НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра математичних методів захисту інформації

«До захисту допущено»
В.о. завідувача кафедри
_____ Михайло САВЧУК
«__» _____ 2021 р.

Дипломна робота

на здобуття ступеня бакалавра

зі спеціальності: 113 Прикладна математика
на тему: **«Моделі та методи побудови безпечних децентралізованих систем з небезпечних елементів»**

Виконав: студент 4 курсу, групи ФІ-73
Чіхладзе Вахтанг Зурабович

Керівник: с.н.с, д.т.н, професор Кудін А.М. _____

Консультант: _ _____

Рецензент: к.т.н., доцент Проскуровський Р.В. _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ — 2021

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра математичних методів захисту інформації

Рівень вищої освіти — перший (бакалаврський)
Спеціальність (освітня програма) — 113 Прикладна математика,
ОПП «Математичні методи криптографічного захисту інформації»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Михайло САВЧУК
«__» _____ 2021 р.

ЗАВДАННЯ
на дипломну роботу

Студент: Чіхладзе Вахтанг Зурабович

1. Тема роботи: «*Моделі та методи побудови безпечних децентралізованих систем з небезпечних елементів*»,

керівник: с.н.с, д.т.н, професор Кудін А.М.,

затверджені наказом по університету №__ від «__» _____ 2021р.

2. Термін подання студентом роботи: 4 червня 2021р.

3. Вихідні дані до роботи: *модель Шеннона побудови надійної системи з ненадійних елементів, стандарт з безпеки реалізації криптографічних модулів FIPS PUB 140-2, принципи побудови системи децентралізованих додатків Ethereum*

4. Зміст роботи:

1) *Проведення огляду існуючих робіт в теорії надійності систем, аналогічних за постановкою задачі побудови безпечної за реалізацією децентралізованої криптографічної системи з небезпечних криптографічних елементів-модулів;*

2) *Побудова математичної моделі безпечної за реалізацією децентралізованої криптографічної системи;*

3) *Реалізація децентралізованої криптографічної системи у вигляді смарт контракту системи Ethereum;*

5. Перелік ілюстративного матеріалу: *презентація доповіді, 8 рисунків.*

6. Дата видачі завдання: 10 вересня 2020 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання	Примітка
1	Узгодження теми роботи із науковим керівником	01-15 вересня 2020 р.	Виконано
2	Огляд існуючих джерел за тематикою дослідження	Вересень-жовтень 2020 р.	Виконано
3	Вибір основи для моделі	Жовтень-листопад 2020 р.	Виконано
4	Побудова моделі	Грудень-Березень 2020-2021 р.	Виконано
5	Реалізація моделі	Лютий-Квітень 2021 р.	Виконано
6	Проходження преддипломної практики	Квітень - Травень 2021 р.	Виконано
7	Написання дипломної роботи	Квітень - Червень 2021 р.	Виконано

Студент _____ Чіхладзе В.З.

Керівник _____ Кудін А.М.

РЕФЕРАТ

Кваліфікаційна робота містить: 83 сторінки, 8 рисунків, 11 джерел.

У даній роботі проведено дослідження безпеки реалізації децентралізованих систем на основі моделі ненадійних реле Шеннона. В якості показників безпеки функціонування використовуються показники, встановлені стандартом безпеки криптографічних модулів FIPS PUB 140-2. Метою дослідження є розробка децентралізованої криптографічної системи із обраними показниками безпеки реалізації з множини криптографічних модулів, кожен з яких може не задовільняти певним показникам щодо безпеки реалізації криптографічних модулів. Об'єктом дослідження є криптографічна система, побудована за децентралізованим принципом. Предметом дослідження є безпека реалізації децентралізованої криптографічної системи, побудованої з небезпечним за реалізацією криптографічних елементів

Результатами данної роботи є математична модель децентралізованого криптографічного модуля, опис функціоналу криптографічного модуля та протокол взаємодії між користувачами в децентралізованій системі, що забезпечується розробленним криптографічним модулем у вигляді смарт контракту.

КРИПТОГРАФІЧНИЙ МОДУЛЬ, БЛОКЧЕЙН, СМАРТ КОНТРАКТ

ABSTRACT

In this work was researched security implementation of decentralized systems based on the model of unreliable Shannon relays. Security indicators were instantiated by security standard of cryptographic modules FIPS PUB 140-2. The aim is to develop a decentralized cryptographic system with selected implementation security indicators from a set of cryptographic modules, each of which may not meet certain implementation security indicators. The object of study is a cryptographic system, which was built on a decentralized principle. The subject of research is the security implementation of this system, which build from unsecure by realization cryptographic elements.

The results of this work is a mathematical model of a decentralized cryptographic module, a description of the functionality of the cryptographic module and a protocol of interaction between users in a decentralized system provided by the developed cryptographic module as smart contract.

CRYPTOGRAPHIC MODULE, BLOCKCHAIN , SMART CONTRACT

ЗМІСТ

ВСТУП

Актуальність дослідження. На сьогодні більшість світу бурхливо розроблює, вдосконалює та використовує децентралізовані технології та комунікації. Системи побудовані за децентралізованим принципом є більш стійкими до відмови, оскільки відмова одного елементу не призводить до зупинки всього додатку. Те саме стосується й компроментації безпеки одного елементу в децентралізованій системі. Але відсутні критерії оцінки безпеки реалізації криптосистем побудовані за децентралізованим принципом подібних до критеріїв, що застосовуються до криптосистем, побудованих за централізованим принципом FIPS PUB 140-2.

Метою дослідження є розробка децентралізованої криптографічної системи із обраними показниками безпеки реалізації з множини криптографічних модулів, кожен з яких може не задовільняти певним показникам щодо безпеки реалізації криптографічних модулів Для досягнення мети необхідно розв'язати **задачу дослідження**. Для розв'язання задачі необхідно вирішити такі завдання:

1) Вибрати математичний апарат як основу для моделювання децентралізованої криптографічного модуля;

2) Провести огляд існуючих робіт в теорії надійності систем, аналогічних за постановкою задачі побудови безпечної за реалізацією децентралізованої криптографічної системи з небезпечних криптографічних елементів-модулів

3) Побудувати математичну модель безпечної за реалізацією децентралізованої криптографічної системи

4) Реалізувати децентралізовану криптографічну систему у вигляді смарт контракту системи Ethereum

Об'єктом дослідження є криптографічна система, побудована за децентралізованим принципом

Предметом дослідження є безпека реалізації децентралізованої криптографічної системи, побудованої з небезпечним за реалізацією криптографічних елементів.

При розв'язанні поставлених завдань використовувались такі *методи дослідження*: дискретної математики, теорії ймовірності, методи математичного і комп'ютерного моделювання.

Наукова новизна отриманих результатів полягає в використанні блокчейнів для підвищення стійкості криптографічних модулів до атак на реалізацію.

Практичне значення результатів полягає в розробці протоколу взаємодії криптографічних модулів на основі смарт-контрактів, які підвищують їх стійкість криптографічних модулів до атак на реалізацію.

1 ОГЛЯД ДЖЕРЕЛ

У даному розділі мова буде йти про

1) модель ненадійних реле та схеми з ненадійних реле

2) стандарт,що описиує криптографічних модулі FIPS PUB

3) блокчейн Ethereum

На основі данних оглядаих джерел буде синтезовано рішення задачі побудови моделі децентралізованого криптографічного модуля та його корректного впровадження в децентралізований додаток.

1.1 Модель Шеннона ненадійного реле

В данній роботі Клод Шеннон запропонував математичну модель, яка описує ненадійне електромагнітне реле та схему з ненадійних електромагнітних реле. Ця модель оперується в термінах теорії ймовірності. Також описан метод побудови надійної схеми із ненадійних реле з якою завгодно надійностю.

1.1.1 Ненадійне реле та схема з ненадійних реле

По-перше, треба визначити що з себе представляє єлектромагнітне реле. Реле - це фізичний пристрій, який складається з котушки, на яку намотана мідний дріт, ключ замикання та з'єднуючий контакт. Коли на котушку з обмотаним мідним дротом подається струм, то виникає електромагнітне поле, що притягує ключ, чим замикає з'єднуючий контакт.

Реле, як і будь який інший технічний пристрій, не завжди працює ідеально. Тобто в роботі пристроїв можуть відбуватись помилки. Помилки в реле можуть виникати через погану пайку, знос деталей і т.д.

Запропонована модель враховує лише два типи помилок і не враховує причини їх виненкнення:

1) контакт, який повинен бути замкнутим, залишається розімкненим;

2) контакт, який повинен бути розімкненим, залишається замкненим;

Ці типи похибок являють собою випадковий характер, що можна описати терминами теорії ймовірності. Часто буде оперуватись термін ймовірність. Ймовірність - це певна міра нашої впевненності у якісь події. Ймовірність приймає на вхід подію, та повертає континуальні значення від 0 до 1 включно.

Якщо реле не збудженно, то вірогідність того що контакт замкнут дорівнює:

$$Pr(\text{якщо реле не збуджено,то контакт замкнений}) = a$$

Тобто це вирогідність помилки.

Якщо реле не збудженно, то вірогідність того що контакт розімкнут дорівнює:

$$Pr(\text{якщо реле не збуджено,то контакт розімкнений}) = 1 - a$$

Тобто це вирогідність правильної роботи.

Якщо реле збудженно,то вірогідність того що контакт замкнут дорівнює:

$$Pr(\text{якщо реле збуджено,то контакт замкнений}) = c$$

Тобто це вирогідність правильної роботи.

Якщо реле збудженно,то вірогідність того що контакт розімкнут дорівнює:

$$Pr(\text{якщо реле збуджено,то контакт розімкнений}) = 1 - c$$

Тобто це вирогідність помилки.

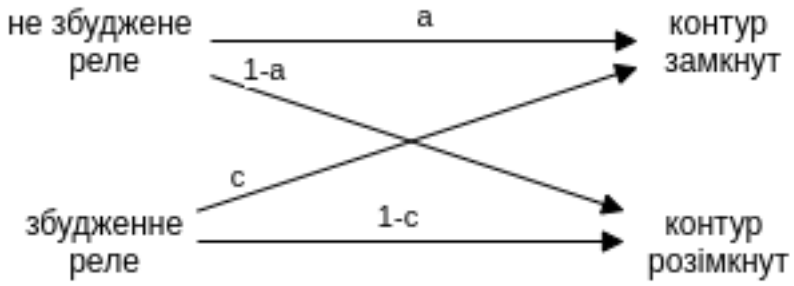


Рисунок 1.1 – Схематичне зображення вирогідностей

Підсумуємо,що a та $1-c$ це вирогідність помилки у роботі реле. Відповіно $1-a$ та c ймовірність правильної роботи. Реле, що має такі ймовірності називається ненадійним.

Якщо $a < c$, то такий контакт називається замикаючим. В моделі Шеннона розглядаються замикаючі контакти.

Нехай є контакт, який має вирогідність замикання p. Тоді вирогідність замикання схеми із m незалежних контактів буде описуватись функцією $h(p)$. Значення цієї можно обчислити за такою формулою:

$$h(p) = \sum_{n=0}^{m} A_n \cdot p^n \cdot (1-p)^{m-n} \tag{1.1}$$

де A_n - кількість способів якими можна обрати множину із n контактів так, що схема буде замкнена

Приклад 1.1. Нехай є чотири реле з вирогідністю замикання p. Якщо їх зібрати у схему зображену на данному рисунку, то вигляд ймовірність замикання данної схеми буде.

$$h(p) = 2 \cdot p^2 - p^4 \tag{1.2}$$

Доведемо це використовуючи формулу 1.1. Спочатку обрахуємо коефіцієнти A_n. Щоб обрахувати дані коефіцієнти,нам потрібно спочтатку сформувати множини тих реле, що утворюють замкнутий контур. Замкнутий контур означає, що струм пройде від початку до

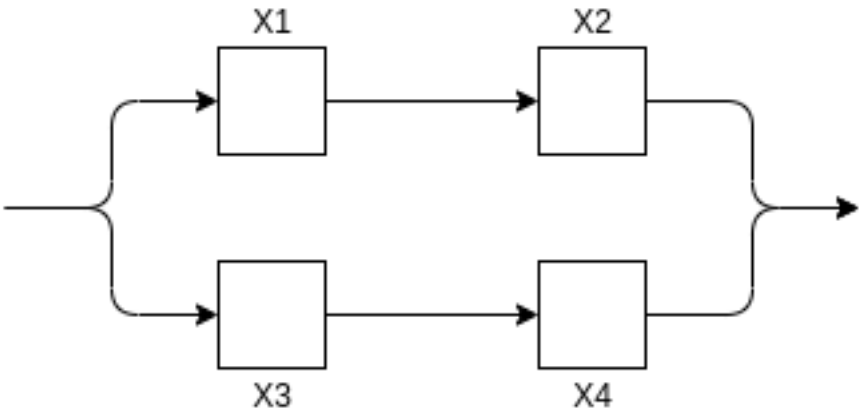


Рисунок 1.2 – Схематичне зображення чотирьох реле

кінця схеми. Множина множин цих реле мають вигляд:

$$\{\{X1,X2\}\{X3,X4\},\{X1,X2,X3\},\{X1,X2,X4\},\{X1,X3,X4\},$$

$$\{X2,X3,X4\},\{X1,X2,X3,X4\}\}$$

З цього легко можна обрахувати коефіцієнти A_n.

$$A_0 = 0, A_1 = 0, A_2 = 2, A_3 = 4, A_4 = 1$$

Якщо підставити дані значення у формулу 1.1 і звести подібні, то отримаємо 1.2.Перевіримо це.

$h(p) = \sum_{n=0}^{4} A_n \cdot p^n \cdot (1-p)^{4-n} = 0 \cdot p^0 \cdot (1-p)^4 + 0 \cdot p^1 \cdot (1-p)^3 + 2 \cdot p^2 \cdot (1-p)^2 +$
$+ 4 \cdot p^3 \cdot (1-p) + 1 \cdot p^4 = 2 \cdot p^2 \cdot (1 - 2 \cdot p + p^2) + 4 \cdot p^3 - 4 \cdot p^4 + p^4 =$
$= 2 \cdot p^2 - 4 \cdot p^3 + 2 \cdot p^4 + 4 \cdot p^3 - 4 \cdot p^4 + p^4 = 2 \cdot p^2 - p^4$

Було з'єднано чотири незалежних реле у одну схему і було побудовано вирогідність замикання схеми 1.2.Також ми порівняли функції $y = p$ та $y = h(p)$ По задумці вирогідність правильної роботи зросла. Але наскільки зросла дана ймовірність? Давайте обрахуємо.

Нехай ми взяли і подивились як працює одне реле і вирухали помилки, що дорівнюють $a = 1 - c = 0{,}1$. Тепер візьмемо нашу функцію 1.2 і підставимо вирогідність помилки $a = 0{,}1$.

$$h(a) = 2 \cdot 0{,}1^2 - 0{,}1^4 = 0{,}0199 \approx 0{,}02$$

, що суттєво менше ніж $a = 0{,}1$.

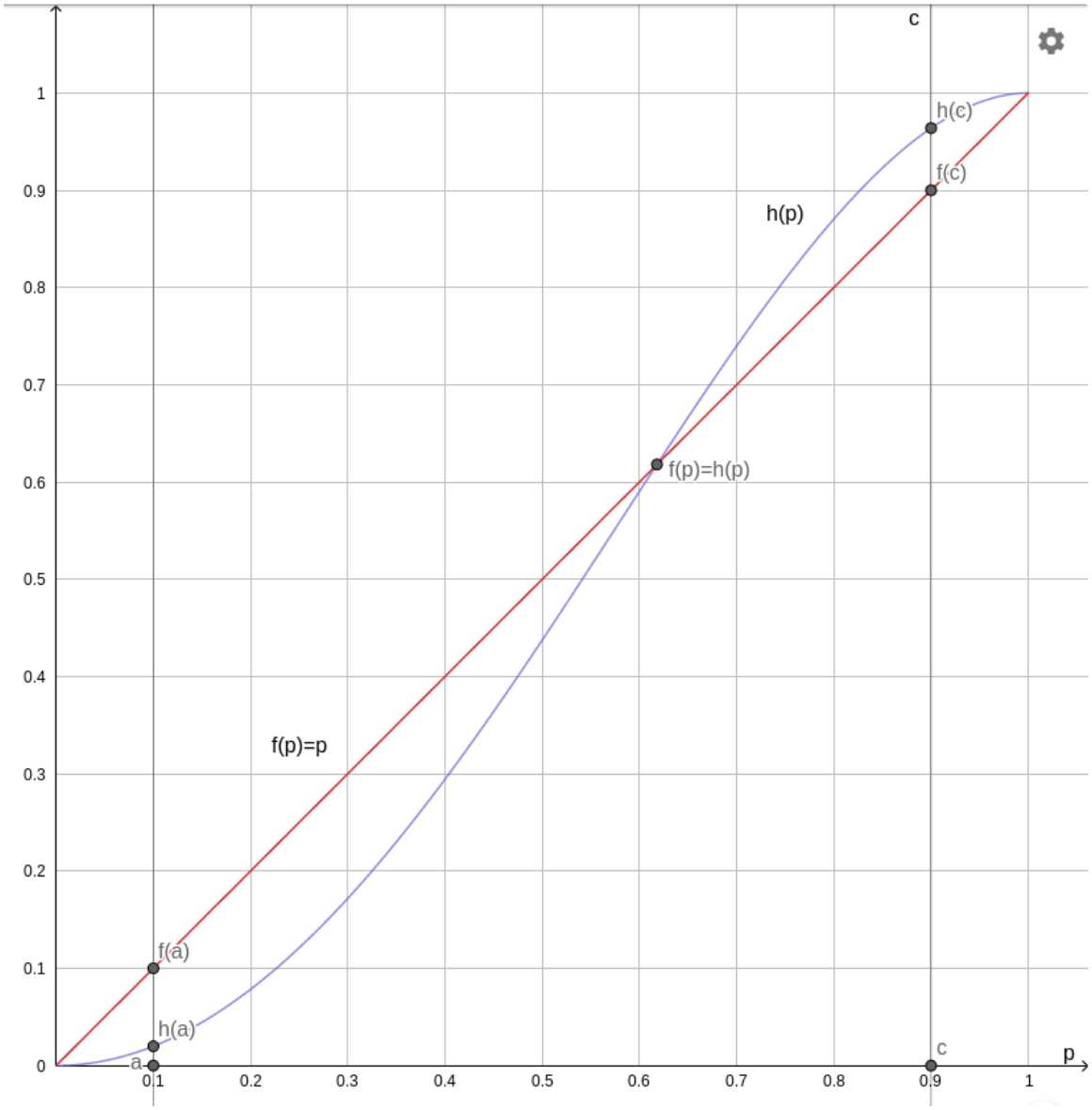


Рисунок 1.3 – Наглядне порівняння $f(p) = p$ та $h(p) = 2 \cdot p^2 - p^4$

Тобто, при об'єднанні ненанадійних реле ми зменшили вирогідність помилки коли контур замкнений.

$$h(c) = 2 \cdot 0{,}9^2 - 0{,}9^4 = 0{,}9639$$

, що суттєво більше ніж $c = 0{,}9$.

Тобто ми збільшили ймовірність правильної роботи коли контур розімкнут. Одночасно зі збільшенням ймовірності правильної роботи ми змешили ймовірність помилки у схемі з m ненадійних реле.

$$h(1-c) = 1 - h(c) = 1 - 0{,}9639 = 0{,}0361$$

, що значно менше нід $1-c = 0{,}1$. Можна побачити,що ймовірність помилки у схемі коли контур розімкнений суттєво зменшилась.

Для наглядності зображено графіки та відповідні значення на рисунку 1.3.

Отже в прикладі 1.1 було показано як схема з незалежних однакових реле може бути кращою ніж одне реле та показано наскільки краще.

1.1.2 Метод побудови надійної схеми

Тепер опишемо метод за яким досягається надійні схеми. Завдяки цьому методу досягається побудова схеми будь-якої надійності. Робиться це в три етапи: початковий, середній і останній.

Початковий етап

Знаходиться таке реле, що ймовірності в точках a та c знаходиться по різні сторони точки $p = 0{,}5$, тобто:

$$a < 0{,}5 < c \tag{1.3}$$

Далі з таких однакових і незалежних реле збирається схема, яка повинна задовільняти виразам:

$$h(a) < 0{,}5 \tag{1.4}$$

$$h(c) > 0{,}5 \tag{1.5}$$

Наприклад такою схемою може слугувати схема з прикладу 1.1. Або може обратись однакові та незалежні реле,що задовільняють 1.3.Потім методом спроб і помилок зібрати якусь схему, що задовільняє виразам 1.4 та 1.5. Для підтвердження цього була сформована теорема:

Теорема 1.1. *Нехай* $\forall a,c : 0 < a < c < 1, b = \frac{a+c}{2}, d = \max(b,1-b)$, $\epsilon = \frac{c-a}{4}$. *Тоді* $\exists$ *схема* N *, що має не більш ніж* $[\frac{\log(\epsilon)}{\log(d)}]$ *контактів і* $h_N(a) \leqslant 0{,}5 - \epsilon$ *та* $h_N(a) \geqslant 0{,}5 - \epsilon$

Середній етап

На цьому этапі вже є замикаюча схема N, що задовільняє 1.4, 1.5, 1.3 та має вирогідність роботи $h_N(p)$. Тепер ми заміняємо кожне реле у схемі N схемою з реле N. Нова схема з математичної точки зору виглядає так

$$h(h(p)) = h^{(2)}(p)$$

,де $h^{(2)}(p)$ ітераційне застосування результату функції $r = h(p)$ як вхід до функції $h(p)$,тобто $h(r)$. Середнім етапом ми повинні отримати, що вирогідність замикання схеми в ϵ-околі точки $p = 0{,}5$ дорівнює

$$h^{(2)}(\frac{1}{2} - \epsilon) \leqslant \frac{1}{4}$$

$$h^{(2)}(\frac{1}{2} + \epsilon) \geqslant \frac{3}{4}$$

Останній етап

На останньому етапі ми застосовуємо s ітерацій для функції $h(p)$ для того щоб отримати сприятну помилки δ_1 та δ_2.

$$h^{(s)}(c) \geqslant \delta_1$$

$$h^{(s)}(a) \leqslant \delta_2$$

Після останнього етапу ми отримаємо надійну схему, що має значно меншу помилку δ порівнянно з a та $1 - c$. Щоб усвідомити, що на данних етапах робилось: можно подивитись на рисунок 1.4.

Також можно оцінити скільки мінімально потрібно реле для визначення надійної схеми. Про це говорить наступна теорема.

Теорема 1.2. *Нехай* $0 < a < c < 1$ *та* N *- схема з функцією ймовірності* $h(p)$ *така що* $\forall\delta_1,\delta_2 > 0$ *що*

$$h(a) \leqslant \delta_1$$

$$h(c) \geqslant 1 - \delta_2$$

Тоді мінімальна кількість контактів n*, що необхідна для заданої надійності* δ_1 *і* δ_2 *дорівнює* $n = \frac{\log(\delta_1)}{\log(a)} \cdot \frac{\log(\delta_2)}{\log(1-c)}$

Приклад 1.2. Якщо ми хочемо, щоб схема робила не більше однієї помилки за 10^{10}, тобто $\delta_1 = \delta_2 = 10^{-10}$, операцій при помилках $a = 1 - c = 10^{-1}$, то $n = \frac{\log(10^{-10})}{\log(10^{-1})} \cdot \frac{\log(10^{-10})}{\log(10^{-1})} = 100$ реле.

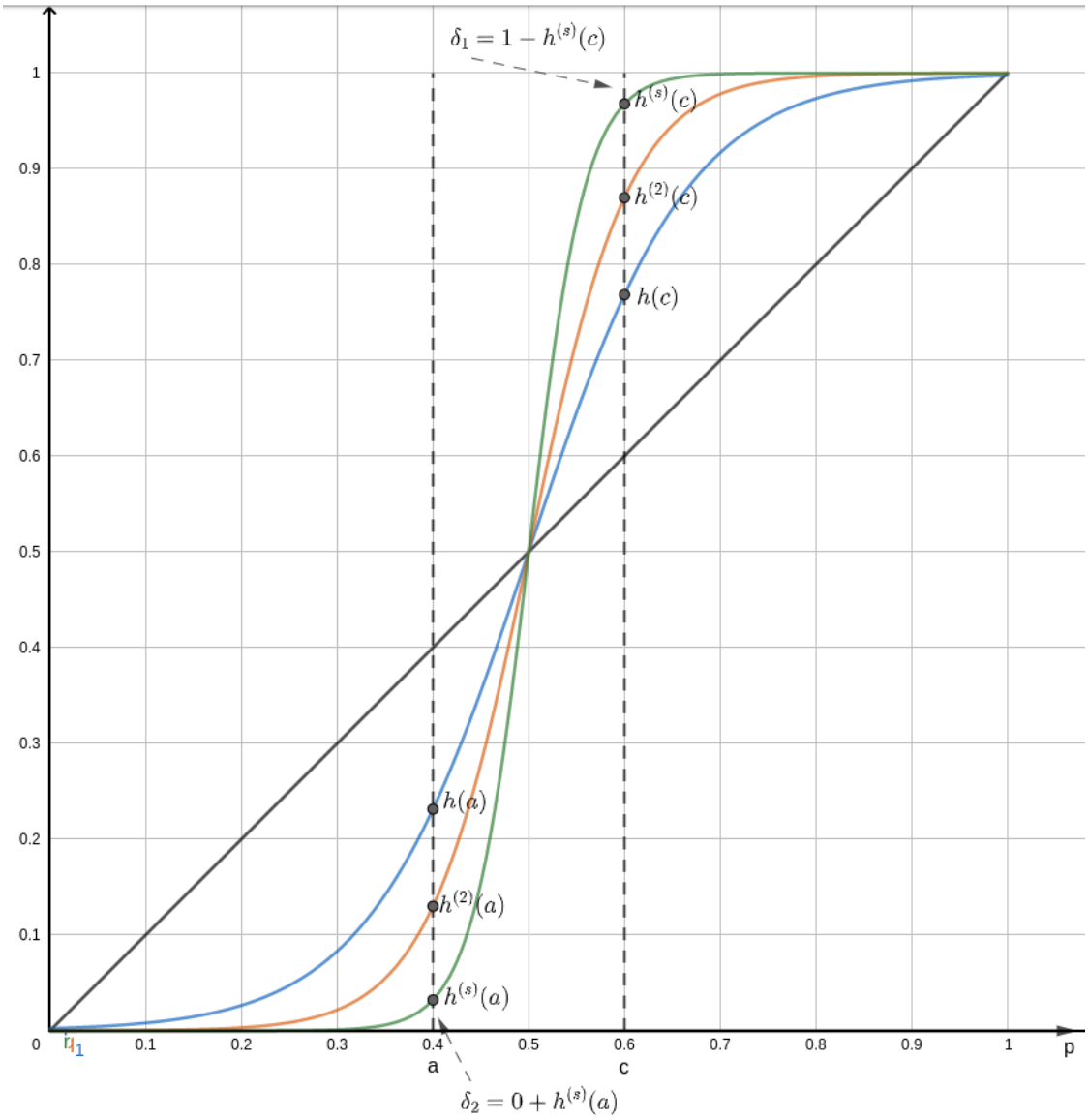


Рисунок 1.4 – Зображення ітерацій функції $h(p)$

Отже, в данному огляді була розглянута робота Шеннона про надійні схеми з ненадійних реле. Була представлена модель ненадійного реле, що описувалась ймовірністю несправної роботи. Слідом описали ймовірності роботи схеми із ненадійних реле. Також було розглянуто метод побудови надійної схеми.

1.2 FIPS PUB 140-2

Публікація Федеральних Стандартів Обробки Інформації (FIPS PUB) - це серія стандартів, які регламентують рекомендації по обробці та забезпечення безпеки інформації в інформаційних системах. Данні статті розробляються та затверджуються Національним Інститутом Стандартів та Технологій (NIST) спільно з урядом та оперативною робочою групою.

FIPS PUB 140-2 - стандарт, який описує вимоги до криптографічний модулів, що впровадженні в системи безпеки телекомунікаційних та інформаційних систем. За цим стандартом можна визначити який рівень захищеності має впровадженний криптографічний модуль, а також на основі цього стандарту можна поліпшувати та будувати криптографічні модулі.

По-перше, треба визначити що є криптографічним модулем. Криптографічним модулем за данним стандартом називається набір апаратного, програмного забезпечення або мікропрограми, що реалізує затверджену безпеку функцій, які включають криптографічні алгоритми та генерацію ключів, і міститься в криптографічному периметрі. Під криптографічним периметром розуміється явно визначені фізичні межі де криптографічний модуль виконує свій функціонал. Наприклад на локальній обчислювальному пристрої, на сервері, в програмному забезпеченні і т.д.

1.2.1 Чотири рівня безпеки

Взагалі, безпека об'єкта - це стан захищеності об'єкту, що зберігає свою стійкість при внутрішніх та зовнішніх впливах. Криптографічний модуль забезпечує інформаційну безпеку. Тобто криптографічний модуль повинен забезпечувати безпеку обробки та зберігання інформації, а також забезпечує цілісність,конфіденційність та доступність. Данний стандарт

визначає 4 рівня безпеки криптографічних модулів.

Перший рівень безпеки.(Security Level 1)

Перший рівень безпеки є найнижчим рівнем безпеки, який потребує самого базового захисту. Базовий захист криптографічного модуля полягає в існуванні шифруючого пристрою в будь-якому операційному середовищі, доступ до якого може бути який завгодно, хоч фізичний, хоч віддалений.

Криптографічні перетворення, що є частиною криптографічного модуля, повинні бути специфікованими за стандартами FIPS або NIST.

Другий рівень безпеки.(Security Level 2)

Другий рівень включає у себе всі вимоги першого рівня безпеки.

Додається вимога на систему захисту від фізичного втручання у роботу криптографічного модуля. Фізичний захист другого рівня передбачає впровадження або наявність системи, що доказує факт втручання у систему. Наприклад, поставити пломбу на корпус ком'пютера,де працює криптографічний модуль. Або повісити замок на корпус,за яким можно з високою вирогідністю сказати що втручання відбулось.

Також додається система ролів, що вимагає від користувачи автентифікації. На цьому рівні безпеки система автентифікації є базовою, тобто не робиться ніяких комплексних дій для автетифікації користувача. Наприклад, для автентифікації просять ввести логін і пароль.

Третій рівень безпеки.(Security Level 3)

Третій рівень безпеки включає в себе всі вимоги другого рівня безпеки.

До фізичної безпеки додається вимога до захисту критичних параметрів безпеки (Critical Security Parameters, CSP). Якщо фізичне втручання відбулось,то видаляється вся інформація пов'язана з приватними ключами, PIN-кодами, сессіями і т.д.

Розширюється механізм автентифікації користувачів, яке є більш комплексним ніж це вимагається на другому рівні безпеки. Наприклад, користувача можуть попросити ввести додатковий пароль або код.

Вводиться вимога на окреме використання логічних та фізичних портів на ввід та вивід критичних параметрів безпеки (CSP).

Четвертий рівень безпеки.(Security Level 4)

Четвертий рівень безпеки є найвищий рівнем безпеки, що є в стандарті FIPS PUB 140-2. Також четвертий рівень включає в себе всі вимоги з третього рівня безпеки.

Додається вимога до повного захисту фізичного середовища де працює крипторафічний модуль. На попередніх рівнях безпеки вимагалась система захисту від втручання і відповідне реагування на втручання. Тепер додається вимога на реагування флуктуалій у середовищі де знаходиться обчислювальний пристрій. Тобто, при відхиленні від норми напруги, температури, або інших фізичних величин повинні відбуватись ряд заходів. Якщо можливо реагувати на аномалії у фізичному середовищі, то відповідна система повинна задіяти, щоб привести обчислювальний пристрій до нормального стану.Якщо ж ні, то будь які операції повинні зупинитись та критичні параметри безпеки повинні бути видалені. Це важливо, оскільки атакуючий може маніпулювати станом системи та зможе обійти захист, що не бажано в криптографічних модулях.

1.2.2 Характеристики криптографічного модуля

В данній секції будуть розглядатись характеристики криптографічного модуля.

Специфікація криптографічного модуля

Специфікація криптографічного модуля - це документ, в якому наявний опис всіх компонентів криптографічного модуля. Цими компонентами можуть слугувати:

1) Алгоритми
2) Структури данних.
3) Режими операцій
4) Обчислювальне середовище
5) Програмне забезпечення
6) Політика безпеки,тощо.

Порти та інтерфейси криптографічного модуля

Щоб працювати з криптографічним модулем, треба давати команди з параметрами. Команди с параметрами подаються порти криптографічного модуля. Вихідні дані також подаються криптографічним модулем на порти. Розрізняють логічні та фізичні порти. Логічні порти - це об'єкти операційної системи на які записуються та считуються дані. Фізичні порти - це об'єкти фізичного середовища обчислювальної системи, де працює криптографічний модуль, на які записуються та считуються дані. Для простоти взаємодії з криптографічним модулем використовують API, що являється інтерфейсом криптографічного модуля. Звичайно, порти та інтерфейси повинні бути задокументовані в специфікації.

Ролі, Сервіси, Автентифікація

Криптографічний модуль можна уявляти як перелік сервісів. Наприклад, такими сервісами можуть слугувати процедура шифрування, розшифрування, генерація ключів, і т.д. Щоб цими сервісами користуватись, потрібно пройти процедуру автентифікації. Після автентифікації користувачу призначають роль, яка задана у системі. Наприклад, користувач, адміністратор, модератор і т.д.

Модель кінцевих станів

Криптографічний модуль повинен бути описан математичною моделлю скінченного автомату. Робиться це для аналізу і виявлення загроз, щоб якмога з більшою вирогідністю запобігти компрометації системи. Якщо не впровадити дану модель, то розробка,підтримка і аналіз нового функціоналу буде відбуватись важко.

Обчислювальне середовище

Під обчислювальним середовищем криптографічного модуля розуміється операційна система або архітектура обчислювального пристрою.

Наприклад, криптографічний модуль може бути програмою з GUI під ОС Windows/Linux. Або може бути вбудованою системою у плати.

Управління криптографічними ключами

Управління криптографічними ключами являє собою такі сервіси:

1) Генерація випадкових чисел для побудови ключів;

2) Розподілення ключів;

3) Ввід та вивід ключів

4) Зберігання ключів

5) Видалення ключів.

1.2.3 Тестування криптографічного модуля

Для перевірки коректної роботи криптографічного модуля потрібно проводити тестування функціоналу і компонентних модулів. Основні тести, що вимагається від криптографічного модуля:

1) Коректность роботи криптографічних алгоритмів.

Тобто тестується, що після шифруючих перетворень вхідного повідомлення, дешифруючи перетворення відновлюють початкове повідомлення.

2) Сумісність програмного забезпечення

Тестується сумісність розробленого криптографічного модуля з обчислювальним середовищем. Проводяться виміри наскільки сумісний криптографічний модуль у різних обчислювальних середовищах.

3) Стрес тести

В данному типі тестів робиться велике навантаження на криптографічний модуль та перевіряється чи працює криптографічний модуль коректно.

4) Тестування генераторів випадкових послідовностей

Дані тести є статистичними тестами, що говорять чи є данний генератор гарний у статистичній манері. Якщо реалізований поганий генератор, то можна передбачати значення, які буде видавати генератор в майбутньому і відповідно вгадувати секретні ключі користувачів.

Отже, в данній секції було розглянуто стандарт FIPS PUB 140-2, який регламентує вимоги з безпеки криптографічних модулів. Було розглянуто чотири рівні безпеки, які описані в стандарті. Також були описані основних характеристики криптографічного модуля, що описує стандарт.

1.3 Технологія Blockchain

В даній секції буде описано основи технології блокчейн.

Блокчейн - це сукупність блоків, що йдуть ланцюжком одним за одним в строгому порядку. В початку ланцюга знаходиться перший ініціалізуючий блок, що зветься genesis block, який містить інформацію про протоколи роботи блокчейну. Кожен блок містить інформацію про часову позначку та транзакції, а також геш всіх транзакцій у блоці в якості корня дерева Меркля.

Транцакція - це певна інформація, що додається у блок. Сам блокчейн забезпечує незмінність данної транзакцій у блоці. Щоб змінити одну або декілька транзакцій у якомусь блокці - треба переписати весь ланцюг блоків, що з практичної точки зору дуже складно. Нові транзакції додаються у новий блок, а цей блок додається до ланцюга блоків згідно правил консенсусу.

Блокчейн можна уявляти як розподілену базу даних, яка має свою структуру, або глобальний комп'ютер. Також блокчейн можна уявляти як множину вузлів, де кожен вузол має копію блокчейна. Тобто, будь-хто може під'єднатись до мережі вузлів, тим самим створити новий вузол. Коли створюється новий вузол, то цей вузол робить копію блокчейну.

Розділяють блокчейни на публічні і приватні. Приватні блокчейни повністю підконтрольні власникам. Такими власниками можуть бути компанія, організація або зацікавлений користувач. В публічних блокчейнах робота базується всіма користувачами системи, які дотримуються консенсусу.

1.3.1 Консенсуси блокчейна

Для роботи публічних блокчейнів вводиться консенсус, який підтримує всю роботу мережі блокчейну. Під роботою розуміється

створення нових блоків та додавання транзакцій у ці блоки.

Proof of Work

Консенсус Proof of Work (PoW) базується на вирішенні складної математичної задачі.

Постанова задачі: нехай $f(x)$ - одностороння функція. За заданим цільовим значенням y знайти таке значення x, що $f(x) < y$. Значенням x є конкатенація всіх транзакцій у блоці та деяке значення nonce, яке заохочені користувачі перебирають.

Як тільки заохочений користувач, якого називають майнером, вирішить дану задачу, тобто сформує та підпише блок з транзакціями, то він розсилає усім майнерам в мережі підписаний своїм секретним ключем блок і додає цей блок до ланцюга блокчейну.

Процедура майнінгу дуже енергозатратна. Також хто володіє більшістю обчислювальних ресурсів, той може маніпулювати порядком слідування блоків в блокчейні, що не дуже добре для децентралізації.

Proof of Stake

Консенсус Proof of Stake базується на ставці певної суми криптовалюти, а натомість цьому стейкер наділяється правом валідувати блоки з транзакціями та в якості нагороди забирати комісію з транзакцій. При цьому підході ймовірність створити новий блок пропорційна ставці, яку зробив стейкер. В даному консенсусі обмеженим ресурсом є кількість криптовалюти, а не на обчислювальні ресурси та електроенергія.

Розроблено багато варіацій та модифікацій консенсусу Proof of Stake. Одним з таких є протокол Casper, що буде впроваджений в Ethereum2.0. Особливістю даного протоколу є покарання нечесних стейкерів, які можуть залишитись без своєї ставки, що значно підвищує

надійність роботи блокчейну і зменшить ймовірність шкідливої поведінки стейкерів.

1.3.2 Блокчейн Ethereum

В 2013 році засновник видання Bitcoin Magazine та розробник Ethereum Віталік Бутерін запропонував ідею створення модифікації блокчейну Bitcoin, що транзакції можуть мати різного роду інформацію, яку можна програмувати. Згодом в 2015 році платформа Ethereum запустилась і почала свою роботу. В перспективі Віталік заявив, що Ethereum перейде з консенсуса Proof of Work до Proof of Stake та створить Ethereum2.0, який буде швидше, маштабовіше і дешевше. Першого грудня 2020 року платформа Ethereum тільки почала перехід до Ethereum2.0 і займе це деякий час.

Аккаунти в Ethereum

В Ethereum аккаунтом називається сутність, що описується 20 байтним значенням та має характеристики такі як

1) nonce - номер транзакції, яку робить аккаунт
2) баланс - кількість криптовалюти, що має аккаунт
3) код - характерне тільки смарт-контрактам
4) сховище(storage) - інформація, яку зберігає аккаунт

Транзакції в Ethereum

Транзакцією називається певне підписане повідомлення, що йде від одного аккаунту до іншого. Щоб транзакція була оброблена в мережі блокчейна, потрібно вказати скільки криптовалюти аккаунт готовий заплатити майнеру(валідатору), щоб він обрав саме його транзакцію. Цю

кількість криптовалюти називають gas. Отже, транзакція містить такі елементи

1) повідомлення аккаунта-відправника

2) підпис аккаунта-відправника

3) кількість ETH

4) необов'язкове поле данних

5) максимальна кількість газу

6) ціна газу

Якщо ми хочемо зчитати якісь дані з блокчейна, то транзакцію створювати не потрібно, треба лише звернутись за адресою, де потрібні нам дані зберігаються.

1.3.3 Смарт контракти Ethereum

Смарт-контрактом в Ethereum є аккаунт, що має код, який виконується на Ethereum Virtual Machine (EVM). Сама назва "контракт"юридичної сили не несе, тільки представляє ідеологію Code is Law. Відповідно, смарт-контракт реалізує якусь домовленість між аккаунтами, що підтверджується кодом. Програмуються мовою програмування Solidity, а скомпільований код програми транслюється в байт код EVM.

Solidity є об'єктно-орієнтованою мовою. Це означає, що смарт-контракт має:

1) стан;

2) поведінку;

Фактично, станом є набір змінних, які описують смарт-контракт. А поведікою є функції, які змінюють стан смарт-контракту. Стани можуть зберігатись у сховищі(storage)

Приклад 1.3. Смарт-контракт, що реалізує парі між двома учасниками відносно завтрашньої погоди. Перший учасник сказав, що

завтра буде нижче нуля градусів по цельсію, другий сказав що завтра буде вище нуля градусів по цельсію. На початку стан обох учасників не визначений. Кожний з учасників визиває функцію і передає свій вибір. Станом s є вибір учасника, тобто $s \in$ {вище,нижче}, а поведінкую тут слугує функція вибору, що міняє стан з невизначеності на "вище або "нижче".

EVM логічно можна вважати одним обчислювальним пристроєм, а фізично EVM розподілений між всіми вузлами мережі Ethereum.

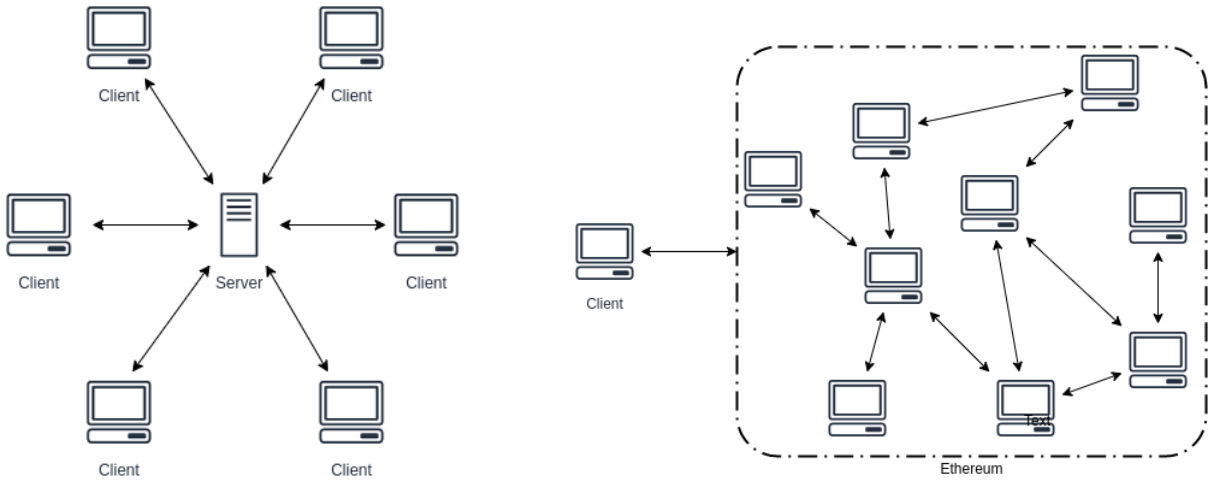
Коли смарт-контракт розроблений, то його додають до блокчейну Ethereum і як результат отримують чек транзакції з адресою на цей контракт. Цей доданий код вже не можна змінити, видалити або модифікувати, оскільки все що додано до блокчейну - те складно видалити або змінити. За цією адресою можно викликати функції смарт-контракту та змінювати стан смарт-контракту.

1.3.4 Децентралізовані додатки

Ethereum здобув популярність через те, що можна програмувати смарт-контракти та розроблювати децентралізовані додатки. Децентралізований додатоком є деякий інтерфейс, що взаємодіє з блокчейном.

В централізованих додатках є сервер, який оброблює запити клієнтів і відповідає на них. В більшості випадків ніхто,окрім розробників серверу, не знає який код виконується на стороні сервера та які внутрішні дані сервер використовує.

Децентралізований додаток відрізняється тим, що через інтерфейс або напряму запит клієнта йде в блокчейн мережу, де знаходиться код контракту. Код, вхідні та вихідні дані записані у всьому ланцьогу блокчейна і подивитись їх може будь хто. Слід також відмітити, що код виконується на вузлі майнера(валідатора).



(а) **(б)**

Рисунок 1.5 – Додатки: (а) централізований, (б) децентралізований

Отже,в даній секції було розглянуто що таке блокчейн.Як блокчейн працює і які бувають консенсуси.Було розглянуто платформу Ethereum.

Висновки до розділу 1

Отже, було розглянуто модель схем ненадійних реле Шеннона, стандарт FIPS PUB 140-2 та технологію блокчейн.

В данному розділі було виконано перше завдання досліждення з вибору основи для моделі. Модель ненадійних реле Шеннона майже підходить під досліджуєму задачу, але дана модель була розроблена для схем з реле з двома станами, а криптографічний модуль також можна вважати схемою з реле, но станів може бути набагато більше. Тож модель ненадійних реле Шеннона було обрано як ідею, яку потрібно переробити під дану задачу.

Ще було виконано друге завдання досліждення з огляду безпеки реалізації криптографічних модулів. А саме був розглянутий стандарт FIPS PUB 140-2, що регламентує компоненти криптографічного модуля та рівень захисту, та модель надійних реле Шеннона.

Також в данному розділі відбудось знайомлення з основами

децентралізованих додатків. Було описано платформу Ethereum як публічний блокчейн, в якому є велика кількість користувачів, що слідують консенсусу і в якій можлива розробка децентралізованих додатків.

2 МАТЕМАТИЧНА МОДЕЛЬ ДЕЦЕНТРАЛІЗОВАНОГО КРИПТОГРАФІЧНОГО МОДУЛЯ

Децентралізований криптографічний модуль є децентралізованим додатком у блокчейні. Тож відмова або не правильна робота одного фізичного вузла не презведе до аварійної зупинки всього додатка. Додаток продовже безпечно працювати без вузла, що відмовив. Щоб кількісно оцінити ймовірність безпечної роботи, буде математично описано модель децентралізованого криптографічного модуля.

Також, за стандартом FIPS PUB 140-2 необхідною є формальна модель криптографічного модуля в вигляді скінченного автомату. Це потрібно для аналізу системи та побудові статистик, для подальшого покращення та дослідження.

Тож, в данному розділі буде побудовано формальну модель децентралізованого криптографічного модуля заснований на ідеї ненадіних реле Шеннона.

2.1 Переформулювання моделі ненадійного реле з двома станами

З точки зору математики, криптографічні модулі в децентралізованих системах та схеми ненанійних реле схожі. Ненадійні реле мають ймовірність не правильної роботи та вузли в децентралізованих системах також помиляються при роботі, тобто безпека функціонування іноді виходить з ладу. Схеми з реле та вузли децентралізованої системи можуть описуватись як мережева топологія. Але є одна відмінність: у реле лише два стани, а в елемента децентралізованого криптографічного модуля $n \geqslant 2$ станів. Щоб узагальнити модель ненадійного реле, де реле має $n \geqslant 2$ станів, потрібно

переформулювати існуючу модель ненадійного реле з двома станами в потрібну нам еквівалентну модель скінченного ймовірнісного автомату.

2.1.1 Перехід до ймовірнісної динамічної системи

Нагадаємо, що в моделі ненадійного реле ймовірності помилок позначались як

$$Pr(\text{якщо реле не збуджено, то контакт замкнений}) = a \tag{2.1}$$

$$Pr(\text{якщо реле збуджено, то контакт розімкнений}) = 1 - c \tag{2.2}$$

Відповідно ймовірність правильної роботи:

$$Pr(\text{якщо реле збуджено, то контакт замкнений}) = 1 - a \tag{2.3}$$

$$Pr(\text{якщо реле не збуджено, то контакт розімкнений}) = c \tag{2.4}$$

Спробуємо перенести події, від яких береться ймовірності 2.1, 2.2, 2.3, 2.4, у новий простір. Нехай ненадійне реле з двома станами має опис в фазовому просторі

$$E = \{0,1,2,3\} \tag{2.5}$$

де елементи фазового простору 2.5 можуть бути інтерпретовані і переформульовані відносно подій так:

$0 \leftrightarrow$ контакт зімкнут правильно

$1 \leftrightarrow$ контакт розімкнут правильно

$2 \leftrightarrow$ контакт зімкнут помилково

$3 \leftrightarrow$ контакт розімкнут помилково

Тобто фазовий простір є множиною станів, що описує реле. Фазовий простір можна розбити на два класи: правильні і помилкові стани. На кожний правильний стан є відповідний помилковий стан.

$$E_+ = \{0,1\} \text{ - простір правильних станів} \tag{2.6}$$

$$E_- = \{2,3\} \text{ - простір помилкових станів} \tag{2.7}$$

Зауваження. Простори E_+ та E_- є підпросторами простора E.

Приклад 2.1. До правильного стану 0 є відповідний йому помилковий стан 2, що симантично описує те що стан реле повинен знаходитись у стані 0(тобто реле повинно бути зімкнуте), а знаходиться у якомусь іншому стані (розімкнено).

Тепер спробуємо переписати дані ймовірності 2.3, 2.4, 2.1, 2.2 в термінах фазового простору.

$$Pr(x_k = 0) = 1 - a \tag{2.8}$$

$$Pr(x_k = 1) = c \tag{2.9}$$

$$Pr(x_k = 2) = a \tag{2.10}$$

$$Pr(x_k = 3) = 1 - c \tag{2.11}$$

Вже задані ймовірності 2.8,2.9,2.10,2.11 не можуть бути розподілом, оскільки сума всіх цих ймовірностей не дорівнює 1. Щоб вирішити дану проблему зробимо деяке відображення в інший простір, де дані ймовірності відобразились у ймовірності у новому простірі та утворювали би розподіл. Випливає задача в побудові відображення, яке перетворить ймовірності 2.8,2.9,2.10,2.11 на розподіл в просторі E.

Треба відмітити, що в моделі Шеннона ненадійного реле простір складається з незалежних подій замикання, розмикання реле та відповідних подій помилкової роботи. На ці події введена ймовірнісна міра, що виражається виразами від 2.1 до 2.4. Виразимо все це математично.

Нехай є два незалежних ймовірнісні простори

$$(\Omega_0,\mathbb{B}_0,\mathbb{P}_0) \tag{2.12}$$

$$(\Omega_1,\mathbb{B}_1,\mathbb{P}_1) \tag{2.13}$$

що описують ймовірнісну характеристику замкнутих та розмикаючих реле відповідно. Також існує простір Ω_∞, що є надмножиною множин Ω_0 та Ω_1. Явно ці множини можна описати так:

$\Omega_\infty = \{0,1,2,3,\ldots\}$

$\Omega_0 = \{0,2\}$ - простір подій, де описуються стани замикання та помилкового замикання реле віповідно до подій, що описують ймовірності 2.1 та 2.3

$\mathbb{B}_0$ - σ-алгебра породжена множиною Ω_0 або просто множина всіх підмножин Ω_0, тобто $\mathbb{B}_0 = \{\varnothing,\{0\},\{2\},\{0,2\}\}$

$\mathbb{P}_0$ - ймовірнісна міра. Є відображенням $\mathbb{P}_0 : \mathbb{B}_0 \to [0,1]$

$$\mathbb{P}_0(\{0\}) = 1 - a \tag{2.14}$$

$$\mathbb{P}_0(\{2\}) = a \tag{2.15}$$

$$\mathbb{P}_0(\Omega_0) = 1$$

$$\mathbb{P}_0(\varnothing_0) = 0$$

$\Omega_1 = \{1,3\}$ - простір подій, де описуються стани розмикання та помилкового розмикання реле віповідно до подій, що описують ймовірності 2.2 та 2.4

$\mathbb{B}_1$ - σ-алгебра породжена множиною Ω_1 або просто множина всіх підмножин Ω_1, тобто $\mathbb{B}_1 = \{\varnothing,\{1\},\{3\},\{1,3\}\}$

$\mathbb{P}_1$ - ймовірнісна міра. Є відображенням $\mathbb{P}_1 : \mathbb{B}_1 \to [0,1]$

$$\mathbb{P}_1(\{1\}) = c \tag{2.16}$$

$$\mathbb{P}_1(\{3\}) = 1 - c \tag{2.17}$$

$$\mathbb{P}_1(\Omega_1) = 1$$

$$\mathbb{P}_1(\varnothing_1) = 0$$

Теорема 2.1. *Нехай існують σ-алгебри $\mathbb{B}_0$ та $\mathbb{B}_1$, що мають*

спільні прості елементи $A^{(1)},\ldots,A^{(n)} \in \mathbb{B}_0,\mathbb{B}_1$*:* $A^{(i)} \cap A^{(j)} = \varnothing$*,* $i \neq j$*,* $i,j \leqslant n$*, тобто* $\mathbb{B}_0 \cap \mathbb{B}_1 = \{A^{(1)},\ldots,A^{(n)}\}$*,* $n \in \mathbb{N}$*.*

Тоді можна:

1)побудувати два відображення, що однозначно зв'язує спільні прості елементи з одної множини до другої і з другої множини до першої

2)унікальний простий елемент відобразити у порожню множину

3)відобразити не просту подію, що складається з об'єднання простих подій, у іншу множину відносно пунктів 1) та 2)

Доведення.

1)Побудуємо такі відображення. Нехай $A_0^{(i)} \in \mathbb{B}_0$, $A_1^{(i)} \in \mathbb{B}_1$, $i \leqslant n$, $n \in \mathbb{N}$. Задамо $Q_{0\to1} : \mathbb{B}_0 \to \mathbb{B}_1$, що в явну записується $Q_{0\to1}(A_0^{(i)}) = A_1^{(i)}$. Тепер для відображення з $\mathbb{B}_1$ в $\mathbb{B}_0$ задамо $Q_{1\to0} : \mathbb{B}_1 \to \mathbb{B}_0$, яке явно виглядає $Q_{1\to0}(A_1^{(i)}) = A_0^{(i)}$.

2)За визначенням σ-алгебри порожня множина завжди належить σ-алгебрі. З цього слідує, що можна завжди відобразити унікальний простий елемент $U_0 \in \mathbb{B}_0\backslash\mathbb{B}_1$ σ-алгебри $\mathbb{B}_0$ у порожню множину $\varnothing_1$, що є елементом σ-алгебри $\mathbb{B}_1$, тобто $Q_{0\to1}(U_0) = \varnothing_1$. Те саме стосується для унікального простого елементу $U_1 \in \mathbb{B}_1\backslash\mathbb{B}_0$ σ-алгебри $\mathbb{B}_1$, тобто $Q_{1\to0}(U_1) = \varnothing_0$.

3)Розглянемо не просту подію $D_0 \in \mathbb{B}_0$, яку можна розкласти на об'єднання простих подій $D_0 = D_0^{(1)} \cup \ldots \cup D_0^{(k)}$, $D_0^{(i)} \in \mathbb{B}_0,D_0^{(i)} \cap D_0^{(j)} = \varnothing$, $i,j \leqslant k$. Тоді $Q_{0\to1}(D_0)$ єдиним чином розкладеться та відобразиться єдиним чином відносно правил відображення побудованих у 1) і 2). $Q_{0\to1}(D_0) = Q_{0\to1}(D^{(1)} \cup \ldots \cup D^{(k)}) = Q_{0\to1}(D^{(1)}) \cup \ldots \cup Q_{0\to1}(D^{(k)})$. Те ж саме виконується для $Q_{1\to0}(\cdot)$.

□

Лема 2.1. *Нехай існують ймовірнісні простори* $(\Omega_0,\mathbb{B}_0,\mathbb{P}_0)$ *та* $(\Omega_1,\mathbb{B}_1,\mathbb{P}_1)$*, які мають одну спільну подію* $\varnothing \in \mathbb{B}_0,\mathbb{B}_1$*. Тоді для будь яких нетрівіальних подій* $A_0 \in \mathbb{B}_0$*,* $A_1 \in \mathbb{B}_1$*,* $A_0 \neq \varnothing_0$*,* $A_1 \neq \varnothing_1$ *виконується* $\mathbb{P}_0(A_1) = 0$ *та* $\mathbb{P}_1(A_0) = 0$

Доведення. В просторах $(\Omega_0,\mathbb{B}_0,\mathbb{P}_0)$ та $(\Omega_1,\mathbb{B}_1,\mathbb{P}_1)$ задані

ймовірнісні міри $\mathbb{P}_0$ та $\mathbb{P}_1$ відносно відповідних σ-алгебр $\mathbb{B}_0$ та $\mathbb{B}_1$. Брати подію від взагалі іншої σ-алгебри є недоречним, тоді за теоремою 2.1 ми можемо побудувати відображення, що відображає елементи одного протору до іншого. Оскільки σ-алгебри мають один спільний елемент $\varnothing$, то вірне наступне. Нехай $\forall i,j = 0,1 : i \neq j,$, тоді

$$Z_{i \to j} : \mathbb{B}_i \to \{\varnothing_j\}$$

Тоді $\forall A_i \in B_i$ виконується що

$$\mathbb{P}_j(Z_{i \to j}(A_i)) = \mathbb{P}_j(\varnothing_j) = 0$$

□

Сформуємо новий ймовірнісний простір

$$(\Omega,\mathbb{B},\mathbb{P}) = (E,\mathbb{B},\mathbb{P}) \tag{2.18}$$

де
$\Omega = \Omega_0 \cup \Omega_1 = \{0,1,2,3\} = E \subset \Omega_\infty$
$\mathbb{B}$ - σ-алгебра на множині $\Omega = E$
$\mathbb{P}$ - ймовірнісна міра на множині $\Omega = E$

σ-алгебру $\mathbb{B}$ в явному вигляді можна записати так:

$$\mathbb{B} = \{\varnothing,\{0\},\{1\},\{2\},\{3\},\{0,1\},\{0,2\},\{0,3\},\{1,2\},\{1,3\},\{2,3\},$$
$$\{0,1,2\},\{0,1,3\},\{0,2,3\},\{1,2,3\},\{0,1,2,3\}\} \tag{2.19}$$

Зауваження. Ймовірнісна міра $\mathbb{P} : \mathbb{B} \to [0,1]$ залежить від вхідного параметру $A \in \mathbb{B}$ з σ-алгебри $\mathbb{B}$ та повертає значення від 0 до 1. Також повинно виконуватись що $\mathbb{P}(\Omega) = 1$ та $\mathbb{P}$ - σ-аддитивна міра, тобто $\forall A_i,A_j \in \mathbb{B}$ такі що $A_i \cap A_j = \varnothing$, $i,j \in \mathbb{N}, i \neq j$ виконується

$$\mathbb{P}(\bigcup_{i \in \mathbb{N}} A_i) = \sum_{i \in \mathbb{N}} \mathbb{P}(A_i) \tag{2.20}$$

Фактично, ймовірнісна міра $\mathbb{P}$ є розподілом на множині $\Omega = E$.

Таким чином ми описали ймовірнісні простори замикання та розмикання ненадійних реле та об'єднавши ці ймовірнісні простори ми утворили бажаний простір E, з якого зробимо ймовірнісний простір.

Потрібно співставити події з ймовірнісних просторів 2.12 та 2.13 на простір 2.18. Нехай $A^{(1)}_{\mathbb{B}_0},\ldots,A^{(n)}_{\mathbb{B}_0} \in \mathbb{B}_0$, $A^{(1)}_{\mathbb{B}_1},\ldots,A^{(n)}_{\mathbb{B}_1} \in \mathbb{B}_1$ такі що $\mathbb{B} \cap \mathbb{B}_0 = \{A^{(1)}_{\mathbb{B}_0},\ldots,A^{(n)}_{\mathbb{B}_0}\}$ та $\mathbb{B} \cap \mathbb{B}_1 = \{A^{(1)}_{\mathbb{B}_1},\ldots,A^{(n)}_{\mathbb{B}_1}\}$, тоді за теоремою 2.1 можна побудувати функції G_0 та G_1 які відображають події з σ-алгебр $\mathbb{B}_0$ та $\mathbb{B}_1$ в σ-алгебру $\mathbb{B}$

$$G_{\mathbb{B}_0\to\mathbb{B}} : \mathbb{B}_0 \to \mathbb{B} \tag{2.21}$$

$$G_{\mathbb{B}_1\to\mathbb{B}} : \mathbb{B}_1 \to \mathbb{B} \tag{2.22}$$

та в явному записі мають вигляд

$$G_{\mathbb{B}_0\to\mathbb{B}}(A^{(i)}_{\mathbb{B}_0}) = A^{(i)}_{\mathbb{B}} \tag{2.23}$$

$$G_{\mathbb{B}_1\to\mathbb{B}}(A^{(i)}_{\mathbb{B}_1}) = A^{(i)}_{\mathbb{B}} \tag{2.24}$$

Зауваження. Оскільки $\mathbb{B}_0$ та $\mathbb{B}_1$ є підмножинами множини $\mathbb{B}$, то побудовані відображення $G_{\mathbb{B}_0\to\mathbb{B}}$ та $G_{\mathbb{B}_1\to\mathbb{B}}$ є ін'єкцією.

Також побудуємо відображення $G_{\mathbb{B}\to\mathbb{B}_0}$ та $G_{\mathbb{B}\to\mathbb{B}_1}$, які співставлять події $\mathbb{B}$ подіям з множин $\mathbb{B}_0$ та $\mathbb{B}_1$, тобто

$$G_{\mathbb{B}\to\mathbb{B}_0} : \mathbb{B} \to \mathbb{B}_0 \tag{2.25}$$

$$G_{\mathbb{B}\to\mathbb{B}_1} : \mathbb{B} \to \mathbb{B}_1 \tag{2.26}$$

Нехай $S^{(1)}_{\mathbb{B}},\ldots,S^{(n)}_{\mathbb{B}} \in \mathbb{B}$ та $S^{(1)}_{\mathbb{B}},\ldots,S^{(m)}_{\mathbb{B}} \in \mathbb{B}$, $n,m \in \mathbb{N}$ такі що $\mathbb{B} \cap \mathbb{B}_0 = \{S^{(1)}_{\mathbb{B}},\ldots,S^{(n)}_{\mathbb{B}}\}$, $\mathbb{B} \cap \mathbb{B}_1 = \{S^{(1)}_{\mathbb{B}},\ldots,S^{(m)}_{\mathbb{B}}\}$ та $S^{(i)}_{\mathbb{B}} \cap S^{(j)}_{\mathbb{B}} = \varnothing, i \neq j$, $i,j < n$ і $i,j < m$, тобто $S^{(i)}_{\mathbb{B}}$ прості події.

Тоді спільні прості події відображаються так

$$G_{\mathbb{B}\to\mathbb{B}_0}(S^{(i)}_{\mathbb{B}}) = S^{(i)}_{\mathbb{B}_0} \tag{2.27}$$

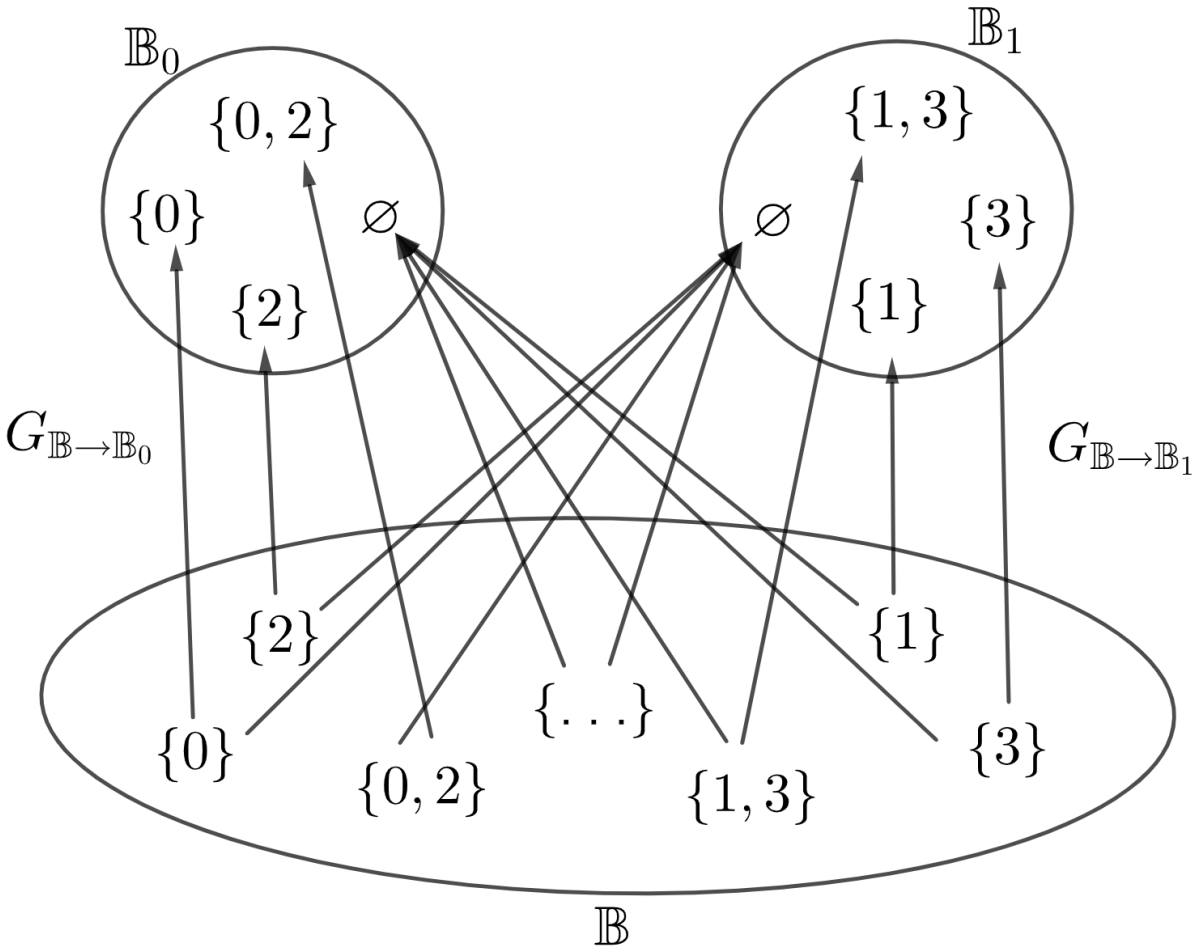


Рисунок 2.1 – Функції $G_{\mathbb{B}\to\mathbb{B}_0}$ та $G_{\mathbb{B}\to\mathbb{B}_1}$

$$G_{\mathbb{B}\to\mathbb{B}_1}(S_{\mathbb{B}}^{(i)}) = S_{\mathbb{B}_1}^{(i)} \tag{2.28}$$

Унікальні прості події S_0,$S_1 \in \mathbb{B}$, тобто такі що $S_0 \notin \mathbb{B} \cap \mathbb{B}_0$ та $S_1 \notin \mathbb{B} \cap \mathbb{B}_1$, відображаються у порожню множину згідно теоремою 2.1.

$$G_{\mathbb{B}\to\mathbb{B}_0}(S_0) = \varnothing_{\mathbb{B}_0} \tag{2.29}$$

$$G_{\mathbb{B}\to\mathbb{B}_1}(S_1) = \varnothing_{\mathbb{B}_1} \tag{2.30}$$

А не прості події $S \in \mathbb{B}$, тобто такі що $S = S_{\mathbb{B}}^{(1)} \cup \ldots \cup S_{\mathbb{B}}^{(k)}$, $S_{\mathbb{B}}^{(i)}$,$i \leqslant k$ розкладаються на прості події відображаються так

$$G_{\mathbb{B}\to\mathbb{B}_0}(S) = G_{\mathbb{B}\to\mathbb{B}_0}(S_{\mathbb{B}}^{(1)}) \cup \ldots \cup G_{\mathbb{B}\to\mathbb{B}_0}(S_{\mathbb{B}}^{(k)}) = S_{\mathbb{B}_0}^{(1)} \cup \ldots \cup S_{\mathbb{B}_0}^{(k)} \tag{2.31}$$

$$G_{\mathbb{B}\to\mathbb{B}_1}(S) = G_{\mathbb{B}\to\mathbb{B}_1}(S_{\mathbb{B}}^{(1)}) \cup \ldots \cup G_{\mathbb{B}\to\mathbb{B}_1}(S_{\mathbb{B}}^{(k)}) = S_{\mathbb{B}_1}^{(1)} \cup \ldots \cup S_{\mathbb{B}_1}^{(k)} \quad (2.32)$$

Приклад 2.2. Нехай $\{0\} \in \mathbb{B}_0$, тоді $G_{\mathbb{B}_0\to\mathbb{B}}(\{0\}_{\mathbb{B}_0}) = \{0\}_{\mathbb{B}}$.

Приклад 2.3. Нехай $\{3\} \in \mathbb{B}$, тоді $G_{\mathbb{B}\to\mathbb{B}_1}(\{3\}_{\mathbb{B}}) = \{3\}_{\mathbb{B}_1}$.

Приклад 2.4. Нехай $\{3\} \in \mathbb{B}$, тоді $G_{\mathbb{B}\to\mathbb{B}_0}(\{3\}_{\mathbb{B}}) = \varnothing_{\mathbb{B}_0}$.

Тепер задамо ймовірнісну міру на просторі $(E,\mathbb{B},\mathbb{P})$ 2.18. Оскільки простір 2.18 був побудований з двох ймовірнісних просторів 2.12 та 2.13, то бажано використати ймовірнісні міри цих просторів для побудови нової ймовірнісної міри. Нагадаємо, що ймовірнісна міра $\mathbb{P}$ - це відображення з σ-алгебри B у відрізок [0,1] та ймовірнісна міра від всього простору дорівнює одиниці, тобто $\mathbb{P}(E) = 1$. Введемо нове означення для синтезу переліченого.

Означення 2.1. Нехай $A \in \mathbb{B}$ подія з σ-алгебри $\mathbb{B}$, $\vec{\mathbb{P}}_{0,n-1}(A) = (\mathbb{P}_0(G_{\mathbb{B}\to\mathbb{B}_0}(A)),\mathbb{P}_1(G_{\mathbb{B}\to\mathbb{B}_1}(A)),\ldots,\mathbb{P}_{n-1}(G_{\mathbb{B}\to\mathbb{B}_{n-1}}(A))), n \in \mathbb{N}$ - вектор ймовірнісних мір, $\vec{w} = (w_0,w_1,\ldots,w_{n-1})$ - вектор ваг. Ймовірнісною мірою на просторі E введемо як скалярне множення вектора ваг $\vec{w}$ на вектор ймовірнісних мір $\vec{\mathbb{P}}_{0,n-1}(A)$

$$\mathbb{P}(A) = (\vec{w},\vec{\mathbb{P}}_{0,n-1}(A)) \quad (2.33)$$

для якої виконується умова

$$\mathbb{P}(E) = (\vec{w},\vec{\mathbb{P}}_{0,n-1}(E)) = 1 \quad (2.34)$$

Зауваження. Вектор ваг $\vec{w}$ задається довільним чином згідно наших припущень та уявлень, але потрібно щоб виконувалась умова 2.34.

Для випадку $n = 2$ маємо вектор ваг $\vec{w} = (w_0,w_1)$ та вектор ймовірнісних мір $\vec{\mathbb{P}}(A) = (\mathbb{P}_0(A),\mathbb{P}_1(A))$. Тепер введемо припущення що ваги однакові. Фактично дане припущення означає, що одна похибка не є більш вагомою ніж друга помилка, тобто дані помилки рівноцінні. Математично даний факт виражається як

$$w_0 = w_1 \quad (2.35)$$

Тоді ймовірнісна міра від елемента $A \in \mathbb{B}$ σ-алгебри $\mathbb{B}$ буде мати вигляд

$$\mathbb{P}(A) = w_0 \cdot \mathbb{P}_0(G_{\mathbb{B}\to\mathbb{B}_0}(A)) + w_0 \cdot \mathbb{P}_1(G_{\mathbb{B}\to\mathbb{B}_1}(A)) \tag{2.36}$$

Зауваження. Можна також вводити припущення, що помилка замикання більш вагома ніж помилка розмикання. Математичний вираз цього факту

$$w_0 = \alpha \cdot w_1, \qquad \alpha = const > 1 \tag{2.37}$$

І тоді ймовірнісна міра буде виглядати так

$$\mathbb{P}(A) = \alpha \cdot w_1 \cdot \mathbb{P}_0(G_{\mathbb{B}\to\mathbb{B}_0}(A)) + w_1 \cdot \mathbb{P}_1(G_{\mathbb{B}\to\mathbb{B}_1}(A)) \tag{2.38}$$

В обох випадках 2.35,2.37 повинна виконуватись умова 2.34, з якої можна знайти елементи вектора $\vec{w}$. Надалі будемо працювати з випадком 2.35, оскільки він наближен до реальної ситуації, де одна помилка не краще і не гірше іншої. Обрахуємо $\vec{w}$ у випадку 2.35.

$$\begin{aligned}
\mathbb{P}(E) &= w_0 \cdot \mathbb{P}_0(G_{\mathbb{B}\to\mathbb{B}_0}(E)) + w_0 \cdot \mathbb{P}_1(G_{\mathbb{B}\to\mathbb{B}_1}(E)) = \\
&= w_0 \cdot \mathbb{P}_0(G_{\mathbb{B}\to\mathbb{B}_0}(\{0,1,2,3\})) + w_0 \cdot \mathbb{P}_1(G_{\mathbb{B}\to\mathbb{B}_1}(\{0,1,2,3\})) = \\
&= w_0 \cdot \mathbb{P}_0(G_{\mathbb{B}\to\mathbb{B}_0}(\{0\} \cup \{1\} \cup \{2\} \cup \{3\})) + \\
&+ w_1 \cdot \mathbb{P}_1(G_{\mathbb{B}\to\mathbb{B}_1}(\{0\} \cup \{1\} \cup \{2\} \cup \{3\})) = \\
&= w_0 \cdot \mathbb{P}_0(G_{\mathbb{B}\to\mathbb{B}_0}(\{0\}) \cup G_{\mathbb{B}\to\mathbb{B}_0}(\{1\}) \cup G_{\mathbb{B}\to\mathbb{B}_0}(\{2\}) \cup G_{\mathbb{B}\to\mathbb{B}_0}(\{3\})) + \\
&+ w_1 \cdot \mathbb{P}_1(G_{\mathbb{B}\to\mathbb{B}_1}(\{0\}) \cup G_{\mathbb{B}\to\mathbb{B}_1}(\{1\}) \cup G_{\mathbb{B}\to\mathbb{B}_1}(\{2\}) \cup G_{\mathbb{B}\to\mathbb{B}_1}(\{3\})) = \\
&= w_0 \cdot \mathbb{P}_0(\{0\} \cup \varnothing \cup \{2\} \cup \varnothing) + w_1 \cdot \mathbb{P}_0(\varnothing \cup \{1\} \cup \varnothing \cup \{3\}) = \\
&= w_0 \cdot \mathbb{P}_0(\{0\} \cup \{2\}) + w_1 \cdot \mathbb{P}_1(\{1\} \cup \{3\}) = |\sigma\text{-аддитивність 2.20}| = \\
&= w_0 \cdot (\mathbb{P}_0(\{0\}) + \mathbb{P}_0(\{2\})) + w_1 \cdot (\mathbb{P}_1(\{1\}) + \mathbb{P}_1(\{3\})) = \\
&= w_0 \cdot ((1-a) + a) + w_1 \cdot (c + (1-c)) = w_0 + w_1 = 1 \\
&\Longrightarrow w_0 + w_1 = 1
\end{aligned}$$

Оскільки $w_0 = w_1 \Longrightarrow w_0 + w_0 = 2_0 = 1 \Longrightarrow w_0 = \dfrac{1}{2}$

Таким чином отримаємо вектор ваги $\vec{w} = (\frac{1}{2},\frac{1}{2})$, що підставляється у вираз 2.36

$$\mathbb{P}(A) = \frac{1}{2} \cdot \mathbb{P}_0(G_{\mathbb{B}\to\mathbb{B}_0}(A)) + \frac{1}{2} \cdot \mathbb{P}_1(G_{\mathbb{B}\to\mathbb{B}_1}(A)) \tag{2.39}$$

Криптографічний модуль є системою, що знаходиться в якомусь одному стані з фазового простору E та переходить в інший стан з простору E. Спробуємо побудувати правила переходу з одного стану до іншого, тим самим завдавши ймовірний розподіл переходу з одного до іншого стану.

Означення 2.2. Динамічною системою називається функція

$$f : E \times [0,1] \to E \tag{2.40}$$

що змінює стан системи за таким правилом

$$x_{k+1} = f(x_k, \vec{p}) \tag{2.41}$$

$$x_k \in E, \vec{p} = (p_{i\to 0}, p_{i\to 1}, \ldots, p_{i\to |E|-1}), p_{i\to j} \geqslant 0, \sum_{u=0}^{|E|-1} p_u = 1, i,j \in E$$

Фактично, функція f є рушійною силою, що змінює стан. Тобто функція приймає значення x_k з фазового простору E та розподіл $\vec{p}$ стану $x_k \in E$ в момент $k \in \mathbb{N}$ та повертає стан $x_{k+1} \in E$, в який перейшла система. Динамічну систему 2.41 для фазового простору 2.5 можна записати матрицею перехідних ймовірностей

$$P_{x_k\to x_{k+1}} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{pmatrix} p_{0\to 0} & p_{0\to 1} & p_{0\to 2} & p_{0\to 3} \\ p_{1\to 0} & p_{1\to 1} & p_{1\to 2} & p_{1\to 3} \\ p_{2\to 0} & p_{2\to 1} & p_{2\to 2} & p_{2\to 3} \\ p_{3\to 0} & p_{3\to 1} & p_{3\to 2} & p_{3\to 3} \end{pmatrix} \end{array} \tag{2.42}$$

де $p_{i\to j}$ -ймовірність переходу із стану x_k до стану x_{k+1}, i,j,x_k,$x_{k+1} \in E$, $k \in \mathbb{N}$. Цифри збоку позначають з якого стану, а зверху позначають в який стан. Дана матриця 2.42 повинна мати властивість стохастичності справа, тобто сума кожного рядка дорвінює одиниці та кожен елемент матриці більше нуля.

$$P_{x_k\to x_{k+1}}\text{- стохастична справа} \Leftrightarrow \forall p_{i\to j} > 0, \sum_{j\in E} p_{i\to j} = 1; i,j \in E \qquad (2.43)$$

Але виникає питання: "Які повинні бути ймовірності в 2.42?". Щоб відповісти на це питання запишемо вирогідність переходу з стану $e_1 \in E$ до стану $e_2 \in E$ як ймовірність того, що система перейде в стан e_2 при умові того, що в поточний момент часу система знаходиться в стані e_1.

$$Pr(x_{k+1} = e_2 | x_k = e_1) = \frac{Pr(x_{k+1} = e_2 \cap x_k = e_1)}{Pr(x_k = e_1)}$$

Виникає ще питання: "Чи незалежні події $x_{k+1} = e_2$ та $x_k = e_1$?". Розглянимо два випадки.

Випадок 1.Якщо події незалежні, то можна стверджувати що перехід з одного стану в другий нічим не кращий або не гірший ніж перехід у третій стан, будь він правильний або помилковий. Продовжимо викладку при умові, що події незалежні.

$$\frac{Pr(x_{k+1} = e_2 \cap x_k = e_1)}{Pr(x_k = e_1)} = \frac{Pr(x_{k+1} = e_2) \cdot Pr(x_k = e_1)}{P(x_k = e_1)} = Pr(x_{k+1} = e_2)$$

В результаті отримали, що ймовірність переходу зі стану e_1 до стану e_2 є тим самим, що вирогідність правильної роботи або помилки стану e_2 у просторі E, тобто можемо скласти таку матрицю перехідних ймовірностей:

$$P_{x_k \to x_{k+1}} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{pmatrix} \frac{1-a}{2} & \frac{c}{2} & \frac{a}{2} & \frac{1-c}{2} \\ \frac{1-a}{2} & \frac{c}{2} & \frac{a}{2} & \frac{1-c}{2} \\ \frac{1-a}{2} & \frac{c}{2} & \frac{a}{2} & \frac{1-c}{2} \\ \frac{1-a}{2} & \frac{c}{2} & \frac{a}{2} & \frac{1-c}{2} \end{pmatrix} \end{array} \tag{2.44}$$

Випадок 2.Якщо події залежні, то не вдасться все гарно звести як у випакду 1. Фактично, для залежних подій можна стверджувати, що наступний стан залежить від поточного. А якщо побудувати оцінки ймовірностей, що зададуть розподіл для кожного стану з фазового простору E, то в результаті отримаєм стохастичну справа матрицю. З цих фактів випливає, що можна отримати послідовність $\{x_k, k \in \mathbb{N}\}$ яка є ланцюгом Маркова.

Фактично, випадок 2 говорить про те, що в рядках матриці можемо робити перерозподіл як нам заманеться, що еквівалентно заданням початкового розподілу, але тільки з вимогою що матриця 2.42 залишилась стохастичною справа. Надалі для наглядності ми будемо працювати з матрицею при випадку 1, а якщо знадобиться інший розподіл елементів фазового простору то можна матрицю 2.44 перетворити на іншу стохастично справа матрицю.

Приклад 2.5. Для стану 0 в рядку, що позначено 0 , лежить ймовірнісний розподіл. Розподіл стану 0 можна інтерпретувати так:

$$p_{0 \to j} = \begin{cases} \frac{1-a}{2} & \text{перейти в стан } j = 0 \\ \frac{c}{2} & \text{перейти в стан } j = 1 \\ \frac{a}{2} & \text{перейти в стан } j = 2 \\ \frac{1-c}{2} & \text{перейти в стан } j = 3 \end{cases}$$

Також можна переконатись, що сума по всіх вирогідностях дорівнює 1.

$$\frac{1-a}{2}+\frac{c}{2}+\frac{a}{2}+\frac{1-c}{2}=\frac{1}{2}-\frac{a}{2}+\frac{c}{2}+\frac{a}{2}+\frac{1}{2}-\frac{c}{2}=\frac{1}{2}+\frac{1}{2}+\frac{a}{2}-\frac{a}{2}+\frac{c}{2}-\frac{c}{2}=1$$

Приклад 2.6. Щоб уявляти повну картину, потрібно зупинитись и і навести приклад.

Нехай помилки дорівнюють $a = 1 - c = 0.1$, тобто реле (або вузол) робить помилку в середньому один раз за десять тактів. Тоді матриця 2.44 в фазовому просторі E буде виглядати так:

$$P_{x_k \to x_{k+1}} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 0.45 & 0.45 & 0.05 & 0.05 \\ 0.45 & 0.45 & 0.05 & 0.05 \\ 0.45 & 0.45 & 0.05 & 0.05 \\ 0.45 & 0.45 & 0.05 & 0.05 \end{pmatrix} \end{array}$$

Нехай ймовірність помилки є меншою, тобто $a = 1 - c = 0.05$. Дана помилка означає, що реле(вузол) робить одну помилку на двадцять тактів. Тоді матриця 2.44 в фазовому просторі E виглядає так:

$$P_{x_k \to x_{k+1}} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 0.475 & 0.475 & 0.025 & 0.025 \\ 0.475 & 0.475 & 0.025 & 0.025 \\ 0.475 & 0.475 & 0.025 & 0.025 \\ 0.475 & 0.475 & 0.025 & 0.025 \end{pmatrix} \end{array}$$

Видно, що кожному випадку елементи матриці є невід'ємним та сума в кожному рядку є одиницею.

2.1.2 Додання керуючого впливу

Криптографічний модуль не є системою, яка живе своїм життям. На порти криптографічного модуля подаються команди та дані, які повинні обробитись та видати результат. Динамічна система, що повинна формально описувати математичну модель криптографічного модуля, з матрицею перехідних ймовірностей 2.42 є сама по собі автономною системою, що переходить з стану x_k до x_{k+1} з ймовірністю $p_{x_k \to x_{k+1}}$, без зовнішніх керуючих впливів.

Використання керуючого впливу повинно змінювати матрицю з розподілом ймовірностей 2.42 так, щоб дана матриця зберігла властивість стохастичності справа 2.43. Тобто, коли подається керуючий сигнал $s \in E_+$, то динамічна система 2.40 з набагато більшою ймовірністю перейде в той стан $s \in E_+$, в який ми хочемо перейти з поточного стану $x_k \in E$. Але все таки система може переходити в помилковий стан, оскільки динамічна система має, зазвичай малі, ймовірності переходу в помилковий стан.

Для матриці $P_{x_k \to x_{k+1}}$ 2.42 має сенс запис

$$P_{x_k \to x_{k+1}} = \begin{pmatrix} p_{0\to0} & p_{0\to1} & p_{0\to2} & p_{0\to3} \\ p_{1\to0} & p_{1\to1} & p_{1\to2} & p_{1\to3} \\ p_{2\to0} & p_{2\to1} & p_{2\to2} & p_{2\to3} \\ p_{3\to0} & p_{3\to1} & p_{3\to2} & p_{3\to3} \end{pmatrix} = \begin{pmatrix} \vec{p_0} \\ \vec{p_1} \\ \vec{p_2} \\ \vec{p_3} \end{pmatrix}$$

який показує, що матриця 2.42 складається з розподілів $\vec{p_i}$ кожного стану i фазового простору E.

Означення 2.3. Керуючим впливом називається функція

$$C : [0,1]^{|E|} \times E_+ \to [0,1] \tag{2.45}$$

яка в явному вигляді записується

$$C(\vec{p_v},s) = \mathbf{1}\{x_{k+1} = s\} \cdot \sum_{i \in E_+} p_{s \to i} \tag{2.46}$$

де

$\vec{p}$ - розподіл стану v, $v \in E$

$s \in E_+$ - правильний стан з простору правильних станів

$\mathbf{1}\{x_{k+1} = s\}$ - індикаторна функція

Нагадаємо, що індикаторна функція виглядає так:

$$\mathbf{1}\{x_{k+1} = s\} = \begin{cases} 1 & \text{, якщо } x_{k+1} = s \\ 0 & \text{, якщо } x_{k+1} \neq s \end{cases}$$

Якщо прийшов керуючий сигнал s, тоді матриця перехідних ймовіностей 2.42 прийме вигляд:

$$\mathbb{C}(s) \circ P_{x_k \to x_{k+1}} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{pmatrix} C(\vec{p_0},s) & C(\vec{p_0},s) & p_{0\to2} & p_{0\to3} \\ C(\vec{p_1},s) & C(\vec{p_1},s) & p_{1\to2} & p_{1\to3} \\ C(\vec{p_2},s) & C(\vec{p_2},s) & p_{2\to2} & p_{2\to3} \\ C(\vec{p_3},s) & C(\vec{p_3},s) & p_{3\to2} & p_{3\to3} \end{pmatrix} \end{array} \tag{2.47}$$

Приклад 2.7. Нехай подається сигнал s на перехід до стану 0, тобто $s = 0$. Тоді матриця ймовірності переходів 2.44 буде виглядати так:

$$\mathbb{C}(0) \circ P_{x_k \to x_{k+1}} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{pmatrix} C(\vec{p_0},0) & C(\vec{p_0},0) & \frac{a}{2} & \frac{1-c}{2} \\ C(\vec{p_1},0) & C(\vec{p_1},0) & \frac{a}{2} & \frac{1-c}{2} \\ C(\vec{p_2},0) & C(\vec{p_2},0) & \frac{a}{2} & \frac{1-c}{2} \\ C(\vec{p_3},0) & C(\vec{p_3},0) & \frac{a}{2} & \frac{1-c}{2} \end{pmatrix} \end{array} =$$

$$= \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{pmatrix} \frac{1-a}{2} + \frac{c}{2} & 0 & \frac{a}{2} & \frac{1-c}{2} \\ \frac{1-a}{2} + \frac{c}{2} & 0 & \frac{a}{2} & \frac{1-c}{2} \\ \frac{1-a}{2} + \frac{c}{2} & 0 & \frac{a}{2} & \frac{1-c}{2} \\ \frac{1-a}{2} + \frac{c}{2} & 0 & \frac{a}{2} & \frac{1-c}{2} \end{pmatrix} \end{array}$$

Використання керуючого впливу 2.46 в матриці 2.44 зберігла властивість стохастичності справа. Інакше кажучі, використання керуючого впливу 2.46 вплинуло на розподіл кожного стану з фазового простору.

2.1.3 Переформулювання схеми з ненадійних реле з двома станами

Оскільки ми тепер працюємо в фазовому просторі E, на якому ми задали ймовірнісну міру 2.36, яка описує розподіл на E, то потрібно коректно вести ймовірність роботи схеми з ненадійних реле з двома станами на простір E

Нагадаємо, що вирогідність замикання схеми описувалась функцією

$$h(p) = \sum_{n=0}^{m} A_n \cdot p^n \cdot (1-p)^{m-n} \tag{2.48}$$

де

m - кількість контактів,

p - ймовірність замикання одного реле,

A_n - кількість способів якими можна обрати множину із n контактів так, що схема замкнена.

Фактично, функція $h(p)$ є деяким перетворенням, що приймає на вхід ймовірності, що задані на ймовірнісному просторі 2.12 і повертає ймовірність роботи схеми з ненадійних реле з двома станами. Спробуємо переформувати функцію $h(p)$ на більш щось узагальнене.

Означення 2.4. Нехай є одний небезпечний вузол, яке в стані $s \in E_+$ має ймовірність правильної роботи $1-\alpha$, а ймовірність помилкової роботи α. Тоді ймовірність безпечної роботи схеми з m однакових і незалежних елементів

$$h_s(p) = \sum_{n=0}^{m} A_n(s) \cdot p^n \cdot (1-p)^{m-n} \tag{2.49}$$

де

$A_n(s)$ -кількість способів, якими можна обрати множину з n вузлів так, що схема знаходиться в стані $s \in E_+$,

p - ймовірність роботи, $p \in [0,1]$

Відповідно до 2.49 ймовірність правильної роботи $h_s(1-\alpha)$, а ймовірність помилки $h_s(\alpha)$. Зауважимо, що $\alpha \geqslant h_s(\alpha)$ та $1-\alpha \leqslant h_s(1-\alpha)$

Ввівши більш узагальнену функцію $h_s(p)$ ми позбавились від прив'язки до замикання реле і узагальнили ймовірність роботи відповідно до стану s.

Тепер виникає питання: "Як рахувати $A_n(s)$?". Для цього введемо модель графу

$$\mathbb{G} =< K, V, e : K \to E > \tag{2.50}$$

де

K - множина вершин;

$V \subseteq K \times K$ - множина ребер;

$e : K \to E$ - відображення з множини вершин K до фазового простору E, яке фактично говорить про те в якому стані знаходиться реле(вузол).

По-перше, треба перелічити які взагалі вузли і кожному вузлу присвоїти унікальний номер. По-друге, треба перелічити зв'язки між вузлами. Маючи вузли та зв'язки між вузлами можна описати деяку модель графу, що описує вузли як вершини і зв'язки як ребра. Нехай вузли перелічені і відома матриця сміжності, в якої показано зв'язок між вузлами. Фактично, в реальності це можна інтерпретувати як користувачів в онлайні, з якими можна взаємодіяти, тобто відправляти дані, запитувати дані і підтримувати роботу децентралізованої мережі. Коли користувач під'єднується до мережі, то він запитує у кожного хто кого знає і у відповідь відправляють список користувачів, а далі користувач ці списки обробляє. Приблизно таким чином створюються матриці сміжності. Повертаючись до обчислення $A_n(s)$, це можна обрахувати як кількість шляхів з початкової вершини до найвіддаленіших вершин довжини n, в яких кожна вершина знаходиться в стані s.

Нехай побудована схема з m реле(вузлів), тоді з ймовірностей 2.14,2.15,2.16,2.17 можна перейти до ймовірностей роботи схеми. Інакше кажучі, ймовірності правильної роботи збільшилась і відповідно ймовірності помилкової роботи зменшились, тобто

$$\mathbb{P}_0(\{0\}) \longrightarrow h_0(\mathbb{P}_0(\{0\})) \tag{2.51}$$

$$\mathbb{P}_0(\{2\}) \longrightarrow h_0(\mathbb{P}_0(\{2\})) \tag{2.52}$$

$$\mathbb{P}_1(\{1\}) \longrightarrow h_1(\mathbb{P}_1(\{1\})) \tag{2.53}$$

$$\mathbb{P}_1(\{3\}) \longrightarrow h_1(\mathbb{P}_1(\{3\})) \tag{2.54}$$

Зауваження. Сумма $h_s(p) + h_s(1-p)$ не завжди дорівнює 1. Дана

сума може дорівнювати одиниці якщо схема(топологія вузлів) симетрична. Симетричні топології з вузлів не завжди досягається в реальних топологіях мереж.

Приклад 2.8. Нехай $a = 1 - c = 0.2$, $h_0(p) = 2 \cdot p^2 - p^4$, що відповідає схемі в прикладі 1.1. Те саме стосується і функції $h_1(p)$. Тоді міркування наведені вище можна зобразити на рисунку:

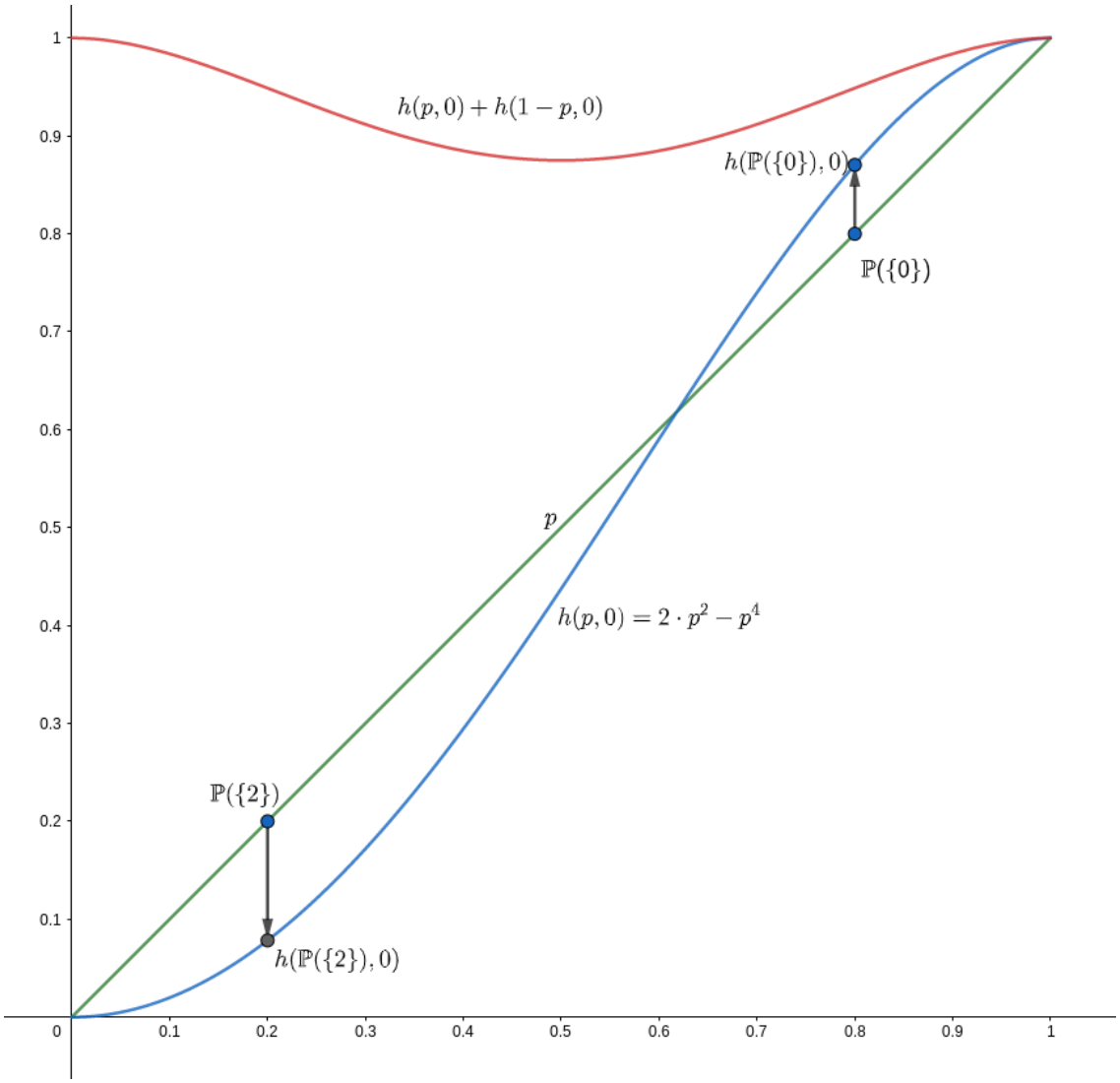


Рисунок 2.2 – Явне зображення того, що $h_s(p) + h_s(1-p) \neq 1$

Також повинна виконуватись умова 2.34, з якої знайдемо $\vec{w}$.

$$\mathbb{P}(E) = (\vec{w}, \vec{\mathbb{P}}_{0,n-1}(E)) = 1 \tag{2.55}$$

$$\mathbb{P}(E) = w_0 \cdot \mathbb{P}_0(G_{\mathbb{B}\to\mathbb{B}_0}(E)) + w_0 \cdot \mathbb{P}_1(G_{\mathbb{B}\to\mathbb{B}_1}(E)) =$$
$$= w_0 \cdot \mathbb{P}_0(G_{\mathbb{B}\to\mathbb{B}_0}(\{0,1,2,3\})) + w_0 \cdot \mathbb{P}_1(G_{\mathbb{B}\to\mathbb{B}_1}(\{0,1,2,3\})) =$$
$$= w_0 \cdot \mathbb{P}_0(G_{\mathbb{B}\to\mathbb{B}_0}(\{0\} \cup \{1\} \cup \{2\} \cup \{3\}))+$$
$$+ w_1 \cdot \mathbb{P}_1(G_{\mathbb{B}\to\mathbb{B}_1}(\{0\} \cup \{1\} \cup \{2\} \cup \{3\})) =$$
$$= w_0 \cdot \mathbb{P}_0(G_{\mathbb{B}\to\mathbb{B}_0}(\{0\}) \cup G_{\mathbb{B}\to\mathbb{B}_0}(\{1\}) \cup G_{\mathbb{B}\to\mathbb{B}_0}(\{2\}) \cup G_{\mathbb{B}\to\mathbb{B}_0}(\{3\}))+$$
$$+ w_1 \cdot \mathbb{P}_1(G_{\mathbb{B}\to\mathbb{B}_1}(\{0\}) \cup G_{\mathbb{B}\to\mathbb{B}_1}(\{1\}) \cup G_{\mathbb{B}\to\mathbb{B}_1}(\{2\}) \cup G_{\mathbb{B}\to\mathbb{B}_1}(\{3\})) =$$
$$= w_0 \cdot \mathbb{P}_0(\{0\} \cup \varnothing \cup \{2\} \cup \varnothing) + w_1 \cdot \mathbb{P}_0(\varnothing \cup \{1\} \cup \varnothing \cup \{3\}) =$$
$$= w_0 \cdot \mathbb{P}_0(\{0\} \cup \{2\}) + w_1 \cdot \mathbb{P}_1(\{1\} \cup \{3\}) = |\sigma\text{-аддитивність 2.20}| =$$
$$= w_0 \cdot (\mathbb{P}_0(\{0\}) + \mathbb{P}_0(\{2\})) + w_1 \cdot (\mathbb{P}_1(\{1\}) + \mathbb{P}_1(\{3\})) \to$$
$$\to |\text{Будуємо схему з реле} \equiv \text{Робимо перерозподіл}| \to$$
$$\to w_0 \cdot (h_0(\mathbb{P}_0(\{0\})) + h_0(\mathbb{P}_0(\{2\})))+$$
$$+ w_1 \cdot (h_1(\mathbb{P}_1(\{1\})) + h_1(\mathbb{P}_1(\{3\}))) =$$
$$= w_0 \cdot (h_0(1-a) + h_0(a)) + w_1 \cdot (h_1(c) + h_1(1-c)) = 1$$

За припущенням, що $w_0 = w_1$ (2.35) :

$$w_0 \cdot (h_0(1-a) + h_0(a) + h_1(c) + h_1(1-c)) = 1$$
$$\implies w_0 = w_1 = \frac{1}{h_0(1-a) + h_0(a) + h_1(c) + h_1(1-c)}$$

Таким чином знайшли вектор ваг $\vec{w}$ для схеми з ненадійних реле з ймовірностями помилок a та $1-c$ для того щоб зробити перерозподіл матриці перехідних ймовірностей в фазовому просторі E. Запишемо матрицю перехідних ймовірностей 2.44 для схеми з реле згідно переходів 2.51,2.52,2.53,2.54.

$$P_{x_k \to x_{k+1}} = \begin{pmatrix} w_0 \cdot h_0(1-a) & w_0 \cdot h_1(c) & w_0 \cdot h_0(a) & w_0 \cdot h_1(1-c) \\ w_0 \cdot h_0(1-a) & w_0 \cdot h_1(c) & w_0 \cdot h_0(a) & w_0 \cdot h_1(1-c) \\ w_0 \cdot h_0(1-a) & w_0 \cdot h_1(c) & w_0 \cdot h_0(a) & w_0 \cdot h_1(1-c) \\ w_0 \cdot h_0(1-a) & w_0 \cdot h_1(c) & w_0 \cdot h_0(a) & w_0 \cdot h_1(1-c) \end{pmatrix}$$

Можна також переконатись, що сумма елементів в кожному рядку дорівнює одиниці.

$$\begin{aligned} &w_0 \cdot h_0(1-a) + w_0 \cdot h_1(c) + w_0 \cdot h_0(a) + w_0 \cdot h_1(1-c) = \\ &\quad = w_0 \cdot (h_0(1-a) + h_1(c) + h_0(a) + h_1(1-c)) = w_0 \cdot w_0^{-1} = 1 \end{aligned}$$

Висновок до пункту 2.1

Отже, було переформульовано модель ненадійного реле з двома станами в термінах скінченного ймовірнісного автомату, що задовільняє потреби бажаної моделі децентралізованого криптографічного модуля. Також був доданий керуючий вплив до переформульваної моделі та переформульована ймовірність роботи схеми з реле, що вкрай необхідно для нової моделі.

2.2 Узагальнення моделі

В секції 2.1 було переформульована модель ненадійного реле з двома станами. Тепер стоїть задача в узагальнені данної моделі для $n \geqslant 2$ правильних станів. Оскільки модель ненадійних реле з двома станами була переформульована так, що вона оперує новими загальними термірнами та має представлення в вигляді скінченного ймовірнісного автомату, то узагальнення є деяким логічним продовженням побудови

моделі.

Повторимо алгоритм, що ми використовували для переформулювання моделі, але вже зробимо це для загального випадку. Нехай є $n \geqslant 2$ станів, де на кожен стан є відповідний помилковий стан. Тоді фазовий простір буде виглядати так:

$$E = \{0,1,\ldots,n-1,n,\ldots,2\cdot n-1\} \tag{2.56}$$

Що може бути розбитий на правильні стани E_+ та помилкові стани E_-.

$$E_+ = \{0,1,\ldots,n-1\} \tag{2.57}$$

$$E_- = \{n,n+1,\ldots,2\cdot n-1\} \tag{2.58}$$

Побудуємо ймовірнісні простори для кожного стану $i \in E_+$

$$(\Omega_i,\mathbb{B}_i,\mathbb{P}_i) \tag{2.59}$$

де простір елементарних подій Ω_i та σ-алгебра $\mathbb{B}_i$ мають вигляд:

$$\Omega_i = \{i,i+n\} \tag{2.60}$$

$$\mathbb{B}_i = \{\varnothing,\{i\},\{i+n\},\{i,i+n\}\} \tag{2.61}$$

А ймовірнісна міра для кожної події, що в явному записується

$$\mathbb{P}_i(\{i\}) = 1-\alpha_i, \tag{2.62}$$

$$\mathbb{P}_i(\{i+n\}) = \alpha_i \tag{2.63}$$

Об'єднаємо простори елементарних подій і отримаємо фазовий простір E

$$\Omega_0 \cup \Omega_1 \cup \ldots \cup \Omega_{n-1} = E \tag{2.64}$$

На просторі E побудуємо σ-алгебру $\mathbb{B}$

$$\mathbb{B} = \{\varnothing,\{0\},\{1\},\ldots,E\} \tag{2.65}$$

Зауваження. Кількість елементів в множині $\mathbb{B}$ дорівнює $2^{2\cdot n}$

Ймовірнісну міру на $\mathbb{B}$ задамо згідно означення 2.1 та функції $G_{\mathbb{B}\to\mathbb{B}_i}$, яка відображає події з σ-алгебри $\mathbb{B}$ на σ-алгебру $\mathbb{B}_i$, що визначає теорема 2.1 та умовою 2.34. Нехай $A \in \mathbb{B}$, тоді

$$\mathbb{P}(A) = w_0 \cdot \mathbb{P}_0(G_{\mathbb{B}\to\mathbb{B}_0}(A)) + \ldots + w_{n-1} \cdot \mathbb{P}_{n-1}(G_{\mathbb{B}\to\mathbb{B}_{n-1}}(A)) \tag{2.66}$$

А матриця перехідних ймовірностей зі стану $x_k \in E$ до стану $x_{k+1} \in E$ буде мати вигляд

$$P_{x_k\to x_{k+1}} = \begin{pmatrix} p_{0\to0} & p_{0\to1} & \cdots & p_{0\to2\cdot n-1} \\ p_{1\to0} & p_{1\to1} & \cdots & p_{1\to2\cdot n-1} \\ \cdots & \cdots & \cdots & \cdots \\ p_{2\cdot n-1\to0} & p_{2\cdot n-1\to1} & \cdots & p_{2\cdot n-1\to2\cdot n-1} \end{pmatrix} \tag{2.67}$$

Зауваження. Розмірність матриці 2.67 дорівнює $2\cdot n \times 2\cdot n$

Матриця 2.67 при керуючому впливі 2.46 буде мати вигляд

$$\mathbb{C}(s)\circ P_{x_k\to x_{k+1}} = \begin{pmatrix} C(\vec{p_0},s) & \ldots & C(\vec{p_0},s) & p_{0\to n} & \cdots & p_{0\to2\cdot n-1} \\ C(\vec{p_1},s) & \ldots & C(\vec{p_1},s) & p_{1\to n} & \cdots & p_{1\to2\cdot n-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ C(\vec{p_{2\cdot n-1}},s) & \ldots & C(\vec{p_{2\cdot n-1}},s) & p_{2\cdot n-1\to n} & \cdots & p_{2\cdot n-1\to2\cdot n-1} \end{pmatrix}$$

А при схемі з вузлів виконається такий перехід із ймовірності роботи або помилки одного елемента в функцію від ймовірності роботи схеми таким чином:

$$\mathbb{P}_0(\{0\}) \longrightarrow h_0(\mathbb{P}_0(\{0\}))$$

$$\mathbb{P}_1(\{1\}) \longrightarrow h_1(\mathbb{P}_0(\{1\}))$$

$$\dots$$

$$\mathbb{P}_{n-1}(\{n-1\}) \longrightarrow h_{n-1}(\mathbb{P}_1(\{n-1\}))$$

$$\mathbb{P}_0(\{n\}) \longrightarrow h_0(\mathbb{P}_1(\{n\}))$$

$$\mathbb{P}_1(\{n+1\}) \longrightarrow h_1(\mathbb{P}_1(\{n+1\}))$$

$$\dots$$

$$\mathbb{P}_{n-1}(\{2\cdot n-1\}) \longrightarrow h_{n-1}(\mathbb{P}_1(\{2\cdot n-1\}))$$

Вектор ваг визначається через вхідний вектор $\vec{r} = (r_0,...,r_{n-1})$ та таких вимог:

$$\begin{cases} r_0 \cdot w_0 = r_1 \cdot w_1 = ... = r_{n-1} \cdot w_{n-1} \\ \sum\limits_{i=0}^{n-1} w_i \cdot (h_i(\mathbb{P}_i(\{i\})) + h_i(\mathbb{P}_i(\{i+n\}))) = 1 \end{cases}$$

І тоді матриця 2.67 прийме вигляд

$$P_{x_k \to x_{k+1}} = \begin{pmatrix} w_0 \cdot h_0(p_{0\to 0}) & \dots & w_{n-1} \cdot h_{n-1}(p_{0\to n-1}) & w_0 \cdot h_0(p_{0\to n}) & \dots & w_{n-1} \cdot h_{n-1}(p_{0\to 2\cdot n-1}) \\ w_0 \cdot h_0(p_{0\to 0}) & \dots & w_{n-1} \cdot h_{n-1}(p_{0\to n-1}) & w_0 \cdot h_0(p_{0\to n}) & \dots & w_{n-1} \cdot h_{n-1}(p_{0\to 2\cdot n-1}) \\ \dots & \dots & \dots & \dots & \dots & \dots \\ w_0 \cdot h_0(p_{0\to 0}) & \dots & w_{n-1} \cdot h_{n-1}(p_{0\to n-1}) & w_0 \cdot h_0(p_{0\to n}) & \dots & w_{n-1} \cdot h_{n-1}(p_{0\to 2\cdot n-1}) \\ w_0 \cdot h_0(p_{0\to 0}) & \dots & w_{n-1} \cdot h_{n-1}(p_{0\to n-1}) & w_0 \cdot h_0(p_{0\to n}) & \dots & w_{n-1} \cdot h_{n-1}(p_{0\to 2\cdot n-1}) \end{pmatrix}$$

Висновок до пункту 2.2

Отже, було формально описано побудову моделі криптографічного модуля з $n \geqslant 2$ станами.

2.3 Алгоритми на основі узагальненої моделі

Вже було побудовано загальну модель децентралізованого криптомодуля в секції 2.2, яка визначається через кількість правильних станів n та ймовірності помилок у кожного помилкового стану α_i, $i = \overline{0,n-1}$. Тепер побудуємо алгоритми на основі побудованої формальної моделі криптографічного модуля, щоб виділити основні результати.

Алгоритм 2.1. Перенесення реального криптографічного модуля в модель

Вхід: кількість станів n; оціночні ймовірності помилок $\vec{\alpha} = (\alpha_0, \ldots, \alpha_{n-1})$; вектор коефіцієнтів $\vec{r} = (r_0, \ldots, r_{n-1})$

Вихід:Матриця перехідних ймовірностей P

1. Побудувати фазовий простір $E = \{0,1, \ldots, n-1, n, \ldots, 2 \cdot n - 1\}$

2. Визначити рівності в векторі ваг $\vec{w} = (w_0, w_1, \ldots, w_{n-1})$, яке формально говорить значущості i-ї помики відностно інших, тобто $r_0 \cdot w_0 = \ldots = r_{n-1} \cdot w_{n-1}$.

3. Обчислити $w_i, i = \overline{0,n-1}$ з умови $\sum\limits_{i=0}^{n-1} w_i \cdot \mathbb{P}_i(G_{\mathbb{B} \to \mathbb{B}_i}(E)) = 1$ та кроку 2.

4. Побудувати матрицю перехідних ймовірностей P зі стану $x_k \in E$ до $x_{k+1} \in E$.

$$\begin{bmatrix} w_0 \cdot (1-\alpha_0) & w_1 \cdot (1-\alpha_1) & \ldots & w_{n-1} \cdot (1-\alpha_{n-1}) & w_0 \cdot \alpha_0 & w_1 \cdot \alpha_1 & \ldots & w_{n-1} \cdot \alpha_{n-1} \\ w_0 \cdot (1-\alpha_0) & w_1 \cdot (1-\alpha_1) & \ldots & w_{n-1} \cdot (1-\alpha_{n-1}) & w_0 \cdot \alpha_0 & w_1 \cdot \alpha_1 & \ldots & w_{n-1} \cdot \alpha_{n-1} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ w_0 \cdot (1-\alpha_0) & w_1 \cdot (1-\alpha_1) & \ldots & w_{n-1} \cdot (1-\alpha_{n-1}) & w_0 \cdot \alpha_0 & w_1 \cdot \alpha_1 & \ldots & w_{n-1} \cdot \alpha_{n-1} \end{bmatrix}$$

Зауваження. Характеристики криптографічного модуля, що йдуть на вхід в алгоритм 2.1 можуть бути виміряні зі схеми з компонентів, що мають n станів. Якщо одна похибка не вагоміша ніж друга, то вектор коефіцієнтів $\vec{r} = (1, \ldots, 1)$.

Приклад 2.9. Наведемо застосування алгоритму 2.1.

Нехай $n = 2$, $\vec{\alpha} = (0.1,0.2)$, $\vec{r} = (1,1)$.

Крок 1. Формуємо фазовий простір: $E = \{0,1,2,3\}$.

Крок 2. Визначаємо рівність в векторі ваг:

$$r_0 \cdot w_0 = r_1 \cdot w_1$$
$$1 \cdot w_0 = 1 \cdot w_1$$
$$w_0 = w_1$$

Крок 3. Обчислюємо $\vec{w} = (w_0,w_1)$ таке що $\sum\limits_{i=0}^{1} w_i \cdot \mathbb{P}_i(G_{\mathbb{B}\to\mathbb{B}_i}(E)) = 1$.

$$\sum_{i=0}^{1} w_i \cdot \mathbb{P}_i(G_{\mathbb{B}\to\mathbb{B}_i}(E)) = w_0 \cdot \mathbb{P}_0(G_{\mathbb{B}\to\mathbb{B}_0}(E)) + w_1 \cdot \mathbb{P}_1(G_{\mathbb{B}\to\mathbb{B}_1}(E)) =$$
$$= w_0 \cdot \mathbb{P}_0(G_{\mathbb{B}\to\mathbb{B}_0}(\{0,1,2,3\})) + w_1 \cdot \mathbb{P}_1(G_{\mathbb{B}\to\mathbb{B}_1}(\{0,1,2,3\})) =$$
$$= w_0 \cdot \mathbb{P}_0(G_{\mathbb{B}\to\mathbb{B}_0}(\{0\}) \cup G_{\mathbb{B}\to\mathbb{B}_0}(\{1\}) \cup G_{\mathbb{B}\to\mathbb{B}_0}(\{2\}) \cup G_{\mathbb{B}\to\mathbb{B}_0}(\{3\}))+$$
$$+ w_1 \cdot \mathbb{P}_1(G_{\mathbb{B}\to\mathbb{B}_1}(\{0\}) \cup G_{\mathbb{B}\to\mathbb{B}_1}(\{1\}) \cup G_{\mathbb{B}\to\mathbb{B}_1}(\{2\}) \cup G_{\mathbb{B}\to\mathbb{B}_1}(\{3\})) =$$
$$= w_0 \cdot \mathbb{P}_0(\{0\} \cup \varnothing \cup \{2\} \cup \varnothing) + w_1 \cdot \mathbb{P}_1(\varnothing \cup \{1\} \cup \varnothing \cup \{3\}) =$$
$$= w_0 \cdot \mathbb{P}_0(\{0\} \cup \{2\}) + w_1 \cdot \mathbb{P}_1(\{1\} \cup \{3\}) = w_0 \cdot \mathbb{P}_0(\{0,2\}) + w_1 \cdot \mathbb{P}_1(\{1,3\}) =$$
$$= w_0 \cdot 1 + w_1 \cdot 1 = w_0 + w_1 = |\text{з кроку 2: } w_0 = w_1| = w_0 + w_0 = 2 \cdot w_0$$

$\Longrightarrow 2 \cdot w_0 = 1 \Longrightarrow w_0 = \frac{1}{2}$.

Оскільки $w_0 = w_1$, то $w_1 = \frac{1}{2}$.

Крок 4. Будуємо матрицю перехідних ймовірностей P:

$$\text{P}= \begin{bmatrix} w_0 \cdot (1-\alpha_0) & w_1 \cdot (1-\alpha_1) & w_0 \cdot \alpha_0 & w_1 \cdot \alpha_1 \\ w_0 \cdot (1-\alpha_0) & w_1 \cdot (1-\alpha_1) & w_0 \cdot \alpha_0 & w_1 \cdot \alpha_1 \\ w_0 \cdot (1-\alpha_0) & w_1 \cdot (1-\alpha_1) & w_0 \cdot \alpha_0 & w_1 \cdot \alpha_1 \\ w_0 \cdot (1-\alpha_0) & w_1 \cdot (1-\alpha_1) & w_0 \cdot \alpha_0 & w_1 \cdot \alpha_1 \end{bmatrix} =$$

$$= \begin{bmatrix} \frac{1}{2}\cdot(1-0.1) & \frac{1}{2}\cdot(1-0.2) & \frac{1}{2}\cdot 0.1 & \frac{1}{2}\cdot 0.2 \\ \frac{1}{2}\cdot(1-0.1) & \frac{1}{2}\cdot(1-0.2) & \frac{1}{2}\cdot 0.1 & \frac{1}{2}\cdot 0.2 \\ \frac{1}{2}\cdot(1-0.1) & \frac{1}{2}\cdot(1-0.2) & \frac{1}{2}\cdot 0.1 & \frac{1}{2}\cdot 0.2 \\ \frac{1}{2}\cdot(1-0.1) & \frac{1}{2}\cdot(1-0.2) & \frac{1}{2}\cdot 0.1 & \frac{1}{2}\cdot 0.2 \end{bmatrix} =$$

$$= \begin{bmatrix} 0.45 & 0.4 & 0.05 & 0.1 \\ 0.45 & 0.4 & 0.05 & 0.1 \\ 0.45 & 0.4 & 0.05 & 0.1 \\ 0.45 & 0.4 & 0.05 & 0.1 \end{bmatrix}$$

Алгоритм 2.2. Побудова матриці P при схемі з небезпечних елементів.

Вхід: m незалежних компонент, що мають n станів кожна; ймовірність помилок $\vec{\alpha} = (\alpha_0, \ldots, \alpha_{n-1})$; вектор коефіцієнтів $\vec{r} = (r_0, \ldots, r_{n-1})$

Вихід: Більш надійний криптомодуль з матрицею перехідних ймовірностей P

1. Побудувати фазовий простір $E = \{0,1,\ldots,n-1,n,\ldots,2\cdot n-1\}$

2. Визначити рівності в векторі ваг $\vec{w} = (w_0, w_1, \ldots, w_{n-1})$, як $r_0 \cdot w_0 = \ldots = r_{n-1} \cdot w_{n-1}$.

3. Обрахувати w_i згідно кроку 2 та умови
$\sum_{i=0}^{n-1} w_i \cdot (h_i(\mathbb{P}_i(\{i\})) + h_i(\mathbb{P}_i(\{i+n\}))) = 1$

4. Сформувати матрицю перехідних ймовірностей P

$$\begin{bmatrix} w_0 \cdot h(1-\alpha_0,0) & \ldots & w_{n-1} \cdot h(1-\alpha_{n-1},n-1) & w_0 \cdot h(\alpha_0,0) & \ldots & w_{n-1} \cdot h(\alpha_{n-1},n-1) \\ w_0 \cdot h(1-\alpha_0,0) & \ldots & w_{n-1} \cdot h(1-\alpha_{n-1},n-1) & w_0 \cdot h(\alpha_0,0) & \ldots & w_{n-1} \cdot h(\alpha_{n-1},n-1) \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ w_0 \cdot h(1-\alpha_0,0) & \ldots & w_{n-1} \cdot h(1-\alpha_{n-1},n-1) & w_0 \cdot h(\alpha_0,0) & \ldots & w_{n-1} \cdot h(\alpha_{n-1},n-1) \end{bmatrix}$$

Приклад 2.10. Нехай схема з прикладу 1.1, яка має $h(p) = 2 \cdot p^2 - p^4$, $m = 4$ та $n = 3$, $\vec{\alpha} = (0.1,0.2,0.15)$, $\vec{r} = (1,1,1)$

Крок 1. Будуємо фазовий простір станів: $E = \{0,1,2,3,4,5\}$

Крок 2. Визначимо рівності в векторі ваг: $w_0 = w_1 = w_2$

Крок 3. Знаходимо $\vec{w}$ таке що $\sum\limits_{i=0}^{n-1} w_i \cdot (h(\mathbb{P}_i(\{i\}),i) + h(\mathbb{P}_i(\{i+n\}),i)) = 1$

$$\sum_{i=0}^{n-1} w_i \cdot (h(\mathbb{P}_i(\{i\}),i) + h(\mathbb{P}_i(\{i+n\}),i)) = w_0 \cdot (h(\mathbb{P}_0(\{0\}),0) + h(\mathbb{P}_0(\{3\}),0)) +$$

$$+ w_1 \cdot (h(\mathbb{P}_1(\{1\}),1) + h(\mathbb{P}_1(\{4\}),1)) + w_2 \cdot (h(\mathbb{P}_2(\{2\}),2) + h(\mathbb{P}_2(\{5\}),2)) =$$

$$= w_0 \cdot (h(\mathbb{P}_0(\{0\})) + h(\mathbb{P}_0(\{3\}))) + w_1 \cdot (h(\mathbb{P}_1(\{1\})) + h(\mathbb{P}_1(\{4\}))) +$$

$$+ w_2 \cdot (h(\mathbb{P}_2(\{2\})) + h(\mathbb{P}_2(\{5\}))) = w_0 \cdot (h(1-\alpha_0) + h(\alpha_0)) +$$

$$+ w_1 \cdot (h(1-\alpha_1) + h(\alpha_1)) + w_2 \cdot (h(1-\alpha_2) + h(\alpha_2))) = w_0 \cdot (h(1-0.1) + h(0.1)) +$$

$$+ w_1 \cdot (h(1-0.2) + h(0.2)) + w_2 \cdot (h(1-0.15) + h(0.15))) = w_0 \cdot (h(0.9) + h(0.1)) +$$

$$+ w_1 \cdot (h(0.8) + h(0.2)) + w_2 \cdot (h(0.85) + h(0.15))) = |h(p) = 2 \cdot p^2 - p^4| \approx$$

$$\approx w_0 \cdot (0.9639 + 0.0199) + w_1 \cdot (0.8704 + 0.0784) + w_2 \cdot (0.9229 + 0.0444)) =$$

$$= w_0 \cdot 0.9838 + w_1 \cdot 0.9488 + w_2 \cdot 0.9674 = |w_0 = w_1 = w_2| =$$

$$= w_0 \cdot (0.9838 + 0.9488 + 0.9674) \approx w_0 \cdot 2.9$$

$\Longrightarrow w_0 \cdot 2.9 = 1 \Longrightarrow w_0 = \frac{1}{2.9} = 0.344827586$

Оскільки $w_0 = w_1 = w_2$, то $w_1 = \frac{1}{2.9} = 0.344827586$, $w_2 = \frac{1}{2.9} = 0.344827586$

Крок 4. Формуємо матрицю перехідних ймовірностей P:

$$\mathrm{P}= \begin{bmatrix} w_0\cdot(1-\alpha_0) & w_1\cdot(1-\alpha_1) & w_2\cdot(1-\alpha_2) & w_0\cdot\alpha_0 & w_1\cdot\alpha_1 & w_2\cdot\alpha_2 \\ w_0\cdot(1-\alpha_0) & w_1\cdot(1-\alpha_1) & w_2\cdot(1-\alpha_2) & w_0\cdot\alpha_0 & w_1\cdot\alpha_1 & w_2\cdot\alpha_2 \\ w_0\cdot(1-\alpha_0) & w_1\cdot(1-\alpha_1) & w_2\cdot(1-\alpha_2) & w_0\cdot\alpha_0 & w_1\cdot\alpha_1 & w_2\cdot\alpha_2 \\ w_0\cdot(1-\alpha_0) & w_1\cdot(1-\alpha_1) & w_2\cdot(1-\alpha_2) & w_0\cdot\alpha_0 & w_1\cdot\alpha_1 & w_2\cdot\alpha_2 \\ w_0\cdot(1-\alpha_0) & w_1\cdot(1-\alpha_1) & w_2\cdot(1-\alpha_2) & w_0\cdot\alpha_0 & w_1\cdot\alpha_1 & w_2\cdot\alpha_2 \\ w_0\cdot(1-\alpha_0) & w_1\cdot(1-\alpha_1) & w_2\cdot(1-\alpha_2) & w_0\cdot\alpha_0 & w_1\cdot\alpha_1 & w_2\cdot\alpha_2 \end{bmatrix} =$$

$$= \begin{bmatrix} \frac{1}{2.9}\cdot h(1-0.1) & \frac{1}{2.9}\cdot h(1-0.2) & \frac{1}{2.9}\cdot h(1-0.15) & \frac{1}{2.9}\cdot h(0.1) & \frac{1}{2.9}\cdot h(0.2) & \frac{1}{2.9}\cdot h(0.15) \\ \frac{1}{2.9}\cdot h(1-0.1) & \frac{1}{2.9}\cdot h(1-0.2) & \frac{1}{2.9}\cdot h(1-0.15) & \frac{1}{2.9}\cdot h(0.1) & \frac{1}{2.9}\cdot h(0.2) & \frac{1}{2.9}\cdot h(0.15) \\ \frac{1}{2.9}\cdot h(1-0.1) & \frac{1}{2.9}\cdot h(1-0.2) & \frac{1}{2.9}\cdot h(1-0.15) & \frac{1}{2.9}\cdot h(0.1) & \frac{1}{2.9}\cdot h(0.2) & \frac{1}{2.9}\cdot h(0.15) \\ \frac{1}{2.9}\cdot h(1-0.1) & \frac{1}{2.9}\cdot h(1-0.2) & \frac{1}{2.9}\cdot h(1-0.15) & \frac{1}{2.9}\cdot h(0.1) & \frac{1}{2.9}\cdot h(0.2) & \frac{1}{2.9}\cdot h(0.15) \\ \frac{1}{2.9}\cdot h(1-0.1) & \frac{1}{2.9}\cdot h(1-0.2) & \frac{1}{2.9}\cdot h(1-0.15) & \frac{1}{2.9}\cdot h(0.1) & \frac{1}{2.9}\cdot h(0.2) & \frac{1}{2.9}\cdot h(0.15) \\ \frac{1}{2.9}\cdot h(1-0.1) & \frac{1}{2.9}\cdot h(1-0.2) & \frac{1}{2.9}\cdot h(1-0.15) & \frac{1}{2.9}\cdot h(0.1) & \frac{1}{2.9}\cdot h(0.2) & \frac{1}{2.9}\cdot h(0.15) \end{bmatrix} =$$

$$= \begin{bmatrix} \frac{1}{2.9}\cdot h(0.9) & \frac{1}{2.9}\cdot h(0.8) & \frac{1}{2.9}\cdot h(0.85) & \frac{1}{2.9}\cdot h(0.1) & \frac{1}{2.9}\cdot h(0.2) & \frac{1}{2.9}\cdot h(0.15) \\ \frac{1}{2.9}\cdot h(0.9) & \frac{1}{2.9}\cdot h(0.8) & \frac{1}{2.9}\cdot h(0.85) & \frac{1}{2.9}\cdot h(0.1) & \frac{1}{2.9}\cdot h(0.2) & \frac{1}{2.9}\cdot h(0.15) \\ \frac{1}{2.9}\cdot h(0.9) & \frac{1}{2.9}\cdot h(0.8) & \frac{1}{2.9}\cdot h(0.85) & \frac{1}{2.9}\cdot h(0.1) & \frac{1}{2.9}\cdot h(0.2) & \frac{1}{2.9}\cdot h(0.15) \\ \frac{1}{2.9}\cdot h(0.9) & \frac{1}{2.9}\cdot h(0.8) & \frac{1}{2.9}\cdot h(0.85) & \frac{1}{2.9}\cdot h(0.1) & \frac{1}{2.9}\cdot h(0.2) & \frac{1}{2.9}\cdot h(0.15) \\ \frac{1}{2.9}\cdot h(0.9) & \frac{1}{2.9}\cdot h(0.8) & \frac{1}{2.9}\cdot h(0.85) & \frac{1}{2.9}\cdot h(0.1) & \frac{1}{2.9}\cdot h(0.2) & \frac{1}{2.9}\cdot h(0.15) \\ \frac{1}{2.9}\cdot h(0.9) & \frac{1}{2.9}\cdot h(0.8) & \frac{1}{2.9}\cdot h(0.85) & \frac{1}{2.9}\cdot h(0.1) & \frac{1}{2.9}\cdot h(0.2) & \frac{1}{2.9}\cdot h(0.15) \end{bmatrix} =$$

$$= \begin{bmatrix} \frac{1}{2.9}\cdot 0.9639 & \frac{1}{2.9}\cdot 0.8704 & \frac{1}{2.9}\cdot 0.9229 & \frac{1}{2.9}\cdot 0.0199 & \frac{1}{2.9}\cdot 0.0784 & \frac{1}{2.9}\cdot 0.0444 \\ \frac{1}{2.9}\cdot 0.9639 & \frac{1}{2.9}\cdot 0.8704 & \frac{1}{2.9}\cdot 0.9229 & \frac{1}{2.9}\cdot 0.0199 & \frac{1}{2.9}\cdot 0.0784 & \frac{1}{2.9}\cdot 0.0444 \\ \frac{1}{2.9}\cdot 0.9639 & \frac{1}{2.9}\cdot 0.8704 & \frac{1}{2.9}\cdot 0.9229 & \frac{1}{2.9}\cdot 0.0199 & \frac{1}{2.9}\cdot 0.0784 & \frac{1}{2.9}\cdot 0.0444 \\ \frac{1}{2.9}\cdot 0.9639 & \frac{1}{2.9}\cdot 0.8704 & \frac{1}{2.9}\cdot 0.9229 & \frac{1}{2.9}\cdot 0.0199 & \frac{1}{2.9}\cdot 0.0784 & \frac{1}{2.9}\cdot 0.0444 \\ \frac{1}{2.9}\cdot 0.9639 & \frac{1}{2.9}\cdot 0.8704 & \frac{1}{2.9}\cdot 0.9229 & \frac{1}{2.9}\cdot 0.0199 & \frac{1}{2.9}\cdot 0.0784 & \frac{1}{2.9}\cdot 0.0444 \\ \frac{1}{2.9}\cdot 0.9639 & \frac{1}{2.9}\cdot 0.8704 & \frac{1}{2.9}\cdot 0.9229 & \frac{1}{2.9}\cdot 0.0199 & \frac{1}{2.9}\cdot 0.0784 & \frac{1}{2.9}\cdot 0.0444 \end{bmatrix} =$$

$$= \begin{bmatrix} 0.3323 & 0{,}3001 & 0.3182 & 0.0068 & 0.0270 & 0.0153 \\ 0.3323 & 0{,}3001 & 0.3182 & 0.0068 & 0.0270 & 0.0153 \\ 0.3323 & 0{,}3001 & 0.3182 & 0.0068 & 0.0270 & 0.0153 \\ 0.3323 & 0{,}3001 & 0.3182 & 0.0068 & 0.0270 & 0.0153 \\ 0.3323 & 0{,}3001 & 0.3182 & 0.0068 & 0.0270 & 0.0153 \\ 0.3323 & 0{,}3001 & 0.3182 & 0.0068 & 0.0270 & 0.0153 \end{bmatrix}$$

Отже, побудували матрицю перехідних ймовірностей P для схеми з чьотирьох реле, кожна з яких має три стани.

Зауваження. В прикладі 2.10 у матриці перехідних ймовірностей P сума елементів рядка дорівнює 0.9997, що близько до одиниці. Це пов'язано з похибкою обчислень чисел з плаваючою точкою та округлень до четвертого знаку після коми.

Алгоритм 2.3. Матриця P при керуючому впливі

Вхід: Матриця перехідних ймовірностей P, сигнал $s \in E_+$

Вихід: Матриця перехідних ймовірностей під дією керуючого впливу $\mathbb{C}(s) \circ P$

1. З матриці перехідних ймовірностей P виділяємо розподіл кожного стану $\vec{p}_i = (p_{i0}, p_{i1}, \ldots, p_{i(2\cdot n-1)})$, $i \in E$, $n = |E_+|$, $\sum\limits_{j=0}^{2\cdot n-1} p_{ij} = 1$

2. Подіяти керуючим впливом $C(\cdot, s)$ на підпростір E_+ матриці P таким

чином

$$\begin{bmatrix} C(\vec{p}_0,s) & \dots & C(\vec{p}_0,s) & w_0\cdot\alpha_0 & w_1\cdot\alpha_1 & \dots & w_{n-1}\cdot\alpha_{n-1} \\ C(\vec{p}_1,s) & \dots & C(\vec{p}_1,s) & w_0\cdot\alpha_0 & w_1\cdot\alpha_1 & \dots & w_{n-1}\cdot\alpha_{n-1} \\ \dots & \dots & \dots & \dots & \dots & \dots & \\ C(\vec{p}_{2\cdot n-1},s) & \dots & C(\vec{p}_{2\cdot n-1},s) & w_0\cdot\alpha_0 & w_1\cdot\alpha_1 & \dots & w_{n-1}\cdot\alpha_{n-1} \end{bmatrix}$$

Приклад 2.11. Наведемо приклад використання алгоритму 2.3. Нехай є матриця перехідних ймовірностей з прикладу 2.9. В формальному вигляді матриця P під дією керуючого впливу $C(\cdot,s)$ буде виглядати так:

$$\mathbb{C}(s)\circ P= \begin{bmatrix} \mathbf{1}(\{s=0\})\cdot\sum\limits_{j=0}^{1}p_{0j} & \mathbf{1}(\{s=1\})\cdot\sum\limits_{j=0}^{1}p_{0j} & 0.05 & 0.1 \\ \mathbf{1}(\{s=0\})\cdot\sum\limits_{j=0}^{1}p_{1j} & \mathbf{1}(\{s=1\})\cdot\sum\limits_{j=0}^{1}p_{1j} & 0.05 & 0.1 \\ \mathbf{1}(\{s=0\})\cdot\sum\limits_{j=0}^{1}p_{2j} & \mathbf{1}(\{s=1\})\cdot\sum\limits_{j=0}^{1}p_{2j} & 0.05 & 0.1 \\ \mathbf{1}(\{s=0\})\cdot\sum\limits_{j=0}^{1}p_{3j} & \mathbf{1}(\{s=1\})\cdot\sum\limits_{j=0}^{1}p_{3j} & 0.05 & 0.1 \end{bmatrix}$$

Якщо приходить сигнал на перехід до стану $s=0$, то матриця P під дією керуючого впливу буде виглядати так:

$$\mathbb{C}(0)\circ P= \begin{bmatrix} 1\cdot(0.45+0.4) & 0\cdot(0.45+0.4) & 0.05 & 0.1 \\ 1\cdot(0.45+0.4) & 0\cdot(0.45+0.4) & 0.05 & 0.1 \\ 1\cdot(0.45+0.4) & 0\cdot(0.45+0.4) & 0.05 & 0.1 \\ 1\cdot(0.45+0.4) & 0\cdot(0.45+0.4) & 0.05 & 0.1 \end{bmatrix} =$$

$$= \begin{bmatrix} 0.85 & 0 & 0.05 & 0.1 \\ 0.85 & 0 & 0.05 & 0.1 \\ 0.85 & 0 & 0.05 & 0.1 \\ 0.85 & 0 & 0.05 & 0.1 \end{bmatrix}$$

Якщо приходить сигнал на перехід до стану $s = 1$, то матриця P під дією керуючого впливу буде виглядати так:

$$\mathbb{C}(1) \circ P = \begin{bmatrix} 0 \cdot (0.45 + 0.4) & 1 \cdot (0.45 + 0.4) & 0.05 & 0.1 \\ 0 \cdot (0.45 + 0.4) & 1 \cdot (0.45 + 0.4) & 0.05 & 0.1 \\ 0 \cdot (0.45 + 0.4) & 1 \cdot (0.45 + 0.4) & 0.05 & 0.1 \\ 0 \cdot (0.45 + 0.4) & 1 \cdot (0.45 + 0.4) & 0.05 & 0.1 \end{bmatrix} =$$

$$= \begin{bmatrix} 0 & 0.85 & 0.05 & 0.1 \\ 0 & 0.85 & 0.05 & 0.1 \\ 0 & 0.85 & 0.05 & 0.1 \\ 0 & 0.85 & 0.05 & 0.1 \end{bmatrix}$$

Алгоритм 2.4. Перехід з стану $x_k \in E$ в стан $x_{k+1} \in E$

Вхід: Матриця перехідних ймовірностей P, стан $x_k \in E$

Вихід: Стан $x_{k+1} \in E$

1. Виділити з матриці перехідних ймовірностей P розподіл $\vec{p_{x_k}}$.

2. Згідно розподілу $\vec{p_{x_k}}$ заданий на множині E згенерувати наступний стан x_{k+1}.

Алгоритм 2.5. Перехід з стану $x_k \in E$ згідно керуючого впливу $C(\cdot,s)$,$s \in E_+$ в стан $x_{k+1} \in E$

Вхід: Матриця перехідних ймовірностей $\mathbb{C}(s) \circ P$, стан $x_k \in E$

Вихід: Стан $x_{k+1} \in E$

1. Виділити з матриці перехідних ймовірностей $\mathbb{C}(s) \circ P$ розподіл $\vec{p_{x_k}}$.

2. Згідно розподілу $\vec{p_{x_k}}$ заданий на множині E згенерувати наступний стан x_{k+1}.

Висновки до розділу 2

Отже, у данному розділі було вирішено задачу з побудови моделі децентралізованого криптографічного модуля. А саме було переформульовано модель ненадійного реле з двома станами і сформульована модель небезпечних елементів, що являє собою децентралізований криптографічний модуль, який може мати більше ніж два стани. Був введений керуючий вплив, який робить систему більш безпечною, а також переформульована схема з ненадійних компонентів, що в свою чергу збільшує оцінку безпеки функціонування криптографічного модуля. В останній секції було введено алгоритми, які дозволяють працювати з основними результатами побудованої формальної моделі децентралізованого криптографічного модуля.

3 РЕАЛІЗАЦІЯ ДЕЦЕНТРАЛІЗОВАНОГО КРИПТОГРАФІЧНОГО МОДУЛЯ

В данному розділі буде описано особливості реалізації криптографічного модуля в вигляді децентралізованого додатку, що побудований на базі блокчейну Ethereum. Дана реалізація є прикладом децентралізованого додатку в реальному блокчейн додатку, що відповідає показникам безпеки встановленими стандартом FIPS PUB 140-2.

3.1 Опис децентралізованого криптографічного модуля відносно стандарту FIPS PUB 140-2

За стандартом FIPS PUB 140-2 криптографічний модуль повинен бути задокументований і перелічувати всі компоненти, що він використовує. Опишемо смарт контракт, який буде розроблений як децентралізований криптографічний модуль на мові програмування Solidity.

3.1.1 Ролі

Блокчейн Ethereum не накладає ніякої цензури на користувачів і не вимагає від них ніякої ідентифікації. Тож зробимо тільки дві ролі відповідно до цієї ідеї:

1) користувач

2) власник

Користувачем може бути тот, хто має аккаунт в блокчейні Ethereum. Користувач може використовувати всі функції, окрім функції позначеним модифікатором onlyOwner.

Власником може бути тільки тот аккаунт, що додав смарт контракт у блокчейн. Також власник може передавати своє володіння іншим аккаунтам або позбавити смарт контракт власника, передавши володіння смарт контрактом аккаунту з нульовою адресою. Власник може викликати всі функції смарт контракту.

Було використано бібліотеку OpenZeppelin, що вже реалізувала функціонал з ролями і проінтегровано до смарт контракту.

3.1.2 Порти і інтерфеси

Мережа блокчейна Ethereum з фізичної точки зору є множиною електронно обчислювальних пристроїв, на яких запущено програмне забезпечення, що являється вузолом мережі Ethereum. В дані вузли інформація йде від інших нод через фізичні канали зв'язку. Тож інформація потрапляє та відправляється на фізичні порти того обчислювального серидовища, де запущена нода.

Логічними портами мережі блокчейна Ethereum виступають об'єкти операційної системи, які слухають вхідні дані по назначеній операційній системі портом.

Вхідним інтерфейсом в смарт контрактах виступають транзакії, які створюють виклик функції смарт контрактів.

Вихідними інтерфейсом є чек транзакції, що свідчить про результат обробки транзакції.

3.1.3 Сервіси

Криптографічний модуль повинен предоставляти сервіси для обробки інформації і забезпечення їх корректної обробки.

Шифрування та Розшифрування

Смарт-контракт повинен виконувати функцію посредника між двома аккаунтами і забезпечити цілісність, конфіденційність і доступність. В Ethereum цілісність реалізована в транзакції як повідомлення від аккаунту та підпис цього аккаунту на повідомлення, а доступність полягає у тому, що будь яку транзакцію та будь який блок можна переглянути у будь який час, але для цього потрібно знати 20 байтну адресу. Тож, треба в логіці смарт-контракту реалізувати конфіденційність аккаунту.

Але в екосистемі Ethereum всі вхідні та вихідні дані і код смарт-контракту відкриті, тому відправлення повідомлення до смарт-контракту, що містить в собі конфіденційні дані, є порушенням конфіденційності. Тим самим реалізація перетворень конфіденційних данних на смарт-контракті є марною операцією. Щоб вирішити дану проблему було побудовано протокол, що забезпечують конфіденційність в відкритих смарт-контрактах Ethereum.

Протокол забезпечення конфіденційності базується на використанні результатів асиметричної криптографії. Суть данного протоколу полягає в тому, що всі перетворення конфіденційних данних виконуются на стороні клієнта, а зашифровані і відкриті дані можуть записуватись у пам'ять смарт контракта.

Учасники протоколу: відправник Боб, ком'ютер Боба, отримувач Аліса, комп'ютер Аліси, смарт контракт.

Протокол 1:

0.Аліса підписується на виклики подій у смарт контракті за її адресою

1.Аліса обирає асиметричну криптосистему та генерує пару відкритого і закритого ключа відповідно до обраної криптосистеми.

2.Комп'ютер Аліси створює транзакцію на запис відкритий ключа Аліси.

3.Боб викликає функцію шифрування, в яку передає повідомлення та

адресу Аліси.

4.Комп'ютер Боба запитує у смарт контракта відкритий ключ Аліси.

5.Комп'ютер Боба шифррує повідомлення Боба.

6.Комп'ютер Боба створює транзакцію, в яку записує шифрованого повідомлення та адресу Аліси.

7.Смарт контракт викликає подію, чим спонукає комп'ютер Аліси відреагувати на подію.

8.Комп'ютер Аліси викликає функцію розшифровки від параметрів, що записані в події.

9.Комп'ютер Аліси нотифікує Алісу, що їй прийшло повідомлення.

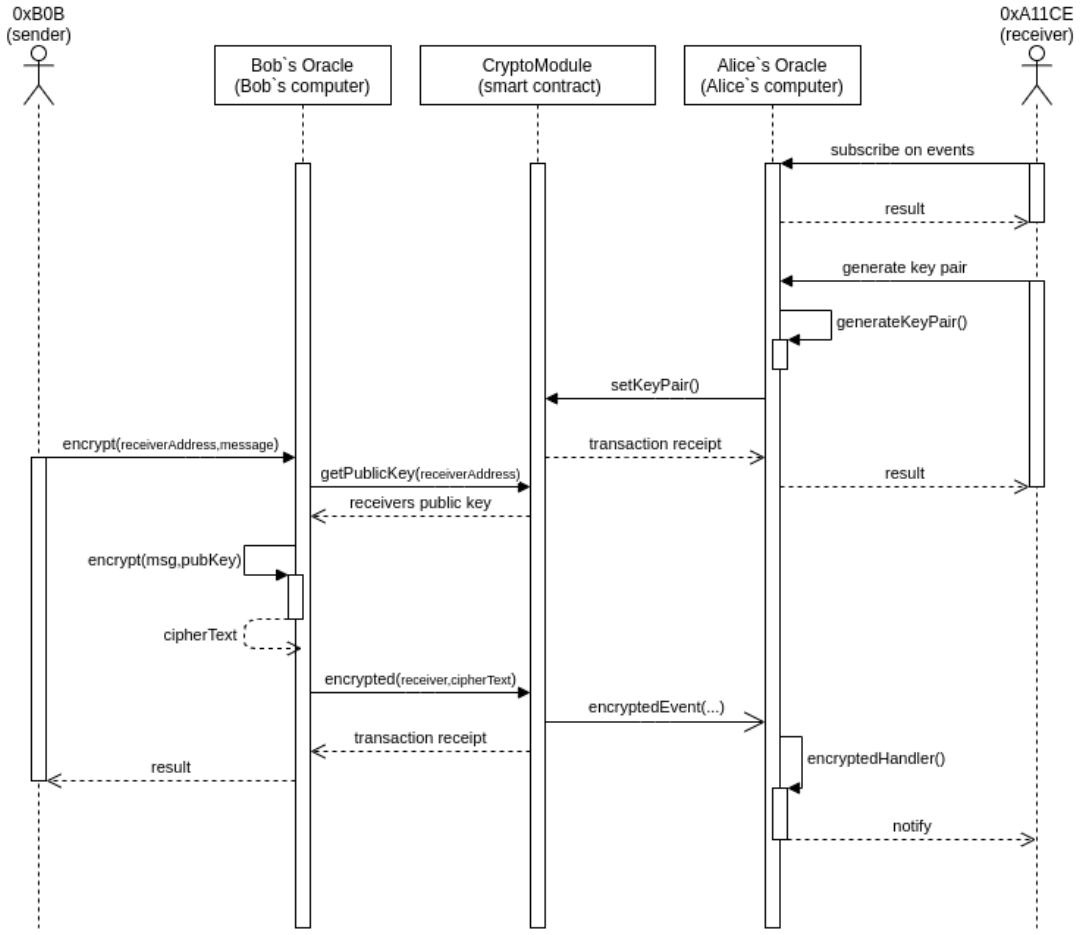


Рисунок 3.1 – Схематичне зображення протоколу

Виникає питання: Скільки треба заплатити, щоб використовувати данний протокол? В блокчейні Ethereum транзакція має вартість, якщо

ця транзакція змінює стан смарт контракту. Лише 2 і 6 кроки протоколу змінюють стан смарт контракту. Інакше кажучи, на кроках 2 і 6 потрібно заплатити за виконання транзакції. За всі інші кроки сплачувати криптовалюту не треба. Всі криптографічні перетворення виконуються на комп'ютерах користувачів, а вони для користувачів майже безкоштовні.

Генерація псевдовипадкових чисел

В смарт контракті можна побудувати алгоритм генерації псевдовипадкових чисел за допомогою хеш функції keccak, часової позначки та зберігаючого стану генератора. Але повної випадковості досягти майже не можливо, оскільки всі вхідні парамети та алгоритм генерації відкриті і можуть бути переглянуті ким і чим завгодно. Оскільки данний генератор змінює стан контракту, а саме стан генератора, то за використання данного функціоналу потрібно платити.

Вираховується псевдовипадкове число за такими параметрами:

1) $seed$ - вхідний параметр;

2) s_i - стан генератора в момент $i \in \mathbb{N}$;

3) $timestamp$ - часова позначка;

s_0 задається при розгортанні смарт контракту. Випадкове число u обчислюється за данною формулою:

$$u = keccak(timestamp \oplus s_i)$$

де

$$s_i = (s_{i-1} + seed) \oplus keccak(seed), i > 1$$

Користувацька операція

Функція, що визначається самим користувачем-відправником. Визначення проходить через вхідні параметри і узгоджується між

користувачем-отримувачем і користувачем-відправником.

Приклад 3.1. Користувач-відправних хоче свій генератор випадкових числел за його алгоритмом генерування. Тому він викликає функцію "операція", в якій вказує адресу отримувача, назву операції, программний код операції, вхідні параметри до программи. І ця програма запускається на комп'ютері користувача.

3.1.4 Модель кінцевих станів

Дана модель була побудована в розділі 2. Можна лише добавити те, що мережа Ethereum складається з великої кількості вузлів, що задовільняє рівню безпеки функціонування відносно побудованої моделі.

3.1.5 Обчислювальне середовище

Смарт контракти виконують свій функціонал на нодах майнерів(валідаторів). Зазвичай нодами є теримінальні або графічні програми під операційну систему, яку розробили розробники блокчейну. Наприклад самим типовим нодом є клієнт geth. Майнери(валідатори) запускають на своїй операційній системі ноду і данний софт здійснює обробку транзакцій та формування блоку з транзаціями, а також синхронизацію з іншими нодами у мережі. Транзакції обираються із пулу транзакцій, які відправили аккаунти. А ось яким чином обираються ці транзакції залежить від реалізації софту, яку створили розробники. В багатьох реалізаціях обираются транзакції з більшою кількістю газу.

У користувачів взаємодія може відбуватись на будь-яких пристроях, де є доступ в інтернет та додаток для взаємодії з блокчейном. Користувачі, використовуючи додаток, створюють транзакції для переводу або виклику функцій смарт контракту і надсилають їх у пул з транзакціями. Ці додатки використовують бібліотеку web3 для взаємодії з

блокчейном Ethereum. Наприклад такий додаток може бути в додатком в смартфоні, веб-додатоком, графічною або термінальною програмою в операційній системі.

3.1.6 Управління криптографічними ключами

Керування ключами повністю перекладається на плечі користувачів. Вони самі вирішують де зберігати і яким способом. Смарт контракт, як криптографічний модуль, лише забезпечує зберігання відкритих данних та перезаписування відкритих ключів за бажанням аккаунтів власників цих ключів.

3.2 Розробка децентралізованого криптографічного модуля у вигляді смарт контракту

В попередній секції було сформовано що вимагається від децентралізованого криптографічного модуля та який функціонал повинен бути реалізований у смарт контракті. У даній секції буде описано кроки розробки та розгортання смарт контракту.

3.2.1 Середовище розробки

Для розробки смарт контракту було використано интегровану середовище розробки RemixIDE [7]. Дану середу розробки не треба встановлювати, оскільки воно є веб-додатком, тож треба мати будь-який браузер для взаємодії з цим середовищем розробки. В RemixIDE зручно писати смарт контракти та розгортати ці контракти у як у приватні, так і у публічні мережі. Дане середовище розробки має вбудований аналізатор, який повідомить розробника про синтаксичні помилки та описки.

RemixIDE має такі складові необхідні для розробки контракту:

1) Локальна файлова система (FILE EXPLORERS)
2) Компілятор (SOLIDITY COMPILER)
3) Розгортання (DEPLOY & RUN TRANSACTIONS)

3.2.2 Написання смарт контракту

У вкладці FILE EXPLORERS Було створено файл з назвою CryptoModule.sol та реалізовувались ідеї запропонованими у описі децентралізованого криптографічного модуля 3.1. Написаний код смарт контракту викладений в додатку А.1.

3.2.3 Компіляція смарт контракту

У вкладці SOLIDITY COMPILER було обрано останню версію компілятору 0.8.4, що на дату написання данної роботи була актуальна. Слідом було натиснуто кнопку компіляції смарт контракту.

3.2.4 Розгортання смарт контракту у публічній мережі

Для розгортання контракту в публічній мережі потрібно мати гаманець, в якому є аккаунти у мережі Ethereum. Таким гаманцем було обрано Metamask, в якому було створено аккаунт. Цей гаманець автоматично під'єднується до обраної мережі. Такою обраною мережею може бути основна мережа Ethereum або тестові мережі Ropsten,Kovan,Rinkeby і так далі. Було обрано тестову мережу Ropsten. Щоб розгорнути смарт контракт треба мати якусь кількість криптовалюти для прийняття транзакції про додання написанного смарт контракту до блокчейну тестової мережі. Для поповнення акаунту криптовалютою використано кран [8].

Далі в середовищі розробки RemixIDE в вкладці DEPLOY & RUN

TRANSACTIONS було обрано в випадаючому списку ENVIRONMENT "Injected Web3". За замовченням обрано перший по списку аккаунт з якого буде відбуватись розгорнення, а також GAS LIMIT та VALUE не були змінені. Після в випадаючому списку CONTRACT обрано файл CryptoModule.sol. Перед натисненням кнопки Deploy, було введено в поле, що знаходиться справа від кнопки, початковий стан генератора 4919. Дане число було обрано випадково. Після натиснення кнопки Deploy, Metamask відкриє вікно де запросить користувача обрати скільки він готовий заплатити криптовалюти та підтвердити транзакцію [10] . Через деякий час створився блок, де знаходиться ця транзація.[12]

3.2.5 Валідація смарт контракту

Смарт контракт з нашим децентралізованим криптографічним модулем вже розгорнут в тестовій мережі. Зазвичай люди користуються експлорерами для перегляду всього ланцюга транзакцій і блоків. Одним з таким сервісом є etherscan [9]. До ньго ми зверталися у субсекції 3.2.4, щоб подивитись на параметри створеної нами транзакції.

Для валідації розгорнутого смарт контракту було здійснено перехід до експлореру смарт контракту [11]. Далі відкрили вкладку з назвою Contract та натиснуто Verify and Publish. Був здійснений перехід до інтуітивно зрозумілого інтерфейсу, де потрібно було ввести ієрархію проекту, версію компілятора, якою було здійнено компіляцію написанного сматр контракту, та обрати ліцензію. Ієрархія було обрано Solidity (Single File), версія компілятора була 0.8.4 та ліцензія MIT. Після натиснення кнопки Continue, відбулось запрошення до вводу коду смарт контракту. Код смарт контракта було скопійовано та вставлено до поля запрошення. Далі треба було пройти перевірку через reCAPTCHA, що було здійснено. Далі було натиснуто синю кнопку Verify and Publish та з'явився інтерактивний інтерфейс, що показував процес перевірки. Після деякого часу цей інтерфейс показав успішну валідацію. І тепер на експлорері

смарт контракту видно код та можна взаємодіяти зі смарт контрактом через гаманець Metamask.

Висновки до розділу 3

Отже, в даному розділі було виконано останнє завдання дослідження з реалізації децентралізованого криптографічного модуля у вигляді смарт контракту. Децентралізований криптографічний модуль у вигляді смарт контракту був описан за стандартом FIPS 140-2. А саме було описано:

1) Ролі користувача та власника. Було введено обмеження на виконання функцій що не повинен викликати звичайний користувач.

2) Порти та інтерфейси, які існують у обчислювальному середовищі майнера(валідатора).

3) Сервіси. Було запропоновано функціонал записування відкритих ключів аккаунтів, записування до блокчейну шифрованих повідомлень з нотифікуванням подій про запис повідомлень, генерація псевдовипадкових чисел, та користувацька операція, яку користувач визначає самостійно.

4) Коротко про модель кінцевих станів, що була описана у розділі 2.

5) Обчислювальне середовище. Вхідні транзакції оброблюються майнерами на спеціальному софті, який розроблений розробниками блокчейну. А користувачі користуються обчислювальним середовищем у вигляді додатків, які можуть бути які завгодно.

6) Управління криптографічними ключами, яке повністю здійснюються користувачем системи.

Також було покроково описано процес виконаної роботи в RemixIDE для написання, компіляції, розгортання смарт контракту в публічну тестову мережу Ropsten та процес валідації смарт контракту на etherscan.

ВИСНОВКИ

У ході даної роботи перш за все було вирішено перша і друга задача дослідження. А саме був провелений аналіз опублікованих джерел за тематикою побудови надійних систем із ненадійних елементів як основа для моделі децентралізованого криптографічного модуля, стандарт криптографічних модулів FIPS PUB 140-2 та технологію блокчейн Ethereum. Модель ненадійних реле Шенона було взято як сиру основу для моделі децентралізованого криптографічного модуля, оскільки суть даної моделі нагадує розподілену і в одночас децентралізовану систему. Стандарт FIPS PUB 140-2 був проаналізований для того щоб знати як описувати криптографічний модуль та які показники безпеки повинні бути. На останок було проаналізовано основу децентралізованих додатків на блокчейні Ethereum, для того щоб коректно впровадили децентралізований додаток як децентралізований криптографічний модуль у вигляді смарт контракту.

У задачі побудови моделі відбулось переформулювання та узагальнення моделі ненадійних реле Шеннона. В узагальнену модель було введено керуючий вплив, що відповідає подачі команд у криптографічний модуль за стандартом FIPS PUB 140-2. На основі узагальненої моделі було побудовано алгоримти, які дозволяють по оцінкам надійності до стану ненадійних елементів перейти до матриці перехідних ймовірностей, за якою можна оцінювати загальну оцінку надійності відносно кожного стану.

Для подальших досліджень до моделі децентралізованого криптографічного модуля може бути знайдено більш практична модель, яка враховує не тільки фіксовану надійність як оцінку ймовірності правильної і не правильної роботи, а й надійність при атаках на децентразізований криптографічний модуль. Було б ще не погано дослідити надійність кожного ненадійного елемента як функцію від часу

та дослідтити надійність системи з ненадійних елементів як функцію від часу.

При вирішенні задачі реалізації було описано децентралізований криптографічний модуль та реалізовано на цьому описі смарт контракт. У смарт контракті реалізовано протокол передачі зашифрованого повідомлення, який використовує особливості платформи Ethereum. Також данний протокол разраховується на двох користувачів: на відправника і отримувача. Головною особливістю є те, що даний протокол можна застосовувати у відкритих блокчейнах, де інформація про транзакції не приховується. Із особливостей можна виділити те, що за дану заємодію по обміну повідомленням треба заплатити певну кількість криптовалюти для виконнання транзакції. Ще було здійнено розгортання смарт контракту в тестовій мережі, де можна скористатись функціоналом смарт контракту.

Для подальших досліджень реалізації децентралізованих криптографічних модулів можна вигадати оптимізацію коду смарт контракту для зменшення вартості транзакцій за рахунок менш узагальнених процедур. Також можна реалізувати сприятливий до людей додаток, що взаємодіє із реалізованим і розгорнутим смарт контрактом. Оскільки розроблений криптомодуль розрахований для користувачів, то було б вигадати деякий функціонал, що міг би викликатись з інших смарт контрактів, що потенційно збільшить можливість використання функціоналу криптографічного модуля.

ПЕРЕЛІК ПОСИЛАНЬ

1. Шеннон К. Работы по теории информации и кибернетике / Шеннон К. - Москва: ИЗДАТЕЛЬСТВО ИНОСТРАННОЙ ЛИТЕРАТУРЫ, 1963. - С. 114-154

2. А. Н. Колмогоров Основные понятия теории вероятностей / Альберт Николаевич Колмогоров ОСНОВНЫЕ ПОНЯТНИЯ ТЕОРИИ ВЕРОЯТНОСТЕЙ (Серия:"Теория вероятностей и математическая статистика"), 1974г.

3. FIPS PUB 140-2 [Електронний ресурс]. — Режим доступу: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf.

4. Siraj Raval Decentralized Applications / Siraj Raval - O'Reilly Media, Inc., 2016.

5. Ethereum docs [Електронний ресурс]. — Режим доступу: https://ethereum.org/en/developers/docs/.

6. Solidity - solidity docs v0.8.4 [Електронний ресурс]. — Режим доступу: https://docs.soliditylang.org/en/v0.8.4/

7. RemixIDE [Електронний ресурс]. — Режим доступу: https://remix.ethereum.org

8. Кран для мережі Ropsten [Електронний ресурс]. — Режим доступу: https://faucet.ropsten.be

9. Експлорер мережі Ethereum [Електронний ресурс]. — Режим доступу: https://etherscan.io

10. Посилання на транзакцію зі розгортання смарт контракту [Електронний ресурс]. — Режим доступу: https://ropsten.etherscan.io/tx/0x4d6546fcf83df7daa37403bd090a94ac17191494bd7b6c983414f850f69e4aa9

11. Посилання на контракт [Електронний ресурс]. — Режим доступу: https://ropsten.etherscan.io/address/0x3e42aa3fb56cfd6b4c1f64038fce5b158bb5f078

12. Посилання на блок [Електронний ресурс]. — Режим доступу:

https://ropsten.etherscan.io/block/10146876

ДОДАТОК А ТЕКСТИ ПРОГРАМ

А.1 Децентралізований криптографічний модуль на мові програмування Solidity

CryptoModule.sol:

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.8.0;

/*
 * @dev Provides information about the current execution context, including the
 * sender of the transaction and its data. While these are generally available
 * via msg.sender and msg.data, they should not be accessed in such a direct
 * manner, since when dealing with GSN meta-transactions the account sending and
 * paying for execution may not be the actual sender (as far as an application
 * is concerned).
 *
 * This contract is only required for intermediate, library-like contracts.
 */
abstract contract Context {
    function _msgSender() internal view virtual returns (address) {
        return msg.sender;
    }

    function _msgData() internal view virtual returns (bytes memory) {
        this; // silence state mutability warning without generating bytecode - see https://
            github.com/ethereum/solidity/issues/2691
        return msg.data;
    }
}

/**
 * @dev Contract module which provides a basic access control mechanism, where
 * there is an account (an owner) that can be granted exclusive access to
 * specific functions.
 *
 * By default, the owner account will be the one that deploys the contract. This
 * can later be changed with {transferOwnership}.
 *
 * This module is used through inheritance. It will make available the modifier
 * `onlyOwner`, which can be applied to your functions to restrict their use to
 * the owner.
```

```
36  */
37  abstract contract Ownable is Context {
38      address private _owner;
39
40      event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);
41
42      /**
43       * @dev Initializes the contract setting the deployer as the initial owner.
44       */
45      constructor () {
46          address msgSender = _msgSender();
47          _owner = msgSender;
48          emit OwnershipTransferred(address(0), msgSender);
49      }
50
51      /**
52       * @dev Returns the address of the current owner.
53       */
54      function owner() public view returns (address) {
55          return _owner;
56      }
57
58      /**
59       * @dev Throws if called by any account other than the owner.
60       */
61      modifier onlyOwner() {
62          require(_owner == _msgSender(), "Ownable: caller is not the owner");
63          _;
64      }
65
66      /**
67       * @dev Leaves the contract without owner. It will not be possible to call
68       * `onlyOwner` functions anymore. Can only be called by the current owner.
69       *
70       * NOTE: Renouncing ownership will leave the contract without an owner,
71       * thereby removing any functionality that is only available to the owner.
72       */
73      function renounceOwnership() public virtual onlyOwner {
74          emit OwnershipTransferred(_owner, address(0));
75          _owner = address(0);
76      }
77
78      /**
79       * @dev Transfers ownership of the contract to a new account (`newOwner`).
80       * Can only be called by the current owner.
```

```
81      */
82      function transferOwnership(address newOwner) public virtual onlyOwner {
83          require(newOwner != address(0), "Ownable: new owner is the zero address");
84          emit OwnershipTransferred(_owner, newOwner);
85          _owner = newOwner;
86      }
87  }
88
89
90  contract CryptoModule is Ownable{
91
92      struct PublicKey{
93          string ALGORITHM_NAME;
94          string KEY;
95      }
96
97      function generateRandomNumber(uint256 seed) public returns(uint256){
98          generatorState=(generatorState+seed)^uint256(keccak256(abi.encodePacked(seed)));
99          return uint256(keccak256(abi.encodePacked(block.timestamp^generatorState)));
100     }
101
102     mapping(address => PublicKey) public publicKeys;// storing public keys
103
104     uint256 private generatorState;
105
106     constructor(uint256 initialGeneratorState){
107         generatorState=initialGeneratorState;
108     }
109
110     event Encrypted(address indexed sender,
111                     address indexed receiver,
112                     string cipherText);
113     event Operation(address indexed sender,
114                     address indexed receiver,
115                     string operationName,
116                     string operationCode,
117                     string parametrs);
118
119     modifier checkPublicKey(string memory algorithm_name,
120                             string memory key){
121         require(bytes(algorithm_name).length>0,"Algorithm name is empty!");
122         require(bytes(key).length>0,"Public key is empty!");
123         _;
124     }
125
```

```
126     /*@dev
127     *@param key is json that stores open keys parametrs.
128     *       for example RSA public key: key="{e:0x1000001,n:0x123456789ABCDE}"
129     */
130     function setPublicKey(  string memory algorithm_name,
131                             string memory key)
132     public checkPublicKey(algorithm_name,key) {
133         publicKeys[msg.sender].ALGORITHM_NAME=algorithm_name;
134         publicKeys[msg.sender].KEY=key;
135     }
136
137     function getPublicKey(address receiver)
138     public view returns(string memory algorithm_name,
139                         string memory key){
140         return (publicKeys[receiver].ALGORITHM_NAME,
141                 publicKeys[receiver].KEY);
142     }
143
144     function encrypted( address receiverAddress,
145                         string memory cipherText)
146     public {
147         emit Encrypted(msg.sender,receiverAddress,cipherText);
148     }
149
150     /*@dev operation is remote procedure,that handles by receiver.
151     *@param operationName  is the name of operation.
152     *@param operationCode is the raw code,that should be executed by receiver
153     *@param parametrs is input to program
154     */
155     function operation( address receiver,
156                         string memory operationName,
157                         string memory operationCode,
158                         string memory parametrs)
159     public {
160         require(bytes(operationName).length>0,"Operation name should not be empty!");
161         emit Operation(msg.sender,receiver,operationName,operationCode,parametrs);
162     }
163
164 }
```