МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

На правах рукопису
УДК 003.26:621.39:530.145

Жмурко Тетяна Олександрівна

МЕТОДИ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ПРОТОКОЛІВ КВАНТОВОЇ КРИПТОГРАФІЇ

Спеціальність 05.13.21 – системи захисту інформації

Дисертація на здобуття наукового ступеня
кандидата технічних наук

Науковий керівник:
Гнатюк Сергій Олександрович
кандидат технічних наук, доцент,
доцент кафедри безпеки
інформаційних технологій
ННІІДС НАУ

Київ – 2016

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ЕЦП – електронний цифровий підпис;

ЗІ – захист інформації;

ІБ – інформаційна безпека;

ІКТ – інформаційно-комунікаційні технології;

ІС – інформаційна система;

ІТ – інформаційні технології;

КЗІ – криптографічний захист інформації;

КПБЗ – квантовий прямий безпечний зв'язок;

КПШ – квантовий потоковий шифр;

КК – квантова криптографія;

КРК – квантовий розподіл ключів;

КРС – квантове розділення секрету;

КС – квантова стеганографія;

КТ – квантова телепортація;

КТІ – квантова теорія ігор;

КЦП – квантовий цифровий підпис;

НСД – несанкціонований доступ;

ПВП – псевдовипадкова послідовність;

AES – Advanced Encryption Standard (удосконалений стандарт шифрування);

DES	– Data Encryption Standard	(стандарт шифрування даних);
NIST STS	– National Institute of Standards and Technology Statistical Test Suite	(Статистичний пакет тестів Національного інституту стандартів і технології);
EPR	– Einstein, Podolsky, Rosen	(Ейнштейн, Подольські, Розен);
GHZ	– Greenberger, Horne, Zeilinger	(Грінбергер, Хорн, Цайлінгер);
QKD	– Quantum Key Distribution	(квантовий розподіл ключів);
QKS	– Quantum Key Server	(квантовий ключовий сервер);
QPN	– Quantum Private Network	(квантова приватна мережа);
RSA	– Rivest, Shamir, Adleman	(Райвест, Шамір, Адлеман).

ВСТУП

Актуальність. Впроваджені в усі найважливіші сфери діяльності суспільства новітні інформаційно-комунікаційні технології (ІКТ), з одного боку, відкривають широкі можливості щодо створення та використання сучасних мережевих та Інтернет сервісів, а з іншого боку – породжують цілу низку нових уразливостей та специфічних загроз. Відкритість та публічність таких сервісів разом з еволюцією атак у кіберпросторі (кібератак), суттєвим збільшенням користувачів ІКТ і обсягів інформації, яка обробляється, зберігається та передається за допомогою ІКТ, ставлять під загрозу конфіденційність інформації, яка, як правило, забезпечується методами симетричної та асиметричної криптографії, що не позбавлені певних недоліків. Симетричним методам, зокрема, характерна проблема розподілу секретних ключів, а асиметричні методи є повільними і потребують значних обчислювальних ресурсів. Крім того, стійкість усіх традиційних криптосистем залежить від обчислювальних можливостей порушника і базується на гіпотетичній неможливості розв’язання певного класу математичних задач за поліноміальний час – пошук по повністю невпорядкованій базі даних, факторизація та логарифмування в дискретних полях великого розміру. Проте ця гіпотеза може бути спростована за допомогою багатокубітних квантових комп’ютерів (D-Wave 2X), GRID-технологій, HPC та інших сучасних ІКТ. З огляду на це, великий інтерес викликає квантова криптографія (КК), яка не залежить від обчислювальних потужностей порушника, використовує специфічні унікальні властивості квантових частинок і ґрунтується на непорушності законів квантової фізики. Основними перевагами методів КК є можливість точного виявлення порушника і забезпечення, в деяких випадках, теоретико-інформаційної (абсолютної) стійкості. На сьогодні такі методи і системи пройшли складний шлях від теоретичних гіпотез і лабораторних експериментів до повноцінних комерційних рішень.

Значний внесок у розвиток теорії й практики квантової криптографії внесли такі вітчизняні та закордонні вчені: Ф. Балаж, Ч. Беннет, Ж. Брассар, Є. Васіліу, Н. Гісін, С. Гнатюк, О. Гомонай, А. Еккерт, П. Завадскі, У. Збінден, Ш. Імре, В. Кінзерявий, С. Кілін, О. Корченко, Н. Люткенхаус, С. Ніколаєнко, М. Нільсен, К. Румянцев, А. Семенов, В. Скарані, А. Цайлінгер, І. Чанг та ін.

Переважна більшість досліджень орієнтовані на методи квантового розподілу ключів (BB84, B92, SARG, E91, Гольденберга-Вайдмана, Коаші-Імото тощо), які дозволяють вирішити проблему розподілу ключів шифрування в умовах секретності і використовуються, як правило, у комплексі з симетричними криптографічними методами (AES, 3DES). Іншим важливим напрямком КК є використання методів квантового прямого безпечного зв'язку (КПБЗ), які дозволяють передавати інформацію відкритим каналом напряму (без попереднього її шифрування – проблема розподілу ключів нівелюється). На сьогодні запропоновано велику кількість методів КПБЗ, що базуються на різних квантових технологіях і можуть використовуватись як для захищеного передавання інформації (за допомогою кубітів або кудитів), так і для розподілу криптографічних ключів. З точки зору інформаційної місткості найбільш ефективними методами є ті, що використовують трійкові квантові системи (кутрити), оскільки найбільшу щільність запису інформації має система числення з основою рівною основі натуральних логарифмів, тобто рівною числу Ейлера (для цілочисельних – це трійкова або тритова система). Зважаючи на асимптотичну стійкість методів КПБЗ відомий підхід до підвищення (забезпечення) стійкості до некогерентних атак зокрема кутритових методів шляхом застосування зворотного хешування за допомогою оборотних трійкових матриць. Генерування останніх потребує великих часових та ресурсних затрат (значна кількість математичних перетворень над полем $GF(3)$), а з огляду на те, що існуючі методи генерування (генератори) орієнтовані на бінарні системи, а методи оцінки рівня випадковості, як правило, працюють з двійковими псевдовипадковими послідовностями (ПВП), то

складно оцінити ефективність і доцільність застосування запропонованого підходу до підвищення стійкості методів КПБЗ.

З огляду на це, розробка і дослідження нових ефективних методів забезпечення стійкості кутритових протоколів квантової криптографії до некогерентних атак, побудови тритових генераторів ПВП та оцінювання їх якості (можливості використання для криптографічних застосувань) є *актуальною науково-практичною задачею,* що має теоретичне і практичне значення.

Зв'язок роботи з науковими програмами, планами, темами. Тематика дисертаційної роботи та одержані результати безпосередньо пов’язані з «Основними науковими напрямами та найважливішими проблемами фундаментальних досліджень у галузі природничих, технічних і гуманітарних наук НАН України на 2014-2018 роки» в частині п.1.2.8.1. «Розробка методів та інформаційних технологій розв’язання задач комп’ютерної криптографії та стеганографії», зі Стратегією національної безпеки України від 26 травня 2015 року № 287/2015 у контексті п.4.12 «Забезпечення кібербезпеки і безпеки інформаційних ресурсів, зокрема реформування системи технічного і криптографічного захисту інформації з урахуванням практики держав-членів НАТО та ЄС», зі Стратегією кібербезпеки України від 15 березня 2016 року №96/2016 і Рамковою програмою ЄС з досліджень та інновацій «Горизонт 2020», зокрема за напрямками DS-05-2016 та DS-06-2017 («Нові напрямки інноваційних наукових досліджень в Європі щодо забезпечення кібербезпеки як відповідь на сучасні виклики, зокрема квантова криптографія»). Результати роботи відображені у звітах держбюджетних науково-дослідних робіт Національного авіаційного університету «Організація систем захисту інформації від кібератак» (д.р. № 0111U000171), «Методи та засоби захисту інформації на основі квантових технологій» (реєстраційний номер № 43/14.02.04), «Методи забезпечення конфіденційності державних інформаційних ресурсів в інформаційно-комунікаційних системах» (реєстраційний номер № 61/09.01.08), «Новітні технології криптографічного

захисту інформації» (реєстраційний номер № 100/14.01.06), «Методи підвищення ефективності систем квантової криптографії» (реєстраційний номер № 26/09.01.08) та Кіровоградського національного технічного університету «Розробка методів синтезу тестових моделей поведінки програмних об’єктів, підвищення оперативності передачі та захисту інформації у телекомунікаційних системах», (д.р. № 0115U003103), у яких здобувач брав участь у якості виконавця.

Мета і задачі дослідження. Метою дисертаційної роботи є підвищення ефективності протоколів квантової криптографії шляхом розробки методів забезпечення стійкості кутритових протоколів і систем, побудови тритових генераторів псевдовипадкових послідовностей та оцінювання їх якості.

Для досягнення поставленої мети **необхідно розв’язати такі основні задачі:**

– проаналізувати сучасні методи та протоколи квантової криптографії, їх ефективність і стійкість до різного роду кібератак для їх класифікації і чіткого визначення завдання дослідження;

– розробити метод забезпечення стійкості кутритових протоколів квантової криптографії до некогерентних атак, що не потребує великих часових та ресурсних затрат;

– розробити метод генерування тритових псевдовипадкових послідовностей для криптографічних застосувань, зокрема для реалізації метода забезпечення стійкості кутритових протоколів квантової криптографії до некогерентних атак;

– розробити метод оцінювання якості (рівня випадковості) тритових псевдовипадкових послідовностей для визначення криптостійкості (оцінювання статистичних параметрів та закономірностей) тритових генераторів і доцільності використання сформованих трійкових послідовностей для криптографічних застосувань;

– розробити спеціалізоване програмне забезпечення та методику для проведення експериментів і верифікації запропонованих методів.

Об'єктом дослідження є процес захисту інформації методами квантової криптографії.

Предметом дослідження є методи, способи та моделі підвищення ефективності протоколів квантового прямого безпечного зв’язку.

Методи дослідження. Проведені дослідження базуються на сучасних методах квантової теорії інформації, квантової механіки та імітаційного моделювання (моделювання процесу передавання кутритів за протоколами КПБЗ, моделювання методів забезпечення стійкості від некогерентних атак, дослідження кібератак на квантові системи), традиційної криптографії (розробка методів забезпечення стійкості та формування трійкових ПВП), об’єктно-орієнтованого програмування (розробка програмного забезпечення для реалізації запропонованих методів) та математичної статистики (розробка низки статистичних тестів для оцінювання якості трійкових ПВП).

Наукова новизна одержаних результатів полягає в наступному:

1) *отримав подальший розвиток* метод забезпечення стійкості кутритових протоколів квантової криптографії, який, за рахунок неквантової функції перевірки цілісності та використання тритової симетричної функції, дозволяє звести до мінімуму кількість перемикань між режимами протоколу (передавання повідомлення та контролю підслуховування), збільшити швидкість роботи при збереженні стійкості до некогерентних атак;

2) *отримав подальший розвиток* метод генерування псевдовипадкових послідовностей, який, за рахунок виконання нової послідовності операцій (підстановок, лінійного розсіювання, динамічного циклічного зсуву та додавання за модулем 3 та 3^l) над вектором внутрішніх станів V_p ($V_p = \{0,1,2\}^p$, $p = 14 \cdot l$) за $r \cdot b$ циклів, дозволяє формувати трійкові незбалансовані («0», «1», «2») псевдовипадкові послідовності $V_{m \cdot b}, m = 4 \cdot l$;

3) *отримав подальший розвиток* метод оцінювання якості псевдовипадкових послідовностей, який, за рахунок комплексної інтерпретації згенерованих чисел, введення диференційованих ймовірностей $P-value_{01}$, $P-value_{02}$, $P-value_{12}$ і трійкових коефіцієнтів для функції помилок *erfc* та неповної гамма функції *igamc*, дає можливість оцінювати статистичні параметри і закономірності тритових псевдовипадкових послідовностей.

Практичне значення одержаних результатів. Отримані в дисертаційній роботі результати можуть бути використані для підвищення ефективності (захищеності, швидкості роботи) систем захисту на базі квантового прямого безпечного зв'язку і квантового розподілу ключів, а також для деяких процедур безпеки в традиційних (неквантових) криптографічних системах захисту інформації. Практична цінність роботи полягає у наступному:

– розроблено класифікацію методів КК, яка, за рахунок розширення множини відомих базових ознак і часткових узагальнень теоретичних положень та практичних досягнень у галузі КК, дозволяє розширити можливості щодо вибору відповідних методів для побудови сучасних квантових систем захисту інформації (на базі КПБЗ та інших квантових технологій);

– використання результатів дисертаційного дослідження дозволило підвищити захищеність інформації з обмеженим доступом, що підтверджується актами впровадження у діяльність ТОВ «Сайфер БІС» (акт від 28.10.2015 року) та Bilfinger HSG (Німеччина) (акт від 03.09.2015 року);

– розроблено низку комп'ютерних програм, захищених свідоцтвами про реєстрацію авторського права на твір, зокрема «Імітаційна модель пінг-понг протоколу в квантовому каналі з шумом» (№ 36373 від 04.01.2011 року), «GenSBOX3» (№ 48037 від 26.02.2013 року), «TrytTon 2012» (№ 48040 від 26.02.2013 року) та «Model ping-pong protocol» (№ 48041 від 26.02.2013 року), подано заявку на отримання патенту України на корисну модель «Спосіб

підсилення стійкості квантових протоколів прямого безпечного зв’язку» u201512445 від 16.12.2015);

– результати дисертації використовуються у навчальному процесі кафедри безпеки інформаційних технологій Національного авіаційного університету (акт від 21.12.2015 року) та кафедри інформаційної безпеки Казахського національного дослідницького технічного університету ім. К.І. Сатпаєва (акт від 07.12.2015 року) для підвищення ефективності підготовки фахівців з інформаційної безпеки (кібербезпеки).

Особистий внесок здобувача. Основні положення і результати дисертаційної роботи, що виносяться до захисту, отримані автором самостійно. У роботах, написаних у співавторстві, автору належать: [56] – аналіз некогерентних методів перехоплення інформації та кібератак у системах КК; [140, 62, 205, 86] – метод забезпечення стійкості кутритових протоколів КК до некогерентних атак; [153] – дослідження методу маршрутизації для безпечного передавання інформації; [55, 20, 18, 204] – метод оцінювання якості ПВП для визначення статистичних параметрів і закономірностей тритових ПВП; [192, 159] – розрахунок для деяких параметрів пінг-понг протоколу КПБЗ необхідних розмірів матриць для хешування блоків повідомлення; [21, 20, 18, 23] – метод генерування трійкових незбалансованих ПВП; [88, 206, 16, 209] – узагальнена класифікація методів КК, зокрема введення додаткових методів і класифікаційної ознаки; [28, 29] – візуалізація процесів моделювання за допомогою пакетів комп’ютерної алгебри задач Бюффона і регресії для вивчення теорії ймовірності та математичної статистики; постановка задачі і формулювання висновків у співавторстві; [206, 141, 207] – аналіз сучасних систем КК та їх комерційних реалізацій; [16] – дослідження сучасних протоколів квантової теорії ігор.

З робіт, що опубліковані у співавторстві, у дисертаційній роботі використовуються виключно результати, отримані особисто здобувачем.

Апробація результатів дисертації. Основні положення дисертаційної роботи доповідалися та обговорювалися на таких наукових конференціях: Основні положення дисертаційної роботи доповідалися та обговорювалися на таких наукових конференціях: МНТК «ITSEC: Безпека інформаційних технологій» (Київ, 2012 р.), МНПК «Інтегровані інтелектуальні робототехнічні комплекси (ІІРТК)» (Київ, 2011 р., 2012 р., 2013 р.), НТК студентів та молодих учених «Наукоємні технології» (Київ, 2012 р.), Всесвітній конгрес «Авіація у XXI столітті» – «Безпека в авіації та космічні технології» (Київ, 2012 р., 2014 р.), МНПК «Проблеми і перспективи розвитку ІТ-індустрії» (Харків, 2013 р.), МНТК «АВІА» (Київ, 2013 р.), Міжнар. конф. «Computer Science & Engineering (CSE)» (Львів, 2013 р.), НПК «Стан та удосконалення безпеки інформаційно-телекомунікаційних систем (SITS)» (Миколаїв, 2014 р., 2015 р.), НПК «Актуальні питання забезпечення кібербезпеки та захисту інформації» (Київ, 2015 р.), Всеукр. НПК «Перспективні напрями захисту інформації» (Одеса, 2015 р.), Міжвідомчий міжрегіональний семінар Наукової Ради НАН України «Технічні засоби захисту інформації» (Київ 2012 р., 2013 р., 2015 р.), Міжнар. конф. «Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS’2015)» (Варшава, 2015 р.) та ін.

Публікації. Основні положення дисертації опубліковано у 24 наукових працях, у тому числі – 1 колективна монографія, 10 наукових статей (2 – у міжнародних рецензованих виданнях, що входять до бази даних SCOPUS, 6 – у вітчизняних фахових наукових журналах та 2 – у інших наукових виданнях), 1 заявка на отримання патенту України на корисну модель, а також 12 матеріалів і тез доповідей на конференціях.

Структура роботи та її обсяг. Дисертація складається із вступу, чотирьох розділів, загальних висновків, додатків, списку використаних джерел і має 127 сторінок основного тексту, 39 рисунків, 16 таблиць, 44 сторінки додатків. Список використаних джерел містить 209 найменувань і займає 22 сторінок. Загальний обсяг роботи 193 сторінки.

РОЗДІЛ 1
СУЧАСНІ МЕТОДИ КВАНТОВОЇ КРИПТОГРАФІЇ

1.1. Аналіз методів і протоколів квантової криптографії

З найдавніших часів люди намагались знайти такі способи комунікації, які б забезпечували збереження переданої інформації в таємниці від третіх осіб, що було, є і буде актуально для потреб дипломатії (держави), торгівлі, військової справи та інших галузей. Технології передачі конфіденційної та секретної інформації можна поділити на три групи [46, 83, 91]:

– створення *абсолютно секретного каналу зв'язку* (найскладніший метод, складність реалізації, якого зростає з розвитком технологій підслуховування);

– приховування самого факту передачі інформації – *стеганографія* (недоліком є те, що досить важко гарантувати непопадання інформації третім особам, і при довготривалому використанні одного і того ж способу стеганографії велика ймовірність того, що зловмисник також читає повідомлення, не виказуючи при цьому себе);

– передача повідомлення з використанням шифрування, під яким розуміємо неможливість отримання корисної інформації без знання відповідних даних (секретного ключа), даний метод отримав назву – *криптографія*.

Найбільшого розповсюдження отримала – криптографія. Однак, так чи інакше, всі методи традиційної криптографії базуються на математичних процедурах та схемах [38, 78, 91], деякі з яких хоча і дуже складно проте можливо розв'язати. Отож, постає питання пошуку нових технологій, які могли би стати альтернативною заміною традиційній криптографії. Одним з можливих варіантів може бути КК [15, 38, 78, 90, 93, 155], що є найбільш розвинутим напрям-

ком квантової теорії інформації, та базується на принциповій непорушності постулатів квантової механіки.

Квантова криптографія – це порівняно новий напрям досліджень, що дозволяє застосовувати ефекти квантової фізики для створення секретних каналів передачі даних [15, 61, 78]. Вона поєднує у собі цілу низку наук, таких як квантова механіка, інформатика, теорія інформації, квантові обчислення та криптографія, та сформувалась внаслідок виникнення задач, що не мали класичного розв’язку, у перерахованих галузях.

Розглянемо революційні задачі двох складових поняття «квантова криптографія», що призвели до утворення нового напрямку:

1. З точки зору *фізики (а саме, квантової механіки).* Ньютон створив загальну картину світу, в якій механіка виступала як універсальний закон руху матеріальних точок або частинок – маленьких порцій матерії [58]. З цих частинок можна було побудувати будь-які об'єкти. Здавалося, що механіка Ньютона здатна теоретично пояснити всі природні явища. Однак наприкінці позаминулого століття з'ясувалося, що класична механіка не здатна пояснити закони теплового випромінювання нагрітих тіл [4]. Це питання привело до необхідності переглянути фізичні теорії і вимагало нових ідей.

Одним з поштовхів до створення нової теорії став закон про спектр теплового випромінювання англійських фізиків Джона Релея та Джеймса Джинса (1905 р.) [4, 58]. Отриманий ними об'єднаний закон свідчив, що інтенсивність випромінювання, що випускається нагрітим тілом, прямо пропорційна його абсолютній температурі і обернено пропорційна квадрату довжини хвилі випромінюваним ним світла.

Цей закон, здавалося, добре відповідав дослідним даними, але несподівано з'ясувалося, що він правдивий тільки для довгохвильової та середньої частини видимого спектру – там, де розташовуються зелені і жовті кольори. У міру

наближення до синіх, фіолетових і ультрафіолетових променів закон все більш порушувався. Згодом це охрестили – «ультрафіолетовою катастрофою» [4].

Проте у 1900 р. з'явилася робота німецького фізика Макса Планка, в якій він припустив, що випромінювання відбувається порціями, квантами. Таке уявлення суперечило класичним поглядам, але чудово пояснювало результати експериментів (у 1918 р. робота була удостоєна Нобелівської премії з фізики) [34, 58]. Відкриття Планка стало подією, яка ознаменувала початок квантової ери.

2. З точки зору *криптографії*. Основною задачею криптографії є шифрування даних та аутентифікація відправника. Відправник повинен провести певні перетворення, можливо при використанні додаткових даних, які називаються ключем, таким чином, щоб отримувач міг по одержаному ним повідомленню визначити, чи було воно змінено.

Класичний підхід полягає у тому, що ключ, котрий використовують для зашифровування, так і для розшифрування повідомлення, повинен бути відомим лише відправнику та отримувачу. Такі системи називаються криптосистемами з закритим ключем. Надійність процедури шифрування доведена лише для метода «одноразових блокнотів», запропонованого в 1917 р. Гілбертом Вернамом [24, 26, 61]. Ідея полягає у тому що обидва учасники обмінюються набором секретних ключів, кожний з яких використовується для шифрування тільки одного повідомлення. Ключі генеруються випадково і ніякої інформації не несуть. Процес шифрування полягає у тому, що кожний символ вихідного повідомлення «складається» з відповідним символом ключа (так що ключ повинен бути досить довгим, а повідомлення – коротким). В «до комп'ютерний» час ключі зберігали в блокнотах з відривними листами (звідси і назва методу). Кожний листок блокноту знищувався після використання. При використанні систем телекомунікації виникає проблема забезпечення секретності під час обміну ключами, оскільки ключ повинен бути доставлений отримувачу повідом-

лення попередньо і з дотриманням строгої секретності. Іншими словами, конфіденційно обмінятися повідомленнями дозволяють ключі, проте як обмінятися самими ключами? Даний недолік носить назву проблема розподілу ключів.

Умови секретності, власне і являються головним недоліком шифру Вернама [26]: необхідно використати випадковий ключ такої ж довжини, як і повідомлення, котре необхідно передавати, при чому використовувати даний ключ можна лише один раз. Відповідно, перед тим, як передати таємне повідомлення, необхідно спочатку передати по захищеному від НСД каналу, такої ж довжини повідомлення, яке містить у собі секретний ключ. Така система є досить громіздкою, незручною у використанні та дорогою, тому і використовується досить рідко.

У 70-х роках було розроблено криптографічну систему з відкритим ключем, де використовується два ключі [26]: один для зашифровування повідомлення (оголошується публічно), а інший для розшифрування (зберігається у секреті). Запропонована система базується на тому, що використовуються спеціальні функції, які розрахувати в одному напрямку дуже легко, а в протилежному – досить складно. Зокрема, проблема обчислення секретного ключа при наявності публічного зводиться до проблеми факторизації великих чисел, на розв’язання, якої до сьогоднішні необхідно витратити від декількох днів до декількох місяців. Однак, у зв'язку з появою квантових комп'ютерів [143], для яких вже розроблені алгоритми швидкої факторизації [183], системи з публічним ключем можуть втратити свою ефективність.

Отож, постало питання використання нових систем ЗІ, принципово нових. З огляду на це, було проведено ряд досліджень, що значно вплинули на розвиток КК [101-108, 131, 134, 149, 150, 183, 184, 197, 200]: 1970 р. – Стівен Віснер винаходить поєднане кодування (сопряженное) [197], та публікує статтю у якій описує ідею можливості використання квантових станів для захисту грошових банкнот. 1973 р. – Олександр Холево видає статтю, в якій демонструє, що L ку-

бітів не може нести більше інформації, ніж *L* класичних бітів (теорема Холево) [149]. 1984 р. – Чарльз Беннет та Жиль Брасард використовують поєднане кодування Віснера для розподілу криптографічних ключів [104], описаний у їх роботі протокол признають першим і базовим протоколом КК і називають на честь науковців – BB84. 1985 р. – Девід Дойч описує перший універсальний квантовий комп'ютер [131], який буде мати обчислювальну потужність, що набагато переважатиме всі нинішні й майбутні комп'ютерні системи. 1991 р. – Артур Екерт розробляє безпечну модель комунікації, базуючись на переплутаності квантових станів [134], в основу покладено принципи парадоксу Ейнштейна-Подольского-Розена, а саме принцип не локальності сплутаних квантових об'єктів. 1994 р. – Пітер Шор розробляє алгоритм призначений для факторизації великих цілих чисел і розв'язку проблеми дискретного логарифмування [183]. 1995 р. – Пітер Шор та Ендрю Стін запропонували перші схеми кореляції квантових помилок [184]. 1996 р. – Лов Гровер створює квантовий алгоритм пошуку в неупорядкованій базі даних [145].

Більш детальний поетапний розвиток КК представлений у роботах [15, 26, 38, 78, 93, 177]. Як зазначається у [90] основна задача КК полягає у створенні каналу передачі інформації, абсолютна захищеність якого буде гарантуватись фундаментальними законами природи, що дозволяють зафіксувати будь-яку спробу проникнення ззовні. Процес створення секретного ключа в КК здійснюється за допомогою передачі одиночних фотонів, закодованих у певному базисі, по оптичному волокну і спирається на важливу теорему квантової механіки про неможливість клонування попередньо невідомого квантового стану [58, 61, 200]. В цьому випадку будь-яка спроба прослухати такий канал призведе до виникнення підвищеного рівня помилок, що може бути з легкістю зафіксованою.

КК базується на використанні в якості носія інформації фізичних систем, які підвладні законам квантової механіки [4, 15, 58, 61, 90]. Отож, для більш

глибоко розуміння принципів роботи методів і протоколів КК розглянемо основні положення квантової механіки, які складають теоретичну базу КК.

Зазначимо, що в дисертаційній роботі використовуються такі міжнародні позначення: Аліса – відправник повідомлення, Боб – отримувач повідомлення, Єва – зловмисник (від анг. eavesdropping – підслуховування).

Постулат 1. З кожної ізольованої фізичної системи зв'язується комплексний векторний простір зі скалярним добутком, тобто гільбертовий простір, який є простором станів систем. Квантова система повністю описується вектором станів, який є одиничним вектором у просторі станів системи [8, 61].

Квантовий біт (кубіт) – це найпростіша квантово-механічна система [34, 61, 90], є аналогом біту в класичній теорії інформації, який має два стани: «0» і «1». Базисні вектори позначаються, як $|0\rangle$ і $|1\rangle$. Символ $|\ \rangle$ називається позначення Дірака, та означає одиничний вектор у гільбертовому просторі [34]. Символ $\langle\ |$ – вектор, ермітово-спряжений вектору $|\ \rangle$. Довільний вектор станів у двомірному гільбертову просторі має вигляд [34, 37, 38, 61]:

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{1.1}$$

де α і β – комплексні числа, що задовольняють умові $|\alpha|^2 + |\beta|^2 = 1$. Ця умова слідує з одиничного вектору $|\Psi\rangle$: $\langle\Psi|\Psi\rangle = 1$, де символ $\langle\ |\ \rangle$ — скалярний добуток векторів [34, 61, 155].

На відміну від біту, кубіт може знаходитись у суперпозиції (1.1) [15, 34, 93, 155] двох базисних станів і тоді неможливо стверджувати, що кубіт знаходиться у стані $|0\rangle$, або у стані $|1\rangle$. Проте, відповідно до постулату виміру в квантовій механіці, при вимірі кубіта буде отримано значення «0» з ймовірністю $|\alpha|^2$ і значення «1» з ймовірністю $|\beta|^2$ [15, 34, 61] Також необхідно підкреслити, що формула (1.1) описує когерентну суперпозицію двох станів, а не їх некогерентну суміш. Для когерентної суперпозиції завжди існує базис, в якому значення кубіту чітко визначено. Так, для стану $|\Psi'\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)$ таким базисом

є, так званий, діагональний базис $\left\{|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle); |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right\}$, повернутий у гільбертовому просторі на 45° відносно базису $\{|0\rangle;|1\rangle\}$. Очевидно, що у такому базисі $|\Psi'\rangle = |+\rangle$.

На сьогодні існує велика кількість різноманітних фізичних реалізацій кубіту, які використовуються для створення квантових обчислювальних пристроїв [61, 78, 82, 87, 90]. Однак в протоколах КК, більшість з яких призначені для передач конфіденційної інформації, єдиним практичним носієм кубіта є фотон, який слабо взаємодіє з іншими фотонами і є одним з найстабільніших носіїв квантової інформації. У випадку, коли носієм кубіта є фотон [61, 82], базисні стани кубіта $|0\rangle$ і $|1\rangle$ відповідають, наприклад, вертикальній і горизонтальній поляризації фотона і іноді позначаються, як $|\updownarrow\rangle$ і $|\leftrightarrow\rangle$.

Кудит – d-рівнева квантова система, що є узагальненням поняття «кубіт» на багатовимірні квантові системи [61, 179].

Кутрит – частковий випадок d-рівневої квантової системи, при d=3 (аналог у класичній інформації – «трит») [15, 34, 61].

Кутритові системи – системи КК в основі, яких лежить використання кутритів, замість кубітів.

Напряму працювати з кутритами досить складно, простішим варіантом є заплутування кутрита з кубітом, після чого впливати на систему, впливаючи на кубіт (наприклад, проводячи виміри його стану) [155].

Постулат 2. Еволюція замкнутої квантової системи описується унітарним перетворенням [61]:

$$|\Psi'\rangle = U|\Psi\rangle, \tag{1.2}$$

де U – унітарний оператор.

Важливими унітарними операціями для одиночного кубіта є оператори Паулі [34, 61, 87]:

$$\sigma_x = |0\rangle\langle 1| + |1\rangle\langle 0|,\ \sigma_z = |0\rangle\langle 0| - |1\rangle\langle 1|,\ \sigma_y = i\sigma_x\sigma_z = -i|0\rangle\langle 1| + i|1\rangle\langle 0| \qquad (1.3)$$

На одиночний кубіт (1.1) вони діють таким чином:

$$\sigma_x|\Psi\rangle = \alpha|1\rangle + \beta|0\rangle,\ \sigma_z|\Psi\rangle = \alpha|0\rangle - \beta|1\rangle,\ i\sigma_y|\Psi\rangle = \alpha|1\rangle - \beta|0\rangle. \qquad (1.4)$$

Оператор σ_x проводить переворот кубіта, тобто. виконує перетворення $|0\rangle \to |1\rangle$ і $|1\rangle \to |0\rangle$; оператор σ_z перевертає фазу, тобто, змінює її на π: $|0\rangle \to |0\rangle$ і $|1\rangle \to -|1\rangle$; оператор $i\sigma_y$ одночасно перевертає кубіт і фазу [61, 90]: $|0\rangle \to |1\rangle$ і $|1\rangle \to -|0\rangle$.

Постулат 3. Квантові виміри описуються набором операторів $\{M_m\}$ виміру. Індекс показує результати виміру, які можуть бути отримані в результаті експериментів. Якщо перед вимірюванням квантова система знаходилась у стані $|\Psi\rangle$, то ймовірність того, що в результаті вимірювання буде отриманий результат *m*, визначається виразом [34, 37, 38]:

$$p(m) = \langle\Psi|M_m^\dagger M_m|\Psi\rangle. \qquad (1.5)$$

Після вимірювання стан квантової системи змінюється стрибком (колапс хвильової функції) і описується виразом:

$$|\Psi'\rangle = \frac{M_m|\Psi\rangle}{\sqrt{\langle\Psi|M_m^\dagger M_m|\Psi\rangle}}. \qquad (1.6)$$

Оператори виміру задовольняють умову повноти, яка означає, що сума ймовірностей різних результатів вимірювання дорівнює одиниці:

$$\sum_m M_m^\dagger M_m = I, \qquad (1.7)$$

де *I* – одиничний оператор.

Окремим випадком квантових вимірювань є вимірювання фон Неймана [8, 37, 38, 61]. У цьому випадку стан квантової системи проектується на один з власних станів вимірюваної величини. В результаті такого виміру система переходить у відповідний власний стан

$$|\Psi'\rangle = \frac{P_m|\Psi\rangle}{\sqrt{p(m)}}, \qquad (1.8)$$

де P_m – проектор на власний підпростір, відповідний оператору M з власним значенням m, а

$$p(m) = \langle \Psi | P_m | \Psi \rangle \tag{1.9}$$

представляє собою ймовірність отримання результату m.

Прикладом таких вимірювань можуть бути виміри в *обчислювальному базисі* $\{|0\rangle;|1\rangle\}$. Відповідні оператори вимірювання (проектори) мають вигляд: $P_0 = |0\rangle\langle 0|$, $P_1 = |1\rangle\langle 1|$.

Постулат 4. Простір станів складеної квантової системи являє собою тензорний добуток просторів станів, які входять до квантових систем [34, 61, 90].

Так, найбільш загальний стан системи двох кубітів записується у вигляді:

$$|\Psi\rangle = \alpha|0\rangle_1 \otimes |0\rangle_2 + \beta|0\rangle_1 \otimes |1\rangle_2 + \gamma|1\rangle_1 \otimes |0\rangle_2 + \delta|1\rangle_1 \otimes |1\rangle_2 \tag{1.10}$$

де α, β, γ і δ – комплексні коефіцієнти $\left(|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1\right)$; індекси позначають номера кубітів.

Зазвичай для скорочення запису (1.10) його записують у вигляді:

$$|\Psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle. \tag{1.11}$$

Переплутування (квантова кореляція).

Дві або більше квантові системи можуть бути переплутані (корельовані). Так, пара фотонів в синглетному поляризаційному стані [34, 37, 38, 61, 90]

$$|\Psi\rangle = \frac{1}{\sqrt{2}}\left(|01\rangle - |10\rangle\right), \tag{1.12}$$

є прикладом максимально переплутаного стану. Стан (1.12) називають парою Ейнштейна – Подольского – Розена (ЕПР-парою).

Якщо вимірювання виконуються у обчислювальному базисі над одним з двох переплутаних кубітів у (1.12), то результат буде випадковим: «0» або «1» з рівною імовірністю 1/2. Стан другого кубіту *антикорельовано* з станом першого, тобто перший кубіт в результаті вимірювання переходить у стан «0», то другий перейде у стан «1», і навпаки. Без проведення вимірювання, однак, ні один з цих кубітів не знаходиться у певному стані (стан кожного з цих кубітів окремо є

змішаним). Квантове переплутування, як і суперпозиція квантових станів, це виключно квантові ефекти, що не мають аналога для об'єктів класичної фізики.

Чотири максимально переплутаних ортогональних станів в системі двох кубітів утворюють базис Белла [34, 61]:

$$\left|\Phi^+\right\rangle = \frac{1}{\sqrt{2}}\left(\left|00\right\rangle + \left|11\right\rangle\right); \qquad \left|\Phi^-\right\rangle = \frac{1}{\sqrt{2}}\left(\left|00\right\rangle - \left|11\right\rangle\right);$$
$$\left|\Psi^+\right\rangle = \frac{1}{\sqrt{2}}\left(\left|01\right\rangle + \left|10\right\rangle\right); \qquad \left|\Psi^-\right\rangle = \frac{1}{\sqrt{2}}\left(\left|01\right\rangle - \left|10\right\rangle\right). \tag{1.13}$$

Одним з простих переплутаних стані трьох кубітів є стан Грінбергера-Хорна-Цайлінгера (ГХЦ) [34]:

$$\left|GHZ\right\rangle = \frac{1}{\sqrt{2}}\left(\left|000\right\rangle + \left|111\right\rangle\right). \tag{1.14}$$

Як і для двухкубітних станів Бела (1.13), в ГХЦ-стані (1.14) кожний з трьох кубітів находиться в повністю змішаному стані і не несе сам по собі ніякої інформації. Однак, якщо виміряти стан одного з кубітів у стані (1.14), то два інших зразу приймають визначені значення. Таким чином, вимір руйнує переплутаність станів виду (1.13), (1.14).

Оператор (матриця) щільності. Про квантову систему, яка описується вектором стану $\left|\Psi\right\rangle$, кажуть, що вона знаходиться в *чистому стані*. Про квантові системи, які не можуть бути описаними вектором стану, оскільки їх стани визначені не точно, кажуть, що вони знаходяться у *змішаному стані*. Квантові системи в змішаному стані описуються операторами щільності [34, 37, 38, 87].

Нехай квантова система представляє собою суміш станів ρ_i з ймовірностями p_i. Така система описується оператором щільності [61, 93]

$$\rho = \sum_i p_i \rho_i . \tag{1.15}$$

Всі постулати квантової механіки можуть бути переформульовані на мові операторів щільності. Так, наприклад, ймовірність отримання результату *m* при вимірюванні задається виразом [34].

$$p(m) = tr\left(M_m^\dagger M_m \rho\right). \tag{1.16}$$

Квантовий стан називається повністю змішаним, якщо його матриця щільності представляє собою одиничну матрицю [37, 61, 93].

За допомогою операторів щільності можна описувати і чисті квантові стани. Таке описання чистих станів еквівалентно їх описанню за допомогою векторів станів. Проте змішані квантові стани описуються тільки операторами щільності. Оператор щільності чистого квантового стану $|\Psi\rangle$ має вигляд [34]

$$\rho = |\Psi\rangle\langle\Psi|. \tag{1.17}$$

Одна з властивостей матриці щільності – $tr(\rho) = 1$ – завжди дорівнює одиниці. Інша властивість матриць щільності служить критерієм чистого стану: $tr(\rho^2) = 1$ для чистих станів, у той час як для змішаних станів $tr(\rho^2) < 1$.

Редукований оператор щільності. Такий оператор ρ_A описує статистику вимірів, які виконують на системою *A*, яка є частиною складеної квантової системи *AB*, яка описується оператором щільності ρ_{AB}. У цьому сенсі редукований оператор щільності ρ_A описує систему *A*. Математичний редукований оператор щільності задається виразом [61, 93]:

$$\rho_A = tr_B(\rho_{AB}). \tag{1.18}$$

де tr_B – операція взяття часткового сліду по станам системи *B*. Ця операція визначається наступним чином:

$$tr_B(|a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2|) \equiv |a_1\rangle\langle a_2| tr(|b_1\rangle\langle b_2|) = \langle b_2|b_1\rangle |a_1\rangle\langle a_2|. \tag{1.19}$$

де $|a_1\rangle$ і $|a_1\rangle$ – два довільних вектора стану системи *A*, $|b_1\rangle$ и $|b_1\rangle$ – системи *B*.

Ентропія фон Неймана. Ентропія квантового стану ρ, вперше виведена фон Нейманом, визначається наступною формулою:

$$S(\rho) \equiv -tr(\rho \log \rho) = -\sum_i \lambda_i \log \lambda_i, \tag{1.20}$$

де λ_i – власні значення матриці щільності ρ.

Досяжна інформація. Теорема Холево [61, 93, 149, 150]. Припустимо, що Аліса має класичне джерело інформації, яке видає символи $X = 0, \ldots, n$ з розподілом ймовірності $p_0, \ldots, p_n$. Аліса готує квантові стани ρ_X, вибираючи його з

фіксованого набору $p_0, \ldots, p_n$, і надсилає цей стан Бобу, який виконує квантове вимірювання над цим станом. Потім він намагається зробити найкраще припущення про те, як визначити *X*, основуючись на результаті свого вимірювання *Y*. Теорема Холево встановлює, що для будь-якого такого виміру Боб виконує нерівність

$$H(X:Y) \le S(\rho) - \sum_x p_x S(\rho_x), \tag{1.21}$$

де $H(X:Y)$ – взаємна інформація між *X* і результатом вимірювання *Y*; $\rho = \sum_x p_x \rho_x$. Величина у правій частині нерівності (1.21) носить назву інформації (ентропії, величини) Холево.

Теорема про заборону точного копіювання невідомого квантового стану [61]. Часто скорочено називають теоремою про заборону клонування і звучить як – неможливо точно скопіювати невідомий квантовий стан. Класичну інформацію практично завжди можна скопіювати, тобто створити копію цифрової інформації (копію файлу). В квантовому випадку теорема про заборону клонування стверджує, що неможливо побудувати квантовий прилад (клонуючу машину) так, щоб при наявності на вході стану $|\psi\rangle$ і $|\varphi\rangle$, на виході отримати дві копії вхідного стану $|\psi\rangle$, $|\psi\rangle$ або $|\varphi\rangle$, $|\varphi\rangle$. З іншого боку, якщо стани $|\psi\rangle$ і $|\varphi\rangle$ ортогональні, то теорема не забороняє їх точного копіювання. Це вирішує суперечність між теоремою про заборону клонування і можливістю точно копіювати класичну інформацію: класична інформація повинна сприйматися, як та що представляється ортогональними станами.

Теорема про заборону клонування невідомих квантових станів представляє собою один з найважливіших принципів КК [61, 93], так як зловмисник не може виготовити точну копію квантових систем, які передаються по комутаційному каналу, для того що провести вимірювання над копією, а оригінал переслати законному користувачу каналу, не проводячи над ним вимірювання. Це змушує зловмисника вимірювати стани квантових систем, що передаються (або заплутувати їх з своїми допоміжними системами), що внаслідок постулату вимірювання призводить до зміни їх станів. Такі зміни станів, що переда-

ються, можуть виявити легітимні користувачі каналу, виконуючи квантові вимірювання і обмінюючись результатами цих вимірювань по відкритому каналу зв’язку.

Нерозрізненість неортогональних квантових станів. Внаслідок викладених вище постулатів квантової механіки, неможливо виконати вимірювання, яке дозволило б точно розрізнити два стани кубіта $|\Psi_1\rangle = \alpha_1|0\rangle + \beta_1|1\rangle$ і $|\Psi_2\rangle = \alpha_2|0\rangle + \beta_2|1\rangle$, окрім випадку, коли ці стани ортогональні $(\langle\Psi_1|\Psi_2\rangle = 0)$. Це твердження справедливо не тільки для кубітів, але й для квантових систем будь-якої розмірності [8, 37, 38, 61].

Як видно із вищесказаного, протоколи КК володіють деякими властивостями, яких не мають класичні аналоги. Як зазначалось, згідно з законами квантової фізики, операції, які проводяться над квантовими системами, призводять до змін їх станів. Отже, стан квантових систем, що передаються, неминуче змінюється при намаганні зловмисника перехопити інформацію, що завжди можуть виявити легітимні користувачі. Їх подальші дії залежать від того, який саме протокол і в яких умовах реалізується. Так як КК бурхливо розвивається протягом останніх 20 років, науковцями всього світу представлено безліч різноманітних протоколів КК. У роботі [45] представлена одна з найбільш повних класифікацій методів КК, на сьогодні існують такі технології (представлені на рис. 1.1): квантові протоколи розподілу секретних ключів, квантові протоколи прямого безпечного зв'язку, квантове розділення секрету, квантовий потоковий шифр, квантовий цифровий підпис та квантова стеганографія.

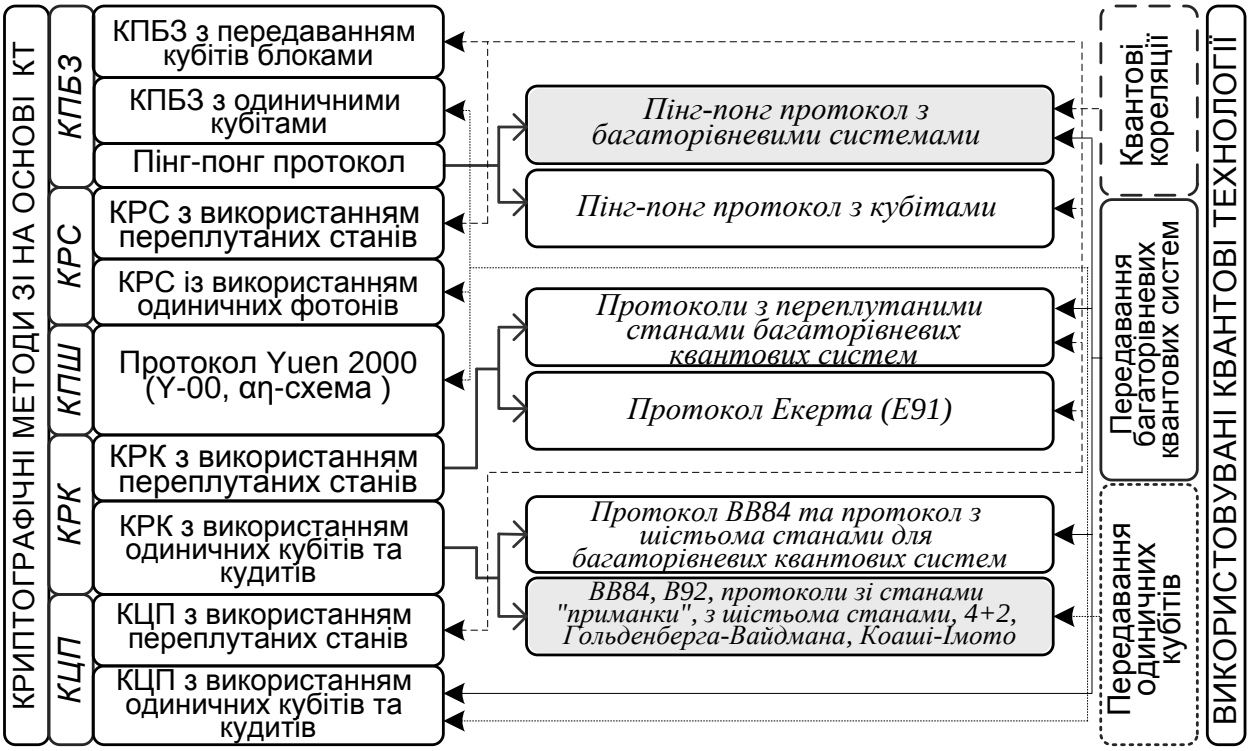

Рис. 1.1. Класифікація квантових методів ЗІ [45]

У роботах [77, 88] проведено детальний аналіз існуючих класифікацій методів КК, та виявлено, що на сьогодні хоча їх і існує декілька, проте вони або орієнтовані на узагальнення протоколів одного виду, наприклад, у роботі [51] представлена класифікація протоколів КРС. У роботі [120] проводиться класифікація квантових протоколів шифрування з відкритим ключем (quantum public-key encryption protocol) за шістьма елементами кортежу та виділяється з 64 видів три основні типи протоколів. За аналогічним методом класифікують квантові протоколи симетричного шифрування (quantum symmetric-key encryption protocols) [121]. У [146] наведена класифікація протоколів КТ, а також представлено двосторонню КТ та двосторонній КПБЗ. Часткова класифікація протоколів КРК, КРС та систематизація деяких атак на КК проводилась у [155]. Проте мало лише провести моделювання або розробити протокол, необхідно підтвердити його стійкість до різного роду атак. Однак науковці, як правило, проводять окремо класифікацію різного роду атак на протоколи КК [5, 50, 56, 181], а найчастіше досліджують конкретний протокол і конкретну атаку на нього [5-7, 56, 57, 181, 188, 193], тобто стійкість до певного класу атак не виділяють як окрему класифікаційну ознаку.

Класифікація Корченка-Васіліу-Гнатюка [45, 162] є однією з найбільш повних, однак вона є досить застарілою – не враховує протоколи КТ, КТІ тощо, та також не враховує класифікаційну ознаку – стійкість до різного роду атак. З огляду на все вище зазначене, виникає потреба побудови узагальненої класифікації сучасних квантових технологій ЗІ, якісний аналіз переваг та недоліків, з урахуванням базової ознаки – стійкість протоколів до певного виду кібератак, що дасть можливість у повній мірі оцінити рівень існуючих досягнень для їх подальшого ефективного використання.

1.2. Комерційні системи квантової криптографії

Значна частина теоретичних та практичних досліджень у галузі КК присвячена розробці та вдосконаленню протоколів КРК [38, 61, 78, 90, 102, 104, 112, 113, 119, 139, 161, 162], єдиному на сьогоднішній день напряму КК, який досягає теоретико-інформаційної (безумовної) стійкості [8] та вже пройшов шлях від теоретичних досліджень до практичної реалізації. Аналіз сучасних розробок проводився у роботах [207, 208, 141]. Існує багато наукових центрів та інститутів, що займаються дослідженням КК, так приведемо деякі з них: Harvard-MIT Center (Бостон, США), The Institute for Photonic Science (Барселона, Іспанія), Max Planck Institute of Quantum Optics (Мюнхен, Германия), Institute for Quantum Optics and Quantum Information of the Austrian Academy of Science (Інсбрук, Австрія), Vienna Center for Quantum Science and Technology (Відень, Австрія), University of Geneva (Женева, Швецарія), Centre for Quantum Technologies at the National University of Singapore (Сінгапур,), Institute for Quantum Computing (IQC) at the University of Waterloo (Ватерлоо, Канада), та багато інших. В Україні дослідження у галузі КК ведуться у Інституті фізики НАН України, Національному авіаційному університеті, НТТУУ «КПІ» та Одеській національній академії зв’язку ім. Попова.

Основними виробниками систем КК, а саме систем КРК, які вже декілька років представлені на ринку комерційними системами є [110, 116, 143, 152, 176, 178, 191]: id Quantique, Inc. (Швейцарія), MagiQ Technologies, Inc. (США),

SmartQuantum, Inc. (Франция), QuintessenceLabs, Pty Ltd (Австралия), D-Wave Systems (Канада), Toshiba Research Europe Ltd (Великобританія), QinetiQ (Великобританія), NEC (Японія). На рис. 1.2 – 1.8 представлені деякі комерційні системи КРК.

Рис. 1.2. Комерційна система Cerberis QKD, фірми id Quantique [116]

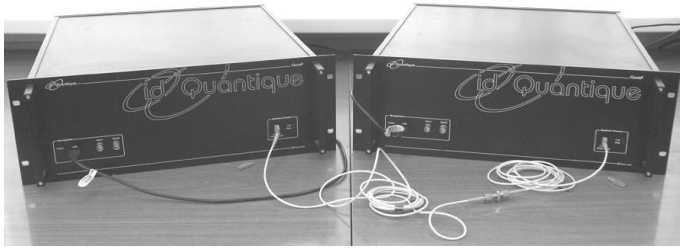

Рис. 1.3. Квантова криптосистема Clavis^{2}QKD

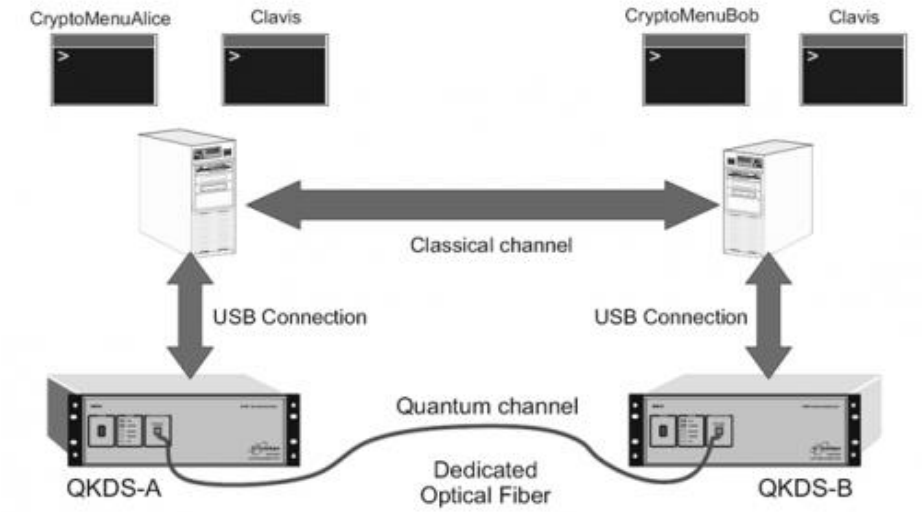

Рис. 1.4. Схема реалізації роботи криптосистеми Clavis^{2}QKD

Рис. 1.5. Квантова криптосистема, розроблена Toshiba Research Europe Ltd [178]

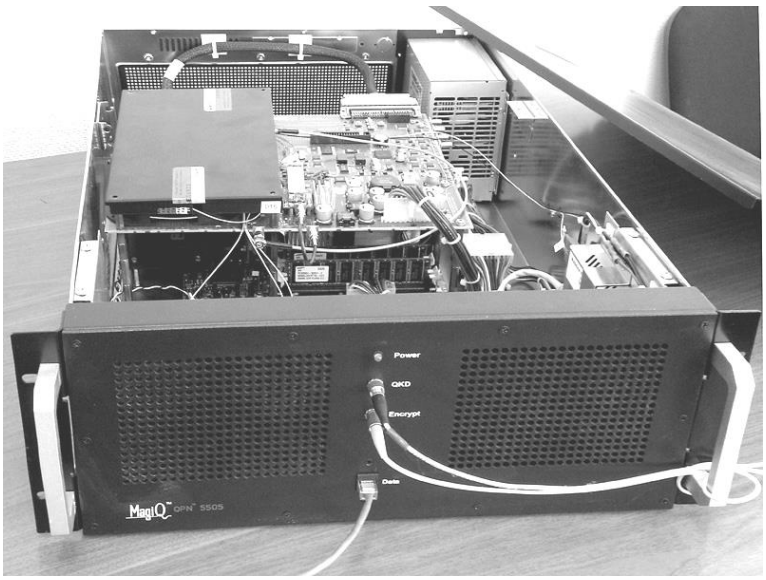

Рис. 1.6. Квантова криптосистема, розроблена MagiQ Technologies, Inc.

Рис. 1.7. Генератор квантових ключів Q-Key maker [110]

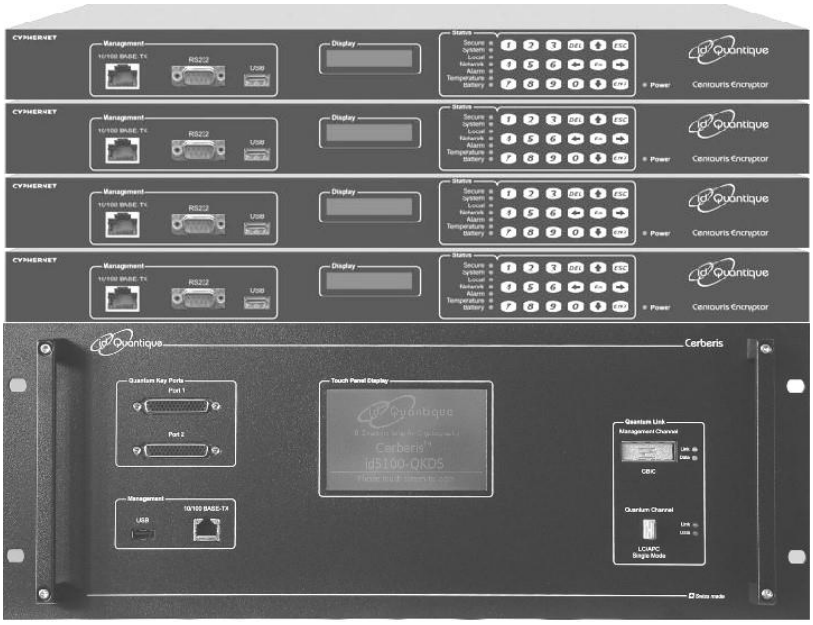

Рис. 1.8. CN8000 Multilink Encryption

Окрім того, на сьогодні системи КРК [141], впроваджені у таких банках Європи (Швейцарія): Notenstein Private Bank, Swissquoute Bank, Hyposwiss Private Bank, Global Bank. А також захист інформації засобами КРК проводиться у таких комерційних компаніях: Battelle's System Heralds (США), Bloombase Extends Enterprise (США), Colt (Великобританія), Cygate (Швеція, Фінляндія).

Проте не всі протоколи КК досягають теоретико-інформаційної стійкості як протоколи КРК, які вже отримали практичну реалізацію та, як видно, з вищезазначенного, активно впроваджуються у світі.

1.3. Підходи до підвищення ефективності протоколів квантової криптографії

Наступним за своїми досягненнями серед всіх технологій КК є напрям КПБЗ [5-13, 35, 41, 43, 63, 85, 86, 109, 119, 122, 130, 137, 138, 140, 146, 159, 165, 186, 192, 194, 205], який є найбільш близьким до отримання практичної реалізації. Назва КПБЗ зумовлена тим, що для передачі секреторної інформації, протоколами, не потрібно виконувати її попереднє шифрування, при цьому надійність буде гарантуватись непорушністю постулатів квантової механіки [8, 34, 61, 78, 79]. Методи та протоколи КК дають можливість легітимним користувачам виявити атаку пасивного перехоплення (підслуховування) в каналі зв'язку [7, 50], що при використанні традиційних методів далеко не завжди можливо. Виявлення такої атаки є критично важливим як при розподілі секретних ключів шифрування, так і при передачі відкритих текстів. Протоколи КПБЗ дозволяють виявити таку атаку та за допомогою спеціальних процедур обробки інформації, що передається, зробити інформацію, яку міг отримати зловмисник, не корисною для нього [5-9]. Таким чином, КПБЗ забезпечують високий рівень безпеки, шляхом прямої, тобто без використання шифрування, передачі відкритих повідомлень. Окрім того, протоколи КПБЗ можуть використовуватись як протоколи КРК.

Схема з відвідним каналом, є своєрідним аналогом протоколів КПБЗ в класичній криптографії. Зловмисник отримує інформацію по додатковому каналу, не маючи прямого доступу до основного каналу між легітимними користувачами [8]. Для забезпечення теоретико-інформаційної стійкості в класичній схемі криптосистеми рівень завад в відвідних каналах повинен бути більше рівня завад в основному каналі. У КПБЗ протоколах таких умов на квантовий канал не накладається. Існують також більш складні схеми протоколів з відвідним каналом, в яких використовується зворотний зв'язок [8]. В таких схемах при деяких умовах теоретико-інформаційна стійкість може бути досягнута, навіть якщо рівень завад в основному каналі більше рівня завад у відвідному.

Таким чином, протоколи КПБЗ, як в певній мірі і схеми з відвідним каналом в класичної криптографії, забезпечують можливість безпечної передачі секретної інформації відкритим каналом зв'язку без її шифрування. Проте більшість протоколів КПБЗ мають лише асимптотичну стійкість до кібератак, тому вони потребують використання додаткових процедур підвищення їх ефективності, наприклад, таких як: підвищення інформаційної місткості за рахунок використання *d*-рівневих квантових систем; використання квантового надщільного кодування; синтезу традиційних та квантових методів ЗІ та ін.

Одним з найпростіших протоколів КПБЗ є пінг-понг протокол. Перший варіант цього протоколу з використанням чотирьох максимально переплутаних ортогональних станів у системі двох кубітів станів Белла (1.13) та без використання квантового надщільного кодування був запропонований в 2002 році [109]. Основною перевагою цього протоколу є те, що він не потребує великої квантової пам'яті і тому може бути реалізований при сьогоднішньому рівні квантових технологій.

В літературі запропоновано декілька пінг-понг протоколів з використанням пар і триплетів переплутаних кубітів і кутритів, а також відповідного квантового надщільного кодування [6-10, 12, 13, 35, 65, 85, 113, 114, 119]. У роботах [109, 113] представлено загальну схему оригінального варіанту пінг-понг протоколу з повністю переплутаними парами кубітів та квантовим надщільним ко-

дуванням. Перший варіант пінг-понг протоколу (оригінал) є одним з найпростіших, проте загальні схеми всіх подальших модифікацій пінг-понг протоколів подібні.

Існують також різновиди пінг-понг протоколу з три- і чотирикубітнимі переплутаними ГХЦ станами [8, 9, 137], інформаційна місткість яких складає 3 і 4 біта на раунд протоколу відповідно, ідея була запропонована в роботі [113]. Також відомими є варіанти пінг-понг протоколи [8-10, 13, 35, 119, 137, 192], деякі з переплутаними парами і триплетами кутритів з інформаційною місткістю $\log_2 9 \approx 3{,}17$ біт/раунд і $\log_2 27 \approx 4{,}75$ біт/раунд відповідно [8], тобто що дозволяє підвищити швидкість та ефективність. Ідея використання трійкової логіки не нова не тільки для КК, а й для традиційних технологій, ще у 1840 р. Т. Фоулер побудував механічну трійкову обчислювальну машину (помножувач з 55-тритним регістром результату), одну з найбільш ранніх механічних обчислювальних машин [190]. А у 1959 р., під керівництвом Н. Брусенцова розроблена перша серійна трійкова електронна обчислювальна машина «Сетунь» [3, 53]. З 1962 р до 1964 р. Казанським заводом математичних машин було вироблено 46 машин «Сетунь». У 1970 р., Н. Брусенцов побудував в Московському державному університеті другу електрону трійковий комп’ютер «Сетунь-70». 2008 р. – Д. Коннеллі, К. Патель і А. Чавез за підтримки професора Ф. Ніко (California Polytechnic State University of San Luis Obispo, San Luis Obispo, Каліфорнія, США) побудували трьохтритову цифрову комп'ютерну систему TCA2, версія v2.0, в трирівневій (3-Level CodedTernary, 3L CT) системі трійкових логічних елементів на 1484-х інтегральних транзисторах [123]. 2013 р. на конкурсі Intel ISEF представлена робота молодого науковця щодо порівняння швидкості роботи двійкових та трійкових систем [52].

Ще одним методом підсилення стійкості є метод, оснований на використанні зворотного хешування за допомогою оборотних трійкових матриць [8, 10, 13, 85]. Недоліком описаного методу підсилення секретності є те, що для передачі всього повідомлення генерується та використовується дуже велика кількість випадкових, оборотних над полем Галуа, тритових матриць, що потребує

великих часових та ресурсних затрат (значна кількість математичних перетворень над полем GF(3)), а з огляду на те, що відомі методи генерування (генератори) орієнтовані на бінарні системи, виникає питання генерування тритових ПВП та постає питання оцінки їх якості (досліджувалось у роботах [18, 20, 21, 55]), адже існуючі методики статистичного тестування (NIST STS [98, 172], DIEHARD [162, 189], CRYPT-X [187], методика Кнута [39]) також орієнтовані на двійкові системи.

1.4. Формалізація завдання роботи

Отже, провівши аналіз можна зробити висновок, що виникала потреба побудови узагальненої класифікації сучасних квантових технологій ЗІ, якісний аналіз переваг та недоліків, з урахуванням базової ознаки – стійкість протоколів до певного виду кібератак, що дасть можливість у повній мірі оцінити рівень існуючих досягнень для їх подальшого ефективного використання. Також, проведений аналіз показав, що переважна більшість досліджень є орієнтованими на методи КРК (BB84, B92, SARG, E91, Гольденберга-Вайдмана, Коаші-Імото тощо), які дозволяють вирішити проблему розподілу ключів шифрування в умовах секретності і використовуються, як правило, у комплексі з симетричними криптографічними методами (AES, 3DES). Іншим важливим напрямком КК є використання методів КПБЗ, які дозволяють передавати інформацію відкритим каналом (без попереднього її шифрування – проблема розподілу ключів нівелюється). На сьогодні запропоновано велику кількість методів КПБЗ, що базуються на різних квантових технологіях і можуть використовуватись як для захищеного передавання інформації (за допомогою кубітів або кудитів), так і для розподілу криптографічних ключів. Науковцями всього світу проведено багато досліджень ефективності протоколів КК з передаванням кубітів (дворівневих квантових систем), але ефективність протоколів з кудитами (багаторівневими квантовими системами), які мають більшу інформаційну місткість (на один раунд протоколу зв’язку), досліджена

на цей час значно менше [19, 21, 179]. Для підвищення їх ефективності у відомих системах використовують квантове надщільне кодування, квантові і неквантові методи підсилення безпеки тощо. Проте ці методи не позбавлені недоліків, з огляду на що, науково-практичною задачею роботи визначено розробку і дослідження нових методів підвищення ефективності кутритових протоколів КК. Зокрема відомий підхід до підвищення ефективності кутритових протоколів КПБЗ, шляхом застосування зворотного хешування за допомогою оборотних трійкових матриць [8, 10]. Проте, генерування останніх потребує великих часових та ресурсних затрат підходу до підвищення стійкості методів КПБЗ. З огляду на це, розробка і дослідження нових ефективних методів забезпечення стійкості кутритових протоколів квантової криптографії до некогерентних атак, побудови тритових генераторів ПВП та оцінювання їх якості (можливості використання для криптографічних застосувань) є актуальною науково-практичною задачею, що має теоретичне і практичне значення.

1.5. Висновки до першого розділу

Таким чином, у першому розділі дисертації проведено якісний аналіз літературних джерел по темі дослідження, розглянута класифікація сучасних квантових методів і технологій захисту інформації. Також на основі проведеного аналізу стану проблеми визначено і обґрунтовано основні задачі дослідження, вирішення яких необхідне для досягнення мети, що поставлена в дисертаційній роботі. Проаналізовані властивості та особливості передавання інформації за допомогою протоколів квантової криптографії. У розділі наведено передумови виникнення нового напрямку захисту інформації, такого як квантова криптографія. Узагальнено теоретичні основи квантової криптографії та описано постулати принципової непорушності квантової механіки, які лежать в основі квантових технологій, та забезпечують їх унікальність. Проведений аналіз методів, протоколів і комерційних технологій КК, дав можливість визначитись та чітко сформулювати задачі наукового дослідження для досягнення мети.

РОЗДІЛ 2

ПІДВИЩЕННЯ СТІЙКОСТІ ПРОТОКОЛІВ КВАНТОВОЇ КРИПТОГРАФІЇ

2.1. Стійкість систем квантової криптографії до різного роду атак

Зважаючи на те, що у квантовому каналі неможливо відрізнити природні завади від тих, що створюються зловмисниками при спробі підслуховування [8, 34, 61], необхідно передбачити цей факт при проектуванні систем та протоколів КК. Фундаментальні закони квантової фізики [61] з одного боку забезпечують велику перевагу протоколів КК, а саме можливість виявлення атаки пасивного перехоплення, а з іншого – допускають можливість реалізації різного роду атак у квантових системах ЗІ.

Кібератаками у квантових системах захисту інформації називаються заходи, які застосовуються для зміни стану передаваних носіїв інформації (фотонів, і т.д.), та інших квантових систем, тобто несанкціонований доступ до передаваної інформації згідно з [56].

У роботах [2, 50, 57, 81] проведено якісний аналіз атак на протоколи КК, а праця [2] містить розширену класифікацію кібератак за ознаковим принципом. Крім того, у роботі [57] наведена класифікація атак на канали квантового розподілу ключів: виділено два класи таких атак – це атаки на кубіти та атаки, що використовують неідеальність компонентів системи.

Загальна класифікація атак за ступенем складності. Згідно [50] *атаками у квантових системах* називаються заходи, які застосовуються для підриву безпеки даних систем чи реалізації загроз базовим характеристикам безпеки (конфіденційності, цілісності, доступності) квантово-криптографічних систем шляхом використання їх уразливостей. На рис. 2.1 наведена узагальнена класифікація атак у квантових СЗІ за ступенем складності [2, 56].

При використанні легітимними користувачами ідеальних однофотоннх джерел, атаки в системах КК (за ступенем складності) можна умовно поділити на *когерентні* та *некогерентні*.

Рис. 2.1. Класифікація атак у квантових системах ЗІ [2, 56]

I. Некогерентні атаки [2, 11, 59, 90]**.**

Некогерентні (індивідуальні) атаки [2, 11, 56] бувають *непрозорими (opaque attacks)* та *напівпрозорими* (*semi-translucent attack)*. Далі детально розглянемо кожен тип атак.

– ***<u>Некогерентна непрозора атака</u>*** (також називають *атакою «перехоплення – повторної посилки кубітів», intercept-resend attack* [56]).

Основна мета: вимірювання Євою безпосередньо квантового стану носія (фотона) і подальшій повторній посилці нового фотона у стані, який отримано в результаті вимірювання. Оскільки зловмисник не пропускає квантові стани Аліси, а генерує нові і відправляє їх Бобові, то даний клас атак називається непрозорим.

– ***1. Некогерентна напівпрозора атака*** [2, 56]**.**

Основна мета: використання Євою своїх допоміжних квантових систем (проб) для переплутування (entanglement) їх з носіями, які Аліса пересилає Бобу по квантовому каналу.

Етапи реалізації: кількість інформації Єви при атаці з використанням допоміжних квантових систем (проб) на один цикл пінг–понг протоколу з n-кубітними ГХЦ-станами визначається ентропією фон Неймана [11, 56]:

$$I_0 = S(\rho) \equiv -Tr\{\rho \log_2 \rho\} = -\sum_i \lambda_i \log_2 \lambda_i , \qquad (2.1)$$

де λ_i – власні значення матриці щільності ρ системи «кубіти, що передаються, проба Єви». Після взаємодії, передаванні та допоміжні стани знаходяться у загальному переплутаному стані, потім перші передаються Бобові, а другі зберігаються у квантовій пам’яті у Єви. Після закінчення відкритого обміну інформацією між Алісою та Бобом на етапі просіювання ключа, зокрема об'явлення базисів, в яких Боб вимірював фотони Аліси, Єва визначає послідовність базисів, яку необхідно використати для вимірювання станів її проб, щоб отримати якомога більше інформації про ключ. Стани фотонів Аліси змінюються після переплутування з пробами Єви, проте рівень помилок при даній атаці значно нижчий, ніж при непрозорій атаці. Проте для реалізації подібної атаки Єві необхідно мати квантову пам'ять великого об’єму для зберігання проб до ооголошення базисів Бобом, та складне обладнання для переплутування проб з фотонами Аліси.

Уразливі протоколи: напівпрозорі атаки є також одним з основних видів атак на КПБЗ. У роботах [8, 11] розглядається атака з використанням квантових проб на пінг-понг протокол КПБЗ з ГХЦ-триплетами [6], а також обчислено повну ймовірність виявлення атаки зловмисника в залежності від кількості отриманої ним інформації для трьох варіантів пінг-понг протоколу. Доведено, що інформаційна місткість та стійкість різних варіантів даного протоколу є обернено пропорційними величинами [11]. При цьому, якщо Єва вибере атаку, яка дає повну інформацію про передані біти, протокол з парами Белла (1.13) і

надщільним кодуванням та протокол з ГХЦ-триплетами мають практично однаковий рівень стійкості до некогерентної атаки.

II. ***Когерентні атаки*** [2, 59, 90, 139]**.** При когерентних атаках Єва може будь-яким (унітарним) способом переплутати пробу будь-якого розміру з групою фотонів, що передаються. Дану атаку поділяють на два види – *колективна атака (collective attack)* [2, 90, 139] та *об'єднана атака (joint attack)* [2, 8, 139]. Враховуючи стан розвитку сучасної техніки, реалізація когерентних атак на даний момент практично неможлива (на відміну від некогерентних атак), так як на сьогодні не існує необхідної квантової пам'яті великого обсягу, проте поява багатокубітного квантового комп'ютера (many-qubit quantum computer) D-Wave 2X може змінити ситуацію. Далі детально розглянемо кожен тип атак.

– ***2. Когерентна колективна атака*** *(collective attack)* [2, 90, 139].

Основна мета: кожний фотон Аліси індивідуально переплутується з окремою пробою (початкова стадія схожа з напівпрозорою атакою).

Етапи реалізації: Єва отримує проби в таких же станах, як і при напівпрозорій атаці. Але після закінчення відкритого обміну інформацією між Алісою та Бобом, Єва виконує вимірювання відразу на всіх пробах, як на єдиній квантовій системі [56].

– ***3. Когерентна об'єднана атака*** *(joint attack)* [2, 8, 139].

Основна мета: переплутування усієї послідовності фотонів з використанням єдиної квантової проби.

Етапи реалізації: Єва використовує єдину квантову пробу (з гільбертового простору станів більшої розмірності) для переплутування з усією послідовністю фотонів, що Аліса передає Бобові. Ця атака є також і найбільш складною з технічної точки зору.

III. Атаки, зумовлені недосконалістю протоколів. Недосконалість протоколів є серйозним чинником для реалізації атак в КК. Найвідомішими атаками цього класу є *атака «людина посередині» (man-in-the-middle attack)* та *атака «відмова в обслуговуванні» (denial of service attack).*

– ***4. Атаки «людина посередині»*** [12, 45, 56, 114]**.**

Основна мета: мати можливість замінювати усі повідомлення, що передаються класичним каналом зв'язку.

Етапи реалізації: Єва має повністю контролювати класичний канал зв'язку між Алісою та Бобом.

Уразливі протоколи: усі існуючі протоколи КРК і ЛПБЗ вразливі до даної атаки [2, 8, 88].

Захист від такої атаки є загальновідомим – автентифікація легітимних користувачів у класичному каналі.

– ***5. Атаки «відмова в обслуговуванні»*** [8, 56].

Основна мета: не отримати інформацію, а зруйнувати квантовий канал між Алісою та Бобом, вивести з ладу їх обладнання.

Етапи реалізації: Єва вимірює стан кубіта на зворотному шляху від Аліси до Боба (в режимі передачі повідомлення) – цим самим порушуючи взаємну кореляцію (mutual correlation) кубітів Аліси та Боба. У результаті Єва не отримає ніякої корисної інформації, проте зруйнує квантовий канал між Алісою та Бобом. У випадку ГХЦ-триплетів Єва може також вимірювати стани одного чи двох кубітів і порушувати таким чином переплутаність стану триплету [6].

Уразливі протоколи: До цієї атаки вразливі практично всі протоколи КК.

IV. ***Атаки, зумовлені недосконалістю обладнання.*** У класичній криптографії атаки, зумовлені недосконалістю обладнання, називають також *атаками, що використовують витік інформації по побічних каналах (side-channel attacks)*. Атаки такого типу можливі також і в КК.

– ***6. Атаки типу «Троянський кінь»*** *(Trojan Horse attack)* [45, 56, 12, 129, 139, 156, 165]**.**

Основна мета: Єва використовує світло іншої довжини хвилі в квантовому каналі між Алісою та Бобом для отримання відбитих імпульсів.

Етапи реалізації: Єва посилає світлові імпульси у квантовий канал, що з'єднує апаратуру Аліси та Боба, і потім аналізує відбите світло. Таким способом у принципі можливо виявити, який лазер або який датчик тільки що спрацював,

або параметри настройки модуляторів поляризації та фази. Така атака не може бути просто відвернена використанням засувки, тому що Аліса та Боб повинні залишити «двері відчиненими» для своїх фотонів. Але Аліса й Боб могли б виявити додаткові фотони Єви, так як при такій атаці відбувається збільшення енергії імпульсів. Тому Єва повинна використовувати світло іншої довжини хвилі, ніж використовують Аліса та Боб, а саме такої довжини хвилі, до якої датчики Аліси й Боба є нечутливими [128].

Уразливі протоколи: До атак даного типу уразливі так звані двосторонні (two-way) протоколи КРК та КПБЗ (наприклад, пінг-понг протокол), а також деякі протоколи КТ та КТІ.

– ***7. Атаки типу «Троянський кінь з затримкою фотона»*** [129].

Основна мета: отримати повну інформацію про кодувальну операцію Аліси, таким чином отримуючи необхідні дані та передвану інформацію, виконавши відповідне вимірювання.

Етапи реалізації: Єва перехоплює сигнал, переданий від Боба до Аліси, і потім вставляє додатковий фотон у сигнал з часом затримки, коротшим ніж часове вікно датчика [129]. Таким чином, Аліса не може виявити цей додатковий фотон, оскільки він не спричинює спрацювання її датчика. Після кодувальної операції, яку виконує Аліса, Єва перехоплює сигнал знову й відокремлює додатковий фотон.

Протидія атакам типу «Троянський кінь». Для протидії атакаам «Троянський кінь» Аліса та Боб повинні встановити фільтр сигналів з іншими довжинами хвиль на вході свого обладнання [2, 56]. На практиці Аліса й Боб повинні експлуатувати фільтр довжини хвилі для фільтрування фонового світла, особливо коли у якості квантового каналу використовується вільний простір (бездротовий оптичний канал). Таким чином, немає проблеми для легітимних користувачів запобігти такій атаці [2]. Для атаки «троянського коня з затримкою фотона» Аліса повинна використовувати світлодільник 50/50 [2], щоб розділити кожний сигнал на дві частини й провести вимірювання їх станів у двох вимірювальних базисах. Якщо є тільки один фотон в оригінальному сигналі, то спрацює

лише один з датчиків, інакше – спрацюють одразу два. Атаки типу «Троянський кінь» можуть бути відвернені технічними засобами. Але той факт, що цей клас атак існує, ілюструє, що безпека квантової криптографії не може гарантуватися тільки принципами квантової механіки, але обов'язково покладається також на технічні засоби [46].

– ***8. Атака поділу числа фотонів (****photon number splitting attack – PNS attack)* [56, 115, 132, 162, 167, 171].

Основна мета: зняти інформацію у звичайних оптичних телекомунікаційних системах за допомогою розділення пучка фотонів. Однак у протоколах КК передача може відбуватися за допомогою одиночних фотонів і, в такому випадку, Єва не може відвести частину сигналу. На практиці в системах КРК використовують слабкі когерентні імпульси, випромінювані лазерними світлодіодами [56]. Число фотонів в імпульсі визначається розподілом Пуассона, тобто частина переданих імпульсів містить два й більше фотони. Таким чином, на реальні системи КРК, які використовують протокол BB84, стає можливим атака поділу числа фотонів [171].

Етапи реалізації: 1. Єва виконує квантове не руйнуюче вимірювання числа фотонів в імпульсі, не впливаючи при цьому на їхню поляризацію, для кожного імпульсу, надісланого Алісою. Дуже складне у виконанні вимірювання але на теперішній час це технічно можливе [99]. Варіант проведення атаки а) якщо в імпульсі більше одного фотона, Єва відводить один, дозволяючи іншим безперешкодно потрапити до Боба. 2. Єва виконує переплутування перехопленого фотона зі своєї пробою і очікує, коли після завершення передавання легітимні користувачі оголосять використані базиси. 3. Виконуючи потім вимірювання стану проби, Єва одержує точне значення біта, що передається, не вносячи при цьому ніяких помилок у просіяний ключ, тобто атака Єви залишається невиявленою. Варіант проведення атаки б) Якщо ж імпульс несе один фотон, то стратегії Єви можуть бути різними. Наприклад, вона може просто пропускати всі однофотонні імпульси, що дозволить їй залишитися невиявленою. Однак при малому середньому числі фотонів в імпульсі (на практиці обладнання налаго-

джають так, щоб це число було порядку 0,1) кількість багатофотонних імпульсів буде невеликою, і це не дозволить Єві отримати будь-яку суттєву інформацію про ключ. Інша стратегія Єви може бути такою: вона виконує некогерентну атаку на однофотонні імпульси, тоді вона вносить помилки в просіяний ключ, кількість яких буде залежати як від типу атаки, так і від частки однофотонних імпульсів при передачі ключа. Ще одна стратегія Єви – блокування частини однофотонних імпульсів – у результаті Боб отримає порожній імпульс, його датчик не зареєструє фотон. Таким блокуванням частки однофотонних імпульсів Єва збільшує частку багатофотонних імпульсів, що дозволяє їй збільшити інформацію про ключ при тому ж рівні внесених у просіяний ключ помилок. Оскільки чутливість сучасних датчиків, які використовуються в комерційних системах КРК, невелика, і вони реєструють в середньому лише 20–30% одиночних фотонів, а крім цього також відбуваються втрати фотонів в каналі, то Єва теоретично може таким чином приховати свою атаку [56]. Але Боб, знаючи ймовірність одержати порожній імпульс при наявному обладнанні, може виявити значне перевищення кількості порожніх імпульсів над очікуваним. Відзначимо, що Боб може також не тільки визначати кількість порожніх імпульсів, але й контролювати всю статистику одержуваних ним сигналів, виконуючи неруйнуюче вимірювання числа фотонів у імпульсі. У цьому випадку Єва змушена буде відводити фотон тільки у невеликої частини багатофотонних імпульсів, а інші пропускати, не одержуючи ніякої інформації.

Уразливі протоколи: перважно протоколи КРК.

Протидія атакам: розроблені вдосконалені протоколи КРК, наприклад такі як КРК протоколи зі станами приманки (decoy states protocols) [174, 180, 196] та протокол SARG04 [161, 181].

– ***9. Атака поділу пучка фотонів*** *(photon beam splitting attack – PBS attack)* [115, 132, 162].

Основна мета: спрощена PNS атака, оскільки має простішу реалізацію.

Етапи реалізації: Єва контролює інше вихідне плече світлодільника й отримує повне значення бітів просіяного ключа (через відстрочене вимірювання), якщо багатофотонний сигнал розділений таким чином, що Боб та Єва обоє одержують хоча б один фотон сигналу. Так, у роботі [56] описано метод адаптивної абсорбції, що дозволяє вилучити точно один фотон з моди. Потужність атаки, що отримала назву *умовної атаки поділу числа фотонів*, наближається до потужності PNS-атаки [132].

Уразливі протоколи: протоколи КРК, наприклад BB84.

Протидія атакам: Один з методів захисту від PBS-атаки, як і від PNS-атаки, – це контролювання Бобом усієї статистики одержуваних сигналів [56, 162], але для цього необхідно виконувати неруйнуюче вимірювання числа фотонів в імпульсі, що є дуже складним з технологічної точки зору. Тому, більш практичним на теперішній час є використання вдосконаленних протоколів КРК, наприклад зі станами приманки (decoy states protocols) [174, 180, 196] та протокол SARG04 [161, 181].

– ***10. Атака заміни існуючого квантового каналу на кращий.***

Основна мета: удосконалення PNS- та PBS-атак.

Етапи реалізації: Єва таємно замінює квантовий канал зі втратами між Алісою та Бобом на ідеальний канал без втрат (або на канал зі значно меншими втратами) [56]. У такому випадку Єва зможе блокувати певну частину однофотонних імпульсів, видаючи таки втрати за природні – тобто Боб отримає приблизно таку ж кількість пустих імпульсів, як до заміни каналу. Для початкового каналу з великими втратами Єва матиме можливість отримати майже весь ключ і залишиться непоміченою. Крім того, якщо рівень втрат у початковому каналі дуже значний, то Єва при заміні його на значно кращий зможе зберегти не лише очікувану Бобом частку пустих імпульсів, а й усю статистику числа фотонів у імпульсі. Відзначимо, що атаку заміни існуючого квантового каналу на кращий практично дуже важко здійснити. У будь-якому випадку для захисту від такого типу атаки Аліса та Боб мають використовувати квантовий канал

обмеженої довжини так, щоб його коефіцієнт передачі залишався достатньо високим [50, 162].

Деякі інші атаки з використанням витоку інформації побічними каналами.

У роботах [2, 56] розглянута атака, при якій Єва вимірює просторові, спектральні або часові характеристики імпульсів, що передаються бездротовим оптичним каналом. Описані експерименти з протоколом BB84 показують, що найбільшу інформацію про передані біти ключа – $6{,}6\times10^{-3}$ біт/імпульс – Єва може отримати при вимірюванні спектральних характеристик. Але ця величина є достатньо малою і, таким чином, цю атаку не можна вважати потужною.

Інша атака, пов'язана з часовою незбалансованістю детектора, розглянута у праці [2] *(timing channel attack)*. Дана атака, на відміну від попередніх, дозволяє Єві отримати значну частину секретного ключа.

Взагалі, теоретичні аспекти безпеки КК є на теперішній час дуже активною областю досліджень, але значно менше досліджень поки присвячено ретельному дослідженню практичних систем. Однак останнім часом спостерігається зростаючий інтерес до аналізу атак з використанням витоку інформації через побічні канали, що є результатом фізичної реалізації принципів квантової криптографії в практичних системах.

2.2. Узагальнена класифікація протоколів квантової криптографії

До методів квантової криптографії та зв'язку відносяться [45, 77, 88]: КРК, КПБЗ, КРС, квантовий потоковий шифр (КПШ), квантовий цифровий підпис (КЦП), квантова стеганографія (КС), КТ та КТІ.

Квантовий розподіл ключів (*quantum key distribution***)** – найбільш розвинутий та досліджений напрямок КК, який у сучасних дослідженнях [38, 61, 78, 90, 102, 104, 112, 113, 119, 139, 161, 162] прийнято розділяти на:

– *DV-QKD (discrete-variable)* – КРК з дискретними змінними – як носій інформації використовується фаза фотонів / поляризація, детектором є лічильники фотонів, діапазон передачі – 100 км, в обов'язкові компоненти має вхо-

дити активне охолоджуюче обладнання. Головна перевага протоколів в тому, що за відсутності помилок, Аліса і Боб (легітимні користувачі) відразу розділяють ідеальний секретний ключ. Проте головними і суттєвими недоліками є відсутність джерел одиночних фотонів і низька ефективність їх детекторів, що спричиняє пошуки інших варіантів реалізації протоколів КРК. Протоколи DV-QKD можуть бути реалізовані з декількома джерелами, але вимагають використання методів обрахунку кідькості фотонів (*photon-counting*). До них відносять такі протоколи [185]: BB84, SARG04, The six-state protocol, Singapore protocol, B92, протоколи типу GV, модифікаціями якого є протоколи Koashi-Imoto та Guo-Shi, а також типу N09, до якого відносять оригінальний протокол N09 та Sun-Wen protocol.

– *CV-QKD (continuous-variable)* – КРК з безперервними змінними – як носій інформації використовуються амплітудно-фазове поле, детектор – когерентний, діапазон передачі – 25 км, компоненти використовуються стандартні. Лічильники фотонів замінені на стандартні p-i-n фотодіоди, які є більш швидкими та більш ефективними. Крім того вони використовують когерентні методи виявлення (*coherent detection techniques*), що широко використовуються в класичних оптичних комунікаціях. Згідно [118, 125] вони поділяються на P&M (*Prepare-and-Measure*) та E-B (*Entanglement-base*).

– *DFS-QKD (differential phase shift)* [154]. На відміну від DV-QKD та CV-QKD протоколів замість того, щоб бути закодованими в кубіти, ключова інформація кодується в фазу послідовних імпульсів слабкого когерентного світла. У протоколі DPS, Аліса кодує логічні біти у фази імпульсів. Якщо фаза модулюється «0», то Аліса посилає логічний нуль, а якщо фаза між двома імпульсами π, то вона кодує логічну одиницю. Якщо відносна фаза між двома імпульсами дорівнює «0», то Боб виявить «0», і аналогічно, якщо фаза між двома імпульсами π, то він отримає логічну «1».

У 1984 році був запропонований Ч. Беннетом та Ж. Брассаром історично перший протокол КРК – *BB84* [104], основними задачами були генерація та розподіл ключів шифрування між двома абонентами, з’єднаними як квантовим

так і класичним каналами зв'язку. У протоколі BB84 використовуються 4 поляризовані стани фотонів (0°, 45°, 90°, 135°), які передаються квантовим каналом зв'язку. Відкритий класичний канал зв'язку (не повинен бути конфіденційним, тільки аутентифікованим) використовувався для пошуку та виправлення помилок. Для виявлення факту дій зловмисника використовується процедура контролю помилок, а для забезпечення безумовної стійкості використовується класична процедура підсилення секретності (*privacy amplification*) [45].

«Six states» протокол [112] аналогічних протоколу BB84 передбачає використання чотирьох станів і додатково вводяться ще два можливих напрямки поляризації – правоциркулярний та лівоциркулярний. Такі зміни з одного боку зменшують кількість інформації, що може бути отримана зловмисником, а з іншого боку ефективність протоколу також зменшується (до 33%). Також запропоновано узагальнення протоколу з шістьома станами на багаторівневі квантові системи. Даний протокол має дещо більшу інформаційну місткість та значно більшу стійкість до атаки «перехоплення – повторної посилки» кудитів [161].

Протокол 4+2 [151] є перехідним між BB84 та B92. У ньому використовуються чотири квантових стани для кодування: «0» та «1» у двох базисах. Стани в кожному базисі вибираються неортогональними, крім того, стани в різних базисах також мають бути попарно неортогональними. Для протоколу характерна менша кількість помилок в порівнянні з протоколом BB84 для кубітів і менша кількість корисної інформації, що може бути отримати зловмисником, але одночасно відносна ефективність протоколу також зменшується [162].

У протоколі *GV* [142] кодування «0» та «1» виконується за допомогою двох ортогональних станів. Кожен з цих двох станів є суперпозицією двох локалізованих нормалізованих хвильових пакетів. Для захисту проти атаки «перехоплення – повторної посилки» використовується випадковий час відправлення пакетів. Модифікований варіант протоколу Гольденберга-Вайдмана – це протокол *Koashi-Imoto* [160], удосконалений тим, що замість випадкового часу відправлення пакетів використовується асиметризація ін-

терферометра, тобто світло розбивається у нерівних пропорціях між довгим і коротким плечами інтерферометра [161].

Протокол *B92* [103] з використанням потужних імпульсів, але зловмисник може одержати більше інформації про ключ для заданого рівня створюваних їм помилок, ніж у протоколі BB84, тобто стійкість протоколу B92 нижче стійкості протоколу BB84. Ефективність протоколу становить 25% [161].

Протокол E91 [134], відноситься до КРК з використанням переплутаних станів. Під час передавання інформації за протоколом E91 перехоплення одного із фотонів пари не дає зловмиснику ніякої корисної інформації. Крім того, запропоновано узагальнення схеми Екерта на тривимірні та багатовимірні квантові системи, що значно збільшує інформаційну місткість протоколу [162].

Протоколи зі станами «приманки» (decoy states protocols) є удосконаленим варіантом протоколу BB84, у якому відправник, шляхом заміни підмножини імпульсів, вводить так звані приманки [196]. Даному типу протоколів характерний більш високий рівень безпеки, ніж у BB84. Крім того, такі протоколи відзначаються стійкістю проти PNS атаки. До явних переваг протоколів зі станами «приманки» також можна віднести і збільшення довжини каналу за рахунок лінійної залежності від втрат у каналі. Проте, без попередньої аутентифікації користувачів на таких протоколах не можливо побудувати завершене повноцінне рішення проблеми розподілення криптографічних ключів.

SARG04 protocol [161, 181] на рівні квантової частини протокол еквівалентний BB84, проте замість джерел одиночних фотонів використовуються послаблені лазерні імпульси. Стійкий до PNS атаки.

COW (coherent one-way) protocol [139] – це новий протокол для практичної квантової криптографії, розроблений Н. Жізаном (N. Gisin) та ін. в 2004 р. У протоколі Аліса (передавач) посилає пару імпульсів, один порожній і один непорожній (містить середнє число фотонів 0,5). Біти кодуються в парі імпульсів, з значенням біту визначають по положенню непорожнього імпульсу: перше положення «0» і друге положення «1». Боб, приймач, використовує детектор, щоб розрізняти імпульси. Аліса і Боб також перевіряють узгодженість імпульсів.

Боб випадковим чином вибирає невелику частину імпульсів, що не використовується як дані, щоб відправити на інтерферометр, який вимірює когерентність між суміжними кубітами. У зв'язку з цим заходом безпеки, зловмисник не може виконати PNS-атаку, так як видалення або блокування фотонів, неможливе без порушення системи. На відміну від інших протоколів інтерферометр використовується тільки для оцінки інформації на когерентність і не може призвести до помилок на ключі. Протокол не використовує посимвольний тип кодування (як BB84, B92, SARG04).

KMB09 [158] (розроблений Khan M.M., Murphy M., Beige A. у 2009 р.) – протокол у якому Аліса і Боб використовують два взаємно неупереджені базиси – один із них кодує «0», а інший кодує «1». Безпека схеми обумовлено швидкістю передачі мінімального індексу помилки (ITER) і квантового рівня помилок, що вноситься зловмисником. ITER значно збільшується для більш високих розмірностей фотонних станів. Це дозволяє мати більше шуму в лінії передачі, тим самим збільшуючи можливу відстань між Алісою та Бобом без необхідності проміжних вузлів.

Квантовий прямий безпечний зв'язок [5-13, 35, 41, 43, 63, 85, 86, 109, 119, 122, 130, 137, 138, 140, 146, 159, 165, 186, 192, 194, 205] характерною особливістю даного методу є відсутність криптографічних перетворень, відповідно відсутня і проблема розподілу ключів шифрування. Протоколи КПБЗ можна поділити на такі типи: пінг-понг (PP) протокол (різні його варіанти) [9, 109, 45, 161, 162] протоколи з передаванням переплутаних кубітів блоками [36], протоколи з одиничними кубітами та протоколи з групами переплутаних кудитів.

Більшість запропонованих до теперішнього часу протоколів КПБЗ потребують передачі кубітів блоками [8, 122, 194]. Це дозволяє виявити прослуховування квантового каналу до початку передачі самого повідомлення й таким способом гарантувати безпеку передачі – якщо прослуховування виявлене до передачі повідомлення, то легітимні сторони переривають сеанс і ніяка інформація не витікає до зловмисника. Але для зберігання таких блоків кубітів необхідна квантова пам'ять великого об'єму [8]. Технологія квантової пам'яті

активно розробляється, але поки ще далека від масового застосування в стандартному телекомунікаційному устаткуванні. Тому з погляду технічної реалізації перевагу мають протоколи, у яких передача здійснюється одиничними кубітами або невеликими їхніми групами (за один цикл протоколу). Таких протоколів запропоновано небагато, і вони мають тільки асимптотичну безпеку [2, 5-8], тобто атака буде виявлена з високою ймовірністю, але до цього зловмисник зможе одержати деяку частину повідомлення. Отже, виникає проблема підсилення безпеки таких протоколів, тобто створення таких методів попередньої обробки передаваної інформації, які зроблять перехоплену зловмисником інформацію для нього некорисною. Протоколи КПБЗ можна поділити на чотири типи: пінг-понг протокол (різні його варіанти) [9, 109, 113], протоколи з передаванням переплутаних кубітів блоками [122, 194], протоколи з одиничними кубітами та протоколи з групами переплутаних кудитів [194]. Однак, пінг-понг протоколи, які потребують квантової пам'яті тільки для зберігання кількох кубітів (або багаторівневих квантових систем) протягом одного раунду протоколу, можуть бути реалізовані при сучасному рівні технологій. Найбільш досліджені протоколи КПБЗ: Оригінальний варіант пінг-понг протоколу з парами переплутаних кубітів, Пінг-понг протокол з парами переплутаних кубітів з використанням квантового надщільного кодування, Пінг-понг протокол з трикубітними ГХЦ-станами

Квантове розділення секрету. Переважна частина квантових протоколів розділення секрету (КПРС) [51, 128, 164, 165, 202, 161, 162]] використовує властивості переплутаних квантових станів [61, 87]. У роботах [45, 51, 161, 162] наведено детальний огляд сучасного стану протоколів КРС та проведено їх класифікацію.

Квантовий потоковий шифр [147, 148, 161, 162] передбачає шифрування даних подібно до класичних потокових шифрів, але із застосуванням квантового шумового ефекту [202] і може використовуватись в оптичних комунікаційних мережах. КПШ базується на протоколі *Yuen 2000 (Y-00)* [45, 168, 203]. Вихідними даними передавача у даній схемі є послідовність когерентних

станів, що переносить інформацію про дані чи ключ. Теоретико-інформаційна стійкість протоколу Y-00 забезпечується рандомізацією, що базується на квантовому шумі, а також на додаткових математичних (обчислювальних) схемах. Ще однією перевагою КПШ є більша захищеність порівняно із звичайними потоковими шифрами завдяки квантовому шумовому ефекту і неможливості клонування квантових станів. Що стосується недоліків КПШ, то варто відмітити, перш за все, складність практичної реалізації системи [147].

Квантовою телепортацією [146, 155, 208] називається передача квантового стану на відстань за допомогою роз'єднаної в просторі заплутаної пари і класичного каналу зв'язку, при якій стан руйнується в точці відправлення при проведенні вимірювання, після чого відтворюється в точці прийому**.** При цьому обов'язковою є передача інформації між джерелом і приймачем класичним, неквантовим каналом, яка може здійснюватися не швидше, ніж зі швидкістю світла. Протоколи КТ розділяють на два класи: *імовірнісні* та *детерміновані*. Експерименти та протоколи групи науковців під керівництвом Цайлінгера та Де-Мартіні – відносять до імовірнісних (для таких протоколів не важливо скільки фотонів загубиться при пересиланні, вони більш придатні для передачі на великі відстані). Детальне дослідження протоколів проведено у [146].

Квантова теорія ігор [133, 136, 170, 209]**.** Одним з напрямків у дослідженнях КК є вивчення різних стратегій передачі повідомлення одержувачу, для цього використовуються основи теорії ігор з елементами квантової фізики. У деяких випадках дії відправника і зловмисника розглядаються як асиметрична квантова гра двох гравців [133]. У КТІ суперпозиція використовується для моделювання невизначеності, коли неможливо знати, якою стратегією в цей момент часу скористається гравець. У класичному математичному апараті теорії ймовірності такої можливості немає, там враховується лише ймовірність, з якою гравець може вибрати ту чи іншу стратегію. А використання суперпозиції для моделювання цієї невизначеності дозволить вирішувати завдання, які стандартна імовірнісна концепція описати не може. Загальна схема квантової гри наведена у [136] та полягає у наступному: побудова квантової гри по-

чинається з вибору початкового стану і визначення можливої заплутаності системи двох гравців. Далі шукають можливі квантові рішення (стратегії) гравців різних типів. Після того, як обидва гравці вибрали свою індивідуальну квантову стратегію, вводиться оператор розчеплення (розплутування) для підготовки вимірювання стану гравців. Оператори заплутування і розплутування залежать від додаткового параметра, який вимірює ступінь заплутаності системи. Очікуваний виграш у квантовій версії загальної гри двох гравців залежить від матриці виграшів і спільної ймовірності спостерігати чотири вимірюваних величини, які і є результатами гри. Квантовий стан заплутаності двох гравців зовсім не означає, що заплуталися самі гравці (або їх думки). Процес квантової декогеренції забороняє такі макроскопічні заплутані системи, створені з мікроскопічних квантових частинок [61].

До КТІ [31] входять такі протоколи як квантове вручення біту (*quantum bit commitment*), квантове підкидання монети (*quantum coin tossing*) та квантова рулетка (*quantum gambling*). Поняття класичного вручення біту (математична версія відправки заклеєного конверту) є загальним примітивом для розробки секретних криптографічних протоколів. Дані, що відправляються від Аліси до Боба перебувають у стані замкнутості (локінгу) і можуть бути розшифровані тільки при вручені Бобу ключа від Аліси. *Квантове вручення біту* [170], як і квантові протоколи розподілу ключів, забезпечує теоретико-інформаційну (безумовну) стійкість. Проте, спираючись лише на властивості квантового каналу, експериментальне квантове вручення бітів неможливе. Ця проблема була вирішена науковцями у 2013 р. шляхом поєднання квантової фізики і теорії відносності, при цьому відбулася передача безумовно стійкого повідомлення (між Женевою та Сінгапуром за 50 мс) [88].

Квантове підкидання монети [170] використовує квантові монети, які, на відміну від звичайних, можуть перебувати у нескінченній кількості станів. Також, можливі деякі змішані стани, які пояснюються явищем суперпозиції у квантовій механіці [34, 61]. Використання змішаної позиції гарантує одній зі сторін постійний виграш. Якщо Аліса використовує змішаний стан, то у будь-

якому випадку, не залежно від дій Боба з монетою, і за умови попередньої домовленості про кінцеве число ходів, вона буде вигравати, оскільки у кінці гри зможе перевести свою монету зі змішаного стану в чистий (абсолютний).

Квантова рулетка [31], завдяки використанню непорушних постулатів квантової механіки, унеможливлює шахрайство, яке можливе у класичній версії гри «вибір виграшної коробки». Наприклад, Аліса приховує м'яч у будь-якій коробці, а Боб відгадує його місцезнаходження. Якщо грали віддалено, Аліса може легко збрехати про виграш. Або, якщо коробки були з Бобом, він міг обдурити, стверджуючи, що він знайшов м'яч. Використовуючи квантову рулетку шахрайство неможливе з огляду на те, що опоненти можуть завжди з'ясувати, який вибір зробив інший, коли гра закінчена (завдяки суперпозиції своїх дій). Хоча зазначені протоколи є лише примітивами безпеки, проте на їх основі можуть бути побудовані більш складні протоколи, здатні повністю змінити уявлення про безпеку в інформаційно-комунікаційних системах. Окрім того, також проведено багато досліджень та багато класичних ігор переведено у квантовий простір, наприклад, «prisoners dilemma», «the battle of the sexes», «the Monty Hall problem», «rock-scissors-paper», «quantum tic-tac-toe» [136, 163].

З урахуванням зазначених методів квантової криптографії та зв’язку, а також нової базової ознаки (стійкість до певного типу кібератак), узагальнена класифікація матиме такий вигляд представлений на рис. 2.2. Точками на перетині ліній позначено уразливість до певного виду атаки. Кібератаки, позначені:

C – coherent attack – когерентні атаки;

NC – non-coherent attack – некогерентні атаки;

MiM – man-in-the-middle – атака «людина посередині»;

DoS – denial of servise attack – атака «відмова в обслуговувані»;

TC – timing channel attack – часова незбалансованість детектора;

FS – заміна існуючого квантового каналу на кращий;

PBS – photon beam splitting attack – атака поділу пучка фотонів;

PNS – photon number splitting attack – атаки поділу числа фотонів;

TH – Trojan Horse attack –атака типу «Троянський кінь».

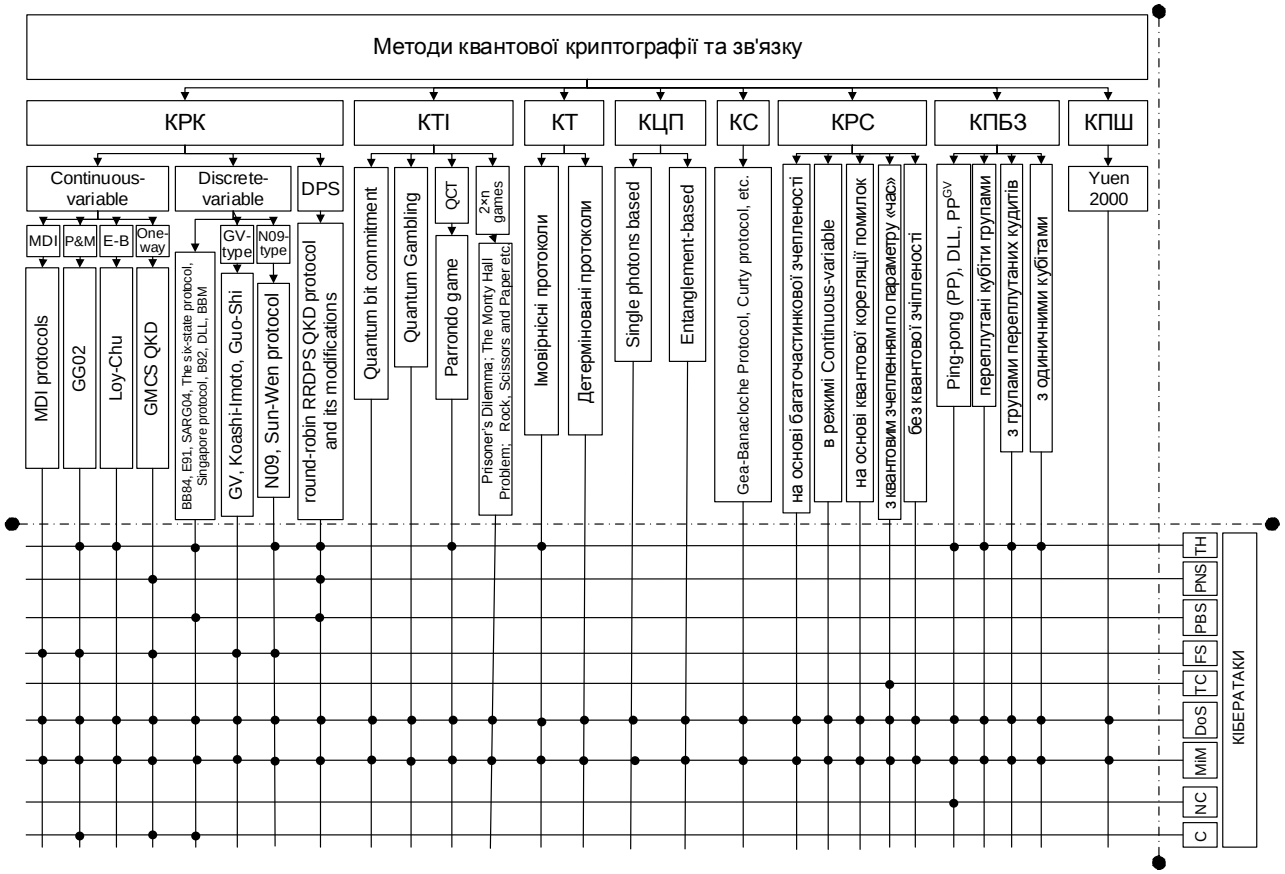

Рис. 2.2. Узагальнена класифікація методів квантової криптографії та зв’язку

2.3. Асимптотична стійкість протоколів квантового прямого безпечного зв’язку

Оскільки більшість протоколів КПБЗ мають лише асимптотичну стійкість до кібератак [8], адже в протоколах КПБЗ кожний біт є секретною інформацію і не повинен потрапити до порушника, тому вони потребують використання додаткових процедур підвищення їх ефективності.

Спершу, розглянемо оригінальний варіант пінг-понг протоколу з парами переплутаних кубітів використовує два з чотирьох повністю переплутаних станів Белла: $\left|\Psi^{+}\right\rangle=\frac{1}{\sqrt{2}}\left(\left|01\right\rangle+\left|10\right\rangle\right)$; $\left|\Psi^{-}\right\rangle=\frac{1}{\sqrt{2}}\left(\left|01\right\rangle-\left|10\right\rangle\right)$ [109].

Боб готує пару кубітів в стані $\left|\Psi^{+}\right\rangle$. В якості двох базисних станів кубітів $\left|0\right\rangle$ та $\left|1\right\rangle$ в цьому протоколі можуть бути використані, наприклад, поля-

ризаційні стани фотонів: $|0\rangle$ відповідає вертикальній, а $|1\rangle$ – горизонтальної поляризації фотона. Боб зберігає один із кубітів стану $|\Psi^+\rangle$ («домашній кубіт») в квантовій пам'яті і надсилає Алісі (сторона яка відправляє повідомлення) другий кубіт («кубіт що передається») квантовим каналом зв’язку. Далі Аліса випадковим чином перемикається між двома режимами (режимом передачі повідомлення та режимом контролю підслуховування). Останній спеціально призначений для виявлення атаки пасивного перехоплення (підслуховування).

Схема режимі передачі повідомлення показана на рис. 2.3. Так Аліса виконує кодування інформації за допомогою унітарної операції над переданим кубітом і посилає цей кубіт назад Бобу.

Кодувальні операції Аліси, що перетворюють стан $|\Psi^+\rangle$ в стани $|\Psi^+\rangle$ або $|\Psi^-\rangle$ і відповідні двійковим «0» і «1», мають вигляд: $U_0 = I$; $U_1 = \sigma_z$, де $I = |0\rangle\langle 0| + |1\rangle\langle 1|$ – тотожний оператор, $\sigma_z = |0\rangle\langle 0| - |1\rangle\langle 1|$ – один з операторів Паулі [90].

Коли Боб отримує кубіт, що передається назад від Аліси, він виконує вимірювання над обома кубітами в базисі $U_0 = I$; $U_1 = \sigma_z$, і тим самим декодує надісланий Алісою біт. Отже, одна пара переплутаних кубітів в цьому пінг-понг протоколі служить для передачі одного біта класичної інформації.

Таким чином, цей варіант пінг-понг протоколу дозволяє передати один біт класичної інформації за один раунд режиму передавання повідомлень і є найпростішим з пінг-понг протоколів з найменшою інформаційною місткістю.

Аліса перемикається у режимі контролю підслуховування та випадковим чином вибирає один з базисів: вертикально-горизонтальний $B_z = \{|0\rangle, |1\rangle\}$ або діагональний $B_x = \{|+\rangle, |-\rangle\}$ та проводить вимірювання стану поляризації переданого кубіта. За допомогою відкритого аутентифікованого класичного каналу зв'язку, Аліса надсилає Бобу результат вимірювання і використаний нею базис.

Боб також перемикається в режим контролю підслуховування та проводить вимірювання стану «домашнього» кубіта в тому ж базисі, який використовувала Аліса. Потім він робить порівняння двох результатів вимірювань - свого та Алісиного. Результати вимірювань повинні бути антикоррельовані, якщо стан $\left|\Psi^{+}\right\rangle$, який приготував Боб, не було змінено при передачі кубіта до Аліси. Таким чином, якщо при вимірюванні в базисі B_z Аліса отримує «0», то Боб повинен отримати «1», і навпаки. Аналогічно, якщо при вимірюванні в базисі B_x Аліса отримує «+», то Боб повинен отримати «–», і навпаки.

Зміна стану $\left|\Psi^{+}\right\rangle$ може відбутися в наслідок двох причин: атаки зловмисника, тобто його операцій над передаваним кубітом або впливу на передаваний кубіт природного шуму, який виникає у каналі. Оскільки згідно з законами квантової механіки не існує способу розрізнити помилки вимірювань в режимі контролю підслуховування, обумовлені цими двома причинами, то Аліса і Боб повинні оцінити середній рівень помилок виконанням достатньої для такої статистичної оцінки кількості раундів контролю підслуховування, а потім порівняти цей рівень з заздалегідь відомим рівнем природного шуму в каналі. Якщо отриманий Алісою і Бобом рівень помилок значно перевищує природний, то вони роблять висновок про атаку і протокол перериваєтся. Однак ясно, що внаслідок чергування режимів передачі повідомлення і контролю підслуховування деяка кількість інформації може витекти до зловмисникові. Хоча існують модифіковані протоколи, які мають вищий рівень стійкості проте їх запропоновано небагато, і вони мають тільки асимптотичну безпеку, тобто атака буде виявлена з високою ймовірністю, але до цього зловмисник зможе одержати деяку частину повідомлення. Отже, виникає проблема підсилення безпеки (підвищення ефективності) таких протоколів, тобто створення таких методів попередньої обробки передаваної інформації, які зроблять перехоплену зловмисником інформацію даремною для нього (наприклад, розробка методів на кшталт [8, 10, 13, 85], згаданого у розділі 1.3. роботи).

2.4. Метод забезпечення стійкості кутритових протоколів

Як показав аналіз кібератак, проведений у розділі 2.1 у протоколах КРК допускається витік частки бітів до порушника під час передавання, а наступна оцінка кількості цих бітів дозволяє виконати процедуру підсилення секретності та звести інформацію порушника про фінальний ключ до нескінченно малої величини (якщо частка бітів, що витекли, не перевищує деякої порогової величини), таким чином досягаючи теоретико-інформаційної (безумовної) стійкості, яка не залежить від обчислювальних та інших технічних можливостей порушника. Проте, вимоги до стійкості протоколів КПБЗ, як вже зазначалось, є значно вищими, ніж до стійкості протоколів КРК, адже в протоколах КПБЗ кожний біт є секретною інформацію і не повинен потрапити до порушника. Отже, хоча протоколи КПБЗ повністю знімають проблему розподілу секретних криптографічних ключів, але мають лише асимптотичну стійкість до некогерентних атак і, безумовно, потребують методів підсилення безпеки. Оскільки імовірність виявити цю атаку при одноразовому контролі підслуховування менше одиниці для всіх відомих протоколів КПБЗ, а крім того помилки в режимі контролю підслуховування будуть створюватися не тільки атакою, але і природним шумом в квантовому каналі зв'язку, то необхідно виконати деяку кількість раундів контролю підслуховування перш, ніж можна буде з упевненістю виявити атаку. Так як режими контролю підслуховування і передачі повідомлення необхідно чергувати випадковим чином, то деяка кількість інформації може бути перехоплена порушником.

Для збільшити швидкості роботи при збереженні стійкості кутритових систем до некогерентних атак запропоновано **метод забезпечення стійкості кутритових протоколів** [22, 62], який виконується у такі етапи:

Етап 1. Аліса обробляє секретне повідомлення $A \in V_n$ ($V_n = \{0,1,2\}^n$, $n \in N$) тритовою симетричною функцією перетворення F_{ska}^{enc}: $B = F_{ska}^{enc}(A, K)$, де K – секретний параметр, $K \in V_k$, $k \in N$, $k < n$, F_{ska}^{enc} – симетрична функція перетворення, $F_{ska}^{enc} : V_n \to V_n$, B – перетворене секретне повідомлення, $B \in V_n$.

Етап 2. Аліса обчислює хеш-код повідомлення B: $H = F_{hf}(B)$, де F_{hf} – тритова хеш функція, $F_{hf} : V_n \to V_h$, $h \in N$, $h < n$, H – хеш-код повідомлення B, $H \in V_h$.

Етап 3. Аліса перетворює хеш-код H асиметричною функцією перетворення F_{aka}^{enc} з використанням відкритого секретного параметру Боба: $J = F_{aka}^{enc}\left(H, K_{op}^{B}\right)$, де K_{op}^{B} – відкритий секретний параметр Боба, $K_{op}^{B} \in V_p$, $p \in N$, F_{aka}^{enc} – асиметрична функція перетворення, $F_{aka}^{enc} : V_h \to V_o$, $o \in N$, J – перетворений хеш-код H, $J \in V_o$.

Етап 4. Аліса формує остаточне повідомлення $C \in V_{n+o}$ для передачі Бобу: $C = (B, J)$, де $B \in V_n$, $J \in V_o$.

Етап 5. Відбувається передача повідомлення C квантовим каналом з використанням протоколів КПБЗ від Аліси до Боба. Навіть якщо Єва перехопить частину повідомлення C залишившись не виявленою, то, не знаючи секретного параметра K, вона не зможе відновити початкове повідомлення A. Слід зауважити, що Аліса і Боб можуть попередньо вибрати таке значення частоти перемикання q між режимами роботи протоколів КПБЗ (із режиму передачі повідомлення в режим контролю підслуховування), при якому ймовірність успішної атаки Єви буде незначною.

Етап 6. Боб отримує повідомлення $C' \in V_{n+o}$ та виділяє з нього частини $B' \in V_n$ та $J' \in V_o$.

Етап 7. Боб обчислює хеш-код повідомлення B': $H' = F_{hs}(B')$, де F_{hf} – тритова хеш-функція, $F_{hf} : V_n \to V_h$, H' – хеш-код повідомлення B', $H' \in V_h$.

Етап 8. Боб виконує зворотне перетворення хеш-коду H'' асиметричним шифром F_{aka}^{dec} з використанням свого закритого секретного параметру: $H'' = F_{aka}^{dec}\left(J', K_{cl}^{B}\right)$, де K_{cl}^{B} – закритий параметр Боба, $K_{cl}^{B} \in V_p$, F_{aka}^{dec} – асиметрична функція зворотного перетворення, $F_{aka}^{dec} : V_o \to V_h$.

Етап 9. Боб порівнює H' і H''. Якщо $H' \neq H''$ – це означає, що повідомлення було модифіковане під час передачі. Одразу припускається, що в сеанс зв'язку втручалась Єва. Тому Боб і Аліса переривають сеанс. Згідно теореми про заборону клонування, порушник не може виготовити точну копію квантових систем, які передаються комунікаційним каналом, щоб провести виміри над копією, а оригінал переслати легітимному користувачу каналу, не проводячи над ним вимірювання. Це змушує порушника вимірювати стан квантових систем, що передаються (або переплутувати їх зі своїми квантовими пробами), що, згідно постулату вимірювання, призводить до зміни їх станів (у такому випадку $B' \neq B$ і $H' \neq H''$). Якщо ж $H' = H''$ – це означає, що втручання Єви не було і $B' = B$.

Етап 10. Боб повідомляє Алісу про те, що при передачі повідомлень втручань не було. Аліса в свою чергу відкритим каналом зв'язку передає Бобу секретний параметр K.

Етап 11. Боб відновлює секретне повідомлення A обробляє тритовою симетричною функцією зворотного перетворення F_{ska}^{dec}: $A = F_{ska}^{dec}\left(B', K\right)$, F_{ska}^{dec} – симетрична функція зворотного перетворення, $F_{ska}^{dec} : V_n \to V_n$.

У якості симетричної функції перетворення та зворотного перетворення можуть бути використані як трійкові блокові, так і потокові перетворення (проте, ці процедури не є шифруванням, так як K передається відкритим каналом для встановлення легітимності користувача, а це не відповідає принципам криптографії). Зауважимо, що при такій побудові роботи протоколів КПБЗ, частоту перемикання q між режимами їх роботи можна зменшити до мінімуму (із рекомендованого значення 0,5 до 0,05), при цьому підвищиться швидкість роботи протоколів і втручання Єви все одно буде детектуватись (на етапах 5 і 9).

Отже, отримав подальший розвиток метод забезпечення стійкості кутритових протоколів квантової криптографії, який, за рахунок неквантової функції перевірки цілісності та використання тритової симетричної функції, дозволяє звести до мінімуму кількість перемикань між режимами роботи протоколу (передавання повідомлення та контролю підслуховування), збільшити швидкість роботи (чис-

ловий показник виграшу обчислюється в четвертому розділі роботи) при збереженні стійкості до некогерентних атак.

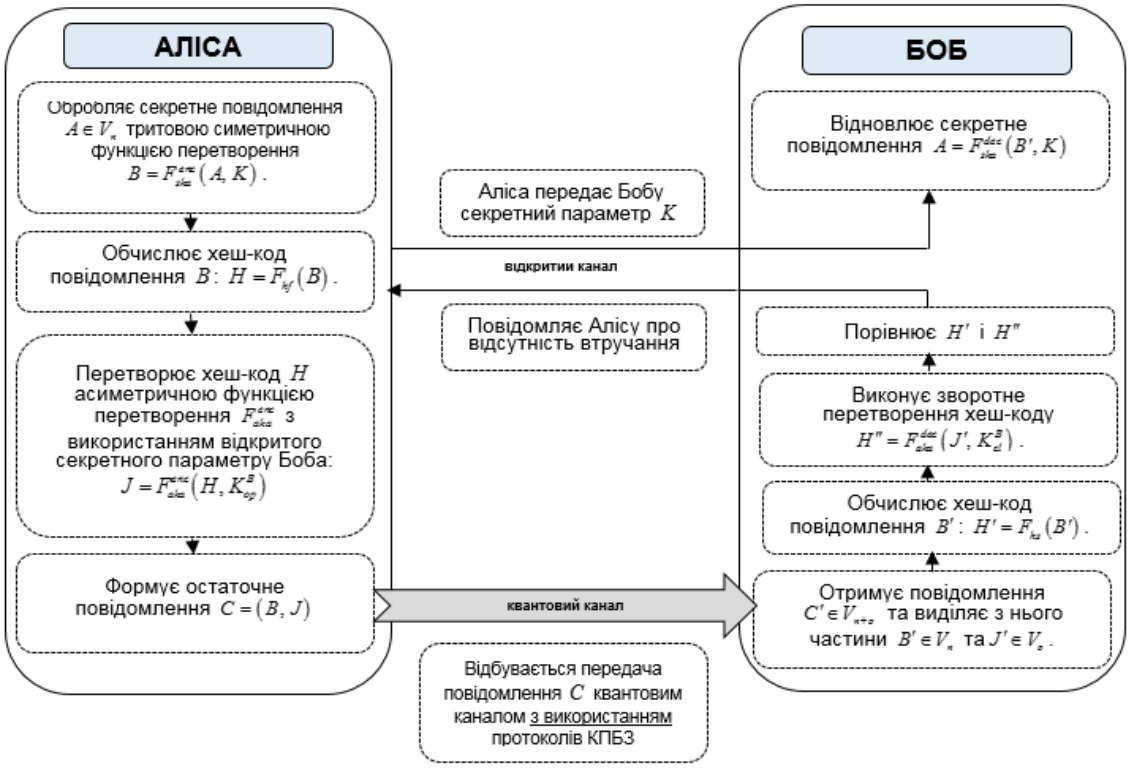

Рис. 2.3. Схема реалізації методу забезпечення стійкості протоколу КПБЗ

2.5. Висновки до другого розділу

У другому розділі дисертаційної роботи були отримані наступні **вагомі результати:**

1. Проведено якісний аналіз сучасних методів та протоколів квантової криптографії, визначено їх переваги і недоліки, стійкість і уразливість до різного роду кібератак (зокрема, до некогерентних атак). На підставі часткових узагальнень теоретичних положень та практичних досягнень у галузі квантової криптографії, розроблено розширену класифікацію методів квантової криптографії, дозволяє розширити можливості щодо вибору відповідних методів для побудови сучасних квантових систем захисту інформації (на базі КПБЗ та інших квантових технологій). Така класифікація також дала можливість чітко визначити завдання дисертаційного дослідження щодо подальшої розробки мето-

дів забезпечення стійкості тритових протоколів і систем, побудови тритових генераторів псевдовипадкових послідовностей та оцінювання їх якості.

2. Розроблено метод забезпечення стійкості тритових протоколів квантової криптографії, що не потребує великих часових та ресурсних затрат і, за рахунок неквантової функції перевірки цілісності та використання тритової симетричної функції, дозволяє підвищити швидкість роботи протоколу при збереженні стійкості до некогерентних атак (оцінка швидкісних характеристик буде проведена у 4 розділі дисертаційної роботи).

РОЗДІЛ 3

МЕТОДИ ГЕНЕРУВАННЯ ТРИТОВИХ ПОСЛІДОВНОСТЕЙ ТА ОЦІНЮВАННЯ ЇХ ЯКОСТІ

3.1. Підвищення інформаційної місткості протоколів квантової криптографії

Оригінальний варіант пінг-понг протоколу КПБЗ, використовуючи два стани Бела переплутаної пари кубітів, передавав за один цикл протоколу лише один класичний біт інформації [108]. Якщо ж використовувати всі чотири стани Бела пари кубітів (використовуючи квантове надщільне кодування), можна передати за цикл два класичні біти інформації [112]. Окрім того, збільшення інформаційної місткості протоколів можливе за умови використання не переплутаних пар кубітів, а переплутаних трійок, четвірок та т.д. кубітів [8], а також інформаційну місткість можна збільшити за рахунок використання переплутаних станів багаторівневих квантових систем (наприклад, кутритів, куквартів тощо). Так у роботах [12, 36, 40, 62, 194] представлено модифікації пінг-понг протоколу з використанням переплутаних станів пар та триплетів трирівневих систем (кутритів) та квантового надщільного кодування для кутритів.

Двійкову логіку, яка сьогодні широко поширена у всіх цифрових пристроях набагато легше реалізувати, ніж будь-яку іншу, проте історично перший електронний цифровий комп’ютер загального призначення ЕНІАК (розробник Дж. Моклі) використовував десяткову систему числення, однак був дуже громіздким та потребував багато елементів, використання ж двійкової логіки значно все спрощувало, адже у розповсюджених сьогодні логічних елементів лише два стани, тому запис даних є заздалегідь простішим. Проте, питома натурально-логарифмічна щільність запису інформації описується функцією [35]:

$$Y(\mathrm{a}) = \frac{\ln y(\mathrm{a})}{a} = \frac{\ln a}{a}, \qquad (3.1)$$

де *a* – це основа системи числення. З (3.1) випливає (і з рис. 3.1 видно), що найбільшу щільність запису інформації має система числення з основою рівною основі натуральних логарифмів, тобто рівною числу Ейлера (e ≈ 2,718281828459045), а з цілочислених – це трійкова система.

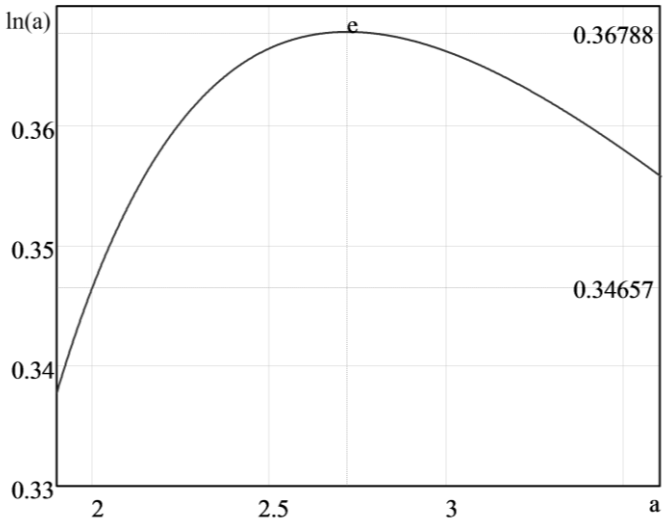

Рис. 3.1. Питома натуральнологарифмічна щільність запису інформації

Окрім того зазначимо переваги використання трійкової логіки згідно з [123, 125, 175] полягають у наступному:

1. *Представлення чисел.* Наприклад, для представлення десяткового числа 16 необхідно: в бінарній системі потрібно 5 біт (1000_2); в трійковій системі потрібно 3 біти (121_3).

2. *Процесори.* Для 8 біт двійковий мікропроцесор має $2^8 = 256$ інструкцій, а трійковий – $3^8 = 6561$.

3. *Конвертація* відповідно п. 2 призводять до точної і ефективної обробки цифрового сигналу.

4. *Трійкова суперпозиція* $\underline{a'|0\rangle + \hat{a}|1\rangle + \tilde{a}|2\rangle}$ [175]. Пари кутритів, можуть представитисm дев'ятьма різними станами: $|00\rangle$, $|01\rangle$, $|02\rangle$, $|10\rangle$, $|11\rangle$, $|12\rangle$, $|20\rangle$, $|21\rangle$, $|22\rangle$. Тоді їх суперпозиція матиме вигляд:

$$\psi_1 = \alpha_1|0\rangle + \beta_1|1\rangle + \gamma_1|2\rangle \qquad \psi_2 = \alpha_2|0\rangle + \beta_2|1\rangle + \gamma_2|2\rangle$$

$$\psi_{12} = \psi_1 \otimes \psi_2 = \alpha_1\alpha_2|00\rangle + \alpha_1\beta_2|01\rangle + \alpha_1\gamma_2|02\rangle + \beta_1\alpha_1|10\rangle + \beta_1\beta_2|11\rangle + \beta_1\gamma_2|12\rangle +$$

$\gamma_1\alpha_1|20\rangle + \gamma_1\beta_2|21\rangle + \gamma_1\gamma_2|22\rangle$, де α, β, γ – комплексні числа.

Логічний елемент пам'яті завдяки якому можна реалізувати трійкову логіку називається – Flip-flap-Flop [123].

3.2. Метод генерування тритових послідовностей

Застосування розробленого у розділі 2.4 методу потребує генерування трійкових (тритових) ПВП. Проведений аналіз дозволив виявити достатню кількість існуючих генераторів ПВП (ГПВП), які можуть використовуватись для різного роду застосувань [14, 25, 27, 33, 36, 44, 47, 60, 75]. Стандарт ISO/IEC 18031, який встановлює концептуальні моделі, термінологію і вимоги, що відносяться до конструктивних елементів і властивостей систем, які використовуються для генерації випадкових бітів в криптографічних застосування, визначає два типи генераторів: *недетерміновані* (механізм генерації випадкових бітів, який використовує джерело ентропії для генерації випадкового потоку бітів) і *детерміновані* (механізм генерації бітів, який використовує детерміновані механізми, такі як криптографічні алгоритми, на джерелі ентропії для генерації випадкового потоку бітів. Використовує особливі вхідні дані, і, якщо необхідно, деякі необов'язкові вхідні дані, які, залежно від їх застосування можуть бути загальнодоступними) генератори випадкових бітів [25].

За способом отримання ГПВП діляться на три принципово різних класи: *табличні* – основний недолік є скінченими, *фізичні* (радіоактивне випромінювання, фізичні генератори шумів, квантові генератори, генератори імпульсних послідовностей тощо) - спільними і найбільш суттєвими недоліками, що утруднюють їх застосування є обмежена швидкодія, низька стабільність основних імовірнісних характеристик, що пояснюється нестабільністю первинних джерел, дрейфом параметрів перетворюючих схем, джерел живлення та вимагає періодичної статистичної перевірки якості; складність апаратурної реалізації, *алгоритмічні* (ГПВП, наприклад, метод серединних квадратів, метод серединних добутків, метод перемішування, лінійний конгруентний метод) [60].

Також ГПВП можна класифікувати за різними законами розподілу [14], проте найбільш повна класифікація представлена у роботі [27] та відображена на рис. 1. Відповідно до цієї класифікації методи формування ПВП поділяються на к*риптостійкі* та *не криптостійкі*. До яких у свою чергу відносять *криптостійкі*: на основі потокових шифрів (наприклад, Dragon-128, SEAL, RC4, RC5, RC6, Grain, Yamb, Phelix), на основі блокових шифрів (наприклад, ГОСТ 28147-89, AES, ANSI X9.17, DES), на основі односторонніх функцій (наприклад, генератори BBS, RSA, Dual_EC_DRBG (еліптичні криві), GPSSD (лінійні коди) та ін.) і *не криптостійкі:* на основі елементарних рекурентів (наприклад, лінійний конгруентний генератор, поліноміальний конгруентний генератор, адитивний генератор Фібоначчі, адитивний генератор Фібоначчі з запізненням, мультиплікативний генератор Фібоначчі з запізненням), на основі операцій в кінцевих полях (наприклад, генератори Галуа, Де Брейна, Фібоначчі, адитивний, Голмана, стискаючий тощо). Проте всі розроблені на сьогодні методи генерування ПВП (генератори) орієнтовані на бінарні послідовності, а отже розробка методу генерування тритових ПВП є актуальним науковим завданням. З огляду на це, запропоновано метод генерування тритових ПВП ξ із множиною векторів внутрішніх станів V_p ($V_p = \{0,1,2\}^p$), множиною секретних ключів V_n, множиною векторів ініціалізації V_e та множиною вихідних послідовностей V_m, де $p = 14 \cdot l$, $n = 4 \cdot l$, $e = p - n = 10 \cdot l$, $m = b \cdot n$, $l = d \cdot s$ і b, d, s – натуральні числа.

Для генерації вихідної тритової послідовності виконується наступне:

Етап 1. Виконується початкова ініціалізація вектора внутрішнього стану U на основі вектора ініціалізації VI та секретного ключа K, $U \in V_p$, $VI \in V_e$, $K \in V_n$.

Нехай $U = (x_1, x_2, x_3, x_4, x_5, x_6, y_1, y_2, y_3, y_4, k_1, k_2, k_3, k_4)$, де x_i, y_j, k_j – частини вектора внутрішнього стану U ($x_i \in V_l$, $y_j \in V_l$, $k_j \in V_l$, $i \in \overline{1,6}$, $j \in \overline{1,4}$); $VI = (VI_1, VI_2, VI_3, VI_4, VI_5, VI_6, VI_7, VI_8, VI_9, VI_{10})$, де VI_o – частини вектора ініціалізації VI ($VI_o \in V_l$, $o \in \overline{1,10}$); $K = (K_1, K_2, K_3, K_4)$, де K_w – частини секретного ключа K ($K_w \in V_l$, $w \in \overline{1,4}$).

Тоді внутрішній стан вектора U ініціалізується таким чином:

$$x_i = VI_i,\ y_j = VI_{6+j},\ k_j = K_j,\ i \in \overline{1,6},\ j \in \overline{1,4}.$$

Етап 2. На основі поточних значень внутрішнього стану вектора U виконується поступова генерація вихідної послідовності $M = (M_1,...,M_b)$, $M \in V_m$, M_q – частини вихідної послідовності M, $M_q \in V_n$, $q \in \overline{1,b}$. Зауважимо, що при генерації кожного M_q поточні значення внутрішнього стану вектора U весь час змінюються.

2.1. Для генерації частини вихідної послідовності M_q r-разів ($q \in \overline{1,b}$, r – натуральне число) виконується наступне:

2.1.1. *Розраховуються нові значення векторів* x_1, x_2, x_3. Спочатку визначається x_1: $x_1' = Sbox(x_1 + k_1)$; $x_1 = (x_1' \oplus x_4) <<< k_4$. Далі обраховується x_2: $x_2' = Sbox(x_2 + k_2)$; $x_2 = (x_2' + x_5) >>> k_3$. Наприкінці розраховується x_3: $x_3' = (x_3 + x_6) \oplus y_3$; $x_3 = Mix(x_3') <<< x_1$.

2.1.2. *Обчислюються нові значення векторів* k_1, k_2, y_1, y_2. Спочатку визначаються k_1 та k_2: $k_1' = Sbox(x_1 \oplus k_1) + x_5$; $k_1 = Sbox(k_1' \oplus y_1)$; $k_2' = Mix(x_2 + k_2 + x_6)$; $k_2 = Sbox(k_2' \oplus y_2)$. Далі обраховуються значення y_1 та y_2: $y_1' = (k_1 + y_1) <<< x_2$; $y_1 = Sbox(y_1' \oplus k_3)$; $y_2' = ((k_2 + y_2) >>> x_3) \oplus k_4$; $y_2 = Mix(Sbox(y_2'))$.

2.1.3. *Розраховуються нові значення векторів* x_4, x_5, x_6. Спочатку визначається x_4: $x_4' = Sbox(x_4 + k_3)$; $x_4 = (x_4' \oplus x_1) <<< k_2$. Далі обраховується x_5: $x_5' = Sbox(x_5 + k_4)$; $x_5 = (x_5' + x_2) >>> k_1$. Наприкінці розраховується x_6: $x_6' = (x_6 + x_3) \oplus y_1$; $x_6 = Mix(x_6') <<< x_4$.

2.1.4. *Обчислюються нові значення векторів* k_3, k_4, y_3, y_4. Спочатку визначаються k_3 та k_4: $k_3' = Sbox(x_4 \oplus k_3) + x_2$; $k_3 = Sbox(k_3' \oplus y_3)$; $k_4' = Mix(x_5 + k_4 + x_3)$; $k_4 = Sbox(k_4' \oplus y_4)$. Далі обраховуються значення y_3 та

y_4: $y'_3 = (k_3 + y_3) <<< x_5$; $y_3 = Sbox(y'_3 \oplus k_1)$; $y'_4 = ((k_4 + y_4) >>> x_6) \oplus k_2$; $y_4 = Mix(Sbox(y'_4))$.

2.2. За допомогою конкатенації векторів y_i обраховується вихідна послідовності M_q: $M_q = (y_1 | y_2 | y_3 | y_4)$.

У зазначених вище формулах, символи $\oplus$ та $+$ відповідають операціям покоординатного додавання за модулем 3 та алгебраїчну операцію додавання за модулем 3^l відповідно. Під операцією $X <<< Y$ розуміємо операцію циклічного зсув вліво числа X на Y разів, а під $X >>> Y$ − циклічного зсуву вправо числа X на Y разів. Під операцією $Sbox(X)$ розуміємо операцію в якій X розбивається на d частин довжиною s тритів, над кожною з яких виконується підстановка на множині V_s: $Sbox(X) = (S(X_1),...,S(X_d))$, де $X = (X_1,..., X_d)$, $X \in V_l$, $X_i \in V_s$, $i \in \overline{1,d}$, а s − підстановка на зазначеній множині. . $Mix(X)$. відповідає операції у якій виконується лінійне розсіювання тритів вектора X (у якості $Mix(X)$ можуть бути використані МДР-коди).

Алгоритм TriGen. На основі запропоновано метода генерування тритових ПВП ξ розроблено алгоритм TriGen (псевдокод алгоритму див. на рис. 3.2). У алгоритмі TriGen використовуються такі параметри $d = 4$, $s = 6$, $l = d \cdot s = 24$, $p = 14 \cdot l = 336$, $n = 4 \cdot l = 96$, $e = p - n = 10 \cdot l = 240$, $m = b \cdot n = 96 \cdot b$, $r = 4$, b − натуральне число.

В операції $Sbox(X)$ виконується нелінійна заміна кожних шести тритів числа X на відповідне їм значення таблиці підстановок. Використовується лише одна таблиця підстановок, що побудована за допомогою обрахунку зворотного елементу поля $(X)^{-1} \in GF(3^6)$ з подальшим виконанням афінного перетворення над полем $GF(3)$: $S(X) = M \cdot (X)^{-1} + V$, де $X, V \in GF(3^6)$, а M − квадратна не вироджена матриця над полем $GF(3)$ розміром 6×6. Кінцеве поле $GF(3^6)$ фіксується кільцем многочленів з операціями за модулем незвідного многочлена $m(x) = x^6 + x + 2$.

Для побудови запропонованої таблиці замін були обрані такі значення матриці M та вектора V:

$$M = \begin{pmatrix} 1 & 2 & 0 & 1 & 1 & 2 \\ 2 & 1 & 2 & 0 & 1 & 1 \\ 1 & 2 & 1 & 2 & 0 & 1 \\ 1 & 0 & 2 & 1 & 2 & 0 \\ 0 & 1 & 0 & 2 & 1 & 2 \\ 2 & 0 & 1 & 1 & 2 & 1 \end{pmatrix}, \; V = \begin{pmatrix} 0 \\ 2 \\ 2 \\ 1 \\ 0 \\ 2 \end{pmatrix}.$$

В операції $Mix(X)$ квадратна не вироджена матриця M' над полем $GF(3)$ розміром 24×24 тритів множиться на X (представлений у вигляді вектора-стовпчика) над полем $GF(3)$. Матриця M' побудована на основі масиву U таким чином: $M'[i][j] = U\left[\left(j + 24 - i\right) \bmod 24\right]$, де $i, j = \overline{0,...,23}$, а масив U приймає значення: U = {1, 0, 2, 2, 1, 0, 2, 0, 1, 1, 1, 2, 0, 1, 2, 1, 0, 2, 0, 0, 1, 2, 0, 2 }.

TriGen

Input: вектор ініціалізації VI, секретний ключ K, $VI \in V_{240}$, $K \in V_{96}$, параметр b.

Output: вихідна послідовність $M = (M_1,...,M_b)$, $M \in V_{96b}$, $M_q \in V_{96}$, $q \in \overline{1,b}$.

1. $x_i = VI_i$, $y_j = VI_{6+j}$, $k_j = K_j$, $i \in \overline{1,6}$, $j \in \overline{1,4}$.

2. *For* $q = 1; q \le b;$ $q{+}{+}$ *do*

2.1. *For* $j = 0;$ $j < 4;$ $j{+}{+}$ *do*

2.1.1. $x_1 = \left(Sbox\left(x_1 + k_1\right) \oplus x_4\right) <<< k_4$; $x_2 = \left(Sbox\left(x_2 + k_2\right) + x_5\right) >>> k_3$;

$x_3 = Mix\left(\left(x_3 + x_6\right) \oplus y_3\right) <<< x_1$;

2.1.2. $k_1 = Sbox\left(\left(Sbox\left(x_1 \oplus k_1\right) + x_5\right) \oplus y_1\right)$; $k_2 = Sbox\left(Mix\left(x_2 + k_2 + x_6\right) \oplus y_2\right)$;

2.1.3. $y_1 = Sbox\left(\left(\left(k_1 + y_1\right) <<< x_2\right) \oplus k_3\right)$; $y_2 = Mix\left(Sbox\left(\left(\left(k_2 + y_2\right) >>> x_3\right) \oplus k_4\right)\right)$;

2.1.4. $x_4 = \left(Sbox\left(x_4 + k_3\right) \oplus x_1\right) <<< k_2$; $x_5 = \left(Sbox\left(x_5 + k_4\right) + x_2\right) >>> k_1$;

$x_6 = Mix\left(\left(x_6 + x_3\right) \oplus y_1\right) <<< x_4$;

2.1.5. $k_3 = Sbox\left(\left(Sbox\left(x_4 \oplus k_3\right) + x_2\right) \oplus y_3\right)$; $k_4 = Sbox\left(Mix\left(x_5 + k_4 + x_3\right) \oplus y_4\right)$;

2.1.6 $y_3 = Sbox\left(\left(\left(k_3 + y_3\right) <<< x_5\right) \oplus k_1\right)$; $y_4 = Mix\left(Sbox\left(\left(\left(k_4 + y_4\right) >>> x_6\right) \oplus k_2\right)\right)$.

2.2. $M_q = \left(y_1 \mid y_2 \mid y_3 \mid y_4\right)$

Рис. 3.2. Псевдокод алгоритму TriGen

3.3. Метод оцінки рівня випадковості тритових послідовностей

Як показав проведений аналіз [39, 73, 74, 92, 97, 98, 167, 172, 173, 187, 189], сьогодні найбільш дослідженою та популярною методикою оцінювання ПВП для криптографічних застосувань є методика NIST STS [172, 173], яку і було взято за основу для розробки тритових тестів.

Запропонований метод оцінювання якості ПВП [21], що дає можливість оцінювати параметри і закономірності тритових ПВП, реалізується у такі етапи: ***Eman 1.*** Перевірка частотним тритовим тестом (Freqeuncy Monotrit Test, FMT). ***Eman 2.*** Дослідження частотним блоковим тестом (Frequency Trit Block Test, FTBT). ***Eman 3.*** Перевірка тритовим тестом серій (Trit Runs Test, TRT). ***Eman 4.*** Дослідження тритовим тестом найдовших серій (Trit Test for the longest run in a block, TTLROB). ***Eman 5.*** Перевірка тритовим тестом на співпадіння з шаблоном без перекриття (Non-Overlapping Template Matching Trit Test, NTMTT). ***Eman 6.*** Дослідження тритовим тестом шаблонів із перекриттям (Trit Overlapping Template Matching Test, TOTMT).

На кожному з шести зазначених етапів послідовність перевіряється таким чином:

1. Спочатку кожна вхідна трійкова послідовність A_{012} розбивається на 3 підпослідовності: A_{01}(послідовність A_{012} із видаленими 2), A_{02} (послідовність A_{012} із видаленими 1), A_{12} (послідовність A_{012} із видаленими 0).

2. Кожна із отриманих послідовностей окремо перевіряється тритовими тестами, аналогічно тестам NIST STS. У результаті перевірки кожним тестом отримуємо 3 значення $P-value$: $P-value_{01}$, $P-value_{02}$, $P-value_{12}$. Як і в тестах NIST STS $P-value_{XY}$ (під XY тут і надалі розуміємо одну із трьох можливих комбінацій послідовностей: «01», «02» та «12») відповідатиме ймовірності того, що досліджувана послідовність A_{XY} не гірша, ніж істино-випадкова, тобто якщо $P-value_{XY}=1$, то згенерована послідовність є ідеально випадковою, а якщо $P-value_{XY}=0$, то послідовність є повністю передбачуваною.

3. Визначені значення $P-value_{01}$, $P-value_{02}$, $P-value_{12}$ кожного тесту порівнюється із α (помилкою першого роду – ймовірність того, що випадкова послідовність є забракованою). Якщо $P-value_{XY} \geq \alpha$, то послідовність A_{XY} є випадковою з рівнем довіри 99%, у іншому випадку $P-value_{XY} \leq \alpha$ – послідовність A_{XY} відбраковується з рівнем довіри 99%. Будемо вважати кожен тест пройденим послідовністю A_{012}, якщо усі отримані значення $P-value_{01}$, $P-value_{02}$, $P-value_{12}$ будуть випадковими з рівнем довіри 99%, тобто виконуватимуться нерівності $P-value_{01} \geq \alpha$, $P-value_{02} \geq \alpha$ та $P-value_{12} \geq \alpha$.

Якщо досліджувана тритова послідовність пройде усі зазначені тести, то вважатимемо її псевдовипадковою. До того ж, у випадку не проходження хоча б одного із етапів, перевірка завершується і послідовність вважається передбачуваною і непридатною для криптографічних застосувань. Перейдемо до опису базових етапів реалізації методу.

Етап 1. Частотний тритовий тест

Мета тесту − визначити, чи буде кількість нулів, одиниць та двійок у трійковій послідовності A_{012} приблизно така ж як у дійсно випадковій послідовності. Тест окремо оцінює наскільки близька пропорція нулів послідовності A_{01}, пропорція одиниць послідовності A_{12} та пропорція двійок послідовності A_{02} до $1/2$.

Позначення: n_{012} – довжина вхідної послідовності тритів $A_{012} = \{0,1,2\}^{n_{012}}$, n_{01} – довжина послідовності $A_{01} = \{0,1\}^{n_{01}}$, n_{02} – довжина послідовності $A_{02} = \{0,2\}^{n_{02}}$ та n_{12} – довжина послідовності $A_{12} = \{1,2\}^{n_{12}}$.

Статистика тесту та граничний розподіл. $Sobs_{XY}$ − абсолютна величина суми M_i по всій довжині підпослідовності A_{XY} ($A_{XY} = \left(a_1, a_2, ..., a_{n_{XY}}\right), a_i = \{X, Y\}, i = \overline{1, n_{XY}}$), що поділена на корінь квадратний з

довжини підпослідовності n_{XY}: $Sobs_{XY} = \frac{\left|S_{n_{XY}}\right|}{\sqrt{n_{XY}}}$, де $S_{n_{XY}} = \sum_{i=1}^{n_{XY}} M_i$, $M_i = -1$ – якщо $a_i = X$ та $M_i = 1$ – якщо $a_i = Y$. Граничний розподіл тестової статистики при великих n_{XY} – напівнормальний. Якщо послідовність A_{XY} випадкова, тоді «+1» та «−1» будуть компенсувати один одного та статистика тесту буде мати значення близькі до нуля. Якщо в послідовності A_{XY} кількість X і Y значно відрізняються, тоді значення статистики тесту будуть значно відхилятися від нуля.

Опис тесту. Для послідовності $A_{XY} = (a_1, a_2, ..., a_n)$, $a_i = \{X, Y\}$, $i = \overline{1, n_{XY}}$ виконуємо перетворення до ±1: всі X замінюємо на -1, а всі Y замінюємо на $+1$, тобто будуємо нову послідовність $M = M_1, ...M_n$, $M_i = \{-1, +1\}$, $i = \overline{1, n_{XY}}$. Підраховуємо величину $S_{n_{XY}} = M_1 + M_2 + ... + M_n$. Далі підраховуємо значення статистики тесту $Sobs_{XY} = \frac{\left|S_{n_{XY}}\right|}{\sqrt{n_{XY}}}$. Після чого підраховуємо $P - value_{XY} = erfc\left(\frac{Sobs_{XY}}{\sqrt{2}}\right)$, де $erfc$ – комплементарна функція похибки, що визначається таким чином: $erfc(z) = \frac{2}{\sqrt{\pi}} \cdot \int_z^{+\infty} e^{-u^2} du$.

Приклад. Якщо $A_{012} = 012120001221110112110112 0$, тоді $A_{01} = 0110001111011110110$, $n_{01} = 19$, $A_{02} = 0220002202020$, $n_{02} = 13$, $A_{12} = 121212211111211112$, $n_{12} = 18$.

Тоді $S_{n_{01}} = 5$, $S_{n_{02}} = -1$ і $S_{n_{12}} = 6$. Звідки $Sobs_{01} = \frac{|5|}{\sqrt{19}} = 1.1470$, $Sobs_{02} = \frac{|-1|}{\sqrt{13}} = 0.2773$, $Sobs_{12} = \frac{|6|}{\sqrt{18}} = 1.4142$.

Тоді $P - value_{01} = erfc\left(\frac{1.1470}{\sqrt{2}}\right) = 0.2513$, $P - value_{02} = erfc\left(\frac{0.2773}{\sqrt{2}}\right) = 0.7815$, $P - value_{12} = erfc\left(\frac{1.4142}{\sqrt{2}}\right) = 0.1573$.

Вирішальне правило (для рівня значущості 1%). Якщо підраховане значення $P-value_{XY}$ менше за 0.01, тоді робимо висновок, що послідовність A_{XY} не випадкова, в іншому випадку робимо висновок, що послідовність A_{XY} випадкова.

Висновки та інтерпретація результатів тесту. Так як значення $P-value_{01}$, $P-value_{02}$, $P-value_{12}$, що отримані у прикладі ≥ 0.01, робимо висновок, що послідовності A_{01}, A_{02} і A_{12} випадкові. Тому вважаємо, що послідовність A_{012} теж випадкова і пройшла цей тест.

Рекомендації щодо вхідних розмірам. Кожна послідовність .., що тестується, повинна складалася як мінімум зі 150 тритів ($n_{012} \geq 150$).

Математична модель: $X = \sum_{i=1}^{N} E_i^{01} + i\sum_{i=1}^{N} E_i^{02}$ (У випадку з бітовими послідовностями $S_n = E_1 + E_2 + ... + E_n$, де $X_i = 2E_i - 1$ і фактично «1» перетворюється в «+1», а «0» в «−1», а у нашому випадку потрібно враховувати додаткове значення «2». Тому згадана рівність прийме комплексний вигляд: $X_i = (2E_i^{01} - 1) + i(E_i - 1)$, де дійсна частина $(2E_i^{01} - 1)$, аналогічно бінарним системам рахуватиме суму «0» та «1» в згенерованій послідовності: $S_n^{01} = E_1^{01} + E_2^{01} + ... + E_n^{01}$, а уявна частина $i(E_i - 1)$ – суму «0» та «2», тобто $S_n^{02} = E_1^{02} + E_2^{02} + ... + E_n^{02}$. Подальші розрахунки враховують суми дійсних і уявних частин цього комплексного числа. Визначаються абсолютні суми $S_{abs}^{01} = \frac{\left|S^{01}\right|}{\sqrt{n}}$, $S_{abs}^{02} = \frac{\left|S^{02}\right|}{\sqrt{n}}$, далі $P-value$ для обох випадків $P_{-value}^{01} = erfc\,(\frac{S_{abs}^{01}}{\sqrt{2}})$, $P_{-value}^{02} = erfc\,(\frac{S_{abs}^{02}}{\sqrt{2}})$, де $erfc\,(\mathrm{x}) = \frac{2}{\sqrt{\pi}}\int_x^{\infty} e^{-t^2} dt$ – додаткова функція помилок. Далі обидва значеення $P-value$ порівнюємо з α і приймаємо рішення щодо проходження послідовністю тесту FMT.)

Етап 2. Частотний блоковий тритовий тест

Мета тесту. Цей тест спрямовано на визначення пропорцій X і Y пос-

лідовності A_{XY} у блоках довжиною M. Мета тесту – визначити, чи буде кількість одиниць послідовності A_{01} нулів послідовності A_{02} та двійок послідовності A_{12} у середині кожного блоку приблизно дорівнювати $M/2$, як це очікується від випадкової послідовності.

Позначення: M – довжина кожного блоку; n_{012} – довжина вхідної послідовності тритів $A_{012}=\{0,1,2\}^{n_{012}}$, n_{01} – довжина послідовності $A_{01}=\{0,1\}^{n_{01}}$, n_{02} – довжина послідовності $A_{02}=\{0,2\}^{n_{02}}$ та n_{12} – довжина послідовності $A_{12}=\{1,2\}^{n_{12}}$.

Статистика тесту та граничний розподіл. $\chi^2(\text{obs})_{XY}$ – міра того, як добре пропорція X і Y послідовності A_{XY} в межах даних блоків довжиною M відповідає пропорції, що очікується за припущенням випадковості послідовності (1/2). Граничний розподіл такої статистики є χ^2 -розподіл.

Опис тесту. Ділимо вхідну підпослідовність $A_{XY}=\left(a_1,a_2,...,a_{n_{XY}}\right)$, $a_i=\{X,Y\}$, $i=\overline{1,n_{XY}}$ на $N_{XY}=\left\lfloor \frac{n_{XY}}{M} \right\rfloor$ блоків, що не перетинаються. Відкидаємо останні символи послідовності, що не утворюють повного блоку довжини M (якщо такі є). Визначаємо пропорцію $\pi_{j,XY}$ для кожного з блоків $j=1,2...N_{XY}$ за формулою: $\pi_{j,XY}=\frac{\sum_{l=1}^{M}\varepsilon_{(j-1)\cdot \mathrm{M}+l}}{M}$, де $\varepsilon_i=0$ – якщо $a_i=X$, $\varepsilon_i=1$ – якщо $a_i=Y$, $i=\overline{1,n_{XY}}$.

Підраховуємо значення статистики $\chi^2(\text{obs})_{XY}=4\cdot \mathrm{M}\cdot\sum_{j=1}^{N_{XY}}\left(\pi_{j,XY}-\frac{1}{2}\right)^2$ та вираховуємо значення $P-value_{XY}=igamc\left(\frac{N_{XY}}{2},\frac{\chi^2(\text{obs})_{XY}}{2}\right)$, де $igams(.)$ – не-

повна гамма функція, що визначається таким чином $igamc(a,b)=\frac{1}{\Gamma(a)}\cdot\int_{b}^{+\infty}e^{-u}\cdot u^{a-1}du$.

Приклад. Якщо $A_{012}=012120001221110112110112 0$, тоді $A_{01}=01100011110111101 10$, $n_{01}=19$, $A_{02}=022000220202 0$, $n_{02}=13$, $A_{12}=121212211111211112$, $n_{12}=18$. Нехай $M=4$.

Тоді із послідовності A_{01} сформуються 4 блоки ($N_{01}=\left\lfloor\frac{19}{4}\right\rfloor=4$): 0110 0011 1101 1110, із послідовності A_{02} сформуються 3 блоки ($N_{02}=\left\lfloor\frac{13}{4}\right\rfloor=3$): 0220 0022 0202, і відповідно з A_{12} – 4 блоки ($N_{12}=\left\lfloor\frac{18}{4}\right\rfloor=4$): 1212 1221 1111 2111.

Для кожної послідовності A_{XY} визначаємо пропорцію $\pi_{j.XY}$ для кожного з блоків $j=1,2...N_{XY}$. Для A_{01}: $\pi_{1,01}=1/2$, $\pi_{2,01}=1/2$, $\pi_{3,01}=3/4$, $\pi_{4,01}=3/4$; для A_{02}: $\pi_{1,02}=1/2$, $\pi_{2,02}=1/2$, $\pi_{3,02}=1/2$; для A_{12}: $\pi_{1,12}=1/2$, $\pi_{2,12}=1/2$, $\pi_{3,12}=0$, $\pi_{4,12}=1/4$.

Тоді $\chi^2(\text{obs})_{01}=4\cdot4\cdot\sum_{j=1}^{4}\left(\pi_{j,01}-\frac{1}{2}\right)^2=2$, $\chi^2(\text{obs})_{02}=4\cdot4\cdot\sum_{j=1}^{3}\left(\pi_{j,02}-\frac{1}{2}\right)^2=0$, $\chi^2(\text{obs})_{12}=4\cdot4\cdot\sum_{j=1}^{4}\left(\pi_{j,12}-\frac{1}{2}\right)^2=5$. Звідки $P-value_{01}=igamc\left(\frac{4}{2},\frac{2}{2}\right)=0.7358$, $P-value_{02}=igamc\left(\frac{3}{2},\frac{0}{2}\right)=1$, $P-value_{12}=igamc\left(\frac{4}{2},\frac{5}{2}\right)=0.2873$.

Вирішальне правило (для рівня значущості 1%). Якщо підраховане значення $P-value_{XY}\geq0.01$, тоді робимо висновок, що A_{XY} випадкова.

Висновки та інтерпретація результатів тесту. Так як значення $P-value_{01}$, $P-value_{02}$, $P-value_{12}$, що отримані в прикладі ≥0.01, робимо висновок, що послідовності A_{01}, A_{02} і A_{12} випадкові. Тому вважаємо, що послідовність A_{012} теж є випадковою і пройшла цей тест.

Рекомендації щодо вхідних розмірів. Кожна послідовність A_{012}, що тестується, повинна складалася як мінімум зі 150 тритів ($n_{012} \geq 150$). Зауважимо, що $M \geq 20$, $M \geq 0.01 \cdot n_{012}$.

Етап 3. Тритовий тест серій

Мета тесту. Тест спрямований на загальну кількість серій в усій послідовності A_{XY}. Під серією розуміється неперервна послідовність однакових символів X та Y. Мета тесту – визначити, чи буде загальна кількість серій з «0» і «1» послідовності A_{01}, кількість серій з «0» і «2» послідовності A_{02} та кількість серій з «1» і «2» послідовності A_{12} такою, яка очікується від випадкової послідовності.

Позначення. n_{012} – довжина вхідної послідовності тритів $A_{012} = \{0,1,2\}^{n_{012}}$, n_{01} – довжина послідовності $A_{01} = \{0,1\}^{n_{01}}$, n_{02} – довжина послідовності $A_{02} = \{0,2\}^{n_{02}}$ та n_{12} – довжина послідовності $A_{12} = \{1,2\}^{n_{12}}$.

Статистика тесту та граничний розподіл. $\mathrm{V}_n(\mathrm{obs})_{XY}$ – загальна кількість серій (тобто загальна кількість серій з X + загальна кількість серій з Y) для усіх n_{XY} символів послідовності A_{XY}. Граничний розподіл такої статистики є χ^2-розподіл.

Опис тесту. Визначаємо пропорцію π_{XY} Y у підпослідовності $A_{XY} = \left(a_1, a_2, ..., a_{n_{XY}}\right)$, $a_i = \{X, Y\}$, $i = \overline{1, n_{XY}}$: $\pi_{XY} = \frac{1}{n_{XY}} \cdot \sum_{i=1}^{n_{XY}} \varepsilon_i$, де $\varepsilon_i = 0$ – якщо $a_i = X$, $\varepsilon_i = 1$ – якщо $a_i = Y$, $i = \overline{1, n_{XY}}$.

Потім перевіряємо нерівність: $\left|\pi_{XY} - 1/2\right| \geq 2/\sqrt{n_{XY}}$. Якщо вона виконується, тоді $P - value_{XY} = 0$ і далі тест можна не виконувати, якщо нерівність не виконується – виконуємо тест далі. Підраховуємо значення статистики тесту для

послідовності A_{XY}: $V_{n_{XY}} = \sum_{i=1}^{n_{XY}-1} r(\mathrm{i}) + 1$, де $r(\mathrm{i}) = 0$ – якщо $a_i = a_{i+1}$, $r(\mathrm{i}) = 1$ – якщо $a_i \neq a_{i+1}$, $i = \overline{1, n_{XY} - 1}$.

Розраховуємо три $P-value$ для кожної підпослідовності A_{XY}:

$$P-value_{XY} = erfc\left(\frac{\left|V_{n_{XY}} - 2 \cdot n_{XY} \cdot \pi_{XY} \cdot (1-\pi_{XY})\right|}{2 \cdot \sqrt{2 \cdot n_{XY}} \cdot \pi_{XY} \cdot (1-\pi_{XY})}\right).$$

Приклад. Якщо $A_{012} = 012120001221110112110112 0$, тоді $A_{01} = 01100011110111101 10$, $n_{01} = 19$, $A_{02} = 0220002202020$, $n_{02} = 13$, $A_{12} = 121212211111211112$, $n_{12} = 18$.

Тоді $\pi_{01} = \frac{1}{19} \cdot \sum_{i=1}^{19} \varepsilon_i = 0.6315$, $\pi_{02} = \frac{1}{13} \cdot \sum_{i=1}^{13} \varepsilon_i = 0.4615$, $\pi_{12} = \frac{1}{18} \cdot \sum_{i=1}^{18} \varepsilon_i = 0.3333$. Перевіряємо нерівність $|\pi_{01} - 1/2| \geq 2/\sqrt{n_{01}}$: $|0.6315 - 0.5| = 0.1315 < 2/\sqrt{19} = 0.4588$ – можна виконувати тест далі. Перевіряємо нерівність $|\pi_{02} - 1/2| \geq 2/\sqrt{n_{02}}$: $|0.4615 - 0.5| = 0.0385 < 2/\sqrt{13} = 0.5547$ – можна виконувати тест далі. Перевіряємо нерівність $|\pi_{12} - 1/2| \geq 2/\sqrt{n_{12}}$: $|0.3333 - 0.5| = 0.1667 < 2/\sqrt{18} = 0.4714$ – можна виконувати тест далі.

Підраховуємо значення статистики тесту для послідовностей:

$V_{n_{01}} = 1+0+1+0+0+1+0+0+0+1+1+0+0+0+1+1+0+1+1 = 9$,

$V_{n_{02}} = 1+0+1+0+0+1+0+1+1+1+1+1+1 = 9$,

$V_{n_{12}} = 1+1+1+1+1+0+1+0+0+0+0+1+1+0+0+0+1+1 = 10$.

Розраховуємо три $P-value$:

$$P-value_{01} = erfc\left(\frac{\left|9 - 2 \cdot 19 \cdot 0.6315 \cdot (1-0.6315)\right|}{2 \cdot \sqrt{2 \cdot 19} \cdot 0.6315 \cdot (1-0.6315)}\right) = 0.9379,$$

$$P-value_{02} = erfc\left(\frac{\left|9 - 2 \cdot 13 \cdot 0.4615 \cdot (1-0.4615)\right|}{2 \cdot \sqrt{2 \cdot 13} \cdot 0.4615 \cdot (1-0.4615)}\right) = 0.1566,$$

$$P-value_{12}=erfc\left(\frac{\left|10-2\cdot 18\cdot 0.3333\cdot(1-0.3333)\right|}{2\cdot\sqrt{2\cdot 18}\cdot 0.3333\cdot(1-0.3333)}\right)=0.2888.$$

Вирішальне правило (для рівня значущості 1%). Якщо підраховане значення $P-value_{XY}\geq 0.01$, тоді робимо висновок, що A_{XY} випадкова.

Висновки та інтерпретація результатів тесту. Так як значення $P-value_{01}$, $P-value_{02}$, $P-value_{12}$, що отримані у прикладі ≥ 0.01, робимо висновок, що послідовності A_{01}, A_{02} і A_{12} випадкові. Тому вважаємо, що трійкова послідовність A_{012} теж є випадковою і пройшла цей тест. Зауважимо, що великі значення $\mathrm{V}_n(\mathrm{obs})_{XY}$ свідчать про дуже швидкі коливання між X та Y, малі значення $\mathrm{V}_n(\mathrm{obs})_{XY}$ свідчать про занадто повільні коливання. Швидкі коливання спостерігаються при великій кількості переходів. Послідовність з повільними коливаннями має менше серій, ніж очікується від випадкової послідовності.

Рекомендації щодо вхідних розмірів. Кожна послідовність A_{012}, що тестується, повинна складалася як мінімум зі 150 тритів ($n_{012}\geq 150$).

Етап 4. Тритовий тест найдовших серій

Мета тесту. Увага спрямована на пошук найдовшої серії «0» послідовності A_{01}, серії «1» послідовності A_{12}, серії «2» послідовності A_{02} в межах M-бітових блоків. Мета тесту – визначити, чи відповідає довжина найдовшої серії X (в послідовності A_{XY}) довжині найдовшої серії, що очікується в випадковій послідовності.

Позначення. n_{012} – довжина вхідної послідовності тритів $A_{012}=\{0,1,2\}^{n_{012}}$, n_{01} – довжина підпослідовності $A_{01}=\{0,1\}^{n_{01}}$, n_{02} – довжина підпослідовності $A_{02}=\{0,2\}^{n_{02}}$ та n_{12} – довжина підпослідовності $A_{12}=\{1,2\}^{n_{12}}$, M – довжина кожного блоку. Для M зафіксовані такі можливі значення: $\mathrm{M}=8$, $\mathrm{M}=128$ та $\mathrm{M}=10000$. Довжина блоку M обирається наступним чином: якщо довжина $n_{012}\leq 9408$ обирається $\mathrm{M}=8$, якщо довжина $9408<n_{012}\leq 1125000$ $\mathrm{M}=128$, якщо довжина $n_{012}>1125000$ обирається $\mathrm{M}=10000$.

Статистика тесту та граничний розподіл. $\chi^2(\text{obs})_{XY}$ – міра того, як добре довжина найдовшої серії X послідовності A_{XY} в межах блоків довжиною M співпадає з довжиною найдовшої серії у межах M –бітових блоків за припущенням випадковості. Граничний розподіл такої статистики є χ^2 -розподіл.

Опис тесту. 1. Спочатку ділимо вхідну підпослідовність $A_{XY}=\left(a_1,a_2,...,a_{n_{XY}}\right)$, $a_i=\{X,Y\}$, $i=\overline{1,n_{XY}}$ на $N_{XY}=\left\lfloor \frac{n_{XY}}{M} \right\rfloor$ блоків, що не перетинаються. Відкидаємо останні символи послідовності, що не утворюють повного блоку довжини M (якщо такі є).

2. Для i-го блоку ($i=\overline{1,n_{XY}}$) знаходимо довжину найдовшої серії з X - $l_{i,XY}{}^{\max}$.

Набір всіх цих довжин $\left\{l_{i,XY}{}^{\max}\right\}_{i=1}^{n_{XY}}$ розіб'ємо на класи $V_{k,XY}$ і будемо підраховувати кількість $v_{k,XY}$ елементів, що потрапили до кожного класу. В залежності від M кількість класів, і умови, за якими довжини відносяться до того чи іншого класу приведені в наступній табл. 3.1 (подібно методиці Nist STS [172, 173]):

Таблиця 3.1

Класи в залежності від параметру M

Клас $V_{k,XY}$	M = 8	M = 128	M = 10000
$V_{0,XY}$	≤ 1	≤ 4	≤ 10
$V_{1,XY}$	2	5	11
$V_{2,XY}$	3	6	12
$V_{3,XY}$	4 ≥	7	13
$V_{4,XY}$		8	14
$V_{5,XY}$		≥ 9	15
$V_{6,XY}$			≥ 16

4. Далі рахуємо статистику $\chi^2(obs)_{XY} = \sum_{i=0}^{K} \frac{\left(v_{i,XY} - N_{XY} \cdot \pi_i\right)^2}{N_{XY} \cdot \pi_i}$, де величини π_i і величина K обирається в залежності від M див. наступну табл.

Таблиця 3.2

Залежність велечини π_i і величина K від M

M	K	π_0	π_1	π_2	π_3	π_4	π_5	π_6
$M = 8$	$K = 3$	0,2148	0,3672	0,2305	0,1875			
$M = 128$	$K = 5$	0,1174	0,2430	0,2493	0,1752	0,1027	0,1124	
$M = 10000$	$K = 6$	0,0882	0,2092	0,2483	0,1933	0,1208	0,0675	0,0727

4. На останьому етапі підраховуємо значення $P - value_{XY} = igamc\left(\frac{K}{2}, \frac{\chi^2(obs)_{XY}}{2}\right)$, де $igams(.)$- неповна гамма функція, що визначається наступним чином $igamc(a,b) = \frac{1}{\Gamma(a)} \cdot \int_b^{+\infty} e^{-u} \cdot u^{a-1} du$.

Приклад. Якщо $A_{012} = 012120001221110112110120$, тоді $A_{01} = 01100011110111 10110$, $n_{01} = 19$, $A_{02} = 0220002202020$, $n_{02} = 13$, $A_{12} = 121212211111211112$, $n_{12} = 18$. Тоді $M = 8$.

1. Тоді із A_{01} утворяться 2 блоки ($N_{01} = \left\lfloor \frac{19}{8} \right\rfloor = 2$): 01100011 11011110,

із A_{02} утворяться 1 блок ($N_{02} = \left\lfloor \frac{13}{8} \right\rfloor = 1$): 02200022,

із A_{12} утворяться 4 блоки ($N_{12} = \left\lfloor \frac{18}{8} \right\rfloor = 2$): 12121221 11112111.

2. Знаходимо довжини найдовших серій.

Для послідовності A_{01}: ${l_{1,01}}^{\max} = 3$, ${l_{2,01}}^{\max} = 1$. Тоді $v_{0,01=1}$, $v_{1,01=0}$, $v_{2,01=1}$, $v_{3,01=0}$.

Для послідовності A_{02}: ${l_{1,02}}^{\max} = 3$. Тоді $v_{0,02=0}$, $v_{1,02=0}$, $v_{2,02=1}$, $v_{3,02=0}$.

Для послідовності A_{12}: ${l_{1,12}}^{\max} = 1$, ${l_{2,12}}^{\max} = 4$. Тоді $v_{0,12=1}$, $v_{1,12=0}$, $v_{2,12=0}$, $v_{3,12=1}$.

3. Рахуємо статистику.

$$\chi^2(obs)_{01} = \sum_{i=0}^{3} \frac{\left(v_{i,01} - N_{01} \cdot \pi_i\right)^2}{N_{01} \cdot \pi_i} = 2.4969,\ \chi^2(obs)_{02} = \sum_{i=0}^{3} \frac{\left(v_{i,02} - N_{02} \cdot \pi_i\right)^2}{N_{02} \cdot \pi_i} = 3.3383,$$

$$\chi^2(obs)_{12} = \sum_{i=0}^{3} \frac{\left(v_{i,12} - N_{12} \cdot \pi_i\right)^2}{N_{12} \cdot \pi_i} = 2.9944.$$

Визначаєм $P - value_{XY}$. $P - value_{01} = igamc\left(\frac{3}{2}, \frac{\chi^2(obs)_{01}}{2}\right) = 0.4758$,

$$P - value_{02} = igamc\left(\frac{3}{2}, \frac{\chi^2(obs)_{02}}{2}\right) = 0.3423,$$

$$P - value_{12} = igamc\left(\frac{3}{2}, \frac{\chi^2(obs)_{12}}{2}\right) = 0.3924.$$

Вирішуюче правило (для рівня значущості 1%). Якщо підраховане значення $P - value_{XY} \geq 0.01$, тоді робимо висновок, що A_{XY} випадкова.

Висновки та інтерпретація результатів тесту. Так як значення $P - value_{01}$, $P - value_{02}$, $P - value_{12}$, що отримані в прикладі ≥ 0.01, робимо висновок, що підпослідовності A_{01}, A_{02} і A_{12} випадкові. Тому рахуємо, що трійкова послідовність A_{012} теж випадкова і пройшла даний тест.

Рекомендації по вхідним розмірам. Кожна послідовність A_{012}, що тестується повинна складалася як мінімум зі 9408.

Етап 5. Тест на співпадіння з шаблоном без перекриття.

Мета тесту. Увага тесту зосереджена на тому, яку кількість разів зустрічається наперед заданий рядок у вхідній послідовності. Мета тесту – виявити генератор, який формує послідовність, що містить дуже велику кількість заданого аперіодичного шаблону. В цьому тесті, для пошуку заданого m-тритного шаблону використовується m-тритне вікно. Вікно зсувається вправо на один трит, якщо під ним не спостерігається заданий шаблон, і зсувається вправо на m біт, якщо послідовність, що в ньому знаходиться співпадає з шаблоном.

Позначення: n_{012} – довжина вхідної послідовності тритів $A_{012} = \{0,1,2\}^{n_{012}}$, n_{01} – довжина послідовності $A_{01} = \{0,1\}^{n_{01}}$, n_{02} – довжина послідовності $A_{02} = \{0,2\}^{n_{02}}$ та n_{12} – довжина послідовності $A_{12} = \{1,2\}^{n_{12}}$;

Статистика тесту та граничний розподіл. $\chi^2(obs)_{XY}$ величина наскільки кількість «співпадінь» з шаблоном у вхідній послідовності відповідає очікуваній кількості «співпадінь» за умови випадковості послідовності.

Опис тесту. Розділяємо вхідну послідовність відповідно загальній методиці. Отримані послідовності розбиваються на *N* блоків довжини *M.*

$W_{i_{XY}}$ – кількість разів, яку шаблон B_{XY} зустрічається у відповідному блоці блоці. Пошук проводиться за допомогою m-тритного вікна, що пересувається вздовж послідовності A_{XY}. Якщо виділена вікном підпослідовність тритів співпадає з шаблоном B_{XY}, тоді зсуваємо вікно на m трит праворуч; якщо ж не співпадає – на один трит праворуч.

Приклад. $A_{01} = 01100011110111101 10$ по 2 блоки, $B_{01} = 111$ та $A_{12} = 121212211111211112$ по 2 блоки, $B_{12} = 112$.

Таблиця 3.3

Приклад проведення тесту

$A_{01} = 01100011110111101 10$				
Розташування вікна	011000111		101111011	
	Трити	W_1	Трити	W_2
1-3	011	0	101	0
2-4	110	0	011	0
3-5	100	0	111	1
4-6	000	0	111 не розгляд.	
5-7	001	0	110 не розгляд.	1
6-8	011	0	101	1
7-9	111	1	011	1

Продовження табл. 3.3

A_{12} =121212211111211112				
Розташування вікна	121212211		111211112	
	Трити	W_1	Трити	W_2
1-3	121	0	111	0
2-4	212	0	112	1
3-5	121	0	121 не розгляд.	
4-6	212	0	211 не розгляд.	
5-7	122	0	111	1
6-8	221	0	111	1
7-9	211	0	112	2

Отже отримуємо: для послідовності A_{01} – $W_1 = 1$, $W_2 = 1$, а для A_{12} – $W_1 = 0$, $W_2 = 2$. За умови випадковості, підраховуємо теоретичне середнє μ та дисперсію $\sigma^2_{XY} : \mu_{XY} = \dfrac{M_{XY} - m_{XY} + 1}{2^{m_{XY}}}$ $\sigma^2_{XY} = M_{XY} \cdot \left(\dfrac{1}{2^{m_{XY}}} - \dfrac{2 \cdot m_{XY} - 1}{2^{2 \cdot m_{XY}}} \right)$

$$\chi^2(obs)_{XY} = \sum_{j=1}^{N_{XY}} \frac{\left(W_{j_{XY}} - \mu_{XY}\right)}{\sigma^2_{XY}} \quad P - value_{XY} = igamc\left(\frac{N_{XY}}{2}, \frac{\chi^2(\text{obs})_{XY}}{2} \right)$$

Висновки та інтерпретація результатів тесту. Так як значення $P - value_{01}$, $P - value_{02}$, $P - value_{12}$, що отримані в прикладі ≥ 0.01, робимо висновок, що послідовності A_{01}, A_{02} і A_{12} випадкові. Тому вважаємо, що послідовність A_{012} теж є випадковою і пройшла цей тест.

Якщо $P - value_{01}$, $P - value_{02}$, $P - value_{12}$, дуже мале (<0,01) тоді вхідна послідовність містить таку кількість шаблонних підпослідовностей, що не узгоджується з гіпотезою про випадковість

Рекомендації щодо вхідних розмірів. Кожна послідовність A_{012}, що тестується, повинна складалася як мінімум зі 150 тритів ($n_{012} \geq 150$). Зауважимо, що $M \geq 20$, $M \geq 0.01 \cdot n_{012}$. Рекомендовано використовувати як мінімум m = 9 або m = 10 для отримання значущих результатів. M та N мають бути обраними так, щоб $M > 0{,}01 \cdot n \; та \; N = \left\lfloor \dfrac{n}{M} \right\rfloor$.

Оскільки, пошук аперіодичного шаблону відбувається у декілька етапів і проводиться пошук у багатьох блоках, то фактично цей тест розбиваються на більшу кількість раз і послідовності згідно цього тесту оцінюються багато раз.

Етап 6. Тест шаблонів з перекриттям

Мета тесту. Аналізується кількість раз при якій шаблон зустрічається у вхідній послідовності. Для пошуку заданого m-тритного шаблону використовується m-тритне вікно. Якщо виділений вікном підрядок загальної послідовності не співпадає з заданим шаблоном, вікно зсувається на один трит праворуч. На відміну від попереднього тесту, якщо навіть цей підрядок співпадає з шаблонним, вікно все одно зсувається на один трит праворуч. Після зсуву вікна пошук продовжується аналогічно.

Позначення: n_{012} – довжина вхідної послідовності тритів $A_{012} = \{0,1,2\}^{n_{012}}$, n_{01} – довжина послідовності $A_{01} = \{0,1\}^{n_{01}}$, n_{02} – довжина послідовності $A_{02} = \{0,2\}^{n_{02}}$ та n_{12} – довжина послідовності $A_{12} = \{1,2\}^{n_{12}}$; B_{01} – шаблон для послідовності $A_{01} = \{0,1\}^{n_{01}}$, B_{02} – шаблон для послідовності $A_{02} = \{0,2\}^{n_{02}}$, B_{12} – шаблон для послідовності $A_{12} = \{1,2\}^{n_{12}}$; m_{XY} – довжина шаблонного рядка у тритах, K – кількість степенів свободи; N_{XY} – кількість незалежних блоків, на які поділяються послідовності A_{XY}; M_{XY} – довжина кожного такого блоку.

Статистика тесту та граничний розподіл. $\chi^2(\text{obs})_{XY}$ – міра того, як добре кількість, яку заданий шаблон зустрічається у послідовності A_{XY} відповідає цій кількості, але за умови випадковості послідовності. Граничний розподіл такої статистики є χ^2 -розподіл.

Опис тесту. Поділяємо вхідну послідовність на послідовності A_{01}, A_{02}, A_{12}, які в свою чергу одразу поділяємо на N_{XY} блоків довжиною M_{XY}. При цьому останні біти послідовності, що не утворюють цілий блок відкидаються.

Підраховуємо кількість, яку зустрічається шаблон у кожному з N блоків. Для кожного блоку кількість шукається окремо. Перед аналізом нового блоку

лічильник встановлюємо в нуль. При пошуку шаблона створюється m-тритне вікно, що пересувається вздовж вхідної послідовності. Якщо виділена вікном послідовність трит співпадає з шаблоном, тоді лічильник збільшуємо на одиницю. Не залежно від результату порівняння, вікно зсувається на один трит праворуч. В залежності від значення лічильника після проходження блоку збільшуємо на одиницю одну з величин.

Підраховуємо значення λ і η, які будуть використовуватися при визначення теоретичних ймовірностей і π, що відповідають класам і ν : $\lambda_{XY} = \frac{M_{XY} - m_{XY} + 1}{2^{m_{XY}}}$, $\eta_{XY} = \frac{\lambda_{XY}}{2}$. Підраховуємо статистику $\chi^2(\text{obs})_{XY} = \sum_{i=0}^{K_{XY}} \frac{\left(\nu_{i_{XY}} - N_{XY} \cdot \pi_{i_{XY}}\right)}{N_{XY} \cdot \pi_{i_{XY}}}$ $\pi_{1_{01}} =$, $\pi_{2_{01}} =, \pi_{3_{01}} =$. Підрахуємо величини $P - value_{XY} = igamc\left(\frac{K_{XY}}{2}, \frac{\chi^2(\text{obs})_{XY}}{2}\right)$

Приклад. Якщо $A_{012} = 012120001221110111211011 20$, тоді $A_{01} = 01100011110111101 10$, $n_{01} = 19$, $A_{02} = 0220002202020$, $n_{02} = 13$, $A_{12} = 121212211111211112$, $n_{12} = 18$. Приклад результатів тесту в табл. 3.4.

Таблиця 3.4

Приклад результатів тесту

$A_{01} = 0110011111011110110$		
$B_{01} = 111$, $n_{01} = 19$, $M_{01} = 5$, $N_{01} = 3$, $K = 2$		
1 блок *01100*		
Позиц.	***Трит***	***Кільк.***
1-3	*011*	*0*
2-4	*110*	*0*
3-5	*100*	*0*
Разів зустрічей шаблону		*0*

$A_{02} = 0220002202020$		
$B_{02} = 200, n_{02} = 13\ M_{02} = 5$, $N_{02} = 2$, $K = 2$		
1 блок 02200		
Позиц.	***Трит***	***Кільк.***
1-3	*022*	*0*
2-4	*220*	*0*
3-5	*200*	*1*
Разів зустрічей шаблону		*1*

$A_{12} = 121212211111211112$		
$B_{12} = 112, n_{12} = 18, M_{12} = 5$, $N_{12} = 3$, $K = 2$		
1 блок 12121		
Позиц.	***Трит***	***Кільк.***
1-3	*121*	*0*
2-4	*212*	*0*
3-5	*121*	*0*
Разів зустрічей шаблону		*0*

Продовження табл. 3.4

2 блок *01111*		
Позиц.	***Трит***	***Кільк.***
1-3	*011*	*0*
2-4	*111*	*1*
3-5	*111*	*2*
Разів зустрічей шаблону		*2*
3 блок *01111*		
Позиц.	***Трит***	***Кільк.***
1-3	*011*	*0*
2-4	*111*	*1*
3-5	*111*	*2*
Разів зустрічей шаблону		*2*

2 блок 02202		
Позиц.	***Трит***	***Кільк.***
1-3	*022*	*0*
2-4	*220*	*0*
3-5	*202*	*0*
Разів зустрічей шаблону		*0*

2 блок 22111		
Позиц.	***Трит***	***Кільк.***
1-3	*221*	*0*
2-4	*211*	*0*
3-5	*111*	*0*
Разів зустрічей шаблону		*0*
3 блок 11211		
Позиц.	***Трит***	***Кільк.***
1-3	*112*	*0*
2-4	*121*	*0*
3-5	*211*	*0*
Разів зустрічей шаблону		*0*

Висновки та інтерпретація результатів тесту. Так як значення $P-value_{01}$, $P-value_{02}$, $P-value_{12}$, що отримані в прикладі ≥ 0.01, робимо висновок, що послідовності A_{01}, A_{02} і A_{12} випадкові. Тому вважаємо, що послідовність A_{012} теж є випадковою і пройшла цей тест. Зауважимо, що для одного шаблону (наприклад, B = 111), якщо послідовність має дуже багато серій з одиниць, тоді: 1) ν буде дуже велике 2) тестова статистика буде великою 3) $P-value_{01}$, $P-value_{02}$, $P-value_{12}$ буде малим (< 0.01) 4) буде зроблено висновок про не випадковість послідовності.

Рекомендації щодо вхідних розмірів. Кожна послідовність A_{012}, що тестується, повинна складалася як мінімум зі 150 тритів ($n_{012} \geq 150$). Зауважимо, що $M \geq 20$, $M \geq 0.01 \cdot n_{012}$. Так само як і попередній тест відбувається розбиття тесту на частини.

3.4. Висновки до третього розділу

Таким чином, у третьому розділі дисертації були отримані наступні основні **результати:**

1. Розроблено метод генерування псевдовипадкових послідовностей, який, за рахунок виконання нової послідовності операцій (підстановок $Sbox(X)$, лінійного розсіювання $Mix(X)$, динамічного циклічного зсуву і додавання за модулем $3\oplus$ та 3^l+) над вектором внутрішніх станів V_p ($V_p=\{0,1,2\}^p$, $p=14\cdot l$) за $r\cdot b$ циклів, дозволяє формувати трійкові незбалансовані («0», «1», «2») псевдовипадкові послідовності $V_{m\cdot b}$, $m=4\cdot l$, що можуть використовуватись для реалізації запропонованого метода забезпечення стійкості кутритових протоколів квантової криптографії до некогерентних атак, а також для інших криптографічних застосувань в сучасних інформаційно-комунікаційних технологіях.

2. Розроблено метод оцінювання якості псевдовипадкових послідовностей, який, за рахунок комплексної інтерпретації згенерованих чисел, введення диференційованих ймовірностей $P-value_{01}$, $P-value_{02}$, $P-value_{12}$ і введення трійкових коефіцієнтів для функції помилок *erfc* та неповної гамма функції *igamc,* дає принципову можливість оцінювати загальноприйняті статистичні параметри та закономірності для тритових псевдовипадкових послідовностей і, відповідно, оцінювати криптостійкість тритових генераторів псевдовипадкових послідовностей та доцільність їх використання для криптографічних застосувань.

РОЗДІЛ 4

ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ЗАПРОПОНОВАНИХ МЕТОДІВ

4.1. Методика проведення експерименту

Метою експериментального дослідження є вивчення якостей оцінюваних об’єктів, перевірка правильності формування та достовірності гіпотез, глибоке вивчення досліджуваної наукової тематики [63]. В залежності від галузі науки, структури об’єктів досліджень, мети дослідження, організаційних явищ та ін., існує багато різних класифікації експериментів [64]. Правильний вибір методики займає особливе значення при проведенні експериментів. У методиці визначається послідовність процесів, у результаті якої досягається мета дослідження [96].

Перший крок у проведенні експериментального дослідження займає складання плану – програми дослідження:

1. Мета та задачі експерименту.

Мета експерименту – дослідити ефективність розроблених методів та перевірити їх адекватність.

Задачі:

1.1. Дослідити запропонований метод забезпечення стійкості кутритових протоколів КК до некогерентних атак (провести моделювання роботи методу).

1.2. Дослідити запропонований та відомий метод генерування тритових ПВП за методикою NIST STS.

1.3. Дослідити запропонований та відомий метод генерування тритових ПВП за розробленим методом оцінювання якості тритових послідовностей.

1.4. Перевірити адекватність роботи розробленого метода оцінювання якості тритових послідовностей.

2. Вибір вхідних та вихідних параметрів:

2.1. Вхідними параметрами для вирішення задачі 1.1 є: швидкість генерування тритових послідовностей (V_{gen}), швидкості передачі тритових послідовностей по квантовому каналі (V_{kv}), швидкості передачі тритових послідовностей по класичному каналі (V_{kl}), швидкість виконання арифметичних операцій в полі $GF(3)$ (V_x), розмір блоку даних (r), кількість блоків даних (l), частота перемикання в режим підслуховування (q), відомий та запропоновані методи забезпечення стійкості кутритових протоколів КК, розмір кроків зміни кожного параметру. Вихідні параметри: зібрана статистика швидкостей роботи відомого та запропонованого методів забезпечення стійкості кутритових протоколів КК в залежності від вхідних параметрів.

2.2. Вхідні параметри для вирішення задачі 1.2: згенеровані послідовності (послідовності переводяться із трійкового у двійковий вид) генератором TritGen та стандартним генератором ПВП C++ (середовище Microsoft Visual Studio 2012). Вихідні параметри: результати проходження за методикою NIST STS [172, 173].

2.3. Вхідні параметри для вирішення задачі 1.3: згенеровані тритові послідовності генератором TritGen та стандартним генератором ПВП C++ (середовище Microsoft Visual Studio 2012). Вихідні параметри: результати проходження за розробленим методом оцінювання якості тритових послідовностей, який реалізований у вигляді консольного застосунку TritSTS.

2.4. Вхідні параметри для вирішення задачі 1.4: підібрані тритові послідовності, що дозволять перевірити адекватність розробленого метода оцінки якості тритових послідовностей. Вихідні параметри: зведені результати дослідження підібраних послідовностей.

3. Послідовність дій:

3.1. Фіксуються базові параметри системи: швидкість генерування тритових послідовностей (V_{gen}), швидкості передачі тритових послідовностей по квантовому каналі (V_{kv}), швидкості передачі тритових послідовностей по класичному каналі (V_{kl}), швидкість виконання арифметичних операцій в полі $GF(3)$

(V_x), розмір блоку даних (r), кількість блоків даних (l), частота перемикання в режим підслуховування (q). Далі імітується виконання усіх етапів квантового протоколу за допомогою розробленого ПЗ «Model of QSDC protocol». Зібрану статистику використаємо для аналізу ефективності запропонованого методу забезпечення стійкості кутритових протоколів КК.

3.2. Згенерувати по п’ять послідовностей генераторами (генератором TritGen та стандартним генератором ПВП C++), конвертувати трити у біти, після чого програмним комплексом NIST STS перевірити їх на псевдовипадковість.

3.3. Згенерувати по п’ять послідовностей генераторами (генератором TritGen та стандартним генератором ПВП C++), після чого розробленим консольним застосунком TritSTS перевірити їх на псевдовипадковість.

3.4. Спочатку обрати завідомо не випадкові послідовності та перевірити їх програмним застосунком TritSTS. 2. Обрати завідомо випадкові послідовностей і також перевірити застосунком TritSTS. 3. Оцінити результати перевірки.

4. Вибір кроку зміни чинників:

4.1. Зміна r від 4 до 100 (з кроком 4). Зміна швидкостей роботи протоколів КК (V_{gen}, V_{kv}, V_{kl} та V_x) із 1000 до 100000.

5. Програмні засоби, що використовувались: Visual Studio 2012 (для створення програмних засобів, що дозволяють провести дослідження), NIST_STS_console (для дослідження ПВП), розроблений програмний засіб Trigen для генерації тритових послідовностей запропонованим алгоритмом Trigen та стандартним генератором ПВП C++, розроблений програмний засіб TritSTS, що використовувався для оцінки якості тритових послідовностей та Model of QSDC protocol, для оцінки ефективності запропонованого методу забезпечення стійкості кутритових протоколів КК.

6. Аналіз результатів.

Другий крок, що здійснюється після затвердження методики, – це визначення об’єму експериментальних досліджень та необхідних програмних засобів. Третім кроком є безпосереднє проведення експерименту. Четвертим кроком є обробка експериментальних даних, систематизація усіх числових даних, пере-

вірка зведення до єдиної системи одиниць, побудова графіків залежностей, таблиць, діаграм. Відповідно до обраної методики проведено експериментальні дослідження, опис та обробка яких містяться в п. 4.2-4.3.

4.2. Верифікація методу забезпечення стійкості протоколів КК

Для дослідження запропонованого у другому розділі методу забезпечення стійкості протоколів КПБЗ було виконане порівняння його швидкодії із вже існуючим не квантовим методом забезпечення стійкості [8, 10].

Нехай потрібно передавати протоколом КПБЗ (з використанням запропонованого та відомого методів забезпечення стійкості) повідомлення $A \in V_n$ ($V_n = \{0,1,2\}^n$, $n = r \cdot l$, $r \in N$ − розмір блоку даних, а $l \in N$ − кількість таких блоків). Для порівняння швидкодії передачі повідомлення A протоколом КПБЗ (з частотою перемикання в режим підслуховування q) буде оцінений час виконання кожного конкретного його етапу. Для оцінки часу виконання кожного етапу було введено такі позначення: V_{gen} − швидкість генерування тритових послідовностей; V_{kv} та V_{kl} − швидкості передачі тритових послідовностей по квантовому та класичному каналі відповідно; V_x − швидкість виконання арифметичних операцій в полі $GF(3)$.

Розглянемо спочатку відомий метод забезпечення стійкості [8, 10]. Передача повідомлення A від Аліси до Боба виконується у шість етапів:

Етап 1. Аліса генерує l матриць M_i розміром $r \times r$ трит, $i \in \overline{1,l}$ використовуючи для цього процедуру генерації тритових послідовностей F_{gen} та секретний параметр K (використовується тільки на етапі генерації матриць): $M_i = F_{gen}\left(K, i, r^2\right)$. Час виконання даної операції буде залежати від швидкості генерації тритових послідовностей V_{gen} із яких формуються матриці, їх кількості l та розмірів $r \times r$. Отже $t_1 = \dfrac{l \cdot r^2}{V_{gen}}$ с. Зауважимо, що при оцінці часу генерації

матриць рахуємо, що усі l матриць M_i, $i \in \overline{1,l}$ із першого разу генерувались невиродженими.

Етап 2. Аліса перемножує блоки даних A_i ($A = (A_1,...,A_l)$, $i \in \overline{1,l}$) розміром r трит із отриманими матрицями M_i розміром $r \times r$ трит: $B_i = A_i \cdot M_i$. Час виконання даної операції буде залежати від швидкості виконання арифметичних операцій в полі $GF(3)$ V_x, кількості блоків даних l та їх розміру r. Тоді $t_2 = \dfrac{l \cdot (2r^2 - r)}{V_x}$ с.

Етап 3. Відбувається передача повідомлення B квантовим каналом з використанням протоколів КПБЗ від Аліси до Боба, при цьому з частотою q відбувається перемикання в режим контролю підслуховування для детектування Єви: $B_i' = F_{kv}(B_i, q)$. Час виконання даної операції буде залежати від швидкості передачі тритових послідовностей по квантовому каналі V_{kv}, кількості блоків даних l, їх розміру r та частоти перемикання в режим підслуховування q. Отже $t_3 = \left(\dfrac{l \cdot r}{V_{kv}}\right) \cdot (1+q)$ с.

Етап 4. Якщо на етапі 3 Аліса і Боб не виявили Єву відбувається передача відкритим каналом від Аліси до Боба матриць M_i $i \in \overline{1,l}$: $M_i' = F_{kl}(M_i)$. Час виконання даної операції буде залежати від швидкості передачі тритових послідовностей по класичному каналі V_{kl}, кількості матриць l та їх розміру $r \times r$. Тоді $t_4 = \dfrac{l \cdot r^2}{V_{kl}}$ с.

Етап 5. Боб обертає отримані у 4 етапі матриці M_i, $i \in \overline{1,l}$: $(M_i')^{-1} = F_{obr}(M_i')$. Час виконання даної операції буде залежати від швидкості виконання арифметичних операцій в полі $GF(3)$ V_x, кількості таких матриць l та їх розміру $r \times r$. Отже $t_5 = \dfrac{l \cdot (4r^3 - 4r^2)}{V_x}$ с.

Етап 6. Боб перемножує отримані блоки B_i ($B=(B_1,...,B_l)$, $i\in\overline{1,l}$) розміром r трит із отриманими оберненими матрицями $(M_i)^{-1}$ розміром $r\times r$ трит та відновлює початкове повідомлення: $A_i'=B_i'\cdot(M_i')^{-1}$. Час виконання даної операції буде залежати від швидкості виконання арифметичних операцій в полі $GF(3)$ V_x, кількості блоків даних l та їх розміру r. Тоді $t_6=\dfrac{l\cdot(2r^2-r)}{V_x}$ с.

Тоді швидкість передачі повідомлення A протоколом КПБЗ з використанням відомого методу забезпечення стійкості: $V=\dfrac{r\cdot l}{t}$ (трит/с), де − загальний час роботи протоколу КПБЗ, $t=\sum_{i=1}^{6}t_i$, t_i − час виконання i-го етапу, $i=\overline{1,6}$.

Тепер розглянемо запропонований у розділі 2 метод забезпечення стійкості [22, 62, 86]. Передача повідомлення A від Аліси до Боба виконується у вісім етапів (для спрощення опису деякі етапи, що були описані у розділі 2 були об’єднані):

Етап 1. Аліса генерує l блоків k_i розміром r трит, $i\in\overline{1,l}$ використовуючи для цього процедуру генерації тритових послідовностей F_{gen} та секретний параметр K: $k_i=F_{gen}(K,i,r)$. Час виконання даної операції буде залежати від швидкості генерації тритових послідовностей V_{gen}, кількості l та розмірів r блоків k_i, $i\in\overline{1,l}$. Отже $t_1=\dfrac{l\cdot r}{V_{gen}}$ с.

Етап 2. Аліса потритово складає блоки даних A_i ($A=(A_1,...,A_l)$, $i\in\overline{1,l}$) із блоками k_i: $B_i=A_i+k_i$. Час виконання даної операції буде залежати від швидкості виконання арифметичних операцій в полі $Gf(3)$ V_x, кількості блоків даних l та їх розміру r. Тоді $t_2=\dfrac{l\cdot r}{V_x}$ с.

Етап 3. Аліса обчислює хеш-код повідомлення B та перетворює його асиметричною функцією перетворення F_{aka}^{enc} з використанням відкритого секре-

тного параметру Боба: $H = F_{hf}(B)$, $J = F_{aka}^{enc}\left(H, K_{op}^{B}\right)$. Час виконання даної операції буде залежати від швидкості виконання арифметичних операцій в полі $GF(3)$ V_x, кількості блоків даних l та їх розміру r. Отже $t_3 = \dfrac{4 \cdot l \cdot r}{V_x}$ с.

Етап 4. Відбувається передача повідомлення (B, J) квантовим каналом з використанням протоколів КПБЗ від Аліси до Боба, при цьому з частотою q відбувається перемикання в режим контролю підслуховування для детектування Єви: $B_i' = F_{kv}(B_i, q)$, $J' = F_{kv}(J, q)$. Час виконання даної операції буде залежати від швидкості передачі тритових послідовностей по квантовому каналі V_{kv}, кількості блоків даних l, їх розміру r, частоти перемикання в режим підслуховування q та довжини зашифрованого хеш-коду J (для простоти обрали його розміром 96 трит). Тоді $t_4 = \left(\dfrac{l \cdot r + 96}{V_{kv}}\right) \cdot (1 + q)$ с.

Етап 5. Боб розраховує нове значення хеш коду H' повідомлення B' та виконує зворотне перетворення хеш коду H'' асиметричним шифром F_{aka}^{dec} з використанням свого закритого секретного параметру: $H' = F_{hf}(B')$ і $H'' = F_{aka}^{dec}\left(J', K_{cl}^{B}\right)$. Час виконання даної операції буде залежати від швидкості виконання арифметичних операцій в полі $GF(3)$ V_x, кількості блоків даних l та їх розміру r. Отже $t_5 = \dfrac{4 \cdot l \cdot r}{V_x}$ с.

Етап 6. Якщо на етапі 4 Аліса і Боб не виявили Єву і на етапі 5 Боб отримав $H' = H''$, то відбувається передача відкритим каналом від Аліси до Боба секретного параметра K: $K' = F_{kl}(K)$. Час виконання даної операції буде залежати від швидкості передачі тритових послідовностей по класичному каналі V_{kl} та розміру K (для простоти обрали його розмір − 96 трит). Тоді $t_6 = \dfrac{96}{V_{kl}}$ с.

Етап 7. Боб генерує таку ж послідовність k_i розміром r трит, яку генерувала Аліса на етапі 1, $i \in \overline{1,l}$, використовуючи для цього процедуру генерації тритових послідовностей F_{gen} та секретний параметр K, що отримав на етапі 6: $k_i' = F_{gen}(K', i, r)$. Час виконання даної операції буде залежати від швидкості генерації тритових послідовностей V_{gen}, кількості l та розмірів r блоків k_i, $i \in \overline{1,l}$. Отже $t_7 = \dfrac{l \cdot r}{V_{gen}}$ с.

Етап 8. Боб потритово віднімає від блоків даних B_i ($B = (B_1, ..., B_l)$, $i \in \overline{1,l}$) блоки k_i та відновлює початкове повідомлення A: $A_i' = B_i' - k_i'$. Час виконання даної операції буде залежати від швидкості виконання арифметичних операцій в полі $GF(3)$ V_x, кількості блоків даних l та їх розміру r. Тоді $t_8 = \dfrac{l \cdot r}{V_x}$ с.

Тоді швидкість передачі повідомлення A протоколом КПБЗ з використанням запропонованого методу забезпечення стійкості: $V = \dfrac{r \cdot l}{t}$ (трит/с), де – загальний час роботи протоколу КПБЗ, $t = \sum_{i=1}^{8} t_i$, t_i – час виконання i-го етапу, $i = \overline{1,8}$.

Для наглядності, у табл. 4.1 наведено основні етапи протоколу КПБЗ із різними методами забезпечення стійкості (відомого та запропонованого) та час їх виконання.

На основі наведених вище розрахунків на мові програмування C++ в середовищі Microsoft Visual Studio 2012 розроблено програмне забезпечення Model of QSDC protocol (на рис. 4.1 наведено вікно програми, програмний код даної програми наведено у додатку Б), що фактично дозволяє провести моделювання запропонованого та відомого методу забезпечення стійкості протоколів КПБЗ та розрахувати їх швидкодію.

Таблиця 4.1

Оцінка часу виконання етапів протоколу КПБЗ

№ Ет.	*Відомий метод*		*Запропонований метод*	
	Операція	Час виконання, *с*	Операція	Час виконання, *с*
1	$M_i = F_{gen}\left(K, i, r^2\right)$	$\frac{l \cdot r^2}{V_{gen}}$	$k_i = F_{gen}\left(K, i, r\right)$	$\frac{l \cdot r}{V_{gen}}$
2	$B_i = A_i \cdot M_i$	$\frac{l \cdot (2r^2 - r)}{V_x}$	$B_i = A_i + k_i$	$\frac{l \cdot r}{V_x}$
3	$B_i' = F_{kv}\left(B_i, q\right)$	$\left(\frac{l \cdot r}{V_{kv}}\right) \cdot (1+q)$	$H = F_{hf}\left(B\right)$ $J = F_{aka}^{enc}\left(H, K_{op}^{B}\right)$	$\frac{4 \cdot l \cdot r}{V_x}$
4	$M_i' = F_{kl}\left(M_i\right)$	$\frac{l \cdot r^2}{V_{kl}}$	$B_i' = F_{kv}\left(B_i, q\right)$ $J' = F_{kv}\left(J, q\right)$	$\left(\frac{l \cdot r + 96}{V_{kv}}\right) \cdot (1+q)$
5	$\left(M_i'\right)^{-1} = F_{obr}\left(M_i'\right)$	$\frac{l \cdot (4r^3 - 4r^2)}{V_x}$	$H' = F_{hf}\left(B'\right)$ $H'' = F_{aka}^{dec}\left(J', K_{cl}^{B}\right)$	$\frac{4 \cdot l \cdot r}{V_x}$
6	$A_i' = B_i' \cdot \left(M_i'\right)^{-1}$	$\frac{l \cdot (2r^2 - r)}{V_x}$	$K' = F_{kl}\left(K\right)$	$\frac{96}{V_{kl}}$
7	-	0	$k_i' = F_{gen}\left(K', i, r\right)$	$\frac{l \cdot r}{V_{gen}}$
8	-	0	$A_i' = B_i' - k_i'$	$\frac{l \cdot r}{V_x}$

Для запуску програми потрібно потрібно виконати наступне:

1. Вказати швидкість генерування тритових послідовностей (V_{gen}).

2. Заповнити швидкість передачі тритових послідовностей по квантовому каналі (V_{kv}).

3. Визначити швидкість передачі тритових послідовностей по класичному каналі (V_{kl}).

4. Вказати швидкість виконання арифметичних операцій в полі $GF(3)$ (V_x).

5. Вказати крок зміни розміру блоку даних (r).

6. Визначити кількість блоків даних (l)

7. Заповнити ймовірність перемикання в режим контролю підслуховування (q).

8. Вказати каталог, куди будуть збережені результати.

9. Запустити програму на виконання.

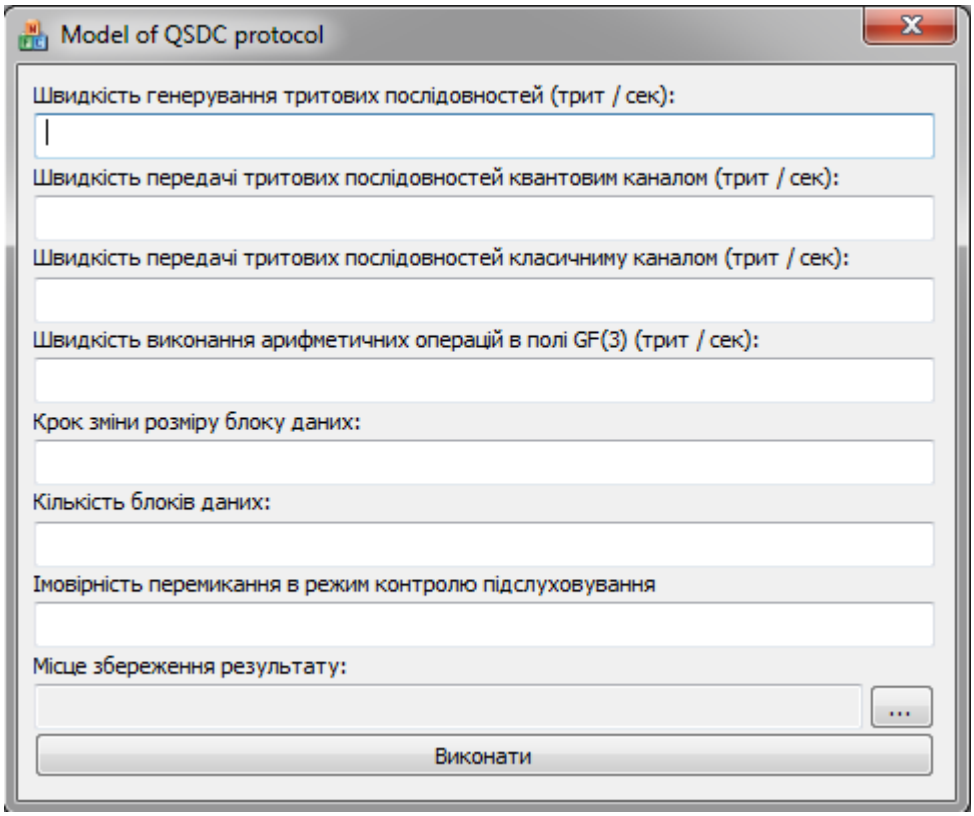

Рис. 4.1. Вікно програми Model of QSDC protocol

Для дослідження швидкодії запропонованого та відомого методів забезпечення стійкості протоколів КПБЗ було проведено 7 експериментів при різних параметрах r, l, q, V_{gen}, V_{kv}, V_{kl} та V_x.

Експеримент 1. $V_x = V_{kl} = 10^6$ трит/сек, $V_{gen} = 10^4$ трит/сек, $V_{kv} = 10^3$ трит/сек, $l = 1000$, $q = 0{,}5$ – для відомого методу забезпечення стійкості прото-

колів КПБЗ, $q = 0,05$ для запропонованого методу. Як зазначалось у другому розділі, ймовірність перемикання у режим контролю підслуховування для запропонованого методу може бути зменшена до мінімуму (із рекомендованого значення 0,5 до 0,05).

На рис. 4.2. наведено результати експерименту 1 із порівняння швидкодії протоколу КПБЗ для різних методів забезпечення його стійкості.

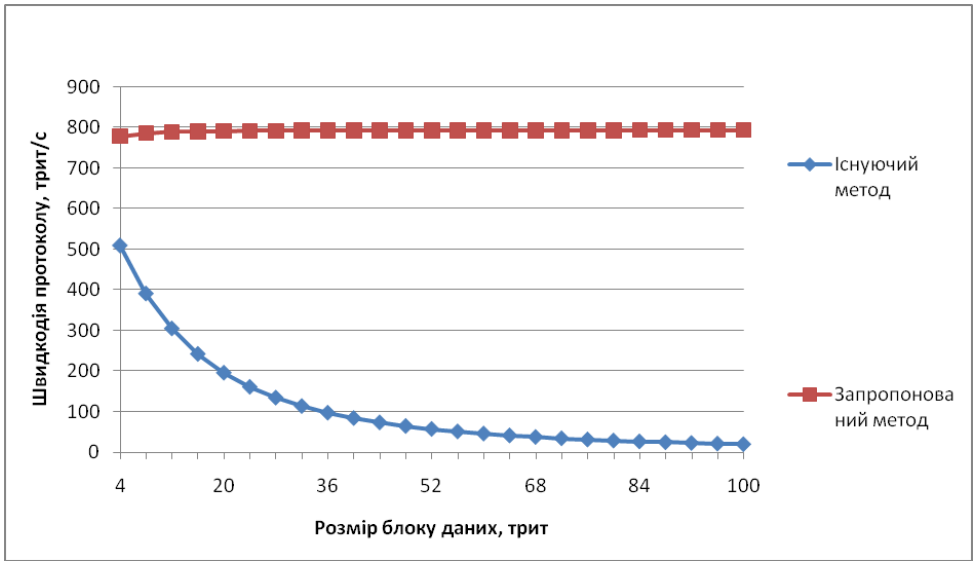

Рис. 4.2. Порівняння швидкісних характеристик протоколу КПБЗ (експеримент 1)

Згідно результатів експерименту, швидкість протоколу КПБЗ із запропонованим методом забезпечення стійкості мінімум у 1,52 раз краща за швидкість відомого методу (для $r = 4$). Причому при збільшенні r покращення швидкодії буде ще більш показовим. Наприклад, при $r = 20$ швидкодія запропонованого методу краща у 4,4 рази.

Експеримент 2. $V_x = V_{kl} = 10^5$ трит/сек, $V_{gen} = 10^4$ трит/сек, $V_{kv} = 10^3$ трит/сек, $l = 1000$, $q = 0,5$ – для відомого методу забезпечення стійкості протоколів КПБЗ, $q = 0,05$ для запропонованого методу.

На рис. 4.3. наведено результати експерименту 2 із порівняння швидкодії протоколу КПБЗ для різних методів забезпечення його стійкості.

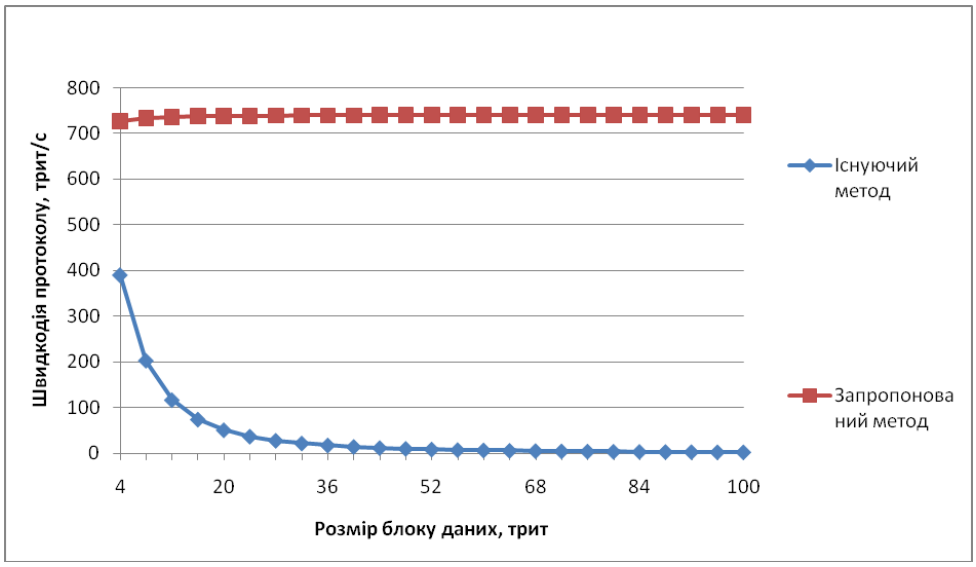

Рис. 4.3. Порівняння швидкісних характеристик протоколу КПБЗ (експеримент 2)

Згідно результатів експерименту, швидкість протоколу КПБЗ із запропонованим методом забезпечення стійкості мінімум у 1,86 раз краща за швидкість відомого методу (для $r = 4$). Причому при збільшенні r покращення швидкодії буде ще більш показовим. Наприклад, при $r = 20$ швидкодія запропонованого методу краща у 14,5 рази.

Експеримент 3. $V_x = V_{kl} = V_{gen} = 10^5$ трит/сек, $V_{kv} = 10^3$ трит/сек, $l = 1000$, $q = 0,5$ – для відомого методу забезпечення стійкості протоколів КПБЗ, $q = 0,05$ для запропонованого методу.

На рис. 4.4. наведено результати експерименту 3 із порівняння швидкодії протоколу КПБЗ для різних методів забезпечення його стійкості.

Згідно результатів експерименту, швидкість протоколу КПБЗ із запропонованим методом забезпечення стійкості мінімум у 1,84 раз краща за швидкість відомого методу (для $r = 4$). Причому при збільшенні r покращення швидкодії буде ще більш показовим. Наприклад, при $r = 20$ швидкодія запропонованого методу краща у 15,21 рази.

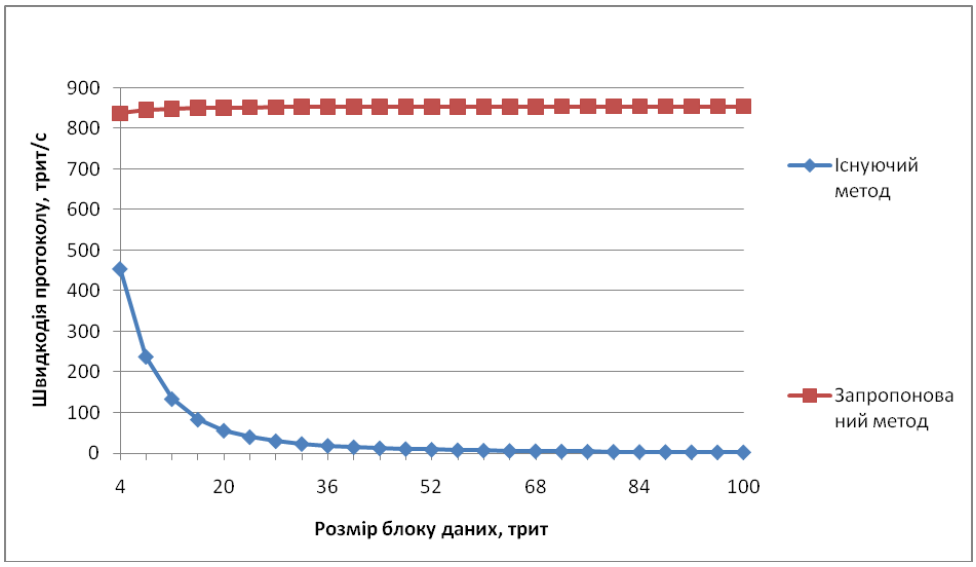

Рис. 4.4. Порівняння швидкісних характеристик протоколу КПБЗ (експеримент 3)

Експеримент 4. $V_x = V_{kl} = V_{gen} = 10^5$ трит/сек, $V_{kv} = 10^4$ трит/сек, $l = 1000$, $q = 0,5$ – для відомого методу забезпечення стійкості протоколів КПБЗ, $q = 0,05$ для запропонованого методу.

На рис. 4.5. наведено результати експерименту 4 із порівняння швидкодії протоколу КПБЗ для різних методів забезпечення його стійкості.

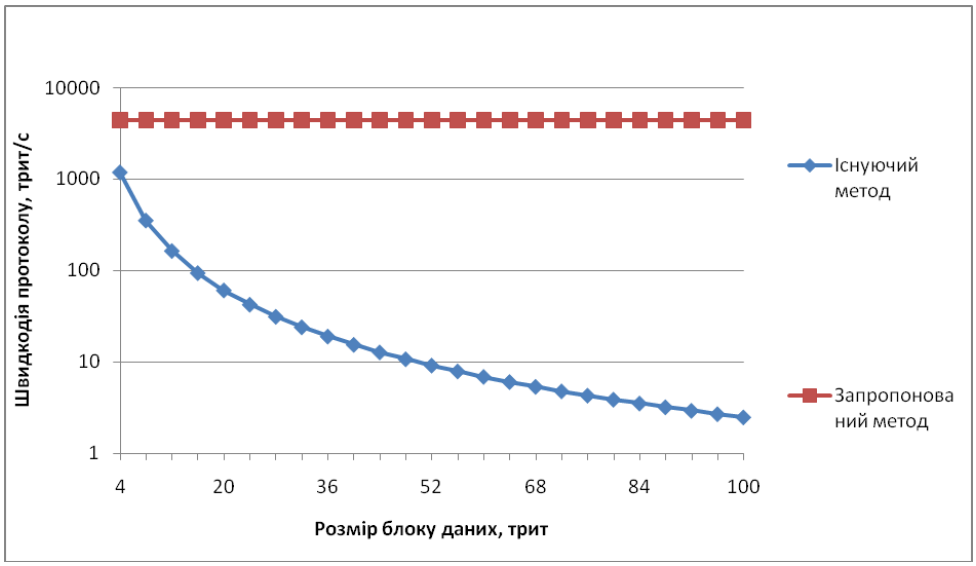

Рис. 4.5. Порівняння швидкісних характеристик протоколу КПБЗ (експеримент 4)

Згідно результатів експерименту, швидкість протоколу КПБЗ із запропонованим методом забезпечення стійкості мінімум у 3,73 раз краща за швидкість

відомого методу (для $r=4$). Причому при збільшенні r покращення швидкодії буде ще більш показовим. Наприклад, при $r=20$ швидкодія запропонованого методу краща у 73,29 раз.

Експеримент 5. $V_x=V_{kl}=V_{gen}=V_{kv}=10^5$ трит/сек, $l=1000$, $q=0,5$ – для відомого методу забезпечення стійкості протоколів КПБЗ, $q=0,05$ для запропонованого методу.

На рис. 4.6. наведено результати експерименту 5 із порівняння швидкодії протоколу КПБЗ для різних методів забезпечення його стійкості.

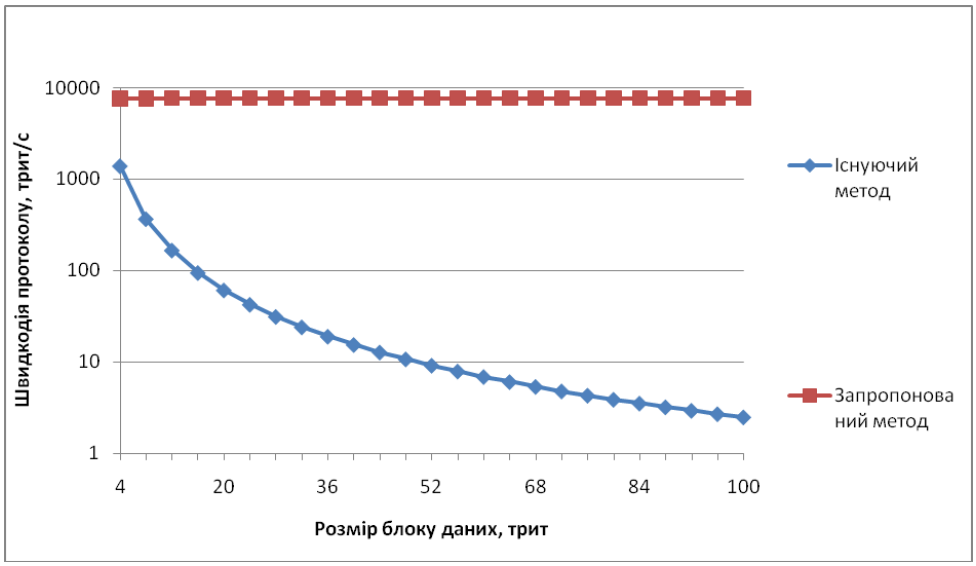

Рис. 4.6. Порівняння швидкісних характеристик протоколу КПБЗ (експеримент 5)

Згідно результатів експерименту, швидкість протоколу КПБЗ із запропонованим методом забезпечення стійкості мінімум у 5,45 раз краща за швидкість відомого методу (для $r=4$). Причому, при збільшенні r покращення швидкодії буде ще більш показовим. Наприклад, при $r=20$ швидкодія запропонованого методу краща у 125,53 раз.

Експеримент 6. $V_x=10^6$ трит/сек, $V_{kl}=10^5$ трит/сек, $V_{gen}=10^4$ трит/сек, $V_{kv}=10^3$ трит/сек, $l=1000$, $q=0,5$ – для відомого методу забезпечення стійкості протоколів КПБЗ, $q=0,05$ для запропонованого методу.

На рис. 4.7. наведено результати експерименту 6 порівняння швидкодії протоколу КПБЗ для різних методів забезпечення його стійкості.

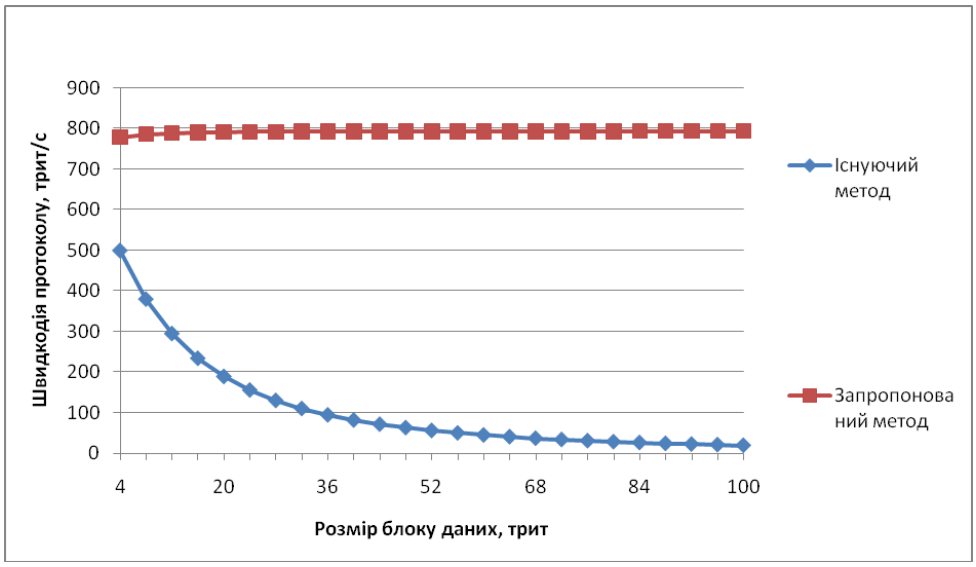

Рис. 4.7. Порівняння швидкісних характеристик протоколу КПБЗ (експеримент 6)

Згідно результатів експерименту, швидкість протоколу КПБЗ із запропонованим методом забезпечення стійкості мінімум у 1,55 раз краща за швидкість відомого методу (для $r = 4$). Причому при збільшенні r покращення швидкодії буде ще більш показовим. Наприклад, при $r = 20$ швидкодія запропонованого методу краща у 4,18 рази.

Експеримент 7. $V_x = 10^5$ трит/сек, $V_{kl} = 10^6$ трит/сек, $V_{gen} = 10^4$ трит/сек, $V_{kv} = 10^3$ трит/сек, $l = 1000$, $q = 0,5$ – для відомого методу забезпечення стійкості протоколів КПБЗ, $q = 0,05$ для запропонованого методу.

На рис. 4.8. наведено результати експерименту 7 із порівняння швидкодії протоколу КПБЗ для різних методів забезпечення його стійкості.

Згідно результатів експерименту, швидкість протоколу КПБЗ із запропонованим методом забезпечення стійкості мінімум у 1,83 раз краща за швидкість відомого методу (для $r = 4$). Причому, при збільшенні r покращення швидкодії буде ще більш показовим. Наприклад, при $r = 20$ швидкодія запропонованого методу краща у 14,39 раз.

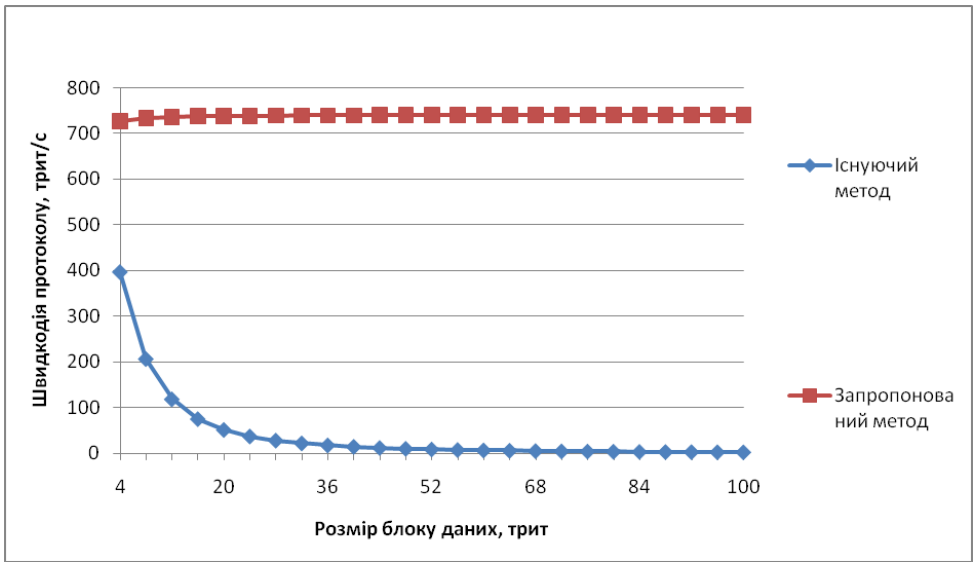

Рис. 4.8. Порівняння швидкісних характеристик протоколу КПБЗ (експеримент 7)

Отже згідно з результатів експериментів, швидкість протоколу КПБЗ із запропонованим методом забезпечення стійкості мінімум у 1,52 раз краща за швидкість відомого методу. Проте зауважимо, що така кількість разів отримана для $r = 4$. У роботі [62] зазначається, що Аліса і Боб можуть підібрати параметри протоколу (розмір блоку r, ймовірність перемикання в режим контролю підслуховування q та інші параметри) так, щоб ймовірність успішної некогерентної атаки Єви після передачі одного блоку розміром r була нехтовно малою величиною. Можна зробити висновок, що для ефективного використання відомого та запропонованого методів забезпечення стійкості протоколів КК рекомендований розмір $r \geq 20$, в такому випадку швидкодія запропонованого методу мінімум краща у 4,4 рази.

4.3. Верифікація методів генерування псевдовипадкових послідовностей та оцінювання якості

Дослідження методів генерування ПВП за методикою NIST STS

Спочатку запропонований метод генерування тритових ПВП було вирішено перевірити за методикою NIST STS [172]. Потрібно було зрозуміти, чи

зможуть стандарті бітові тести адекватно оцінити псевдовипадковість тритових послідовностей.

Статистичні тести NIST STS використовуються для визначення якісних та кількісних ознак випадковості послідовності чисел [173]. Для прийняття висновків щодо проходження послідовностями випадкових чисел статистичних тестів використовується три основні критерії [172, 173]:

1. Критерій прийняття рішення на основі установлення деякого граничного рівня.

2. Критерій на основі установлення фіксованого довірчого інтервалу.

3. Критерій визначення для статистики тесту деякого відповідного значення ймовірності P-value (ймовірності випадку, що статистика тесту прийме значення більше за значення, що спостерігається при випробуванні послідовності, в передбаченні її випадковості).

В основі статистичного тесту лежить перевірка деякої нульової гіпотези H_0 такої, що досліджувана послідовність – випадкова [73, 98, 173]. Також передбачена альтернативна гіпотеза H_1 − досліджувана послідовність не випадкова. Отже, згенерована послідовність досліджується набором тестів, у кожному з яких робиться висновок щодо відхилення або прийняття нульової гіпотези H_0. Для кожного тесту обирається адекватна статистика випадковості, на підставі котрої далі відхиляється або приймається гіпотеза H_0. Така статистика, відповідно припущенню на випадковість, володіє деяким розподілом випадкових значень. Теоретично розподіл статистики для нульової гіпотези вираховується із застосуванням математичних методів. Далі із такого зразкового розподілу визначається критичне значення. При проведенні тесту вираховується значення тестової статистики, яке порівнюється із критичним значенням. При перевищенні тестового критичного значення над еталонним, приймається гіпотеза H_1, в іншому випадку − гіпотеза H_0.

Для проведення експериментів були обрані такі вхідні параметри для застосування статистичних тестів NIST STS (згідно [73, 172, 173]):

1. Довжина послідовності, що тестується $n = 10^6$ біт.

2. Кількість послідовностей, що тестується $m = 100$.
3. Рівень значущості $\alpha = 0{,}01$.
4. Кількість тестів q = 188, серед яких такі тести: *Frequency* − 1, *BlockFrequency* − 1, *CumulativeSums* − 2, *Runs* − 1, *LongestRun* − 1, *Rank* − 1, *FFT* − 1, *NonOverlappingTemplate* − 148, *OverlappingTemplate* − 1, *Universal* − 1, *ApproximateEntropy* − 1, *RandomExcursions* − 8, *RandomExcursionsVariant* − 18, *Serial* − 2, *LinearComplexity* − 1.

Таким чином, обсяг вибірки, що тестується, склав $N = 10^6 \times 100 = 10^8$ біт, кількість тестів (q) для різних довжин $q = 188$, таким чином, статистичний портрет генератора містить 18800 значень ймовірності P.

В ідеальному випадку при $m = 100$ і $\alpha = 0{,}01$ у ході тестування може бути відкинута тільки одна послідовність зі ста, тобто коефіцієнт проходження кожного тесту має складати 99%. Але це занадто жорстке правило, тому аналогічно роботі [73] було застосовано правило на основі довірчого інтервалу, нижня межа дорівнює 0,96015.

Для проведення експериментів був розроблений консольний програмний засіб TriGen на мові програмування C++ в середовищі Microsoft Visual Studio 2012 (програмний код якого наведений у додатку Б), який дозволяє генерувати тритові послідовності запропонованим алгоритмом TriGen та стандартним генератором ПВП C++.

Експерименти було вирішено проводити наступним чином:

1. Кожним досліджуваним генератором (алгоритмом Trigen та стандартним генератором ПВП C++) виконувалось генерування трьох послідовностей довжиною 60000000 трит.
2. Відбувалось переведення тритової послідовності у двійкову систему числення. Для цього тритова послідовніть ділилася на блоки довжиною 24 трита, кожен блок окремо переводився у 40 бітну послідовність. В результаті чого отримувалась двійкова послідовність довжиною 10^8 біт.

3. Отримана двійкова послідовність перевірялася статистичними тестами NIST STS. В результаті отримувались статистичні портрети послідовностей.

Генерація тритових ПВП стандартним генератором C++ виконувалась наступний чином:

1. Обиралися довільні значення векторів k і a (для кожної послідовності окремо) та вказувалась потрібна довжина послідовності n.
2. За допомогою функції rand() відбувалось генерування кожного трита послідовності.

Псевдокод процедури генерування наведений на рис. 4.9.

C++ gen

Input: вектори k і a, $k \in V_{32}$, $a \in V_{32}$ ($V_n = \{0,1\}^n$).

Output: вихідна послідовність $M = (M_1,...,M_n)$, $M_q \in \{0,1,2\}^n$, $q \in \overline{1,\mathrm{n}}$.

1. $x = k$, $y = a$;

2. $std : srand\left(std :: time(0)\right)$;

2. $For\ q = 1; q \le n;\ q{+}{+}\ do$

2.1. $For\ j = 0;\ j < 4;\ j{+}{+}\ do$

2.1.1. $b = (\mathrm{rand()} + \mathrm{x}) \oplus y$;

2.1.2. $c = (\mathrm{rand()} + \mathrm{y}) \oplus x$;

2.1.3. $y = (\mathrm{rand()} + \mathrm{b}) \oplus c$

2.1.4. $x = (\mathrm{rand()} + \mathrm{y}) \oplus b$

2.2. $M[\mathrm{q}] = x\%3$;

Рис. 4.9. Псевдокод процедури генерування тритових ПВП стандартним генератором C++

Генерація тритових ПВП запропонованим алгоритмом TriGen виконувалась згідно опису, що наведений у розділі 3. Для кожної послідовності окремо задавались початкові значення 24 тритних векторів k_i, $i \in \overline{1,4}$ після чого виконувалось генерування послідовностей.

Спочатку опишемо результати дослідження стандартного генератора ПВП C++.

Послідовність 1 (C++ gen).

Для генерації було обрані наступні початкові параметри: $k = 314342312$, $a = 403242341$. На рис. 4.10 зображен статистичний портрет проходження тестів Nist STS.

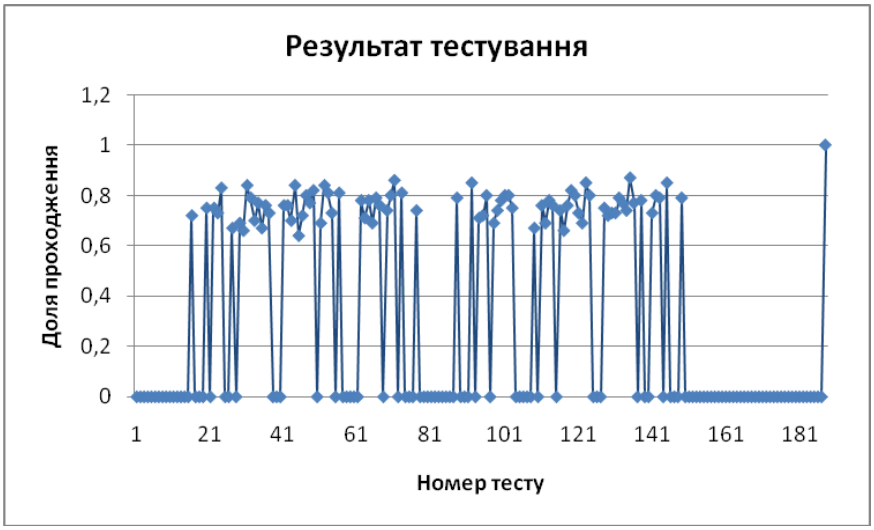

Рис. 4.10. Статистичний портрет C++ gen послідовності 1

Як видно із результатів кількість тестів, що пройшло 99% досліджуваних послідовностей − 1, пройшло 96% − 1. Що свідчить, що згенерована послідовність, що була переведела у двійковий вид тестування не пройшла.

Послідовність 2 (C++ gen).

Для генерації було обрані наступні початкові параметри: $k = 3834425654$, $a = 234525320$. На рис. 4.11 зображен статистичний портрет проходження тестів Nist STS.

Рис. 4.11. Статистичний портрет C++ gen послідовності 2

Як видно із результатів кількість тестів, що пройшло 99% досліджуваних послідовностей − 9, пройшло 96% − 20. Що свідчить, що згенерована послідовність також не пройшла тестування.

Послідовність 3 (C++ gen).

Для генерації було обрані наступні початкові параметри: $k = 2577261391$, $a = 980904215$. На рис. 4.12 зображен статистичний портрет проходження тестів Nist STS.

Рис. 4.12. Статистичний портрет C++ gen послідовності 3

Як видно із результатів кількість тестів, що пройшло 99% досліджуваних послідовностей − 6, пройшло 96% − 33. Що свідчить, що згенерована послідовність також не пройшла тестування.

Тепер опишемо результати дослідження алгоритму TriGen.

Послідовність 1 (TriGen).

Для генерації було обрані наступні початкові параметри:

$k_1 = 000000\ 000000\ 000000\ 000000$, $k_2 = 000000\ 000000\ 000000\ 000001$,

$k_3 = 000000\ 000000\ 000200\ 000000$, $k_4 = 000000\ 100000\ 000000\ 000000$.

На рис. 4.13 зображен статистичний портрет проходження тестів Nist STS.

Рис. 4.13. Статистичний портрет TriGen послідовності 1

Як видно із результатів кількість тестів, що пройшло 99% досліджуваних послідовностей − 32, пройшло 96% − 41. Що свідчить, що згенерована послідовність алгоритмом TriGen також не пройшла тестування.

Послідовність 2 (TriGen).

Для генерації було обрані наступні початкові параметри:

$k_1 = 000001\,000000\,000001\,011000$, $k_2 = 120000\,000000\,000210\,201001$,

$k_3 = 200100\,200100\,010200\,100001$, $k_4 = 201010\,111111\,211000\,002222$.

На рис. 4.14 зображен статистичний портрет проходження тестів Nist STS.

Рис. 4.14. Статистичний портрет TriGen послідовності 2

Як видно із результатів кількість тестів, що пройшло 99% досліджуваних послідовностей − 24, пройшло 96% − 37. Що свідчить, що згенерована послідовність алгоритмом TriGen також не пройшла тестування.

Послідовність 3 (TriGen).

Для генерації було обрані наступні початкові параметри:

$k_1 = 102210\ 210120\ 001022\ 010110$, $k_2 = 201001\ 020210\ 121210\ 121011$,

$k_3 = 102021\ 022011\ 120101\ 122102$, $k_4 = 120220\ 110200\ 102112\ 011012$.

На рис. 4.15 зображен статистичний портрет проходження тестів Nist STS.

Рис. 4.15. Статистичний портрет TriGen послідовності 3

Як видно із результатів кількість тестів, що пройшло 99% досліджуваних послідовностей − 9, пройшло 96% − 51. Що свідчить, що згенерована послідовність алгоритмом TriGen також не пройшла тестування.

Отже можна зробии висновок, що стандарті бітові тести не можуть корректно оцінити псевдовипадковість тритових послідовностей. Це можна пояснити переведення тритової послідовності у двійкову систему числення. При такому переведенні тритова блок (в даному випадку 24 трита) не може покрити повністю утворений бітовий блок (в даному випадку 40 біта). Оскільки $3^{24} = 282429536481$, а $2^{40} = 1099511627776$ то 817082091295 значення бітового блоку ніколи не появляться у ньому, тому послідовність є передбачуваною.

Дослідження методів генерування ПВП запропонованим методом оцінювання їх якості

У розділі 3 детально описан метод оцінювання якості ПВП. У даному розділі опишемо вхідні параметри для його застосування (параметри обрано аналогічно тестам NIST STS):

1. Довжина послідовності тритів n = 150000 трит.
2. Кількість послідовностей, що тестується m = 100.
3. Рівень значущості α = 0,01.
4. Кількість тестів q = 152, серед яких такі тести: *Freqeuncy monotrit test* – 1, *Frequency block trit test* – 1, *Trit runs test* – 1, *Test for the longest run in a block* – 1, *Trit non-overlapping template matching test* – 148, *Trit overlapping template matching test* – 1.

Таким чином, обсяг вибірки, що тестується, склав $N = 1{,}5 \cdot 10^7$ трит, кількість тестів (q) для різних довжин q = 152, таким чином, статистичний портрет генератора містить 152 значень ймовірності P. Як і для тестів NIST STS будемо застосовувати правило довірчих інтервалів, тобто коефіцієнт проходження кожного тесту має складати 0,96015%.

На основі запропонованого методу оцінки якості ПВП та наведених вище вхідних параметрах було розроблене консольне програмне забезпечення TritSTS (на мові програмування C++ в середовищі Microsoft Visual Studio 2012, програмний код якого наведений у додатку Б), що дозволяє оцінювати тритові послідованості на псевдовипадковість. Даний засіб використовувався при проведенні експериментів.

Експерименти було вирішено проводити наступним чином:

1. Кожним досліджуваним генератором (алгоритмом Trigen та стандартним генератором ПВП C++) виконувалось генерування п’ятьох послідовностей довжиною $N = 1{,}5 \cdot 10^7$ трит.
2. Отримана тритова послідовність перевірялася програмою TritSTS. В результаті отримувались статистичні портрети послідовностей.

Генерація тритових послідовностей стандартним генератором C++ та запропонованим алгоритом Trigen розписана вище у доному розділі.

Перевірку на псевдовипадковість програмою TritSTS почнемо із результатів дослідження алгоритму TriGen.

Послідовність 1 (TriGen).

Для генерації було обрані наступні початкові параметри (які вже використовувались при перевірці тестами NIST STS):

$k_1 = 102210\ 210120\ 001022\ 010110$, $k_2 = 201001\ 020210\ 121210\ 121011$,

$k_3 = 102021\ 022011\ 120101\ 122102$, $k_4 = 120220\ 110200\ 102112\ 011012$.

На рис. 4.16 зображен статистичний портрет проходження тестів TritSTS.

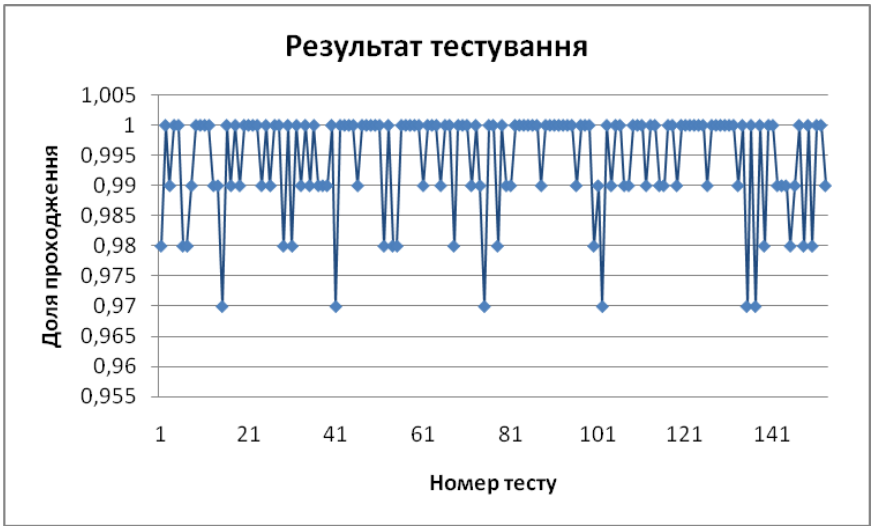

Рис. 4.16. Статистичний портрет TriGen послідовності 1

Як видно із результатів кількість тестів, що пройшло 99% досліджуваних послідовностей − 132, пройшло 96% − 153. Кількість $P_{value_{01}} \geq 0{,}01$ − 153, $P_{value_{02}} \geq 0{,}01$ − 153, $P_{value_{12}} \geq 0{,}01$ − 153. Такі результати свідчать, що згенерована послідовність пройшла тестування.

Послідовність 2 (TriGen).

Для генерації було обрані наступні початкові параметри (які вже використовувались при перевірці тестами NIST STS):

$k_1 = 000000\ 000000\ 000000\ 000000$, $k_2 = 000000\ 000000\ 000000\ 000001$,

$k_3 = 000000\ 000000\ 000200\ 000000$, $k_4 = 000000\ 100000\ 000000\ 000000$.

На рис. 4.17 зображен статистичний портрет проходження тестів TritSTS.

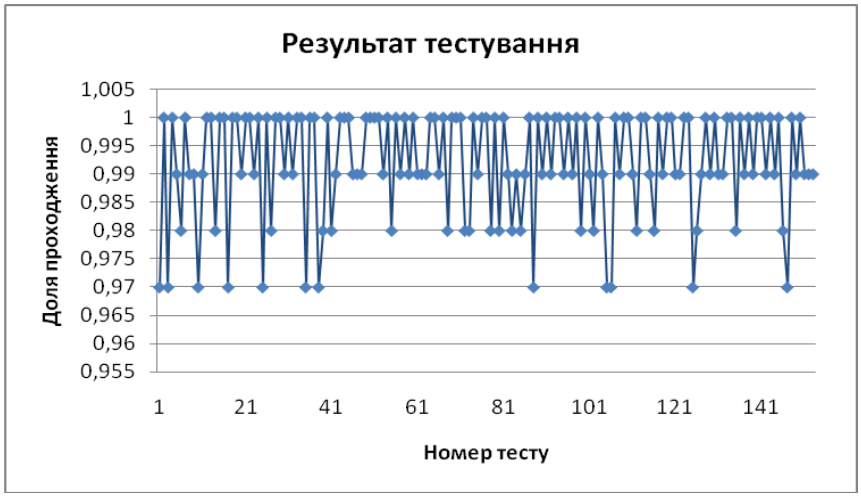

Рис. 4.17. Статистичний портрет TriGen послідовності 2

Як видно із результатів кількість тестів, що пройшло 99% досліджуваних послідовностей − 121, пройшло 96% − 153. Кількість $P_{value_{01}} \geq 0{,}01$ − 152, $P_{value_{02}} \geq 0{,}01$ − 151, $P_{value_{12}} \geq 0{,}01$ − 153. Такі результати свідчать, що згенерована послідовність пройшла тестування (не зважаючи на $P_{value_{XY}} < 0{,}01$).

Послідовність 3 (TriGen).

Для генерації було обрані наступні початкові параметри (які вже використовувались при перевірці тестами NIST STS):

$k_1 = 000000\ 000000\ 000000\ 000000$, $k_2 = 000000\ 000000\ 000000\ 000001$,

$k_3 = 000000\ 000000\ 000200\ 000000$, $k_4 = 000000\ 100000\ 000000\ 000000$.

На рис. 4.18 зображен статистичний портрет проходження тестів TritSTS.

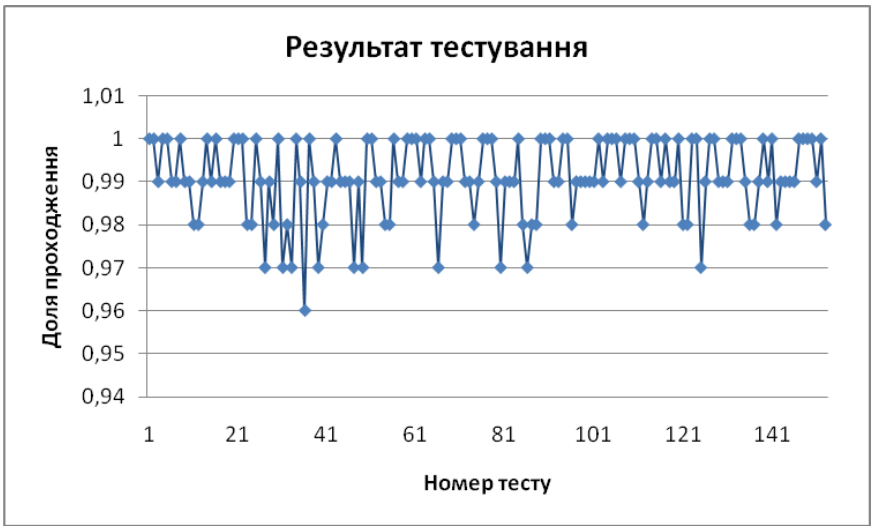

Рис. 4.18. Статистичний портрет TriGen послідовності 3

Як видно із результатів кількість тестів, що пройшло 99% досліджуваних послідовностей − 121, пройшло 96% − 152. Кількість $P_{value_{01}} \geq 0,01$ − 151, $P_{value_{02}} \geq 0,01$ − 153, $P_{value_{12}} \geq 0,01$ − 152. Такі результати свідчать, що згенерована послідовність не погана, проте вона не пройшла тестування.

Послідовність 4 (TriGen).

Для генерації було обрані наступні початкові параметри :

$k_1 = 100020\,102000\,100022\,011102$, $k_2 = 221100\,010101\,020202\,212121$,

$k_3 = 111002\,020100\,210200\,110202$, $k_4 = 120022\,102021\,022001\,100202$.

На рис. 4.19 зображен статистичний портрет проходження тестів TritSTS.

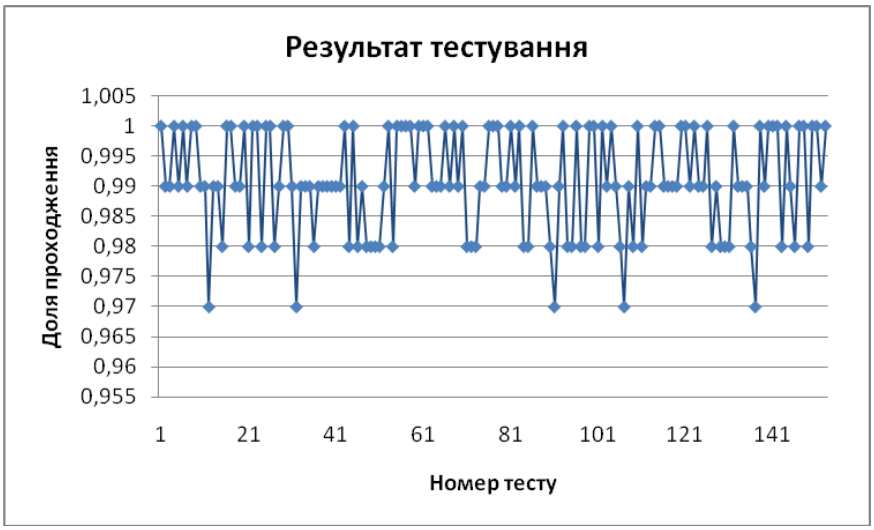

Рис. 4.19. Статистичний портрет TriGen послідовності 4

Як видно із результатів кількість тестів, що пройшло 99% досліджуваних послідовностей − 114, пройшло 96% − 153. Кількість $P_{value_{01}} \geq 0,01$ − 152, $P_{value_{02}} \geq 0,01$ − 152, $P_{value_{12}} \geq 0,01$ − 151. Такі результати свідчать, що згенерована послідовність пройшла тестування (не зважаючи на $P_{value_{XY}} < 0,01$).

Послідовність 5 (TriGen).

Для генерації було обрані наступні початкові параметри:

$k_1 = 221020\,102001\,100022\,010112$, $k_2 = 021222\,110201\,120001\,202021$,

$k_3 = 121110\,022001\,112202\,210201$, $k_4 = 102212\,001021\,122011\,201202$.

На рис. 4.20 зображен статистичний портрет проходження тестів TritSTS.

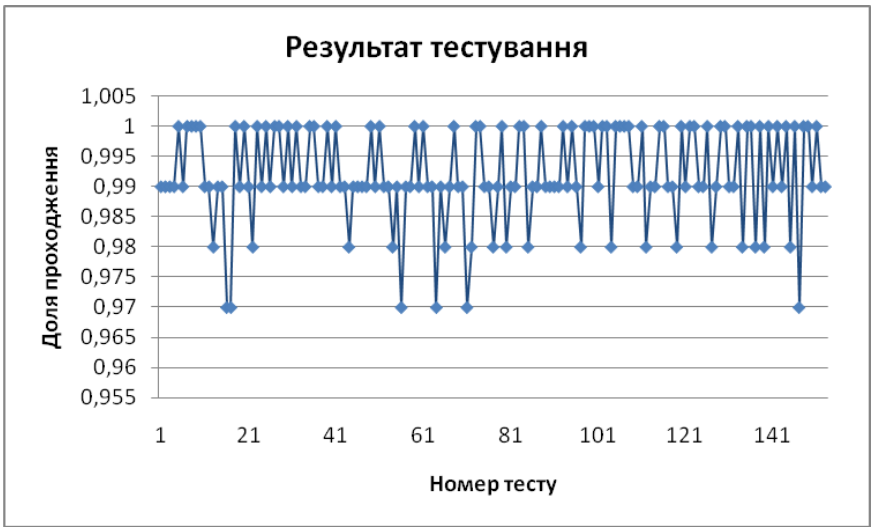

Рис. 4.20. Статистичний портрет TriGen послідовності 5

Як видно із результатів кількість тестів, що пройшло 99% досліджуваних послідовностей − 129, пройшло 96% − 153. Кількість $P_{value_{01}} \geq 0{,}01$ − 153, $P_{value_{02}} \geq 0{,}01$ − 153, $P_{value_{12}} \geq 0{,}01$ − 153. Такі результати свідчать, що згенерована послідовність пройшла тестування.

Тепер опишемо результати дослідження стандартного генератора ПВП C++ програмою TritSTS.

Послідовність 1 (C++ gen).

Для генерації було обрані наступні початкові параметри: $k = 314342312$, $a = 403242341$ (дані параметри вже використовувались при перевірці тестами NIST STS). На рис. 4.21 зображен статистичний портрет проходження тестів TritSTS.

Рис. 4.21. Статистичний портрет C++ gen послідовності 1

Кількість тестів, що пройшло 99% досліджуваних послідовностей − 109, пройшло 96% − 150. Кількість $P_{value_{01}} \geq 0,01$ − 143, $P_{value_{02}} \geq 0,01$ − 146, $P_{value_{12}} \geq 0,01$ − 141. Такі результати свідчать, що згенерована послідовність не погана, проте вона не пройшла тестування.

Послідовність 2 (C++ gen).

Для генерації було обрані наступні початкові параметри: $k = 3834425654$, $a = 234525320$ (дані параметри вже використовувались при перевірці тестами NIST STS). На рис. 4.22 зображен статистичний портрет проходження тестів TritSTS.

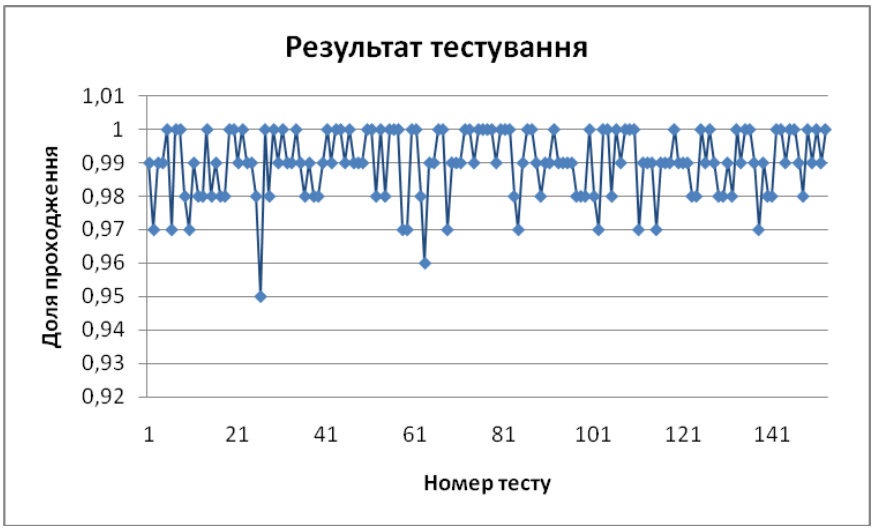

Рис. 4.22. Статистичний портрет C++ gen послідовності 2

Кількість тестів, що пройшло 99% досліджуваних послідовностей − 111, пройшло 96% − 151. Кількість $P_{value_{01}} \geq 0,01$ − 145, $P_{value_{02}} \geq 0,01$ − 142, $P_{value_{12}} \geq 0,01$ − 143. Такі результати свідчать, що згенерована послідовність не погана, проте вона не пройшла тестування.

Послідовність 3 (C++ gen).

Для генерації було обрані наступні початкові параметри: $k = 2577261391$, $a = 980904215$ (дані параметри вже використовувались при перевірці тестами NIST STS). На рис. 4.23 зображен статистичний портрет проходження тестів TritSTS.

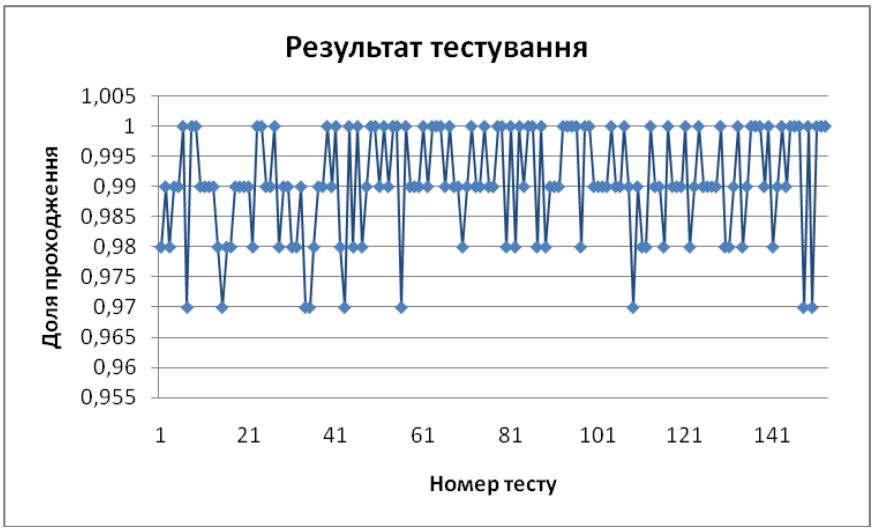

Рис. 4.23. Статистичний портрет C++ gen послідовності 3

Кількість тестів, що пройшло 99% досліджуваних послідовностей − 117, пройшло 96% − 153. Кількість $P_{value_{01}} \geq 0,01$ − 153, $P_{value_{02}} \geq 0,01$ − 153, $P_{value_{12}} \geq 0,01$ − 153. Такі результати свідчать, що згенерована послідовність пройшла тестування.

Послідовність 4 (C++ gen).

Для генерації було обрані наступні початкові параметри: $k = 5674312$, $a = 8476892$. На рис. 4.24 зображен статистичний портрет проходження тестів TritSTS.

Рис. 4.24. Статистичний портрет C++ gen послідовності 4

Кількість тестів, що пройшло 99% досліджуваних послідовностей − 108, пройшло 96% − 143. Кількість $P_{value_{01}} \geq 0,01$ − 133, $P_{value_{02}} \geq 0,01$ − 131,

$P_{value_{12}} \geq 0,01$ – 128. Такі результати свідчать, що згенерована послідовність не погана, проте вона не пройшла тестування.

Послідовність 5 (C++ gen).

Для генерації було обрані наступні початкові параметри: $k = 7890212$, $a = 34095422$. На рис. 4.25 зображен статистичний портрет проходження тестів TritSTS.

Рис. 4.25. Статистичний портрет C++ gen послідовності 5

Кількість тестів, що пройшло 99% досліджуваних послідовностей – 114, пройшло 96% – 150. Кількість $P_{value_{01}} \geq 0,01$ – 139, $P_{value_{02}} \geq 0,01$ – 145, $P_{value_{12}} \geq 0,01$ – 141. Такі результати свідчать, що згенерована послідовність не погана, проте вона не пройшла тестування.

Зведено результати експериментів в одну таблицю (див. табл. 4.2).

Згідно отриманих результатів послідовності згенеровані запропонованим алгоритмом TriGen показали кращі результати ніж ПВП згенеровані стандартним генератором C++.

Зауважимо, що із п’яти послідовностей згенерованим стандартним генератором C++ тільки одна пройшла тестування програмою TritSTS, а за допомогою генератора TriGen успішно пройшли вже чотири із п’яти послідовностей. Якщо брати середні результати, то послідовності згенеровані алгоритмом TriGen на 11,6% частіше успішно проходять 99% послідовностей і на 3,4% частіше успішно проходять 96% послідовностей.

Таблиця 4.2

Результати проходження тестів TritSTS

Посл.	$P_{value_{01}} \geq 0,01$	$P_{value_{02}} \geq 0,01$	$P_{value_{12}} \geq 0,01$	Кількість тестів у яких тестування пройшло	
				99%	96%
TriGen-1	153	153	153	132	153
TriGen-2	152	151	153	121	153
TriGen-3	151	153	152	121	152
TriGen-4	152	152	151	114	153
TriGen-5	153	153	153	129	153
C++ gen-1	143	146	141	109	150
C++ gen-2	145	142	143	111	151
C++ gen-3	153	153	153	117	153
C++ gen-4	133	131	128	108	143
C++ gen-5	139	145	141	114	150
TriGen сред.	**152,2**	**152,4**	**152,4**	**123,4**	**152,8**
C++ gen сред.	**142,6**	**143,4**	**141,2**	**111,8**	**149,4**

Перевірка адекватності роботи розробленого метода оцінювання якості тритових послідовностей.

Для перевірки адекватності розробленого методу оцінки якості тритових послідовностей (програми TritSTS) виконувалось наступне:

1. Спочатку підбирались послідовності, які завідомо не повинні проходити тести. Виконувалось перевірка їх псевдовипадковості.
2. Потім TriGen генерувались послідовності, які завідомо повинні пройти ці тести. Виконувалось перевірка їх псевдовипадковості.
3. Аналізувались отримані результати.

Підібрана послідовність 1:

00000000000000000000000000000000000000000000000000000000000001111
11111111111111111111111111111111111111111111111122222222222222222
22222222222222222222222222222222222222222222.

Очікуваний результат – послідовність не випадкова. Отримані результати перевірки див. табл. 4.3. Згідно таблиці послідовність не є випадковою.

Таблиця 4.3

Результати перевірки послідовності 1

Тест	$P_{value_{01}}$	$P_{value_{02}}$	$P_{value_{12}}$	Тест пройдений
FrequencyTritTest	0.409355	0.788447	0.577469	+
FrequencyTritTestWithinBlock	0.000000	0.000000	0.000000	-
RunsTritTest	0.000000	0.000000	0.000000	-
LongestRunOfTrit	0.000055	0.000041	0.000107	-
NonOverlappingTemplateTritTest	0.001978	0.001978	0.001978	-
NonOverlappingTemplateTritTest	0.000596	0.000596	0.000596	-
NonOverlappingTemplateTritTest	0.001978	0.001978	0.000596	-
NonOverlappingTemplateTritTest	0.000596	0.000596	0.000596	-
OverlappingTemplateTritTest	0.000596	0.000596	0.000596	-

Підібрана послідовність 2:

000011112222000011112222000011112222000011112222000011112222000011112222000011112222000011112222000011112222000011112222000011112222000011112222000011112222.

Очікуваний результат – послідовність не випадкова. Отримані результати перевірки див. табл. 4.4. Згідно таблиці послідовність не є випадковою.

Таблиця 4.4

Результати перевірки послідовності 2

Тест	$P_{value_{01}}$	$P_{value_{02}}$	$P_{value_{12}}$	Тест пройдений
FrequencyTritTest	0.676657	0.393769	0.662521	+
FrequencyTritTestWithinBlock	0.000000	0.000000	0.000000	-
RunsTritTest	0.000002	0.000001	0.000002	-
LongestRunOfTrit	0.000000	0.000000	0.000000	-
NonOverlappingTemplateTritTest	0.330388	0.330388	0.330388	+
NonOverlappingTemplateTritTest	0.000596	0.000596	0.000596	-
NonOverlappingTemplateTritTest	0.330388	0.330388	0.330388	+
NonOverlappingTemplateTritTest	0.636565	0.330388	0.330388	+
OverlappingTemplateTritTest	0.000596	0.000596	0.000596	-

Підібрана послідовність 3:

000000001111111111222222000000000111111111222000111001202102101020102102010201202010201021012021012012012012002101200111111111120212102202101201021020111112001111002102020202012 01.

Очікуваний результат – послідовність не випадкова. Отримані результати перевірки див. табл. 4.5. Згідно таблиці послідовність не є випадковою.

Таблиця 4.5

Результати перевірки послідовності 3

Тест	$P_{value_{01}}$	$P_{value_{02}}$	$P_{value_{12}}$	Тест пройдений
FrequencyTritTest	0.376759	0.024745	0.001883	-
FrequencyTritTestWithinBlock	0.042406	0.304453	0.074754	+
RunsTritTest	0.913690	0.013064	0.859162	+
LongestRunOfTrit	0.419660	0.085076	0.411822	+
NonOverlappingTemplateTritTest	0.636565	0.142906	0.931334	+
NonOverlappingTemplateTritTest	0.006313	0.001978	0.142906	-
NonOverlappingTemplateTritTest	0.054678	0.019184	0.636565	+
NonOverlappingTemplateTritTest	0.330388	0.330388	0.330388	+
OverlappingTemplateTritTest	0.019184	0.019184	0.054678	+

Підібрана послідовність 4:

012012012012012012012012012012012012012110022010202121010202121010202121010202121010202121010202121010202121010202121010202121111100000222222000002222221111110001112220001112220120120 12.

Очікуваний результат – послідовність не випадкова. Отримані результати перевірки див. табл. 4.6. Згідно таблиці послідовність не є випадковою.

Підібрана послідовність 5:

111022211102221110222111022200012220001222000122200012220001222111200011120001112000111200011120001112000111200011102221110222111022211102221110222111022211102221110222000122200012220001222000122200 01.

Очікуваний результат – послідовність не випадкова. Отримані результати перевірки див. табл. 4.7. Згідно таблиці послідовність не є випадковою.

Таблиця 4.6

Результати перевірки послідовності 4

Тест	$P_{value_{01}}$	$P_{value_{02}}$	$P_{value_{12}}$	Тест пройдений
FrequencyTritTest	1.000000	0.925704	0.925704	+
FrequencyTritTestWithinBlock	0.573045	0.681536	0.781291	+
RunsTritTest	0.039352	0.006825	0.019688	-
LongestRunOfTrit	0.480254	0.442637	0.232703	+
NonOverlappingTemplateTritTest	0.330388	0.636565	0.330388	+
NonOverlappingTemplateTritTest	0.330388	0.142906	0.142906	+
NonOverlappingTemplateTritTest	0.636565	0.636565	0.330388	+
NonOverlappingTemplateTritTest	0.636565	0.636565	0.636565	+
OverlappingTemplateTritTest	0.142906	0.330388	0.330388	+

Таблиця 4.7

Результати перевірки послідовності 5

Тест	$P_{value_{01}}$	$P_{value_{02}}$	$P_{value_{12}}$	Тест пройдений
FrequencyTritTest	0.853923	0.715001	0.856311	+
FrequencyTritTestWithinBlock	0.678247	0.675132	0.725032	+
RunsTritTest	0.065980	0.069434	0.103713	+
LongestRunOfTrit	0.000056	0.000016	0.000016	-
NonOverlappingTemplateTritTest	0.142906	0.142906	0.142906	+
NonOverlappingTemplateTritTest	0.019184	0.001978	0.006313	-
NonOverlappingTemplateTritTest	0.019184	0.142906	0.019184	+
NonOverlappingTemplateTritTest	0.054678	0.142906	0.142906	+
OverlappingTemplateTritTest	0.001978	0.001978	0.006313	-

Згенерована послідовність TriGen-1:

000202221120021111222111202011101212102120001220211011100021020211221202022021210022121121101111002001021220022001001222222012010101221221001211012211020120210200201010022011010012100011222011.

Очікуваний результат – послідовність випадкова. Отримані результати перевірки див. табл. 4.8. Згідно таблиці послідовність є випадковою.

Таблиця 4.8

Результати перевірки згенерованої послідовності TriGen-1

Тест	$P_{value_{01}}$	$P_{value_{02}}$	$P_{value_{12}}$	Тест пройдений
FrequencyTritTest	0.783309	0.783309	1.000000	+
FrequencyTritTestWithinBlock	0.625491	0.173564	0.149750	+
RunsTritTest	0.921404	0.921404	1.000000	+
LongestRunOfTrit	0.677032	0.901010	0.890900	+
NonOverlappingTemplateTritTest	0.931334	0.931334	0.931334	+
NonOverlappingTemplateTritTest	0.054678	0.330388	0.330388	+
NonOverlappingTemplateTritTest	0.054678	0.330388	0.931334	+
NonOverlappingTemplateTritTest	0.931334	0.636565	0.931334	+
OverlappingTemplateTritTest	0.330388	0.636565	0.636565	+

Згенерована послідовність TriGen-2 :

022200121010202220022201120210200102112021022212200011221001222202222110101021111101111011220022111221201110121002100220122022011120120210120202222010101211102110122200202222202211020220022 2.

Очікуваний результат – послідовність випадкова. Отримані результати перевірки див. табл. 4.9. Згідно таблиці послідовність є випадковою.

Таблиця 4.9

Результати перевірки згенерованої послідовності TriGen-2

Тест	$P_{value_{01}}$	$P_{value_{02}}$	$P_{value_{12}}$	Тест пройдений
FrequencyTritTest	0.857462	0.643902	0.521075	+
FrequencyTritTestWithinBlock	0.569233	0.414004	0.625491	+
RunsTritTest	0.855151	0.504054	0.156279	+
LongestRunOfTrit	0.643037	0.105588	0.675400	+
NonOverlappingTemplateTritTest	0.330388	0.636565	0.931334	+
NonOverlappingTemplateTritTest	0.330388	0.330388	0.142906	+
NonOverlappingTemplateTritTest	0.330388	0.330388	0.636565	+
NonOverlappingTemplateTritTest	0.636565	0.330388	0.142906	+
OverlappingTemplateTritTest	0.636565	0.142906	0.931334	+

Згенерована послідовність TriGen-3:

101122001110022101012210101200002120210111210012110020100122021
220020201111212100202212200120112120010112220200012221010022110000202022200220220200001002022102211202110012020202121110102112 02.

Очікуваний результат – послідовність випадкова. Отримані результати перевірки див. табл. 4.10. Згідно таблиці послідовність є випадковою.

Таблиця 4.10

Результати перевірки згенерованої послідовності TriGen-3

Тест	$P_{value_{01}}$	$P_{value_{02}}$	$P_{value_{12}}$	Тест пройдений
FrequencyTritTest	0.169117	0.595883	0.397191	+
FrequencyTritTestWithinBlock	0.465066	0.878295	0.200873	+
RunsTritTest	0.933872	0.149234	0.432928	+
LongestRunOfTrit	0.907669	0.212677	0.482375	+
NonOverlappingTemplateTritTest	0.636565	0.931334	0.142906	+
NonOverlappingTemplateTritTest	0.330388	0.330388	0.019184	+
NonOverlappingTemplateTritTest	0.636565	0.636565	0.636565	+
NonOverlappingTemplateTritTest	0.636565	0.636565	0.636565	+
OverlappingTemplateTritTest	0.330388	0.330388	0.330388	+

Згенерована послідовності TriGen-4:

221110202101100012221001221120220220210202210220201021111012001
200012101022002021121120011122010200121121200210120022222022102100211
021200202111002011001011222101010201101212222011101101100021 2.

Очікуваний результат – послідовність випадкова. Отримані результати перевірки див. табл. 4.11. Згідно таблиці послідовність є випадковою.

Згенерована послідовність TriGen-5:

100120112111112122022011120000010211211012002100100212210110012
120002012000021202110110221222110112212011112221010110201201011211211
112012002212102121221020022122012112202020122211022202100120.

Очікуваний результат – послідовність випадкова. Отримані результати перевірки див. табл. 4.12. Згідно таблиці послідовність є випадковою.

Таблиця 4.11

Результати перевірки згенерованої послідовності TriGen-4

Тест	$P_{value_{01}}$	$P_{value_{02}}$	$P_{value_{12}}$	Тест пройдений
FrequencyTritTest	0.926960	0.855132	0.927565	+
FrequencyTritTestWithinBlock	0.962181	0.972575	0.415407	+
RunsTritTest	0.520545	0.067356	0.649955	+
LongestRunOfTrit	0.013365	0.164464	0.413815	+
NonOverlappingTemplateTritTest	0.636565	0.054678	0.931334	+
NonOverlappingTemplateTritTest	0.636565	0.330388	0.054678	+
NonOverlappingTemplateTritTest	0.931334	0.636565	0.931334	+
NonOverlappingTemplateTritTest	0.636565	0.142906	0.931334	+
OverlappingTemplateTritTest	0.330388	0.006313	0.330388	+

Таблиця 4.12

Результати перевірки згенерованої послідовності TriGen-5

Тест	$P_{value_{01}}$	$P_{value_{02}}$	$P_{value_{12}}$	Тест пройдений
FrequencyTritTest	0.084119	0.445600	0.332797	+
FrequencyTritTestWithinBlock	0.623271	0.685012	0.806029	+
RunsTritTest	0.998771	0.804540	0.378102	+
LongestRunOfTrit	0.598916	0.884853	0.131484	+
NonOverlappingTemplateTritTest	0.054678	0.142906	0.142906	+
NonOverlappingTemplateTritTest	0.636565	0.330388	0.142906	+
NonOverlappingTemplateTritTest	0.636565	0.931334	0.142906	+
NonOverlappingTemplateTritTest	0.019184	0.330388	0.330388	+
OverlappingTemplateTritTest	0.054678	0.330388	0.636565	+

Отже, в результаті перевірки п’яти завідомо не випадкових послідовності та п’яти завідомо випадкових послідовностей розроблений метод оцінки якості тритових ПВП (TritSTS) їх успішно визначив, що підтверджує його ефективність для застосування на практиці.

4.4 Висновки до четвертого розділу

1. Запропоновано методику проведення експериментального дослідження, що дозволила провести експериментальне дослідження запропонованих методів.

2. Проведено моделювання роботи протоколу КПБЗ із запропонованим та відомим методом забезпечення стійкості протоколів КК до некогерентних атак. Згідно з результатами, швидкість протоколу КПБЗ із запропонованим методом забезпечення стійкості мінімум у 1,52 раз при $r = 4$ краща за швидкість відомого методу, а оскільки у КК рекомендований розмір $r \geq 20$, в такому випадку швидкодія запропонованого методу мінімум краща у 4,4 рази.

3. Досліджено запропонований та відомий метод генерування тритових ПВП за методикою NIST STS. Дане дослідження показало, що дану методику не можливо використовувати для оцінки якості тритових послідовностей.

4. Досліджено запропонований та відомий метод генерування тритових ПВП за розробленим методом оцінки якості тритових послідовностей. В результаті, визначено, що запропонований метод генерування формує якісні ПВП та придатний для використання на практиці.

5. Перевірено адекватність запропонованого методу оцінки якості тритових ПВП (створений на його основі програний засіб TritSTS). Усі підібрані та згенеровані послідовності даний метод успішно протестував та коректно визначив їх випадковість.

ВИСНОВКИ

Результатом виконаної роботи є розв'язання актуальної науково-практичної задачі розробки і дослідження нових більш ефективних методів забезпечення стійкості тритових протоколів квантової криптографії до некогерентних атак, побудови тритових генераторів псевдовипадкових послідовностей, оцінювання їх якості та можливості використання для криптографічних застосувань.

У процесі виконання дисертаційної роботи отримані такі вагомі результати:

1. Проведено якісний аналіз сучасних методів та протоколів квантової криптографії, визначено їх переваги і недоліки, стійкість та уразливість до різного роду кібератак (зокрема, до некогерентних атак). На підставі часткових узагальнень теоретичних положень та практичних досягнень у галузі квантової криптографії, розроблено розширену класифікацію методів квантової криптографії, яка дозволяє розширити можливості щодо вибору відповідних методів для побудови сучасних квантових систем захисту інформації. Така класифікація також дала можливість чітко визначити завдання дисертаційного дослідження щодо подальшої розробки методів забезпечення стійкості тритових протоколів і систем, побудови тритових генераторів псевдовипадкових послідовностей та оцінювання їх якості.

2. Розроблено метод забезпечення стійкості тритових протоколів квантової криптографії, що не потребує великих часових та ресурсних затрат і, за рахунок неквантової функції перевірки цілісності та використання тритової симетричної функції, дозволяє підвищити швидкість роботи протоколу в 4,4 рази при збереженні стійкості до некогерентних атак.

3. Розроблено метод генерування псевдовипадкових послідовностей, який, за рахунок виконання нової послідовності операцій (підстановок $Sbox(X)$, лінійного розсіювання $Mix(X)$, динамічного циклічного зсуву і додавання за модулем $3\oplus$ та 3^l+) над вектором внутрішніх станів

V_p ($V_p = \{0,1,2\}^p$, $p = 14 \cdot l$) за $r \cdot b$ циклів, дозволяє формувати трійкові незбалансовані («0», «1», «2») псевдовипадкові послідовності $V_{m \cdot b}, m = 4 \cdot l$, що можуть використовуватись для реалізації запропонованого метода забезпечення стійкості кутритових протоколів квантової криптографії до некогерентних атак, а також для інших криптографічних застосувань в сучасних інформаційно-комунікаційних технологіях;

4. Розроблено метод оцінювання якості псевдовипадкових послідовностей, який, за рахунок комплексної інтерпретації згенерованих чисел, введення диференційованих ймовірностей $P-value_{01}$, $P-value_{02}$, $P-value_{12}$ і введення трійкових коефіцієнтів для функції помилок *erfc* та неповної гамма функції *igamc,* дає принципову можливість оцінювати загальноприйняті статистичні параметри та закономірності (FMT, FTBT, TRT, TTLROB, NTMTT та TOTMT) для тритових псевдовипадкових послідовностей і, відповідно, оцінювати криптостійкість тритових генераторів псевдовипадкових послідовностей та доцільність їх використання для криптографічних застосувань.

5. Розроблено спеціалізоване програмне забезпечення, за допомогою якого проведено експерименти, що дозволило верифікувати запропоновані методи з точки зору підвищення ефективності протоколів квантової криптографії і реалізації деяких процедур забезпечення безпеки в традиційних (неквантових) криптографічних системах захисту інформації. Розроблені програмні продукти захищені вітчизняними свідоцтвами про реєстрацію авторського права на твір.

6. Зазначені результати роботи впроваджено у діяльність ТОВ «Сайфер БІС» (акт впровадження від 28.10.2015 року) та Bilfinger HSG (Німеччина) (акт впровадження від 03.09.2015 року), Національного авіаційного університету (акт від 21.12.2015 року) та Казахського національного дослідницького технічного університету ім. К.І. Сатпаєва (акт від 07.12.2015 року), що підтверджено відповідними актами впровадження, які містяться у додатках до дисертаційної роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Алексеев Д.А. Практическая реальность квантово-криптографических систем распределения ключей / Д.А. Алексеев, А.В. Корнейко // Захист інформації. – 2007. – № 1. – С. 72–76.

2. Атаки в квантових системах захисту інформації / О.Г. Корченко, Є.В. Василіу, С.О. Гнатюк, В.М. Кінзерявий // Вісник інженерної академії України. – 2010. – № 2. – С. 109–115.

3. Брусенцов Н.П. Вычислительная машина «Сетунь» Московского государственного университета / Брусенцов Н. П. // Новые разработки в области вычислительной математики и вычислительной техники. Материалы научно-технической конференции. – Киев, 1960. – С. 226-234.

4. Вакарчук І. О. Квантова механіка : Підручник / І.О. Вакарчук. – 3-тє вид., доп. – Львів : ЛДУ ім. І. Франка., 2007. – 848 с.

5. Василиу Е.В. Анализ атаки двух злоумышленников на протокол квантовой прямой безопасной связи / Е.В. Василиу, С.В. Николаенко // Труды Северо-Кавказского филиала Московского технического университета связи и информатики. – 2013. – С.324–330.

6. Василиу Е.В. Анализ атаки на пинг-понг протокол с триплетами Гринбергера – Хорна – Цайлингера / Е.В. Василиу // Наукові праці ОНАЗ ім. О.С. Попова. – 2008. – № 1. – С. 15–24.

7. Василиу Е.В. Анализ атаки пассивного перехвата на пинг-понг протокол с полностью перепутанными парами кутритов / Е.В. Василиу, Р.С. Мамедов // Восточноевропейский журнал передовых технологий. – 2009. – № 4/2 (40). – С. 4-11.

8. Василиу Е.В. Безопасные системы передачи конфиденциальной информации на основе протоколов квантовой криптографии : монография / Е.В. Василиу, В.Я. Мильчевич, С.В. Николаенко, А.В. Мильчевич. – Харьков: Цифровая типография № 1, 2013.– 168 с.

9. Василиу Е.В. Пинг-понг протокол с трех- и четырехкубитными состояниями Гринбергера-Хорна-Цайлингера / Е.В. Василиу, Л.Н. Василиу // Труды Одесского политехнического университета – 2008. – Вып. 1(29). – С. 171-176.

10. Василиу Е.В. Синтез основанной на пинг-понг протоколе квантовой связи безопасной системы прямой передачи сообщений / Е.В. Василиу, С.В. Николаенко // Наукові праці ОНАЗ ім. О.С. Попова. — 2009, № 1. — С. 83-91.

11. Василиу Е.В. Сравнительный анализ эффективности и стойкости к некогерентным атакам квантовых протоколов распределения ключей с передачей многомерных квантовых систем / Е.В. Василиу, Р.С. Мамедов // Наукові праці ОНАЗ ім. О.С. Попова. – 2008. – № 2. – С. 20–27.

12. Василиу Е.В. Стойкость пинг-понг протокола с триплетами Гринбергера-Хорна-Цайлингера к атаке с использованием вспомогательных квантовых систем / Е.В. Василиу // Информатика : Объединенный институт проблем информатики НАН Беларуси. – 2009. – № 1 (21). – С. 117-128.

13. Василіу Є.В. Оцінки обчислювальної складності способу підсилення безпеки пінг-понг протоколу з переплутаними станами кубітів та кутритів / Є.В. Василіу, Р.С. Мамедов // Наукові праці ОНАЗ ім. О.С. Попова. – 2009. – № 2. – С. 14-25.

14. Гарасимчук О.І. Генератори псевдовипадкових чисел, їх застосування, класифікація, основні методи побудови і оцінка якості / Гарасимчук О.І., Максимович В.М. // Захист інформації. – 2003. – №3. – С. 29-36.

15. Гиршов Е. Введение в квантовую криптографию : основные понятия, протоколы, примеры, технологические аспекты [Електронний ресурс] / Е. Гиршов // Теоретический минимум по информатике: [Материалы учебных курсов по теоретической информатике, криптографии, информационному поиску]. – Режим доступа: http://www.teormin.ifmo.ru/courses/intro/38.pdf. – Дата останнього доступу: 05.08.2015. – Назва з екрану.

16. Гнатюк С.А. Квантовые протоколы защиты информации в информационно-коммуникационных системах / С.А. Гнатюк, Т.А. Жмурко, М.А. Рябый // Стан та удосконалення безпеки інформаційно-телекомунікаційних сис-

тем (SITS’2014): 6 всеукраїнська наук.-практ. конф., 09-12 вересня 2014 р. : тези доп. – Миколаїв – Коблево, 2014 – С. 59-62.

17. Гнатюк С.О. Використання тритових псевдовипадкових послідовностей в криптографії / С.О. Гнатюк, Т.О. Жмурко // Інтегровані інтелектуальні робототехнічні комплекси (ІІРТК-2011) : IV міжнар. наук.-практ. конф., 23-25 травня 2011 р. : тези доп. – К., 2011. – С. 414-416.

18. Гнатюк С.О. Генерування та оцінка випадкових послідовностей для підвищення ефективності кутритових квантових криптосистем / С.О. Гнатюк, Т.О. Жмурко // Інтегровані інтелектуальні робототехнічні комплекси (ІІРТК-2012) : V міжнар. наук.-практ. конф., 15-16 травня 2012 р. : тези доп. – К., 2012. – С. 305-307.

19. Гнатюк С.О. Квантові технології: загроза чи захист інформаційних систем? / С.О. Гнатюк, Т.О. Жмурко // Управління знаннями та конкурентна розвідка: міжнар. конф., 14-16 квітня 2014 р.: тези доп. – Х., 2014. – С. 75-76.

20. Гнатюк С.О. Метод генерування та оцінки випадкових послідовностей для кутритових систем квантового прямого безпечного зв’язку / С.О. Гнатюк, Т.О. Жмурко // Безпека інформаційних технологій (ITSEC-2012) : ІІ наук.-техн. конф., 24-25 квітня 2012 р. : тези доп. – К., 2012. – С. 16-17.

21. Гнатюк С.О. Метод генерування тритових псевдовипадкових послідовностей для систем квантової криптографії / С.О. Гнатюк, Т.О. Жмурко, В.М. Кінзерявий, Н.А. Сєйлова // Безпека інформації – 2015. – Т. 22. – № 2. – С. 140-147.

22. Гнатюк С.О. Метод підвищення захищеності систем захисту інформації на базі квантових технологій / С.А. Гнатюк, Т.А. Жмурко, А. Стояновіч, Н.А. Сєйлова // Стан та удосконалення безпеки інформаційно-комунікаційних систем (SITS’2015) – Миколаїв: 2015. – С. 93-96.

23. Гнатюк С.О. Метод формування трійкових псевдовипадкових послідовнсотей / С.О. Гнатюк, Т.О. Жмурко, В.М. Кінзерявий, Р.С. Одарченко // Перспективні напрями захисту інформації: матеріали першої всеукр. наук.-пр. конф. м. Одеса, 7-9 вересня 2015 р. – Одеса: ОНАЗ, 2015. – С. 11-15.

24. Гомонай О.В. Лекції з квантової інформатики : Навчальний посібник / О.В. Гомонай. – Вінниця : О. Власюк, 2006. – С. 62.

25. Горбенко І.Д. Обґрунтування вимог до генераторів випадкових бітів згідно ISO/IEC 18031 / Горбенко І.Д., Шапочка Н.В., Козулін О.О. // Радіоелектронні і комп'ютерні системи. – 2009. – № 6 (40). – С. 94-97.

26. Гроздов Г.Г. Современные методы криптографической защиты информации : Обзор по материалам открытой печати / Г.Г. Гроздов. – К.: 2002. – 32 с. – (Препринт /НАН Украины. Отделение гибридных моделирующих и управляющих систем в энергетике ИПМЭ ; № 1/2001).

27. Євсеєв С.П. Аналіз сучасних методів формування псевдовипадкових послідовностей / С.П. Євсеєв, Р.В. Корольов, М.В. Краснянська // Восточно-Европейский журнал передовых технологий. – 2010. – 3/4 (45). – С. 11-15.

28. Жмурко О.І. Моделі до освоєння регресії / О.І. Жмурко, Т.О. Жмурко // Актуальні проблеми математики, фізики і технологічної освіти. Зб. наук. пр.– Вінниця: ФОП Данилюк В.Г., 2012. – Вип. № 9 – С. 58-62.

29. Жмурко О.І. Моделювання задачі Бюффона – ефективний шлях до розуміння випадкових величин / О.І. Жмурко, Т.О. Жмурко // Актуальні проблеми математики, фізики і технологічної освіти. Зб. наук. пр.– Вінниця: ФОП Данилюк В.Г., 2012. – Вип. № 9 – С. 62-67.

30. Жмурко Т.О. Оцінка рівня випадковості трійкових послідовностей / Т.О. Жмурко // Наукоємні технології: наук.-техн. конф. студ. та мол. учених, 14-18 листопада 2011 р. : тези доп. – К., 2012. – С. 15.

31. Жмурко Т.О. Протоколи квантової теорії ігор /Т.О. Жмурко // Політ. Сучасні проблеми науки. Комп'ютерні технології: тези доповідей XIV між нар. наук.-практ. конф. молодих учених і студентів, м. Київ, 2-3 квітня 2014 р., НАУ / редкол.: М.С. Кулик [та ін.]. – С. 6.

32. Замула О.А. Аналіз алгоритмів формування псевдовипадкових послідовностей для систем зв'язку з кодовим розподілом каналів / О.А. Замула, В.І. Грабчак // Системи обробки інформації. – 2007. – № 5 (63). – С. 47-50.

33. Иванов М.А. Теория, применение и оценка качества генераторов псевдослучайных последовательностей / М.А. Иванов, И.В. Чугунков. – М.: КУДИЦ-ОБРАЗ, 2003. – 240 с.

34. Имре Ш. Квантовые вычисления и связь : Инженерный подход / Ш. Имре, Ф. Балаж ; Пер. с англ. под. ред. В.В. Самарцева. – М. : ФИЗМАТЛИТ, 2008. – 320 с.

35. Імітаційна модель пінг-понг протоколу з парами переплутаних кутритів у квантовому каналі з шумом / О.Г. Корченко, Є.В. Василіу, С.О. Гнатюк, В.М. Кінзерявий // Захист інформації. – 2010. – № 3. – С. 46–56.

36. Калугин А.Н. Модификация многомерных псевдослучайных последовательностей с использованием двойственных LFSR-CNS генераторов / А. Н. Калугин // Компьютерная оптика. – 2005. – № 28. – С. 112-118 .

37. Килин С.Я. Квантовая информация / С.Я. Килин // Успехи физических наук. – 1999. – Том 169. – № 5. – С. 507–526.

38. Килин С.Я. Квантовая криптография: идеи и практика / Килин С.Я., Хорошко Д.Б., Низовцев А.П. – Минск: ИД «Белорусская наука», 2008. – 392 с.

39. Кнут Д.Э. Искусство программирования / Д.Э. Кнут. – 3-е изд. – М.: Вильямс, 2000. – 832.

40. Комп’ютерна програма «GenSBOX3» : Свідоцтво про реєстрацію авторського права на твір № 48037 від 26.02.2013 / О.Г. Корченко, В.М. Кінзерявий, О.М. Кінзерявий, С.О. Гнатюк, В.О. Гнатюк, Т.О. Жмурко ; Ю.Є. Хохлачова, Національний авіаційний університет. – К. : Державна служба інтелектуальної власності України, 2013.

41. Комп’ютерна програма «Model Ping-pong protocol» : Свідоцтво про реєстрацію авторського права на твір № 48041 від 26.02.2013 / О.Г. Корченко, В.М. Кінзерявий, С.В. Казмірчук, Т.О. Жмурко, С.О. Гнатюк, А.І. Гізун, О.М. Кінзерявий; Національний авіаційний університет. – К. : Державна служба інтелектуальної власності України, 2013.

42. Комп’ютерна програма «TrytTon 2012» : Свідоцтво про реєстрацію авторського права на твір № 48040 від 26.02.2013 / О.Г. Корченко, В.М. Кінзерявий, О.М. Кінзерявий, В.О. Гнатюк, С.О. Гнатюк, Т.О. Жмурко, С.В. Казмірчук; Національний авіаційний університет. – К. : Державна служба інтелектуальної власності України, 2013.

43. Комп'ютерний програмний комплекс «Імітаційна модель пінг-понг протоколу в квантовому каналі з шумом» : Свідоцтво про реєстрацію авторського права на твір № 36373 від 04.01.2011 / О.Г. Корченко, Є.В. Василіу, В.М. Кінзерявий, С.О. Гнатюк, Т.О. Жмурко ; Національний авіаційний університет. – К. : Державна служба інтелектуальної власності України, 2011.

44. Корольов Р.В. Дослідження періодичних властивостей генераторів псевдовипадкових чисел, заснованих на використанні надмірних блокових кодів / Р.В. Корольов // Системи озброєння і військова техніка. – 2008. – № 3(15). – С.126-128.

45. Корченко О.Г. Сучасні квантові технології захисту інформації / О.Г. Корченко, Є.В. Василіу, С.О. Гнатюк // Захист інформації. – 2010. – № 1. – С. 77–89.

46. Котенко В.В. Теория информации и защита телекоммуникации: монография / В.В. Котенко, К.Е. Румянцев. – Ростов-на-Дону : Изд-во «ЮФУ», 2009. – 369 с.

47. Кренгель Е. И. Исследование и разработка новых классов псевдослучайных последовательностей и устройств их генерации для систем с кодовым разделением каналов: дис. канд. техн. наук: 05.12.13 / Кренгель Евгений Ильич. – М., 2002. – 214 с.

48. Кузнецов А.А. Усовершенствованный метод быстрого формирования последовательностей псевдослучайных чисел / А.А. Кузнецов, Р.В. Королёв, Ю.Н. Рябуха // Кібернетика та системний аналіз. – 2008. – № 3(18). – С. 101-105.

49. Кузнецов О.О. Метод швидкого формування послідовностей псевдовипадкових чисел доказової стійкості / О.О. Кузнецов, Р.В. Корольов, Ю.М. Рябуха // Системи озброєння і військова техніка. – 2008. – № 4(16). – С. 141-147.

50. Кузнецова А.В. Стратегии атак на квантовые протоколы защиты информации. – Цифрові технології. – № 14. – 2013. – С. 134-137.

51. Лимарь И.В., Василиу Е.В. Классификация квантовых технологий разделения секрета / Захист інформації. – Том 16. – №3. – 2014 – С. 201-214.

52. Макарычев А. Битва за скорость: троичная логика против двоичной [Електронний ресурс]. – Режим доступа: https://www.societyforscience.org/. – Дата останнього доступу: 05.08.2015. – Назва з екрану.

53. Малая цифровая вычислительная машина «Сетунь» / Брусенцов Н.П., Маслов С.П., Розин В.П., Тишулина А.М. – М.: Изд-во МГУ, 1962. – 140 с.

54. Матвеева Е.Л. Основы научных исследований : Конспект лекций / Е.Л. Матвеева. – К. : КМУГА, 1999. – 128 с.

55. Метод оцінювання якості тритових псевдовипадкових послідовностей для криптографічних застосувань / С.О. Гнатюк, Т.О. Жмурко, В.М. Кінзерявий, Н.А. Сєйлова // Information technology and Security. – 2015. – Vol. 3. – №2(5). – С. 108-116.

56. Методы перехвата информации в информационно-коммуникационных системах на основе квантових технологий / А.Г. Корченко, Е.В. Василиу, Т.А. Жмурко, С.А. Гнатюк // Информационные технологии и системы в управлении, образовании, науке: Монография [под. ред. В.С. Пономаренко]. – Харків: Цифрова друкарня № 1, 2013. – С. 98-110.

57. Методы перехвата информации в информационно-коммуникационных системах на основе квантових технологий / А.Г. Корченко, Е.В. Василиу, С.А. Гнатюк, Т.А. Жмурко // Проблеми і перспективи розвитку ІТ-індустрії : V міжнар. наук.-практ. конф., 25-26 квітня 2013 р. : тези доп. – Харків, 2013. – С. 204.

58. Милантьев В.П. История возникновения квантовой механики и развитие представлений об атоме. – М.: ЛИБРОКОМ, 2009. – 248 с.

59. Молотков С.Н. О коллективной атаке на ключ в квантовой криптографии на двух неортогональных состояниях / С.Н. Молотков // Письма в ЖЭТФ. – 2004. – Т. 80, вып.8. – С. 639–644.

60. Назаров Є.О. Генератори псведовипадкових послідовностей для криптографічних систем / Назаров Є.О., Чернишова А.В., Губенко Н.Є.// збірник наукових праць міжнародної науково-технічної конференції «Інформатика і компютерні технології -2012». – ДонНТУ: 2012. – С. 139-144.

61. Нильсен М. Квантовые вычисления и квантовая информация / М. Нильсен, И. Чанг. — М.: Мир, 2006. – 824 с.

62. Новий метод підсилення секретності пінг-понг протоколу з парами переплутаних кутритів / В.М. Кінзерявий, Є.В. Васіліу, С.О. Гнатюк, Т.О. Жмурко// Захист інформації. – 2012. – №2 (55). – С. 5–13.

63. Основи методології та організації наукових досліджень: Навч. посіб. для студентів, курсантів, аспірантів і ад'юнтів / за ред. А. Є. Конверського. – К.: Центр учбової літератури, 2010. – 352 с.

64. Основи технічної творчості та наукових досліджень: Навч. Посіб. / А.В. Журахівський, А.Я. Яцейко, Н.Б. Дьяченко. – Львів: Видавництво Львівської політехніки, 2012. – 380 с.

65. Оцінка корегувальної здатності завадостійких трійкових РС-кодів при передачі інформації повністю переплутаними станами кутритів квантовим каналом із шумом / О.Г. Корченко, Є.В. Васіліу, С.О. Гнатюк, В.М. Кінзерявий // Захист інформації. – 2010. – № 4. – С. 44–53.

66. Пат. 2325039 РФ, H04L9/00 (2006.01). Спосіб кодування та передачі криптографічних ключів / Молотков С., Кулик С. – № 2006119652 ; 06.06.2006.

67. Пат. 6438234 США, H04L 9/08 (20060101), H04K 001/00. Пристрій та метод квантової криптографії / Gisin N., Zbinden H. – 20.08.2002.

68. Пат. 6748081 США, H04L 9/08 (20060101), C09K 19/02 (20060101), G02F 1/13 (20060101). Квантова система криптографії для секретного розподілу довільних ключів, використовуючи метод поляризації / W. Dultz. – 08.06.2004.

69. Пат. 6895092 США, H04L 9/08 (20060101), G06F 017/00. Метод розподілу криптографічних ключів та його апарат / Tomita. – 17.05.2005.

70. Пат. 7178277 США, H04K 1/00 (20060101). Квантова криптографічна комунікаційна система і метод поширення криптографічних ключів, що у ній використовується / Takeuchi. – 20.02.2007.

71. Пат. 7461323 США, H03M 13/00 (20060101). Метод квантового розподілу ключів і комутативний пристрій / Matsumoto. – 02.12.2008.

72. Пат. 7539308 США, H04K 1/00 (20060101). Квантова стеганографія / Conti, Ralph S., Kenneth A. et al. – 21.05.2004.

73. Потий А.В. Статистическое тестирование генераторов случайных и псевдослучайных чисел с использованием набора статистических тестов NISTSTS / А.В. Потий, С.Ю. Орлова, Т.А. Гриненко // Правове, нормативне та метрологічне забезпечення систем захисту інформації в Україні. – 2001. – Вип. 2. – С. 206-214.

74. Проценко О.Г. Тестування генераторів псевдовипадкових чисел систем програмування на основі стандарту FIPS 140-1/ О.Г. Проценко. І.В. Лисенко // Системи обробки інформації. – 2010 р. – № 2(83). – С. 130-132.

75. Рисований О.М. Генератор псевдовипадкових послідовностей по модулю 3 з різною частотою формування псевдовипадкових послідовностей / О.М. Рисований, В.В. Гоготов // Системи обробки інформації – 2010 р. – № 2 (83). – С. 141-143.

76. Розова Я.С. Классификация атак на каналы квантового распределения ключей / Я.С. Розова // Сборник трудов конференции молодых ученых, В. 6. Информационные технологии. – СПб : СПбГУ ИТМО, 2009. – С. 167–172.

77. Розширення номенклатури методів квантової криптографії та зв'язку / Ахмєтов Б.С., Кінзерявий В.М, Жмурко Т.О., Юбузова Х.І. // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: ІІ міжн. наук.-практ. конф., 24-27 лютого 2016 р.: тези доп. – К.: Видавництво Європейського університету, 2016. – С. 11-14.

78. Румянцев К.Е. Квантовая криптография : принципы, протоколы, системы / К.Е. Румянцев, Д.М. Голубчиков // Всероссийский конкурсный отбор обзорно-аналитических статей по приоритетному направлению «Информационно-телекоммуникационные системы», 2008. – 37 с.

79. Рябуха Ю.М. Метод формування псевдовипадкових послідовностей максимального періоду із використанням модулярних перетворень / Ю.М. Рябуха // Системи озброєння і військова техніка. – 2009. – № 4(20) . – С. 166-169.

80. Система передачі криптографічних ключів: пат. 43779 України, МПК H04L 9/08. / Корченко О.Г., Паціра Є.В., Гнатюк С.О., Кінзерявий В.М. ; заявник та патентовласник Національний авіаційний університет. – № u200904239 ; заявл. 29.04.2009 ; опубл. 25.08.2009, Бюл. № 16. – 8 с.

81. Скобелев В.Г. Анализ атак на квантовый протокол передачи ключа / В.Г. Скобелев // Прикладная дискретная математика. – 2008. – № 2 (2). – С. 62-66.

82. Слепов Н. Квантовая криптография: передача квантового ключа. Проблемы и решения / Н. Слепов // Электроника: наука, технология, бизнес. – 2006. – № 2. – С. 54–61.

83. Современные задачи криптографии [Електронний ресурс] / Лифшиц Ю. // Курс лекций. 2005. Лекция 9: Псевдослучайные генераторы – Режим доступа: http://yury.name/crypto/09cryptonote.pdf. – Дата останнього доступу: 05.08.2015. – Назва з екрану.

84. Спосіб криптографічного перетворення інформації: пат. 45776 України, МПК H04L 9/06. / Корченко О.Г., Паціра Є.В., Кінзерявий В.М., Гнатюк С.О. ; заявник та патентовласник Національний авіаційний університет. – № u200905972 ; заявл. 10.06.2009 ; опубл. 25.11.2009, Бюл. № 22. – 8 с.

85. Спосіб підсилення безпеки пінг-понг протоколу квантового безпечного зв’язку: пат. 59732. / П.П. Воробієнко, Є.В. Васіліу, С.В. Ніколаєнко; заявник і патентовласник Одеська національна академія зв’язку ім. О.С. Попова. – заявл. 19.11.2010; опублік. 25.05.2011, Бюл. № 10. – 8 с.

86. Спосіб підсилення безпеки протоколів квантового прямого безпечного зв’язку // Заявка на отримання патенту України на корисну модель, № u201512445 від 16.12.2015.

87. Стин Э. Квантовые вычисления / Э. Стин ; Пер. с англ. И.Д. Пасынкова. – Москва – Ижевск : Изд. НИЦ «Регулярная и хаотическая динамика», 2000. – 111 с.

88. Узагальнена класифікація сучасних методів квантової криптографії та зв’язку / Т.О. Жмурко, В.М. Кінзерявий, Х.І. Юбузова, А.С. Стояновіч // Безпека інформації. – 2015. – № 3. – Т. 22. – С. 287-293.

89. Усенко В.К. Застосування двомодових когерентно-корельованих променів в квантовій криптографії : Автореф. дис. на здобуття наук. ступеня канд. фіз.-мат. наук : 01.04.02 / В.К. Усенко ; НАН України. Ін-т теорет. фізики ім. М.М. Боголюбова. – К., 2006. – 18 с.

90. Физика квантовой информации : Квантовая криптография. Квантовая телепортация. Квантовые вычисления / Ред. Д. Боумейстер [и др.] ; Пер. с англ. С.П. Кулик, Е.А. Шапиро ; Ред. пер. С.П. Кулик, Т.А. Шмаонов. – М. : Постмаркет, 2002. – С. 33–73.

91. Харин Ю.С. Математические основы криптологии: учебное пособие / Ю.С.Харин, В.И. Берник, Г.В.Матвеев. – Минск.: БГУ, 1999. – 319 с.

92. Харин Ю.С. Методы и алгоритмы статистического тестирования генераторов случайных и псевдослучайных последовательностей в системах информационной безопасности / Ю.С. Харин, А.Н. Ярмола, А.И. Петлицки// Штучний інтелект. – 2006. – №3. – С. 793-803.

93. Холево А.С. Введение в квантовую теорию информации / А.С. Холево. – Москва: МЦНМО, 2002. – 228 с.

94. Чевардин В.Е. Построение генераторов псевдослучайных чисел на основе арифметики эллиптических кривих / В.Е. Чевардин // Системи управління, навігації та зв'язку. – 2010. – № 2(14). — С.148-151.

95. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайер. – М.: Триумф, 2002. – 816 с.

96. Юринець В.Є. Методологія наукових досліджень: навч. посібник / В.Є. Юринець. – Львів : ЛНУ імені Івана Франка, 2011. – 178 с.

97. Яковлев М.А. Повышение эффективности оценочных тестов для псевдослучайных последовательностей / М. А. Яковлев, И. В. Чугунков // Информационные технологии. – 2009.– № 2. — С. 63-65.

98. A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications. NIST Special Publication 800-22.— May 15, 2001. — 164 p.

99. Advanced Encryption Standard (AES) [Electronic resource] : FIPS 197. – Electronic data (1 file: 279 457 byte). – NIST, 2001. – Mode of access: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf. – [Accessed Jul. 5, 2015]. – Description based on screen.

100. Alleaume R. SECOQC White Paper on Quantum Key Distribution and Cryptography / Romain Alleaume, Jan Bouda, Cyril Branciard [et al.] // ArXiv.org [Online] – 2007. – Mode of access: http://arxiv.org/abs/quant-ph/0701168v1.

101. Bennett C. Experimental quantum cryptography / C.H. Bennett, G. Brassard, J. Smolin // Journal of Cryptology. – 1992. – № 5(1). – P. 14.

102. Bennett C. Quantum cryptography using any two non-orthogonal states / C. Bennett // Physical Review Letters. – 1992. – Vol. 68. – P. 3121-3124.

103. Bennett C. Quantum cryptography using any two non-orthogonal states / C. Bennett // Physical Review Letters. – 1992. – Vol. 68, Iss. 21. – P. 3121-3124.

104. Bennett C. Quantum cryptography: public key distribution and coin tossing / Bennett C., Brassard G. // Proceedings of the IEEE International Conference on Computers, Systems and Signal Processing. – 1984. – P. 175-179.

105. Bennett C., Brassard G. An Update on Quantum Cryptography / Bennett C., Brassard G. // Advances in Cryptology: Proceedings of CRYPTO 84. – 1985. – Vol.196. – P.475-480.

106. Blum L. A Simple Unpredictable Pseudo-Random Number Generator / L. Blum, M. Blum, M. Shub // SIAM Journal on Computing. – 1986. – № 15. – P. 364-383.

107. Blum L. Comparison of two pseudo-random number generators/ Blum, M. Blum, M. Shub // Advances in Cryptology: Proceedings of Crypto 82. – 1983. – P. 61-68.

108. Blum M. How to generate cryptographically strong sequences of pseudo-random bits / Blum M., Micali S. // SIAM Journal on Computing. – 1984. – № 4(13). – P. 850-864.

109. Bostrom K. Deterministic secure direct communication using entanglement / Bostrom K., Felbinger T. // Physical Review Letters. – 2002. – Vol. 89, Iss. 18. – P. 187902.

110. Bovino F.A. Practical Quantum Cryptography: The Q-KeyMaker / Bovino F.A., Giardina M. // ArXiv.org [Online] – 2011. – Mode of access: http://arxiv.org/abs/1104.2475.

111. Brassard G. Limitations on practical quantum cryptography / Brassard G., Lutkenhaus N., Mor T., Sanders B. // Physical Review Letters. – 2000. – Vol. 85, Iss. 6. – P. 1330-1333.

112. Bruss D. Optimal Eavesdropping in Quantum Cryptography with Six States / D. Bruss // Physical Review Letters. – 1998. – Vol. 81, Iss. 14. – P. 3018-3021.

113. Cai Q.-Y. Improving the capacity of the Bostrom-Felbinger protocol / Q.-Y. Cai, B.-W. Li // Physical Review Letters. – 2004. – Vol. 69, Iss. 5. – P. 054301.

114. Cai Q.-Y. The «Ping-Pong» Protocol Can Be Attacked without Eavesdropping / Q.-Y. Cai // Physical Review A. – 2003. – Vol. 91, Iss. 10. – P. 109801.

115. Calsamiglia J. Conditional beam splitting attack on quantum key distribution / J. Calsamiglia, S.M. Barnett, N. Lutkenhaus // Physical Review A. – 2001. – Vol. 65, Iss. 1. – P. 012312.

116. Cerberis Encryption Solution [Electronic resource] : Layer 2 Encryption with Quantum Key Distribution. – Electronic data. – Geneva : ID Quantique SA, 2010. – Mode of access: http://www.idquantique.com/products/cerberis.htm. – Description based on screen.

117. Cerf N. Security of quantum key distribution using d-level systems / Cerf N., Bourennane M., Karlsson A., Gisin N. // Physical Review Letters. – 2002. – Vol. 88, Iss.№ 12. – 127902.

118. Cerf N.J., Leuchs G., Polzik E.S. Quantum information with CV of atoms and Light. – Imperial College Press: 2007. – 604 p.

119. Chamoli A. Secure direct communication based on ping-pong protocol / A. Chamoli, C. M. Bhandari // Quantum Information Processing. – 2009. – Vol. 8, Iss. 4. – P. 347-356.

120. Chenmiao W. A complete Classification of Quantum Public-key Encryption Protocols / Chenmiao W., Li Ya. // SPIE Proceedings. – 2015. – Vol. 9648. – P. 8.

121. Chong X. The Classification of Quantum Symmetric-Key Encryption Protocols / Chong X., Li Y., Yong P., Dongqing Chen // Proceedings of SPIE. – 2015. – Vol.1. – P. 9269

122. Chuan W. Multi-step quantum secure direct communication using multi-particle Greenberg-Horne-Zeilinger state / Chuan W., Fu Guo D., Gui Lu L. // Optics Communications. – 2005. – Vol. 253. – P. 15-19.

123. Connelly J. Ternary Computing Testbed: 3-Trit Computer Architecture / Jeff Connelly, Chirag Patel, Antonio Chavez // Cal. Poly., San Luis Obispo. – 2008. – 193 p.

124. Curty M. Quantum steganography / Curty M., Santos D. J. // Proceedings of 2nd Bielefeld Workshop on Quantum Information and Complexity. – 2000. – P. 12.

125. Cutolo A., Mignani A.G., Tajani A. Photonics for safety and security, World Scientific Publishing Co. Pte. Ltd. – Singapore: 2014. – 422 p.

126. D'Ariano G.M. Quantum bit commitment: a complete classification of protocols / G.M. D'Ariano // ArXiv.org [Online] – 2002. – Mode of access: http://arxiv.org/abs/quant-ph/0209150.

127. Deng F.-G. Bidirectional quantum key distribution protocol with practical faint laser pulses / Fu-Guo Deng, Gui Lu Long // Physical Review A. – 2004. – Vol. 70. – P. 012311.

128. Deng F.G. Improving the security of multiparty quantum secret sharing against Trojan horse attack / Deng F.G., Li X.H., Zhou H.Y., Zhang Z.J. // Physical Review A. – 2005. – Vol. 72. – P. 044302.

129. Deng F.-G. Robustness of two-way quantum communication protocols against Trojan horse attack / F.-G. Deng, P. Zhou, X.-H. Li [et al.] // [// ArXiv.org [Online] – 2011. – Mode of access: http://arxiv.org/abs/quant-ph/0508168.

130. Deng F.-G. Two-step quantum direct communication protocol using the Einstein-Podolsky-Rosen pair block / Fu-Guo Deng, Gui Lu Long, Xiao-Shu Liu // Physical Review A. – 2003. – Vol. 68. – P. 042317.

131. Deutsch D. Quantum theory, the Church-Turing principle, and the universal quantum computer / Deutsch D. // Proc. Roy. Soc. of Lond. A, Mathematical and Physical Sciences. – 1985. – Vol. 400, № 1818. – P.97-117.

132. Dusek M. Generalized beam-splitting attack in quantum cryptography with dim coherent states / Dusek M., Haderka O., Hendrych M. // Opt. Commun. – 1999. – Vol. 169. – P. 103-108.

133. Eisert J. Quantum games and quantum strategies / Eisert J., Wilkens M., Lewenstein M. // Physical Review Letters. – 1999. – Vol. 83, Iss. 15. – P. 3077-3080.

134. Ekert A. Quantum cryptography based on Bell's theorem / Ekert A. // Physical Review Letters. – 1991. – Vol. 67, Iss.6 – P. 661–663.

135. Elliot C. Quantum Cryptography in Practice / Elliot C., Pearson D., Troxel G. // Proceeding of SIGCOMM '03. – 2003. – P. 227-238.

136. Flitney A.P. An introduction to quantum game theory / Flitney A.P., Abbott D. // Fluctuation and Noise Letters. – 2002. – Vol. 02, Iss. 04. – P. 175-187.

137. Gao T. Deterministic secure direct communication using GHZ states and swapping quantum entanglement / T. Gao, F.-L. Yan, Zh.-X. Wang // Journal of Physics A. – 2005. – Vol. 38, № 25. – P. 5761–5770.

138. Gao T. Quantum secure conditional direct communication via EPR pairs / T. Gao, F.-L. Yan, Zh.-X. Wang // International Journal of Modern Physics C. – 2005. – Vol. 16, № 8. – P. 1293-1301.

139. Gisin, N. Quantum cryptography / N. Gisin, G. Ribordy, W.Tittel, H. Zbinden // Rewiews of Modern Physics. – 2002. – Vol. 74, №1. – P. 145-195.

140. Gnatyuk S. Efficiency increasing method for quantum secure direct communication protocols / S. Gnatyuk, T. Zhmurko, P. Falat // Proceedings of the 2015 IEEE 8th International Conference on «Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications» (IDAACS’2015), Warsaw, Poland, September 24-26, 2015: Vol. 1. – P. 468-472.

141. Gnatyuk S.O. Contemporary Commercial Quantum Information Security Systems / S.O. Gnatyuk, T.O. Zhmurko, M.O. Riabyi // Proceedings of the CSE-2013. – 2013. – P. 74-77.

142. Goldenberg L. Quantum Cryptography Based On Orthogonal States / Goldenberg L., Vaidman L. // Physical Review Letters. – 1995. – Vol. 75, № 7. – P. 1239-1243.

143. Google and NASA Launch Quantum Computing AI Lab [Electronic resource] : MIT Technology Review. – Electronic data. – Big Sandy : MIT, 2013. – Mode of access: http://www.technologyreview.com/news/514846/google-and-nasa-launch-quantum-computing-ai-lab. – Description based on screen.

144. Gottesman D. Quantum digital signatures / Gottesman D., Chuang I. // ArXiv.org [Online] – 2001. – Mode of access: http://arxiv.org/abs/quant-ph/0105032v2.

145. Grover L.K. A fast quantum mechanical algorithm for database search // Proceedings of the 28th Annual ACM Symposium on the Theory of Computing. – 1996. – P. 212-219.

146. Hassanpour S. Bidirectional quantum teleportation and secure direct communication via entanglement swapping / Hassanpour S., Houshmand M. // ArXiv.org [Online] – 2014. – Mode of access: http://arxiv.org/abs/1411.0206.

147. Hirota O. An immunity against correlation attack on quantum stream cipher by Yuen 2000 protocol / Hirota O., Kurosawa K. // ArXiv.org [Online] – 2006. – Mode of access: http://arxiv.org/abs/quant-ph/0604036v1.

148. Hirota O. Quantum stream cipher by the Yuen 2000 protocol: Design and experiment by an intensity-modulation scheme / Hirota O., Sohma M., Fuse M., Kato K. // Physical Review A. – 2005. – Vol. 72, Iss. 2. – P. 022335.

149. Holevo A. Some estimates of the information transmitted by quantum communication channels // Probl. Inform. Trans. – 1973. – Vol.9, № 3. – P.177-183.

150. Holevo A.S. Problems in the mathematical theory of quantum communication channels / A.S. Holevo // Rep. Math. Phys. – 1977. – V.12, № 2. – P. 273-278.

151. Huttner B. Quantum Cryptography with Coherent States/ Huttner B., Imoto N., Gisin N., Mor T. // Physical Review A. – 1995. – Vol. 51, Iss. 3. – P. 1863-1869.

152. ID Quantique SA, Cerberis Encryption Solution: Layer 2 Encryption with Quantum Key Distribution [Online] Available: http://www.idquantique.com/products/cerberis.htm. [Accessed Aug. 5, 2015].

153. Improved Method of Routing in UAV Network / R. Odarchenko, O. Tkalich, S. Gnatyuk, T. Zhmurko // Proceedings of the APUAVD-2015. – 2015. – P. 145-150.

154. Inoue K. Differential phase shift quantum key distribution / Inoue K., Waks E., Yamamoto Y. // Physical Review Letters. – 2002. – Vol. 89, Iss. 3. – P. 037902.

155. Introduction to Quantum Cryptography [Electronic resource]. By Xiaoqing Tan. – Electronic data. – InTechOpen, 2013. – Mode of access: http://www.intechopen.com/books/theory-and-practice-of-cryptography-and-network-security-protocols-and-technologies/introduction-to-quantum-cryptography. – [Accessed Jul. 5, 2015]. – Description based on screen.

156. Jain N. Trojan-horse attacks threaten the security of practical quantum cryptography. / Jain N., Anisimova E., Khan I. et al. // ArXiv.org [Online] – 2011. – Mode of access: http://arxiv.org/abs/1406.5813.

157. Kendall M. Tables of Random Sampling Numbers / Kendall M., Babington-Smith B. – Cambridge: University Press, 1939. – 60 p.

158. Khan M. High error-rate quantum key distribution for long-distance communication / Khan M., Murphy M., Beige A. // ArXiv.org [Online] – 2009. – Mode of access: http://arxiv.org/abs/0901.3909.

159. Kinzeryavyy V.M. Improvement of ping-pong protocol with many-qubit GHZ-states / V.M. Kinzeryavyy, T.O. Zhmurko, S.O. Gnatyuk // Proceedings of the AVIA-2013. – 2013. – P. 2.22-2.26.

160. Koashi M. Quantum Cryptography Based on Split Transmission of One-Bit Information in Two Steps / Koashi M., Imoto N. // Physical Review Letters. – 1997. – Vol. 79, Iss. 12. – P. 2383-2386.

161. Korchenko O. Quantum Secure Telecommunication Systems / Korchenko O., Vasiliu Ye., Gnatyuk S. et al. // Telecom-munications Networks – Current Status and Future Trends (ed. by J.H. Ortiz). – InTech, 2012. – P. 211-236.

162. Korchenko O. Modern quantum technologies of information security against cyber-terrorist attacks / O. Korchenko, Y. Vasiliu, S. Gnatyuk // Aviation. Vilnius : Technika. – 2010. – Vol. 14, Iss. 2. – P. 58–69.

163. Leaw J.N. Strategic Insights From Playing the Quantum Tic-Tac-Toe / Leaw J.N., Cheong S.A. // Journal of Physics A: Mathematical and Theoretical. – 2010. – Vol. 43, Iss. 45. – P. 455304.

164. Li Q. Semi-quantum secret sharing using entangled states / Li Q., Chan W. H., Long D.-Y. // Physical Review A. – 2010. – Vol. 82, Iss. 2. – P. 022303.

165. Li X.-H. Improving the security of secure direct communication based on the secret transmitting order of particles / X.-H. Li, F.-G. Deng, H.-Y. Zhou // Physical Review A. – 2006. – Vol. 74, Iss. 5. – P. 054302.

166. Lutkenhaus N. Quantum key distribution with realistic states : photon-number statistics in the photon-number splitting attack / N. Lutkenhaus, M. Jahma // New Journal of Physics. – 2002. – Vol. 4. – P. 44.1-44.9.

167. Marsaglia G. DIEHARD Statistical Tests. – Available from: http://stat.fsu.edu/~geo/ diehard.html

168. Nair R. On the Security of the Y-00 Direct Encryption Protocol / Nair R., Yuen H. // ArXiv.org [Online] – 2007. – Mode of access: http://arxiv.org/abs/quant-ph/0702093v2.

169. Nauerth S. Information leakage via side channels in freespace BB84 quantum cryptography / S. Nauerth, M. Furst, T. Schmitt-Manderbach [et al.] // New Journal of Physics. – 2009. – Vol. 11, Iss. 6. – P. 065001.

170. Nayak A. Quantum and classical coin-flipping protocols based on bit-commit-ment and their point games / Nayak A., Sikora J., Tunзel L. // ArXiv.org [Online] – 2015. – Mode of access: http://arxiv.org/abs/1504.04217.

171. Niederberger A. Photon-number-splitting versus cloning attacks in practical implementations of the Bennett-Brassard 1984 protocol for quantum cryptography / A. Niederberger, V. Scarani, N. Gisin // Physical Review A. – 2005. – Vol. 71, Iss. 4. – P. 042316.

172. NIST STS [Electronic resource] : Download documentation and software. – Electronic data. – NIST, 2014. – Mode of access: http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html. – [Accessed Jul. 5, 2015]. – Description based on screen.

173. On the Interpretation of Results from the NIST Statistical Test Suite / Marek SYS, Zdenek RIHA, Vashek MATYAS, Kinga MARTON, Alin SUCIU // Romanian jurnal of information science and technology. – Vol. 18. – № 1. – 2015. – P. 18-32.

174. Peng C.Z. Experimental long-distance decoy-state quantum key distribution based on polarisation encoding / Peng C. Z., Zhang J., D. Yang, Gao W.-B. [et al.] // Physical Review Letters. – 2007. – Vol. 98, Iss 1 – P. 010505.

175. Porat D.I. Three valued digital system / Porat D.I. // Proceedings of the IEEE. – 1969. – Vol. 116, Iss. 6. – P. 947-955.

176. QPN-8505 Security Gateway: Data Sheet [Electronic data] (1 file: 143 754 bytes). – MagiQ Technologies Inc, 2007. – Mode of access: http://www.magiqtech.com/MagiQ/Products_files/8505_Data_Sheet.pdf. – [Accessed Jul. 5, 2015]. – Description based on screen.

177. Quantum Computation and Information. From Theory to Experiment / Hiroshi Imai, Masahito Hayashi [et al.]. – Springer-Verlag Berlin Heidelberg, 2006. – P. 235.

178. Quantum Key Distribution System [Electronic resource] : / Toshiba Research Europe Ltd., Cambridge Research Laboratory. – Electronic data. – Tokyo, Japan : Toshiba Corporation, 2010. – Mode of access: http://www.toshiba-europe.com/research/crl/qig/quantumkeyserver.html. – [Accessed Jul. 5, 2015]. – Description based on screen.

179. Qudit entanglement / P. Rungta [et al.] // Directions in Quantum Optics. – Heidelberg : Springer Berlin. – 2001. – P. 149-164.

180. Rosenberg D. Long-distance decoy-state quantum key distribution in optical fiber / Rosenberg D., Harrington J. W., Rice P. R. [et al.] // Physical Review Letters. – 2007. – Vol. 98, Iss. 1. – P. 010503.

181. Scarani V. Quantum Cryptography Protocols Robust against Photon Number Splitting Attacks for Weak Laser Pulse Implementations / Scarani V., Acin A., Ribordy G., Gisin N. // Physical Review Letters. – 2004. – Vol. 92, Iss. 5. – P. 057901.

182. Scarani V. The security of practical quantum key distribution / V. Scarani, H. Bechmann-Pasquinucci, N. J. Cerf [et al.] // Review of Modern Physics. – 2009. – Vol. 81, Iss. 3. – P. 1301-1350.

183. Shor P.W. Polynomial-time algorithms for prime factorization and discrete logs on a quantum computer / Shor P.W. // SIAM J. Comput. – 1997. – Vol.26. – P. 1484-1509.

184. Shor P.W. Scheme for reducing decoherence in quantum computer memory // Physical Review A. – 1995. – Vol.52, Iss.4. – P. 2493-2496.

185. Shukla C. Secure Quantum Communication with Orthogonal States / Shukla C., Banerjee A., Pathak A. // ArXiv.org [Online] – 2014. – Mode of access: http://arxiv.org/abs/1407.3412.

186. Song L. Quantum secure direct communication with χ-type entangled states / Song Lin, Qiao-Yan Wen, Fei Gao, Fu-Chen Zhu // Physical Review A. – 2008. – Vol. 78, Iss. 6. – P. 064304.

187. Statistical test suite Crypt-X // [Електронний ресурс]: http://www.isi.qut.edu./resources/cryptx/

188. Suzuki S. Classification of Quantum Repeater Attacks / Suzuki S., Rodney V.M. // NDSS Symposium 2015. – 2015. – 7 p.

189. The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness [Electronic resource]. – Electronic data. – George Marsaglia, 1995. – Mode of access: http://stat.fsu.edu/pub/diehard/. – [Accessed Jul. 5, 2015]. – Description based on screen.

190. The ternary calculating machine of Thomas Fowler [Electronic resource]. – Electronic data. – Mark Glusker, 2005. – Mode of access: http://www.mortati.com/glusker/fowler/index.htm. – [Accessed Jul. 5, 2015]. – Description based on screen.

191. Toshiba QKD system [Electronic resource]. – Electronic data. – Toshiba Research Europe Ltd, 2015. – Mode of access: http://www.toshiba.eu/eu/Cambridge-Research-Laboratory/Quantum-Information-Group/Quantum-Key-Distribution / Toshiba-QKD-system/. – [Accessed Jul. 5, 2015]. – Description based on screen.

192. Vasiliu Ye. Security amplification of the ping-pong protocol with many-qubit Greenberger-Horne-Zeilinger states / Ye. Vasiliu, S. Gnatyuk, S. Nikolayenko, T. Zhmurko// Безпека інформації. – 2012. – Т. 18. – № 2. – С. 84–88.

193. Waks E. Security of Quantum Key Distribution with Entangled Photons Against Individual Attacks / E. Waks, A. Zeevi, Y. Yamamoto // Physical Review A. – 2002. – Vol. 65, Iss. 5. – P. 052310.

194. Wang Ch. Quantum secure direct communication with high dimension quantum superdense coding / Wang Ch., Deng F.-G., Li Y.-S. et al. // Physical Review A. – 2005. – Vol. 71, Iss. 4. – P. 044305.

195. Wang J. Quantum signature scheme with single photons / Wang J., Zhang Q., Tang C. // Optoelectronics Letters. – 2005. – Vol. 2, Iss. 3. – P. 209-212.

196. Wang X.-B. Comment on Decoy State Quantum Key Distribution / X.-B. Wang // ArXiv.org [Online] – 2005. – Mode of access: http://arxiv.org/abs/quant-ph/0501143.

197. Wiesner S. Conjugate coding / Wiesner S. // Sigact News. – 1983. – Vol.15(1). – P. 78-88.

198. Williamson M. Eavesdropping on practical quantum cryptography / M. Williamson, V. Vedral // Journal of Modern Optics. – 2003. – Vol. 50, Iss. 13. – P. 1989-2011.

199. Wojcik A. Eavesdropping on the «Ping-Pong» Quantum Communication Protocol / A. Wojcik // Physical Review Letters. – 2003. – Vol. 90, Iss. 15. – P. 157901.

200. Wooters W.K. A single quantum cannot be cloned / W.K. Wooters, W.H. Zurek // Nature. – 1982. – V. 299. – P. 802.

201. Xiao-Jun W. Quantum Signature Protocol without the Trusted Third Party / W.Xiao-Jun, L.Yun // ArXiv.org [Online] – 2005. – Mode of access: http://arxiv.org/abs/quant-ph/0509129v2.

202. Yan F.-L. Quantum secret sharing protocol between multiparty and multiparty with single photons and unitary transformations / Yan F.-L., Gao T., Li Yu.-Ch. // Chinese Physics Letters. – 2008. – V. 25. – P. 1187-1190.

203. Yuen H. P. KCQ: A New Approach to Quantum Cryptography I. General Principles and Key Generation / H.P. Yuen // ArXiv.org [Online] – 2003. – Mode of access: http://arxiv.org/abs/quant-ph/0311061

204. Zhmurko T.O. Assessment of randomness for ternary sequences in quantum cryptography / T.O. Zhmurko, S.O. Gnatyuk // Proceedings of the the V world congress Aviation in the XXI-st century. Safety in Aviation and Space Technologies. – 2012. – P. 1.7.50-1.7.53.

205. Zhmurko T.O. Efficiency increasing of the quantum cryptography systems based on the ping-pong protocol / T.O. Zhmurko, V.M. Kinzeryavyy, S.O. Gnatyuk // Сучасний захист інформації. – 2012. – №3. – С. 5-11.

206. Zhmurko T.O. Modern quantum key distribution protocols / T.O. Zhmurko, S.O. Gnatyuk // Інтегровані інтелектуальні робототехнічні комплекси (ІІРТК-2013) : VI міжнар. наук.-практ. конф., 27-29 травня 2013 р. : тези доп. – К., 2013. – С. 289-291.

207. Zhmurko T.O. Practical aspects of quantum cryptography using in real information & communication systems / T.O. Zhmurko, S.O. Gnatyuk // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: I міжн. наук.-практ. конф., 25-28 лютого 2015 р.: тези доп. – К.: Видавництво Європейського університету, 2015. – С. 34-36.

208. Zhmurko T.O. Quantum cryptography innovations / T.O. Zhmurko, S.O. Gnatyuk // Політ. Сучасні проблеми науки: тези доп. XV міжнар. наук.-практ. конф. молодих учених і студентів, м. Київ, 8-9 квітня 2015 р., НАУ / редкол.: М.С. Кулик [та ін.]. – 2015. – С. 126.

209. Zhmurko T.O. Quantum game theory in classification of quantum information security methods / T.O. Zhmurko, S.O. Gnatyuk // Proceedings of the the VI world congress Aviation in the XXI-st century. Safety in Aviation and Space Technologies. – 2014. – Vol. 1 – P. 1.11.36-1.11.39.

Додаток А. Документи, що підтверджують впровадження результатів дисертації

САЙФЕР
Системи захисту інформації

ТОВ «Сайфер БІС»
Адреса: 04107, Київ, вул. Нагірна, 25
Тел./Факс: (044) 484-46-17, 484-46-12, 483-03-22
E-mail: sales@cipher.kiev.ua
http://www.elpay.com; http://www.cipher.kiev.ua

Вих. № 046/15
28.10.2015 р.

АКТ

впровадження результатів дисертаційної роботи
Жмурко Тетяни Олександрівни
«Методи підвищення ефективності протоколів квантової криптографії»
на здобуття наукового ступеню кандидата технічних наук у діяльності
ТОВ «Сайфер БІС»

Комісія у складі голови – виконавчого директора товариства з обмеженою відповідальністю «Сайфер БІС», Зацепіна О.В., членів комісії – заступника директора з виробництва, кандидата технічних наук Ковтуна В.Ю., провідного розробника Бойко С.Т., склала цей акт про те, що при розробці «Системи криптографічного захисту інформації «Шифр-Х.509» (далі СКЗІ «Шифр-Х.509»), використано результати дисертаційної роботи Жмурко Тетяни Олександрівни «Методи підвищення ефективності протоколів квантової криптографії».

У СКЗІ «Шифр-Х.509» використовується генерування псевдовипадкових послідовностей на основі трійкової незбалансованої псевдовипадкової послідовності з можливістю контролю якості вироблених послідовностей на основі оцінювання статистичних параметрів і закономірностей.

У ядрі СКЗІ «Шифр-Х.509» використано такі результати дисертаційної роботи дисертації Жмурко Т.О.:

- **Метод генерування псевдовипадкових послідовностей TriGen**, який, за рахунок виконання нової послідовності операцій дозволяє формувати трійкові незбалансовані псевдовипадкові послідовності для генерації ключів електронного цифрового підпису за ДСТУ 4145-2002, вироблення спільного секрету за ДСТУ ISO/IEC 15946-3, симетричного блочного шифру за ДСТУ ГОСТ 28147-89.
- **Метод оцінювання якості псевдовипадкових послідовностей**, який призначений для оцінювання якості псевдовипадкових послідовностей, завдяки оцінюванню статистичних параметрів і закономірностей трійкових у незбалансованих псевдовипадкових послідовностей.

Проведено тестування модуля генерації ключів СКЗІ «Шифр-Х.509» і перевірено їх якість та придатність для використання в реальних умовах.

Отже, результати, отримані Жмурко Т.О. під час написання дисертаційної роботи, дозволили підвищити швидкість та статистичну якість ключів для СКЗІ «Шифр-Х.509».

Виконавчий директор ТОВ «Сайфер БІС» О.В. ЗАЦЕПІН

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
КАЗАХСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ имени К.И. САТПАЕВА

УТВЕРЖДАЮ
Директор ИИиТТ
КазНИТУ имени К.И. Сатпаева
Ахметов Б.С.
«____» _____ 2015 г.

АКТ № 2-16

внедрения результатов диссертационного исследования

в учебный процесс

Комиссия в составе: заведующая кафедрой «Информационная безопасность» Сейлова Н.А (председатель), профессор Айтхожаева Е.Ж., ст. преп. Алимсеитова Ж.К. составили настоящий акт о том, что следующие результаты диссертационной работы Жмурко Татьяны Александровны внедрены в учебный процесс и используются на кафедре «Информационная безопасность»:

1. Лекции «Методы и средства квантового распределения криптографических ключей», «Методы квантовой прямой безопасной связи», «Коммерческие системы квантовой криптографии».
2. Лабораторные работы «Моделирование протоколов квантовой криптографии», «Помехоустойчивое кодирование в квантовом канале связи».

Апробировано в учебном процессе магистратуры «Криптологические методы и средства защиты информации», «Алгоритмы криптографической защиты информации (альтернативные дисциплины)» и бакалавриата «Математические основы защиты информации».

Актуальность и научная новизна заключается в возможности методов квантовой криптографии обеспечить теоретико-информационную стойкость, которая не зависит от временных, ресурсных и других возможностей криптоаналитика. Кроме этого, предложены новые квантовые методы распределения ключей шифрования и прямой безопасной связи, процедуры усиления их стойкости к разным типам атак, а также проведена оценка криптостойкости существующих и предложенных методов квантовой криптографии.

Принято решение **о внедрении** в учебный процесс.

Председатель комиссии _____ Сейлова Н.А.
(ФИО, подпись)

Члены комиссии _____ Айтхожаева Е.Ж.
(ФИО, подпись)

_____ Алимсеитова Ж.К.
(ФИО, подпись)

Ф КазНИТУ 601-34. Акт внедрения в учебный процесс

ЗАТВЕРДЖУЮ:

Проректор з наукової роботи
Національного авіаційного
університету
_____ В. Харченко
«21» грудня 2015 р.

АКТ

впровадження у навчальний процес результатів дисертаційної роботи Жмурко Тетяни Олександрівни «Методи підвищення ефективності протоколів квантової криптографії» на здобуття кандидата технічних наук.

Комісія у складі: голова – завідувач кафедри безпеки інформаційних технологій (БІТ) Корченко О.Г., доцент кафедри БІТ Гнатюк С.О., доцент кафедри БІТ Кінзерявий В.М. склали даний акт про те, що результати дисертаційної роботи Жмурко Тетяни Олександрівни впровадженні у навчальний процес та використовуються на кафедрі БІТ у 2015-2016 навчальному році при викладанні дисципліни «Безпека інформації в інформаційно-комунікаційних системах», що входить до навчальних планів підготовки фахівців у галузі знань «Інформаційна безпека».

№ з/п	Назва роботи, що впроваджується	Форма впровадження	Ефективність від впровадження
	1	2	3
1.	Розширена класифікація сучасних квантових технологій та протоколів на їх основі.	Лекція	Ознайомлення студентів з сучасними квантово-криптографічними протоколами, системами та алгоритмами.
2.	Підвищення ефективності (секретності, доступності, швидкості роботи) квантових комунікаційних протоколів	Лабораторна робота	Ознайомлення студентів з сучасними методами і процедурами підвищення рівня секретності та швидкості роботи квантових протоколів за рахунок збільшення їх інформаційної місткості (використання кудитів).

Голова комісії,

завідувач кафедри безпеки інформаційних технологій — О. Корченко

Члени комісії:

доцент кафедри безпеки інформаційних технологій — С. Гнатюк

доцент кафедри безпеки інформаційних технологій — В. Кінзерявий

ЗАТВЕРДЖУЮ:
Проректор з наукової роботи
Кіровоградського національного технічного університету
_____О.М. Левченко
«15» 10 2015 р.

АКТ
про впровадження результатів дисертаційної роботи
асистента кафедри безпеки інформаційних технологій
Національного авіаційного університету
Жмурко Тетяни Олександрівни

Цей акт складено у тому, що під час роботи над держбюджетною темою № 36Б115 «Розробка методів синтезу тестових моделей поведінки програмних об'єктів, підвищення оперативності передачі та захисту інформації у телекомунікаційних системах» (№ДР 0115U003103), яка виконується у Кіровоградському національному технічному університеті, реалізовано такі результати наукових досліджень Жмурко Тетяни Олександрівни:

1. На підставі часткових узагальнень теоретичних положень та практичних досягнень у галузі квантової криптографії, розроблено розширену класифікацію методів квантової криптографії, яка дозволяє більш ефективно вибирати відповідні методи для побудови сучасних квантових систем захисту інформації.

2. Метод забезпечення стійкості тритових протоколів квантової криптографії, що не потребує великих часових та ресурсних затрат і, за рахунок використання попередньої ідентифікації легітимних користувачів та тритової симетричної функції, дозволяє підвищити швидкість роботи протоколу в 11,1 разів і забезпечити його стійкість до некогерентних атак.

Запропоновані, у результаті виконання наукових досліджень Жмурко Тетяни Олександрівни, програмні реалізації (зокрема «GenSBOX2» (свідоцтво №48036 від 26.02.2013 року), «GenSBOX3» (свідоцтво №48037 від 26.02.2013 року) та «TrytTon 2012» (свідоцтво №48040 від 26.02.2013 року)), дозволяють реалізувати окремі процедури безпеки і підвищити рівень конфіденційності електронних даних в інформаційно-комунікаційних системах.

Керівник ДБ № 36Б115 «Розробка методів синтезу тестових моделей поведінки програмних об'єктів, підвищення оперативності передачі та захисту інформації у телекомунікаційних системах»

доктор технічних наук, професор О.А. Смірнов

Implementation act

of Tetiana O. Zhmurko

the dissertation work (PhD) results

for obtaining the PhD (candidate of technical sciences)

in activities of the company «Bilfinger HSG»

This act drawn up that result of Tetiana Zhmurko dissertation work (PhD) are implemented and used in the activities of the company «Bilfinger HSG». During dissertation work (PhD) by author was developed a number of methods and relevant software tools for security amplification, which allowed increasing the level of information resources confidentiality. Proposed method provides security amplification for quantum secure direct communication protocols and could increase it rate at least in 3 times in comparison with the existed methods.

Also, proposed set of evaluation methods for pseudorandom sequences and an appropriate software by which you can estimate the quality of pseudorandom sequences, which can be used in cryptographic applications (for example, as encryption keys, gamma etc.)

Proposed approaches were used in the company «Bilfinger HSG» for information security solutions implementation and allowed to improve security level.

Company director (Signature) Surname

U. A. Wagner

Bilfinger HSG
Facility Management GmbH
An der Gehespitz 50
63263 Neu-Isenburg
Deutschland

03. 09. 2015

Додаток Б. Лістинги (коди) програмних засобів

1. GenSBOX3

```
/// NewGenerator.h
#pragma once
#include <fstream>
#include <iostream>
#include <string>
#include "PolyaGalya3.h"
#include <time.h>

class NewGenerator
{
	std::string imyaPapki;
	std::string imyaF_Sbox;
	std::string imyaF_InvSbox;
	std::string imyaF_Param;
	std::ofstream outParam;
	int **SBox;
	int **InvSBox;
	int KolElVhoda;
	int KolElVuhoda;
	int StrokVsego;
	int StolbzovVsego;
	int KolVhTrut;
	int KolNyznuhSboxov;
	int IndexNyznuhSboxov;
	int *TablicaSootvetstviu;
	unsigned long DopystimoeZna4SumSboxov;
	PolyaGalya3 galyaField;

public:
	NewGenerator( std::string imyaPapki_ );
	~NewGenerator( void );
	void NautiLy4wueSBloki( int KolNyznuhSboxov_, int KolTestov_ );
	int Generazuya( int **My_, int Vy_, bool pe4at );
	void InizualizazuyaTablicuSbox( );
	void YdelenieTablicuSbox( );
	void PrintSbox( bool pe4at );
	int Poly4it4isloSBloka( int X, int Vy_, int **My );
	void YmnozutKvMatrNaStol( int **My_, int *X, int *Z );
	void DodatDvaMasuva( int *X, int *Y, int *Z );
	int* Poly4it_3kod (int X);
};

/// NewGenerator.cpp
#include "StdAfx.h"
#include "NewGenerator.h"
#include <math.h>

NewGenerator::NewGenerator( std::string imyaPapki_ = "" )
{
	imyaPapki = imyaPapki_;
	imyaF_Param = "Pod_Param_3.txt";
	std::string tempF_Param_ = imyaPapki + imyaF_Param;
	outParam.open( tempF_Param_.c_str() );
	KolVhTrut = 6;
	KolElVuhoda = 729;
	KolElVhoda = pow( (float) 3 , KolVhTrut);
	StrokVsego = KolElVhoda / pow( (float) 3 , 2);
	StolbzovVsego = KolElVhoda / StrokVsego;
	DopystimoeZna4SumSboxov = 0;
	for( int i = 0; i < KolElVuhoda; i++ )
	{
		DopystimoeZna4SumSboxov = DopystimoeZna4SumSboxov + i;
	}
	galyaField.Inizualizacuya_TablicaSootvetstviu();
	galyaField.Inizualizacuya_TabSootvVPolyahGalya();
```

```
}

NewGenerator::~NewGenerator( void )
{
}

int NewGenerator::Generazuya( int **My_, int Vy_, bool pe4at )
{
	int rez = 0;
	InizualizazuyaTablicuSbox( );
	unsigned long summa			= 0;
	bool EstFiksrovTo4ka		= false;
	for( int i = 0; i < KolElVhoda; i++ )
	{
		int ii = i / StolbzovVsego, jj = i % StolbzovVsego;
		SBox[ ii ][ jj ] = Poly4it4isloSBloka( i, Vy_, My_ );
		TablicaSootvetstviu[ SBox[ ii ][ jj ] ]++;
		int iii = SBox[ ii ][ jj ] / StolbzovVsego, jjj = SBox[ ii ][ jj ] % StolbzovVsego;
		InvSBox[ iii ][ jjj ] = i;
		summa = summa + SBox[ ii ][ jj ];
		if ( SBox[ ii ][ jj ] == i )
		{
			EstFiksrovTo4ka = true;
		}
	}

	if( summa != DopystimoeZna4SumSboxov )
	{
		return rez;
	}

	if( EstFiksrovTo4ka )
	{
		return rez;
	}

	for( int i = 0; i < KolElVhoda; i++ )
	{
		if( TablicaSootvetstviu[ i ] != 1 )
		{
			return rez;
		}
	}

	if ( ( ( summa == DopystimoeZna4SumSboxov ) && ( EstFiksrovTo4ka == false ) ) || ( pe4at == true ) )
	{
		PrintSbox( true );

		outParam << "\n\n\t Nomer " << IndexNyznuhSboxov;

		int yyyy[6] = {0};
		galyaField.Poly4itMasivPoTablicuSootvetstviy( Vy_, yyyy );
		outParam << "\n\n\tVy = \n";
		for( int iii = 0; iii < KolVhTrut; iii++ )
		{
			outParam << "\t" << yyyy[iii];
		}
		outParam << "\n\n\tMy = \n";

		std::hex( outParam );

		for( int iii = 0; iii < KolVhTrut; iii++ )
		{
			for( int jjj = 0; jjj < KolVhTrut; jjj++ )
			{
				outParam << "\t" << My_[iii][jjj];
			}
			outParam << "\n";
		}
```

```
                std::dec( outParam );

                outParam << "\n\n";
                IndexNyznuhSboxov++;
        }

        YdelenieTablicuSbox( );

        return rez;
}

void NewGenerator::InizualizazuyaTablicuSbox( )
{
        SBox = new int*[ StrokVsego ];
        for ( int i = 0; i < StrokVsego; i++ ) SBox[ i ] = new int[ StolbzovVsego ];

        for( int i = 0 ; i < StrokVsego ; i++ )
        {
                for( int j = 0 ; j < StolbzovVsego ; j++ )
                {
                        SBox[ i ][ j ] = 0;
                }
        }
        InvSBox = new int*[ StrokVsego ];
        for ( int i = 0; i < StrokVsego; i++ ) InvSBox[ i ] = new int[ StolbzovVsego ];

        for( int i = 0 ; i < StrokVsego ; i++ )
        {
                for( int j = 0 ; j < StolbzovVsego ; j++ )
                {
                        InvSBox[ i ][ j ] = 0;
                }
        }
}

void NewGenerator::YdelenieTablicuSbox( )
{
        for ( int i = 0; i < StolbzovVsego; i++ )
        {
                delete[]SBox[ i ];
        }
        delete[]SBox;
        for ( int i = 0; i < StolbzovVsego; i++ )
        {
                delete[]InvSBox[ i ];
        }
        delete[]InvSBox;
}

void NewGenerator::PrintSbox( bool pe4at )
{
        if ( pe4at )
        {
                imyaF_Sbox                                                      = "Sbox_3tryt_6__";
                imyaF_InvSbox                                           = "InvSbox_3tryt_6__";
                char indexSboxa[ 10 ];
                itoa( IndexNyznuhSboxov, indexSboxa, 10 );
                imyaF_Sbox                      = imyaF_Sbox + indexSboxa;
                imyaF_InvSbox     = imyaF_InvSbox + indexSboxa;
                imyaF_Sbox                      = imyaF_Sbox + ".txt";
                imyaF_InvSbox     = imyaF_InvSbox + ".txt";
                std::string imyaF_Sbox_ = imyaPapki + imyaF_Sbox;
                std::ofstream out1( imyaF_Sbox_.c_str() );
                for( int i = 0 ; i < StrokVsego ; i++ )
                {
                        for( int j = 0 ; j < StolbzovVsego ; j++ )
                        {
                                if( SBox[ i ][ j ] < 10 ) out1 << " ";
                                if( SBox[ i ][ j ] < 100 ) out1 << " ";
```

```
                                        out1 <<"\t"<< SBox[ i ][ j ] << ", ";
                                }
                                out1 << "\n";
                        }
                        std::string imyaF_InvSbox_ = imyaPapki + imyaF_InvSbox;
                        std::ofstream out2( imyaF_InvSbox_.c_str() );
                        for( int i = 0 ; i < StrokVsego ; i++ )
                        {
                                for( int j = 0 ; j < StolbzovVsego ; j++ )
                                {
                                        if( InvSBox[ i ][ j ] < 10 ) out2 << " ";
                                        if( InvSBox[ i ][ j ] < 100 ) out2 << " ";

                                        out2 <<"\t"<< InvSBox[ i ][ j ] << ", ";
                                }
                                out2 << "\n";
                        }
                }
}

int NewGenerator::Poly4it4isloSBloka( int X, int Vy_, int **My_ )
{
        int x[6] = {0};
        galyaField.Poly4itMasivPoTablicuSootvetstviy( X, x );
        galyaField.NautiobratnuuElement( x, x );
        YmnozutKvMatrNaStol( My_, x, x );
        int y[6] = {0};
        galyaField.Poly4itMasivPoTablicuSootvetstviy( Vy_, y );
        DodatDvaMasuva( x, y, x );
        int result = galyaField.Poly4it10Chislo( x, KolVhTrut );
        return result;
}

void NewGenerator::YmnozutKvMatrNaStol( int **My_, int *X, int *Z )
{
        int rez[6] = {0};
        for ( int strM = 0; strM < KolVhTrut; strM++ )
        {
                int tmp = 0;
                for ( int strX = 0; strX < KolVhTrut; strX++ )
                {
                        tmp = ( tmp + ( My_[ strM ][ strX ] * X[ strX ] ) )%3;
                }
                rez[ strM ] = tmp;
        }
        for ( int i = 0; i < KolVhTrut; i++ )
        {
                Z[ i ] = rez[ i ];
        }
}
void NewGenerator::DodatDvaMasuva( int *X, int *Y, int *Z )
{
        for ( int i = 0; i < KolVhTrut; i++ )
        {
                Z[ i ] = ( X[ i ] + Y[ i ] )%3;
        }
}
void NewGenerator::NautiLy4wueSBloki( int KolNyznuhSboxov_, int KolTestov_ )
{
        KolNyznuhSboxov         = KolNyznuhSboxov_;
        IndexNyznuhSboxov       = 0;
        srand ( time(NULL) );
        unsigned int KolProudeno = 0;
        for ( int i = 0; i < KolTestov_; i++ )
        {
                if ( IndexNyznuhSboxov >= KolNyznuhSboxov )
                {
                        break;
                }
                int **My_;
```

```
                My_ = new int*[ KolVhTrut ];
                for( int i = 0; i < KolVhTrut; i++ )
                {
                        My_[ i ] = new int[ KolVhTrut ];
                }
                int Vrem_perem = ( rand() )%KolElVhoda ;
                for( int j = 0; j < KolVhTrut; j++ )
                {
                        int x                   = Vrem_perem % 3;
                        Vrem_perem      = Vrem_perem / 3;
                        My_[ 0 ][ ( j + 0 )%6 ] = x;
                        My_[ 1 ][ ( j + 1 )%6 ] = x;
                        My_[ 2 ][ ( j + 2 )%6 ] = x;
                        My_[ 3 ][ ( j + 3 )%6 ] = x;
                        My_[ 4 ][ ( j + 4 )%6 ] = x;
                        My_[ 5 ][ ( j + 5 )%6 ] = x;
                }
                bool zanovo = false;
                for ( int k = 0; k < 20; k++ )
                {
                        if ( zanovo )
                        {
                                break;
                        }
                        int Vy_ = ( rand() )%KolElVhoda ;
                        TablicaSootvetstviu                     = new int[ KolElVuhoda ];
                        for( int i = 0; i < KolElVuhoda; i++ )
                        {
                                TablicaSootvetstviu[ i ]        = 0;
                        }
                        if ( Generazuya( My_, Vy_,  false ) == 1 )
                        {
                                zanovo = true;
                                KolProudeno = KolProudeno + 1;
                                IndexNyznuhSboxov++;
                        }
                        delete[]TablicaSootvetstviu;
                }

                for ( int i = 0; i < KolVhTrut; i++ ) delete[]My_[ i ];
                delete[]My_;
        }
}


/// PolyaGalya3.h
#pragma once
#include <math.h>
class PolyaGalya3
{
        public:
        int Poly4it10Chislo( int *X, int Kol );
        int YmnozutChuslo_Na_X( int *y, int *Pol );
        void Inizualizacuya_TablicaSootvetstviu();
        void Inizualizacuya_TabSootvVPolyahGalya();
        void Ymnozut( int *X, int *Y, int *Z );
        void Podelit( int *X, int *Y, int *Z );
        void NautiobratnuuElement( int *X, int *Z );
        void Poly4itMasivPoTablicuSootvetstviy( int x, int *Z );
        int TabSootvVPolyahGalya[729];
        int TabSootvVPolyahGalya_obr[729];
        int TablicaSootvetstviu[729][6];
};

/// PolyaGalya3.cpp
#include "StdAfx.h"
#include "PolyaGalya3.h"

int PolyaGalya3::Poly4it10Chislo( int *X, int Kol )
{
```

```
	int rez = 0;
	for( int i = 0; i < Kol; i++ )
	{
		rez = rez + X[i] * pow( (float) 3, i);
	}
	return rez;
}
int PolyaGalya3::YmnozutChuslo_Na_X( int *y, int *Pol )
{
	int rez[6] = {0};
	if( y[5] == 0 )
	{
		for( int i = 0 ; i < 5 ; i++ )
		{
			rez[ i + 1 ] = y[i];
		}
	}
	else
	{
		int koef = ( y[5] == 1 )?( 1 ):( 2 );
		for( int i = 0 ; i < 5 ; i++ )
		{
			rez[ i + 1 ] = y[i];
		}
		for( int i = 0 ; i < 6 ; i++ )
		{
			rez[ i ] = ( rez[ i ] - koef * Pol[ i ] )%3;

			if( rez[ i ] < 0 )
			{
				rez[ i ] = rez[ i ] + 3;
			}
		}
	}
	for( int i = 0 ; i < 6 ; i++ )
	{
		y[i] = rez[i];
	}
	return Poly4it10Chislo( rez, 6 );
}
void PolyaGalya3::Inizualizacuya_TablicaSootvetstviu()
{
	for( int i = 0; i < 729; i++ )
	{
		int temp = i;
		TablicaSootvetstviu[i][0] = temp%3;
		temp = temp/3;
		TablicaSootvetstviu[i][1] = temp%3;
		temp = temp/3;
		TablicaSootvetstviu[i][2] = temp%3;
		temp = temp/3;
		TablicaSootvetstviu[i][3] = temp%3;
		temp = temp/3;
		TablicaSootvetstviu[i][4] = temp%3;
		TablicaSootvetstviu[i][5] = temp/3;
	}
}

void PolyaGalya3::Inizualizacuya_TabSootvVPolyahGalya()
{
	int Pol[7] = { 2, 1, 0, 0, 0, 0, 1 };
	for( int i = 0; i < 729; i++ )
	{
		TabSootvVPolyahGalya[i]		= -1;
		TabSootvVPolyahGalya_obr[i] = -1;
	}
	int x[ 6 ]		= { 0, 1, 0, 0, 0, 0 };
	int st			= 1;
	TabSootvVPolyahGalya[ Poly4it10Chislo( x , 6 ) ]	= st;
	TabSootvVPolyahGalya_obr[ st ]		= Poly4it10Chislo( x , 6 );
```

```
		int		tekEl[6]  = { 0, 1, 0, 0, 0, 0 };
		while ( 1 )
		{
			int Novoe4islo = YmnozutChuslo_Na_X( tekEl, Pol );
			st++;
			st = ( st )%729;
			if( st == 0 )
			{
				break;
			}
			TabSootvVPolyahGalya[ Novoe4islo ] = st;
			TabSootvVPolyahGalya_obr[ st ] = Novoe4islo;
		}
}

void PolyaGalya3::Ymnozut( int *X, int *Y, int *Z )
{
		int x			= Poly4it10Chislo( X , 6 );
		int y			= Poly4it10Chislo( Y , 6 );
		if ( ( x == 0 ) || ( y == 0 ) )
		{
			for( int i = 0; i < 6; i++ )
			{
				Z[i] = TablicaSootvetstviu[ 0 ][ i ];
			}
			return;
		}
		int st_x  = TabSootvVPolyahGalya[ x ];
		int st_y  = TabSootvVPolyahGalya[ y ];
		int st_z  =  st_x + st_y;
		if ( st_z >= 729)
		{
			st_z = ( st_z%729 ) + 1 ;
		}
		int z			= TabSootvVPolyahGalya_obr[ st_z ];
		for( int i = 0; i < 6; i++ )
		{
			Z[i] = TablicaSootvetstviu[ z ][ i ];
		}
}

/// Podelit
void PolyaGalya3::Podelit( int *X, int *Y, int *Z )
{
		int x			= Poly4it10Chislo( X , 6 );
		int y			= Poly4it10Chislo( Y , 6 );
		if ( ( x == 0 ) || ( y == 0 ) )
		{
			for( int i = 0; i < 6; i++ )
			{
				Z[i] = TablicaSootvetstviu[ 0 ][ i ];
			}
			return;
		}
		int st_x  = TabSootvVPolyahGalya[ x ];
		int st_y  = TabSootvVPolyahGalya[ y ];
		int st_z  =  st_x - st_y;
		if ( st_z <= 0 )
		{
			st_z = st_z + 728 ;
		}
		int z			= TabSootvVPolyahGalya_obr[ st_z ];
		for( int i = 0; i < 6; i++ )
		{
			Z[i] = TablicaSootvetstviu[ z ][ i ];
		}
}
void PolyaGalya3::NautiobratnuuElement( int *X, int *Z )
{
		int x			= Poly4it10Chislo( X , 6 );
```

```
	if ( x == 0 )
	{
		for( int i = 0; i < 6; i++ )
		{
			Z[i] = TablicaSootvetstviu[ 0 ][ i ];
		}
		return;
	}
	if ( x == 1 )
	{
		for( int i = 0; i < 6; i++ )
		{
			Z[i] = TablicaSootvetstviu[ 1 ][ i ];
		}
		return;
	}
	int st_x  = TabSootvVPolyahGalya[ x ];
	int st_z = 728 - st_x;
	int z		= TabSootvVPolyahGalya_obr[ st_z ];
	for( int i = 0; i < 6; i++ )
	{
		Z[i] = TablicaSootvetstviu[ z ][ i ];
	}
}
void PolyaGalya3::Poly4itMasivPoTablicuSootvetstviy( int x, int *Z )
{
	for( int i = 0; i < 6; i++ )
	{
		Z[i] = TablicaSootvetstviu[ x ][ i ];
	}
}
```

2. TriGen

```
/// TRIGEN.h
#pragma once
#include "TRIGEN_Sbox3.h"
#include <iostream>
#include <fstream>
#include <conio.h>
#include <string>
unsigned long long SLozenieMod3_24( unsigned long long X, unsigned long long Y );
unsigned long long SLozenieMod3		( unsigned long long X, unsigned long long Y );
unsigned long long Left24		( unsigned long long X, unsigned long long Y );
unsigned long long Right24		( unsigned long long X, unsigned long long Y );
unsigned long long Sbox			( unsigned long long X );
unsigned long long Mix			( unsigned long long X );
void pods4et				( int X );
void Print4uslo				( unsigned long long X1, unsigned long long X2, unsigned long long
X3, unsigned long long X4 );
void Print4uslo2			( unsigned long long ind, unsigned long long X1, unsigned long long X2,
unsigned long long X3, unsigned long long X4 );

class TRIGEN
{
	public:
		TRIGEN(void);
		~TRIGEN(void);
		void GenerateTrytSequence( unsigned long long *Key, unsigned long long Count_96_Blokov, char* dir, char*
FileName );
		void MixColumns( unsigned long long &t );
		void RoundTransformation( unsigned long long &x1, unsigned long long &x2, unsigned long long &x3, unsigned
long long &x4, unsigned long long &A , unsigned long long &B , unsigned long long &C , unsigned long long &D , unsigned long long
&E, unsigned long long &F, unsigned long long &y1, unsigned long long &y2, unsigned long long &y3, unsigned long long &y4 );
};

/// TRIGEN.cpp
#include "StdAfx.h"
#include "TRIGEN.h"
#define TRIGEN_KolZuklov 5
#define Mod3_24 282429536481
```

```
#define Izmenit_A                                                       \
{       A                       = SLozenieMod3_24( A, x1 );             \
        A                       = Sbox( A );                                    \
        A                       = SLozenieMod3( A, D );                 \
        A                       = Left24( A, x4 );                      \
}
#define Izmenit_D                                                       \
{       D                       = SLozenieMod3_24( D, x3 );             \
        D                       = Sbox( D );                                    \
        D                       = SLozenieMod3( D, A );                 \
        D                       = Left24( D, x2 );                      \
}

#define Izmenit_B                                                       \
{       B                       = SLozenieMod3_24( B, x2 );             \
        B                       = Sbox( B );                                    \
        B                       = SLozenieMod3_24( B, E );      \
        B                       = Right24( B, x3 );                     \
}
#define Izmenit_E                                                       \
{       E                       = SLozenieMod3_24( E, x4 );             \
        E                       = Sbox( E );                                    \
        E                       = SLozenieMod3_24( E, B );      \
        E                       = Right24( E, x1 );                     \
}

#define Izmenit_C                                                       \
{       C                       = SLozenieMod3_24( C, F );      \
        C                       = SLozenieMod3( C, y3 );        \
        C                       = Mix( C );
        \
        C                       = Left24( C, A );                       \
}
#define Izmenit_F                                                       \
{       F                       = SLozenieMod3_24( F, C );      \
        F                       = SLozenieMod3( F, y1 );        \
        F                       = Mix( F );
        \
        F                       = Left24( F, D );                       \
}

#define Izmenit_x1                                                      \
{       x1                      = SLozenieMod3( x1, A );        \
        x1                      = Sbox( x1 );                                   \
        x1                      = SLozenieMod3_24( x1, E );             \
        x1                      = SLozenieMod3( x1, y1 );       \
        x1                      = Sbox( x1 );                                   \
}
#define Izmenit_x3                                                      \
{       x3                      = SLozenieMod3( x3, D );        \
        x3                      = Sbox( x3 );                                   \
        x3                      = SLozenieMod3_24( x3, B );             \
        x3                      = SLozenieMod3( x3, y3 );       \
        x3                      = Sbox( x3 );                                   \
}
#define Izmenit_x2                                                      \
{       x2                      = SLozenieMod3_24( x2, B );             \
        x2                      = SLozenieMod3_24( x2, F );             \
        x2                      = Mix( x2 );                                    \
        x2                      = SLozenieMod3( x2, y2 );       \
        x2                      = Sbox( x2 );                                   \
}
#define Izmenit_x4                                                      \
{       x4                      = SLozenieMod3_24( x4, E );             \
        x4                      = SLozenieMod3_24( x4, C );             \
        x4                      = Mix( x4 );                                    \
        x4                      = SLozenieMod3( x4, y4 );       \
        x4                      = Sbox( x4 );                                   \
}
```

```
#define Izmenit_y1                                                                      \
{       y1                      = SLozenieMod3_24( y1, x1 );                    \
        y1                      = Left24( y1, B );                              \
        y1                      = SLozenieMod3( y1, x3 );                       \
        y1                      = Sbox( y1 );                                           \
}
#define Izmenit_y3                                                                      \
{       y3                      = SLozenieMod3_24( y3, x3 );                    \
        y3                      = Left24( y3, E );                              \
        y3                      = SLozenieMod3( y3, x1 );                       \
        y3                      = Sbox( y3 );                                           \
}

#define Izmenit_y2                                                                      \
{       y2                      = SLozenieMod3_24( y2, x2 );                    \
        y2                      = Right24( y2, C );                             \
        y2                      = SLozenieMod3( y2, x4 );                       \
        y2                      = Sbox( y2 );                                           \
        y2                      = Mix( y2 );                                            \
}
#define Izmenit_y4                                                                      \
{       y4                      = SLozenieMod3_24( y4, x4 );                    \
        y4                      = Right24( y4, F );                             \
        y4                      = SLozenieMod3( y4, x2 );                       \
        y4                      = Sbox( y4 );                                           \
        y4                      = Mix( y4 );                                            \
}
unsigned int IntKolSumv[ 3 ] = { 0 };
unsigned long long Stepen_3[24];
std::ofstream out_;
FILE *file_write;
TRIGEN::TRIGEN(void)
{
        unsigned long long d = 1;
        Stepen_3[ 0 ] = d;
        for( int i = 1 ; i <= 23 ; i++ )
        {
                d = d * 3;
                Stepen_3[ i ] = d;
        }
}

TRIGEN::~TRIGEN(void)
{
        fclose( file_write );
}

void TRIGEN::GenerateTrytSequence( unsigned long long *Key, unsigned long long Count_96_Blokov, char* dir, char* FileName )
{
        std::string file_name1(dir);
        file_name1 = file_name1 + "\\" + FileName;
        out_.open( file_name1.c_str() );

        std::string file_name2(dir);
        file_name2 = file_name2 + "\\Nist_" + FileName;
        file_write= fopen( file_name2.c_str(), "wb" );

        std::cout << "\n\GenerateTrytSequence - " << FileName << " ...\n\n";

        unsigned long long A                    =       136123896560, \
                                                B               =        21165431162, \
                                                C               =       203437243109, \
                                                D               =       168877652118, \
                                                E               =       222496321257, \
                                                F               =        34457354235;
        unsigned long long y[ 4 ]       = { 176986532478, \
                                                                        229963452239, \
                                                                        109342578983, \
                                                                        239348532765 };
```

```
		unsigned long long x1 = Key[ 0 ] % Mod3_24, \
		                     x2 = Key[ 1 ] % Mod3_24, \
		                     x3 = Key[ 2 ] % Mod3_24, \
		                     x4 = Key[ 3 ] % Mod3_24;

		for( unsigned long i = 0; i < Count_96_Blokov; i++)
		{
			for( unsigned long j = 0; j < TRIGEN_KolZuklov; j++)
			{
				RoundTransformation( x1, x2, x3, x4, A, B, C, D, E, F, y[0], y[1], y[2], y[3] );
			}

			if ( ( i % 1000 ) == 0 )
			{
				std::cout << "\t" << i;
			}

			Print4uslo2( i, y[0], y[1], y[2], y[3] );
		}

		out_.close();
		fclose( file_write );
}

void TRIGEN::RoundTransformation  ( unsigned long long &x1, unsigned long long &x2, unsigned long long &x3, unsigned long long
&x4,
                                    unsigned long long &A , unsigned long long &B ,
unsigned long long &C , unsigned long long &D , unsigned long long &E, unsigned long long &F,
                                    unsigned long long &y1, unsigned long long &y2,
unsigned long long &y3, unsigned long long &y4 )
{
		Izmenit_A
		Izmenit_B
		Izmenit_C
		Izmenit_x1
		Izmenit_x2
		Izmenit_y1
		Izmenit_y2

		Izmenit_D
		Izmenit_E
		Izmenit_F
		Izmenit_x3
		Izmenit_x4
		Izmenit_y3
		Izmenit_y4
}

unsigned long long SLozenieMod3_24 ( unsigned long long X, unsigned long long Y )
{
		unsigned long long Z;
		Z = ( X + Y ) % Mod3_24;
		return Z;
}
unsigned long long SLozenieMod3                ( unsigned long long X, unsigned long long Y )
{
		int x, y, z;
		unsigned long long k = 1;
		unsigned long long Z = 0;
		for( int i = 0; i < 24; i++ )
		{
			x = X % 3;
			y = Y % 3;
			z = ( x + y ) % 3;

			X = X / 3;
			Y = Y / 3;
			Z = Z + z * k;
			k = k * 3;
		}
```

```
		return Z;
}
unsigned long long Left24			( unsigned long long X, unsigned long long Y )
{
		int y, z;
		y = Y % 24;
		unsigned long long k = 1;
		for( int j = 0; j < y; j++ )
		{
			k = k * 3;
		}
		unsigned long long Z = 0;
		for( int i = 0; i < 24; i++ )
		{
			z = X % 3;
			X = X / 3;

			Z = Z + z * k;
			k = k * 3;
			k = k % Mod3_24;
			if ( k == 0 ) k = 1;
		}
		return Z;
}
unsigned long long Right24			( unsigned long long X, unsigned long long Y )
{
		int y, z;
		y = Y % 24;
		y = 24 - y;

		unsigned long long k = 1;
		for( int j = 0; j < y; j++ )
		{
			k = k * 3;
		}
		unsigned long long Z = 0;
		for( int i = 0; i < 24; i++ )
		{
			z = X % 3;
			X = X / 3;

			Z = Z + z * k;
			k = k * 3;
			k = k % Mod3_24;
			if ( k == 0 ) k = 1;
		}
		return Z;
}
unsigned long long Sbox				( unsigned long long X )
{
		int z;
		unsigned long long k = 1;
		unsigned long long Z = 0;
		for( int i = 0; i < 4; i++ )
		{
			z = X % 729;
			X = X / 729;
			z = SBLOK[ z ];

			Z = Z + z * k;
			k = k * 729;
		}
		return Z;
}
unsigned long long Mix				( unsigned long long X )
{
		int x, z;
		unsigned long long k = 1;
		unsigned long long Z = 0, X_ = X;
		for( int i = 0; i < 24; i++ )
```

```
		{
			z		= 0;
			X_		= X;
			for( int j = 0; j < 24; j++ )
			{
				x		= X_ % 3;
				X_		= X_ / 3;
				z		= ( z + ( MixTable[ i ][ j ] * x ) ) % 3;
			}

			Z = Z + z * k;
			k = k * 3;
		}
		return Z;
}
void Print4uslo2 ( unsigned long long ind, unsigned long long X1, unsigned long long X2,
unsigned long long X3, unsigned long long X4 )
{
		unsigned long long temp[4] = {0};
		//		IntKol_1
		int x;
		for( int i = 0; i < 24; i++ )
		{
			x		= X1 % 3;
			X1		= X1 / 3;
			out_ << x;
			temp[0] = temp[0] + x * Stepen_3[23 - i];
			IntKolSumv[ x ]++;
		}

		for( int i = 0; i < 24; i++ )
		{
			x		= X2 % 3;
			X2		= X2 / 3;
			out_ << x;
			temp[1] = temp[1] + x * Stepen_3[23 - i];
			IntKolSumv[ x ]++;
		}

		for( int i = 0; i < 24; i++ )
		{
			x		= X3 % 3;
			X3		= X3 / 3;
			out_ << x;
			temp[2] = temp[2] + x * Stepen_3[23 - i];
			IntKolSumv[ x ]++;
		}

		for( int i = 0; i < 24; i++ )
		{
			x		= X4 % 3;
			X4		= X4 / 3;
			out_ << x;
			temp[3] = temp[3] + x * Stepen_3[23 - i];
			IntKolSumv[ x ]++;
		}

		temp[0] = temp[0]&0xffffffffff;
		temp[1] = temp[1]&0xffffffffff;
		temp[2] = temp[2]&0xffffffffff;
		temp[3] = temp[3]&0xffffffffff;
		fwrite( &temp[0], 5, 1, file_write );
		fwrite( &temp[1], 5, 1, file_write );
		fwrite( &temp[2], 5, 1, file_write );
		fwrite( &temp[3], 5, 1, file_write );
}

/// TRIGEN_Sbox3.h
#pragma once
```

```
const int SBLOK[729] =
{
                537,    301,    11,     717,    355,    493,    573,    557,    200 ,
                280,    61,     399,    137,    79,     350,    637,    510,    539 ,
                32,     180,    260,    656,    538,    564,    418,    241,    269 ,
                704,    405,    369,    45,     395,    143,    37,     26,     18  ,
                170,    41,     609,    195,    376,    50,     483,    697,    234 ,
                156,    125,    189,    83,     337,    687,    139,    131,    168 ,
                580,    210,    147,    275,    318,    322,    291,    421,    169 ,
                423,    363,    439,    416,    393,    451,    472,    594,    218 ,
                394,    681,    271,    105,    345,    614,    366,    289,    215 ,

                206,    119,    433,    458,    87,     84,     198,    707,    543 ,
                0,      643,    514,    692,    225,    256,    536,    524,    691 ,
                623,    78,     381,    285,    705,    498,    392,    647,    203 ,
                261,    390,    12,     655,    177,    202,    552,    683,    290 ,
                567,  1,511,    452,    551,    330,    505,    334,    721 ,
                274,    108,    90,     188,    603,    86,     583,    607,    658 ,
                199,    346,    276,    181,    618,    60,     66,     106,    134 ,
                615,    494,    431,    277,    630,    695,    441,    107,    30  ,
                672,    633,    149,    595,    255,    579,    230,    575,    649 ,

                358,    122,    436,    354,    495,    577,    133,    468,    474 ,
                661,    171,    267,    163,    352,    673,    33,     540,    576 ,
                309,    527,    668,    532,    602,    517,    601,    56,     327 ,
                356,    42,     236,    243,    457,    485,    401,    258,    699 ,
                645,    406,    648,    325,    635,    718,    689,    702,    54  ,
                693,    160,    556,    111,    279,    484,    44,     616,    651 ,
                75,     297,    162,    486,    49,     610,    638,    353,    384 ,
                38,     462,    444,    710,    626,    677,    403,    469,    678 ,
                714,    566,    311,    563,    590,    221,    130,    231,    490 ,

                28,     605,    726,    382,    471,    35,     521,    338,    315 ,
                211,    120,    205,    113,    40,     465,    722,    461,    686 ,
                141,    671,    703,    344,    379,    712,    103,    706,    372 ,
                467,    348,    329,    489,    25,     500,    719,    248,    164 ,
                190,    534,    36,     508,    561,    242,    430,    662,    519 ,
                667,    159,    51,     454,    71,     182,    314,    123,    665 ,
                89,     283,    148,    310,    223,    515,    112,    694,    676 ,
                597,    653,    708,    67,     636,    336,    599,    278,    201 ,
                302,    235,    701,    596,    584,    224,    646,    476,    174 ,

                473,  3,48,     214,    70,     253,    680,    417,    184 ,
                520,    307,    424,    62,     569,    76,     167,    491,    373 ,
                178,    571,    228,    124,    613,    58,     448,    723,    669 ,
                422,    232,    586,    127,    110,    408,    562,    592,    664 ,
                331,    343,    227,    146,    191,    470,    194,    20,     22  ,
                196,    666,    237,    157,    415,    488,    459,    544,    362 ,
                47,     437,    588,    192,    73,     114,    326,    320,    453 ,
                411,    166,    446,    213,    660,    136,    6,      570,    129 ,
                496,    555,    724,    77,     463,    542,  4,487,    482 ,

                533,    625,    513,    370,    548,    547,    522,    367,    63  ,
                187,    333,    208,    627,    558,    478,    385,    456,  8  ,
                16,     287,    321,    652,    435,    396,    13,     696,    335 ,
                426,    481,    298,    270,    727,    53,     249,    68,     175 ,
                410,    509,    414,    634,    282,    98,     412,    501,    99  ,
                222,    641,    700,    402,    264,    155,    303,    23,     460 ,
                74,     397,    252,    52,     606,    357,    464,    121,    31  ,
                512,    600,    304,    518,    80,     611,    716,    449,    154 ,
                257,    46,     102,    387,    413,    624,    172,    247,  9  ,

                284,    591,    679,    529,    21,     217,    173,    286,    81  ,
                420,    581,    640,    479,    207,    578,    238,    572,    176 ,
                371,    359,    432,    589,    598,    94,     442,    96,     272 ,
                475,    407,    29,     443,    608,    617,  2,526,    341 ,
                10,     619,    347,    674,    378,    88,     688,    340,    709 ,
                684,    582,    631,    685,    351,    43,     245,    216,    654 ,
                97,     226,    240,    574,    391,    64,     549,    541,    323 ,
                644,    294,    429,  7,628,    438,    128,    400,    250 ,
```

```
                365,    273,    531,    161,    528,    622,    560,    349,    504 ,

                535,    525,    659,    516,    246,    197,    212,    503,    502 ,
                305,    24,     34,     299,    220,    612,    632,    183,    117 ,
                404,    374,    219,    179,    313,    132,    663,    104,    507 ,
                265,    57,     185,    91,     281,    434,    296,    204,    675 ,
                55,     477,    293,    383,    300,    65,     165,    657,    151 ,
                565,    17,     690,    568,    428,    115,    523,    682,    268 ,
                153,    14,     101,    69,     380,    244,    39,     295,    593 ,
                339,    620,    670,    15,     95,     316,    186,    427,    72  ,
                145,    138,    559,    152,    480,    546,    650,    466,    27  ,

                82,     288,    306,    604,    398,    135,    377,    59,     251 ,
                386,    324,    713,    116,    639,    585,    440,    254,    698 ,
                530,    158,  5, 388,    209,    550,    259,    263,    715 ,
                292,    720,    118,    229,    126,    19,     360,    447,    266 ,
                545,    587,    492,    308,    100,    554,    262,    499,    92  ,
                150,    109,    389,    312,    450,    711,    375,    419,    621 ,
                142,    725,    332,    506,    629,    728,    455,    144,    445 ,
                368,    342,    642,    93,     193,    497,    425,    553,    140 ,
                233,    328,    239,    361,    317,    319,    409,    85,     364
};

const int MixTable[ 24 ][ 24 ] ={
{ 1, 0, 2, 2, 1, 0, 2, 0, 1, 1, 1, 2, 0, 1, 2, 1, 0, 2, 0, 0, 1, 2, 0, 2 },
{ 2, 1, 0, 2, 2, 1, 0, 2, 0, 1, 1, 1, 2, 0, 1, 2, 1, 0, 2, 0, 0, 1, 2, 0 },
{ 0, 2, 1, 0, 2, 2, 1, 0, 2, 0, 1, 1, 1, 2, 0, 1, 2, 1, 0, 2, 0, 0, 1, 2 },
{ 2, 0, 2, 1, 0, 2, 2, 1, 0, 2, 0, 1, 1, 1, 2, 0, 1, 2, 1, 0, 2, 0, 0, 1 },
{ 1, 2, 0, 2, 1, 0, 2, 2, 1, 0, 2, 0, 1, 1, 1, 2, 0, 1, 2, 1, 0, 2, 0, 0 },
{ 0, 1, 2, 0, 2, 1, 0, 2, 2, 1, 0, 2, 0, 1, 1, 1, 2, 0, 1, 2, 1, 0, 2, 0 },
{ 0, 0, 1, 2, 0, 2, 1, 0, 2, 2, 1, 0, 2, 0, 1, 1, 1, 2, 0, 1, 2, 1, 0, 2 },
{ 2, 0, 0, 1, 2, 0, 2, 1, 0, 2, 2, 1, 0, 2, 0, 1, 1, 1, 2, 0, 1, 2, 1, 0 },
{ 0, 2, 0, 0, 1, 2, 0, 2, 1, 0, 2, 2, 1, 0, 2, 0, 1, 1, 1, 2, 0, 1, 2, 1 },
{ 1, 0, 2, 0, 0, 1, 2, 0, 2, 1, 0, 2, 2, 1, 0, 2, 0, 1, 1, 1, 2, 0, 1, 2 },
{ 2, 1, 0, 2, 0, 0, 1, 2, 0, 2, 1, 0, 2, 2, 1, 0, 2, 0, 1, 1, 1, 2, 0, 1 },
{ 1, 2, 1, 0, 2, 0, 0, 1, 2, 0, 2, 1, 0, 2, 2, 1, 0, 2, 0, 1, 1, 1, 2, 0 },
{ 0, 1, 2, 1, 0, 2, 0, 0, 1, 2, 0, 2, 1, 0, 2, 2, 1, 0, 2, 0, 1, 1, 1, 2 },
{ 2, 0, 1, 2, 1, 0, 2, 0, 0, 1, 2, 0, 2, 1, 0, 2, 2, 1, 0, 2, 0, 1, 1, 1 },
{ 1, 2, 0, 1, 2, 1, 0, 2, 0, 0, 1, 2, 0, 2, 1, 0, 2, 2, 1, 0, 2, 0, 1, 1 },
{ 1, 1, 2, 0, 1, 2, 1, 0, 2, 0, 0, 1, 2, 0, 2, 1, 0, 2, 2, 1, 0, 2, 0, 1 },
{ 1, 1, 1, 2, 0, 1, 2, 1, 0, 2, 0, 0, 1, 2, 0, 2, 1, 0, 2, 2, 1, 0, 2, 0 },
{ 0, 1, 1, 1, 2, 0, 1, 2, 1, 0, 2, 0, 0, 1, 2, 0, 2, 1, 0, 2, 2, 1, 0, 2 },
{ 2, 0, 1, 1, 1, 2, 0, 1, 2, 1, 0, 2, 0, 0, 1, 2, 0, 2, 1, 0, 2, 2, 1, 0 },
{ 0, 2, 0, 1, 1, 1, 2, 0, 1, 2, 1, 0, 2, 0, 0, 1, 2, 0, 2, 1, 0, 2, 2, 1 },
{ 1, 0, 2, 0, 1, 1, 1, 2, 0, 1, 2, 1, 0, 2, 0, 0, 1, 2, 0, 2, 1, 0, 2, 2 },
{ 2, 1, 0, 2, 0, 1, 1, 1, 2, 0, 1, 2, 1, 0, 2, 0, 0, 1, 2, 0, 2, 1, 0, 2 },
{ 2, 2, 1, 0, 2, 0, 1, 1, 1, 2, 0, 1, 2, 1, 0, 2, 0, 0, 1, 2, 0, 2, 1, 0 },
{ 0, 2, 2, 1, 0, 2, 0, 1, 1, 1, 2, 0, 1, 2, 1, 0, 2, 0, 0, 1, 2, 0, 2, 1 },
};
```

3. TritSTS

```
/// AnalysisTritSequence.h
#pragma once

#include <iostream>
#include <fstream>
#include <stdio.h>
#include <cmath>
#include <string>

#include "DopFunc.h"

class AnalysisTritSequence
{
        private:
                std::string FileNameIn;
                std::string DirNameOut;
                unsigned long KolPosledovatelnosteu;
                unsigned long KolTritOdnouPosledovatelnosti;

                std::string FileNameOut;
```

```
		std::string FileNameOut_FrequencyTritTest;
		std::string FileNameOut_FrequencyTritTestWithinBlock;
		std::string FileNameOut_RunsTritTest;
		std::string FileNameOut_LongestRunOfTrit;
		std::string FileNameOut_NonOverlappingTemplateTritTest;
		std::string FileNameOut_OverlappingTemplateTritTest;
		std::string FileNameOut_UniversalTritTest;
		std::string FileNameOut_SerialTrittest;

		void FrequencyTritTest( void ); /// 1
		void FrequencyTritTestWithinBlock( void ); /// 2
		void RunsTritTest( void ); /// 3
		void LongestRunOfTrit( void ); /// 4
		void NonOverlappingTemplateTritTest( void ); /// 5
		void OverlappingTemplateTritTest( void ); /// 6
		void VuvestiWapkyTrit( void );
		void VuvestiTestTrit( char *nameTest, int C01[], int C02[], int C12[], int C012_Neproudeno );
		void VuvestiPodvalTrit( void );
		std::string Poly4itWablon(int Chuslo, char X, char Y, int m);

	public:
		AnalysisTritSequence( void );
		~AnalysisTritSequence( void );
		void RunTests( char *FileNameIn_, unsigned long KolPosledovatelnosteu_, unsigned long
KolTritOdnouPosledovatelnosti_, char *DirNameOut_ );
};

/// AnalysisTritSequence.cpp
#include "StdAfx.h"
#include "AnalysisTritSequence.h"

std::ofstream OutFileNew;
AnalysisTritSequence::AnalysisTritSequence( void )
{
	KolPosledovatelnosteu				= 0;
	KolTritOdnouPosledovatelnosti		= 0;
	OutFileNew.open( "D:\\trit\\Rez.xls" );
}
AnalysisTritSequence::~AnalysisTritSequence( void )
{
}
void AnalysisTritSequence::RunTests( char *FileNameIn_, unsigned long KolPosledovatelnosteu_, unsigned long
KolTritOdnouPosledovatelnosti_, char *DirNameOut_ )
{
	FileNameIn				= FileNameIn_;
	KolPosledovatelnosteu		= KolPosledovatelnosteu_;
	KolTritOdnouPosledovatelnosti	= KolTritOdnouPosledovatelnosti_;
	DirNameOut				= DirNameOut_;

	FileNameOut				= DirNameOut + "Result.txt";
	FileNameOut_FrequencyTritTest	= DirNameOut + "1. FrequencyTritTest.xls";
	FileNameOut_FrequencyTritTestWithinBlock	= DirNameOut + "2. FrequencyTritTestWithinBlock.xls";
	FileNameOut_RunsTritTest		= DirNameOut + "3. RunsTritTest.xls";
	FileNameOut_LongestRunOfTrit	= DirNameOut + "4. LongestRunOfTrit.xls";
	FileNameOut_NonOverlappingTemplateTritTest= DirNameOut + "5. NonOverlappingTemplateTritTest.xls";
	FileNameOut_OverlappingTemplateTritTest		= DirNameOut + "6. OverlappingTemplateTritTest.xls";

	VuvestiWapkyTrit();

	FrequencyTritTest();/// 1 +++
	FrequencyTritTestWithinBlock();/// 2 +++
	RunsTritTest();/// 3 +++
	LongestRunOfTrit();///4 +++
	NonOverlappingTemplateTritTest();///5 +++
	OverlappingTemplateTritTest();///6 +++
	VuvestiPodvalTrit();
}

void AnalysisTritSequence::FrequencyTritTest( void )
{
	FILE *file_read;
	file_read = fopen( FileNameIn.c_str(), "r" );
```

```
std::ofstream out_file;
out_file.open( FileNameOut_FrequencyTritTest.c_str() );

out_file << "\n\tTest" << "\tS01"        << "\tPv01"
                                          << "\tS02"        << "\tPv02"
                                          << "\tS12"        << "\tPv12"
                                          << "\tP1"         << "\tP2"
                                          << "\tKol01" << "\tKol02" << "\tKol12";

std::cout<< "\nTesting FrequencyTrit...\n";

int C01[10]={0};
int C02[10]={0};
int C12[10]={0};
int C012_Neproudeno = 0;

int Index = 0;
for( unsigned long i = 0; i < KolPosledovatelnosteu; i++ )
{
        double S01 = 0;
        double S02 = 0;
        double S12 = 0;
        double KolTrit01 = 0;
        double KolTrit02 = 0;
        double KolTrit12 = 0;
        double Pvalue01 = 0;
        double Pvalue02 = 0;
        double Pvalue12 = 0;
        bool P1;
        bool P2;

        for( unsigned long j = 0; j < KolTritOdnouPosledovatelnosti; j++ )
        {
                char OneTrit;
                int numread = fread( &OneTrit, 1, 1, file_read );
                Index = Index + 1;
                if( numread == 0)
                {
                        break;
                }
                if (OneTrit == '0')
                {
                        S01 = S01 - 1;
                        S02 = S02 - 1;
                        KolTrit01++;
                        KolTrit02++;
                }
                if (OneTrit == '1')
                {
                        S01 = S01 + 1;
                        S12 = S12 - 1;
                        KolTrit01++;
                        KolTrit12++;
                }
                if (OneTrit == '2')
                {
                        S02 = S02 + 1;
                        S12 = S12 + 1;
                        KolTrit02++;
                        KolTrit12++;
                }
        }
        S01 = (double) fabs( S01 ) / sqrt( KolTrit01 );
        S02 = (double) fabs( S02 ) / sqrt( KolTrit02 );
        S12 = (double) fabs( S12 ) / sqrt( KolTrit12 );

        Pvalue01 = erfc( S01 / sqrt( (double) 2 ) );
        Pvalue02 = erfc( S02 / sqrt( (double) 2 ) );
        Pvalue12 = erfc( S12 / sqrt( (double) 2 ) );

        if ( ( Pvalue01 >= 0.01 ) && ( Pvalue02 >= 0.01 ) && ( Pvalue12 >= 0.01 ) )
```

```
                {
                        P1 = false;
                        P2 = false;
                }
                else
                {
                        if ( ( Pvalue01 < 0.001 ) || ( Pvalue02 < 0.001 ) && ( Pvalue12 < 0.001 ) )
                        {
                                P1 = true;
                                P2 = true;
                        }
                        else
                        {
                                P1 = true;
                                P2 = false;
                        }
                }

                out_file.setf(std::ios::fixed);
                out_file.precision(6);
                out_file << "\n\t" << i << "\t" << S01 << "\t" << Pvalue01
                                                                << "\t" << S02 << "\t" << Pvalue02
                                                                << "\t" << S12 << "\t" << Pvalue12
                                                                << "\t" << P1  << "\t" << P2
                                                                << "\t" << KolTrit01 << "\t" << KolTrit02 << "\t" <<
KolTrit12;

                OutFileNew.setf(std::ios::fixed);
                OutFileNew.precision(6);
                OutFileNew << "\n" << Pvalue01<< "\t" << Pvalue02<< "\t" << Pvalue12;

                int Pv01 = ( Pvalue01 == 1 )?( 9 ):( (int) ( Pvalue01 * 10 ) );
                int Pv02 = ( Pvalue02 == 1 )?( 9 ):( (int) ( Pvalue02 * 10 ) );
                int Pv12 = ( Pvalue12 == 1 )?( 9 ):( (int) ( Pvalue12 * 10 ) );

                C01[Pv01]++;
                C02[Pv02]++;
                C12[Pv12]++;

                if ( ( Pvalue01 < 0.001 ) || ( Pvalue02 < 0.001 ) || ( Pvalue12 < 0.001 ) )
                {
                        C012_Neproudeno++;
                }
        }

        VuvestiTestTrit( "FrequencyTrit", C01, C02, C12, C012_Neproudeno );

        fseek( file_read, 0, SEEK_END );
        long filesize = ftell( file_read );
  fseek( file_read, 0, SEEK_SET );

        fclose( file_read );
}

void AnalysisTritSequence::FrequencyTritTestWithinBlock( void )
{
        FILE *file_read;
        file_read  = fopen( FileNameIn.c_str(), "r" );

        std::ofstream out_file;
        out_file.open( FileNameOut_FrequencyTritTestWithinBlock.c_str() );

        int M = 0;
        int KolBlokob192trut = KolTritOdnouPosledovatelnosti / 192;

        if( ( KolBlokob192trut >= 100 ) && (M == 0 ) )
        {
                M = 192;
        }
        if( ( KolBlokob192trut >= 50 ) && (M == 0 ) )
        {
                M = 96;
```

```
}
if( ( KolBlokob192trut >= 25 ) && (M == 0 ) )
{
	M = 48;
}
if( ( KolBlokob192trut >= 12 ) && (M == 0 ) )
{
	M = 24;
}
if( ( KolBlokob192trut >= 6 ) && (M == 0 ) )
{
	M = 12;
}
if( M == 0 )
{
	M = 4;
}

out_file << "\n\tTest" << "\tFiKv01" << "\tPvalue01"
                                                << "\tFiKv02" << "\tPvalue02"
                                                << "\tFiKv12" << "\tPvalue12"
                                                << "\tP1"  << "\tP2";

std::cout<< "\nTesting BlockFrequencyTrit...\n";

int C01[10]={0};
int C02[10]={0};
int C12[10]={0};
int C012_Neproudeno = 0;

int Index = 0;
for( unsigned long i = 0; i < KolPosledovatelnosteu; i++ )
{
	double FiKv01	= 0;
	double FiKv02	= 0;
	double FiKv12	= 0;
	double Pvalue01 = 0;
	double Pvalue02 = 0;
	double Pvalue12 = 0;
	bool P1;
	bool P2;

	int IndexN01	= 0;
	int IndexN02	= 0;
	int IndexN12	= 0;
	double Pi01		= 0;
	double Pi02		= 0;
	double Pi12		= 0;
	int sum01		= 0;
	int sum02		= 0;
	int sum12		= 0;
	int N01		= 0;
	int N02		= 0;
	int N12		= 0;
	for( unsigned long j = 0; j < KolTritOdnouPosledovatelnosti; j++ )
	{
		char OneTrit;
		int numread = fread( &OneTrit, 1, 1, file_read );
		Index++;
		if( numread == 0)
		{
			break;
		}
		if (OneTrit == '0')
		{
			IndexN01++;
			IndexN02++;
			sum02++;
		}
		if (OneTrit == '1')
		{
			IndexN01++;
```

```
                IndexN12++;
                sum01++;
        }
        if (OneTrit == '2')
        {
                IndexN02++;
                IndexN12++;
                sum12++;
        }
        if( ( IndexN01 > 0 ) && ( IndexN01 % M == 0 ) )
        {
                Pi01 = Pi01 + ( (double) sum01 / M - (double) 1/2 ) * ( (double) sum01 / M - (double) 1/2 );
                IndexN01        = 0;
                sum01           = 0;
                N01++;
        }
        if( ( IndexN02 > 0 ) && ( IndexN02 % M == 0 ) )
        {
                Pi02 = Pi02 + ( (double) sum02 / M - (double) 1/2 ) * ( (double) sum02 / M - (double) 1/2 );
                IndexN02        = 0;
                sum02           = 0;
                N02++;
        }
        if( ( IndexN12 > 0 ) && ( IndexN12 % M == 0 ) )
        {
                Pi12 = Pi12 + ( (double) sum12 / M - (double) 1/2 ) * ( (double) sum12 / M - (double) 1/2 );
                IndexN12        = 0;
                sum12           = 0;
                N12++;
        }
}
FiKv01 = (double) 4 * M * Pi01;
FiKv02 = (double) 4 * M * Pi02;
FiKv12 = (double) 4 * M * Pi12;

Pvalue01 = cephes_igamc( (double) N01/2, (double) FiKv01/2 );
Pvalue02 = cephes_igamc( (double) N02/2, (double) FiKv02/2 );
Pvalue12 = cephes_igamc( (double) N12/2, (double) FiKv12/2 );

if ( ( Pvalue01 >= 0.01 ) && ( Pvalue02 >= 0.01 ) && ( Pvalue12 >= 0.01 ) )
{
        P1 = false;
        P2 = false;
}
else
{
        if ( ( Pvalue01 < 0.001 ) || ( Pvalue02 < 0.001 ) && ( Pvalue12 < 0.001 ) )
        {
                P1 = true;
                P2 = true;
        }
        else
        {
                P1 = true;
                P2 = false;
        }
}

out_file.setf(std::ios::fixed);
out_file.precision(6);
out_file << "\n\t" << i << "\t" << FiKv01 << "\t" << Pvalue01
                                                << "\t" << FiKv02 << "\t" << Pvalue02
                                                << "\t" << FiKv12 << "\t" << Pvalue12
                                                << "\t" << P1  << "\t" << P2;

OutFileNew.setf(std::ios::fixed);
OutFileNew.precision(6);
OutFileNew << "\n" << Pvalue01<< "\t" << Pvalue02<< "\t" << Pvalue12;

int Pv01 = ( Pvalue01 == 1 )?( 9 ):( (int) ( Pvalue01 * 10 ) );
int Pv02 = ( Pvalue02 == 1 )?( 9 ):( (int) ( Pvalue02 * 10 ) );
int Pv12 = ( Pvalue12 == 1 )?( 9 ):( (int) ( Pvalue12 * 10 ) );
```

```
                C01[Pv01]++;
                C02[Pv02]++;
                C12[Pv12]++;

                if ( ( Pvalue01 < 0.001 ) || ( Pvalue02 < 0.001 ) || ( Pvalue12 < 0.001 ) )
                {
                        C012_Neproudeno++;
                }
        }

        VuvestiTestTrit( "BlockFrequencyTrit", C01, C02, C12, C012_Neproudeno );

        fseek( file_read, 0, SEEK_END );
        long filesize = ftell( file_read );
   fseek( file_read, 0, SEEK_SET );

        fclose( file_read );
}

void AnalysisTritSequence::RunsTritTest( void )
{
        FILE *file_read;
        file_read  = fopen( FileNameIn.c_str(), "r" );

        std::ofstream out_file;
        out_file.open( FileNameOut_RunsTritTest.c_str() );

        out_file << "\n\tTest" << "\tS01" << "\tPvalue01"
                                                << "\tS02" << "\tPvalue02"
                                                << "\tS12" << "\tPvalue12"
                                                << "\tP1"  << "\tP2";

        std::cout<< "\nRunsTrit...\n";

        int C01[10]={0};
        int C02[10]={0};
        int C12[10]={0};
        int C012_Neproudeno = 0;

        int Index = 0;
        for( unsigned long i = 0; i < KolPosledovatelnosteu; i++ )
        {
                double S0                       = 0;
                double S1                       = 0;
                double S2                       = 0;
                double D01                      = 0;
                double D02                      = 0;
                double D12                      = 0;
                double S01                      = 0;
                double S02                      = 0;
                double S12                      = 0;
                double PS01                     = 0;
                double PS02                     = 0;
                double PS12                     = 0;
                double Vabs01      = 0;
                double Vabs02      = 0;
                double Vabs12      = 0;
                double Pvalue01 = 0;
                double Pvalue02 = 0;
                double Pvalue12 = 0;
                bool P1;
                bool P2;

                char LastTrit01 = '*';
                char LastTrit02 = '*';
                char LastTrit12 = '*';
                for( unsigned long j = 0; j < KolTritOdnouPosledovatelnosti; j++ )
                {
                        char OneTrit;
                        int numread = fread( &OneTrit, 1, 1, file_read );
                        Index = Index + 1;
```

```
                if( numread == 0)
                {
                        break;
                }
                if (OneTrit == '0')
                {
                        S0 = S0 + 1;
                }
                if (OneTrit == '1')
                {
                        S1 = S1 + 1;
                }
                if (OneTrit == '2')
                {
                        S2 = S2 + 1;
                }

                if( OneTrit == '0' || OneTrit == '1' )
                {
                        if ( ( LastTrit01 != '*') && ( LastTrit01!= OneTrit) )
                        {
                                Vabs01 = Vabs01 + 1;
                        }
                        LastTrit01 = OneTrit;
                }
                if( OneTrit == '0' || OneTrit == '2' )
                {
                        if ( ( LastTrit02 != '*') && ( LastTrit02!= OneTrit) )
                        {
                                Vabs02 = Vabs02 + 1;
                        }
                        LastTrit02 = OneTrit;
                }
                if( OneTrit == '1' || OneTrit == '2' )
                {
                        if ( ( LastTrit12 != '*') && ( LastTrit12!= OneTrit) )
                        {
                                Vabs12 = Vabs12 + 1;
                        }
                        LastTrit12 = OneTrit;
                }
        }

        Vabs01 = Vabs01 + 1;
        Vabs02 = Vabs02 + 1;
        Vabs12 = Vabs12 + 1;

        D01 = (S0 + S1);
        D02 = (S0 + S2);
        D12 = (S1 + S2);

        S01     = (double) S0 / D01;
        S02     = (double) S2 / D02;
        S12     = (double) S1 / D12;

        PS01 = (double) fabs( S01 - ( (double) 1 / 2 ) );
        PS02 = (double) fabs( S02 - ( (double) 1 / 2 ) );
        PS12 = (double) fabs( S12 - ( (double) 1 / 2 ) );

        if( ( PS01 >= ( (double) 2 / sqrt( (double) D01) ) ) )
        {
                Pvalue01 = 0;
        }
        else
        {
                double temp01;
                temp01 = (double) ( fabs( Vabs01 - (double) 2 * D01 * S01 * ( 1 - S01 ) ) ) / ( (double) 2 * sqrt( (double) 2
* D01 ) * S01 * ( 1 - S01 ) );
                Pvalue01 = erfc(temp01);
        }

        if( ( PS02 >= ( (double) 2 / sqrt( (double) D02) ) ) )
```

```
                {
                        Pvalue02 = 0;
                }
                else
                {
                        double temp02;
                        temp02 = (double) ( fabs( Vabs02 - (double) 2 * D02 * S02 * ( 1 - S02 ) ) ) / ( (double) 2 * sqrt( (double) 2
* D02 ) * S02 * ( 1 - S02 ) );
                        Pvalue02 = erfc(temp02);
                }

                if( ( PS12 >= ( (double) 2 / sqrt( (double) D12) ) ) )
                {
                        Pvalue12 = 0;
                }
                else
                {
                        double temp12;
                        temp12 = (double) ( fabs( Vabs12 - (double) 2 * D12 * S12 * ( 1 - S12 ) ) ) / ( (double) 2 * sqrt( (double) 2
* D12 ) * S12 * ( 1 - S12 ) );
                        Pvalue12 = erfc(temp12);
                }

                if ( ( Pvalue01 >= 0.01 ) && ( Pvalue02 >= 0.01 ) && ( Pvalue12 >= 0.01 ) )
                {
                        P1 = false;
                        P2 = false;
                }
                else
                {
                        if ( ( Pvalue01 < 0.001 ) || ( Pvalue02 < 0.001 ) && ( Pvalue12 < 0.001 ) )
                        {
                                P1 = true;
                                P2 = true;
                        }
                        else
                        {
                                P1 = true;
                                P2 = false;
                        }
                }

                out_file.setf(std::ios::fixed);
                out_file.precision(6);
                out_file << "\n\t" << i << "\t" << S01 << "\t" << Pvalue01
                                                                        << "\t" << S02 << "\t" << Pvalue02
                                                                        << "\t" << S12 << "\t" << Pvalue12
                                                                        << "\t" << P1  << "\t" << P2;
                OutFileNew.setf(std::ios::fixed);
                OutFileNew.precision(6);
                OutFileNew << "\n" << Pvalue01<< "\t" << Pvalue02<< "\t" << Pvalue12;

                int Pv01 = ( Pvalue01 == 1 )?( 9 ):( (int) ( Pvalue01 * 10 ) );
                int Pv02 = ( Pvalue02 == 1 )?( 9 ):( (int) ( Pvalue02 * 10 ) );
                int Pv12 = ( Pvalue12 == 1 )?( 9 ):( (int) ( Pvalue12 * 10 ) );

                C01[Pv01]++;
                C02[Pv02]++;
                C12[Pv12]++;

                if ( ( Pvalue01 < 0.001 ) || ( Pvalue02 < 0.001 ) || ( Pvalue12 < 0.001 ) )
                {
                        C012_Neproudeno++;
                }
        }

        VuvestiTestTrit( "RunsTrit", C01, C02, C12, C012_Neproudeno );

        fseek( file_read, 0, SEEK_END );
        long filesize = ftell( file_read );
fseek( file_read, 0, SEEK_SET );
```

```
	fclose( file_read );
}

/// +++
void AnalysisTritSequence::LongestRunOfTrit( void )
{
	FILE *file_read;
	file_read  = fopen( FileNameIn.c_str(), "r" );

	std::ofstream out_file;
	out_file.open( FileNameOut_LongestRunOfTrit.c_str() );

	int M;
	int K;
	double PP[7] = {0};
	if (KolTritOdnouPosledovatelnosti <= 9408)
	{
		M = 8;
		K = 3;
		PP[0] = 0.2148; PP[1] = 0.3672;
		PP[2] = 0.2305; PP[3] = 0.1875;
	}
	else
	{
		if (KolTritOdnouPosledovatelnosti <= 1125000)
		{
			M = 128;
			K = 5;
			PP[0] = 0.1174; PP[1] = 0.2430;
			PP[2] = 0.2493; PP[3] = 0.1752;
			PP[4] = 0.1027; PP[5] = 0.1124;
		}
		else
		{
			M = 10000;
			K = 6;
			PP[0] = 0.0882; PP[1] = 0.2092;
			PP[2] = 0.2483; PP[3] = 0.1933;
			PP[4] = 0.1208; PP[5] = 0.0675;
			PP[6] = 0.0727;
		}
	}

	out_file << "\n\tTest" << "\tFiKv01" << "\tPvalue01"
                                                << "\tFiKv02" << "\tPvalue02"
                                                << "\tFiKv12" << "\tPvalue12"
                                                << "\tP1"  << "\tP2";

	std::cout<< "\nLongestRunTrit...\n";

	int C01[10]={0};
	int C02[10]={0};
	int C12[10]={0};
	int C012_Neproudeno = 0;

	int Index = 0;
	for( unsigned long i = 0; i < KolPosledovatelnosteu; i++ )
	{
		double v01[7]	= { 0 };
		double v02[7]	= { 0 };
		double v12[7]	= { 0 };
		double FiKv01	= 0;
		double FiKv02	= 0;
		double FiKv12	= 0;
		double Pvalue01	= 0;
		double Pvalue02	= 0;
		double Pvalue12	= 0;
		bool P1;
		bool P2;

		int IndexN01 = 0;
		int IndexN02 = 0;
```

```
int IndexN12 = 0;

int N01 = 0;
int N02 = 0;
int N12 = 0;

int tek01 = 0;
int tek02 = 0;
int tek12 = 0;

int max01 = 0;
int max02 = 0;
int max12 = 0;

char lastTrit01 = '*';
char lastTrit02 = '*';
char lastTrit12 = '*';
for( unsigned long j = 0; j < KolTritOdnouPosledovatelnosti; j++ )
{
	char OneTrit;
	int numread = fread( &OneTrit, 1, 1, file_read );
	Index++;

	if( numread == 0)
	{
		break;
	}

	if (OneTrit == '0')
	{
		IndexN01++;
		IndexN02++;
		if (lastTrit01 == OneTrit)
		{
			tek01++;
		}
		else
		{
			if (max01 < tek01)
			{
				max01 = tek01;
			}
			tek01 = 1;
		}
		lastTrit01 = OneTrit;

		if(lastTrit02 == OneTrit)
		{
			tek02++;
		}
		else
		{
			if (max02 < tek02)
			{
				max02 = tek02;
			}
			tek02 = 1;
		}
		lastTrit02 = OneTrit;
	}
	if (OneTrit == '1')
	{
		IndexN01++;
		IndexN12++;
		if(lastTrit12 == OneTrit)
		{
			tek12++;
		}
		else
		{
			if (max12 < tek12)
			{
```

```
                                        max12 = tek12;
                                }
                                tek12 = 1;
                        }
                        lastTrit01 = OneTrit;
                        lastTrit12 = OneTrit;
                        if (max01 < tek01)
                        {
                                max01 = tek01;
                        }
                        tek01 = 0;
                }
                if (OneTrit == '2')
                {
                        IndexN02++;
                        IndexN12++;
                        lastTrit02 = OneTrit;
                        lastTrit12 = OneTrit;
                        if (max02 < tek02)
                        {
                                max02 = tek02;
                        }
                        tek02 = 0;
                        if (max12 < tek12)
                        {
                                max12 = tek12;
                        }
                        tek12 = 0;
                }

                if ( (IndexN01 != 0) && (IndexN01 % M == 0) )
                {
                        if (max01 < tek01)
                        {
                                max01 = tek01;
                        }

                        if ( M == 8 )
                        {
                                if( max01 <= 1 ) v01[0]++;
                                if( max01 == 2 ) v01[1]++;
                                if( max01 == 3 ) v01[2]++;
                                if( max01 >= 4 ) v01[3]++;
                        }
                        if ( M == 128 )
                        {
                                if( max01 <= 4 ) v01[0]++;
                                if( max01 == 5 ) v01[1]++;
                                if( max01 == 6 ) v01[2]++;
                                if( max01 == 7 ) v01[3]++;
                                if( max01 == 8 ) v01[4]++;
                                if( max01 >= 9 ) v01[5]++;
                        }
                        if ( M == 10000 )
                        {
                                if( max01 <= 10 ) v01[0]++;
                                if( max01 == 11 ) v01[1]++;
                                if( max01 == 12 ) v01[2]++;
                                if( max01 == 13 ) v01[3]++;
                                if( max01 == 14 ) v01[4]++;
                                if( max01 == 15 ) v01[5]++;
                                if( max01 >= 16 ) v01[6]++;
                        }
                        max01 = 0; tek01 = 0; lastTrit01 = '*'; N01++; IndexN01 = 0;
                }

                if ( (IndexN02 != 0) && (IndexN02 % M == 0) )
                {
                        if (max02 < tek02)
                        {
                                max02 = tek02;
                        }
```

```
                    if ( M == 8 )
                    {
                        if( max02 <= 1 ) v02[0]++;
                        if( max02 == 2 ) v02[1]++;
                        if( max02 == 3 ) v02[2]++;
                        if( max02 >= 4 ) v02[3]++;
                    }
                    if ( M == 128 )
                    {
                        if( max02 <= 4 ) v02[0]++;
                        if( max02 == 5 ) v02[1]++;
                        if( max02 == 6 ) v02[2]++;
                        if( max02 == 7 ) v02[3]++;
                        if( max02 == 8 ) v02[4]++;
                        if( max02 >= 9 ) v02[5]++;
                    }
                    if ( M == 10000 )
                    {
                        if( max02 <= 10 ) v02[0]++;
                        if( max02 == 11 ) v02[1]++;
                        if( max02 == 12 ) v02[2]++;
                        if( max02 == 13 ) v02[3]++;
                        if( max02 == 14 ) v02[4]++;
                        if( max02 == 15 ) v02[5]++;
                        if( max02 >= 16 ) v02[6]++;
                    }
                    max02 = 0; tek02 = 0; lastTrit02 = '*'; N02++; IndexN02 = 0;
                }

                if ( (IndexN12 != 0) && (IndexN12 % M == 0) )
                {
                    if (max12 < tek12)
                    {
                        max12 = tek12;
                    }

                    if ( M == 8 )
                    {
                        if( max12 <= 1 ) v12[0]++;
                        if( max12 == 2 ) v12[1]++;
                        if( max12 == 3 ) v12[2]++;
                        if( max12 >= 4 ) v12[3]++;
                    }
                    if ( M == 128 )
                    {
                        if( max12 <= 4 ) v12[0]++;
                        if( max12 == 5 ) v12[1]++;
                        if( max12 == 6 ) v12[2]++;
                        if( max12 == 7 ) v12[3]++;
                        if( max12 == 8 ) v12[4]++;
                        if( max12 >= 9 ) v12[5]++;
                    }
                    if ( M == 10000 )
                    {
                        if( max12 <= 10 ) v12[0]++;
                        if( max12 == 11 ) v12[1]++;
                        if( max12 == 12 ) v12[2]++;
                        if( max12 == 13 ) v12[3]++;
                        if( max12 == 14 ) v12[4]++;
                        if( max12 == 15 ) v12[5]++;
                        if( max12 >= 16 ) v12[6]++;
                    }
                    max12 = 0; tek12 = 0; lastTrit12 = '*'; N12++; IndexN12 = 0;
                }
            }

            for( int k = 0; k <= K; k++ )
            {
                FiKv01 = FiKv01 + pow( ( v01[k] - (double) N01  * PP[k] ), 2 ) / ( (double) N01  * PP[k] );
                FiKv02 = FiKv02 + pow( ( v02[k] - (double) N02  * PP[k] ), 2 ) / ( (double) N02  * PP[k] );
                FiKv12 = FiKv12 + pow( ( v12[k] - (double) N12  * PP[k] ), 2 ) / ( (double) N12  * PP[k] );
```

```
                }

                Pvalue01 = cephes_igamc( (double) K/2, (double) FiKv01/2 );
                Pvalue02 = cephes_igamc( (double) K/2, (double) FiKv02/2 );
                Pvalue12 = cephes_igamc( (double) K/2, (double) FiKv12/2 );

                if ( ( Pvalue01 >= 0.01 ) && ( Pvalue02 >= 0.01 ) && ( Pvalue12 >= 0.01 ) )
                {
                        P1 = false;
                        P2 = false;
                }
                else
                {
                        if ( ( Pvalue01 < 0.001 ) || ( Pvalue02 < 0.001 ) && ( Pvalue12 < 0.001 ) )
                        {
                                P1 = true;
                                P2 = true;
                        }
                        else
                        {
                                P1 = true;
                                P2 = false;
                        }
                }

                out_file.setf(std::ios::fixed);
                out_file.precision(6);
                out_file << "\n\t" << i << "\t" << FiKv01 << "\t" << Pvalue01
                                                                << "\t" << FiKv02 << "\t" << Pvalue02
                                                                << "\t" << FiKv12 << "\t" << Pvalue12
                                                                << "\t" << P1  << "\t" << P2;
                OutFileNew.setf(std::ios::fixed);
                OutFileNew.precision(6);
                OutFileNew << "\n" << Pvalue01<< "\t" << Pvalue02<< "\t" << Pvalue12;

                int Pv01 = ( Pvalue01 == 1 )?( 9 ):( (int) ( Pvalue01 * 10 ) );
                int Pv02 = ( Pvalue02 == 1 )?( 9 ):( (int) ( Pvalue02 * 10 ) );
                int Pv12 = ( Pvalue12 == 1 )?( 9 ):( (int) ( Pvalue12 * 10 ) );

                C01[Pv01]++;
                C02[Pv02]++;
                C12[Pv12]++;

                if ( ( Pvalue01 < 0.001 ) || ( Pvalue02 < 0.001 ) || ( Pvalue12 < 0.001 ) )
                {
                        C012_Neproudeno++;
                }
        }

        VuvestiTestTrit( "LongestRunTrit", C01, C02, C12, C012_Neproudeno );

        fseek( file_read, 0, SEEK_END );
        long filesize = ftell( file_read );
  fseek( file_read, 0, SEEK_SET );

        fclose( file_read );
}

void AnalysisTritSequence::NonOverlappingTemplateTritTest( void )
{
        int m = 9;
        int N = 8;
        unsigned long M012          = KolTritOdnouPosledovatelnosti / N;
        unsigned long M    = M012 * 2/3;

        std::ofstream out_file;
        out_file.open( FileNameOut_NonOverlappingTemplateTritTest.c_str() );

        out_file << "\n\tWablon" << "\tTest" << "\tFiKv01" << "\tPvalue01"
                                                        << "\tFiKv02" << "\tPvalue02"
                                                        << "\tFiKv12" << "\tPvalue12"
                                                        << "\tP1"  << "\tP2";
```

```
int predel = pow( (double) 2, m);
if (predel >= 148) predel = 148;
for (int xx = 1; xx <= predel; xx++)//berem poka vse vozmoznue wablonu
{
	FILE *file_read;
	file_read  = fopen( FileNameIn.c_str(), "r" );
	std::cout << "\nNonOverlappingTemplateTritTest[ " << xx << " ]...\n";

	std::string Wablon01 = Poly4itWablon(xx, '0', '1', m);
	std::string Wablon02 = Poly4itWablon(xx, '0', '2', m);
	std::string Wablon12 = Poly4itWablon(xx, '1', '2', m);

	int C01[10]={0};
	int C02[10]={0};
	int C12[10]={0};
	int C012_Neproudeno = 0;

	int Index = 0;
	for( unsigned long i = 0; i < KolPosledovatelnosteu; i++ )
	{
		double FiKv01	= 0;
		double FiKv02	= 0;
		double FiKv12	= 0;
		double Pvalue01	= 0;
		double Pvalue02	= 0;
		double Pvalue12	= 0;
		bool P1;
		bool P2;

		unsigned long IndexN01 = 0;
		unsigned long IndexN02 = 0;
		unsigned long IndexN12 = 0;

		unsigned long N01 = 0;
		unsigned long N02 = 0;
		unsigned long N12 = 0;

		unsigned long W01[20] = {0};
		unsigned long W02[20] = {0};
		unsigned long W12[20] = {0};
		unsigned long IndexBlok01 = 0;
		unsigned long IndexBlok02 = 0;
		unsigned long IndexBlok12 = 0;

		std::string Tek01;
		std::string Tek02;
		std::string Tek12;

		for( unsigned long j = 0; j < KolTritOdnouPosledovatelnosti; j++ )
		{
			char OneTrit;
			int numread = fread( &OneTrit, 1, 1, file_read );
			Index++;

			if( numread == 0)
			{
				break;
			}

			if (OneTrit == '0')
			{
				N01++;
				N02++;
				Tek01 = Tek01 + OneTrit;
				Tek02 = Tek02 + OneTrit;
				IndexN01++;
				IndexN02++;
			}

			if (OneTrit == '1')
			{
				N01++;
```

```
                N12++;
                Tek01 = Tek01 + OneTrit;
                Tek12 = Tek12 + OneTrit;
                IndexN01++;
                IndexN12++;
}

if (OneTrit == '2')
{
                N02++;
                N12++;
                Tek02 = Tek02 + OneTrit;
                Tek12 = Tek12 + OneTrit;
                IndexN02++;
                IndexN12++;
}

/// 01
if ( (IndexN01 != 0) && (IndexN01 % m == 0) )
{
                if(Wablon01 == Tek01)
                {
                                W01[IndexBlok01]++;
                                IndexN01 = 0;
                                Tek01 = "";
                }
                else
                {
                                IndexN01--;
                                Tek01.erase(0,1);
                }
}

if ( (N01 != 0) && (N01 % M == 0) )
{
                IndexBlok01++;
                IndexN01 = 0;
                Tek01 = "";
                N01 = 0;
}

/// 02
if ( (IndexN02 != 0) && (IndexN02 % m == 0) )
{
                if(Wablon02 == Tek02)
                {
                                W02[IndexBlok02]++;
                                IndexN02 = 0;
                                Tek02 = "";
                }
                else
                {
                                IndexN02--;
                                Tek02.erase(0,1);
                }
}

if ( (N02 != 0) && (N02 % M == 0) )
{
                IndexBlok02++;
                IndexN02 = 0;
                Tek02 = "";
                N02 = 0;
}

/// 12
if ( (IndexN12 != 0) && (IndexN12 % m == 0) )
{
                if(Wablon12 == Tek12)
                {
                                W12[IndexBlok12]++;
                                IndexN12 = 0;
```

```
                                        Tek12 = "";
                                }
                                else
                                {
                                        IndexN12--;
                                        Tek12.erase(0,1);
                                }
                        }

                        if ( (N12 != 0) && (N12 % M == 0) )
                        {
                                IndexBlok12++;
                                IndexN12 = 0;
                                Tek12 = "";
                                N12 = 0;
                        }
                }

                double TeoreticSred = ( (double) ( M - m + 1) ) / ( pow((double) 2, m) );
                double Dispersiya  = ( (double) M ) * ( pow((double) 2, -m) - ( (double) (2 * m - 1)) / ( pow((double) 2,
2*m) ) );

                for(int ii = 0; ii < N; ii++)
                {
                        FiKv01 = FiKv01 + ( pow( (double) (W01[ii]-TeoreticSred), 2) ) / Dispersiya;
                        FiKv02 = FiKv02 + ( pow( (double) (W02[ii]-TeoreticSred), 2) ) / Dispersiya;
                        FiKv12 = FiKv12 + ( pow( (double) (W12[ii]-TeoreticSred), 2) ) / Dispersiya;
                }

                Pvalue01 = cephes_igamc( (double) N/2, (double) FiKv01/2 );
                Pvalue02 = cephes_igamc( (double) N/2, (double) FiKv02/2 );
                Pvalue12 = cephes_igamc( (double) N/2, (double) FiKv12/2 );

                if ( ( Pvalue01 >= 0.01 ) && ( Pvalue02 >= 0.01 ) && ( Pvalue12 >= 0.01 ) )
                {
                        P1 = false;
                        P2 = false;
                }
                else
                {
                        if ( ( Pvalue01 < 0.001 ) || ( Pvalue02 < 0.001 ) && ( Pvalue12 < 0.001 ) )
                        {
                                P1 = true;
                                P2 = true;
                        }
                        else
                        {
                                P1 = true;
                                P2 = false;
                        }
                }

                out_file.setf(std::ios::fixed);
                out_file.precision(6);
                out_file << "\n\t" << xx << "\t" << i << "\t" << FiKv01 << "\t" << Pvalue01
                                                                        << "\t" << FiKv02 << "\t" << Pvalue02
                                                                        << "\t" << FiKv12 << "\t" << Pvalue12
                                                                        << "\t" << P1  << "\t" << P2;

                OutFileNew.setf(std::ios::fixed);
                OutFileNew.precision(6);
                OutFileNew << "\n" << Pvalue01<< "\t" << Pvalue02<< "\t" << Pvalue12;

                int Pv01 = ( Pvalue01 == 1 )?( 9 ):( (int) ( Pvalue01 * 10 ) );
                int Pv02 = ( Pvalue02 == 1 )?( 9 ):( (int) ( Pvalue02 * 10 ) );
                int Pv12 = ( Pvalue12 == 1 )?( 9 ):( (int) ( Pvalue12 * 10 ) );

                C01[Pv01]++;
                C02[Pv02]++;
                C12[Pv12]++;

                if ( ( Pvalue01 < 0.001 ) || ( Pvalue02 < 0.001 ) || ( Pvalue12 < 0.001 ) )
```

```
                        {
                                C012_Neproudeno++;
                        }
                }

                VuvestiTestTrit( "NonOverlappingTemplateTritTest", C01, C02, C12, C012_Neproudeno );

                fseek( file_read, 0, SEEK_END );
                long filesize = ftell( file_read );
                fseek( file_read, 0, SEEK_SET );

                fclose( file_read );
        }
}

void AnalysisTritSequence::OverlappingTemplateTritTest( void )
{
        int m = 3;
        int N = 8;
        unsigned long M012              = KolTritOdnouPosledovatelnosti / N;
        unsigned long M     = M012 * 2/3;

        std::ofstream out_file;
        out_file.open( FileNameOut_OverlappingTemplateTritTest.c_str() );

        out_file << "\n\tWablon" << "\tTest" << "\tFiKv01" << "\tPvalue01"
                                                                << "\tFiKv02" << "\tPvalue02"
                                                                << "\tFiKv12" << "\tPvalue12"
                                                                << "\tP1"  << "\tP2";

        for (int xx = 5; xx < 6; xx++)
        {
                FILE *file_read;
                file_read  = fopen( FileNameIn.c_str(), "r" );
                std::cout << "\nOverlappingTemplateTritTest...\n";

                std::string Wablon01 = Poly4itWablon(xx, '0', '1', m);
                std::string Wablon02 = Poly4itWablon(xx, '0', '2', m);
                std::string Wablon12 = Poly4itWablon(xx, '1', '2', m);

                int C01[10]={0};
                int C02[10]={0};
                int C12[10]={0};
                int C012_Neproudeno = 0;

                int Index = 0;
                for( unsigned long i = 0; i < KolPosledovatelnosteu; i++ )
                {
                        double FiKv01       = 0;
                        double FiKv02       = 0;
                        double FiKv12       = 0;
                        double Pvalue01     = 0;
                        double Pvalue02     = 0;
                        double Pvalue12     = 0;
                        bool P1;
                        bool P2;

                        unsigned long IndexN01 = 0;
                        unsigned long IndexN02 = 0;
                        unsigned long IndexN12 = 0;

                        unsigned long N01 = 0;
                        unsigned long N02 = 0;
                        unsigned long N12 = 0;

                        unsigned long W01[20] = {0};
                        unsigned long W02[20] = {0};
                        unsigned long W12[20] = {0};
                        unsigned long IndexBlok01 = 0;
                        unsigned long IndexBlok02 = 0;
                        unsigned long IndexBlok12 = 0;

                        std::string Tek01;
```

```
std::string Tek02;
std::string Tek12;

for( unsigned long j = 0; j < KolTritOdnouPosledovatelnosti; j++ )
{
	char OneTrit;
	int numread = fread( &OneTrit, 1, 1, file_read );
	Index++;

	if( numread == 0)
	{
		break;
	}

	if (OneTrit == '0')
	{
		N01++;
		N02++;
		Tek01 = Tek01 + OneTrit;
		Tek02 = Tek02 + OneTrit;
		IndexN01++;
		IndexN02++;
	}

	if (OneTrit == '1')
	{
		N01++;
		N12++;
		Tek01 = Tek01 + OneTrit;
		Tek12 = Tek12 + OneTrit;
		IndexN01++;
		IndexN12++;
	}

	if (OneTrit == '2')
	{
		N02++;
		N12++;
		Tek02 = Tek02 + OneTrit;
		Tek12 = Tek12 + OneTrit;
		IndexN02++;
		IndexN12++;
	}

	/// 01
	if ( (IndexN01 != 0) && (IndexN01 % m == 0) )
	{
		if(Wablon01 == Tek01)
		{
			W01[IndexBlok01]++;
			IndexN01 = 0;
			Tek01 = "";
		}
		else
		{
			IndexN01--;
			Tek01.erase(0,1);
		}
	}

	if ( (N01 != 0) && (N01 % M == 0) )
	{
		IndexBlok01++;
		IndexN01 = 0;
		Tek01 = "";
		N01 = 0;
	}

	/// 02
	if ( (IndexN02 != 0) && (IndexN02 % m == 0) )
	{
		if(Wablon02 == Tek02)
```

```
                {
                        W02[IndexBlok02]++;
                        IndexN02 = 0;
                        Tek02 = "";
                }
                else
                {
                        IndexN02--;
                        Tek02.erase(0,1);
                }
        }

        if ( (N02 != 0) && (N02 % M == 0) )
        {
                IndexBlok02++;
                IndexN02 = 0;
                Tek02 = "";
                N02 = 0;
        }

        /// 12
        if ( (IndexN12 != 0) && (IndexN12 % m == 0) )
        {
                if(Wablon12 == Tek12)
                {
                        W12[IndexBlok12]++;
                        IndexN12 = 0;
                        Tek12 = "";
                }
                else
                {
                        IndexN12--;
                        Tek12.erase(0,1);
                }
        }

        if ( (N12 != 0) && (N12 % M == 0) )
        {
                IndexBlok12++;
                IndexN12 = 0;
                Tek12 = "";
                N12 = 0;
        }
}

double TeoreticSred = ( (double) ( M - m + 1) ) / ( pow((double) 2, m) );
double Dispersiya  = ( (double) M ) * ( pow((double) 2, -m) - ( (double) (2 * m - 1)) / ( pow((double) 2,
2*m) ) );


for(int ii = 0; ii < N; ii++)
{
        FiKv01 = FiKv01 + ( pow( (double) (W01[ii]-TeoreticSred), 2) ) / Dispersiya;
        FiKv02 = FiKv02 + ( pow( (double) (W02[ii]-TeoreticSred), 2) ) / Dispersiya;
        FiKv12 = FiKv12 + ( pow( (double) (W12[ii]-TeoreticSred), 2) ) / Dispersiya;
}

Pvalue01 = cephes_igamc( (double) N/2, (double) FiKv01/2 );
Pvalue02 = cephes_igamc( (double) N/2, (double) FiKv02/2 );
Pvalue12 = cephes_igamc( (double) N/2, (double) FiKv12/2 );

if ( ( Pvalue01 >= 0.01 ) && ( Pvalue02 >= 0.01 ) && ( Pvalue12 >= 0.01 ) )
{
        P1 = false;
        P2 = false;
}
else
{
        if ( ( Pvalue01 < 0.001 ) || ( Pvalue02 < 0.001 ) && ( Pvalue12 < 0.001 ) )
        {
                P1 = true;
                P2 = true;
        }
```

```
                                        else
                                        {
                                                P1 = true;
                                                P2 = false;
                                        }
                                }

                                out_file.setf(std::ios::fixed);
                                out_file.precision(6);
                                out_file << "\n\t" << xx << "\t" << i << "\t" << FiKv01 << "\t" << Pvalue01
                                                                                                << "\t" << FiKv02 << "\t" << Pvalue02
                                                                                                << "\t" << FiKv12 << "\t" << Pvalue12
                                                                                                << "\t" << P1  << "\t" << P2;

                                OutFileNew.setf(std::ios::fixed);
                                OutFileNew.precision(6);
                                OutFileNew << "\n" << Pvalue01<< "\t" << Pvalue02<< "\t" << Pvalue12;

                                int Pv01 = ( Pvalue01 == 1 )?( 9 ):( (int) ( Pvalue01 * 10 ) );
                                int Pv02 = ( Pvalue02 == 1 )?( 9 ):( (int) ( Pvalue02 * 10 ) );
                                int Pv12 = ( Pvalue12 == 1 )?( 9 ):( (int) ( Pvalue12 * 10 ) );

                                C01[Pv01]++;
                                C02[Pv02]++;
                                C12[Pv12]++;

                                if ( ( Pvalue01 < 0.001 ) || ( Pvalue02 < 0.001 ) || ( Pvalue12 < 0.001 ) )
                                {
                                        C012_Neproudeno++;
                                }
                        }

                        VuvestiTestTrit( "OverlappingTemplateTritTest", C01, C02, C12, C012_Neproudeno );

                        fseek( file_read, 0, SEEK_END );
                        long filesize = ftell( file_read );
                        fseek( file_read, 0, SEEK_SET );

                        fclose( file_read );
                }
}

void AnalysisTritSequence::VuvestiWapkyTrit( )
{
        std::ofstream out_file;
        out_file.open( FileNameOut.c_str() );

        out_file << "-----------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------\n";
        out_file << "RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING TRIT
SEQUENCES  \n";
        out_file << "-----------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------\n";
        out_file << "   generator is <"<< FileNameIn.c_str() <<">
                        \n";
        out_file << "-----------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------\n";
        out_file << "C1" << "\tC2" << "\tC3" << "\tC4" << "\tC5" << "\tC6" << "\tC7" << "\tC8" << "\tC9" << "\tC10" << "\tP-
VALUE1" << "\tP-VALUE2" << "\tP-VALUE3" << "\tPROPORTION" << "\tSTATISTICAL TEST\n";
        out_file << "-----------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------\n";
}

void AnalysisTritSequence::VuvestiTestTrit( char *nameTest, int C01[], int C02[], int C12[], int C012_Neproudeno )
{
        std::ofstream out_file;
        out_file.open( FileNameOut, std::ofstream::out | std::ofstream::app );

        unsigned int C[10] = {0};

        double AvgPvalue01          = 0;
        double AvgPvalue02          = 0;
```

```
	double AvgPvalue12		= 0;
	double Proportion	= 0;

	for( int i = 0; i < 10; i++)
	{
		C[i]= C01[i] + C02[i] + C12[i];
		AvgPvalue01 = AvgPvalue01 + pow( (double) ( C01[i] - (double) (KolPosledovatelnosteu / 10) ), 2);
		AvgPvalue02 = AvgPvalue02 + pow( (double) ( C02[i] - (double) (KolPosledovatelnosteu / 10) ), 2);
		AvgPvalue12 = AvgPvalue12 + pow( (double) ( C12[i] - (double) (KolPosledovatelnosteu / 10) ), 2);
	}
	AvgPvalue01 = AvgPvalue01 / ( (double) (KolPosledovatelnosteu / 10) );
	AvgPvalue02 = AvgPvalue02 / ( (double) (KolPosledovatelnosteu / 10) );
	AvgPvalue12 = AvgPvalue12 / ( (double) (KolPosledovatelnosteu / 10) );

	AvgPvalue01 = cephes_igamc( (double) 9/2, (double) AvgPvalue01/2 );
	AvgPvalue02 = cephes_igamc( (double) 9/2, (double) AvgPvalue02/2 );
	AvgPvalue12 = cephes_igamc( (double) 9/2, (double) AvgPvalue12/2 );

	Proportion = (double) KolPosledovatelnosteu - (double) C012_Neproudeno;
	Proportion = (double) Proportion / (double) KolPosledovatelnosteu;

	char *dop1;
	char *dop2;
	char *dop3;
	char *dop4;
	dop1 = (AvgPvalue01 < 0.01)?(" *"):("");
	dop2 = (AvgPvalue02 < 0.01)?(" *"):("");
	dop3 = (AvgPvalue12 < 0.01)?(" *"):("");
	dop4 = (Proportion < 0.97)?(" *"):("");

	out_file.setf(std::ios::fixed);
	out_file.precision(6);
	out_file	<< ""	<< C[0]
					<< "\t" << C[1]
					<< "\t" << C[2]
					<< "\t" << C[3]
					<< "\t" << C[4]
					<< "\t" << C[5]
					<< "\t" << C[6]
					<< "\t" << C[7]
					<< "\t" << C[8]
					<< "\t" << C[9]
					<< "\t" << AvgPvalue01 << dop1
					<< "\t" << AvgPvalue02 << dop2
					<< "\t" << AvgPvalue12 << dop3
					<< "\t" << Proportion << dop4
					<< "\t" << nameTest
					<< "\n";
}

void AnalysisTritSequence::VuvestiPodvalTrit( )
{
	std::ofstream out_file;
	out_file.open( FileNameOut, std::ofstream::out | std::ofstream::app );

	out_file << "\n";
	out_file << "\n";
	out_file << "- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - -\n";
	out_file << "The minimum pass rate for each statistical trit test with the exception of the\n";
	out_file << "random excursion (variant) test is approximately = 0.960150 for a\n";
	out_file << "sample size = 100 ternary sequences.\n";
	out_file << "\n";
	out_file << "The minimum pass rate for the random excursion (variant) test\n";
	out_file << "is approximately 0.954822 for a sample size = 72 ternary sequences.\n";
	out_file << "\n";
	out_file << "For further guidelines construct a probability table using the MAPLE program\n";
	out_file << "provided in the addendum section of the documentation.\n";
	out_file << "- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - -\n";
}
```

```
std::string AnalysisTritSequence::Poly4itWablon(int Chuslo, char X, char Y, int m)
{
	std::string Rez;

	for(int i = (m-1); i >= 0; i--)
	{
		int bit = (Chuslo >> i)&0x1;
		if (bit == 0)
		{
			Rez = Rez + X;
		}
		else
		{
			Rez = Rez + Y;
		}
	}

	return Rez;
}
```

4. Model of QSDC protocol

```
/// ModelQSDCprotocol.h
#pragma once

#include <fstream>
#include <iostream>
#include <string>

class ModelQSDCprotocol
{
	long double VgenNa4;
	long double VxNa4;
	long double VkvNa4;
	long double VklNa4;
	long double Wag;
	unsigned long KolWagov;
	long double l;
	long double r;
	int ProzedyraPidsulennyaBezpeku;
	public:
		ModelQSDCprotocol(void);
		~ModelQSDCprotocol(void);
		void Ras4itatSkorostRaspredeleniyaKly4eu( long double Vgen_na4, long double Vx_na4, long double Vkv_na4,
long double Vkl_na4, long double Razmer_, long double Wag_, unsigned long KolWagov_, char *ImyaFaulaRezyltata );
};

/// ModelQSDCprotocol.cpp
#include "StdAfx.h"
#include "ModelQSDCprotocol.h"
ModelQSDCprotocol::ModelQSDCprotocol(void)
{
}
ModelQSDCprotocol::~ModelQSDCprotocol(void)
{
}
void ModelQSDCprotocol::Ras4itatSkorostRaspredeleniyaKly4eu( long double Vgen_na4, long double Vx_na4, long double Vkv_na4,
long double Vkl_na4, long double Razmer_, long double Wag_, unsigned long KolWagov_, char *ImyaFaulaRezyltata )
{
	std::ofstream Vuhod;
	Vuhod.open( ImyaFaulaRezyltata );

	VgenNa4 = ( Vgen_na4 == 0 )?( 1 ):( Vgen_na4  );
	VxNa4		= ( Vx_na4 == 0    )?( 1 ):( Vx_na4     );
	VkvNa4		= ( Vkv_na4 == 0   )?( 1 ):( Vkv_na4    );
	VklNa4		= ( Vkl_na4 == 0   )?( 1 ):( Vkl_na4    );
	Wag		= ( Wag_ == 0                )?( 1 ):( Wag_                 );
	KolWagov	= ( KolWagov_ == 0  )?( 1 ):( KolWagov_ );
	Vuhod << "Vgen =";
	for( long double Vgen_index = VgenNa4; Vgen_index < ( VgenNa4 + KolWagov * Wag); Vgen_index += Wag )
	{
		for ( unsigned long j = 0; j < ( KolWagov * KolWagov * KolWagov ); j++)
		{
			Vuhod    << "\t" << Vgen_index << "\t" << Vgen_index;
```

```
		}
	}
	Vuhod << std::endl;
	Vuhod << "Vx =";
	for ( unsigned long i = 0; i < ( KolWagov ); i++ )
	{
		for( long double Vx_index = VxNa4; Vx_index < ( VxNa4 + KolWagov * Wag); Vx_index += Wag )
		{
			for ( unsigned long j = 0; j < ( KolWagov * KolWagov ); j++)
			{
				Vuhod << "\t" << Vx_index << "\t" << Vx_index;
			}
		}
		if ( i == ( KolWagov - 1 ) ) Vuhod << std::endl;
	}
	Vuhod << "Vkv =";
	for ( unsigned long i = 0; i < ( KolWagov * KolWagov); i++ )
	{
		for( long double Vkv_index = VkvNa4; Vkv_index < ( VkvNa4 + KolWagov * Wag); Vkv_index += Wag )
		{
			for ( unsigned long j = 0; j < ( KolWagov ); j++)
			{
				Vuhod	<< "\t" << Vkv_index << "\t" << Vkv_index;

			}
		}
		if ( i == ( KolWagov * KolWagov  - 1 ) ) Vuhod << std::endl;
	}
	Vuhod << "Vkl =";
	for ( unsigned long i = 0; i < ( KolWagov * KolWagov * KolWagov ); i++ )
	{
		for( long double Vkl_index = VklNa4; Vkl_index < ( VklNa4 + KolWagov * Wag); Vkl_index += Wag )
		{
			Vuhod << "\t" << Vkl_index << "\t" << Vkl_index;
		}
		if ( i == ( KolWagov * KolWagov * KolWagov - 1 ) ) Vuhod << std::endl;
	}
	Vuhod << std::endl << std::endl;
	Vuhod << "r";
	unsigned long KolZuklovVsego = KolWagov * KolWagov * KolWagov * KolWagov;
	for( unsigned long i = 0; i < KolZuklovVsego; i++)
	{
		Vuhod << "\t" << "Obu4nuu" << "\t" << " Miu";
		if ( i == ( KolZuklovVsego - 1 ) ) Vuhod << std::endl;
	}
	bool PervayaZapisBula = false;
	long double  V_0_min;
	long double  V_0_max;
	long double  V_1_min;
	long double  V_1_max;
		for( r = 4 ; r <= 96 ; r = r + 4 )
	{
		Vuhod << r;
		for( long double Vgen_index = VgenNa4; Vgen_index < ( VgenNa4 + KolWagov * Wag); Vgen_index += Wag )
		{
			for( long double Vx_index = VxNa4; Vx_index < ( VxNa4 + KolWagov * Wag); Vx_index += Wag )
			{
				for( long double Vkv_index = VkvNa4; Vkv_index < ( VkvNa4 + KolWagov * Wag); Vkv_index
+= Wag )
				{
					for( long double Vkl_index = VklNa4; Vkl_index < ( VklNa4 + KolWagov * Wag);
Vkl_index += Wag )
					{
						long double t1_0, t2_0, t3_0, t4_0, t5_0, t6_0;
						long double t1_1, t2_1, t3_1, t4_1, t5_1, t6_1;
						l = Razmer_ / r;
						for ( ProzedyraPidsulennyaBezpeku = 0; ProzedyraPidsulennyaBezpeku < 2;
ProzedyraPidsulennyaBezpeku++ )
						{
							if( ProzedyraPidsulennyaBezpeku == 0 )
							{
								t1_0 = l * r * r / Vgen_index;
							}
							else
```

```
                                                {
                                                        t1_1 = l * r / Vgen_index;
                                                }
                                                if( ProzedyraPidsulennyaBezpeku == 0 )
                                                {
                                                        t2_0 = l * (2 * r * r - r ) / Vx_index;
                                                }
                                                else
                                                {
                                                        t2_1 = l * r / Vx_index;
                                                }
                                                if( ProzedyraPidsulennyaBezpeku == 0 )
                                                {
                                                        t3_0 = l * r / Vkv_index;
                                                }
                                                else
                                                {
                                                        t3_1 = l * r / Vkv_index;
                                                }
                                                if( ProzedyraPidsulennyaBezpeku == 0 )
                                                {
                                                        t4_0 = l * r * r / Vkl_index;
                                                }
                                                else
                                                {
                                                        t4_1 = 96 / Vkl_index;
                                                }
                                                if( ProzedyraPidsulennyaBezpeku == 0 )
                                                {
                                                        t5_0 = l * (4 * r * r * r - 4 * r * r ) / Vx_index;
                                                }
                                                else
                                                {
                                                        t5_1 = l * r / Vgen_index;
                                                }
                                                if( ProzedyraPidsulennyaBezpeku == 0 )
                                                {
                                                        t6_0 = l * (2 * r * r - r ) / Vx_index;
                                                }
                                                else
                                                {
                                                        t6_1 = l * r / Vx_index;
                                                }
                                        }
                                        long double  V_0 = Razmer_ / ( t1_0 + t2_0 + t3_0 + t4_0 + t5_0 + t6_0 );
                                        long double  V_1 = Razmer_ / ( t1_1 + t2_1 + t3_1 + t4_1 + t5_1 + t6_1 );
                                        Vuhod <<"\t" << V_0 << "\t" << V_1;
                                        if ( PervayaZapisBula )
                                        {
                                                if ( V_0_min > V_0 ) V_0_min = V_0;
                                                if ( V_0_max < V_0 ) V_0_max = V_0;
                                                if ( V_1_min > V_1 ) V_1_min = V_1;
                                                if ( V_1_max < V_1 ) V_1_max = V_1;
                                        }
                                        else
                                        {
                                                V_0_min = V_0;
                                                V_0_max         = V_0;
                                                V_1_min = V_1;
                                                V_1_max         = V_1;
                                                PervayaZapisBula = true;
                                        }
                                }
                        }
                }
        }
        Vuhod << std::endl;
    }
    Vuhod << std::endl << std::endl;
    Vuhod << "V_0_min = \t" << V_0_min << std::endl;
    Vuhod << "V_0_max = \t" << V_0_max << std::endl;
    Vuhod << "V_1_min = \t" << V_1_min << std::endl;
    Vuhod << "V_1_max = \t" << V_1_max << std::endl;
}
```