Uniwersytet Kardynała Stefana Wyszyńskiego w Warszawie
Wydział Matematyczno-Przyrodniczy
Szkoła Nauk Ścisłych

Sebastian Kocielnik
Nr albumu: 104089

SZYFROWANIE Z KLUCZEM PUBLICZNYM W ŚWIECIE OBLICZEŃ KWANTOWYCH

Praca licencjacka na kierunku *informatyka*

Praca wykonana pod kierunkiem
dr. Konrada Zdanowskiego

Wrzesień 2019

Słowa kluczowe

kryptografia, szyfrowanie, bezpieczeństwo, LWE, obliczenia kwantowe

Dziedzina Socrates-Erasmus

11.3 Informatyka

Klasyfikacja tematyczna

Security and privacy∼Public key encryption

English title

PUBLIC-KEY CRYPTOGRAPHY IN THE WORLD OF QUANTUM COMPUTING

Załącznik do Zarządzenia Nr 78/2014 Rektora UKSW
z dnia 14 listopada 2014 r.
Załącznik nr 3 do Zarządzenia Nr 39/2007 Rektora UKSW
z dnia 9 listopada 2007r.

……
Imię i nazwisko studenta/studentki
……
Nr albumu
……..
Wydział
……
Instytut
……..
Kierunek

Dziekan Wydziału ……..

……..

OŚWIADCZENIE

Świadomy(a) odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w żadnej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Oświadczam, że poinformowano mnie o zasadach dotyczących kontroli samodzielności prac dyplomowych i zaliczeniowych. W związku z powyższym oświadczam, że wyrażam zgodę na przetwarzanie* moich prac pisemnych (w tym prac zaliczeniowych i pracy dyplomowej) powstałych w toku studiów i związanych z realizacją programu kształcenia w Uczelni, a także na przechowywanie pracy dyplomowej w celach realizowanej procedury antyplagiatowej w ogólnopolskim repozytorium pisemnych prac dyplomowych.

…….....
podpis studenta

*Przez przetwarzanie pracy rozumie się porównywanie przez system antyplagiatowy jej treści z innymi dokumentami (w celu ustalenia istnienia nieuprawnionych zapożyczeń) oraz generowanie raportu podobieństwa.

OŚWIADCZENIE

Oświadczam, że niniejsza praca napisana przez

Pana/Panią....., nr albumu ..... została przygotowana pod moim kierunkiem i stwierdzam, że spełnia ona warunki do przedstawienia jej w postepowaniu o nadanie tytułu zawodowego.

…….....
podpis promotora

Spis treści

Streszczenie

W związku z ciągłym rozwojem technicznym tworzone są coraz bardziej złożone komputery kwantowe, które – według niektórych opinii – wieszczą całkowite złamanie obecnej kryptografii. Jakkolwiek nic nie zapowiada rychłego upadku bezpieczeństwa informacji, kryptolodzy pracują nad tym, aby mimo wszystko być o krok do przodu przed dostępnymi metodami łamania zabezpieczeń. W pracy przedstawimy, zaproponowaną przez Regeva, metodę szyfrowania opartą o trudność problemów CVP i LWE. Uważa się, że jest to metoda odporna na ataki przy użyciu komputerów kwantowych.

Abstract

While the strength of quantum computers increases, the methods of standard public key cryptography may be at risk. Even though this perspective seems to be distant, cryptologists work on new, quantum proof methods of encrypting. We present one of those methods, proposed by Regev and based on hardness of CVP and LWE problems.

Rozdział 1

Wstęp

W dziedzinie kryptografii bardzo popularnym algorytmem szyfrowania z kluczem publiczym jest RSA, który został zaprojektowany przez Rona Rivesta, Adiego Shamira, oraz Leonarda Adlemana (akronim, jakim jest jego nazwa, powstał z pierwszych liter nazwisk autorów) i został opublikowany w 1977 roku. Autorzy wykorzystali koncepcję asymetrycznej metody szyfrowania z kluczem publicznym, która została przedstawiona w roku 1976 przez Whitfelda Diffiego oraz Martina Hellmana.

RSA jest algorytmem, który – przy właściwym doborze parametrów – trudno jest złamać, w związku z czym jest wykorzystywany przede wszystkim w początkowych etapach ustanawiania zaszyfrowanej komunikacji (algorytm jest stosunkowo wolny, przez co nie jest wykorzystywany do szyfrowania całej transmisji), gdzie następuje wymiana kluczy innych algorytmów.

Przyszłość RSA nie jest jednak w pełni pewna i bezpieczna. Powody są następujące:

- Wraz z rozwojem technicznym coraz szerzej dostępna jest coraz większa moc obliczeniowa. Jakkolwiek faktoryzacja jest zadaniem trudnym na komputerach klasycznych, posiadając odpowiednią ilość czasu można wykonać stosowne obliczenia. Nie jest to jednak zagrożenie, którego urzeczywistnienia należy się spodziewać w krótkim czasie – w roku 2010 największa sfaktoryzowana liczba miała 768 bitów (232 znaki dziesiętne), przy czym proces ten zajął około dwa lata czasu rzeczywistego, mimo wykorzystania wielu komputerów pracujących równolegle (ilość czasu potrzebnego na faktoryzację tej liczby przyrównano do prawie 2000 lat pracy komputera wykorzystującego jednordzeniowy procesor AMD Opteron 2.2GHz)[1]. Należy przy tym zwrócić uwagę na fakt, że był to czas potrzebny na złamanie **jednego** klucza.

  Ponadto, nawet w miarę postępu w kwestii algorytmów czy też dostępnej mocy obliczeniowej, trudno będzie mówić całkowitym złamaniu tego algorytmu – zasadniczo będzie można mówić o możliwości łamania kluczy o określonym rozmiarze w racjonalnym czasie. Łamanie większych kluczy (jak na przykład 4096-bitowych) zapewne jeszcze długo będzie poza zasięgiem komputerów klasycznych, a właśnie „złamaniem szyfrowania" można nazwać stan, gdzie możliwość łamania szyfru nie jest zależna od rozmiaru jego klucza. Dopiero znalezienie efektywnej metody rozkładu liczb na czynniki pierwsze pozwoliłoby na łamanie szyfrowania RSA. Bez takiej metody na zwiększenie mocy obliczeniowej atakującego można odpowiedzieć zwiększeniem długości klucza.

- W roku 1994 Peter Shor pokazał, że komputer kwantowy będzie mógł dokonywać faktoryzacji dużych liczb w zdecydowanie krótszym czasie niż komputery klasyczne. Przedstawił on wielomianowy algorytm kwantowy dokonujący tej operacji (tzw. algorytm Shora). W roku 2014 największą liczbą sfaktoryzowaną algorytmami kwantowymi było 56153 co nie jest liczbą dużą i

[1] *Vide* https://en.wikipedia.org/wiki/RSA_numbers

można ją zapisać na 11 bitach. Nie znamy jednak odpowiedzi na pytanie, kiedy powstaną komputery kwantowe zdolne łamać szeroko wykorzystywane rozmiary kluczy – problem budowania dużych komputerów kwantowych okazuje się być bardzo trudny do rozwiązania. W szczególności trudno utrzymać stabilny stan tzw. q-bitów, czyli bitów kwantowych, a także wprowadzić odpowiednią korekcję błędów w obliczeniach. Nie wiemy jednak, czy postęp technologiczny nie pozwoli nam budować takich komputerów w niedalekiej przyszłości.

Powyższe wątpliwości dotyczą większości popularnych algorymów szyfrowania, których bezpieczeństwo opiera się na problemie faktoryzacji liczb całkowitych, czy też na problemie logarytmu dyskretnego – oba te problemy mogą być rozwiązywane w sposób wydajny na odpowiednio mocnym komputerze kwantowym za pomocą algorytmów Shora. Niezależnie od tego co (i kiedy) przyniesie przyszłość kryptolodzy chcą być pewni, że nic ich nie zaskoczy. Opracowują wobec tego rozwiązania, które zapewnią stałe bezpieczeństwo wymiany informacji. Jednym z nich jest Oded Regev, który jako pierwszy w 2005 roku przedstawił metodę szyfrowania opartą na kratach oraz problem LWE (za opisanie którego w artykule [6] w 2018 roku otrzymał nagrodę Gödla). Jak argumentuje, algorytm ten (oczywiście przy odpowiednim doborze parametrów) jest trudny do rozwiązania, także dla komputerów kwantowych. Z punktu widzenia kryptografii algorytm ten jest bardzo obiecujący – od czasu jego przedstawienia wielu kryptologów poswięciło się ulepszaniu dowodu bezpieczeństwa tej metody i zwiększaniu jej wydajności, a także tworzeniu nowych algorytmów opartych na problemach związanych między innymi z kratami, ale nie tylko.

W niniejszej pracy przedstawiam oryginalny algorytm przedstawiony przez Regeva. W rozdziale drugim pokrótce opisane są podstawowe pojęcia wykorzystywane w tej metodzie szyfrowania. W rozdziale trzecim znajduje się jej opis formalny, w czwartym – krótki opis implementacji algorytmu, w piątym podsumowanie pracy wraz z potencjalnymi kierunkami rozwoju metod szyfrowania opartych na kratach.

Rozdział 2

Podstawowe pojęcia

Przedstawię teraz podstawowe pojęcia matematyczne oraz informatyczne wykorzystywane w metodzie szyfrowania opartej na LWE.

2.1. Szyfrowanie z kluczem publicznym

Opisywana metoda szyfrowania jest jednym z algorytmów kryptografii klucza publicznego, która jest również nazywana kryptografią asymetryczną. Ten rodzaj kryptografii wykorzystuje dwa klucze – publiczny i prywatny. Oba te klucze są tworzone przez osobę, która chce umożliwić innym bezpieczne wysyłanie wiadomości do siebie.

Jednym z założeń kryptografii klucza publicznego jest to, iż uzyskanie klucza prywatnego z klucza publicznego musi być obliczeniowo trudne, dzięki czemu dostępność klucza publicznego nie będzie oznaczała podatności na ataki mające na celu złamanie szyfru. W związku z tym, że klucz prywatny jest z założenia posiadany tylko przez właściciela klucza, jest on jedyną osobą mogącą odczytać wiadomości zaszyfrowane odpowiadającym mu kluczem publicznym. To zaś powoduje, że algorytm do wysłania wiadomości w drugą stronę wymaga drugiej pary kluczy.

2.2. Kryptografia oparta na kratach

Kryptografia oparta na kratach (ang. *lattice-based cryptography*) jest ogólnym terminem wykorzystywanym do określania systemów kryptograficznych wykorzystujących kraty w samej konstrukcji systemu lub w dowodzie bezpieczeństwa algorytmu. Krata zaś jest zdefiniowana następująco: krata $L \subset \mathbb{R}^n$ jest zbiorem wszystkich liniowych kombinacji bazy wektorów $\mathbf{b_1}, ..., \mathbf{b}_n \in \mathbb{R}^n$ postaci

$$a_1\mathbf{b}_1 + \cdots + a_n\mathbf{b}_n,$$

gdzie, dla $i \leq n$, $a_i \in \mathbb{Z}$. Ilość wektorów w bazie, które są niezależne liniowo, określa wymiar kraty. Podkreślmy, że współczynniki kombinacji liniowej wektorów są liczbami całkowitymi. Nie mówimy więc tutaj o podprzestrzeni wektorowej w klasycznym sensie, gdzie skalary należą do ustalonego ciała.

2.2.1. SVP oraz CVP

Jednym z argumentów bezpieczeństwa algorytmu zaproponowanego przez Regeva jest przyrównanie przewidywanej trudności przeprowadzenia ataku do trudności problemu CVP (problemu najbliższego wektora - *closest vector problem*) lub SVP (problemu najkrótszego wektora - *shortest vector problem*).

Są to także ważne problemy obliczeniowe związane z kratami. Ich trudność w przypadku średnim tworzy podstawę wielu dowodów bezpieczeństwa schematów kryptograficznych. Problemy o wysokiej średniej złożoności są najlepszymi kandydatami na podstawę systemów odpornych na łamanie komputerami kwantowymi.

Niech L będzie kratą rozpinaną przez wektory $b_1, ..., b_n$, a $\lambda(L)$ oznacza długość najkrótszego niezerowego wektora w kracie L, to znaczy

$$\lambda(L) = \min_{v \in L \setminus \{0\}} ||v||_N$$

gdzie N jest normą (funkcją przypisującą dodatnią długość lub rozmiar każdemu wektorowi w przestrzeni wektorowej).

SVP Problem ten zdefiniowany jest następująco: dla ustalonej bazy przestrzeni wektorowej V oraz normy N (zazwyczaj jest to norma Euklidesowa) na wejściu opisującym wektory bazy kraty L należy znaleźć najkrótszy niezerowy wektor w L według normy N. Innymi słowy, wynikiem algorytmu rozwiązującego SVP powinien być niezerowy wektor $v \in L$ taki, że $||v||_N = \lambda(L)$.

Zgodnie z [1] problem opierający się na normie Euklidesowej jest NP-trudny dla losowych redukcji.

CVP Jest to uogólnienie SVP, gdzie trudność SVP implikuje taką samą trudność dla CVP. W tym problemie dla ustalonej bazy przestrzeni wektorowej V oraz metryki M (zazwyczaj jest tu wykorzystywana metryka Euklidesowa) na wejściu opisującym kratę L oraz wektor $v \in V$ (ale niekoniecznie $v \in L$), należy znaleźć wektor na L najbliższy v według miary M.

Zauważmy, że dla wektorów bazowych z $\mathbb{R}^n$ powyżej zdefiniowane $\lambda(L)$ może nie istnieć, gdy np. wektory w bazie są niewspółmierne (jak $|1|$ i $|\sqrt{2}|$). Wejściem do naszych algorytmów są jednak zawsze wektory o współczynnikach wymiernych, co gwarantuje istnienie rozwiązania. Co więcej, możemy narzucić też ograniczenie na wielkość współczynników szukanej kombinacji liniowej aby ograniczyć przestrzeń poszukiwań.

2.2.2. LWE

Terminem LWE (ang. *learning with errors*) określa się w uczeniu maszynowym problem, w którym na podstawie skończonej liczby próbek o pewnym jednolitym zaburzeniu należy wydedukować funkcję f, która była wykorzystana do utworzenia próbek. Został on sformułowany przez O. Regeva w 2005 roku w pracy [6].

Wejściem problemu LWE jest zbiór próbek postaci (x, y), gdzie $x \in \mathbb{Z}_q^n$ jest wektorem liczb całkowitych modulo q, a $y \in \mathbb{Z}_q$. Zakładamy, że dla pewnej ustalonej funkcji liniowej $f : \mathbb{Z}_q^n \longrightarrow \mathbb{Z}_q$, $y_i = f(x) + \varepsilon_i$, gdzie $i \leq n$ a ε_i jest błędem generowanym na podstawie ustalonego wcześniej losowego zaburzenia (ε_i może być równe zero). Algorytm rozwiązuje LWE jeśli z dużym prawdopodobieństwem może ustalić funkcję f lub jej bliskie przybliżenie.

W pracy opiszę niżej przykładowy problem LWE, na którym oparte jest szyfrowanie Regeva.

2.3. RSA

Jest to jeden z pierwszych systemów kryptograficznych oparty na szyfrowaniu z kluczem publicznym i jest szeroko stosowany w szyfrowaniu transmisji danych przez internet. Opiera się on na dwóch

odpowiednio dużych liczbach pierwszych, których iloczyn N i obliczenia w pierścieniu $\mathbb{Z}_N$ jest podstawą zaszyfrowania i odczytywania zaszyfrowanej wiadomości. Obecnie bezpieczeństwo tego algrytmu jest trudne do określenia, gdyż problem faktoryzacji może być, zgodnie ze stanem naszej wiedzy, rozwiązywalny w czasie wielomianowym. Nie ma jednak obecnie opublikowanych metod łamania tego szyfru jeżeli zastosowany jest odpowiednio duży klucz. Z uwagi na to, że algorytm ten jest stosunkowo wolny, jest on głównie wykorzystywany do przekazywania kluczy wykorzystywanych w szybszych algorytmach kryptografii symetrycznej.

2.4. Algorytm Shora dla rozkładu liczb na czynniki pierwsze

Jest to algorytm przeznaczony na komputery kwantowe, który rozwiązuje wspomniany wyżej problem faktoryzacji: dla liczby całkowitej N należy znaleźć jej czynniki pierwsze. Został on wymyślony w roku 1994 przez amerykańskiego matematyka Petera Shora. Algorytm ten jest wielokrotnie szybszy niż klasyczne algorytmy faktoryzujące i jest uważany za zagrożenie dla algorytmu RSA. Pierwsze udane implementacje tego algorytmu miały miejsce w 2001 roku, ale obecnie faktoryzowane są stosunkowo niewielkie liczby (chociaż wciąż bite są kolejne rekordy).

2.5. Kryptografia postkwantowa

Kryptografia postkwantowa zajmuje się tworzeniem i rozwojem algorytmów i protokołów, które mogą być uważane za bezpieczne wobec ataków przeprowadzanych za pomocą komputerów kwantowych. Potrzeba prac w tej dziedzinie i upowszechnianie jej osiągnięć została bardzo dobrze opisane w raporcie [3] stworzonym przez Europejski Instytut Norm Telekomunikacyjnych (ETSI), gdzie także opisane są główne podejścia do kwestii konstruowania postkwantowych algorytmów kryptograficznych:

- Kryptografia oparta na kratach – jest to ogólny termin określający systemy kryptograficzne wykorzystujące kraty, zarówno w konstukcji algorytmu jak i dowodzie bezpieczeństwa.
- Kryptografia oparta na wielu zmiennych – jest to ogólne określenie asymetrycznych systemów kryptograficzych, które opierają się na wielomianach wielu zmiennych. Jakkolwiek nie udało się stworzyć odpowiednio silnych metod szyfrowania opartych na wielu zmiennych, jest to podejście które tworzy bardzo wydajne algorytmy podpisów cyfrowych odpornych na łamanie za pomocą komputerów kwantowych.
- Kryptografia oparta na funkcjach haszujących – są to systemy kryptograficzne, których dowód bezpieczeństwa opiera się na bezpieczeństwie funkcji haszujących. Podejście to ma zastosowanie jedynie w obszarze podpisów cyfrowych.
- Kryptografia oparta na kodach korygujących błędy – tego typu systemy kryptograficzne opierają się na trudności dekodowania ogólnego kodu liniowego (to znaczy takiego, w którym ciąg słów-kodów jest także słowem-kodem). Jednym z przedstawicieli tego systemu jest algorytm McEliece.

Rozdział 3

Metoda szyfrowania z kluczem publicznym oparta na LWE

3.1. Istota działania algorytmu

Algorytm opiera się na prostych założeniach: posiadając klucz publiczny można zaszyfrować wiadomość poprzez odpowiednie obliczenia. Co więcej, przy odpowiednio dobranych parametrach algorytmu, macierze klucza prywatnego oraz publicznego a także zaszyfrowana wiadomość przypominają zwykłe, losowe dane. Zaszyfrowany komunikat można ujawnić jedynie za pomocą klucza prywatnego.

Stwierdzenie, iż algorytm ten „opiera się na LWE", należy rozumieć następująco: jak zostanie pokazane poniżej, odczytanie klucza prywatnego z klucza publicznego jest utrudnione poprzez wprowadzone podczas szyfrowania zaburzenie klucza prywatnego za pomocą losowo wygenerowanej macierzy. Aby na podstawie klucza publicznego obliczyć klucz prywatny, czyli złamać szyfr, należy rozłożyć klucz publiczny na parę, którą będą właśnie macierze klucza prywatnego oraz macierz zaburzenia, przy czym nieznane są zależności między nimi. Uważa się wobec tego, że jest to problem o trudności porównywalnej do problemu LWE lub CVP.

3.2. Opis działania algorytmu

Parametry algorytmu Parametrami opisywanego algorytmu są następujące elementy:

- $t \in \mathbb{Z}$ – rozmiar alfabetu słów, które szyfrujemy,
- $l \in \mathbb{Z}$ – długość szyfrowanej wiadomości, o której myślimy jako o wektorze, elemencie $\mathbb{Z}_t^l$,
- $q \in \mathbb{Z}$ – jest to liczba pierwsza określająca ciało $\mathbb{Z}_q$, nad którym rozpatrujemy kratę (wymagamy spełnienia zależności $q > t$),
- $n \in \mathbb{Z}$ – długość wektora wykorzystywanego do szyfrowania pojedynczej litery; jest to w pewnym sensie główny parametr bezpieczeństwa algorytmu, który też posiada największy wpływ na jego wydajność,
- $m \in \mathbb{Z}$ – rozmiar drugiego wymiaru przestrzeni klucza publicznego postaci $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ oraz macierzy zaburzeń postaci $\mathbf{E} \in \mathbb{Z}_q^{m \times l}$,
- $r \in \mathbb{Z}$ – parametr zakresu zaburzenia elementów klucza publicznego przy szyfrowaniu wiadomości,

- $\alpha \in \mathbb{R}$ – parametr rozkładu jednostajnego (normalnego), $\alpha > 0$, służącego do generowania losowego zaburzenia.

Rozkład Ψ_α wykorzystywany przy generowaniu macierzy zaburzenia klucza prywatnego $\mathbf{E}$ zdefiniowany jest w sposób następujący: dla $\alpha > 0$ niech Ψ_α oznacza rozkład na $\mathbb{Z}_q$, gdzie próbkujemy zmienną normalną na $\mathbb{R}$ ze średnią 0 oraz odchyleniem standardowym $\alpha q/\sqrt{2\pi}$, następnie zaokrąglając otrzymaną liczbę do najbliższej liczby całkowitej, a na końcu obliczając jej wartość modulo q.

Tłumacząc litery komunikatu należące do $\mathbb{Z}_t$ na litery szyfrogramu, należące do $\mathbb{Z}_q$, będziemy posługiwać się funkcją f zdefiniowaną jak poniżej,

$$f(x) = \lceil x(q/t) \rfloor,$$

gdzie $\lceil x \rfloor$ to funkcja zaokrąglająca swój argument do najbliższej liczby całkowitej.

W dalszej części pracy będziemy posługiwać się symbolem f tylko w powyższym znaczeniu. Funkcja f przekształca w prawie liniowy sposób alfabet komunikatu $\mathbb{Z}_t$ na alfabet szyfrogramu $\mathbb{Z}_q$. Ponieważ algorytm szyfrujący działa na wektorach liczb, potrzebujemy zdefiniować też wektoryzację funkcji f, ozn. $\overrightarrow{f}$, w następujący sposób.

Dla wektora wartości $\overrightarrow{x} = [x_1, \ldots, x_r]$,

$$\overrightarrow{f}(\overrightarrow{x}) = [f(x_1), \ldots, f(x_n)]$$

Będziemy też potrzebowali funkcji odwracającej działanie funkcji f i $\overrightarrow{f}$. Oznaczamy je przez f^* i $\overrightarrow{f}^*$ i definiujemy następująco. Dla $x \in \mathbb{Z}_t$,

$$f^*(x) = \lceil x(t/q) \rfloor.$$

Funkcja $\overrightarrow{f}^*$ jest zaś wektoryzacją f^* analogicznie jak funkcja $\overrightarrow{f}$ jest wektoryzacją f.

Konstrukcja klucza prywatnego Klucz prywatny to macierz $\mathbf{S} \in \mathbb{Z}_q^{n \times l}$, której elementy są losowane według rozkładu jednostajnego dyskretnego.

Konstrukcja klucza publicznego Klucz publiczny to para ($\mathbf{A}$, $\mathbf{P}$), która tworzona jest w następujący sposób:

1. Elementy macierzy $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ losowane są według rozkładu jednostajnego dyskretnego,
2. Elementy macierzy $\mathbf{E} \in \mathbb{Z}_q^{m \times l}$ losowane są według rozkładu Ψ_α zdefiniowanego jak wyżej,
3. Macierz $\mathbf{P}$ jest wynikiem operacji $\mathbf{P} = \mathbf{AS} + \mathbf{E}$.

Szyfrowanie Mając element przestrzeni wiadomości $\mathbf{v} \in \mathbb{Z}_t^l$ oraz klucz publiczny ($\mathbf{A}$, $\mathbf{P}$) należy zastosować funkcję szyfrującą

$$e(\mathbf{v}) = (\mathbf{u}, \mathbf{c}) = \left(\mathbf{A}^T\mathbf{a}, \mathbf{P}^T\mathbf{a} + \overrightarrow{f}(\mathbf{v})\right) \in \mathbb{Z}_q^l \times \mathbb{Z}_q^l$$

gdzie

- ($\mathbf{A}$, $\mathbf{P}$) – klucz publiczny,
- $\mathbf{a} \in \mathbb{Z}^m$ – wektor, którego współrzędne losujemy zgodnie z rozkładem jednostajnym dyskretnym z zakresu $[-r, r]$,

Szyfrogram $e(\mathbf{v})$ będziemy oznaczać często jak powyżej, przez $(\mathbf{u}, \mathbf{c})$. Jak widzimy, w procesie szyfrowania wykorzystujemy wygenerowany losowo wektor $\mathbf{a}$. Z jednej strony zwiększa to koszt procesu szyfrowania, z drugiej strony jednak sprawia, że dane po zaszyfrowaniu wyglądają na bardziej losowe.

Deszyfrowanie Mając tekst zaszyfrowany $(\mathbf{u}, \mathbf{c})$ oraz klucz prywatny $\mathbf{S}$ należy zastosować funkcję deszyfrującą

$$d(\mathbf{u}, \mathbf{c}) = \overrightarrow{f}^*(\mathbf{c} - \mathbf{S}^T\mathbf{u}).$$

3.3. Poprawność algorytmu szyfrowania LWE

Podchodząc do dowodzenia poprawności niniejszego algorytmu należy zaznaczyć, iż nie jest to dowód poprawności działania jak w klasycznych algorytmach szyfrujących, gdzie wykazujemy, że funkcja szyfrująca e oraz deszyfrująca d spełniają następującą zależność

$$\forall x \, d\left(e\left(x\right)\right) = x.$$

Przykładem takiego algorytmu może być RSA, którego zasada działania opisana jest dokładnie w [4]. Szyfrowanie i deszyfrowanie w RSA polega na podnoszeniu argumentu do ustalonej potęgi modulo iloczyn dwóch liczb pierwszych pq. Poprawność działania RSA można wyrazić następującym twierdzeniem.

Twierdzenie 3.3.1. *(O poprawności algorytmu RSA)*
$\forall m \in \mathbb{Z} \;\; \forall p, q \colon p, q \in \mathbb{P} \;\; \forall e, d \colon ed \equiv 1 \big(\textit{mod}\ (p-1)(q-1)\big)$

$$(m^e)^d \equiv m \ (\textit{mod}\ pq)$$

W przypadku algorytmu zaproponowanego przez Regeva w [5] dowód poprawności ma inną postać, gdyż identyczność złożenia funkcji szyfrującej i deszyfrującej otrzymujemy tylko z pewnym prawdopodobieństwem, które można maksymalizować poprzez odpowiedni dobór parametrów algorytmu. Od doboru parametrów zależy również działanie poszczególnych funkcji algorytmu, wobec czego dowód składa się z następujących elementów:

- pokazanie poprawności działania funkcji mapujących f i f^*,
- określenie warunków odporności algorytmu na wprowadzane zaburzenia,
- oszacowanie prawdopodobieństwa wystąpienia błędu w odszyfrowanym komunikacie.

Pokażemy teraz, w dużym uproszczeniu, problem związany z dodawaniem losowych wartości podczas szyfrowania. Załóżmy, że dodajemy do wartości $f(x)$ losową wartość ε, a następnie obliczamy $f^*\big(f\left(x\right) + \varepsilon\big)$.

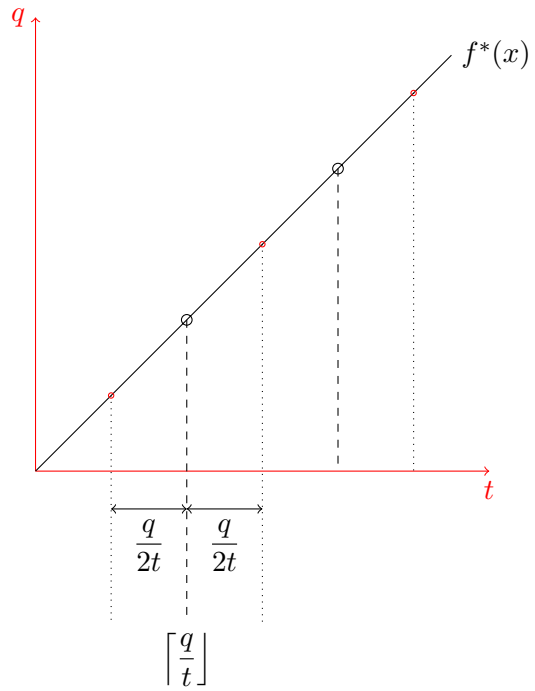
Aby $f^*\big(f\left(x\right) + \varepsilon\big) = x$, złożenie tych funkcji musi być odporne na zaburzenie ε. W tym celu wartość ε musi znajdować się w zakresie $\left(-\frac{q}{2t}, \frac{q}{2t}\right]$, aby funkcje mapujące dawały poprawne rezultaty, jak zostało to pokazane na rys. 3.1. W oryginalnym algorytmie zaburzenie to jest wprowadzane macierzą E. Widać zatem, że musi być spełniony następujący warunek:

$$\forall \varepsilon \in \mathbf{E} \;\; \left| \, \varepsilon \, \right| < \frac{q}{2t}$$

Powyższą dyskusję podsumuję twierdzeniem.

Twierdzenie 3.3.2 (O poprawności funkcji mapujących)**.** *Niech $t,\ q \in \mathbb{Z},\ t < q$. Niech $\varepsilon \in \mathbb{Z}$, błąd wynikający z wprowadzanych zaburzeń, taki, że $|\varepsilon| < \frac{q}{2t} - \frac{1}{2}$, a także niech $f : \ \mathbb{Z}_t \to \mathbb{Z}_q$ i $f^* : \ \mathbb{Z}_q \to \mathbb{Z}_t$ takie, że $f(a) = \left\lceil a(q/t) \right\rfloor$ oraz $f^*(b) = \left\lceil b(t/q) \right\rfloor$. Wówczas*

$$\forall a \in \mathbb{Z}_t \;\; f^*\big(f(a) + \varepsilon\big) = a.$$



Rysunek 3.1: Graficzne ukazanie warunku poprawnego działania funkcji mapujących

Dowód. Niech ε, $f(a)$ oraz $f^*(b)$ zdefiniowane jak wyżej. Wtedy

$$f^*\big(f(x)\big) \colon \mathbb{Z}_t \to \mathbb{Z}_t,$$

$$f^*\big(f(a)+\varepsilon\big) = \Big\lceil \big(\lceil a\,(q/t)\rfloor + \varepsilon\big)(t/q) \Big\rfloor$$

Przedstawiamy $f(a)$ jako

$$\lceil a\,(q/t)\rfloor = a\,(q/t) + e,$$

gdzie e - wartość wynikająca z zaokrąglenia do najbliższej liczby całkowitej, $e \in \left(-\frac{1}{2}, \frac{1}{2}\right]$. Wówczas niech

$$d = \big(a\,(q/t) + e + \varepsilon\big)\,(t/q) = a + (e+\varepsilon)\,(t/q)\,.$$

Wtedy

$$f^*(f(a)+\varepsilon) = \lceil d \rfloor$$

Wystarczy, żeby

$$\big|(e+\epsilon)\,(t/q)\big| < \frac{1}{2}$$

aby

$$\lceil b \rfloor = a$$

Skoro zaś $e < \frac{q}{2t} - \frac{1}{2}$ i $\epsilon \leq \frac{1}{2}$, to łatwo otrzymujemy żądane ograniczenie, i tym samym tezę twierdzenia.

□

Powyższy prosty przykład ułatwi nam omówienie problemu poprawności szyfrowania LWE.

Przyjmijmy, że wybrane zostały klucz prywatny $\mathbf{S}$ oraz klucz publiczny $(\mathbf{A}, \mathbf{P})$, gdzie $P = AS + E$, dla pewnej macierzy zaburzeń E. Kluczem publicznym szyfrujemy wiadomość $\mathbf{v}$, którą następnie deszyfrujemy. Taką operację można przedstawić następująco. Niech para $(\mathbf{u}, \mathbf{c}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^l$ postaci

$$e(\mathbf{v}) = (\mathbf{u}, \mathbf{c}) = (\mathbf{A}^T\mathbf{a}, \mathbf{P}^T\mathbf{a} + \overrightarrow{f}(\mathbf{v})),$$

będzie zaszfrywoną wiadomością, gdzie $\mathbf{a}$ to losowy wektor z $[-r, r]^m$. Wtedy deszyfracja opisana jest następującym wzorem.

$$\begin{aligned} d(\mathbf{u}, \mathbf{c}) &= \overrightarrow{f}^* \left(\mathbf{P}^{\mathrm{T}}\mathbf{a} + \overrightarrow{f}(\mathbf{v}) - \mathbf{S}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{a} \right) \\ &= \overrightarrow{f}^* \left((\mathbf{AS} + \mathbf{E})^{\mathrm{T}}\mathbf{a} + \overrightarrow{f}(\mathbf{v}) - \mathbf{S}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{a} \right) \\ &= \overrightarrow{f}^* \left(\mathbf{E}^{\mathrm{T}}\mathbf{a} + \overrightarrow{f}(\mathbf{v}) \right), \end{aligned}$$

Jak zostało pokazane, przy odpowiednim doborze parametrów funkcje mapujące dają poprawne rezultaty. Mając na uwadze warunek odporności funkcji mapujących na zaburzenia należy zauważyć, iż norma maksimum wektora $\mathbf{E}^T\mathbf{a}$, $||\mathbf{E}^T\mathbf{a}||_{\max}$, musi być nie większa od $\frac{q}{2t} - \frac{1}{2}$. Twierdzenie o poprawności dla niniejszego algorytmu polega na oszacowaniu prawdopodobieństwa wystąpienia błędu w odszyfrowanym komunikacie, to znaczy zdarzenia $||\mathbf{E}^T\mathbf{a}||_{\max} \geq \frac{q}{2t}$.

Twierdzenie 3.3.3 (O prawdopodobieństwie wystąpienia błędu przy deszyfracji [5])**.** *Prawdopodobieństwo wystapienia błędu dla każdej litery może zostać oszacowane przez prawdopodobieństwo, że wartość bezwzględna liczby losowanej według rozkładu normalnego o średniej 0 i odchyleniu standardowym $aq\sqrt{r(r+1)m/(6\pi)}$ jest większa od $\frac{q}{2t}$. Zachodzi następujące szacowanie:*

$$\textit{prawdopodobieństwo wystąpienia błędu dla każdej z liter} \approx 2\left(1 - \Phi\left(\frac{1}{2t\alpha} \cdot \sqrt{\frac{6\pi}{r(r+1)m}}\right)\right)$$

gdzie Φ to dystrybuanta rozkładu normalnego.

Przy dobieraniu parametrów algorytmu należy zwrócić szczególną uwagę na to, aby prawdopodobieństwo wykroczenia zaburzeń poza określony wyżej próg było możliwie małe. Jednocześnie poszczególne wartości macierzy zaburzeń nie mogą być zbyt małe, aby odgadnięcie klucza prywatnego było problemem możliwie trudnym.

Dla ustalonego rozmiaru szyfrowanego alfabetu t, wpływ na szacowanie prawdopodobieństwa wystąpienia błędu przy deszyfracji poszczególnych liter mają parametry α, r oraz m. Aby prawdopodobieństwo to było możliwie jak najmniejsze wartość $\Phi\left(\frac{1}{2t\alpha} \cdot \sqrt{\frac{6\pi}{r(r+1)m}}\right)$ musi osiągać wartości jak najbliższe 1. Im niższy parametr α, tym większą wartość przyjmuje $\frac{1}{2t\alpha}$, co dla niedużej wartości iloczynu $r(r+1)m$ daje oczekiwane odpowiednio duże wartości Φ.

Prawdopodobieństwo wystąpienia błędu możemy znacząco obniżyć poprzez wykorzystanie kodu korygującego błędy. Dla przykładu przedstawimy najprostszy wariant kodu korygującego błędy. Każdy bit komunikatu powielamy i przesyłamy jako ciąg $2k + 1$, dla pewnego $k > 0$. Wówczas algorytm przy odczycie kolejnych ciągów bitów zwraca wartość, którą zawierała większość z nich, jako kolejny bit odczytanej wiadomości. Przykładowo, jeżeli chcemy zakodować bit 1 poprzez ciąg pięciu bitów, otrzymamy ciąg bitów 11111. Aby po przesłaniu do odbiorcy odczytana wartość była poprawna, ciąg odczytywany musi się składać z przynajmniej 3 bitów o wartości 1, zatem kod 10101, pomimo zawierania w sobie dwóch błędów, zostanie w dalszym ciągu odczytany poprawnie. Przy wykorzystaniu takiego jak na przykładzie kodu korekcji błędów możemy oszacować współczynnik zmniejszenia

ryzyka wystąpienia błędu dla poszczególnych liter wiadomości jako

$$P = \binom{5}{3} p^3(1-p)^2 + 5p^4(1-p) + p^5,$$

gdzie p - prawdopodobieństwo wystąpienia błędu w pojedynczym bicie. Należy przy tym pamiętać, że użycie kodu korygującego błędy może znacząco zwiększyć ilość danych, które trzeba przesłać, oraz ilość obliczeń które trzeba wykonać, aby przesłać i odebrać wiadomość.

Wniosek 3.3.4. *Jeżeli argument $\frac{1}{2t\alpha}\sqrt{\frac{6\pi}{r(r+1)m}}$ funkcji Φ jest odpowiednio duży, to znaczy parametr α jest bardzo mały i jednocześnie iloczyn $r(r+1)m$ nie jest zbyt duży (r jest małe, a m nie jest duże), wówczas prawdopodobieństwo wystąpienia błędu możemy określić jako „małe". Prawdopodobieństwo to można dodatkowo obniżyć w sposób znaczny poprzez użycie odpowiedniego kodu korygującego błędy.*

3.4. Wydajność algorytmu

Niniejszy algorytm jest stosunkowo wymagający obliczeniowo, przy czym jest to kwestia istotna przede wszystkim w wypadku zastosowań komercyjnych, gdzie szyfrowanie danych dla dużej liczby użytkowników musi odbywać się w bardzo krótkim czasie – w zastosowaniach „domowych" dopuszczalne są dłuższe czasy przetwarzania danych. Zgodnie z [5] algorytm szyfrowania powinien zostać zaprojektowany w następujący sposób:

- rozmiar klucza prywatnego: $nl \log q$,
- rozmiar klucza publicznego: $m(n+l)\log q$, gdzie $m = ((n+l)\log q + 200)/\log(2r+1)$.

Przy tak dobranych parametrach rozmiar szyfrowanych wiadomości to $l \log_2 t$ a rozmiar szyfrogramów to $(n+l)\log_2 q$. Wtedy otrzymamy następujące własności algorytmu:

- operacje wymagane do zaszyfrowania jednego bitu: $\Theta(m(1+\frac{n}{l}))$,
- operacje wymagane do odszyfrowania jednego bitu: $\Theta(n)$.

Algorytm ten posiada też bardzo duży (w porównaniu z innymi algorytmami) klucz publiczny. Jak pokazuje sam autor w [5], dla parametrów: $n = l = 233,\ m = 4536$ rozmiar tego klucza wynosi 3.96MB.

3.5. Bezpieczeństwo algorytmu

Zasadniczym argumentem pokazującym bezpieczeństwo tego algorytmu jest związek problemu obliczenia klucza prywatnego z klucza publicznego z trudnością problemu najkrótszego wektora (closest vector) lub LWE. Pokazuje to, że może być trudno obliczyć klucz prywatny i tym samym odczytać zaszyfrowaną wiadomość.

Klucz prywatny $\mathbf{S}$ jest wykorzystywany przy tworzeniu pary stanowiącej klucz publiczny $(\mathbf{A}, \mathbf{P})$, gdzie element $\mathbf{P} = \mathbf{AS} + \mathbf{E}$. Ponowne jego rozbicie na macierze składowe może być uważane za problem trudny do rozwiązania, gdyż nieznane są własności takie, jak chociażby „proporcje" pomiędzy elementami $\mathbf{S}$ oraz $\mathbf{E}$ na odpowiednich polach analizowanej macierzy, a analiza szyfrogramu, który powstał przy wykorzystaniu klucza publicznego nie niesie ze sobą statystycznych informacji na temat budowy klucza prywatnego.

W [5] autor pokazuje, że przy odpowiednim doborze parametrów algorytmu za pomocą szyfrowania kluczem publicznym $(\mathbf{A}, \mathbf{P})$ otrzymuje się szyfrogram bardzo podobny do pary $(\mathbf{A}', \mathbf{P}') \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^{m \times l}$ wylosowanej według rozkładu jednostajnego.

Wobec powyższego można porównać trudność obliczenia klucza prywatnego z klucza publicznego do rozwiązywania problemu LWE opisanego parametrami n, m, q, Ψ_α, lub też do problemu CVP (najbliższego wektora), gdzie dużo wyraźniej widać analogię pomiędzy problemem znaleznienia najbliższego wektora na kracie rozpiętej na odpowiedniej bazie, a obliczeniem iloczynu $\mathbf{AS}$ z macierzy $\mathbf{P} = \mathbf{AS} + \mathbf{E}$. Parafrazując, można określić zaburzony klucz prywatny $\mathbf{AS} + \mathbf{E}$ jako wektor, dla którego należy rozwiązać CVP, a $\mathbf{AS}$ jako jego rozwiązanie.

Uściślając to porównanie, aby poznać klucz prywatny należy dla macierzy $\mathbf{P}$, traktowanej jako zbiór wektorów, rozwiązać CVP, gdzie $\mathbf{E}$ jest wektorem zaburzeń, natomiast wektory z $\mathbf{A}$ definiują kratę, w której poszukujemy rozwiązania. Rozwiązaniem tym jest macierz $\mathbf{S}$. Widać tutaj też pewną analogię do LWE, gdzie z wykorzystaniem określonej ilości próbek (w tym wypadku wektorów) należy znaleźć rozwiązanie (wektory składające się na klucz prywatny).

Należy ponadto zauważyć, że zgodnie z [2] problem obliczenia klucza prywatnego (którego trudność jest porównywalna do SVP i CVP) jest w klasie NP (jego rozwiązanie można obliczyć w czasie wielomianowym). Wszystkie te problemy są też uważane za trudne do rozwiązania za pomocą obliczeń przeprowadzanych na komputerach kwantowych, dzięki czemu możemy rozważać ten algorytm jako obiecujący sposób szyfrowania odporny na ataki przy pomocy komputerów kwantowych.

Rozdział 4

Opis implementacji

Prezentowany algorytm jest prosty w implementacji, zwłaszcza przy odpowiednim doborze narzędzi programistycznych. We własnej implementacji tej metody wykorzystałem język Python wraz z biblioteką NumPy, która udostępnia przystępny zestaw metod operujących na macierzach. Dodatkowym atutem tego języka jest wbudowana biblioteka *secrets*, która pozwala na generowanie liczb losowych odpowiednich do zastosowań kryptograficznych. Bardzo przydatne było również środowisko PyCharm, które dodatkowo ułatwiło wykorzystywanie zasobów bibliotek języka. Dzięki wyżej wymienionym narzędziom udało się mi w niedługim czasie napisać funkcjonalny i niezwykle kompaktowy skrypt (całość może zająć nawet mniej niż 70 linii kodu). Implementacja algorytmu pozwala na obliczanie prawdopodobieństwa wystąpienia błędu dla litery. Ponadto można zmierzyć za pomocą biblioteki *time* czas działania programu, a także zasoby wykorzystywane przez poszczególne macierze. Dużą zaletą implementacji algorytmu jest możliwość automatycznego sprawdzenia wyników dla dużej liczby kombinacji parametrów poprzez odpowiednie wywołania funkcji.

Przykładowo, dla parametrów $n = 2136, m = 2008, l = 10136, t = 24, r = 1, q = 2003, \alpha = 0.00021$, gdzie algorytm był wykonywany na procesorze Intel i5 M520 2.4GHz, otrzymałem następujące wartości:

- ilość niepoprawnie zdeszyfrowanych znaków: 0,
- prawdopodobieństwo wystąpienia błędu dla litery: 2%,
- czas działania algorytmu (szyfrowanie oraz deszyfrowanie): 180.63 s,
- Rozmiar macierzy S: 173 MB,
- Rozmiar macierzy A+P: 197 MB,
- Rozmiar wektora v: 81 kB,
- Rozmiar macierzy u+c: 98 kB.

W procesie implementacji należy jednak zwrócić szczególną uwagę na działanie wykorzystywanych funkcji. W tym przypadku wymagana była szczególna wersja funkcji zaokrąglającej liczby do najbliższej liczby całkowitej. Wykorzystałem funkcję zaokrąglającą znajdującą się w bibliotece *Decimal* – funkcja round() języka Python nie działa w sposób wymagany przez algorytm, przez co może wpływać na osiągane wyniki.

Poniżej przedstawię najważniejsze funkcje implementacji.

4.1. Funkcja generująca klucz prywatny

```
def generate_private_key():
    private_key = zeros([n, l])
    for i in range(n):
        for j in range(l):
            private_key[i][j] = secrets.randbelow(q)
    return private_key
```

4.2. Funkcja generująca klucz publiczny

```
def generate_public_key(private_key):
    public_key_a = zeros([m, n])

    for i in range(m):
        for j in range(n):
            public_key_a[i][j] = secrets.randbelow(q)

    error = np.random.normal(0, alpha * q / np.sqrt(2*np.pi), (m, l))

    public_key_p = np.dot(public_key_a, private_key)
    public_key_p = np.add(public_key_p, error)
    public_key_p = np.around(public_key_p)

    return public_key_a, public_key_p
```

4.3. Funkcja szyfrująca

```
def encrypt(message, pub_key_a, pub_key_p):
    a = zeros([m])
    for i in range(m):
        a[i] = generate_a_element(r)

    u = np.dot(pub_key_a.transpose(), a)
    c = np.dot(pub_key_p.transpose(), a)
    for i, c_element, v_element in zip(range(l), c, message):
        c[i] = c_element + f(v_element)

    return np.array(u), np.array(c)
```

4.4. Funkcja deszyfrująca

```
def decrypt(u, c, S):
    msg = np.subtract(c, np.dot(S.transpose(), u))
    for i in range(l):
        msg[i] = f_rev(msg[i])
    return msg
```

Rozdział 5

Podsumowanie

Przedstawiona metoda szyfrowania jest tylko jedną z wielu już istniejących, które mają zapewnić ciągłe bezpieczeństwo informacji w świecie, gdzie obliczenia kwantowe będą coraz bardziej popularne. Posiadamy już teoretyczne podstawy postkwantowej kryptografii, oraz szereg implementacji stworzonych przez ekspertów.

Teraz jednak pojawia się kolejne wyzwanie – wdrożenie i rozpropagowanie tych rozwiązań w istniejących systemach. Są jednak już przeprowadzane testy implementacji algorytmów takich jak PICNIC (R&D Microsoft) oraz NewHope wraz z jego włączeniem do biblioteki OpenSSL oraz protokołu TLS (R&D Google). Ponadto w roku 2016 został rozpoczęty projekt Open Quantum Safe, który ma na celu rozwój oraz implementację algorytmów postkwantowych, a także włączanie ich do pojedynczej biblioteki *liboqs*, która przede wszystkim skupia się na algorytmach wymiany kluczy. Wydaje się, że właśnie one są w tym momencie najbardziej potrzebne gdyż „najsłabszym ogniwem" szyfrowanej komunikacji może być etap wymiany kluczy do szyfrowania symetrycznego.

Sam temat kryptografii postkwantowej może być punktem wyjścia do zadania wielu interesujących pytań, jak na przykład:

- Kiedy nadejdzie moment, w którym kryptografia postkwantowa będzie już konieczna dla poufności przesyłanych danych?
- Kto i kiedy zbuduje pierwszy komputer kwantowy, który będzie w stanie przeprowadzać ataki na obecnie wykorzystywane tradycyjne algorytmy?
- Jaki będzie pierwszy skuteczny atak komputerem kwantowym? Kto go będzie przeprowadzał – władze kraju, czy grupa ludzi?
- Czy będą istnieć metody łamiące algorytmy, które teraz są opracowywane?

Może być to także punkt wyjścia do poznawania samej kryptografii, która jest niezwykle interesującą dziedziną, i z którą mamy styczność – świadomie lub nie – wiele razy każdego dnia.

Bibliografia

[1] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 99–108, New York, NY, USA, 1996. ACM.

[2] Miklós Ajtai. The shortest vector problem in l2 is np-hard for randomized reductions (extended abstract). In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 10–19, New York, NY, USA, 1998. ACM.

[3] European Telecommunications Standards Institute. Quantum safe cryptography and security, June 2015.

[4] Czesław Kościelny, Mirosław Kurkowski, and Marian Srebrny. *Modern cryptography primer. Theoretical foundations and practical applications*. 01 2013.

[5] Daniele Micciancio and Oded Regev. Lattice-based cryptography. November 2008.

[6] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 84–93, New York, NY, USA, 2005. ACM.