiarach przy użyciu odpowiednich zmiennych i ograniczeń, kolejnym krokiem jest rozszerzenie zestawu ograniczeń, które umożliwią wymuszenie na syntezowanym układzie, aby realizował zadaną funkcję odwracalną.



Rysunek 5.2: Transformacje realizowane przez układ `MCT`

Rysunek 5.2 przedstawia przepływ słowa bitowego i jego zmiany bramka po bramce. Znając zasady działania bramek `MCT` oraz obserwując układ powyżej, można wywnioskować dwa warunki wymagane, aby linia bitowa zmieniła wartość po wykonaniu bramki:

1. Pole w danej linii i kolumnie musi zawierać qubit docelowy;
2. Dla wszystkich pól zawierających qubity kontrolne, wartość przetwarzanego bitu musi wynosić 1.

Formalnie zapisując, bit zmienia swoją wartość przy przejściu z $b_{i,j}$ do $b_{i+1,j}$ gdy:

1. $t_{i,j} = 1 \wedge c_{i,j} = 0$;
2. $\bigwedge_{j'=0, j' \neq j}^{w-1} (c_{i,j'} \implies b_{i,j'})$.

Niefortunnie, drugi warunek jest złożony i musi być wyznaczony dla każdego indeksu j. Można jednak zauważyć, że pierwszy warunek wymusza stan zmiennej $t_{i,j} = 1$, a co za tym idzie $c_{i,j} = 0$ z równania 5.2.

Z tego wynika, że dla $j' = j$ $(c_{i,j} \implies b_{i,j'}) = 1$, korzystając z właściwości koniunkcji $a \wedge 1 = a$, warunek można uprościć:

$$\begin{aligned}
\left(\bigwedge_{j'=0, j' \neq j}^{w-1} (c_{i,j'} \implies b_{i,j'}) \right) &\equiv \left(\bigwedge_{j'=0, j' \neq j}^{w-1} (c_{i,j'} \implies b_{i,j'}) \right) \wedge 1 \equiv \\
&\equiv \left(\bigwedge_{j'=0, j' \neq j}^{w-1} (c_{i,j'} \implies b_{i,j'}) \right) \wedge (c_{i,j} \implies b_{i,j}) \equiv \quad (5.5) \\
&\equiv \left(\bigwedge_{j'=0}^{w-1} (c_{i,j'} \implies b_{i,j'}) \right).
\end{aligned}$$

Nowa postać warunku jest zupełnie niezależna od indeksu wiersza j i może zostać wyznaczona raz na kolumnę zamiast raz na pole. Takie przekształcenie warunku zmiany istotnie zmniejsza liczbę zmiennych i klauzul w ostatecznej formule `CNF`, co przekłada się na szybsze rozwiązanie problemu przez SAT-solver.

Posiadając tak wyznaczone warunki zmiany stanu bitu przechodzącego przez bramkę w kolumnie o indeksie i można wyznaczyć ogólne równanie rekurencyjne stanu bitu o indeksie $b_{i+1,j}$:

$$b_{i+1,j} = b_{i,j} \oplus \left(t_{i,j} \wedge \bigwedge_{j=0}^{w-1} (c_{i,j} \implies b_{i,j}) \right) \tag{5.6}$$

Tak opisane ograniczenia są wystarczające do zakodowania transformacji jednego słowa bitowego przez układ do pojedynczego wyjścia. Należy uwzględnić, że formułując zadania dla SAT-solvera, należy wymusić na układzie całą funkcję, a nie tylko pojedyncze powiązanie pomiędzy wybranym wejściem i wyjściem funkcji.

Do opisania kompletu powiązań wejść i wyjść zostanie wprowadzony dodatkowy indeks górny zmiennych danych k:

$$\begin{aligned} b_{i,j}^{k} : \; & i \; \in \{0, \ldots, G-1\}, \\ & j \; \in \{0, \ldots, W-1\}, \\ & k \; \in \{0, \ldots, 2^{W}-1\} \end{aligned} \tag{5.7}$$

Dodatkowy indeks oznacza, z jakiego wejściowego słowa bitowego dany stan został wytworzony. Zmienną $b_{i,j}^{k}$ należy zatem interpretować jako stan j-tej linii bitowej rozpoczętej od wejścia k, po przetworzeniu przez i bramek. Dla uproszczenia indeks k będzie też zapisywany liczbą naturalną reprezentującą słowo bitowe.

Należy zwrócić uwagę na to, że zmienne $t_{i,j}, c_{i,j}$ nie posiadają tego indeksu ze względu na to, że opisują one sam układ, a nie przepływ danych przez układ.

Kodowanie przekształceń bitów musi być powtórzone dla każdego słowa bitowego, czyli $k \in \{0, \ldots, 2^{\mathtt{W}} - 1\}$. Wynikiem tego jest potrzeba użycia dodatkowych $(2^{\mathtt{W}} - 1) * \mathtt{W} * (\mathtt{D} + 1)$ zmiennych.

Na koniec, należy wprowadzić ograniczenia graniczne dla powiązanych ze sobą wartości wejść i wyjść funkcji. Jest to realizowane wprost poprzez przypisanie wartości do wszystkich zmiennych $b_{0,j}^{k}$ żądanych wartości wejść, które reprezentują, a do zmiennych $b_{\mathtt{D},j}^{k}$ żądanych wartości wyjść funkcji odwracalnej, dla której układ ma zostać wygenerowany. Wartości tych zmiennych są jedynymi znanymi wartościami przed rozpoczęciem rozwiązywania układu przez SAT-solver.

Ostateczna formuła CNF zawierająca opisane wcześniej ograniczenia zawiera nie mniej niż $D*W*2^W$ zmiennych. Dokładna liczba jest dodatkowo zależna od wybranego kodowania ograniczeń kardynalnych, a także od tego, czy zostały użyte pośrednie zmienne do reprezentacji opisanych ograniczeń.

Rozwiązanie tak postawnego problemu spełnialności zwraca wartości zmiennych $c_{i,j}$ i $t_{i,j}$, które mogą zostać zdekodowane i użyte do utworzenia układu, który implementuje zadaną funkcję odwracalną.

Warto zauważyć, że formuły CNF używane do syntezy dwóch różnych funkcji o tym samym rozmiarze są prawie identyczne. Jedyną różnicą są graniczne przypisania do zmiennych reprezentujących wyjścia funkcji.

5.2 Część zasadnicza badania

W ramach badania postanowiono odnaleźć optymalne implementacje skrzynek podstawieniowych S-box i innych przekształceń nielinowych używanych w aktualnie używanych lub badanych szyfrach symetrycznych i funkcjach skrótu, w szczególności skupiono się na finalistach niedawno zakończonych konkursów:

1. *NIST Lightweight Cryptography Call* [7];
2. *NIST Hash Function Call* [90];
3. *NESSIE* [91].

Zbadano funkcje następujących prymitywów kryptograficznych:

Rozmiar funkcji (n)	Nazwa prymitywu	Nazwa funkcji
3×3	Xoodyak [19]	Chi
4×4	Elephant [11]	–
	GIFT-COFB [12]	SubCells
	PHOTON-Beetle [15]	–
	MiniAES [92]	–
	Whirlpool [93]	e, r
	JH [94]	0, 1
5×5	ASCON / ISAP-A [10] [14]	–
	KECCAK / ISAP-K [95] [14]	Chi

Tabela 5.1: Lista badanych funkcji nieliniowych

Wszystkie funkcje zostały zsyntezowane zarówno przy pomocy zestawu bramek NCT, jak i MCT. Podobnie dla każdej funkcji przeprowadzono syntezę bez użycia qubitów pomocniczych, jednego qubitu pomocniczego lub n qubitów pomocniczych, tworząc w ten sposób sześć różnych zestawów warunków początkowych, którym przypisano następujące oznaczenia:

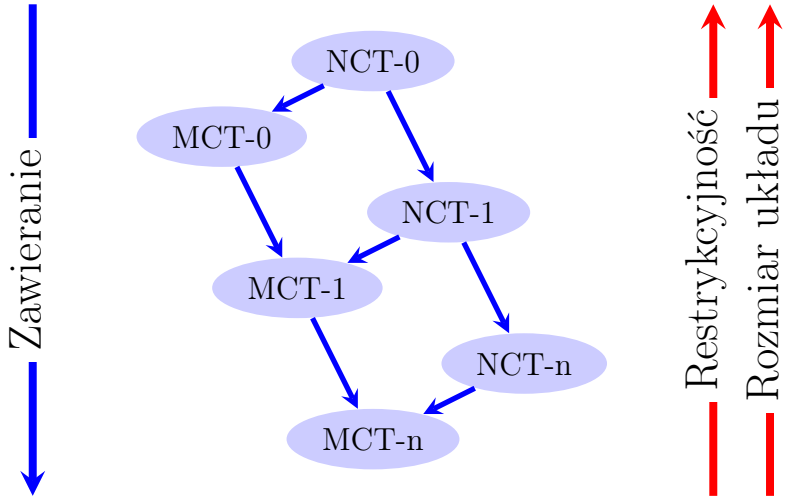
B\N	0	1	n
NCT	NCT-0	NCT-1	NCT-n
MCT	MCT-0	MCT-1	MCT-n

B – zestaw bramek, N – liczba bitów pomocniczych.

Tabela 5.2: Kombinacje warunków początkowych syntezy

5.2.1 Relacje pomiędzy zestawami warunków początkowych

Należy zauważyć, że pomiędzy wskazanymi zestawami ograniczeń zachodzą relacje, z których dalej wynika hierarchia. Np. każde rozwiązanie dla zestawu warunków NCT-0 jest poprawnym rozwiązaniem dla warunków MCT-0, w związku z tym optymalne rozwiązanie w zestawie MCT-0 nie może być większe niż w zestawie NCT-0. Podobnie, każde rozwiązanie w MCT-0 jest poprawne w MCT-1 i MCT-n.



Rysunek 5.3: Hierarchia zestawów warunków początkowych

Niebieskie krawędzie grafu 5.3 oznaczają zawieranie zbioru rozwiązań, gdzie wierzchołek grafu przy grocie krawędzi zawiera wszystkie rozwiązania z wierzchołka z przeciwnej strony krawędzi. Graf jest przechodni, a niebezpośrednie relacje zawierania zostały pominięte.

Istotnym spostrzeżeniem, które może zostać użyte do skrócenia czasu badania, jest fakt, że na podstawie tego grafu można wyznaczyć zależności pomiędzy rozmiarami rozwiązań, a co za tym idzie, przedziały, w jakich będą znajdować się rozmiary optymalnych układów. Wierzchołek `NCT-0` jest minimalny w sensie zawierania, a wierzchołek `MCT-n` jest maksymalny. Co za tym idzie, dla zestawu ograniczeń `NCT-0` rozwiązanie będzie miało największy rozmiar, a dla `MCT-n` najmniejszy.

Niech dla wybranej funkcji odwracalnej znaleziony zostanie optymalny układ o rozmiarze d_1 przy założeniach NCT-0, oraz d_2 przy założeniach MCT-n. dla pozostałych założeń rozwiązania muszą mieć rozmiar w przedziale $[d_1, d_2]$. Znając te wartości można przyspieszyć obliczenia dla pozostałych czterech wierzchołków, a w szczególnym przypadku gdy $d_1 = d_2$ pominąć je.

5.2.2 Środowisko obliczeniowe

Do rozwiązania opracowanych problemów spełnialności, które reprezentują zadanie syntezy poszczególnych układów, użyto 5 różnych SAT-solverów rozproszonych, które uzyskiwały najlepsze wyniki na czas badania, które zostały potwierdzone w konkursie *2022 SAT-Solver Competition* [96]. Po przetestowaniu wszystkich pięciu na mniejszych problemach syntezy zauważono, że wyniki najszybciej zwracało narzędzie `Parkisat`, które jednocześnie wygrało konkurs. W związku z tym zostało ono wykorzystane do rozwiązywania zasadniczych problemów. Narzędzie zostało uruchomione na serwerze zawierającym dwa procesory po 64 rdzenie.

5.2.3 Wyniki badania

Funkcja	War. P.	#G	#NOT	#CNOT	#Toff
Xoodyak	NCT-0	6	0	3	3
	NCT-n	6	0	3	3

Tabela 5.3: Wyniki syntezy funkcji nieliniowych rozmiaru 3×3

Funkcja	War. P.	#G	#NOT	#CNOT	#Toff	#Toff-3
Elephant	NCT-0	10	1	4	5	-
	MCT-n	10	1	4	5	0
GIFT-COFB	NCT-0	9	2	3	4	-
	MCT-n	9	2	3	4	0
PHOTON-Beetle	NCT-0	11	1	5	5	-
	MCT-n	11	1	5	5	0
Mini-AES	NCT-0	13	2	6	5	-
	NCT-1	13	2	6	5	-
	NCT-n	13	2	6	5	-
	MCT-0	12	2	6	2	2
	MCT-1	12	2	6	2	2
	MCT-n	12	2	6	2	2
Whirlpool-e	NCT-0	13	1	4	8	-
	NCT-1	13	1	4	8	-
	NCT-n	13	1	4	8	-
	MCT-0	12	1	4	5	2
	MCT-1	12	1	4	5	2
	MCT-n	12	1	4	5	2
Whirlpool-r	NCT-0	13	3	5	5	-
	MCT-n	13	3	5	5	0
JH-0	NCT-0	10	1	2	7	-
	MCT-n	10	1	2	7	0
JH-1	NCT-0	12	2	4	6	-
	MCT-n	12	2	4	6	0

Tabela 5.4: Wyniki syntezy funkcji nieliniowych rozmiaru 4×4

Funkcja	War. P.	#G	#NOT	#CNOT	#Toff	#Toff-3	#Toff-4
Ascon	NCT-0	17	1	6	9	-	-
	NCT-1	17	1	6	9	-	-
	NCT-n	17	1	6	9	-	-
	MCT-0	16	1	6	7	2	0
	MCT-1	16	1	6	7	2	0
	MCT-n	16	1	6	7	2	0
Keccak	NCT-0	13	0	5	8	-	-
	NCT-1	13	0	5	8	-	-
	NCT-n	13	0	5	8	-	-
	MCT-0	12	3	3	5	1	0
	MCT-1	12	3	3	5	1	0
	MCT-n	12	3	3	5	1	0

Tabela 5.5: Wyniki syntezy funkcji nieliniowych rozmiaru 5×5

W pierwszej kolejności obliczenia były wykonywane przy założeniach `NCT-0` i `MCT-n`. Jeżeli w obu przypadkach znalezione układy były tego samego rozmiaru, pozostałe zestawy założeń były pomijane.

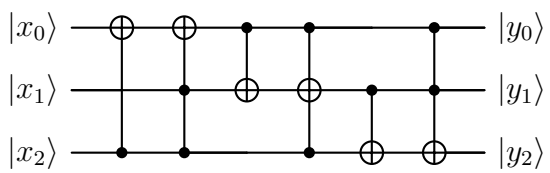
- `Xoodyak` – Jedyna syntezowana funkcja o rozmiarze 3×3, w związku z tym dla tej funkcji zestaw `NCT` i `MCT` są identyczne. dla obu granicznych zestawów warunków początkowych otrzymano identyczny układ. Zgodnie z wcześniejszym opisem, dla pozostałych czterech zestawów wszystkie rozwiązania będą mieć ten sam rozmiar.

- `Elephant, GIFT-COFB, PHOTON-Beetle, Whirlpool-r, JH-0, JH-1` – W przypadku tych funkcji dla obu granicznych przypadków otrzymano różne układy ale o tym samym rozmiarze. Analogicznie pozostałe cztery przypadki muszą mieć rozwiązania o tym samym rozmiarze.

- `Mini-AES, Whirlpool-e, Keccak` – W przeciwieństwie do poprzednich funkcji, tutaj nie otrzymano układów o tym samym rozmiarze. Można zauważyć, że rozmiar układu nie zależy od liczby dodanych qubitów pomocniczych, a wyłącznie z rozszerzenia zestawu bramek.

- `Ascon` – dla tej funkcji nie udało się potwierdzić optymalności układu. Dla wszystkich zestawów ograniczeń udało się znaleźć rozwiązania nie większe niż 17 bramek i udało się dowieść, że nie istnieją rozwiązania mniejsze niż 13 bramek.

W tabeli 5.6, 5.7 oraz 5.8, przedstawiono wszystkie znalezione optymalne implementacje. Układy są zapisane w notacji przedstawionej w rozdziale 3, czyli jako listę krotek, każda krotka reprezentuje jedną bramkę. Pierwszy indeks w krotce to indeks linii qubitu docelowego, a dalsze, jeżeli występują, to indeksy qubitów kontrolujących. Rysunek 5.4 przedstawia optymalną implementację skrzynki z szyfru `Xoodyak`.



Rysunek 5.4: Optymalna implementacja skrzynki podstawieniowej `Xoodyak`

Funkcja	War. P.	Układ
Xoodyak	Dowolne	(0,2); (0,1,2); (1,0); (1,0,2); (2,1); (2,0,1)

Tabela 5.6: Optymalne implementacje skrzynek podstawieniowych rozmiaru 3×3

Funkcja	War. P.	Układ
Elephant	Dowolne	(0,1,2); (2,0,3); (0,1); (1); (0,3); (2,1); (1,2,3); (3,1,2); (1,0); (3,0,1)
GIFT-COFB	Dowolne	(1,0,2); (0,1,3); (2,1); (3); (1); (3,0,1); (3,2); (1,3); (2,0,3)
PHOTON-Beetle	Dowolne	(3,1,2); (2,3); (1,2,3); (3,1,2); (0,2); (1,0,3); (2); (3,0); (3,2); (3,1); (2,1,3)
Mini-AES	NCT	(0,1); (1,0); (1); (3,1); (2,0,3); (0,3); (1,2); (0,1,2); (1,0,3); (3,1,2); (1,0,3); (2,0); (0)
	MCT	(0,1); (1,0); (1); (3,1); (2,0,3); (1,2); (3,1,2); (0,3); (1,0,2,3); (3,0,1,2); (2,0); (0)
Whirlpool-e	NCT	(2,0,1); (1,0,3); (0,1,2); (3,0); (0); (1,2,3); (2,0,3); (0,1,2); (1,0,2); (3,1); (1,3); (3,0,1); (0,1)
	MCT	(3,1,2); (0,1,2,3); (2,3); (1,2,3); (2,0,1); (1,0); (1,0,2,3); (0); (3,1); (1,0,3); (3,2); (0,1,3)
Whirlpool-r	Dowolne	(3); (2,0); (0,2,3); (2); (2,0,1); (1,3); (3,1,2); (1,0,3); (0,3); (0,1); (1); (2,3); (1,0,2)
JH-0	Dowolne	(0,2,3); (0,1); (3,0,1); (2,1,3); (1,0,2); (0,1,3); (3,0,2); (1,0,3); (0); (3,0)
JH-1	Dowolne	(0,2,3); (2,1); (3,0,1); (0,2); (2,1,3); (3,0); (1,0,3); (3,0,2); (0,1,2); (2,3); (1); (0)

Tabela 5.7: Optymalne implementacje skrzynek podstawieniowych rozmiaru 4×4

Funkcja	War. P.	Układ
Ascon	NCT	(4,2,3); (2,0); (4,2); (2,0,1); (0,3,4); (2,3); (0,1,2); (1,0,2); (2); (3,1,2); (0,3); (4,0,2); (1,3,4); (4,0,2); (3,4); (4,0); (1,3,4)
	MCT	(4,2,3); (2,0); (4,2); (2,3); (2,0,1); (0,1); (3,2,4); (1,0,2); (0,3); (1,0,2,3); (2); (1,2,4); (0,3,4); (3,1,2); (1,0,2,4); (4,0)
Keccak	NCT	(1,0,3); (1,2,3); (3,0,4); (3,0); (0,2); (1,0,3); (0,1,2); (2,4); (2,3,4); (4,1); (4,0,1); (1,2,3); (1,3)
	MCT	(4); (3,0,4); (0,1,2); (0,2); (2,4); (1,0,2,4); (4); (2,3,4); (4,0,1); (4,1); (1,2,3); (2)

Tabela 5.8: Optymalne implementacje skrzynek podstawieniowych rozmiaru 5×5

5.2.4 Transpilacja do bramek Clifforda

Otrzymane układy posiadają optymalną liczbę bramek `NCT` lub `MCT`. Układy logiki kwantowej, wychodzące poza zakres klasycznej logiki odwracalnej, zapisywane są często przy użyciu zestawu `Clifford+T` złożonego z trzech bramek *Clifforda* – $H, S = \sqrt{\texttt{NOT}}$ i `CNOT` oraz dołączonych dwóch bramek – $\texttt{T} = \sqrt{S} = \sqrt[4]{\texttt{NOT}}$ i $\texttt{T}^\dagger = \texttt{T}^{-1}$.

Zestaw ten jest kompletny, oznacza to, że pozwala przybliżyć każdą bramkę kwantową z dowolnie wybraną dokładnością, natomiast nie wszystkie bramki są możliwe do przedstawienia perfekcyjnie. Układy reprezentujące klasyczną logikę odwracalną są możliwe do przedstawienia w tej postaci z pełną dokładnością.

W praktyce, taka translacja może zostać wykonana ręcznie lub przy użyciu ogólnie dostępnych narzędzi, np. `Qiskit`[97], które oferuje funkcjonalności przydatne w projektowaniu doświadczeń związanych z obliczeniami kwantowymi, takie jak wykonywanie obliczeń w różnych środowiskach, np. symulatorach lub rzeczywistych komputerach kwantowych, transpilacje układów kwantowych pomiędzy zestawami bramek, badanie wpływu zakłóceń na obliczenia i inne.

```python
from qiskit import QuantumCircuit
from qiskit.compiler import transpile
basis_gates = ["h", "s", "t", "tdg", "cx"]
xoodyak = [(0, 2), (0, 1, 2), (1, 0), (1, 0, 2), (2, 1), (2, 0, 1)]
size = max([max(tuple(g)) for g in l]) + 1
qc = QuantumCircuit(size)
for gate in xoodyak:
    target = gate[0]
    controls = gate(tup[1:])
    if controls == []:
        qc.x(target)
    else:
        qc.mcx(controls, target)
transpiled = transpile(qc, basis_gates=basis_gates)
cs = transpiled.count_ops()
print(qc)
print(f"Xoodyak - {cs}")
```

Kod źródłowy 1: Transpilacja do bramek `Clifford+T`

Korzystając z wycinka kodu źródłowego 1 otrzymano wynik transpilacji dla skrzynki `Xoodyak`, a także dla innych układów. Wyniki transpilacji zamieszczono w tabelach 5.9, 5.10 i 5.11.

Funkcja	#H	#S	#CNOT	#(T/T†)
Xoodyak	6	0	21	21

Tabela 5.9: Wyniki transpilacji `Clifford+T` funkcji rozmiaru 3×3

Funkcja	#H	#S	#CNOT	#(T/T†)
Elephant	12	2	34	35
GIFT-COFB	12	4	27	28
PHOTON-Beetle	12	2	35	35
Mini-AES	14	4	36	35
Whirlpool-e	18	2	52	56
Whirlpool-r	15	6	35	36
JH-0	16	2	44	49
JH-1	16	4	40	42

Tabela 5.10: Wyniki transpilacji `Clifford+T` funkcji rozmiaru 4×4

Funkcja	#H	#S	#CNOT	#(T/T†)
Ascon	22	2	66	70
Keccak	16	0	53	56

Tabela 5.11: Wyniki transpilacji `Clifford+T` funkcji rozmiaru 5×5

Należy zwrócić uwagę, że metryki w postaci liczby poszczególnych bramek z zestawu `Clifford+T` nie są już optymalne po przeniesieniu do nowego zestawu bramek. Pomimo że wyniki te są tylko heurystycznie zbliżone do najlepszych możliwych, to i tak dają one pewien pogląd na koszt tych układów, biorąc pod uwagę zmieniony zestaw bramek.

5.3 Dalsze kierunki rozwoju

Problem syntezy optymalnej składa się z dwóch zasadniczych części – redukcji problemu syntezy do problemu spełnialności oraz rozwiązania problemu spełnialności. Druga część jest realizowana przez zewnętrzne narzędzie, wykonane nie tylko z aktualnym stanem wiedzy, w związku z tym optymalizacja tego etapu jest poza zakresem tej pracy.

Podczas opracowywania pierwszej części, modelowania problemu spełnialności, podjęte zostały próby uproszczenia kodowania problemu syntezy w celu zmniejszenia liczby klauzul oraz zmiennych. Nie jest jednak wykluczone, że dalsze prace nad opracowaniem lepszej metody reprezentacji syntezy jako problemu SAT umożliwią przyspieszenie obliczeń, a co za tym idzie, zwiększenie rozmiarów skrzynek podstawieniowych zdatnych do syntezy w wyznaczonym czasie.

Drugą ścieżką rozwoju tego badania jest opracowanie metody syntezy optymalnej dla zestawu bramek `Clifford + T` oraz znalezienie optymalnych implementacji tych układów z użyciem tego zestawu bramek.

Rozdział 6

Synteza szablonów identyczności

Celem badania jest znalezienie kompletnej przestrzeni szablonów identyczności o wybranych rozmiarach.

6.1 Metoda badawcza

Badanie zostało oparte o algorytm syntezy opisany w rozdziale 5. Dla uproszczenia, metoda syntezy opisana w tym rozdziale zostanie wprowadzona funkcja $\texttt{synth}(\texttt{W}, \texttt{D}, f) \to c/\bot$, gdzie:

- $\texttt{W}$ - żądana szerokość układu;
- $\texttt{D}$ - żądana głębokość układu;
- f - funkcja do zaimplementowania przez układ;
- $c/\bot$ - układ o wymiarach $\texttt{W} \times \texttt{D}$ implementujący funkcję f lub symbol $\bot$, jeżeli taki układ nie istnieje.

Algorytm ten rozszerzony dalej jest o dodatkowy parametr $C = \{c_1, c_2 \cdots, c_n\}$, który jest listą wcześniej znalezionych układów o zadanych parametrach $\texttt{W}, \texttt{D}, f$. Rozszerzony algorytm znajduje nowy układ $c' \notin C$ lub zwraca $\bot$ jeżeli C jest kompletną listą takich układów – w szczególności może być pustą listą.

Rozszerzenie to może być łatwo zrealizowane poprzez zmodyfikowanie oryginalnego układu `CNF`. Odbywa się to przez dodanie nowej klauzuli dla każdego układu z C, takiej, że wyklucza ten układ jako rozwiązanie.

Nowa procedura zostanie oznaczona jako $\texttt{synth}(\texttt{W}, \texttt{D}, f, C) \to c/\bot$, gdzie:

- $\texttt{W}, \texttt{D}, f$ - Jak wcześniej;
- $c/\bot$ - Jak wcześniej, z różnicą, że $c \notin C$ lub $\bot$, jeżeli taki układ nie istnieje. Równoważnie oznacza to, że C jest kompletnym zbiorem układów o wymiarach $\texttt{W} \times \texttt{D}$ i implementujących f.

6.1.1 Naiwny algorytm syntezy szablonów

Na podstawie ogólnego opisu syntezy układów odwracalnych można skonstruować naiwny algorytm, który pozwala na znalezienie wszystkich szablonów identyczności o wymiarach $\mathtt{W} \times \mathtt{D}$.

Algorithm 1 Naiwny algorytm kompletnej syntezy szablonów identyczności

Require: $\mathtt{W} \times \mathtt{D}$

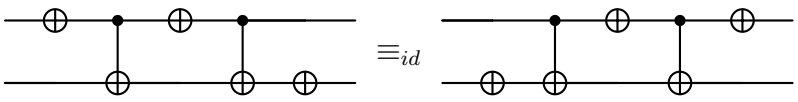
```
1: C := {}
2: while True do
3:    c := synth(W, D, id, C)
4:    if c = ⊥ then
5:       return C
6:    else
7:       C := C ∪ {c}
8:    end if
9: end while
```

Algorytm ten pozwala na znalezienie wszystkich szablonów, jednak jest nieefektywny, wynika to z faktu, iż wymaga rozwiązania jednego problemu SAT na każdy szablon identyczności do odnalezienia.

6.1.2 Generowanie szablonów

Korzystając z właściwości szablonów identyczności, istnieje możliwość generowania nowych układów przy znajomości jednego z nich. Generowanie nie wymaga rozwiązywania dodatkowych problemów SAT, a odbywa się przy użyciu właściwości szablonów i pewnych ich symetrii.

Pierwszą metodą jest odbicie, wiedząc, że odbity układ `MCT` zawsze implementuje funkcję odwrotną, a odwrotnością funkcji identyczności jest identyczność $id^{-1} = id$, odbicie szablonu też jest szablonem identyczności.



Rysunek 6.1: Odbicie szablonu

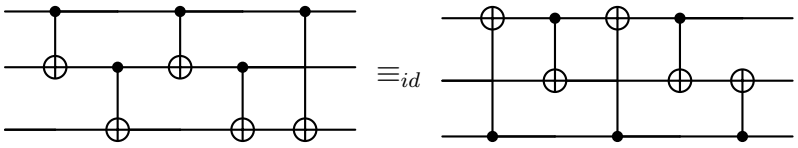
Drugą metodą generowania szablonów jest permutacja ich linii logicznych. Funkcja implementowana przez nowy układ, którego linie zostały poddane permutacji $\rho(f)$, może zostać zapisana jako złożenie początkowej permutacji ρ, oryginalnej funkcji f i permutacji odwrotnej ρ^{-1}.

$$\rho(f) = \rho \circ f \ \circ \rho^{-1} \tag{6.1}$$

Dla $f = id$:

$$\begin{aligned} \rho(id) &= \rho \circ id \ \circ \rho^{-1} \\ &= \rho \circ \rho^{-1} = id \end{aligned} \tag{6.2}$$

Zamiana kolejności linii logicznych generuje nowe szablony identyczności.

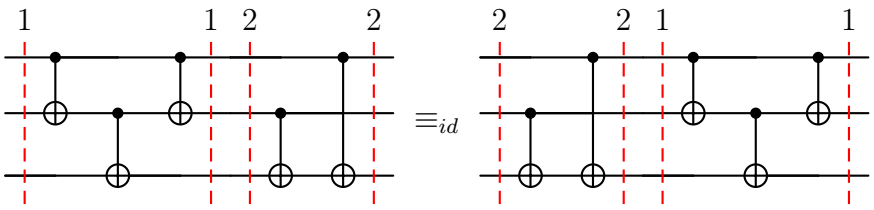


Rysunek 6.2: Permutacja linii szablonu

Następną metodą generowania jest rotowanie szablonów, niech szablon będzie listą bramek $c = g_n \circ \cdots \circ g_2 \circ g_1$. z lewej strony szablonu można dołączyć bramkę g_1, a z prawej g_1^{-1}, co skutkuje:

$$\begin{aligned} id \ &= g_n \circ \cdots \circ g_2 \circ g_1 \\ g_1 \circ id \ \circ g_1^{-1} &= g_1 \circ g_n \circ \cdots \circ g_2 \circ g_1 \circ g_1^{-1} \\ g_1 \circ g_1^{-1} &= g_1 \circ g_n \circ \cdots \circ g_2 \circ id \\ id \ &= g_1 \circ g_n \circ \cdots \circ g_2 \end{aligned} \tag{6.3}$$

Lewa strona równania nie uległa zmianie, zachowując funkcję identyczności. Po prawej stronie równości można zobaczyć, że bramka g_1 została przeniesiona na przeciwną stronę listy. Proces ten można powtórzyć dla pozostałych bramek g_i. Równoważnie, rotację szablonu można opisać jako przecięcie go w wybranym miejscu na dwie części i zamianę ich kolejności. Dla wybranego szablonu o rozmiarach $\mathtt{W} \times \mathtt{D}$, metoda ta umożliwia utworzenie aż do $\mathtt{D} - 1$ nowych układów.

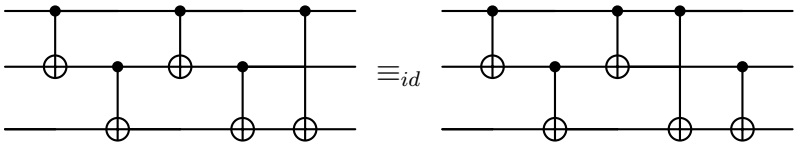


Rysunek 6.3: Rotacja szablonu identyczności

Ostatnią metodą generowania nowych szablonów jest zamiana niezależnych bramek. z pracy [64], sąsiednie bramki g_i, g_{i+1} są niezależne i mogą zostać ze sobą zamienione bez wpływu na funkcję układu wtedy i tylko wtedy, gdy:

$$\begin{cases} t_i \notin C_{i+1} \\ t_{i+1} \notin C_i \end{cases} \tag{6.4}$$

Gdzie t_i to indeks qubitu docelowego bramki g_i, a C_i to indeksy qubitów kontrolujących. W układzie z rysunku 6.2 oraz 6.3 dwie ostatnie bramki mogą zostać zamienione.



Rysunek 6.4: Zamiana bramek w szablonie identyczności

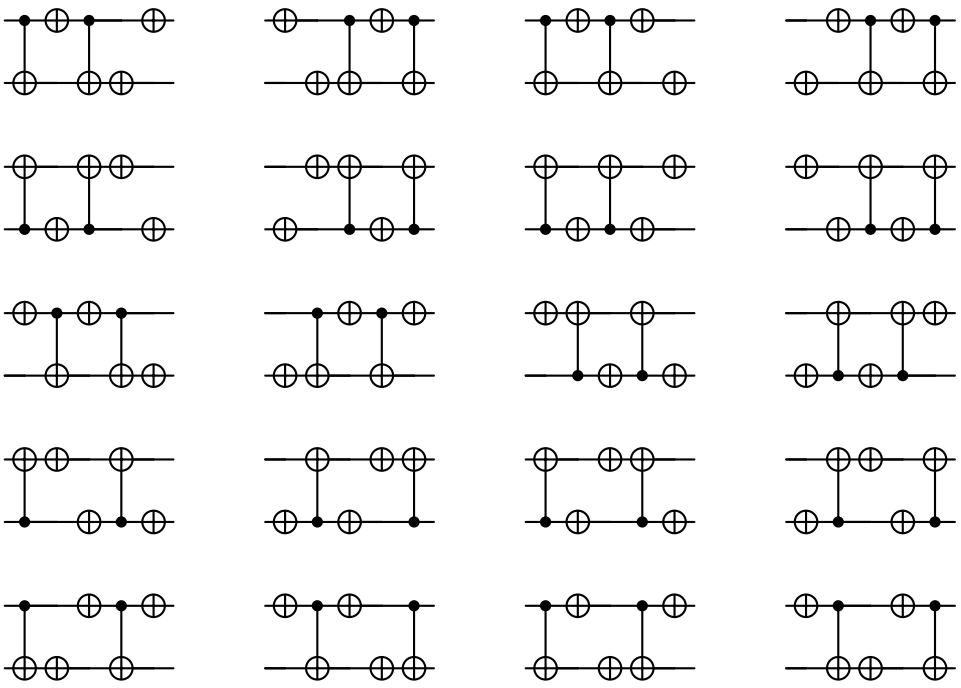
Dla większości układów, więcej niż jedna para bramek może zostać ze sobą zamieniona, a dodatkowo zamiana bramek może spowodować powstanie nowej takiej pary. Z tego powodu, do pełnego przejrzenia przestrzeni takich układów, możliwych do wygenerowania przez zamiany bramek, należało zaimplementować odpowiedni algorytm przeszukiwania, np. `DFS`. Algorytm ten umożliwił rekurencyjne odnalezienie wszystkich takich szablonów.

Zbierając wcześniej opisane metody, złożenie ich może zostać wykorzystane w celu wygenerowania całej klasy szablonów identyczności. Algorytm zwraca klasę abstrakcji szablonów identyczności, które w relacji ze sobą są wtedy i tylko wtedy, gdy jeden z nich może zostać uzyskany z drugiego przy użyciu kombinacji wcześniej opisanych przekształceń. Kolejność przekształceń została wybrana doświadczalnie tak, aby wynik algorytmu 2 zwrócony został w najkrótszym czasie.

Algorithm 2 Algorytm generowania szablonów `generate`

Require: c_t
1: $C := \{c_t\}$
2: **for** $c_i \in C$ **do**
3: $\quad C := C \cup DFS(c_i)$
4: **end for**
5: **for** $c_i \in C$ **do**
6: $\quad C := C \cup ROTATE(c_i)$
7: **end for**
8: **for** $c_i \in C$ **do**
9: $\quad C := C \cup MIRROR(c_i)$
10: **end for**
11: **for** $c_i \in C$ **do**
12: $\quad C := C \cup PERMUTE(c_i)$
13: **end for**
14: **return** C

Wynikiem uruchomienia tego algorytmu na układzie z rysunku 6.1 jest przedstawiona w rysunku 6.5 klasa abstrakcji układów, złożona z 20 elementów.



Rysunek 6.5: Klasa abstrakcji szablonu identyczności

Jak widać na rysunku 6.5, procedura generowania szablonów jest w stanie znacząco zwiększyć liczbę otrzymywanych szablonów, na każdy rozwiązany problem spełnialności. Dla najprostszego, nietrywialnego przykładu szablonu może zostać on użyty do utworzenia klasy zawierającej 20 elementów, a dla bardziej złożonych

szablonów, korzyść wynikająca z tej procedury będzie znacznie większa. w szczególności, wszystkie szablony rozmiaru 2×5 mogą zostać odnalezione przez rozwiązanie jednego problemu spełnialności i rozwinięcie otrzymanego układu.

Ponieważ algorytm generuje kompletną klasę abstrakcji, nie jest istotne, który jej element zostanie odnaleziony. Wykonanie algorytmu na dowolnym elemencie z danej klasy zwróci tę samą klasę.

6.1.3 Usprawniony algorytm syntezy szablonów

Włączając tę procedurę, naiwny algorytm kompletnej syntezy szablonów wygląda następująco:

Algorithm 3 Rozszerzony algorytm kompletnej syntezy szablonów identyczności

Require: $\mathtt{W}, \mathtt{D}$

1: $C = \{\}$
2: **while** True **do**
3: $\quad c = \mathtt{synth}(\mathtt{W}, \mathtt{D}, id, C)$
4: $\quad$ **if** $c = \bot$ **then**
5: $\quad\quad$ **return** C
6: $\quad$ **else**
7: $\quad\quad [c] = \mathtt{generate}(c)$
8: $\quad\quad C := C \cup [c]$
9: $\quad$ **end if**
10: **end while**

Dodatkowo, zakładając, że istnieje n szablonów, o rozmiarach $W \times D$, i najmniejsza klasa abstrakcji szablonów o tych wymiarach zawiera k elementów, metoda ta umożliwia zredukowanie liczby rozwiązywanych problemów do liczby mniejszej niż $\frac{n}{k}$, a dodatkowo pozwala ominąć co najmniej $k - 1$ ostatnich iteracji algorytmu naiwnego 1, które są najbardziej złożone w związku z rosnącymi ograniczeniami problemów w kolejnych iteracjach.

W przeciwieństwie do syntezy nowych obwodów z użyciem SAT-solverów, generowanie jest prostą procedurą, która korzysta z właściwości szablonów do tworzenia nowych. Przyspieszenie w czasie syntezy zostało zaprezentowane w tabeli 6.1

Wymiary	Alg. Naiwny	Alg. Rozszerzony	Przyspieszenie
2×6	1103ms	73ms	$\times$15.1
2×7	2113ms	84ms	$\times$25.2
3×6	1087s	8.49s	$\times$128.0
3×7	5468s	8.61s	$\times$635.1

Tabela 6.1: Porównanie algorytmów kompletnej syntezy szablonów

Jak widać, optymalizacja ta jest kluczowym elementem algorytmu, umożliwiającym zakończenie go w czasie o rzędy wielkości mniejszym niż w przypadku algorytmu naiwnego.

6.1.4 Świadkowie suboptymalności

Motywacją do utworzenia tej koncepcji jest spostrzeżenie, iż szablony identyczności znalazły zastosowanie do optymalizacji już istniejących układów. Podejście to zostało opisane po raz pierwszy w pracy D. Millera [64]. Procedura ta może zostać do polepszania rozmiaru już znanych układów dla funkcji, których optymalne implementacje nie są znane.

Zaproponowana w tym badaniu definicja świadków suboptymalności ma na celu wykorzystanie zalet szablonów identyczności w metodach syntezy optymalnej. Jednak tym razem nie do optymalizacji i zmniejszenia układów, jak ma to miejsce w przypadku metod heurystycznych, a do przyspieszenia uzyskiwania wyników.

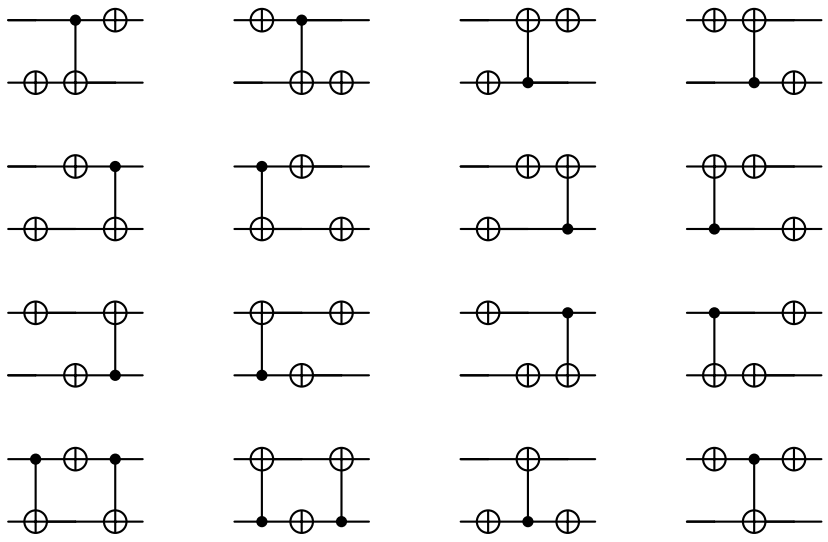
W rozdziale 5 opisana została metoda generowania problemu spełnialności, którego rozwiązanie zwraca układ implementujący zadaną funkcję i o zadanych wymiarach. Do tak wygenerowanej formuły możliwe jest dodanie nowych ograniczeń, aby wykluczały pewne rozwiązania, jak zostało to opisane w tym rozdziale. Można zauważyć, że wykluczyć można nie tylko całe obwody będące rozwiązaniami, ale też pewne kombinacje bramek z docelowego rozwiązania.

Na rysunku 6.1 prezentowany układ podzielony został na dwie części, które z dokładnością do odbicia realizują tę samą funkcję i różnią się rozmiarem. Można zauważyć, że dla dowolnej funkcji jej optymalna implementacja nie może zawierać sekwencji bramek z większej części nr 1, ponieważ mogłaby ona być mniejszą częścią nr 2, otrzymując mniejszy układ, co jest sprzeczne z założeniem optymalności.

W związku z tym z każdej formuły użytej do syntezy optymalnych układów możemy jawnie wskazać, aby rozwiązanie nie posiadało takiej sekwencji bramek. Oczywiście, takie ograniczenie jest domniemanie zawarte w takiej formule w związku z tym, że reprezentuje układ optymalny, jednakże dodanie takiego ograniczenia w sposób bezpośredni może skutkować przyspieszeniem obliczeń przez SAT-solver.

Definicja 23. *Świadek suboptymalności – dowolna sekwencja bramek, która może zostać zastąpiona mniejszą sekwencją, realizującą tę samą funkcję.*

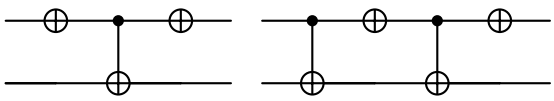
Dla każdego szablonu identyczności o rozmiarach $\mathtt{W} \times \mathtt{D}$, może zostać utworzony świadek suboptymalności złożony z jego $\left\lfloor \frac{\mathtt{D}}{2} \right\rfloor + 1$ pierwszych bramek. Liczba ta jest najmniejszą wartością większą od połowy rozmiaru szablonu, co powoduje, że pozostała część musi być mniejsza, a zatem układ złożony z tych bramek jest zastępowalny. Dodatkowo, należy zauważyć, że im mniejszy ciąg bramek jest wykluczony, tym bardziej restrykcyjne jest takie ograniczenie, dlatego też wybrany został najmniejszy rozmiar przekraczający połowę. Na rysunku 6.6 przedstawiono kompletny zbiór świadków utworzonych z klasy szablonów 6.5.



Rysunek 6.6: Wybrana klasa świadków suboptymalności

Na podstawie tej klasy można utworzyć 16 unikatowych świadków o rozmiarze 2×3. 4 szablony z tej klasy tworzą powtarzające się kopie świadków. Łącznie istnieje 22 świadków o rozmiarach 2×3, z czego pozostałych 6 może zostać uzyskanych z szablonów o rozmiarze 2×4.

Należy zauważyć, że świadkowie rozmiaru $\mathtt{W} \times \mathtt{D}$ mogą zostać utworzeni zarówno z szablonów o rozmiarze $(2\mathtt{W}-2) \times \mathtt{D}$ oraz $(2\mathtt{W}-1) \times \mathtt{D}$. Kolejną właściwością świadków jest ich redundancja, istnieje możliwość, aby jeden świadek zawierał się w innym, wtedy świadek o większym rozmiarze jest zbędny, gdyż wykluczenie mniejszego podciągu automatycznie wykluczy też większy podciąg. Na rysunku 6.7 przedstawiono przypadek dwóch takich świadków, gdzie większy układ po prawej jest zbędny.



Rysunek 6.7: Redundancja świadków suboptymalności

6.2 Wyniki badań

W ramach badania udało się znaleźć pełną przestrzeń szablonów identyczności o rozmiarach do 6×7. Następnie na podstawie tego zbioru znaleziono kompletną przestrzeń unikalnych świadków suboptymalności o rozmiarach do 6×4. Liczby poszczególnych układów przedstawiono w tabelach 6.2 oraz 6.3.

D\W	1	2	3	4	5	6
1	0	0	0	0	0	0
2	1	4	24	176	1540	13757
3	0	0	0	0	0	0
4	1	34	348	3296	33220	323273
5	0	20	240	2400	24800	244600
6	1	360	13104	269744	4626800	66741377
7	0	504	28644	674352	12343240	185127880

Tabela 6.2: Unikalne szablony identyczności

D\W	1	2	3	4	5	6
1	0	0	0	0	0	0
2	1	4	12	32	80	192
3	0	22	252	1952	12720	75072
4	0	10	1992	50028	840300	11715270

Tabela 6.3: Unikalni świadkowie suboptymalności

Opracowana procedura nie wyklucza syntezy przestrzeni szablonów identyczności większych rozmiarów. Obliczenia zostały zatrzymane ze względu na ograniczenia czasowe. Zwiększenie rozmiaru zwiększa wykładniczo zarówno rozmiar przestrzeni do wyznaczenia, jak i czas rozwiązywania poszczególnych problemów spełnialności wymaganych do określenia przestrzeni. Maksymalna głębokość została celowo ograniczona liczbą nieparzystą, ponieważ zarówno szablony głębokości $2n-2$ jak i $2n-1$ wymagane są do znalezienia kompletu świadków suboptymalności o głębokości n. Wyznaczenie przestrzeni wymiarów 6×7 zajęło około 112 godzin i oszacowano, że zwiększenie któregokolwiek z wymiarów o 1 skutkowałoby czasem syntezy zwiększonym do co najmniej kilku miesięcy. W związku z ograniczeniami czasowymi oraz dostępu do mocy obliczeniowych postanowiono badania zakończyć na tych wymiarach.

6.3 Dalsze kierunki rozwoju

Podobnie jak w przypadku badania opisanego w rozdziale 5 dt. optymalnych implementacji skrzynek podstawieniowych, tutaj również można zaobserwować dwie ścieżki rozwoju, pierwszą wynikającą z ciągłego usprawniania `SAT-solverów`, a drugą z polepszenia metod reprezentacji problemu syntezy jako problemu spełnialności. Zarówno rozwój w jednym, jak i w drugim zakresie może pozwolić na wyznaczenie przestrzeni szablonów o większych rozmiarach.

Trzecią ścieżką rozwoju badań jest zbadanie postawionej hipotezy o zasadności użycia świadków suboptymalności w celu przyspieszenia metod syntezy formalnej dowolnych układów odwracalnych. Jeżeli zostaną zaobserwowane przesłanki o tym, że ich użycie daje wymierne korzyści, to w dalszej kolejności należy podjąć próby badania doboru odpowiedniego zbioru takich świadków, który przyniesie największe przyspieszenie czasu syntezy.

Rozdział 7

Korelacja właściwości kryptograficznych

Celem badania jest zbadanie związku pomiędzy rozmiarami optymalnych implementacji funkcji nielinowych jako układy odwracalne `MCT`, a standardowymi właściwościami bezpieczeństwa kryptograficznego. W celu przeprowadzenia analizy tego zjawiska, zbadana została kompletna przestrzeń 3 i 4-bitowych skrzynek podstawieniowych pod kątem następujących właściwości:

- `#G` – Rozmiar optymalnej implementacji jako układ `MCT`;
- `AD` – Stopień algebraiczny *(z j. ang. algebraic degree)*;
- `ND` – Stopień nieliniowości *(z j. ang. nonlinearity degree)*;
- `SAC` – Ścisłe kryterium lawinowe *(z j. ang. strict avalanche criterion)* [88];
- `DDT` – Tablica rozkładu różniczek *(z j. ang. differential distribution table)* [98];
- `LAT` – Tab. aproksymacji liniowych *(z j. ang. linear approximation table)* [99].

W oparciu o te wyniki została zbadana korelacja pomiędzy optymalnym rozmiarem implementacji a pozostałymi właściwościami. Podczas analizy obszaru badawczego nie znaleziono publikacji, w których problem ten został sformułowany, a tym bardziej zbadany. Poniższe badanie należy traktować jako próbę rozstrzygnięcia, czy można zaobserwować nowe, niezbadane zjawisko, jakim jest korelacja, a nie próbę wyjaśnienia powodów tego zjawiska.

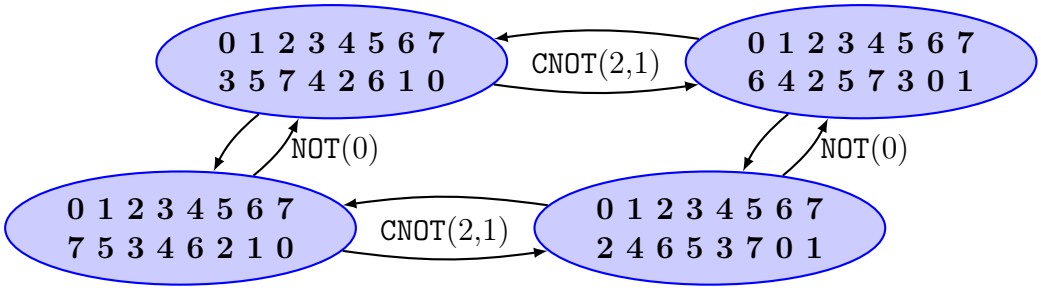
7.1 Metoda badawcza

7.1.1 Przegląd przestrzeni skrzynek podstawieniowych

Pierwszym istotnym wyzwaniem tego badania był przegląd pełnej przestrzeni skrzynek. Istnieje dokładnie 2^n różnych skrzynek podstawieniowych rozmiaru n. Dla skrzynek rozmiaru 3, skutkuje to przestrzenią o rozmiarze $8! = 40320$, a dla rozmiaru $4 - 16! \approx 2^{44.25}$. Biorąc pod uwagę tak dużą przestrzeń w drugim przypadku, synteza każdego układu z osobna jest nieosiągalna.

W celu rozwiązania problemu syntezy układów dla wszystkich funkcji należących do przestrzeni wykorzystana została metoda opisana w pracy O. Golubitskiego [81] oparta o algorytm grafowy przeglądu wszerz `BFS` *(z j. ang. breadth-first search)*.

Algorytm `BFS` wymaga określenia grafu, który ma zostać przeszukany. W przypadku tego badania, wierzchołkami grafu są permutacje 8 lub 16 elementów. Wierzchołki połączone są ze sobą wtedy i tylko wtedy, gdy istnieje bramka `MCT`, która pozwala przekształcić jedną permutację w drugą. Jako że wszystkie bramki `MCT` są samoodwracalne, łatwo zauważyć, że wszystkie krawędzie w grafie są dwustronne.



Rysunek 7.1: Wycinek grafu permutacji

Algorytm przeszukiwania grafu wszerz rozpoczyna się od permutacji identyczności, w dalszej kolejności przeglądane są wszystkie permutacje, które można utworzyć poprzez dodanie jednej bramki `MCT`. Z odwiedzonych układów tworzone są wszystkie układy dwubramkowe itd. Każda permutacja przy pierwszym odwiedzeniu zostaje oznaczona jako sprawdzona, jeżeli zostanie odwiedzona ponownie, to jest wtedy pomijana. Dzięki takiemu przejściu przez graf otrzymane zostają optymalne rozmiary dla całej przestrzeni permutacji. Algorytm kończy się po przejrzeniu wszystkich permutacji.

W takiej wersji algorytm ten jest wystarczający do przejrzenia przestrzeni permutacji 3-bitowych. W przypadku 4. bitów, algorytm ten musiałby przejrzeć aż $2^{44.25}$ permutacji. Z założenia algorytm `BFS` wymaga śledzenia odwiedzonych wierzchołków grafu. Przyjmując, że na każdy wierzchołek przypisany zostanie pojedynczy bit do oznaczenia, czy został odwiedzony, wymagałoby to użycia $2^{44.25}$ bitów lub około 2.38 TiB pamięci.

Aby zaistniała możliwość skorzystania z tego algorytmu do przejrzenia całej przestrzeni skrzynek 4 bitowych, w pracy Golubitskiego [81] został wprowadzony podział funkcji na klasy abstrakcji. Do każdej klasy abstrakcji należą funkcje równoważne co do kolejności bitów wejściowych i wyjściowych, a także co do odwrotności. Dla 4 bitów daje to 24 permutacje wejść i wyjść, a dokładając odwrotności funkcji, w każdej klasie może znajdować się nawet 48 funkcji. Z powodu symetrii niektórych funkcji, większość klas w rzeczywistości zawiera mniej elementów.

Biorąc pod uwagę, że klasy są w rzeczywistości mniejsze niż 48 elementów, a także potrzebę efektywnego kodowania klas jako indeksy naturalne, w rzeczywistości możliwe jest zredukowanie wymaganej przestrzeni pamięci do $21 * 14!$ bitów, czyli 213 GiB. Daje to redukcję pamięci o czynnik ≈ 11.43.

Kolejnym etapem przyspieszenia działania algorytmu była implementacja wersji zrównoleglonej. Ostatecznie algorytm został uruchomiony przy użyciu 64 rdzeni znajdujących się na dwóch procesorach w obrębie jednego urządzenia obliczeniowego.

Do techniki optymalizacji opisanych w oryginalnej pracy zostały dodane inne techniki, które dalej polepszyły wydajność obliczeń:

- Transformacje pomiędzy postaciami permutacji – Oryginalny artykuł opisuje metodę reprezentowania permutacji jako 64 bitowa liczba całkowita bez znaku. Liczba dzielona jest na 16 słów, po 4 bity każde. Kolejne słowa 4-bitowe reprezentują wartości permutacji dla argumentów $0, 1, \ldots, 15$. Reprezentacja taka daje efektywne przechowywanie permutacji w kolejce zadań algorytmu przeszukiwania. Dodatkowo korzystając z manipulacji bitowych istnieje możliwość wykonywania operacji bezpośrednio na takiej postaci. Jak się jednak okazuje, eksperymentalnie dowiedziono, że bardziej opłacalne jest transformowanie permutacji z postaci liczbowej to postaci tablicy bajtów, wykonywanie operacji w tej postaci, a następnie transformowanie powrotem do postaci liczby. Wy-

nika to z faktu, że w postaci tablicy lepiej używane są operacje wektorowe ze zbioru instrukcji AVX procesora, natomiast pomiędzy dwom transformacjami wykonywana jest wystarczająca liczba operacji, tak aby ich realizacja byłą opłacalna.

- Wykorzystanie wzorca puli wątków – Algorytm `BFS`, oparty o kolejkę zadań, gdzie zadaniami są kolejne wierzchołki do przejrzenia, jest idealnym kandydatem do zrównoleglenia w oparciu o wzorzec projektowy puli wątków. Każdy z wątków odczytuje zadania (wierzchołki) z jednej kolejki, wyznacza wierzchołki sąsiednie i wpisuje je ponownie do kolejki jako nowe zadania do wykonania. Dzięki temu, każdy wątek wykonuje identyczne zadanie, co skutkuje dobrym wykorzystaniem ich zasobów.

- Własne implementacje struktur danych – Na potrzeby implementacji algorytmu, utworzone zostały własne wersje dwóch struktur danych - zbioru oraz kolejki. W przypadku zrównoleglonej wersji algorytmu `BFS`, wszystkie wątki muszą odwoływać się do współdzielonej kolejki zadań oraz do zbioru odwiedzonych wierzchołków. Użycie standardowych implementacji tych struktur powoduje, że wątki muszą długo oczekiwać na dostęp do nich. Zmodyfikowana wersja struktur umożliwia dostęp do nich wielu wątkom jednocześnie.

  W przypadku naiwnej implementacji tych struktur, ponad 80% wszystkich wątków oczekiwało na dostęp do zablokowanej struktury w każdej chwili. Po dodaniu nowych struktur, mniej niż 1% czasu obliczeń wszystkich wątków marnowany był na oczekiwanie na dostęp do struktur współdzielonych.

Po dodaniu opisanych technik optymalizacji, czas przeszukania przestrzeni funkcji 4-bitowych został zmniejszony do $3,5$ dnia w porównaniu z 13 dniami wskazanymi w oryginalnej pracy.

7.1.2 Badane właściwości funkcji

- `AD` – **Stopień algebraiczny**– Niech f będzie funkcją z $\mathbb{B}^n$ do $\mathbb{B}$. Każda funkcja może zostać zapisana jako unikalny wielomian n zmiennych $x_0, x_1 \ldots, x_{n-1}$ nad $GF(2)$. Wielomian ten nazywany jest też wielomianem Żegalkina[100] lub

algebraiczną formą normalną *(z j. ang. Algebraic Normal Form)*.

$$\begin{aligned} f(x) = a \ &\oplus a_0 x_0 \oplus a_1 x_1 \oplus a_{01} x_0 x_1 \oplus a_2 x_2 \oplus \\ &\oplus a_{02} x_0 x_2 \oplus \cdots \oplus a_{01\cdots n-1} x_0 x_1 \cdots x_{n-1} \end{aligned} \tag{7.1}$$

Dla danej funkcji f, stopniem algebraicznym $\mathtt{AD}(f)$ nazywa się największą długość termu, przy którym współczynnik jest niezerowy, a w przypadku $GF(2)$, gdy term ten jest równy jeden $a_X = 1$.

Dla n-bitowej funkcji odwracalnej $\mathtt{S}$, złożonej z n boolowskich funkcji częściowych n zmiennych $\mathtt{S}_i$, zdefiniowane zostają dwie miary stopnia algebraicznego:

- **Maksymalny St. Algebraiczny**: $\mathtt{AD}_{\mathtt{max}}(\mathtt{S}) = \max_{i=0}^{n-1} \mathtt{AD}(\mathtt{S}_i)$
- **Minimalny St. Algebraiczny**: $\mathtt{AD}_{\mathtt{min}}(\mathtt{S}) = \min_{i=0}^{n-1} \mathtt{AD}(\mathtt{S}_i)$

Kolejno jest to największy i najmniejszy stopień algebraiczny występujący pośród funkcji częściowych funkcji odwracalnej.

Dla dowolnej funkcji $\mathtt{S}$ zachodzą ograniczenia:

$$1 \leq \mathtt{AD}_{\mathtt{min}}(\mathtt{S}) \leq \mathtt{AD}_{\mathtt{max}}(\mathtt{S}) \leq n - 1 \tag{7.2}$$

Jest możliwe, aby funkcja n zmiennych posiadała stopień 0 lub n, nie jest to jednak możliwe dla funkcji częściowych z racji, że muszą one być zbalansowane. Przypadek stopnia 0 wymaga funkcji stałej, a przypadek stopnia n wymaga, aby funkcja przyjmowała wartość 1 dla nieparzystej liczby wejść.

W przypadku ogólnym pożądane jest, aby kryptograficznie bezpieczne skrzynki podstawieniowe posiadały jak największe wartości stopnia algebraicznego.

- $\mathtt{ND}$ – **Nieliniowość** – Niech f będzie funkcją boolowską n zmiennych, $\mathbb{F}_n$ będzie zbiorem wszystkich funkcji boolowskich n zmiennych, a $\mathbb{A}_n \subset \mathbb{F}_n$ będzie zbiorem wszystkich afinicznych funkcji boolowskich n zmiennych.

$$\mathbb{A}_n = \{f \in \mathbb{F}_n : \mathtt{AD}(f) \leq 1\} \subset \mathbb{F}_n \tag{7.3}$$

Zdefiniowany zostaje dystans Δ pomiędzy dwoma funkcjami boolowskimi tego samego rozmiaru:

$$\Delta(g,h) = |\{x \in \mathbb{B}^n : g(x) \neq h(x)\}| \tag{7.4}$$

Stopień nieliniowości funkcji boolowskiej n zmiennych f jest zdefiniowany jako najmniejszy możliwy dystans do jednej z funkcji afinicznych tego samego rozmiaru:

$$\mathtt{ND}(f) = \min_{g \in \mathbb{A}_n} \Delta(f,g) \tag{7.5}$$

Dla n-bitowej funkcji odwracalnej $\mathtt{S}$, złożonej z n boolowskich funkcji częściowych n zmiennych $\mathtt{S}_i$, zdefiniowane zostają dwie miary nieliniowości:

- **Maksymalny st. nieliniowości**: $\mathtt{ND}_{\mathtt{max}}(\mathtt{S}) = \max_{i=0}^{n-1} \mathtt{ND}(\mathtt{S}_i)$
- **Minimalny st. nieliniowości**: $\mathtt{ND}_{\mathtt{min}}(\mathtt{S}) = \min_{i=0}^{n-1} \mathtt{ND}(\mathtt{S}_i)$

Dla dowolnej funkcji boolowskiej f n zmiennych, stopień nieliniowości ograniczony jest przedziałem:

$$0 \leq \mathtt{ND}(f) \leq 2^{n-1} - 2^{\frac{n}{2}-1}$$

Górna granica tego przedziału jest niemożliwa do osiągnięcia dla funkcji częściowych. Osiągnięcie tej wartości możliwe jest wyłącznie dla funkcji bent[101], które są niezbalansowane. Doświadczalnie wyznaczono, że dla funkcji częściowych rozmiaru 3 i 4 stopnie nieliniowości są nie większe niż kolejno 2 i 4.

W przypadku ogólnym pożądane jest, aby kryptograficznie bezpieczne skrzynki podstawieniowe posiadały jak największe wartości stopnia algebraicznego.

- $\mathtt{SAC}$ – **Ścisłe Kryterium Lawinowe** – Dla każdej funkcji boolowskiej f zdefiniowany jest poboczny współczynnik $\mathtt{sac}(f)$, który oznacza liczbę par wejść, różniących się dokładnie jednym bitem (odległość Hamminga $\mathtt{HD}$ pomiędzy nimi wynosi 1), że dla obu z tych argumentów wartości funkcji f są różne.

$$\mathtt{sac}(f) = |\{(x,y) : f(x) \neq f(y) \wedge x,y \in \mathbb{B}^n \wedge \mathtt{HD}(x,y) = 1\}\}| \tag{7.6}$$

Ponieważ liczba wszystkich par słów, różniących się jednym bitem, wynosi $n2^n$, współczynnik ten jest ograniczony przedziałem:

$$0 \leq \mathtt{sac}(\mathtt{S}_i) \leq n2^n \tag{7.7}$$

Dla bezpiecznej skrzynki podstawieniowej pożądane jest, aby przy losowo i jednostajnie wybranym wejściu, zmiana dowolnego bitu skutkowała zmianą wartości funkcji z prawdopodobieństwem $\frac{1}{2}$, a także aby prawdopodobieństwa te były niezależne pomiędzy funkcjami częściowymi. Współczynnik poboczny $\mathtt{sac}(\mathtt{S}_i)$ sprawdza wszystkie $n2^n$ możliwych przypadków wejść i zmiany wartości pojedynczych bitów, dla danej funkcji $\mathtt{S}_i$, w związku z tym, w przypadku idealnym pożądane jest, aby współczynnik poboczny był jak najbardziej zbliżony do wartości $\frac{1}{2}n2^n$.

Właściwy współczynnik ścisłego kryterium lawinowego $\mathtt{SAC}(f)$ zdefiniowany jest w pracy E. Dawsona [88] jako odległość współczynnika pobocznego `sac` od pożądanej wartości $\frac{1}{2}n2^n$:

$$\mathtt{SAC}(f) = \left|\mathtt{sac}(f) - \frac{n}{2}2^n\right| \tag{7.8}$$

Analogicznie, zdefiniowane zostają dwie miary badające jakość spełnienia ścisłego kryterium lawinowego przez skrzynkę podstawieniową $\mathtt{S}$:

- **Maksymalny współczynnik** `SAC`: $\mathtt{SAC}_{\mathtt{max}}(\mathtt{S}) = \max_{i=0}^{n-1} \mathtt{SAC}(\mathtt{S}_i)$
- **Minimalny współczynnik** `SAC`: $\mathtt{SAC}_{\mathtt{min}}(\mathtt{S}) = \min_{i=0}^{n-1} \mathtt{SAC}(\mathtt{S}_i)$

Dla bezpiecznych skrzynek podstawieniowych pożądane jest, aby współczynniki `SAC` były możliwie najmniejsze.

- `DDT` – **Tablica dystrybucji różniczek** – Dla dowolnej skrzynki podstawieniowej zdefiniowana zostaje macierz (tablica) rozmiaru $2^n \times 2^n$, która zawiera całkowite i nieujemne współczynniki, nazywana tablicą dystrybucji różniczek. Dla dowolnego indeksu wiersza i kolumny $\alpha, \beta \in \mathbb{B}^n$ współczynnik $D_{\alpha,\beta}$ tej tablicy jest równy:

$$D_{\alpha,\beta} = |\{x \in \mathbb{B}^n : \mathtt{S}(x) \oplus \mathtt{S}(x \oplus \alpha) = \beta \}| \tag{7.9}$$

Na podstawie tej konstrukcji tablicy zdefiniowana zostaje miara jakości skrzynki podstawieniowej:

– **Maksymalny współczynnik** DDT: $\mathtt{DDT}_{\mathtt{max}}(\mathtt{S}) = \max_{\alpha,\beta \in \mathbb{B}^n \setminus \{0\}} D_{\alpha,\beta}$

Od bezpiecznych skrzynek podstawieniowych oczekiwane jest, aby wszystkie współczynniki $D_{\alpha,\beta}$ miały jak najbardziej zbliżone do siebie wartości, a ponieważ suma tych współczynników jest stała, to aby maksymalny współczynnik $\mathtt{DDT}_{\mathtt{max}}$ był możliwie najmniejszy.

- LAT – **Tablica przybliżeń liniowych** – Dla dowolnej skrzynki podstawieniowej zdefiniowana zostaje macierz (tablica) rozmiaru $2^n \times 2^n$, która zawiera całkowite i nieujemne współczynniki, nazywana tablicą aproksymacji liniowych. Dla dowolnego indeksu wiersza i kolumny $\alpha, \beta \in \mathbb{B}^n$ współczynnik $\mathtt{lat}_{\alpha,\beta}$ tej tablicy jest równy:

$$\mathtt{lat}_{\alpha,\beta} = |\{x \in \mathbb{B}^n : \alpha \cdot x \ = \beta \cdot \mathtt{S}(x)\}| \tag{7.10}$$

Dla bezpiecznej skrzynki podstawieniowej i dowolnej wartości α oraz β, przy jednostajnie losowo wybranej wartości x równanie $\alpha \cdot x \ = \beta \cdot \mathtt{S}(x)$ powinno być spełnione z prawdopodobieństwem $\frac{1}{2}$.

Na tej podstawie zostaje zdefiniowana tablica odchyleń od wartości oczekiwanej, której współczynniki to:

$$\mathtt{LAT}_{\alpha,\beta} = \left| \mathtt{lat}_{\alpha,\beta} - \frac{1}{2}2^n \right| \tag{7.11}$$

Zdefiniowana zostaje miara jakości skrzynki w oparciu o właściwości LAT:

– **Maksymalny współczynnik** LAT: $\mathtt{LAT}_{\mathtt{max}}(\mathtt{S}) = \max_{\alpha,\beta \in \mathbb{B}^n \setminus \{0\}} L_{\alpha,\beta}$

Analogicznie jak w przypadku współczynnika DDT, oczekiwane jest, aby maksymalny współczynnik LAT był możliwie najmniejszy, co oznacza niskie odchylenia od wartości oczekiwanych w całości tablicy.

Właściwość	Ograniczenia 3×3		Ograniczenia 4×4		Pożądane wartości
	Dolne	Górne	Dolne	Górne	
$\texttt{AD}_{\texttt{max}}$	1	2	1	3	$\nearrow$
$\texttt{AD}_{\texttt{min}}$	1	2	1	3	$\nearrow$
$\texttt{ND}_{\texttt{max}}$	0	2	0	4	$\nearrow$
$\texttt{ND}_{\texttt{min}}$	0	2	0	4	$\nearrow$
$\texttt{SAC}_{\texttt{max}}$	0	12	0	32	$\searrow$
$\texttt{SAC}_{\texttt{min}}$	0	4	0	16	$\searrow$
$\texttt{DDT}_{\texttt{max}}$	2	8	4	16	$\searrow$
$\texttt{LAT}_{\texttt{max}}$	2	4	4	8	$\searrow$

Tabela 7.1: Podsumowanie badanych właściwości skrzynek podstawieniowych

Podsumowując, tabela 7.1 przedstawia ograniczenia dla wszystkich wartości miar, w przypadku skrzynek 3-bitowych oraz 4-bitowych. Dodatkowo tablica wskazuje, czy pożądane jest, aby dana miara była jak najmniejsza lub jak największa, mając na uwadze bezpieczeństwo kryptograficzne.

Dla każdej skrzynki podstawieniowej `S` zdefiniowana zostaje krotka złożona z 8 elementów $[\texttt{AD}_{\texttt{max}}, \texttt{AD}_{\texttt{min}}, \ldots, \texttt{LAT}_{\texttt{max}}]$. Krotka ta nazywana będzie też *profilem* skrzynki podstawieniowej. Dla skrzynki podstawieniowej rozmiaru 3 kryptograficznie najlepszym możliwym kryptograficznie jest $[2, 2, 2, 2, 0, 0, 2, 2]$, a dla rozmiaru 4 jest to $[3, 3, 4, 4, 0, 0, 4, 4]$.

7.1.3 Optymalizacja wyznaczania właściwości

Należy zauważyć, że dla wszystkich badanych właściwości skrzynki podstawieniowej, które różnią się z dokładnością do permutacji funkcji częściowych, będą miały identyczne profile. Dla stopnia algebraicznego, stopnia nieliniowości oraz kryterium lawinowego, wartości przypisane poszczególnym funkcjom częściowym zostaną między sobą zamienione, natomiast wartość minimalna i maksymalna nie ulegnie zmianie.

W przypadku tablic `DDT` oraz `LAT`, permutacja funkcji częściowych powoduje zmianę kolejności kolumn i wierszy w macierzy, a zatem jedynie zmianę uporządkowania współczynników macierzy, co nie wpływa na wielkość wartości maksymalnej.

Biorąc pod uwagę, że algorytm `BFS` do przeszukiwania przestrzeni skrzynek operuje na klasach abstrakcji skrzynek, równoważnych co do permutacji funkcji częściowych i inwersji, dla każdej klasy wystarczające jest wyznaczenie profilu nie więcej

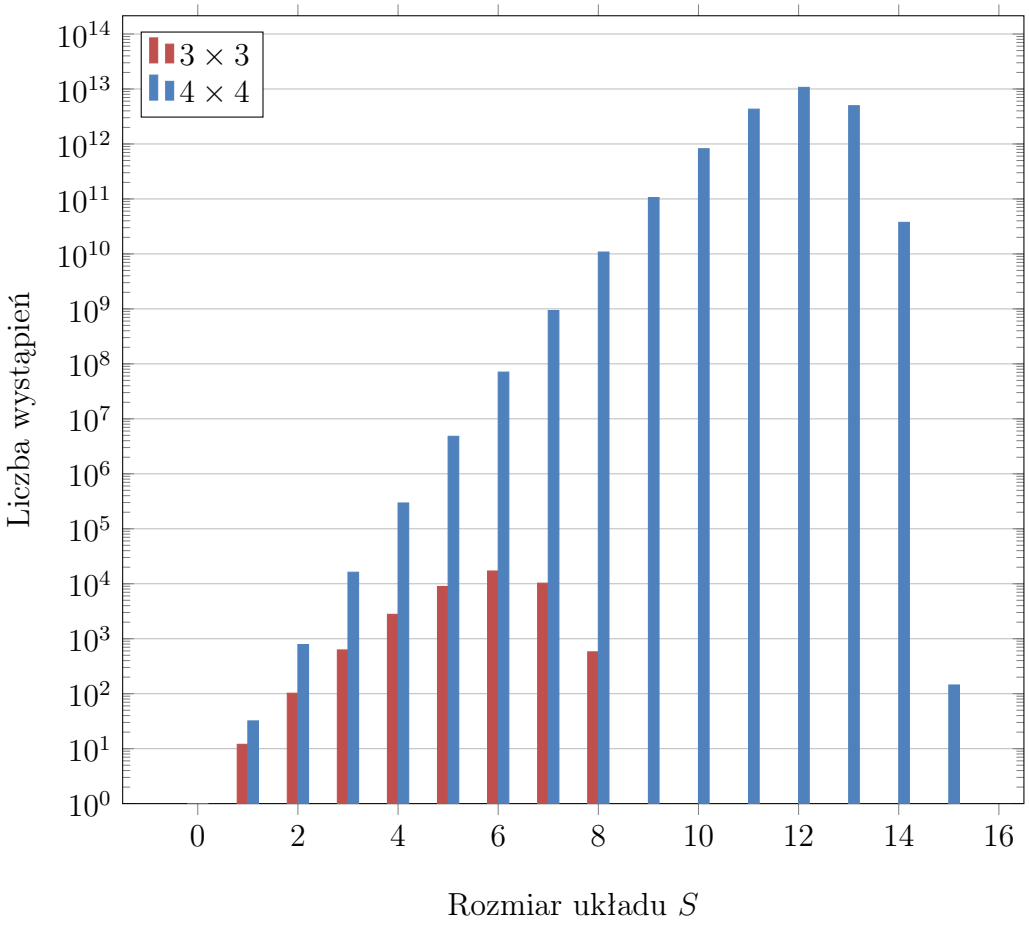
niż dwa razy. Po raz pierwszy dla reprezentanta tej klasy, a po raz drugi dla jego odwrotności, ale tylko wtedy, gdy jego odwrotność nie może zostać utworzona poprzez permutację funkcji częściowych.

7.2 Wyniki badań

Wszystkie skrzynki podstawieniowe zostały zbadane pod kątem ich profilu oraz rozmiaru – `S`. Zliczono liczbę wystąpień skrzynek o zadanej charakterystyce. Następnie została zbadana korelacja pomiędzy rozmiarem a poszczególnymi właściwościami bezpieczeństwa kryptograficznego.

7.2.1 Analiza pełnej przestrzeni skrzynek

Po zmierzeniu rozmiarów układów utworzono histogram reprezentujący rozkład rozmiarów wszystkich skrzynek podstawieniowych o szerokościach 3×3 oraz 4×4.



Rysunek 7.2: Histogram rozkładu rozmiarów układów odwracalnych

Należy zwrócić uwagę na logarytmiczną skalę pionowej osi histogramu. Dodatkowo, na histogramie znajdują się dwa słupki o niedostrzegalnym rozmiarze równym 1 przy argumencie 0, które oznaczają wystąpienie funkcji identyczności, których implementacje mają zerowy rozmiar.

Na podstawie histogramu wyznaczono statystyki rozkładu rozmiarów układów dla przestrzeni skrzynek 3×3 oraz 4×4:

- `A` – Średni rozmiar układu – 5.87 oraz 11.94 odpowiednio.
- `D` – Dominujący rozmiar układu – 6 oraz 12 odpowiednio.

Zaobserwowano, że niewiele mniej niż połowa układów posiada rozmiar dominujący, a zarówno rozmiar $\texttt{D} + 1$ jak i $\texttt{D} - 1$ spotykane są z prawdopodobieństwem zbliżonym do 1/4 każdy. Oznacza to, że razem około $90 - 95\%$ układów mieści się w przedziale $\texttt{D} \pm 1$.

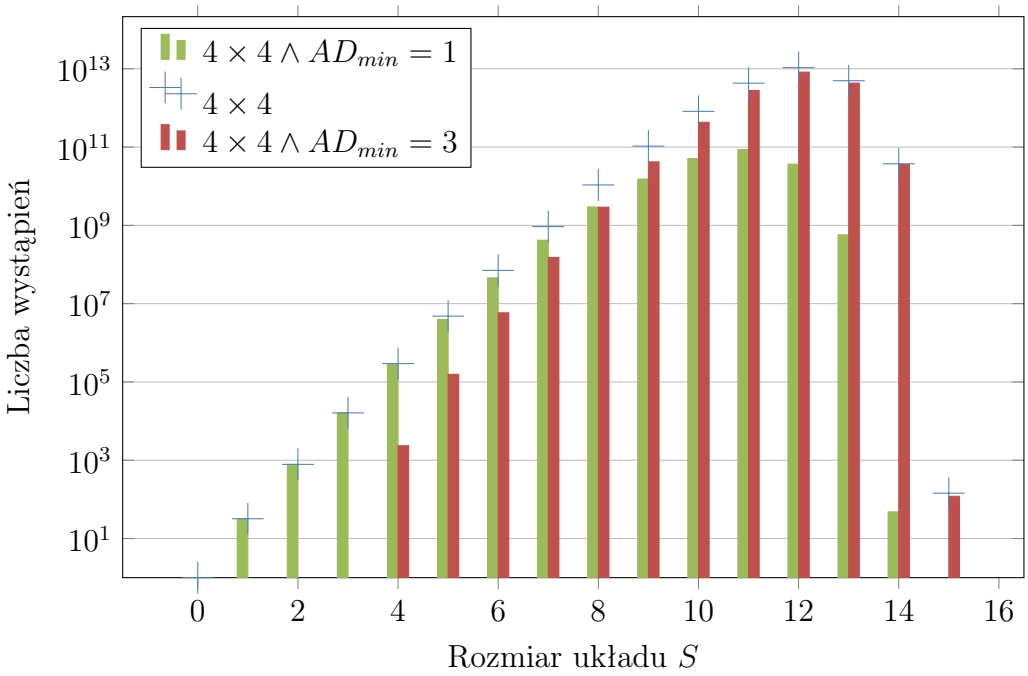
W związku z tym, poszukując *szczególnie* małych albo dużych układów, należy skupić się na poszukiwaniu układów poza *typowym* przedziałem i odpowiednio poniżej albo powyżej przedziału $\texttt{D} \pm 1$.

Kierując się powyższą obserwacją, dla histogramu rozmiarów zostały wyznaczone następujące statystyki:

- $\texttt{P}_\texttt{m}$ – Prawdopodobieństwo typowego rozmiaru – Prawdopodobieństwo, że rozmiar układu `#G` mieści się przedziale $\texttt{D} \pm 1$. Dla skrzynek 3-bitowych jest to 89.9% oraz 95.3% dla 4-bitowych.
- $\texttt{P}_\texttt{l}$ – Prawdopodobieństwo obniżonego rozmiaru – Prawdopodobieństwo, że rozmiar układu jest mniejszy od typowego ($\texttt{\#G} < \texttt{D} - 1$). Dla skrzynek 3-bitowych jest to 8.7% oraz 4.5% dla 4-bitowych.
- $\texttt{P}_\texttt{h}$ – Prawdopodobieństwo podwyższonego rozmiaru – Prawdopodobieństwo, że rozmiar układu jest większy od typowego ($\texttt{\#G} > \texttt{D} + 1$). Dla skrzynek 3-bitowych jest to 1.4% oraz 0.18% dla 4-bitowych.

7.2.2 Rozkłady warunkowe

Rozkład rozmiarów układów może zostać zbadany warunkowo, przy ustaleniu jednej z miary bezpieczeństwa na wybraną wartość. Rysunek 7.3 przedstawia histogram układów 4×4 po ustaleniu $\texttt{AD}_{\texttt{min}}$, kolejno na wartość 1 i 3, oraz porównanie tych rozkładów z rozkładem bezwarunkowym.



Rysunek 7.3: Histogram rozkładu rozmiarów układów odwracalnych

Na histogramie 7.3 widoczny jest wpływ ustalenia warunku na rozkład wielkości układów. Widoczna jest też różnica pomiędzy wpływem tych ograniczeń. Ograniczenie $\mathtt{AD}_{\mathtt{min}} = 1$ nie zmniejsza znacząco liczby wystąpień dla układów o niskich rozmiarach, natomiast znacząco wpływa na występowanie układów o większych rozmiarach. W przypadku ograniczenia $\mathtt{AD}_{\mathtt{min}} = 3$ wpływ ten jest odwrotny.

Statystyki rozkładów $\mathtt{A}, \mathtt{D}, \mathtt{P}_{\mathtt{m}}, \mathtt{P}_{\mathtt{l}}, \mathtt{P}_{\mathtt{h}}$ również mogą zostać wyznaczone warunkowo. Następnie ich wielkość może zostać porównana z wielkościami bezwarunkowymi, co daje możliwość wnioskowania o korelacji pomiędzy właściwościami bezpieczeństwa a rozmiarem.

Dodatkowo, w celu lepszego zobrazowania wpływu ograniczeń dla rozkładów warunkowych wyznaczane są trzy charakterystyki, których celem jest sprawdzenie, czy dane ograniczenie ułatwia odnajdywanie mniejszych lub większych układów.

- $\mathtt{G}_{\mathtt{l}}$ – Przyrost p-stwa. zaniżonego rozmiaru – Stosunek warunkowej wartości $\mathtt{P}_{\mathtt{l}}$ do bezwarunkowej wartości $\mathtt{P}_{\mathtt{l}}$.
- $\mathtt{G}_{\mathtt{m}}$ – Przyrost p-stwa. typowego rozmiaru – Analogicznie do $\mathtt{G}_{\mathtt{l}}$.
- $\mathtt{G}_{\mathtt{h}}$ – Przyrost p-stwa. zawyżonego rozmiaru – Analogicznie do $\mathtt{G}_{\mathtt{l}}$.

7.2.3 Stopień algebraiczny

Tabela 7.2 oraz 7.3 przedstawiają statystyki rozkładu rozmiarów pod warunkiem ustalenia właściwości $\mathtt{AD}_{\mathtt{max}}$ i $\mathtt{AD}_{\mathtt{max}}$ na dopuszczalne wartości:

S-box	3×3			4×4			
$\mathtt{AD}_{\mathtt{max}}$	Dowolna	1	2	Dowolna	1	2	3
A	5.87	4.47	5.91	11.94	6.88	9.45	11.94
D	6	5	6	12	7	10	12
$\mathtt{P}_{\mathtt{l}}$	.087	.477	.074	.045	.001	.871	.045
$\mathtt{P}_{\mathtt{m}}$	.899	.523	.911	.953	.000	.129	.953
$\mathtt{P}_{\mathtt{h}}$	.014	.000	.015	.002	.000	.000	.002
$\mathtt{G}_{\mathtt{l}}$	–	5.46	0.85	–	22.33	19.44	1.00
$\mathtt{G}_{\mathtt{m}}$	–	0.58	1.01	–	0.00	0.14	1.00
$\mathtt{G}_{\mathtt{h}}$	–	0.00	1.03	–	0.00	0.00	1.00

Tabela 7.2: Wpływ ustalenia wartości $\mathtt{AD}_{\mathtt{max}}$ na rozkład rozmiarów.

S-box	3×3			4×4			
$\mathtt{AD}_{\mathtt{min}}$	Dowolna	1	2	Dowolna	1	2	3
A	5.87	5.43	6.20	11.94	10.72	11.67	12.04
D	6	6	6	12	11	12	12
$\mathtt{P}_{\mathtt{l}}$	.087	.170	.024	.045	.360	.081	.030
$\mathtt{P}_{\mathtt{m}}$	.899	.829	.952	.953	.640	.919	.968
$\mathtt{P}_{\mathtt{h}}$	.014	.001	.024	.002	.000	.000	.002
$\mathtt{G}_{\mathtt{l}}$	–	1.95	0.27	–	8.05	1.80	0.67
$\mathtt{G}_{\mathtt{m}}$	–	0.92	1.06	–	0.67	0.96	1.02
$\mathtt{G}_{\mathtt{h}}$	–	0.07	1.71	–	0.00	0.21	1.25

Tabela 7.3: Wpływ ustalenia wartości $\mathtt{AD}_{\mathtt{min}}$ na rozkład rozmiarów.

Zarówno w przypadku skrzynek 3×3 jak i 4×4 widoczna jest ściśle rosnąca relacja pomiędzy wartością właściwości $\mathtt{AD}_{\mathtt{min}}$ oraz $\mathtt{AD}_{\mathtt{max}}$ a rozmiarami skrzynek.

Ustalenie wartości $\mathtt{AD}_{\mathtt{min}}$ na maksymalną dopuszczalną zwiększa prawdopodobieństwo uzyskania układu o podwyższonym rozmiarze nawet o 70% dla skrzynek 3×3 i 25% dla skrzynek 4×4.

Analogicznie, ustalenie wartości $\mathtt{AD}_{\mathtt{max}}$ na najmniejszą możliwą zwiększa prawdopodobieństwo znalezienia małych układów nawet kilkukrotnie.

7.2.4 Stopień nieliniowości

Tabela 7.4 oraz 7.5 przedstawiają statystyki rozkładu rozmiarów pod warunkiem ustalenia właściwości $\mathtt{ND}_{\mathtt{min}}$ i $\mathtt{ND}_{\mathtt{max}}$ na dopuszczalne wartości:

S-box	3×3			4×4			
$\mathtt{ND}_{\mathtt{max}}$	Dowolna	0	2	Dowolna	0	2	4
A	5.87	4.47	5.91	11.94	6.88	11.40	11.94
D	6	5	6	12	7	12	12
$\mathtt{P}_{\mathtt{l}}$	.087	.477	.074	.045	1.00	.174	.045
$\mathtt{P}_{\mathtt{m}}$	.899	.523	.911	.953	.000	.825	.953
$\mathtt{P}_{\mathtt{h}}$	.014	.000	.015	.002	.000	.007	.002
$\mathtt{G}_{\mathtt{l}}$	–	5.46	0.85	–	22.33	3.88	1.00
$\mathtt{G}_{\mathtt{m}}$	–	0.58	1.01	–	0.00	0.87	1.00
$\mathtt{G}_{\mathtt{h}}$	–	0.00	1.03	–	0.00	0.38	1.00

Tabela 7.4: Wpływ ustalenia wartości $\mathtt{ND}_{\mathtt{max}}$ na rozkład rozmiarów.

S-box	3×3			4×4			
$\mathtt{ND}_{\mathtt{min}}$	Dowolna	0	2	Dowolna	0	2	4
A	5.87	5.43	6.20	11.94	10.72	11.91	11.98
D	6	6	6	12	11	12	12
$\mathtt{P}_{\mathtt{l}}$	.087	.170	.024	.045	.360	.048	.037
$\mathtt{P}_{\mathtt{m}}$	.899	.829	.952	.953	.640	.951	.961
$\mathtt{P}_{\mathtt{h}}$	.014	.001	.024	.002	.000	.002	.002
$\mathtt{G}_{\mathtt{l}}$	–	1.95	0.27	–	8.05	1.06	0.82
$\mathtt{G}_{\mathtt{m}}$	–	0.92	1.06	–	0.67	1.00	1.01
$\mathtt{G}_{\mathtt{h}}$	–	0.07	1.71	–	0.00	0.91	1.09

Tabela 7.5: Wpływ ustalenia wartości $\mathtt{ND}_{\mathtt{min}}$ na rozkład rozmiarów.

Analogicznie do stopnia algebraicznego widać ściśle rosnącą relację pomiędzy rozmiarem a wartością właściwości. Ustalanie $\mathtt{ND}_{\mathtt{min}}$ na wysokie wartości skutkuje polepszeniem prawdopodobieństwa znalezienia dużego układu, a ustalenie $\mathtt{ND}_{\mathtt{max}}$ na niskie wartości polepsza szanse znalezienia małego układu.

7.2.5 Ścisłe kryterium lawinowe

S-box	3×3			4×4					
$\texttt{SAC}_{\texttt{min}}$	Dow.	0	4	Dow.	0	4	8	12	16
A	5.87	6.02	5.23	11.94	11.94	11.95	11.71	10.68	8.02
D	6	6	6	12	12	12	12	11	8
$\texttt{P}_\texttt{l}$	.087	.051	.234	.045	.045	.042	.087	.401	.998
$\texttt{P}_\texttt{m}$	.899	.932	.764	.953	.953	.956	.912	.599	.002
$\texttt{P}_\texttt{h}$	.014	.017	.002	.002	.002	.002	.001	.000	.000
$\texttt{G}_\texttt{l}$	–	0.58	2.68	–	1.01	0.95	1.94	8.95	22.28
$\texttt{G}_\texttt{m}$	–	1.04	0.85	–	1.00	1.00	0.96	0.63	0.00
$\texttt{G}_\texttt{h}$	–	1.22	0.12	–	1.02	0.93	0.40	0.01	0.00

Tabela 7.6: Wpływ ustalenia wartości $\texttt{SAC}_{\texttt{min}}$ na rozkład rozmiarów.

S-box	3×3				4×4							
$\texttt{SAC}_{\texttt{max}}$	Dow.	0	4	12	Dow.	0	4	8	12	16	20	32
A	5.87	6.34	5.86	5.32	11.94	11.92	12.02	11.94	11.93	11.57	11.82	10.95
D	6	6	6	6	12	12	12	12	12	12	12	11
$\texttt{P}_\texttt{l}$	.087	.007	.090	.171	.045	.051	.035	.044	.042	.118	.055	.253
$\texttt{P}_\texttt{m}$	.899	.967	.896	.829	.953	.946	.962	.954	.957	.882	.945	.747
$\texttt{P}_\texttt{h}$	.014	.026	.014	.000	.002	.002	.003	.002	.001	.001	.000	.000
$\texttt{G}_\texttt{l}$	–	0.08	1.03	1.95	–	1.15	0.79	0.98	0.93	2.62	1.22	5.65
$\texttt{G}_\texttt{m}$	–	1.08	1.00	0.92	–	0.99	1.01	1.00	1.00	0.93	0.99	0.78
$\texttt{G}_\texttt{h}$	–	1.80	0.99	0.00	–	1.21	1.42	1.01	0.60	0.29	0.24	0.00

Tabela 7.7: Wpływ ustalenia wartości $\texttt{SAC}_{\texttt{max}}$ na rozkład rozmiarów.

W przeciwieństwie do poprzednich właściwości, $\texttt{SAC}_{\texttt{min}}$ i $\texttt{SAC}_{\texttt{max}}$ są negatywnie skorelowane z rozmiarem układów. Zmniejszenie którejkolwiek z tych wartości skutkuje zwiększeniem układu. Włączając jednak fakt, że pożądane jest, aby wartości te były możliwie najmniejsze ze względu na bezpieczeństwo, to wciąż istnieje pozytywna korelacja pomiędzy tą miarą bezpieczeństwa a rozmiarem układów.

Relacja ta nie jest ściśle monotoniczna. Pomimo iż trend jest obserwowalny dla prawie wszystkich wartości, to można zauważyć, że dochodzi do przełamania trendu dla wartości $\texttt{SAC}_{\texttt{min}} = 0$ oraz $\texttt{SAC}_{\texttt{min}} = 0$ pośród układów 4×4.

7.2.6 Tabela rozkładu różniczek

S-box	3×3				4×4						
$\mathtt{DDT}_{\mathtt{max}}$	Dow.	2	4	8	Dow.	4	6	8	10	12	16
A	5.87	6.40	5.93	5.22	11.94	12.08	12.02	11.83	11.57	10.83	9.68
D	6	7	6	6	12	12	12	12	12	11	10
$\mathtt{P}_{\mathtt{l}}$	.087	.011	.056	.219	.045	.028	.031	.061	.106	.332	.778
$\mathtt{P}_{\mathtt{m}}$	.899	.945	.939	.781	.953	.969	.967	.938	.894	.668	.222
$\mathtt{P}_{\mathtt{h}}$	.014	.044	.006	.000	.002	.003	.002	.001	.000	.000	.000
$\mathtt{G}_{\mathtt{l}}$	–	0.13	0.64	2.51	–	0.63	0.69	1.36	2.37	7.42	17.37
$\mathtt{G}_{\mathtt{m}}$	–	1.05	1.05	0.87	–	1.02	1.01	0.98	0.94	0.70	0.23
$\mathtt{G}_{\mathtt{h}}$	–	3.05	0.39	0.02	–	1.66	1.16	0.70	0.20	0.01	0.00

Tabela 7.8: Wpływ ustalenia wartości $\mathtt{DDT}_{\mathtt{max}}$ na rozkład rozmiarów.

Podobnie jak w przypadku SAC, miara $\mathtt{DDT}_{\mathtt{max}}$ jest negatywnie skorelowana z rozmiarem. Jednak w tym przypadku zależność jest ścisła. Tak jak wcześniej, uwzględniając, że wartości mniejsze są bardziej pożądane ze względu na bezpieczeństwo, to rozmiar układu jest pozytywnie skorelowany z jego jakością zmierzoną tą miarą.

7.2.7 Tabela przybliżeń liniowych

S-box	3×3			4×4			
$\mathtt{LAT}_{\mathtt{max}}$	Dow.	2	4	Dow.	4	6	8
A	5.87	6.40	5.67	11.94	11.95	11.97	11.01
D	6	7	6	12	12	12	11
$\mathtt{P}_{\mathtt{l}}$	.087	.011	.115	.045	.042	.037	.252
$\mathtt{P}_{\mathtt{m}}$	.899	.945	.881	.953	.956	.961	.748
$\mathtt{P}_{\mathtt{h}}$	.014	.044	.004	.002	.002	.002	.000
$\mathtt{G}_{\mathtt{l}}$	–	0.13	1.32	–	0.93	0.83	5.63
$\mathtt{G}_{\mathtt{m}}$	–	1.05	0.98	–	1.00	1.01	0.78
$\mathtt{G}_{\mathtt{h}}$	–	3.05	0.25	–	1.13	1.03	0.00

Tabela 7.9: Wpływ ustalenia wartości $\mathtt{LAT}_{\mathtt{max}}$ na rozkład rozmiarów.

Właściwość $\mathtt{LAT}_{\mathtt{max}}$ zachowuje się podobnie do SAC, jest negatywnie skorelowana z rozmiarami przy jednoczesnej potrzebie minimalizacji tej wartości ze względów bezpieczeństwa. Dla tej właściwości również można zaobserwować przełamanie w trendzie dla skrzynek 4×4 przy najbardziej pożądanej wartości.

7.2.8 Podsumowanie

Dla wszystkich ośmiu właściwości widać pozytywną korelację pomiędzy jakością skrzynki zmierzonej w oparciu o tę właściwość, a jej rozmiarem. W pięciu na osiem przypadków, relacja ta jest ściśle rosnąca. Ustalenie wartości na bardziej pożądaną kryptograficznie skutkuje wzrostem średniego rozmiaru skrzynki, a także zwiększeniem prawdopodobieństwa, że losowo wybrana funkcja będzie implementowana przez układ o dużym rozmiarze. Wyjątkiem są trzy właściwości, dla $\mathtt{SAC}_{\mathtt{max}}$ oraz $\mathtt{SAC}_{\mathtt{min}}$ druga najlepsza wartość skutkuje największym oczekiwanym rozmiarem i prawdopodobieństwem. Natomiast dla $\mathtt{LAT}_{\mathtt{max}}$ tylko dla oczekiwanego rozmiaru monotoniczność rozmiaru jest przełamana, prawdopodobieństwo $\mathtt{G}_{\mathtt{h}}$ jest ściśle rosnące.

Tabele 7.10 oraz 7.11 zawierają podsumowanie wpływu poszczególnych ograniczeń na wartości $\mathtt{G}_{\mathtt{h}}, \mathtt{G}_{\mathtt{l}}$. Dla każdej z właściwości została wybrana taka wartość, która powoduje największy wzrost prawdopodobieństwa uzyskania układu o zaniżonym rozmiarze oraz zawyżonym:

Lp.	Właściwość	3×3		4×4	
		Wartość	$\mathtt{G}_{\mathtt{h}}$	Wartość	$\mathtt{G}_{\mathtt{h}}$
1.	$\mathtt{DDT}_{\mathtt{max}}$	2	3.05	4	1.66
2/3.*	$\mathtt{LAT}_{\mathtt{max}}$	2	3.05	4	1.13
2/3.*	$\mathtt{SAC}_{\mathtt{max}}$	0	1.80	4	1.42
4.	$\mathtt{AD}_{\mathtt{min}}$	2	1.71	3	1.25
5.	$\mathtt{ND}_{\mathtt{min}}$	2	1.71	4	1.09
6.	$\mathtt{SAC}_{\mathtt{min}}$	0	1.22	0	1.02
7.	$\mathtt{AD}_{\mathtt{min}}$	2	1.03	3	1.00
8.	$\mathtt{ND}_{\mathtt{min}}$	2	1.03	4	1.00

*zależnie od szerokości funkcji

Tabela 7.10: Hierarchia właściwości pod względem $\mathtt{G}_{\mathtt{h}}$

Lp.	Właściwość	3×3		4×4	
		Wartość	G_1	Wartość	G_1
1*.	AD_{max}	1	5.46	1	22.33
1*.	ND_{max}	0	5.46	0	22.33
3.	SAC_{max}	4	2.68	16	22.28
4.	DDT_{max}	8	2.51	16	17.37
5.	AD_{min}	1	1.95	1	8.05
6.	ND_{min}	0	1.95	0	8.05
7.	SAC_{max}	12	1.95	32	5.65
8.	LAT_{max}	4	1.32	8	5.63

**ex-æquo* – Efektywnie oba warunki są równoważne.

Tabela 7.11: Hierarchia właściwości pod względem G_1

7.2.9 Czas realizacji

Czas badania wszystkich permutacji wraz ze wskazanymi właściwościami wzrósł do około 18 dni w porównaniu z 3.5 dniami wymaganymi do samego przejścia grafu i wyznaczeniem samych rozmiarów.

7.2.10 Szczególne profile skrzynek

Posiadając wiedzę, jak poszczególne właściwości wpływają na rozmiary układów, można zdefiniować trzy szczególne profile, które pozwolą na znalezienie układów o najbardziej pożądanych właściwościach:

- Zoptymalizowany kryptograficznie (A) – profil z najlepszymi właściwościami kryptograficznymi.
- Zmaksymalizowany rozmiar (B) – profil, który zgodnie z tabelą 7.10 powinien posiadać układy o największym oczekiwanym rozmiarze;
- Zminimalizowany rozmiar (C) – profil, który zgodnie z tabelą 7.11 powinien posiadać układy o największym oczekiwanym rozmiarze;

Szer.	Profil	$\mathtt{AD}_{\mathtt{max}}$	$\mathtt{AD}_{\mathtt{min}}$	$\mathtt{ND}_{\mathtt{max}}$	$\mathtt{ND}_{\mathtt{min}}$	$\mathtt{SAC}_{\mathtt{max}}$	$\mathtt{SAC}_{\mathtt{min}}$	$\mathtt{DDT}_{\mathtt{max}}$	$\mathtt{LAT}_{\mathtt{max}}$
3×3	A/B	2	2	2	2	0	0	2	2
	C	1	1	0	0	4	12	8	4
4×4	A	3	3	4	4	0	0	4	4
	B	3	3	4	4	4	4	4	6
	C	1	1	0	0	16	32	16	8

Tabela 7.12: Szczególne profile skrzynek

S-box	3×3			4×4			
Profil	Dowolny	A/B	C	Dowolny	A	B	C
Min. `#G`	0	4	2	0	6	6	3
Maks. `#G`	8	8	6	15	14	14	9
`A`	5.87	6.51	4.41	11.94	12.09	12.23	6.83
	–	+11%	−25%	–	+1.3%	+2.4%	−44%
`D`	6	7	5	12	12	12	7
$\mathtt{P}_{\mathtt{l}}$	.087	.005	.516	.045	.028	.013	1.00
$\mathtt{P}_{\mathtt{m}}$	.899	.943	.484	.953	.968	.982	.000
$\mathtt{P}_{\mathtt{h}}$	.014	.052	.000	.002	.004	.005	.000
$\mathtt{G}_{\mathtt{l}}$	–	0.06	5.91	–	0.62	0.29	22.33
$\mathtt{G}_{\mathtt{m}}$	–	1.05	0.54	–	1.02	1.03	0.00
$\mathtt{G}_{\mathtt{h}}$	–	3.64	0.00	–	2.22	2.72	0.00

Tabela 7.13: Wpływ wyboru szczególnego profilu na rozkład rozmiarów

Tabela 7.13 pokazuje istnienie pozytywnej korelacji pomiędzy właściwościami bezpieczeństwa, a rozmiarem optymalnej implementacji. Zarówno w przypadku skrzynek 3-bitowych, jak i 4-bitowych widoczne jest, że wybór odpowiedniego profilu kryptograficznego skrzynki podstawieniowej może skutkować zarówno zmniejszeniem, jak i zwiększeniem oczekiwanego rozmiaru.

Wyniki wskazują, że zjawisko jest znacznie silniejsze w przypadku poszukiwania małych obwodów i odpowiedni dobór profilu może zmniejszyć średni rozmiar o nawet 25% lub 44%. W przypadku poszukiwania dużych układów, oczekiwany wzrost średniej wartości rozmiaru jest nie większy niż 11% i 2.5% dla skrzynek 3×3 i 4×4 kolejno.

Pomimo, że wzrost wartości oczekiwanej nie jest istotny, szczególnie dla skrzynek 4-bitowych, to należy zwrócić uwagę na wpływ wyboru profilu na prawdopodobieństwa znalezienia się w przedziale rozmiarów zawyżonych i zaniżonych. Wybór pro-

filu przeznaczonego do poszukiwania dużych układów skutkuje spadkiem wartości $\mathtt{P_l}$ o ponad $\frac{2}{3}$ i prawie 3-krotnym wzrostem wartości $\mathtt{P_h}$ dla skrzynek 4-bitowych, a w dodatku skutkuje wykluczeniem układów złożonych z mniej niż 6 bramek.

7.3 Wnioski

Wyniki badania wskazują na pozytywną korelację pomiędzy właściwościami bezpieczeństwa skrzynek podstawieniowych, a optymalnym rozmiarem jej implementacji jako układ `MCT`. Biorąc pod uwagę, że rozmiar skrzynki również może być rozpatrywany jako właściwość bezpieczeństwa w kontekście ataków kwantowych, to wyniki te dostarczają informacji o tym jak projektować bezpieczne skrzynki podstawieniowe, zarówno pod względem ataków klasycznych jak i kwantowych.

Dodatkowo, na podstawie tych wyników, można praktycznie wnioskować o oczekiwanych rozmiarach układów dużych skrzynek podstawieniowych, dla których optymalne implementacje nie zostały znalezione, np. o 8-bitowej skrzynce szyfru `AES`. Korzystając z istniejącej korelacji możliwe jest wnioskowanie, że tego rodzaju skrzynki, które zostały zaprojektowane z myślą o spełnieniu najwyższych standardów bezpieczeństwa, nie powinny posiadać prostej implementacji, a co za tym idzie być również bezpieczne w aspekcie kwantowym.

7.4 Dalsze kierunki rozwoju

Jak wcześniej zostało wskazane. Badanie to jest pierwszym etapem nowo postawionego problemu jakim jest korelacja między właściwościami funkcji odwracalnej. Badanie pozwoliło zaobserwować, że taka korelacja zachodzi i jest ona niezależna od wybranej próby funkcji, gdyż zbadano ją na pełnej przestrzeni funkcji odwracalnych rozmiarów 3 i 4. Naturalnym dalszym krokiem rozwoju tego badania jest rozszerzenie do funkcji większych rozmiarów, pamiętając jednak, że już dla funkcji rozmiaru 5 może nie być możliwe zbadanie pełnej przestrzeni, a jedynie pewnej próby. Kolejnym etapem realizacji tego badania może być próba zbadania przyczyn tej korelacji, a w szczególności dowiedzenie jej w ogólnym przypadku, jeżeli jest to możliwe.

Rozdział 8

Atak na szyfr SPARKLE

Celem badania jest wyznaczenie kosztu przeprowadzenia ataku na szyfry lekkie z rodziny SPARKLE [17] w modelu ze znanym tekstem jawnym opisanym w roz. 4.

W badaniu przeprowadzona została analiza kosztu implementacji wyroczni kwantowej realizującej funkcję szyfrującą. Wynikiem tego badania jest układ odwracalny realizujący ten szyfr, zoptymalizowany pod kątem liczebności użytych bramek NCT i głębokości oraz dokładnie wyznaczony koszt tego układu.

Rodzina szyfrów SPARKLE została zgłoszona do konkursu NIST na kryptografię lekką – *"NIST Lightweight Cryptography Call"* [7] i po pozytywnym przejściu dwóch rund konkursu dostała się do finału wraz z siedmioma innymi kandydatami [9], w którym została odrzucona [20].

Szyfry z tej grupy oparte są o konstrukcję gąbki *(z j. ang. sponge)* i są szyframi z mechanizmem uwierzytelniania poprzez tag – *(z j. ang. Authenticated Encryption with Associated Data - AEAD)*. Do rodziny SPARKLE należą 4 warianty szyfru:

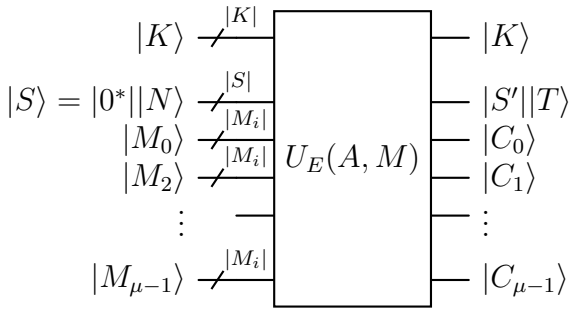
Wariant	$\|K\|, \|T\|$	$\|N\|, \|A_i\|, \|M_i\|, \|C_i\|$	$\|S\|$
SCHWAEMM128-128	128	128	256
SCHWAEMM192-192	192	192	384
SCHWAEMM256-128	128	256	384
SCHWAEMM256-256	256	256	512

Tabela 8.1: Warianty szyfrów z rodziny SPARKLE

- $|K|$ – rozmiar klucza K;
- $|N|$ – rozmiar wartości jednorazowej N *(z j. ang. nonce)*;
- $|T|$ – rozmiar tagu uwierzytelnienia T;
- $|A_i|$ – rozmiar bloku A_i danych dodatkowych A;
- $|M_i|$ – rozmiar bloku M_i wiadomości M;
- $|C_i|$ – rozmiar bloku C_i szyfrogramu C;
- $|S|$ – rozmiar wewnętrznego stanu funkcji szyfrującej S.

8.1 Opracowany układ

Układ, który został opracowany w badaniu, ma postać spójną z wymaganiami opisanymi w rozdziale 4 do przeprowadzenia ataku ze znanym tekstem jawnym w oparciu o algorytm Grover'a i wygląda następująco:



Rysunek 8.1: Wyrocznia szyfrująca `SPARKLE`

Układ jest parametryczny i przyjmuje dwa argumenty:

- Dane dodatkowe A, podzielone na α bloków A_i o rozmiarze $|A_i| \in \{128, 192, 256\}$;
- Wiadomość M, podzieloną na μ bloków M_i o rozmiarze $|M_i| \in \{128, 192, 256\}$;

Operuje na trzech grupach rejestrów:

- Rejestr klucza K – Przetrzymuje niezmieniony klucz przez cały proces;
- Stan wewnętrzny szyfru S – $|S|$-bitowy rejestr zainicjowany zerami i wartością jednorazową N. Po zakończeniu fragment tego rejestr zawiera tag uwierzytelniania T. Pozostałe $|S| - |T|$ bitów oznaczone jest symbolem S';
- Rejestry szyfrogramu – μ $|M_i|$-bitowych rejestrów, które rozpoczynają w stanach równych wartościom bloków wiadomości M_i. Po zakończeniu szyfrowania zawierają μ bloków szyfrogramu C_i.

8.2 Permutacja główna

Głównym elementem szyfru jest $|S|$-bitowa permutacja p, która jest odwracalną funkcją boolowską $p : \mathbb{B}^{|S|} \to \mathbb{B}^{|S|}$ i skonstruowana jest na bazie sieci podstawieniowo-przestawieniowej. Składa się z następujących operacji:

- Warstwa dodawania stałych wartości rundowych - p_C;
- Warstwa podstawieniowa - p_S
- Warstwa przestawieniowa - p_L;

Permutacja występuje w sześciu wariantach – $p^7_{256}, p^7_{384}, p^8_{512}, p^{10}_{256}, p^{11}_{384}, p^{12}_{512}$. Indeks dolny oznacza szerokość stanu głównego, na jakim operuje, indeks górny oznacza liczbę iteracji wszystkich warstw w danym wariancie. Koszt permutacji jest równy sumie kosztów poszczególnych warstw pomnożonej przez liczbę iteracji.

Każdy z czterech wariantów szyfru korzysta z dwóch wariantów permutacji, które odpowiadają rozmiarowi jego stanu głównego, np. `SCHWAEMM128-128`, którego stan główny ma rozmiar 256, korzysta z permutacji p^7_{256} i p^{10}_{256}. Zgodnie z dokumentacją, dla każdego wariantu szyfru wariant permutacji z mniejszą liczbą iteracji nazywany jest *"small step"*, a z większą *"big step"*.

8.2.1 Warstwa dodawania stałych wartości rundowych

Operacja polega na dodaniu wartości stałej do $|S|$-bitowego stanu przy użyciu operacji `C-XOR`. w każdej rundzie dodawany do stanu jest indeks rundy r oraz inna wartość stała c_r. Koszt tej operacji jest równy wadze Hamminga indeksu rundy $HW(r)$ i wartości stałej $HW(c_r)$ w bramkach `NOT`.

Indeks r	Wartość c_r	$HW(r)$	$HW(c_r)$	Razem
0	`0xB7E15162`	0	16	16
1	`0xBF715880`	1	15	16
2	`0x38B4DA56`	1	16	17
3	`0x324E7738`	2	16	18
4	`0xBB1185EB`	1	17	18
5	`0x4F7C7B57`	2	21	23
6	`0xCFBFA1C8`	2	19	21
7	`0xC2B3293D`	3	16	19

Tabela 8.2: Wartości stałe permutacji `SPARKLE`

Dla uproszczenia obliczeń, zostanie przyjęte ograniczenie górne liczby bramek dla tej operacji, równe maksymalnej wartości obecnej w tabeli, czyli 23.

Operacja	IO	ANC	W	#G	#NOT	#CNOT	#Toff	D
p_C	32	0	32	23	23	0	0	1

Tabela 8.3: Koszt warstwy dodawania stałych w permutacji SPARKLE

8.2.2 Warstwa podstawieniowa

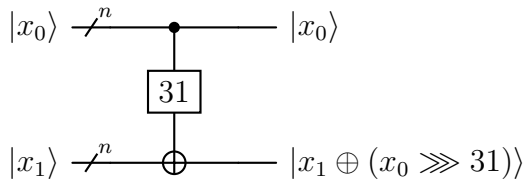
Celem warstwy podstawieniowej jest zapewnienie właściwości nieliniowych w szyfrze oraz konfuzji. Elementem kluczowym tej warstwy jest skrzynka ARX – Alzette przekształcająca dwa słowa 32-bitowe. Skrzynka jest oparta o zestaw operacji ARX omówionych w rozdziale 3.4.

Tabela 8.4 przedstawia podsumowanie kosztów przekształceń ARX uwzględniając rozmiary operacji w skrzynce Alzette. Koszt operacji dodawania w miejscu dwóch 32-bitowych rejestrów został zaczerpnięty z pracy [63], na ten moment jest to najmniejsza implementacja pod względem liczby bramek i głębokości. Operacja C-XOR w skrzynce Alzette korzysta z tego samego zestawu wartości stałych co warstwa dodawania wartości stałych, w związku z tym również przyjęty został koszt tej operacji wynoszący 23 bramki NOT.

Operacja	IO	ANC	W	#G	#NOT	#CNOT	#Toff	D
Add	$2 * 32$	53	117	439	62	123	254	22
C-XOR	32	0	32	23	23	0	0	1
R-XOR	$2 * 32$	0	64	32	0	32	0	1

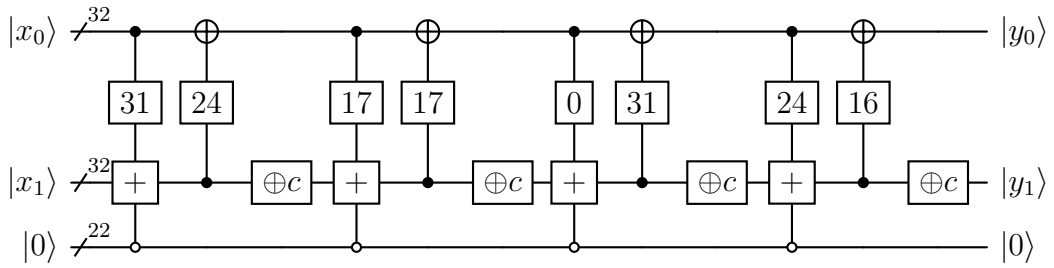
Tabela 8.4: Koszt operacji ARX

Na potrzeby badania, używana będzie notacja operacji rotowanych. Notacja oznacza, że jedno ze słów jeszcze przed wykonaniem operacji zostaje obrócone bitowo. Rotacja w tym przypadku oznacza, że qubity do operacji używane są w innej kolejności, w związku z tym koszt takiej operacji jest identyczny jak w przypadku oryginalnym.

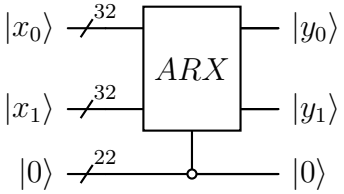


Rysunek 8.2: Przykład bramki rotowanej

Skrzynka `ARX Alzette`, jest parametryczna i przyjmuje jeden argument - wartość stałą c_i. Pomijając rotacje, wszystkie operacje w tym działaniu odbywają się sekwencyjnie na dwóch wyrazach i nie mogą zostać wykonane równolegle. Z tego powodu zarówno liczba bramek, jak i głębokość układu skrzynki `ARX` jest sumą bramek i głębokości kolejnych operacji niebędących rotacjami. Implementacja skrzynki korzysta z rejestru pomocniczego do wykonania dodawań. Stan rejestru pomocniczego jest niezmieniony po wykonaniu skrzynki. Rysunek 8.3 przedstawia układ implementujący przekształcenie.



Rysunek 8.3: Układ implementujący skrzynkę `ARX Alzette`



Rysunek 8.4: Blok skrzynki `ARX Alzette`

Koszt implementacji skrzynki `ARX` oraz warstwy podstawieniowej dla permutacji o szerokości $256, 384, 512$ przedstawiono w tabeli 8.5.

Operacja	IO	ANC	W	#G	#NOT	#CNOT	#Toff	D
Alzette	$2 * 32$	53	117	1976	340	620	1016	96
$p_{S,256}$	256	212	468	7904	1360	2480	4064	96
$p_{S,384}$	384	318	702	11856	2040	3720	6096	96
$p_{S,512}$	512	414	926	15808	2720	4960	8128	96

Tabela 8.5: Koszt warstwy podstawieniowej permutacji SPARKLE

8.2.3 Warstwa przestawieniowa

Celem warstwy przestawieniowej jest zapewnienie dyfuzji w permutacji SPARKLE. Elementem odpowiedzialnym za to jest permutacja Λ, która jest operacją liniową. Występuje w trzech wariantach $\Lambda_4, \Lambda_6, \Lambda_8$, które operują kolejno na $4, 6$ lub 8 par słów 32-bitowych. Dla permutacji Λ_4 są to pary $(x_0, y_0), \ldots, (x_3, y_3)$. Liczba par słów w permutacji jest oznaczana jako parametr L. Każda z permutacji złożona jest z przesunięć, rotacji i operacji R-XOR.

Do wykonania permutacji, niezbędne są dwa rejestry pomocnicze t_x oraz t_y o rozmiarze 32 qubitów, każdy rozpoczynający w stanie $|0\rangle$. Zgodnie z opisem w dokumentacji szyfru, dla $l = L/2$, permutacja rozpoczyna się od przypisania do rejestrów pomocniczych wartości:

$$\begin{aligned} t_x &\leftarrow t_x \oplus x_0 \oplus x_1 \oplus \cdots \oplus x_{l-1} \\ t_y &\leftarrow t_y \oplus y_0 \oplus y_1 \oplus \cdots \oplus y_{l-1} \end{aligned} \tag{8.1}$$

Operacja ta wymaga wykonania $2*l = L$ działań R-XOR na rejestrach o rozmiarze 32, koszt jej zatem wynosi $32 * 2 * l = 32 * L$ bramek CNOT. Dodatkowo, obie wartości wyznaczane są niezależnie, operacje te można zatem wykonać w l warstwach układu.

Następnie na obu rejestrach pomocniczych wykonywana jest operacja zamiany Feistela zgodnie z opisem przedstawionym na rysunku 3.4.

$$\begin{aligned} t_x &\leftarrow FS(t_x) \\ t_y &\leftarrow FS(t_y) \end{aligned} \tag{8.2}$$

Koszt tej operacji wynosi łącznie 32 bramki CNOT w jednej warstwie układu.

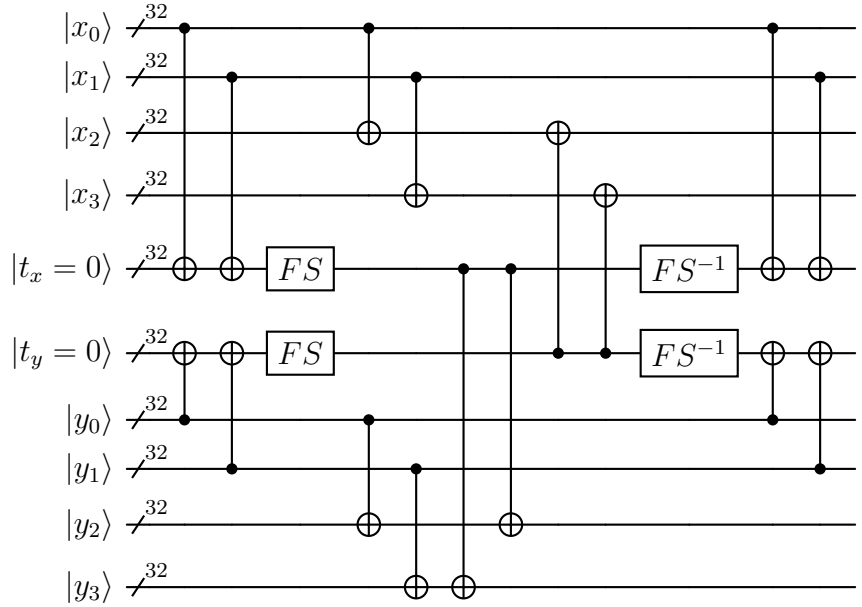
Kolejny krok, to wyznaczenie wartości wyjściowych x_i, y_i zgodnie z równaniem:

$$\begin{aligned} x_{l+i} &\leftarrow x_{l+i} \oplus x_i \oplus t_x \\ y_{l+i} &\leftarrow x_{y+i} \oplus y_i \oplus t_y \end{aligned} \tag{8.3}$$

Dla $i \in \{0, 1, \ldots l-1\}$. Wyznaczanie kolejnych wartości x_{l+i} i y_{l+i} odbywa się niezależnie i mogą one zostać obliczone równolegle. Wyznaczenie każdej z wartości wymaga użycia dwóch operacji `R-XOR` jedna po drugiej. Całkowity koszt l iteracji tej operacji, dla dwóch rejestrów rozmiaru 32, to $2 * 2l * 32 = 64L$ bramek `CNOT`, wszystkie znajdujące się w dwóch warstwach układu.

Po tym etapie, rejestry t_x oraz t_y nie są dalej używane i ich wartości muszą zostać wyzerowane. W tym celu dwa pierwsze kroki muszą zostać odwrócone. Koszt wyzerowania tych rejestrów jest identyczny jak koszt dwóch pierwszych kroków.

Ostatnim krokiem permutacji jest zamiana indeksów słów x_i oraz y_i. Ponieważ jest to jedynie zmiana indeksowania, to jej koszt jest zerowy i może zostać pominięty w tym badaniu. Rysunek 8.5 przedstawia układ implementujący permutację Λ_4.



Rysunek 8.5: Układ permutacji Λ_4

Warstwa permutacji zawsze składa się z jednej instancji permutacji Λ. Koszt wariantów warstw i permutacji został przedstawiony w tabeli 8.6

Operacja	IO	ANC	W	#G	#NOT	#CNOT	#Toff	D
$p_{L,256}(\Lambda_4)$	256	64	320	576	0	576	0	8
$p_{L,384}(\Lambda_6)$	384	64	448	832	0	832	0	10
$p_{L,512}(\Lambda_8)$	512	64	576	1088	0	1088	0	12

Tabela 8.6: Koszt warstwy przestawieniowej permutacji SPARKLE

8.2.4 Koszt permutacji

Liczby bramek w jednej iteracji są prostymi sumami liczb bramek poszczególnych jej etapów. Podobnie, głębokość iteracji jest sumą głębokości poszczególnych etapów. Zakładając, że wszystkie operacje zostają realizowane równolegle tam, gdzie jest to możliwe, to do wykonania warstwy podstawieniowej wymagane jest kolejno $4 * 53, 6 * 53$ lub $8 * 53$ qubitów pomocniczych, w zależności od wariantu. Niezależnie od opcji, permutacja Λ, która odbywa się sekwencyjnie po skrzynkach ARX wymaga mniejszej liczby qubitów pomocniczych – 64, zatem szerokość układu jest podyktowana szerokością warstwy podstawieniowej.

Operacja	IO	ANC	W	#G	#NOT	#CNOT	#Toff	D
p_{256}	256	212	468	8688	1568	3056	4064	105
p_{384}	384	318	702	12968	2320	4552	6096	107
p_{512}	512	424	936	17248	3072	6048	8128	109
p_{256}^{7}	256	212	468	59521	9681	21392	28448	735
p_{256}^{10}	256	212	468	85030	13830	30560	40640	1050
p_{384}^{7}	384	318	702	88977	14441	31864	42672	749
p_{384}^{11}	384	318	702	139821	22693	50072	67056	1177
p_{512}^{8}	512	424	936	135352	21944	48384	65024	872
p_{512}^{12}	512	424	936	203028	32916	72576	97536	1308

Tabela 8.7: Koszt iteracji permutacji głównej i jej wariantów dla szyfru SPARKLE

8.3 Operacja pomocnicze

Funkcja szyfrująca SPARKLE korzysta z dwóch pomocniczych funkcji ρ oraz $\mathcal{W}$.

8.3.1 Przekształcenie ρ

Przekształcenie ρ jest używane do wprowadzenia wiadomości M oraz danych dodatkowych A do stanu gąbki przy użyciu operacji zamiany Feistela oraz C-XOR.

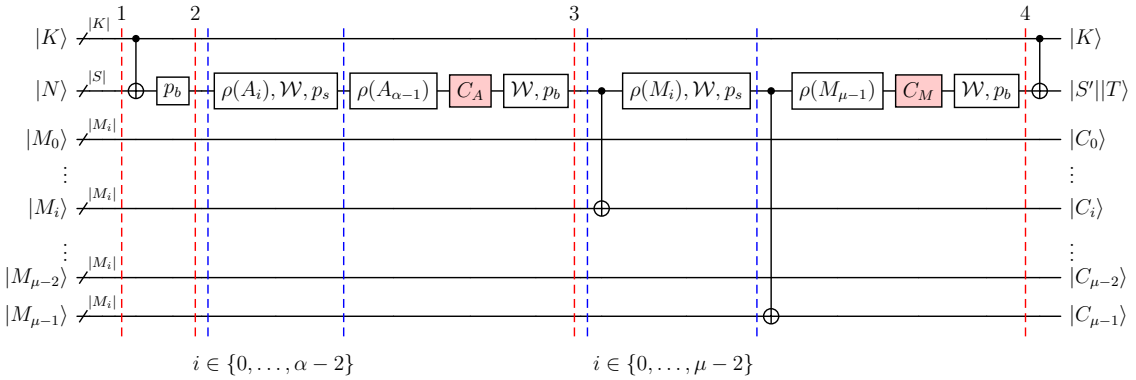
Rysunek 8.6: Przekształcenie ρ

8.3.2 Przekształcenie $\mathcal{W}$

Przekształcenie $\mathcal{W}$ wykonywane jest przed rozpoczęciem każdej permutacji SPARKLE poza pierwszą. Operacja wykonywana jest w dwóch wariantach, zależnie od wariantu szyfru. Jeżeli rozmiar klucza $|K|$ jest taki sam jak rozmiar wartości tymczasowej $|N|$, to $\mathcal{W}$ polega na dodaniu pierwszej połowy rejestru głównego S do drugiej przy użyciu operacji R-XOR. Jeżeli $|N| = 2|K|$, to rejestr główny S dzielony jest na trzy równe części, a następnie pierwsza część jest dodawana osobno do drugiej i trzeciej części operacją R-XOR.

8.4 Układ szyfrujący

Korzystając z opisu i grafu funkcji szyfrującej zawartych w dokumentacji, utworzony został następujący układ, który będzie równoważny z szyfrem SPARKLE.



Rysunek 8.7: Układ szyfrów SPARKLE

Bramki oznaczone kolorem czerwonym, na $|S|$-bitowym rejestrze stanu S oznaczają operację dodawania C-XOR wartości zapisanej w argumencie bramki. Bramki realizujące operację ρ są zależne od wartości danych dodatkowych A oraz wiadomości M i są punktem parametryzacji układu.

Bloki permutacji p_s zgodnie z dokumentacją szyfru są wariantami permutacji o mniejszej liczbie iteracji przewidzianymi dla danego wariantu szyfru – $p^7_{256}, p^7_{384}, p^8_{512}$, bloki p_b są większymi wariantami permutacji – $p^{10}_{256}, p^{11}_{384}, p^{12}_{512}$.

Czerwone przerywane linie na rysunku 8.7 oznaczają rozpoczęcie poszczególnych etapów, natomiast niebieskie przerywane linie oznaczają, że dany element układu jest powtarzany wielokrotnie dla indeksów oznaczonych poniżej układu.

Tabele 8.8, 8.9, 8.10 oraz 8.11 przedstawiają listy operacji wykonywanych przez szyfr wraz z liczbą iteracji, gdzie α oznacza liczbę bloków wiadomości dodatkowych A_i, a μ oznacza liczbę bloków wiadomości M_i.

Operacja	L. iteracji	IO	ANC	W	#G	#NOT	#CNOT	#Toff	D
1. Inicjalizacja stanu i wstępne przetwarzanie									
R-XOR K	1	2*128	0	256	128	0	128	0	1
p_{256}^{10}	1	256	212	468	85030	13830	30560	40640	1050
2. Absorpcja danych dodatkowych									
$\rho(A_i)$	α	128	0	128	192	128	64	0	2
$\mathcal{W}$(R-XOR)	α	2*128	0	256	128	0	128	0	1
p_{256}^{7}	$\alpha - 1$	256	212	468	59521	9681	21392	28448	735
C-XOR C_A	1	3	0	3	3	3	0	0	1
p_{256}^{10}	1	256	212	468	85030	13830	30560	40640	1050
3. Absorpcja wiadomości i wyciskanie szyfrogramu									
R-XOR C_i	μ	2*128	0	256	128	0	128	0	1
$\rho(M_i)$	μ	128	0	128	192	128	64	0	2
$\mathcal{W}$(R-XOR)	μ	2*128	0	256	128	0	128	0	1
p_{256}^{7}	$\mu - 1$	256	212	468	59521	9681	21392	28448	735
C-XOR C_M	1	3	0	3	3	3	0	0	1
p_{256}^{10}	1	256	212	468	85030	13830	30560	40640	1050
4. Generowanie tagu uwierzytelnienia									
R-XOR K	1	2*128	0	256	128	0	128	0	1
Razem									
Podstawa	-	384	212	596	139270	25094	49152	65024	1685
$+\alpha*$	-	-	-	-	61136	11104	21584	28448	738
$+\mu*$	-	128	-	128	61264	11104	21712	28448	740

Tabela 8.8: Koszt układu szyfrującego SCHWAEMM128-128

Operacja	L. iteracji	IO	ANC	W	#G	#NOT	#CNOT	#Toff	D
1. Inicjalizacja stanu i wstępne przetwarzanie									
R-XOR K	1	2*192	0	384	192	0	192	0	1
p_{384}^{11}	1	384	318	702	139821	22693	50072	67056	1177
2. Absorpcja danych dodatkowych									
$\rho(A_i)$	α	192	0	192	288	192	96	0	2
$\mathcal{W}$(R-XOR)	α	2*192	0	384	192	0	192	0	1
p_{384}^{7}	$\alpha - 1$	384	318	702	88977	14441	31864	42672	749
C-XOR C_A	1	3	0	3	3	3	0	0	1
p_{384}^{11}	1	384	318	702	139821	22693	50072	67056	1177
3. Absorpcja wiadomości i wyciskanie szyfrogramu									
R-XOR C_i	μ	2*192	0	384	192	0	192	0	1
$\rho(M_i)$	μ	192	0	192	288	192	96	0	2
$\mathcal{W}$(R-XOR)	μ	2*192	0	384	192	0	192	0	1
p_{384}^{7}	$\mu - 1$	384	318	702	88977	14441	31864	42672	749
C-XOR C_M	1	3	0	3	3	3	0	0	1
p_{384}^{11}	1	384	318	702	139821	22693	50072	67056	1177
4. Generowanie tagu uwierzytelnienia									
R-XOR K	1	2*192	0	384	192	0	192	0	1
Razem									
Podstawa	-	576	318	894	246782	44086	86872	115824	2031
$+\alpha*$	-	-	-	-	91256	16432	32152	42672	752
$+\mu*$	-	192	-	192	91448	16432	32344	42672	753

Tabela 8.9: Koszt układu szyfrującego SCHWAEMM192-192

Operacja	L. iteracji	IO	ANC	W	#G	#NOT	#CNOT	#Toff	D
1. Inicjalizacja stanu i wstępne przetwarzanie									
R-XOR K	1	2*128	0	256	128	0	128	0	1
p_{384}^{11}	1	384	318	702	139821	22693	50072	67056	1177
2. Absorpcja danych dodatkowych									
$\rho(A_i)$	α	256	0	256	384	256	128	0	2
$\mathcal{W}$(R-XOR)	α	3*128	0	384	256	0	256	0	2
p_{384}^{7}	$\alpha - 1$	384	318	702	88977	14441	31864	42672	749
C-XOR C_A	1	3	0	3	3	3	0	0	1
p_{384}^{11}	1	384	318	702	139821	22693	50072	67056	1177
3. Absorpcja wiadomości i wyciskanie szyfrogramu									
R-XOR C_i	μ	2*256	0	512	256	0	256	0	1
$\rho(A_i)$	μ	256	0	256	384	256	128	0	2
$\mathcal{W}$(R-XOR)	μ	3*128	0	384	256	0	256	0	2
p_{384}^{7}	$\alpha - 1$	384	318	702	88977	14441	31864	42672	749
C-XOR C_M	1	3	0	3	3	3	0	0	1
p_{384}^{11}	1	384	318	702	139821	22693	50072	67056	1177
4. Generowanie tagu uwierzytelnienia									
R-XOR K	1	2*128	0	256	128	0	128	0	1
Razem									
Podstawa	-	512	318	830	246654	44086	86744	115824	2037
$+\alpha*$	-	-	-	-	91416	16496	32248	42672	753
$+\mu*$	-	256	-	256	91672	16496	32504	42672	754

Tabela 8.10: Koszt układu szyfrującego SCHWAEMM256-128

Operacja	L. iteracji	IO	ANC	W	#G	#NOT	#CNOT	#Toff	D
1. Inicjalizacja stanu i wstępne przetwarzanie									
R-XOR K	1	2*256	0	512	256	0	256	0	1
p_{512}^{12}	1	512	424	936	203028	32916	72576	97536	1308
2. Absorpcja danych dodatkowych									
$\rho(A_i)$	α	256	0	256	384	256	128	0	2
$\mathcal{W}$(R-XOR)	α	2*256	0	512	256	0	256	0	1
p_{512}^{8}	$\alpha - 1$	512	424	936	135352	21944	48384	65024	872
C-XOR C_A	1	3	0	3	3	3	0	0	1
p_{512}^{12}	1	512	424	936	203028	32916	72576	97536	1308
3. Absorpcja wiadomości i wyciskanie szyfrogramu									
R-XOR C_i	μ	2*256	0	512	256	0	256	0	1
$\rho(M_i)$	μ	256	0	256	384	256	128	0	2
$\mathcal{W}$(R-XOR)	μ	2*256	0	512	256	0	256	0	1
p_{512}^{8}	$\mu - 1$	512	424	936	135352	21944	48384	65024	872
C-XOR C_M	1	3	0	3	3	3	0	0	1
p_{512}^{12}	1	512	424	936	203028	32916	72576	97536	1308
4. Generowanie tagu uwierzytelnienia									
R-XOR K	1	2*256	0	512	256	0	256	0	1
Razem									
Podstawa	-	768	414	1192	345478	61446	121472	162560	2184
$+\alpha*$	-	-	-	-	138624	24832	48768	65024	875
$+\mu*$	-	256	-	256	138880	24832	49024	65024	876

Tabela 8.11: Koszt układu szyfrującego SCHWAEMM256-256

8.4.1 Koszt układu

Kolejnym krokiem jest wyznaczenie kosztu wyroczni, uwzględniając liczbę wymaganych bloków wiadomości do przeprowadzenia ataku. Koszt ataku został obliczony dla trzech przypadków:

1. Bez użycia tagu uwierzytelniania – Wariant podstawowy, w którym ignorowana jest obecność tagu uwierzytelniania do zmniejszenia liczby rozwiązań. Najmniej wydajny z pośród trzech wariantów;
2. Korzystają z tagu uwierzytelniania – Wariant poprawiony, w którym użyta jest własność szyfrowania z uwierzytelnianiem tym samym kluczem do zredukowania rozmiaru wymaganej znanej wiadomości jawnej;
3. Minimalny – Wariant korzystający z minimalnej możliwej liczby bloków wiadomości, jednocześnie ryzykując kolizję kluczy.

Operacja	μ	IO	ANC	W	#G	#NOT	#CNOT	#Toff	D
SCHWAEMM128-128									
Bez tagu uw.	3	768	212	920	316217	51561	114288	150368	3901
Z tagiem uw.	2	640	212	852	256248	41752	92576	121920	3162
Minimalny	1	512	212	724	196279	31943	70864	93472	2423
$+\alpha*$	–	–	–	–	59841	9809	21584	28448	738
SCHWAEMM192-192									
Bez tagu uw.	3	1152	318	1470	510846	83102	183904	243840	4296
Z tagiem uw.	2	960	318	1278	421197	68469	151560	201168	3543
Minimalny	1	768	318	1086	331548	53836	119216	158496	2790
$+\alpha*$	–	–	–	–	89457	14633	32152	42672	752
SCHWAEMM256-128									
Bez tagu uw.	2	1280	318	1598	421517	68597	151752	201168	3545
Z tagiem uw.	2	1024	318	1342	421517	68597	151752	201168	3545
Minimalny	1	768	318	1086	331644	53900	119248	158496	2791
$+\alpha*$	–	–	–	–	89617	14697	32248	42672	753
SCHWAEMM256-256									
Bez tagu uw.	3	1536	424	1960	747642	121466	268544	357632	4812
Z tagiem uw.	2	1280	424	1704	611394	99266	219520	292608	3936
Minimalny	1	1024	424	1448	475146	77066	170496	227584	3060
$+\alpha*$	–	–	–	–	135992	22200	48768	65024	875

Tabela 8.12: Koszt wyroczni szyfrującej dla szyfrów SPARKLE

8.5 Koszt ataku

Ostatnim etapem badania jest wyznaczenie kosztu pełnego ataku w oparciu o koszt wyroczni szyfrującej oraz model ataku. w tym celu koszt wyroczni został pomnożony przez wymaganą liczbę rund algorytmu Grovera, która wynosi $\lfloor \frac{\pi}{4} * 2^{|K|/2} \rfloor$ oraz przez czynnik 2 ze względu na fakt, że każda runda wymaga wykonania wyroczni szyfrującej oraz odwrotnej wyroczni szyfrującej.

Operacja	μ	IO	ANC	W	#G	#NOT	#CNOT	#Toff	D
SCHWAEMM128-128									
Bez tagu uw.	3	768	212	920	$2^{82,92}$	$2^{80,31}$	$2^{81,45}$	$2^{81,85}$	$2^{76,58}$
Z tagiem uw.	2	640	212	852	$2^{82,62}$	$2^{80,00}$	$2^{81,15}$	$2^{81,55}$	$2^{76,28}$
Minimalny	1	512	212	724	$2^{82,23}$	$2^{79,61}$	$2^{80,76}$	$2^{81,16}$	$2^{75,89}$
$+\alpha*$	–	–	–	–	$2^{80,52}$	$2^{77,91}$	$2^{79,05}$	$2^{79,45}$	$2^{74,18}$
SCHWAEMM192-192									
Bez tagu uw.	3	1152	318	1470	$2^{115,61}$	$2^{112,99}$	$2^{114,14}$	$2^{114,55}$	$2^{108,72}$
Z tagiem uw.	2	960	318	1278	$2^{115,34}$	$2^{112,71}$	$2^{113,86}$	$2^{114,27}$	$2^{108,44}$
Minimalny	1	768	318	1086	$2^{114,99}$	$2^{112,37}$	$2^{113,51}$	$2^{113,93}$	$2^{108,10}$
$+\alpha*$	–	–	–	–	$2^{113,10}$	$2^{110,49}$	$2^{111,62}$	$2^{112,03}$	$2^{106,21}$
SCHWAEMM256-128									
Bez tagu uw.	2	1280	318	1598	$2^{83,34}$	$2^{80,72}$	$2^{81,86}$	$2^{82,27}$	$2^{76,44}$
Z tagiem uw.	2	1024	318	1342	$2^{83,34}$	$2^{80,72}$	$2^{81,86}$	$2^{82,27}$	$2^{76,44}$
Minimalny	1	768	318	1086	$2^{82,99}$	$2^{80,37}$	$2^{81,52}$	$2^{81,93}$	$2^{76,10}$
$+\alpha*$	–	–	–	–	$2^{81,10}$	$2^{78,49}$	$2^{79,63}$	$2^{80,03}$	$2^{74,21}$
SCHWAEMM256-256									
Bez tagu uw.	3	1536	424	1960	$2^{148,16}$	$2^{145,54}$	$2^{146,69}$	$2^{147,10}$	$2^{140,88}$
Z tagiem uw.	2	1280	424	1704	$2^{147,87}$	$2^{145,25}$	$2^{146,40}$	$2^{146,81}$	$2^{140,59}$
Minimalny	1	1024	424	1448	$2^{147,51}$	$2^{144,89}$	$2^{146,03}$	$2^{146,45}$	$2^{140,23}$
$+\alpha*$	–	–	–	–	$2^{145,70}$	$2^{143,09}$	$2^{144,23}$	$2^{144,64}$	$2^{138,42}$

Tabela 8.13: Koszt ataku na szyfry SPARKLE

8.6 Dalsze kierunki rozwoju

Opracowany układ z całą pewnością nie jest optymalny i zasadne jest stwierdzenie, że znalezienie optymalnego układu i dowiedzenie optymalności jest niezwykle trudne, z racji rozmiaru funkcji, jaką jest funkcja szyfrująca. W związku z tym dalsze badania nad znalezieniem mniejszego układu, który realizuje ten szyfr, są jak najbardziej możliwe do przeprowadzenia.

Zmniejszenie układu może się odbyć na kilka różnych sposobów, np. poprzez znalezienie lepszych implementacji elementów, liniowych lub nieliniowych, znalezienie bardziej efektywnych metod łączenia poszczególnych elementów lub poprzez syntezę większych elementów razem, a nie poszczególnych ich fragmentów.

Niezależnie od przyjętej strategii każda redukcja w rozmiarze implementacji zbliża ją do optymalnego rozmiaru, a co za tym idzie, określenia prawdziwego, minimalnego kosztu ataku, a nie jego górnego ograniczenia.

Rozdział 9

Atak na szyfr Ascon

Cel badania jest analogiczny do badania w rozdziale 8. W tym przypadku badaniu poddany został szyfr Ascon. Szyfr Ascon[10] został zgłoszony do konkursu NIST na standard kryptografii lekkiej – *"NIST Lightweight Cryptography Call"* [7], w którym zwyciężył i został wybrany na nowy standard w lutym 2023 [20] [21].

Podobnie jak SPARKLE oparty jest o konstrukcję gąbki i jest szyfrem z mechanizmem uwierzytelniania poprzez tag *AEAD*. Szyfr występuje w dwóch wariantach:

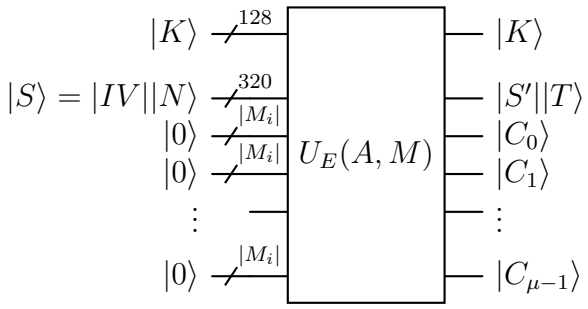
Wariant	$\|K\|, \|N\|, \|T\|$	$\|A_i\|, \|M_i\|, \|C_i\|$	$\|S\|$
Ascon-128	128	64	320
Ascon-128a	128	128	320

Tabela 9.1: Warianty szyfrów z rodziny Ascon

- $|K|$ – rozmiar klucza K;
- $|N|$ – rozmiar wartości jednorazowej N *(z j. ang. nonce)*;
- $|T|$ – rozmiar tagu uwierzytelnienia T;
- $|A_i|$ – rozmiar bloku A_i danych dodatkowych A;
- $|M_i|$ – rozmiar bloku M_i wiadomości M;
- $|C_i|$ – rozmiar bloku C_i szyfrogramu C;
- $|S|$ – rozmiar wewnętrznego stanu funkcji szyfrującej S.

9.1 Opracowany układ

Układ, który został opracowany w badaniu, ma postać spójną z wymaganiami opisanymi w rozdziale 4 do przeprowadzenia ataku ze znanym tekstem jawnym w oparciu o algorytm Grover'a i wygląda następująco:



Rysunek 9.1: Wyrocznia szyfrująca `Ascon`

Układ jest parametryczny i przyjmuje dwa argumenty:

- Dane dodatkowe A, podzielone na α bloków A_i o rozmiarze $|A_i| \in \{64, 128\}$;
- Wiadomość M, podzieloną na μ bloków M_i o rozmiarze $|M_i| \in \{64, 128\}$;

Operuje na trzech grupach rejestrów:

- Rejestr klucza K – Przetrzymuje niezmieniony klucz przez cały proces;
- Stan wewnętrzny szyfru S – 320 bitowy rejestr zainicjowany stałą wartością inicjującą IV oraz wartością jednorazową N. Po zakończeniu rejestr ten zawiera również tag uwierzytelniania T złożony z 128 bitów. Pozostałe 192 bity oznaczone są symbolem S';
- Rejestry szyfrogramu – μ $|M_i|$-bitowych rejestrów, które rozpoczynają wyzerowane. Po zakończeniu szyfrowania zawierają μ bloków szyfrogramu C_i.

9.2 Permutacja główna

Głównym elementem szyfru jest 320-bitowa permutacja p, która jest odwracalną funkcją boolowską $p : \mathbb{B}^{320} \to \mathbb{B}^{320}$ i skonstruowana jest na bazie sieci podstawieniowo-przestawieniowej. Składa się z następujących operacji:

- Warstwa dodawania stałych wartości rundowych - p_C;
- Warstwa podstawieniowa - p_S
- Warstwa przestawieniowa - p_L;

Permutacja występuje w trzech wariantach – p^6, p^8 i p^{12}. Indeks górny oznacza liczbę iteracji wszystkich warstw w danym wariancie. Koszt permutacji jest równy sumie kosztów poszczególnych warstw pomnożonej przez liczbę iteracji.

9.2.1 Warstwa dodawania stałych wartości rundowych

Operacja polega na dodaniu wartości stałej do 320-bitowego stanu przy użyciu operacji `C-XOR`. w każdej rundzie o indeksie r inna wartość c_r jest dodawana do stanu przy użyciu działania `C-XOR`. Koszt tej operacji jest równy wadze Hamminga wartości stałej $HW(c_r)$ w bramkach `NOT`.

Indeks r			Wartość c_r	$HW(c_r)$
p^{12}	p^8	p^6		
0			`0xF0`	4
1			`0xE1`	4
2			`0xD2`	4
3			`0xC3`	4
4	0		`0xB4`	4
5	1		`0xA5`	4
6	2	0	`0x96`	4
7	3	1	`0x87`	4
8	4	2	`0x78`	4
9	5	3	`0x69`	4
10	6	4	`0x5A`	4
11	7	5	`0x4B`	4

Tabela 9.2: Wartości stałe permutacji `Ascon`

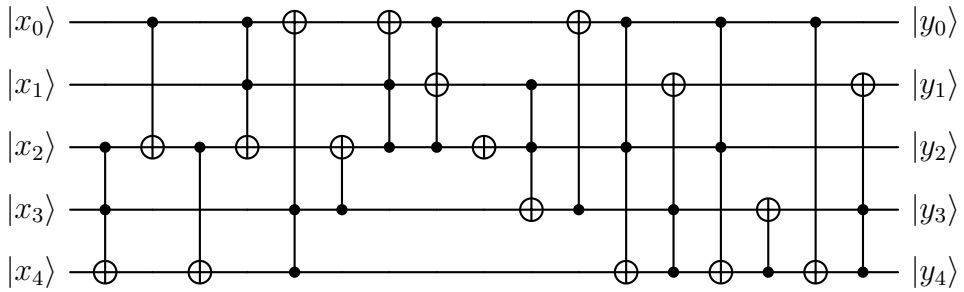
Łatwo spostrzec, że niezależnie od rundy, waga Hamminga wszystkich stałych jest taka sama. Jest to skutkiem tego, że pierwsze 4 bity stałej są dopełnieniem ostatnich 4 bitów słowa.

Operacja	IO	ANC	W	#G	#NOT	#CNOT	#Toff	D
p_C	8	0	8	4	4	0	0	1

Tabela 9.3: Koszt warstwy dodawania stałych w permutacji `Ascon`

9.2.2 Warstwa podstawieniowa

Celem warstwy podstawieniowej jest zapewnienie właściwości nieliniowych w szyfrze oraz konfuzji. Elementem kluczowym tej warstwy jest 5-bitowa skrzynka podstawieniowa. Implementacja tej skrzynki przy użyciu bramek NCT została znaleziona w badaniu opisanym w rozdziale 5.



Rysunek 9.2: Skrzynka podstawieniowa `Ascon`

Ponieważ znaleziona implementacja operuje w miejscu i nie wykorzystuje qubitów dodatkowych, to implementacja całej warstwy może zostać utworzona przy użyciu 64 równoległych układów skrzynki podstawieniowej. W konsekwencji koszt całej warstwy podstawieniowej jest równy kosztowi 64 skrzynek, a jej głębokość jest równa głębokości jednej skrzynki.

Operacja	IO	ANC	W	#G	#NOT	#CNOT	#Toff	D
Skrzynka	5	0	5	17	1	6	10	17
p_S	320	0	320	1088	64	384	640	17

Tabela 9.4: Koszt warstwy podstawieniowej permutacji `Ascon`

9.2.3 Warstwa przestawieniowa

Celem warstwy przestawieniowej jest zapewnienie dyfuzji. Polega ona na wykonaniu mnożenia przez macierz binarną 5 słów 64-bitowych, w związku z tym nie wymaga ona użycia bramek NOT oraz Toffoliego. Na moment pisania pracy, najmniejsza implementacja tej operacji w miejscu może zostać znaleziona w pracy [102]

Operacja	IO	ANC	W	#G	#NOT	#CNOT	#Toff	D
p_L	320	0	320	1595	0	1595	0	119

Tabela 9.5: Koszt warstwy przestawieniowej permutacji `Ascon`

9.2.4 Koszt permutacji

Podsumowując koszty poszczególnych warstw permutacji, można wyznaczyć koszt całości. Ponieważ żadna z warstw nie korzysta z qubitów pomocniczych, to koszt jednej iteracji permutacji jest równy sumie kosztów poszczególnych warstw, podobnie z głębokością. Koszt i głębokość każdego wariantu permutacji są równe kosztowi i głębokości jednej iteracji pomnożonych przez ich liczbę.

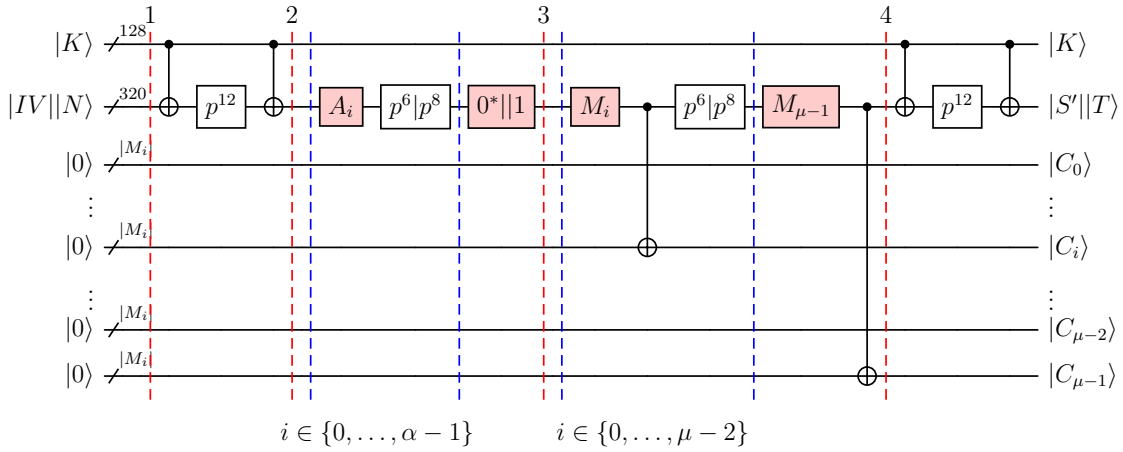
Operacja	IO	ANC	W	#G	#NOT	#CNOT	#Toff	D
p	320	0	320	2687	68	1979	640	137
p^6	320	0	320	16122	408	11874	3840	822
p^8	320	0	320	21496	544	15832	5120	1096
p^{12}	320	0	320	32244	906	23748	7680	1644

Tabela 9.6: Koszt iteracji permutacji p i jej wariantów p^6, p^8, p^{12} dla szyfru `Ascon`

Permutacja główna szyfru `Ascon` realizowana jest w pełni w miejscu i bez użycia qubitów dodatkowych. Z tego względu jest optymalna pod względem szerokości.

9.3 Układ szyfrujący

Korzystając z opisu i grafu funkcji szyfrującej zawartych w dokumentacji, utworzony został następujący układ, który będzie równoważny z szyfrem `Ascon`.



Rysunek 9.3: Układ szyfru `Ascon-128` oraz `Ascon-128a`

Bramki oznaczone kolorem czerwonym, na 320-bitowym rejestrze stanu S oznaczają operację dodawania `C-XOR` wartości zapisanej w argumencie bramki. Czerwone bramki zależne od wartości danych dodatkowych A i wiadomości M są punktem parametryzacji układu.

Bloki permutacji p^{12} zgodnie z dokumentacją szyfru są wykonywane przez oba warianty, w przypadku bloku $p^6|p^8$ realizuje on permutację p^6 dla szyfru `Ascon-128` i p^8 dla szyfru `Ascon-128a`.

Czerwone przerywane linie na rysunku 9.3 oznaczają rozpoczęcie poszczególnych etapów, natomiast niebieskie przerywane linie oznaczają, że dany element układu jest powtarzany wielokrotnie dla indeksów oznaczonych poniżej układu.

Niech μ oznacza liczbę bloków wiadomości oraz α oznacza liczbę bloków danych dodatkowych. Koszt oraz liczba iteracji każdej z operacji na każdym z etapów szyfrowania zostały przedstawione w tabeli 9.7 oraz 9.8.

Operacja	L.iteracji	IO	ANC	W	#G	#NOT	#CNOT	#Toff	D
1. Inicjalizacja i stanu wstępne przetwarzanie									
R-XOR K	1	256	0	256	128	0	128	0	1
p^{12}	1	320	0	320	32244	906	23748	7680	1644
R-XOR K	1	256	0	256	128	0	128	0	1
2. Absorpcja danych dodatkowych									
C-XOR A_i	α	64	0	64	64	64	0	0	1
p^6	α	320	0	320	16122	408	11874	3840	822
C-XOR $0^*\|\|1$	1	1	0	1	1	1	0	0	1
3. Absorpcja wiadomości i wyciskanie szyfrogramu									
C-XOR M_i	μ	64	0	64	64	64	0	0	1
R-XOR C_i	μ	128	0	128	64	0	64	0	1
p^6	$\mu - 1$	320	0	320	16122	408	11874	3840	822
4. Generowanie tagu uwierzytelnienia									
R-XOR K	2	256	0	256	128	0	128	0	1
p^{12}	1	320	0	320	32244	906	23748	7680	1644
Razem									
Podstawa	-	448	0	448	49059	1405	36134	11520	2471
$+\alpha*^*$	-	-	-	-	16186	472	11874	3840	823
$+\mu*^*$	-	64	-	64	16250	472	11938	3840	824

Tabela 9.7: Koszt układu szyfrującego Ascon-128

Operacja	L.iteracji	IO	ANC	W	#G	#NOT	#CNOT	#Toff	D
1. Inicjalizacja stanu i wstępne przetwarzanie									
R-XOR K	1	256	0	256	128	0	128	0	1
p^{12}	1	320	0	320	32244	906	23748	7680	1644
R-XOR K	1	256	0	256	128	0	128	0	1
2. Absorpcja danych dodatkowych									
C-XOR A_i	α	128	0	128	128	128	0	0	1
p^8	α	320	0	320	21496	544	15832	5120	1096
C-XOR $0^*\|\|1$	1	1	0	1	1	1	0	0	1
3. Absorpcja wiadomości i wyciskanie szyfrogramu									
C-XOR M_i	μ	128	0	128	128	128	0	0	1
R-XOR C_i	μ	256	0	256	128	0	128	0	1
p^8	$\mu - 1$	320	0	320	21496	544	15832	5120	1096
4. Generowanie tagu uwierzytelnienia									
R-XOR K	2	256	0	256	128	0	128	0	1
p^{12}	1	320	0	320	32244	906	23748	7680	1644
Razem									
Podstawa	-	448	0	448	43685	1269	32176	10240	2197
$+\alpha*^*$	-	-	-	-	21624	672	15832	5120	1097
$+\mu*^*$	-	128	-	128	21752	672	15960	5120	1098

Tabela 9.8: Koszt układu szyfrującego Ascon-128a

9.3.1 Koszt układu

Kolejnym krokiem jest wyznaczenie kosztu wyroczni uwzględniając liczbę wymaganych bloków wiadomości do przeprowadzenia ataku. Koszt ataku został obliczony dla dwóch przypadków:

1. Bez użycia tagu uwierzytelniania – Wariant podstawowy, w którym ignorowana jest obecność tagu uwierzytelniania do zmniejszenia liczby rozwiązań. Najmniej wydajny z pośród trzech wariantów;
2. Korzystają z tagu uwierzytelniania – Wariant poprawiony, w którym użyta jest własność szyfrowania z uwierzytelnianiem tym samym kluczem do zredukowania rozmiaru wymaganej znanej wiadomości jawnej;
3. Minimalny – Wariant korzystający z minimalnej możliwej liczby bloków wiadomości, jednocześnie ryzykując kolizję kluczy.

Operacja	μ	IO	ANC	W	#G	#NOT	#CNOT	#Toff	D
				Ascon-128					
Bez tagu uw.	5	768	0	768	130309	3765	95824	30720	6592
Z tagiem uw.	3	640	0	640	97809	2821	71948	23040	4944
Minimalny	1	512	0	512	65309	1877	48072	15360	3296
$+\alpha*$	–	–	–	–	16186	472	11874	3840	823
				Ascon-128a					
Bez tagu uw.	3	832	0	832	108941	3285	80056	25600	5492
Z tagiem uw.	2	704	0	704	87189	2613	64096	20480	4394
Minimalny	1	576	0	576	65437	1941	48136	15360	3296
$+\alpha*$	–	–	–	–	21624	672	15832	5120	1097

Tabela 9.9: Koszt wyroczni szyfrującej dla szyfru Ascon-128 i Ascon-128a

9.4 Koszt ataku

Ostatnim etapem badania jest wyznaczenie kosztu pełnego ataku w oparciu o koszt wyroczni szyfrującej oraz model ataku. W tym celu koszt wyroczni został pomnożony przez wymaganą liczbę rund algorytmu Grovera, która wynosi $\lfloor \frac{\pi}{4} * 2^{|K|/2} \rfloor$ oraz przez czynnik 2 ze względu na fakt, że każda runda wymaga wykonania wyroczni szyfrującej oraz odwrotnej wyroczni szyfrującej.

Operacja	μ	IO	ANC	W	#G	#NOT	#CNOT	#Toff	D
					Ascon-128				
Bez tagu uw.	5	768	0	768	$2^{81,64}$	$2^{76,52}$	$2^{81,20}$	$2^{79,56}$	$2^{77,34}$
Z tagiem uw.	3	640	0	640	$2^{81,23}$	$2^{76,11}$	$2^{80,79}$	$2^{79,14}$	$2^{76,92}$
Minimalny	1	512	0	512	$2^{80,65}$	$2^{75,53}$	$2^{80,20}$	$2^{78,56}$	$2^{76,34}$
$\alpha*$	–	–	–	–	$2^{78,63}$	$2^{73,53}$	$2^{78,19}$	$2^{76,56}$	$2^{74,34}$
					Ascon-128a				
Bez tagu uw.	3	832	0	832	$2^{81,38}$	$2^{76,33}$	$2^{80,94}$	$2^{79,30}$	$2^{77,07}$
Z tagiem uw.	2	704	0	704	$2^{81,06}$	$2^{76,00}$	$2^{80,62}$	$2^{78,97}$	$2^{76,75}$
Minimalny	1	576	0	576	$2^{80,65}$	$2^{75,57}$	$2^{80,21}$	$2^{78,56}$	$2^{76,34}$
$+\alpha*$	–	–	–	–	$2^{79,05}$	$2^{74,04}$	$2^{78,60}$	$2^{76,97}$	$2^{74,75}$

Tabela 9.10: Koszt przeprowadzenia ataku szyfru Ascon-128 i Ascon-128a

9.5 Dalsze kierunki rozwoju

Podobnie jak w przypadku badania szyfru SPARKLE, główną ścieżką dalszego rozwoju jest optymalizacja układu reprezentującego funkcję szyfrującą. Każda redukcja w rozmiarze implementacji zbliża ją do optymalnego rozmiaru, a co za tym idzie, określenia prawdziwego, minimalnego kosztu ataku, a nie jego górnego ograniczenia. W przypadku szyfru Ascon kontynuacja tych badań jest szczególnie istotna, z racji zostania nowym standardem kryptografii symetrycznej.

Podsumowanie

Przedłożona rozprawa doktorska zawiera wyniki pięciu badań dotyczących syntezy odwracalnych i kwantowych układów logicznych. W ramach przeprowadzonych badań uzyskano następujące wymierne oraz nowe wyniki:

1. Znaleziono optymalne implementacje skrzynek podstawieniowych wybranych szyfrów. Między innymi odnalezione zostały implementacje dla skrzynek używanych przez kandydatów rundy finałowej konkursu NIST na nowy standard kryptografii lekkiej, a w tym dla wybranego standardu - szyfru `Ascon`. Wyniki pracy mogą być użyte do badania bezpieczeństwa kwantowego warstw nieliniowych i właściwości konfuzji szyfrów. Wyniki badania zostały opublikowane w pracy [103];
2. Znaleziono kompletną przestrzeń szablonów identyczności o rozmiarach nie większych niż 6×7. Równoważne jest to ze znalezieniem wszystkich możliwych reguł podstawieniowych o tych rozmiarach, które mogą zostać użyte do optymalizacji układów odwracalnych. Dodatkowo, w pracy zaproponowano nową metodę przyspieszania obliczeń syntezy optymalnej z użyciem `SAT-solverów` poprzez wykorzystanie znalezionych szablonów identyczności. Wyniki pracy zostały opublikowane w pracy [104];
3. Zbadano korelację pomiędzy klasycznymi właściwościami bezpieczeństwa skrzynek podstawieniowych, a rozmiarem optymalnej implementacji jako układ odwracalny. Dowiedziono, że istnieje pozytywna korelacja pomiędzy tymi właściwościami oraz dowiedziono, że skrzynki spełniające wysokie wymagania klasycznych właściwości bezpieczeństwa powinny być bardziej kosztowne do zaimplementowania jako układ odwracalny, a zatem być bezpieczniejsze w kontekście ataków kwantowych. Wyniki pracy mogą zostać wykorzystane w dalszych badaniach nad bezpieczeństwem kwantowym skrzynek podstawieniowych oraz do opracowania nowych metod tworzenia skrzynek uwzględniających zagrożenia kwantowe. Wyniki badań zostały zgłoszone do opublikowania.

4. Opracowano dwa ataki z użyciem algorytmu Grovera, na szyfry AEAD o strukturze gąbki – SPARKLE oraz Ascon. Oba badania przedstawiają dokładny koszt ataku wyrażony w poszczególnych bramkach kwantowych oraz rozmiarze rejestru kwantowego. Badania uwzględniają możliwość zrównoleglania układów odwracalnych i również prezentują głębokości układów potrzebnych do przeprowadzenia ataku. Badanie dotyczące szyfru Ascon zostało zgłoszone do opublikowania. Badanie dotyczące szyfru SPARKLE zostało opublikowane w pracy [105], która została zacytowana w raporcie [20] dotyczącym rozstrzygnięcia rundy finałowej konkursu NIST na nowy standard kryptografii lekkiej.

Wyniki badań są całkowicie odtwarzalne, deterministyczne i niezależne od wybranych prób badawczych lub źródeł losowych. Wyniki badań posiadają opisaną metodykę przeprowadzenia badania, dzięki czemu mogą zostać całkowicie i deterministycznie odtworzone oraz porównane z wynikami opisanymi w innych pracach dotyczących podobnych zagadnień.

Dalsze kierunki rozwoju

Wraz ze znalezieniem odpowiedzi na wybrane pytania naturalnym jest, że odpowiedzi te otwierają kolejne ścieżki potencjalnych badań. Między innymi zostały zidentyfikowane nowe zagadnienia:

1. Zbadanie wpływu świadków suboptymalności na proces syntezy optymalnej z użyciem SAT-solverów. W ramach badania została postawiona teza jakoby użycie tego zjawiska mogłoby polepszyć czas uzyskania wyników natomiast w praktyce wymaga to zbadania;
2. Znalezienie optymlanych implementacji dla skrzynek większych niż 5×5. Pierwsze badanie zostało zakończone na tym rozmiarze ze względu na ograniczenia czasowe. Możliwe jest, że znalezienie efektywniejszych metod reprezentowania problemu w postaci formuł CNF umożliwi przyspieszenie rozwiązywania tego rodzaju zadań i znalezienie optymalnych implementacji dla większych skrzynek;
3. Optymalizacja układów reprezentujących szyfry. Badania dotyczące szyfrów SPARKLE oraz Ascon wykorzystują elementy, dla których optymalność nie została udowodniona. W przypadku szyfru Ascon użyta jest implementacja war-

stwy liniowej znaleziona heurystycznie, natomiast dla szyfru SPARKLE operacja dodawania arytmetycznego korzysta z implementacji, dla której optymalność nie została dowiedziona. Postęp w badaniach nad tymi elementami może skutkować zmniejszeniem kosztu ataków.

4. Zbadanie przyczyn korelacji pomiędzy właściwościami klasycznymi, a optymalnym rozmiarem. Badanie korelacji miało charakter obserwacji zjawiska, naturalnym dalszym krokiem jest wyjaśnienie jego przyczyn w sposób analityczny, a docelowo ścisłe dowiedzenie takiej korelacji jeżeli jest to możliwe.

Praktyczne wykorzystanie wyników pracy

Podsumowując wyniki osiągnięte w pracy można zidentyfikować potencjalne i rzeczywiste ścieżki użycia wyników:

1. Opracowanie metodyki oceny szyfrów symetrycznych pod względem ataków kwantowych – badanie dt. szyfru SPARKLE zostało wzięte pod uwagę w raporcie dt. rozstrzygnięcia konkursu NIST na kryptografię lekką. Świadczy to o tym, że badania takie mogą stanowić podstawę oceny bezpieczeństwa szyfrów. Dalsze prace nad metodykami i narzędziami do realizacji takich ocen mogą przyczynić się do poprawy jakości kryptografii symetrycznej. Biorąc pod uwagę szerokie zainteresowanie prowadzeniem takich badań nad szyframi wykorzystywanymi przez wiele krajów NATO oraz spoza tej organizacji można wnioskować, że wdrożenie takiej metodyki i narzędzi jest możliwe do krajowych procesów analizy bezpieczeństwa w celu badania rozwiązań kryptografii narodowej, wykorzystywanej na szczeblu Obrony Narodowej oraz Bezpieczeństwa Wewnętrznego.
2. Wyniki badań dotyczące poszukiwania optymalnych implementacji oraz szablonów identyczności mogą zostać wykorzystane do poszukiwania bezpiecznych funkcji nieliniowych, czyli takich, których optymalna implementacja jest możliwie wysoka. Opierając się o te wyniki możliwe jest polepszenie jakości projektowanych rozwiązań kryptografii symetrycznej. Podobnie jak w poprzednim punkcie, wyniki mogą zostać wykorzystane do projektowania rozwiązań kryptografii narodowej.

Bibliografia

1. Toffoli, T. „*Reversible computing*". *Automata, Languages and Programming* **85,** 632–644 (1980).
2. Feynman, R. P. „*Simulating physics with computers*". *International Journal of Theoretical Physics* **21,** 467–488 (1982).
3. Feynman, R. „*Quantum mechanical computers*". *Foundations of Physics* **16,** 507–532 (1986).
4. Feynman, R. P. *Feynman Lectures on Computation* 1 wyd. (red. Hey, T. & Allen, R. W.) (CRC Press, 2018).
5. NIST. *Lightweight Cryptography Workshop 2015* NIST. (2025).
6. McKay, K. A., Bassham, L., Turan, M. S. & Mouha, N. „*Report on lightweight cryptography*". Raport wenętrzny NIST IR 8114 (National Institute of Standards and Technology USA, Gaithersburg, MD, 2017).
7. National Institute of Standards and Technology USA. „*Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process*". Raport wenętrzny (National Institute of Standards and Technology USA, Gaithersburg, MD, 2018).
8. Turan, M. S., McKay, K. A., Çalık, Ç., Chang, D. & Bassham, L. „*Status report on the first round of the NIST lightweight cryptography standardization process*". Raport wenętrzny NIST IR 8268 (National Institute of Standards and Technology USA, Gaithersburg, MD, 2019).
9. Sonmez Turan, M. *i in.* „*Status report on the second round of the NIST lightweight cryptography standardization process*". Raport wenętrzny NIST IR 8369 (National Institute of Standards and Technology USA, Gaithersburg, MD, 2021).

10. Dobraunig, C., Eichlseder, M., Mendel, F. & Schläffer, M. „*Ascon v1.2*". Specyfikacja Algorytmu, Zgłoszenie (National Institute of Standards and Technology USA, Gaithersburg, MD, 2021).

11. Beyne, T., Yu Long, C., Dobraunig, C. & Mennink, B. „*Elephant v2*". Specyfikacja Algorytmu, Zgłoszenie (National Institute of Standards and Technology USA, Gaithersburg, MD, 2021).

12. Banik, S. *i in.* „*GIFT-COFB v1.1*". Specyfikacja Algorytmu, Zgłoszenie (National Institute of Standards and Technology USA, Gaithersburg, MD, 2021).

13. Hell, M. *i in.* „*Grain-128 AEAD v2 - A lightweight AEAD stream cipher*". Specyfikacja Algorytmu, Zgłoszenie (National Institute of Standards and Technology USA, Gaithersburg, MD, 2021).

14. Dobraunig, C. *i in.* „*ISAP v2.0*". Specyfikacja Algorytmu, Zgłoszenie (National Institute of Standards and Technology USA, Gaithersburg, MD, 2021).

15. Bao, Z. *i in.* „*PHOTON - Beetle Authenticated Encryption and Hash Family*". Specyfikacja Algorytmu, Zgłoszenie (National Institute of Standards and Technology USA, Gaithersburg, MD, 2021).

16. Guo, C., Iwata, T., Khairallah, M., Minematsu, K. & Peyrin, T. „*Romulus v1.3*". Specyfikacja Algorytmu, Zgłoszenie (National Institute of Standards and Technology USA, Gaithersburg, MD, 2021).

17. Beierle, C. *i in.* „*Schwaemm and Esch: Lightweight Authenticated Encryption and Hashing using the Sparkle Permutation Family*". Specyfikacja Algorytmu, Zgłoszenie (National Institute of Standards and Technology USA, Gaithersburg, MD, 2021).

18. Wu, H. & Huang, T. „*TinyJAMBU: A Family of Lightweight Authenticated Encryption Algorithms (Version 2)*". Specyfikacja Algorytmu, Zgłoszenie (National Institute of Standards and Technology USA, Gaithersburg, MD, 2021).

19. Daemen, J., Hoffert, S., Peeters, M., Van Assche, G. & Van Keer, R. „*Xoodyak, a lightweight cryptographic scheme*". Specyfikacja Algorytmu, Zgłoszenie (National Institute of Standards and Technology USA, Gaithersburg, MD, 2021).

20. Sonmez Turan, M. *i in.* „*Status report on the final round of the NIST lightweight cryptography standardization process*". Raport wenętrzny NIST IR 8454 (National Institute of Standards and Technology USA, Gaithersburg, MD, 2023).

21. Sonmez Turan, M. „*ASCON Family for Constrained Environments: Authenticated Encryption, Hash, and Extendable Output Functions*". Szkic standardu NIST SP 800-232 (National Institute of Standards and Technology, Gaithersburg, MD, 2024).

22. Grassl, M., Langenberg, B., Roetteler, M. & Steinwandt, R. „*Applying Grover's Algorithm to AES: Quantum Resource Estimates*". *Post-Quantum Cryptography - PQCrypto 2016,* 29–43 (2016).

23. Jaques, S., Naehrig, M., Roetteler, M. & Virdia, F. „*Implementing Grover Oracles for Quantum Key Search on AES and LowMC*". *Advances in Cryptology – EUROCRYPT 2020* (red. Canteaut, A. & Ishai, Y.) 280–310 (2020).

24. Langenberg, B., Pham, H. & Steinwandt, R. „*Reducing the Cost of Implementing the Advanced Encryption Standard as a Quantum Circuit*". *IEEE Transactions on Quantum Engineering* **1,** 1–12 (2020).

25. Beaulieu, R. *i in.* „*The SIMON and SPECK lightweight block ciphers*". *Proceedings of the 52nd Annual Design Automation Conference,* 1–6 (2015).

26. Leander, G. & May, A. „*Grover Meets Simon – Quantumly Attacking the FX-construction*". *Advances in Cryptology – ASIACRYPT 2017,* 161–178 (2017).

27. Anand, R., Maitra, A. & Mukhopadhyay, S. „*Grover on SIMON*". *Quantum Information Processing* **19,** 340 (2020).

28. Jang, K., Choi, S., Kwon, H. & Seo, H. „*Grover on SPECK: Quantum Resource Estimates*". *ePrint IACR* (2020).

29. „*Evaluation of Quantum Cryptanalysis on SPECK*". *Progress in Cryptology – INDOCRYPT 2020. Lecture Notes in Computer Science* (red. Bhargavan, K., Oswald, E. & Prabhakaran, M.) (2020).

30. De Cannière, C., Dunkelman, O. & Knežević, M. „*KATAN and KTANTAN — A Family of Small and Efficient Hardware-Oriented Block Ciphers*". *Cryptographic Hardware and Embedded Systems - CHES 2009* **5747** (red. Clavier, C. & Gaj, K.) Series Title: Lecture Notes in Computer Science, 272–288 (2009).

31. Rahman, M. & Paul, G. „*Grover on KATAN: Quantum Resource Estimation*". *IEEE Transactions on Quantum Engineering* **3,** 1–9 (2022).

32. Nir, Y. & Langley, A. „*ChaCha20 and Poly1305 for IETF Protocols*". Request for Comments RFC 7539. Num Pages: 45 (Internet Engineering Task Force, 2015).

33. Bathe, B., Anand, R. & Dutta, S. „*Evaluation of Grover's algorithm toward quantum cryptanalysis on ChaCha*". *Quantum Information Processing* **20,** 394 (2021).

34. Jang, K. *i in.* „*Grover on Korean Block Ciphers*". *Applied Sciences* **10,** 6407 (2020).

35. Jang, K. *i in.* „*Grover on PIPO*". *Electronics* **10,** 1194 (2021).

36. Li, Y., Lin, H., Liang, M. & Sun, Y. „*A new quantum cryptanalysis method on block cipher Camellia*". *IET Information Security* **15,** 487–495 (2021).

37. Denisenko, D. V. „*Quantum Circuits for S-Box Implementation without Ancilla Qubits*". *Journal of Experimental and Theoretical Physics* **128,** 847–855 (2019).

38. Denisenko, D. V., Marshalko, G. B., Nikitenkova, M. V., Rudskoi, V. I. & Shishkin, V. A. „*Estimating the Complexity of Grover's Algorithm for Key Search of Block Ciphers Defined by GOST R 34.12-2015*". *Journal of Experimental and Theoretical Physics* **128,** 552–559 (2019).

39. Song, G. *i in.* „*Grover on SM3*". *Information Security and Cryptology – ICISC 2021* **13218** (red. Park, J. H. & Seo, S.-H.) Series Title: Lecture Notes in Computer Science, 421–433 (2022).

40. Anand, R., Maitra, A., Maitra, S., Mukherjee, C. S. & Mukhopadhyay, S. „*Quantum Resource Estimation for FSR Based Symmetric Ciphers and Related Grover's Attacks*". *Progress in Cryptology – INDOCRYPT 2021* **13143.** Series Title: Lecture Notes in Computer Science, 179–198 (2021).

41. Burek, E., Wronski, M. J., Mank, K. & Misztal, M. „*Algebraic attacks on block ciphers using quantum annealing*". *IEEE Transactions on Emerging Topics in Computing,* 1–1 (2022).

42. Shor, P. W. „*Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*". *SIAM Review* **41,** 303–332 (1999).

43. Bach, E. „*Discrete Logarithms and Factoring*" (University of California at Berkeley, USA, 1984).

44. Rivest, R. L., Shamir, A. & Adleman, L. „*A method for obtaining digital signatures and public-key cryptosystems*". *Commun. ACM* **21,** 120–126 (1978).

45. National Institute of Standards and Technology USA. „*Digital Signature Standard*". Specyfikacja Algorytmu NIST FIPS 186 (National Institute of Standards and Technology USA, Washington, D.C., 1984).

46. National Institute of Standards and Technology USA. „*Digital Signature Standard*". Specyfikacja Algorytmu NIST FIPS 186-5 (National Institute of Standards and Technology USA, Washington, D.C., 2013).

47. Diffie, W. & Hellman, M. „*New directions in cryptography*". *IEEE Transactions on Information Theory* **22,** 644–654 (1976).

48. Barker, E., Chen, L., Roginsky, A. & Smid, M. „*Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*". Specyfikacja Algorytmu NIST SP 800-56 (National Institute of Standards and Technology USA, 2013).

49. National Institute of Standards and Technology USA. „*Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*". Raport wenętrzny (National Institute of Standards and Technology USA, Gaithersburg, MD, 2017).

50. Alagic, G. *i in.* „*Status report on the fourth round of the NIST post-quantum cryptography standardization process*". Raport wenętrzny NIST IR 8545 (National Institute of Standards and Technology USA, Gaithersburg, MD, 2025).

51. National Institute of Standards and Technology USA. „*Module-lattice-based Digital Signature Standard*". Specyfikacja Algorytmu NIST FIPS 204 (National Institute of Standards and Technology USA, Washington, D.C., 2024).

52. Fouque, P.-A. *i in.* „*Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU*". Specyfikacja Algorytmu (National Institute of Standards and Technology USA, Washington, D.C., 2020).

53. National Institute of Standards and Technology USA. „*Stateless Hash-based Digital Signature Standard*". Specyfikacja Algorytmu NIST FIPS 205 (National Institute of Standards and Technology USA, Washington, D.C., 2024).

54. National Institute of Standards and Technology USA. „*Module-lattice-based Key-encapsulation Mechanism Standard*". Specyfikacja Algorytmu NIST FIPS 203 (National Institute of Standards and Technology USA, Washington, D.C., 2024).

55. Gaborit, P. *i in.* „*Hamming Quasi-Cyclic (HQC) Fourth round version*". Specyfikacja Algorytmu (National Institute of Standards and Technology USA, Washington, D.C., 2025).

56. Bloch, F. „*Nuclear Induction*". *Physical Review* **70,** 460–474 (1946).

57. Cohen-Tannoudji, C. *Quantum Mechanics* 3 t. (1977).

58. Shankar, R. *Principles of Quantum Mechanics* (Springer US, New York, NY, 1994).

59. Barenco, A. *i in.* „*Elementary gates for quantum computation*". *Physical Review A* **52,** 3457–3467 (1995).

60. Adamus, E. „*Przestrzenie unitarne*". Wykład 10 (Wydział Matematyki Stosowanej Akademia Górniczo-Hutnicza w Krakowie, 2023).

61. Nielsen, M. A. & Chuang, I. L. *Quantum computation and quantum information* 10th anniversary edition (Cambridge university press, Cambridge, 2010).

62. Feistel, H. *pat. USA* 3798359A (1974).

63. Draper, T., Kutin, S., Rains, E. & Svore, K. „*A logarithmic-depth quantum carry-lookahead adder*". *Quantum Information and Computation* **6,** 351–369 (2006).

64. Miller, D. M., Maslov, D. & Dueck, G. W. „*A transformation based algorithm for reversible logic synthesis*". w *Proceedings of the 40th annual Design Automation Conference* DAC03: 2003 40th Annual Design Automation Conference (ACM, Anaheim CA USA, 2003), 318–323.

65. Kazuo, Q. C., Iwama, K. & Kambayashi, Y. *Transformation Rules for Designing CNOT-based* 2002.

66. Monfared, A. T. „*Advanced Methods in Quantum and Reversible Circuit Synthesis for Boolean and Multivalued Logic*". Rozprawa Doktorska (Università degli Studi di Milano, 2024).

67. Younis, E. & Iancu, C. „*Quantum Circuit Optimization and Transpilation via Parameterized Circuit Instantiation*". w *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)* 2022 IEEE International Conference on Quantum Computing and Engineering (QCE) (IEEE, Broomfield, CO, USA, 2022), 465–475.

68. Davis, M. G. *i in.* „*Towards Optimal Topology Aware Quantum Circuit Synthesis*". w *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)* 2020 IEEE International Conference on Quantum Computing and Engineering (QCE) (IEEE, Denver, CO, USA, 2020), 223–234.

69. Matteo, O. D. & Mosca, M. „*Parallelizing quantum circuit synthesis*". *Quantum Science and Technology* **1,** 015003 (2016).

70. Gupta, P., Agrawal, A. & Jha, N. „*An Algorithm for Synthesis of Reversible Logic Circuits*". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **25,** 2317–2330 (2006).

71. Ge, Y. *i in.* „*Quantum Circuit Synthesis and Compilation Optimization: Overview and Prospects*" (2024).

72. Saeedi, M. & Markov, I. L. „*Synthesis and optimization of reversible circuits—a survey*". *ACM Computing Surveys* **45,** 1–34 (2013).

73. Dueck, G. W., Miller, D. M. & Maslov, D. *Templates for Toffoli Network Synthesis* 2003.

74. Maslov, D., Dueck, G. & Miller, D. „*Simplification of Toffoli networks via templates*". w (8 paź. 2003), 53–58.

75. Maslov, D., Young, C., Miller, D. & Dueck, G. „*Quantum Circuit Simplification Using Templates*". w *Design, Automation and Test in Europe* Design, Automation and Test in Europe (IEEE, Munich, Germany, 2005), 1208–1213.

76. Maslov, D., Dueck, G. & Miller, D. „*Toffoli network synthesis with templates*". *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* **24,** 807–817 (1 lip. 2005).

77. Wille, R. & Grosse, D. „*Fast exact toffoli network synthesis of reversible logic*". w *2007 IEEE/ACM International Conference on Computer-Aided Design* 2007 IEEE/ACM International Conference on Computer Aided Design (IEEE, San Jose, CA, list. 2007), 60–64.

78. Patel, K., Markov, I. & Hayes, J. „*Optimal Synthesis of Linear Reversible Circuits*". *Quantum Information & Computation* **8,** 282–294 (2008).

79. Meuli, G., Soeken, M. & De Micheli, G. „*SAT-based CNOT, T Quantum Circuit Synthesis*". *Reversible Computation* **11106** (red. Kari, J. & Ulidowski, I.) 175–188 (2018).

80. Maslov, D. & Miller, D. M. „*Comparison of the Cost Metrics for Reversible and Quantum Logic Synthesis*". *arXiv:quant-ph/0511008* (4 lut. 2006).

81. Golubitsky, O. & Maslov, D. „*A Study of Optimal 4-bit Reversible Toffoli Circuits and Their Synthesis*". *IEEE Transactions on Computers* **61,** 1341–1353 (wrz. 2012).

82. Ostaszewski, M., Trenkwalder, L. M., Masarczyk, W., Scerri, E. & Dunjko, V. „*Reinforcement learning for optimization of variational quantum circuit architectures*" (2021).

83. Fösel, T., Niu, M. Y., Marquardt, F. & Li, L. „*Quantum circuit optimization with deep reinforcement learning*" (2021).

84. Fürrutter, F., Muñoz-Gil, G. & Briegel, H. J. „*Quantum circuit synthesis with diffusion models*". *Nature Machine Intelligence* **6,** 515–524 (2024).

85. Grover, L. K. „*A fast quantum mechanical algorithm for database search*". *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96,* 212–219 (1996).

86. Hirvensalo, M. *Quantum Computing* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2004).

87. Zalka, C. „*Grover's quantum searching algorithm is optimal*". *Physical Review A* **60,** 2746–2751 (1999).

88. Dawson, E., Gustafson, H. & Pettitt, A. „*Strict Key Avalanche Criterion*". *Computers and Security,* 687–697 (1994).

89. Biere, A., Heule, M. J. H., Maaren, H. v. & Walsh, T. *Handbook of satisfiability* Second edition (IOS Press, Amsterdam Berlin Washington, DC, 2021).

90. Chang, S. J. *i in.* „*Third-Round Report of the SHA-3 Cryptographic Hash Algorithm Competition*". Raport wenętrzny NIST IR 7896. Edition: 0 (National Institute of Standards and Technology, Gaithersburg, MD, 2012).

91. Preneel, B. w *Encyclopedia of Cryptography and Security* (red. Van Tilborg, H. C. A. & Jajodia, S.) 831–836 (Springer US, Boston, MA, 2011).

92. Phan, R. C.-W. „*Mini Advanced Encryption Standard (MINI-AES): a Testbed for Cryptanalysis Sudents*". *Cryptologia* **26,** 283–306 (2002).

93. Rijmen, V. & Barreto, P. S. L. M. „*The Whirlpool Hashing Function*". Specyfikacja Algorytmu, Zgłoszenie (2003).

94. Hongjun, W. „*The Hash Function JH*". Specyfikacja Algorytmu, Zgłoszenie (2011).

95. Bertoni, G., Daemen, J., Peeters, M. & Van Assche, G. „*Keccak specifications*". Specyfikacja Algorytmu, Zgłoszenie (2008).

96. *SAT Competition 2022* https://satcompetition.github.io/2022/.

97. Javadi-Abhari, A. *i in.* „*Quantum computing with Qiskit*". Version Number: 3 (2024).

98. Biham, E. & Shamir, A. „*Differential cryptanalysis of DES-like cryptosystems*". *Journal of Cryptology* **4,** 3–72 (1991).

99. Matsui, M. „*Linear Cryptanalysis Method for DES Cipher*". w *Advances in Cryptology — EUROCRYPT '93* (red. Helleseth, T.) (Springer, Berlin, Heidelberg, 1994), 386–397.

100. Zhegalkin, I. I. „*O rachunku zdań w logice symbolicznej*". *Matematicheskii Sbornik* **34** (1927).

101. Rothaus, O. „*On bent functions*". *Journal of Combinatorial Theory, Series A* **20,** 300–305 (1976).

102. Roy, S., Baksi, A. & Chattopadhyay, A. „*Quantum Implementation of ASCON Linear Layer*" (2023).

103. Jagielski, A. „*Optimal SAT Solver Synthesis of Quantum Circuits Representing Cryptographic Nonlinear Functions*". *International Journal of Electronics and Telecommunications* **69,** 261–267 (26 lip. 2023).

104. Jagielski, A. „*Complete synthesis of identity templates for quantum and reversible logic MCT circuits using SAT-solvers and proposal of suboptimality witness notion*". *International Journal of Electronics and Telecommunications; 2024; vol. 70; No 3; 727-732* (2024).

105. Jagielski, A. & Kanciak, K. „*Grover on sparkle quantum resource estimation for a NIST LWC call finalist*". *Quantum Information and Computation* **22,** 1132–1143 (wrz. 2022).

Spis rysunków

Spis tabel