УДК 004.056.5 **DOI:10.30837/rt.2022.4.211.02**

С.О. КАНДІЙ

АНАЛІЗ БЕЗПЕКИ ДСТУ 8961:2019 У МОДЕЛІ ВИПАДКОВОГО ОРАКУЛА

Вступ

ДСТУ 8961:2019 є діючим українським стандартом постквантового асиметричного шифрування та інкапсуляції ключів [1]. Стандарт грунтується на ANSI X9.98 [2], проте має ряд відмінностей. У літературі відомо не так багато робіт, що присвячені безпеці ДСТУ 8961:2019 та ANSI X9.98. Більшість досліджень присвячено або генерації загальносистемних параметрів [3], або атакам на основі редукції решіток [4].

У той же час, для механізмів інкапсуляції ключів, що є фіналістами конкурсу NIST PQC [5], у літературі можливо знайти всесторонній аналіз безпеки як для класичних атак, так і для атак з використанням квантових комп'ютерів [6, 7]. Безсумнівно, аналіз конкретних значень безпеки для наборів загальносистемних параметрів є важливим. Проте, такі оцінки мають сенс тільки у випадку, якщо криптоаналітик не може знайти "обхідні шляхи", якщо кожна атака зводиться до певного набору добре досліджених складних у теоретико-числовому сенсі проблем.

Зазвичай, для практичних застосувань механізми інкапсуляції ключів (KEM) вважаються безпечними [6, 7], якщо складність будь-якої атаки з адаптивно підібраними шифротекстами є занадто великою для практичної реалізації. Якщо KEM містить доказ того, що він є безпечним у зазначеному вище сенсі за умови складності певного невеликого набору теоретико-числових проблем при конкретних значеннях загальносистемних параметрів, то кажуть, що він є безпечним у стандартній моделі [8].

На жаль, докази у стандартній моделі часто неможливо отримати для реальних KEM. Причиною цього є використання криптографічних геш-функцій [8]. Часто відсутні інструменти для того, щоб врахувати вплив алгебраїчної структури та властивостей геш-функції на безпеку перетворень. Тому, на практиці поширеною модифікацією є модель випадкового оракула [9], у якій усі геш-функції замінюються на випадкові оракули – ідеалізовані геш-функції, що не мають внутрішньої структури.

У роботі викладено перші результати спроби комплексного аналізу безпеки ДСТУ 8961:2019 у моделі випадкового оракула. Представлено докази IND-CCA2 безпеки схеми асиметричного шифрування та інкапсуляції ключів, що описані в ДСТУ 8961:2019.

1. Структура механізмів інкапсуляції ключів

З формальної точки зору, механізм інкапсуляції ключів є трійкою алгоритмів $(Gen, Encaps, Decaps)$ [7, 8], де:

- $Gen: 1^{\lambda} \to (pk, sk)$ – поліноміальний ймовірнісний алгоритм генерації ключової пари. Приймає параметр безпеки 1^{λ} та повертає ключову пару (pk, sk).
- $Encaps: pk \to (K, C)$ – поліноміальний ймовірнісний алгоритм інкапсуляції ключа. Приймає публічний ключ pk та повертає випадковий ключ K та його інкапсуляцію C.
- $Decaps: (sk, C) \to \{K, \perp\}$ – детермінований поліноміальний алгоритм декапсуляції ключа. Приймає секретний ключ sk та інкапсуляцію ключа C і повертає ключ K у разі вдалої декапсуляції та символ помилки $\perp$ – у разі виникнення помилок.

У класичному випадку [10] від механізмів інкапсуляції ключів вимагається властивість коректності:

$$\Pr[Decaps(sk, C) = k \mid (pk, sk) \leftarrow Gen(1^{\lambda}); (K, C) \leftarrow Encaps(pk)] = 1. \quad (1)$$

Згодом вимога коректності була узагальнена на випадок [11], коли алгоритм декапсуляції у незначній кількості випадків містить помилки декапсуляції:

$$\Pr[Decaps(sk,C)=k \mid (pk,sk)\leftarrow Gen(1^{\lambda});(K,C)\leftarrow Encaps(pk)]=negl(\lambda), \quad (2)$$

де $negl(\lambda)$ позначає незначну функцію, що залежить від параметра безпеки λ. Незначна функція – це функція, що зменшується швидше за будь-який поліном. Формальне визначення буде надано у розд. 2.

Сучасні механізми інкапсуляції ключів зазвичай не будуються *ad hoc* [5]. За основу береться деяка схема асиметричного шифрування. Схема асиметричного шифрування є трійкою алгоритмів (Gen, Enc, Dec), де:

- $Gen: 1^{\lambda}\rightarrow(pk,sk)$ – поліноміальний ймовірнісний алгоритм генерації ключової пари. Приймає параметр безпеки 1^{λ} та повертає ключову пару (pk,sk).
- $Enc:(pk,m)\rightarrow C$ – поліноміальний ймовірнісний алгоритм шифрування. Приймає публічний ключ pk та повідомлення m і повертає шифротекст C.
- $Dec:(sk,C)\rightarrow\{m,\perp\}$ – детермінований поліноміальний алгоритм розшифрування. Приймає секретний ключ sk та шифротекст C і повертає повідомлення m у разі вдалої декапсуляції та символ помилки $\perp$ – у разі виникнення помилок.

Від схеми шифрування, аналогічно, вимагається коректність:

$$\Pr[Dec(sk,Enc(pk,m))=m \mid (pk,sk)\leftarrow Gen(1^{\lambda})]=1. \quad (3)$$

Або її послаблений варіант

$$\Pr[Dec(sk,Enc(pk,m))=m \mid (pk,sk)\leftarrow Gen(1^{\lambda})]=negl(\lambda). \quad (4)$$

Для перетворення схеми асиметричного шифрування на механізм інкапсуляції ключів виконується дерандомізація схеми шифрування [12]. Ймовірнісний алгоритм $Enc(pk,m)$ перетворюється на детермінований алгоритм $Enc'(pk,r,m)$, який приймає випадкове значення r. Щоб детермінований алгоритм залишався безпечним, необхідно щоб для будь-якого m виконувалася нерівність

$$|r: Enc'(pk,r,m)=C|\leq\gamma(\lambda)=negl(\lambda). \quad (5)$$

Інакше кажучи, необхідно щоб кількість значень r, за яких при фіксованих pk,m можливо отримати шифротекст C, була незначною відносно параметра безпеки λ. Вимоги безпеки до асиметричної схеми при побудові механізму інкапсуляції ключів є дещо слабшими, ніж при загальному використанні схеми [6, 7, 11, 12]. Зазвичай вимагається лише безпека до атак з адаптивно підібраним повідомленням (зазвичай у моделі IND-CPA. Більш детально розглянуто у розд. 2). Далі до схеми застосовується перетворення типу CPA-to-CCA. Наприклад, перетворення Дента [10] або Фуджісакі – Окамото [11]. Узагальнена схема побудови механізму інкапсуляції ключів представлена на рис. 1. Зауважимо, що, звичайно, існують інші підходи до розробки механізмів інкапсуляції ключів, проте описаний є найбільш поширеним, і аналіз ДСТУ 8961:2019 зручно проводити саме при такій декомпозиції КЕМ.

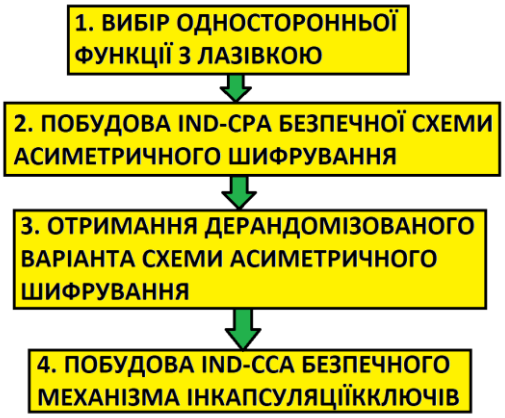

Рис. 1. Методологія побудови механізмів інкапсуляції ключів

2. Моделі безпеки на основі нерозрізнювальності

Найбільш поширеним підходом до оцінки безпеки криптографічних систем є моделі на основі нерозрізнювальності [13]. Такі моделі грунтуються на простій ідеї: якщо супротивник не зможе відрізнити шифротекст від випадкового набору бітів, то він не зможе витягти інформації з шифротексту про відкритий текст. Звичайно, на практиці завжди існує ймовірність, що криптоаналітик зможе витягти інформацію про відкритий текст. Він може хоча б просто вгадати його. Модель вимагає щоб ця ситуація мала незначну ймовірність, яку на практиці можна ігнорувати. Для формального визначення таких незначних ймовірностей використовується нотація незначних функцій [8]. Функція f є незначною, якщо для усіх поліномів p існує константа N_p, для якої виконується

$$f(x) \le \frac{1}{p(x)} \forall x \ge N_p . \tag{6}$$

Позначимо той факт, що функція $f(x)$ є незначною, як $f(x) = negl(x)$.

Звичайно, однієї нерозрізнювальності на практиці недостатньо. Криптоаналітик може методом грубої сили перебрати усі можливі варіанти, або використати певні алгебраїчні властивості для зменшення простору пошуку. Тому, усі оцінки у моделях на основі нерозрізнювальності передбачають, що існує деякий алгоритм $GenParams : \lambda \to params$, який для рівня безпеки λ генерує набір загальносистемних параметрів $params$, за яких складна задача, що лежить в основі криптографічної системи, є складною на практиці у сенсі рівня безпеки λ [11]. Зазвичай, докази у моделях на основі нерозрізнювальності носять асимптотичний характер, проте іноді доказ може використовувати конкретні властивості конкретних загальносистемних параметрів для доведення безпеки системи у обмеженій практичними потребами кількості випадків.

Найбільш поширеними моделями є [13]:

- IND-CPA (нерозрізнювальність для атак на основі підібраного відкритого текста);
- IND-CCA (нерозрізнювальність для атак на основі підібраного шифротекста).

Доказ безпеки у кожній моделі є доказом того, що криптоаналітик може отримати лише незначну перевагу при спробі відрізнити шифротекст від випадкових бітів при наявності доступу до деяких оракулів шифрування/розшифрування (інкапсуляції/декапсуляції). Структура оракулів і порядок доступу залежать від моделі. Для IND-CCA виділяють два випадки – IND-CCA1 та IND-CCA2. У IND-CCA1 доступ до оракулів можливий лише обмежений період часу.

Для деякої криптографічної системи $П$ позначимо перевагу криптоаналітика (супротивника) A у моделях IND-CPA, IND-CCA, IND-CCA2 як $Adv_{A,П}^{IND-ATK}(\lambda), ATK \in \{CPA, CCA1, CCA2\}$. Якщо виконується умова

$$Adv_{A,П}^{IND-ATK}(\lambda) = negl(\lambda), \tag{7}$$

то криптографічна система є безпечною у грі IND-ATK, де $ATK \in \{CPA, CCA1, CCA2\}$ відповідно. За визначенням криптоаналітик має відрізнити гру (експеримент) $Adv_{A,П}^{IND-ATK-0}(\lambda)$, у якій у якості віхідних даних дається справжній шифротекст, та гру $Adv_{A,П}^{IND-ATK-1}(\lambda)$, у якій у на вхід криптоаналітику даються випадкові дані. Відповідно для $ATK \in \{CPA, CCA1, CCA2\}$ маємо наступне визначення переваги:

$$Adv_{A,П}^{IND-ATK}(\lambda) = \left|\Pr\left[Exp_{A,П}^{IND-ATK-1}(\lambda) = 1\right] - \Pr\left[Exp_{A,П}^{IND-ATK-0}(\lambda) = 1\right]\right|. \tag{8}$$

Вміст ігор $Adv_{A,П}^{IND-ATK-0}(\lambda)$ та $Adv_{A,П}^{IND-ATK-1}(\lambda)$ залежить від криптографічної схеми. Для схеми асиметричного шифрування та механізмів інкапсуляції ключів він дещо відрізняється, проте сутність залишається та ж сама.

2.1. Моделі безпеки для асиметричних схем шифрування

Для асиметричної схеми шифрування $П = (Gen, Enc, Dec)$ ігри $Adv_{A,П}^{IND-ATK-0}(\lambda)$ та $Adv_{A,П}^{IND-ATK-1}(\lambda)$ для $ATK \in \{CPA, CCA1, CCA2\}$ зображені на рис. 2. Кожна гра відбувається між випробовувачем та супротивником (криптоаналітиком) [6, 13]. На початку гри випробовувач генерує випадкову ключову пару, два повідомлення m_0, m_1 та надсилає відкритий ключ і повідомлення супротивнику. Супротивник робить запити до оракула O_1 (під оракулом мається на увазі деякий алгоритм, який є "чорним ящиком" для супротивника). Після чого супротивник повідомляє випробовувачу, що готовий отримати завдання. У грі $Adv_{A,П}^{IND-ATK-0}(\lambda)$ випробовувач шифрує повідомлення m_0 та надсилає його шифротекст супротивнику, у грі $Adv_{A,П}^{IND-ATK-1}(\lambda)$ відповідні дії відбуваються для повідомлення m_1. Супротивник робить запити до оракула O_2 та генерує 0 якщо грає у гру $Adv_{A,П}^{IND-ATK-0}(\lambda)$ та 1, якщо грає у гру $Adv_{A,П}^{IND-ATK-1}(\lambda)$.

$$\begin{array}{ccc}
(pk, sk) & \leftarrow & Gen(1^{\lambda}) \\
(m_0, m_1, pk) & \rightarrow & A^{O_1}(m_0, m_1, pk) \\
b & \leftarrow & \{0,1\} \\
c & \leftarrow & Enc_{pk}(m_b) \\
b' & \leftarrow & A^{O_2}(m_0, m_1, pk, c) \\
Повернути & b' = b &
\end{array}$$

Рис. 2. Ігри $Adv_{A,П}^{IND-ATK-0}(\lambda)$ та $Adv_{A,П}^{IND-ATK-1}(\lambda)$ для асиметричної схеми шифрування

Значення оракулів O_1, O_2 для IND-CPA, IND-CCA1, IND-CCA2 зведені у табл. 1. Фактично використовується тільки оракул дешифрування.

Таблиця 1

Значення оракулів O_1, O_2 для схеми асиметричного шифрування

Модель	Оракул O_1	Оракул O_2
IND-CPA	-	-
IND-CCA1	Оракул дешифрування O_{Dec}	-
IND-CCA2	Оракул дешифрування O_{Dec}	Оракул дешифрування O_{Dec}

Оракул дешифрування може розшифрувати будь-який шифротекст, проте має деяку відмінність. Якщо шифротекст був виданий супротивнику як завдання, то оракул дешифрування незалежно від дій супротивника повертатиме $\perp$ на запит розшифрування завдання.

2.2. Моделі безпеки для механізмів інкапсуляції ключів

Для механізму інкапсуляції ключів $\Pi = (Gen, Encaps, Decaps)$ ігри $Adv_{A,\Pi}^{IND-ATK-0}(\lambda)$ та $Adv_{A,\Pi}^{IND-ATK-1}(\lambda)$ для $ATK \in \{CPA, CCA1, CCA2\}$ зображено на рис. 3. Кожна гра аналогічно відбувається між випробовувачем та супротивником (криптоаналітиком), проте вміст ігор відрізняється. Спочатку випробовувач генерує випадкову ключову пару та надсилає відкритий ключ супротивнику, який робить запити до оракула O_1. Після чого супротивник повідомляє випробовувачу, що готовий отримати завдання. У грі $Adv_{A,\Pi}^{IND-ATK-0}(\lambda)$ випробовувач генерує ключ K та його інкапсуляцію C, потім надсилає ключ і інкапсуляцію супротивнику. У грі $Adv_{A,\Pi}^{IND-ATK-1}(\lambda)$ випробовувач генерує ключ K та його інкапсуляцію C, але замість ключа надсилає супротивнику випадковий рядок бітів. Супротивник робить запити до оракула O_2 та генерує 0, якщо грає у гру $Adv_{A,\Pi}^{IND-ATK-0}(\lambda)$, та 1, якщо грає у гру $Adv_{A,\Pi}^{IND-ATK-1}(\lambda)$.

$$
\begin{array}{ccc}
(pk, sk) & \leftarrow & Gen(1^{\lambda}) \\
(pk) & \rightarrow & A^{O_1}(pk) \\
b & \leftarrow & \{0,1\} \\
(K_0, C) & \leftarrow & Encaps(pk) \\
K_1 & \leftarrow & \{0,1\}^{KeyLen} \\
b' & \leftarrow & A^{O_2}(pk, C, K_b) \\
\textit{Повернути} & b' == b &
\end{array}
$$

Рис. 3. Ігри $Adv_{A,\Pi}^{IND-ATK-0}(\lambda)$ та $Adv_{A,\Pi}^{IND-ATK-1}(\lambda)$ для механізму інкапсуляції ключів

Значення оракулів O_1, O_2 для IND-CPA, IND-CCA1, IND-CCA2 зведені у табл. 2. Фактично використовується тільки оракул декапсуляції.

Таблиця 2

Значення оракулів O_1, O_2 для механізму інкапсуляції ключів

Модель	Оракул O_1	Оракул O_2
IND-CPA	-	-
IND-CCA1	Оракул декапсуляції O_{Decaps}	-
IND-CCA2	Оракул декапсуляції O_{Decaps}	Оракул декапсуляції O_{Decaps}

Оракул декапсуляції діє аналогічно до оракула дешифрування та може декапсулювати ключ з інкапсуляції з обмеженнями на інкапсуляцію, що обрана у якості завдання.

Варто зазначити, що моделі $IND-CPA, IND-CCA1, IND-CCA2$ утворюють ланцюг включень:

$$IND-CPA \subseteq IND-CCA1 \subseteq IND-CCA2. \quad (9)$$

Тобто з безпеки у моделі IND-CCA2 випливає безпека у моделі IND-CCA1 і з безпеки у моделі IND-CCA1 випливає безпека у моделі IND-CPA.

2.3. Модель випадкового оракула

Докази у моделях на основі нерозрізнювальності мають, як правило, схожу структуру. В основі лежить техніка доказу "GAME HOPING" [9, 14]. Сутність техніки полягає у тому, що спочатку розглядається оригінальна гра, для якої складно зробити оцінку переваги супротивника. Далі гра дещо змінюється і вимірюється як зміниться при цьому перевага супротивника. Після серії таких змін можливо отримати гру, для якої легко оцінити перевагу, наприклад звести до гри, у якій супротивнику завжди даються виключно випадкові дані. Оскільки для останньої гри оцінка відома і відомо як змінюється перевага при переходах від однієї гри до іншої, то можливо отримати обмеження зверху для оригінальної гри.

Проблема такого підходу полягає у тому, що для деяких моментів важко оцінити як зміниться вплив на безпеку при їх заміні. В першу чергу це стосується геш-функцій. В їх алгебраїчній структурі можуть міститися вразливості або комбінація їх алгебраїчних властивостей з властивостями криптографічної системи може приводити до вразливостей.

На жаль, у загальному випадку рішень цієї проблеми невідомо. У сучасних розробках використовується так звана модель випадкового оракула, у якій геш-функції або елементи, що ведуть себе схожим чином, замінюються на випадкові оракули. Випадковий оракул – це ідеальна геш-функція, що не має структури. Супротивник замість викликів реальних функцій робить виклики випадкових оракулів для функцій, чий вплив важко оцінити. Існують приклади систем, які є безпечними у моделі випадкового оракула, проте небезпечні на практиці. Незважаючи на те, що таке послаблення дещо виключає з розгляду ряд атак, модель випадкового оракула де-факто є стандартом у сучасній криптографії. Безпека переважної більшості криптографічних систем оцінюється саме у моделі випадкового оракула. Більш того, доказ у моделі випадкового оракула дозволяє використовувати криптографічну схему як будівельний блок у більш складних моделях безпеки криптографічних протоколів на кшталт моделі Канетті – Кравчека [15] для протоколів узгодження ключів.

3. Відомості з теорії решіток

Введемо позначення: нехай $R_q = \mathbb{Z}_q[X]/(X^n - X - 1)$ – кільце поліномів над $\mathbb{Z}_q$ з твірним поліномом $X^n - X - 1$, а R_3 – множина поліномів кільця R_q, усі коефіцієнти яких належать до множини $\{-1, 0, 1\}$. Позначимо як $R_3^{a,b}$ множину усіх поліномів у R_3, що мають кількість ненульових елементів у діапазоні $[a, b]$. Якщо $a = b$, то використовується скорочене позначення $R_3^{t,t} = R_3^t$.

ДСТУ 8961:2019 грунтується на проблемі NTRU [16]. У загальному вигляді проблему NTRU можливо визначити наступним чином. Нехай $q > 2$ – ціле число, $\gamma > 2$ – дійсне чило, D – деякий розподіл ймовірностей над полем R_q. Проблема (D, γ, γ')-NTRU (проблема

пошуку) полягає у тому, щоб для $h \in R_q$ з розподілу D знайти пару $(f, g) \in R_q \times R_q \setminus \{0,0\}$, для яких виконується $g \cdot h = f \bmod q$ та $\| f \|, \| g \| \leq \sqrt{q} / \gamma$.

NTRU-припущенням є припущення, що для будь-якого рівня безпеки λ існують такі параметри для задачі NTRU, що перевага будь-якого алгоритма у вирішенні проблеми NTRU є незначною (у сенсі визначень в розд. 2).

На основі проблеми NTRU природнім чином будується одностороння функція з лазівкою. Функція $NTRU_h : R_q \times R_q \to R_q$ для деякого полінома $h \in R_q$ визначається так:

$$NTRU_h(m, r) = m + rh. \tag{10}$$

Поліном h задає базис-решітку з базисом $(1, h) \in R_q \times R_q$. Відображення $NTRU_h$ семплує деяку точку на цій решітці. Ісходні координати точки відновити доволі важко, якщо відомий тільки "поганий" базис $(1, h) \in R_q \times R_q$, проте, якщо відомий деякий гарний базис цієї ж решітки $(f, g) \in R_q \times R_q$, то задачу можливо вирішити за поліноміальний час. Припущення, що така функція (при відповідних параметрах) є односторонньою функцією з лазівкою, є необхідним припущенням для безпеки ДСТУ 8961:2019.

4. ДСТУ 8961:2019

Перед тим, як перейти до розгляду ДСТУ 8961:2019, зауважимо, що текст стандарту є доволі технічним і опис перетворення містить багато технічних деталей, які не впливають на аналіз у моделі випадкового оракула. Тому для полегшення аналізу введемо наступні геш-функції, які є еквівалентними до перетворень, що використовуються у ДСТУ 8961:2019:

$$\begin{gathered} BPGM : \{0,1\}^{8*\max MsgLenBytes + db} \times R_q \to R_q \\ MGF : R_q \to \{0,1\}_3 \\ H : R_q \to \{0,1\}^{\lambda} \\ KDF : R_q \to \{0,1\}^{K\_bytes} \end{gathered},$$

де λ – параметр безпеки, t – загальносистемний параметр, а maxMsgLenBytes, db, K_bytes – константи, що визначаються на основі загальносистемних параметрів. Також використовується бієктивне відображення:

$$\begin{gathered} Pad : \{0,1\}^{8*\max MsgLenBytes} \times \{0,1\}^{db} \to R_3 \\ Pad^{-1} : R_3 \to \{0,1\}^{8*\max MsgLenBytes} \times \{0,1\}^{db} \end{gathered},$$

яке кодує повідомлення (у вигляді строки бітів), довжину повідомлення та випадкову строку бітів у поліном з R_3.

Механізм інкапсуляції ключів ДСТУ 8961:2019 використовує у якості CPA-to-CCA перетворення гібридний варіант перетворень Дента власної розробки [10, 3] для отримання IND-CCA2 безпечного механізму інкапсуляції ключів з ймовірнісної схеми асиметричного шифрування. Схема асиметричного шифрування, що лежить в основі протоколу інкапсуляції ключів, наведена на рис. 4. Протокол інкапсуляції ключів наведено на рис. 5.

SkelyaPKE.Gen(1^λ):
1. $f \leftarrow_R R_3^{2t}$
2. $g \leftarrow_R R_3^{\left\lfloor \frac{2n}{3}+1 \right\rfloor}$
3. $if\ (3f+1)^{-1} = \perp$ goto 1
4. $h = (3f+1)^{-1} g \in R_q$
5. $return\ (pk = h, sk = f$

SkelyaPKE.Enc($msg, coins, h$):
1. $m = Pad(msg, coins)$
2. $r = BPGM(msg, coins, h)$
3. $R = rh$
4. $m' = m + MGF(R)$
5. $if\ m' \notin R_3^{2t,n-2t}$ return $\perp$
6. $c = R + m'$
7. $return(c)$

SkelyaPKE.Dec($c, (f, h)$):
1. $a = fc$
2. $m' = a\ mod\ 3$
3. $if\ m' \notin R_3^{2t,n-2t}$ return $\perp$
4. $R = c - m'$
5. $m = m' - MGF(R)$
6. $(msg, coins) = Pad^{-1}(m)$
7. $r' = BPGM(msg, coins, h)$
8. $R' = r'h$
9. $if\ R' = R\ return\ msg$
10. $return\ \perp$

Рис. 4. Ймовірністна асиметрична схема шифрування у ДСТУ 8961:2019

SkelyaKEM.Gen(1^λ):
1. $return\ (pk, sk) = SkelyaPKE.Gen(1^\lambda)$

SkelyaKEM.Encaps($pk = h$):
1. $x \leftarrow_R \{0,1\}^{MsgLen}$
2. $r = BPGM(x, h)$
3. $C_1 = SkelyaPKE.Enc(x, r, pk)$
4. $if\ C_1 = \perp$ goto 1
5. $C_2 = H(r)$
6. $K = KDF(r)$
7. $C = (C_1, C_2)$
8. 8. $return\ (C, K)$

SkelyaKEM.Decaps($C = (C_1, C_2), sk = (f, h)$):
1. $x = SkeylaPKE.Dec(C_1, sk)$
2. $if\ x = \perp$ return $\perp$
3. $r = BPGM(x, h)$
4. $C_2' = H(r)$
5. $C_1' = SkelyaPKE.Enc(x, r, h)$
6. $if\ C_1' = C_1$ && $C_2' = C_2$
7. $return\ K = KDF(r)$
8. $return\ \perp$

Рис. 5. Механізм інкапсуляції ключів ДСТУ 8961:2019

ДСТУ 8961:2019 підтримує три режими роботи для 256, 384 та 512 біт безпеки. Опис загальносистемних параметрів та їх значення для рівнів безпеки наведений у табл. 3.

Таблиця 3

Загальносистемні параметри для 256, 384, 512 біт безпеки

Параметр	Опис	256 біт	384 біта	512 біт
N	Ступінь поліномів	881	1201	1471
q	Великий модуль (модуль поля)	7673	9221	12251
p	Малий модуль	3	3	3
t	Визначає кількість ненульових коефіцієнтів в малих поліномах	159	192	255
d_g	Кількість ненульових коефіцієнтів в поліномі g. Визначається за формулою $d_g = \lfloor N/3 \rfloor$	293	400	490
d_f	Кількість ненульових коефіцієнтів в поліномі f. Визначається за формулою $d_f = 2t$	318	384	510
d_r	Кількість ненульових коефіцієнтів в поліномі r. Визначається за формулою $d_r = 2t$	318	384	510

5. Конкретні оцінки безпеки

Оскільки перетворення CPA-to-CCA гарантує безпеку механізму інкапсуляції ключа, то загальносистемні параметри, які забезпечують безпеку схеми асиметричного шифрування, будуть також безпечними і для механізму інкапсуляції ключів, то можливо сфокусуватися лише на схемі асиметричного шифрування.

На високому рівні, за умови безпеки загальної конструкції схеми, атаки для ДСТУ 8961:2019 можна поділити на три класи [17]:

- Атаки, що передбачають знання деякої кількості пар текст/шифротекст, які викликають помилки дешифрування.
- Атаки, направлені на односторонню функцію $NTRU_h$ у процедурі шифрування. Тобто, знайти прообраз функції для $e = NTRU_h(m', r)$.
- Атаки, направлені на односторонню функцію $NTRU_h$ у процедурі генерації ключової пари. Тобто, знайти прообраз функції для $NTRU_h(-f, g) = 0 \bmod q$.

Розглянемо перший клас атак. Інтуітивно зрозуміло, що чим складніше знайти помилку дешифрування, тим складніше реалізувати таку атаку. Помилка дешифрування відбувається у випадку, якщо $p(rg + m'F)$ має хоча б один коефіцієнт з абсолютним значенням більшим за $q/2$. Оскільки для будь-яких $u, v \in R_q$ має місце нерівності:

$$\begin{array}{l} \| uv \|_\infty \le 2 \| u \|_\infty \cdot \| v \|_1 \\ \| uv \|_\infty \le 2 \| u \|_2 \cdot \| v \|_2 \end{array}. \tag{11}$$

То, враховуючи, що $\| m' \|_\infty = \| g \|_\infty = 1$ та $\| F \|_1 = \| r \|_1 = 2t$, маємо умову відсутності помилок дешифрування:

$$\begin{array}{c} \| prg + m'F \|_\infty \le \| m \|_\infty + 3 \| m'F + rg \|_\infty \le \\ | 1 + 6(\| m' \|_\infty \| F \|_1 + \| g \|_\infty \| r \|_1) \le 1 + 24t < q/2 \end{array}. \tag{12}$$

Отже, якщо $q \ge 48t + 3$, то помилки дешифрування в ДСТУ 8961:2019 відсутні. Оскільки ця умова виконується для усіх наборів загальносистемних параметрів в стандарті, то помилки дешифрування відсутні і атаки такого класу відсутні для цієї криптосистеми.

У роботі [18] запропоновано техніку, яка показує як звести задачу вирішення рівняння $NTRU_h(-f, g)$ до знаходження прообраза $NTRU_h(m', r)$, що поєднує ці два вектори атак в один напрямок. Задача криптоаналізу ДСТУ 8961:2019 зводиться до криптоаналізу проблеми NTRU. Робіт з криптоаналізу NTRU існує чимало, зокрема для ДСТУ 8961:2019 авторами опубліковано методологію оцінки для гібридних атак та атак з використанням редукції решіток [3]. Детальний огляд результатів щодо редукції решіток та гібридним атакам виходить за межі цієї роботи.

Наостанок зауважимо, що ДСТУ 8961:2019 використовує поле $\mathbb{Z}_q[X]/(X^n - X - 1)$, яке не має нетривіальних підполів. Такий вибір дає захист від алгебраїчних атак на підполе [19] та S-Unit атак [20], теорія яких в останні роки дуже розвинулась для поля $\mathbb{Z}_q[X]/(X^n - X - 1)$. Алгебраїчні квантові атаки, що були викладені у серії робіт [21], також нерелевантні для ДСТУ 8961:2019 через використання нетипового поля та модульної структури NTRU.

6. Аналіз асиметричної схеми

Основний результат цього розділу викладено в теоремі 1.

Т е о р е м а 1. Якщо існує алгоритм A, що може перемогти у IND-CCA2 грі для SkeyaPKE за поліноміальний час з ймовірністю ε та робить $q_{BPGM}, q_{MGF}, q_{Dec}$ запитів до

випадкових оракулів BPGM, MGF та оракула дешифрування, то існує алгоритм B, що може знайти прообраз односторонньої функції NTRU з ймовірністю ε':

$$\varepsilon' \geq \varepsilon - \frac{q_{BPGM}}{|R_3^t|} - \frac{q_{Dec}\gamma}{|R_3^t|}. \tag{13}$$

Доказ.

Алгоритм B приймає у якості аргумента шифротекст c^*, відкритий ключ $pk = h$ та повертає m^*, r^* – відкритий текст та випадкове значення, що було використано для шифрування. Сутність алгоритму B полягає у симуляції IND-CCA2 гри для алгоритму A та працює наступним чином:

- Алгоритм B передає до A відкритий ключ pk.
- Алгоритм B отримує два випадкових повідомлення M_0, M_1 від IND-CCA2 іспитувача та передає їх до A.
- Алгоритм B обирає випадковий біт σ та передає c^* до B як шифротекст повідомлення $M^* = M_\sigma$ (тобто випадкові оракули відповідатимуть на запити від A таким чином, щоб повідомлення M_σ було результатом дешифрування c^*. Опис роботи оракулів нижче).
- Алгоритм A робить запити до оракулів.
- Алгоритм A повертає відповідь σ'. Якщо у запитах до оракулів є інформація, з якої можливо відновити m^*, r^*, то повернути m^*, r^*.

Повернути випадкові значення m^*, r^*.

Для симуляції випадкових оракулів використовуються два списки: $BPGM_{LIST}$ та MGF_{LIST}. Випадкові оракули виконують часткову симуляцію відповідних геш-функцій, як показано нижче.

Оракул для $BPGM(x)$ працює наступним чином:

- Якщо $(x, r) \in BPGM_{LIST}$, то повернути r.
- Якщо $x = M^* \| coins^* \| h$, то повернути r^*.
- Інакше обрати випадкове r, додати до $BPGM_{LIST}$ (x, r) та повернути r.

Оракул для $MGF(x)$ працює наступним чином:

- Якщо $(x, mask) \in MGF_{LIST}$, то повернути $mask$.
- Якщо $x = r^* \cdot h$, то повернути $m^* - MGF(r^* \cdot h)$.
- Інакше обрати випадкове $mask$, додати до MGF_{LIST} $(x, mask)$ та повернути $mask$.

Оракул дешифрування $Dec(c)$ працює наступним чином:

- Якщо $c = c^*$, то повернути $\perp$.
- Якщо у $BPGM_{LIST}$ містяться (x, r), для яких $Enc(x, r, pk) = c$, то повернути x.
- Інакше повернути помилку дешифрування.

Позначимо як W_{real}^{win}, та W_{oracle}^{win}, події, що відповідають перемозі A у IND-CCA2 грі з реальними геш-функціями та з оракулами відповідно, а $W_{real}^{bpgm}, W_{real}^{mgf}$ та $W_{oracle}^{bpgm}, W_{oracle}^{mgf}$ – події, що відповідають виклику оракулів для BPGM та MGF на даних з шифротексту у грі з реальними геш-функціями та з оракулами відповідно. Також введемо наступні позначення:

$$\begin{aligned} W_{real}^{query} &= W_{real}^{bpgm} \vee W_{real}^{mgf} \\ W_{oracle}^{query} &= W_{oracle}^{bpgm} \vee W_{oracle}^{mgf} \end{aligned}. \tag{14}$$

Розглянемо ймовірність перемоги у грі з реальними геш-функціями. За формулою повної ймовірності маємо:

$$\Pr[W_{real}^{win}] = \Pr[W_{real}^{win} \mid \neg W_{real}^{query}] \Pr[\neg W_{real}^{query}] + \Pr[W_{real}^{win} \mid W_{real}^{query}] \Pr[W_{real}^{query}]. \quad (15)$$

Якщо алгоритм A не робить запитів до геш-функцій, то він, відповідно, не має жодної інформації про біт σ, отже $\Pr[W_{real}^{win} \mid \neg W_{real}^{query}] = 1/2$. Підставивши у вираз для $\Pr[W_{real}^{win}]$, маємо:

$$\begin{aligned} \Pr[W_{real}^{win}] &= \frac{1}{2} \Pr[\neg W_{real}^{query}] + \Pr[W_{real}^{win} \mid W_{real}^{query}] \Pr[W_{real}^{query}] \\ &= \frac{1}{2}\left(1 - \Pr[\neg W_{real}^{query}]\right) + \Pr[W_{real}^{win} \mid W_{real}^{query}] \Pr[W_{real}^{query}] \\ &= \frac{1}{2} + \Pr[W_{real}^{query}]\left(\Pr[W_{real}^{win} \mid W_{real}^{query}] - \frac{1}{2}\right) \end{aligned} . \quad (16)$$

Оскільки $\Pr[W_{real}^{win} \mid W_{real}^{query}] - \frac{1}{2}$ завжди менше $\frac{1}{2}$, то маємо:

$$\Pr[W_{real}^{win}] \le \frac{1}{2}(1 + \Pr[W_{real}^{query}]). \quad (17)$$

Звідси випливає, що перевага у IND-CCA2 грі є меншою, ніж $\Pr[W_{real}^{query}]$, яку можливо розкласти як

$$\Pr[W_{real}^{query}] = \Pr[W_{real}^{bpgm} \wedge \neg W_{real}^{mgf}] + \Pr[W_{real}^{mgf}]. \quad (18)$$

Якщо подія W_{real}^{mgf} не відбулася, то A відповідно не має інформації про σ^*, отже можливо обмежити $\Pr[W_{real}^{bpgm} \wedge \neg W_{real}^{mgf}] \le \frac{q_{BPGM}}{|R_3^t|}$:

$$\Pr[W_{real}^{win}] \le \Pr[\neg W_{real}^{query}] \le \frac{q_{BPGM}}{|R_3^t|} + \Pr[W_{real}^{mgf}]. \quad (19)$$

Розглянемо як зміниться ймовірність перемоги, якщо замінити справжні геш-функції на випадкові оракули. Для алгоритму A різниця не буде помітна, якщо оракул дешифрування не поверне $\perp$ на валідний запит

Позначимо як W_{ind} подію, що зазначені вище випадки не стануться. У цьому випадку маємо:

$$\begin{aligned} \Pr[W_{oracle}^{mgf}] &\ge \Pr[W_{oracle}^{mgf} \mid W_{ind}] \Pr[W_{ind}] \\ &= \Pr[W_{real}^{mgf} \mid W_{ind}] \Pr[W_{ind}] \\ &\ge \left(\Pr[W_{real}^{win}] - \frac{q_{BPGM}}{|R_3^t|}\right) \Pr[W_{ind}] \end{aligned} . \quad (20)$$

Розглянемо ймовірність того, що оракул дешифрування не поверне $\perp$ на валідний запит. Нехай $c \ne c^*$ – деякий запит до оракула дешифрування, на який повернули $\perp$. Валідний шифротекст може бути відкинуто у грі з оракулами, якщо не було відповідного запиту до BPGM. Оскільки ймовірність звернення до випадкового оракула BPGM обмежена зверху як $\frac{\gamma}{|R_3^t|}$, де γ визначено формулою (5), то відповідно ймовірність того, що не відбудеться повернення $\perp$, складає $1 - \frac{q_{Dec}\gamma}{|R_3^t|}$.

Поєднуючи формули, маємо:

$$\varepsilon' \geq \varepsilon - \frac{q_{BPGM}}{|R_3^t|} - \frac{q_{Dec}\gamma}{|R_3^t|}. \quad (21)$$

7. Аналіз CPA-to-CCA перетворення

Доказ IND-CCA2 безпеки SkelyaPKE у моделі випадкового оракула грунтується на стандартних техніках. Використовується наступна технічна лема [10]:

Л е м м а 1. Позначимо як A, B, E деякі події в ймовірнісному просторі. Якщо $\Pr[A|\neg E] = \Pr[B|\neg E]$, то має місце нерівність $|\Pr[A] - \Pr[B]| \leq \Pr[E]$.

Т е о р е м а 2. Нехай $PPKE = (Gen, Enc, Dec)$ – деяка ймовірнісна схема асиметричного шифрування, а SkelyaKEM – механізм інкапсуляції ключей, що побудований за допомогою застосування перетворення на рис. 2 до PPKE. Якщо існує алгоритм A, що може перемогти у IND-CCA2 грі SkelyaKEM за поліноміальний час з ймовірністю ε та робить $q_{BPGM}, q_H, q_{KDF}, q_{Decaps}$ запитів до випадкових оракулів BPGM, H, KDF та оракула дешифрування, то існує алгоритм B, що може інвертувати PPKE з ймовірністю

$$\varepsilon' \geq \varepsilon - \frac{q_{Decaps}}{|R_3^{2t,N-2t}|} - \frac{\gamma q_{Decaps}}{2^{\lambda}}. \quad (22)$$

Доказ.

Алгоритм B приймає у якості аргумента шифротекст $PPKE - C_1^*$, відкритий ключ pk та повертає x. Сутність алгоритму B полягає у симуляції IND-CCA2 гри для алгоритму A та працює наступним чином:

- Алгоритм B генерує випадкові бітові строки C_2^*, K^*.
- Алгоритм B передає pk до A.
- Алгоритм B чекає доки A запросить завдання, на запит B посилає пару $\left(C^* = (C_1^*, C_2^*), K^*\right)$.
- Алгоритм B чекає доки A поверне біт σ'.
- Алгоритм B перевіряє чи існує $(x, r) \in BPGM_{LIST}$, для якого виконується $Enc(x, r, pk) = C_1^*$. Якщо так, то обчислює та повертає x.
- Алгоритм B перевіряє чи існує $(r, K) \in KDF_{LIST}$ або $(r, hash) \in H_{LIST}$, для якого виконується $x = Pad^{-1}(C_1^* - rh - MGF(rh)); Enc(x, r, pk) = C_1^*$. Якщо так, то повертає x.
- Алгоритм B повертає випадкове x.

Оракул для $BPGM(x)$ працює наступним чином:

- Якщо $(x, r) \in BPGM_{LIST}$, то повернути r.
- Якщо $Decaps(C_1^*, sk) = x$, то повернути $BPGM(x)$.
- Інакше обрати випадкове r, додати до $BPGM_{LIST}$ (x, r) та повернути r.

Оракул для $H(r)$ працює наступним чином:

- Якщо $(r, hash) \in H_{LIST}$, то повернути $hash$.
- Якщо $BPGM(Decaps(C_1^*, sk), pk) = r$, то повернути $H(r)$.
- Інакше обрати випадкове $hash$, додати до H_{LIST} $(r, hash)$ та повернути $hash$.

Оракул для $KDF(r)$ працює наступним чином:

- Якщо $(r, K) \in KDF_{LIST}$, то повернути K.
- Якщо $BPGM(Decaps(C_1^*, sk), pk) = r$, то повернути $KDF(r)$.
- Інакше обрати випадкове K, додати до KDF_{LIST} (r, K) та повернути K.

Оракул декапсуляції $Decaps((C_1, C_2))$ працює наступним чином:

- Якщо $C_1 = C_1^*$, то повернути $\perp$.
- Для кожного значення $(x, r) \in BPGM_{LIST}$ перевірити, чи виконується $Enc(x, r, pk)$ та $H(r) = C_2$. Якщо такої пари не знайдено, то повернути $\perp$.
- Обчислити $K = KDF(r)$ з використанням оракула для KDF.
- Повернути K.

Позначимо як W_{real}^{win} та W_{oracle}^{win} події, що відповідають перемозі A у IND-CCA2 грі з реальними геш-функціями та з оракулами відповідно. Розглянемо як зміниться ймовірність перемоги, якщо замінити справжні геш-функції на випадкові оракули. Для алгоритму A різниця не буде помітна, якщо:

- A робив запит на розшифрування завдання до того, як отримав завдання.
- A робив запит на декапсуляцію валідного шифротексту C_1, C_2, але A не робив відповідних запитів до H або BPGM.

Позначимо як W_{ind} подію, що зазначені вище випадки не стануться, отже, використовуючи Л е м м у 1, маємо:

$$\begin{gathered} \Pr[W_{real}^{win} \mid \neg W_{ind}] = \Pr[W_{oracle}^{win} \mid \neg W_{ind}] \Rightarrow \\ \mid \Pr[W_{real}^{win}] - \Pr[W_{oracle}^{win}] \mid \leq \Pr[W_{ind}] \end{gathered}. \tag{23}$$

Так як шифротекст був обраний випадково, то ймовірність першої події обмежена $\frac{q_{Decaps}}{\mid R_3^{2t,N-2t} \mid}$. Ймовірність того, що $H(BPGM(Decaps(C, sk), pk))$ обмежена $\frac{\gamma q_{Decaps}}{2^{\lambda}}$:

$$\frac{q_{Decaps}}{\mid R_3^{2t,N-2t} \mid} + \frac{\gamma q_{Decaps}}{2^{\lambda}}. \tag{24}$$

Оскільки перевага у грі з оракулом є нижньою оцінкою ймовірності того, що серед запитів до геш-функцій буде інформація, якої достатньо для дешифрування шифротексту C_1^*, то маємо

$$\varepsilon' \geq \varepsilon - \frac{q_{Decaps}}{\mid R_3^{2t,N-2t} \mid} - \frac{\gamma q_{Decaps}}{2^{\lambda}}. \tag{25}$$

8. Пропозиції щодо покращення

У розд. 6, 7 було показано, що схема асиметричного шифрування та механізм інкапсуляції ключів, що використовуються в ДСТУ 8961:2019, є безпечними в моделі випадкового оракула, проте також варто зауважити, що з аналізу також випливає ряд недоліків.

По-перше, стандарт визначає механізм інкапсуляції ключів, який фактично будується на основі IND-CCA2 безпечної схеми асиметричного шифрування. Для CPA-to-CCA перетворення, що використовується у ДСТУ 8961:2019, достатньо було б IND-CPA безпечної схеми. Механізм інкапсуляції ключів надзвичайно збитковий. Також у механізмі присутні деякі деталі, які не впливають на безпеку, проте при реалізації дещо уповільнюють схему. Наприклад, стандарт передбачає, що ключем інкапсуляції є поліном r, який виконує роль випадкового значення. Такий вибір потребує додаткових обчислень з кодування-декодування полінома в бітову строку, при тому, що не збільшує безпеку, а навпаки – ускладнює аналіз. В інших механізмах інкапсуляції ключів роль ключа виконує випадкове повідомлення m, що відповідає семантиці механізму інкапсуляції ключів як криптопримітива і не створює додаткових перешкод при аналізі.

Варто зауважити, що такий незвичний вибір може призвести до більш зручних доказів у моделі квантового оракула. Необхідні додаткові дослідження для цього випадку. Наведені докази у прямому вигляді доволі важко узагальнити на квантовий випадок через те, що квантовий оракул приймає усі запити в суперпозиції, і аргументи, що використовуються, неможк-

ливо використати. Втім, у роботі [12] запропоновано ряд технік, які могли б усунути цю перешкоду.

В реальному світі протоколи інкапсуляції ключів використовуються як складова частина більш складних протоколів. Зокрема, механізм інкапсуляції ключів можливо використовувати як основу для протоколів автентифікованого обміну ключами.

У роботах [22, 23] запропонований узагальнений протокол автентифікованого обміну ключів на основі довільного механізму інкапсуляції ключей, що є безпечним у моделі IND-CCA2. На рис. 6, 7 запропоновано протоколи SkelyaAKE1 з односторонньою автентифікацією та SkelyaAKE2 – з двохсторонньою автентифікацією.

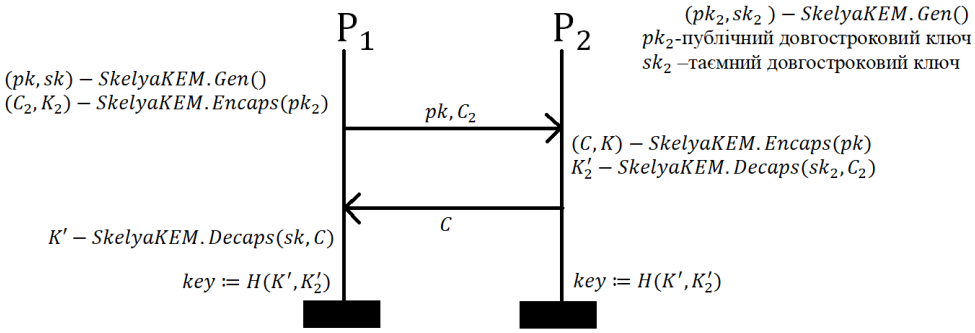

Рис. 6. Протокол SkelyaAKE1 з односторонньою автентифікацією

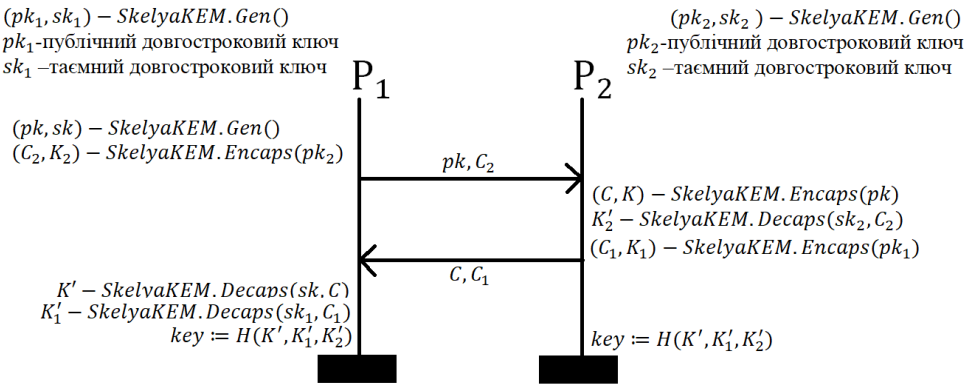

Рис. 7. Протокол SkelyaAKE2 з двохсторонньою автентифікацією

Сторони, що проводять автентифікацію, мають довгострокові ключі. Публічний довгостроковий ключ відомий іншій стороні. Цікавою задачею є створення протоколів на основі ДСТУ 8961:2019 з врахуванням його особливостей, а не з використанням узагальнених доказово безпечних конструкцій.

Висновки

1. В моделі випадкового оракула IND-CCA2 доведено безпеку схеми асиметричного шифрування, що описана в ДСТУ 8961:2019, та безпеку відповідного механізму інкапсуляції ключів. Оскільки стандарт містить лише технічний опис перетворень, у розд. 4 було введено формалізовану математичну модель без зайвих технічних деталей, що не впливають на оцінки безпеки.

2. Оскільки загальносистемні параметри в стандарті було обрано таким чином, щоб схема не мала помилок дешифрування, вдалося значно спростити доказ. У розд. 5 наведено схематичний огляд можливих векторів атак на ДСТУ 8961:2019, проте детальний аналіз є предметом подальших досліджень.

3. Важливо розуміти, що доказ у моделі випадкового оракула не означає, що взагалі атак немає, а лише доводить захист від широкого класу атак, що підпадають під модельні припущення. Наведений доказ може в подальшому бути вдосконалений з використанням більш сильних технік та з меншою кількістю модельних припущень.

4. Однією з важливих задач подальших досліджень є узагальнення отриманих результатів на квантовий випадок. Проте, для цього необхідні принципово інші техніки через те, що квантовий випадок приймає запити в суперпозиції, і побудувати оракулів дешифрування/декапсуляції аналогічно до того, як це приведено в доказах, стає майже неможливим.

5. Аналіз показав, що ДСТУ 8961:2019 має певну збитковість у сенсі безпеки. Конструкція може бути значно спрощена та пришвидшена без втрати безпеки. Безпеку, навпаки, можна значно підвищити.

6. Оцінка в моделі випадкового оракула сама по собі не забезпечує безпеку. Необхідно, щоб загальносистемні параметри відповідали необхідним рівням безпеки. Досліджуючи властивості загальних параметрів та доповнюючи існуючі докази, можна створити більш комплексні моделі безпеки, що враховують більшу кількість загроз і мають менше ризиків.

Список літератури:

1. ДСТУ 8961:2019. Інформаційні технології. Криптографічний захист інформації. Алгоритми асиметричного шифрування та інкапсуляції ключів. Чинний від 21.12.2019. Вид. офіц. Київ : УкрНДНЦ, 2019. 72 с.
2. Горбенко І. Д., Горбенко Ю. І. Прикладна криптологія. Теорія. Практика. Застосування : монографія. Харків : Форт, 2012. 880 с.
3. Calculation of general parameters for NTRU Prime Ukraine of 6-7 levels of stability / I. D. Gorbenko and other // Telecommunications and Radio Engineering. Vol. 78. P. 327 – 340
4. Methods of building general parameters and keys for ntru prime Ukraine of 5th-7th levels of stability. product form / I.D. Gorbenko and other // Telecommunications and Radio Engineering. 2018. Vol 78. P. 579 – 594.
5. NIST Post-Quantum Cryptography Standartization Project [Electronic resource] // Online: https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization
6. CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM // Cryptology ePrint Archive, Report 2017/634. [Electronic resource]. Online: https://eprint.iacr.org/2017/634.pdf
7. FrodoKEM. Learning With Errors Key Encapsulation Algorithm Specifications. And Supporting Documentation // [Electronic resource]. Online: https://frodokem.org/files/FrodoKEM-specification-20210604.pdf
8. Katz J., Lindell Y. Introduction to Modern Cryptography: Principles and Protocols // Chapman and Hall/CRC, 2007. 603 P.
9. Canetti R., Goldreich O., Halevi S. The random oracle methodology, revisited // 30th symposium on theory of computing. STOC, 1998. P. 209 – 218.
10. A. Dent. A Designer's Guide to KEMs // Lecture Notes in Computer Science. Vol. 2898. P. 28 – 44.
11. Howgrave-Graham N., Silverman J., Singer A. and William Whyte. NAEP: Provable security in the presence of decryption failures // Cryptology ePrint Archive, Report 2003/172. [Electronic resource]. Online: https://eprint.iacr.org/2003/172.
12. Hofheinz D., Hovelmanns K., Kiltz E. A modular analysis of the fujisaki-okamoto transformation // Lecture Notes in Computer Science. 2017. Vol. 10677. P. 341 – 371.
13. Goldreich O. Foundations of Cryptography/ Vol. 2. Cambridge University Press, 2000. 392 p.
14. Bellare M., Rogaway P. Random oracles are practical: A paradigm for designing efficient protocols // ACM Conference on Computer and Communications Security. 1993. Vol. 1. P. 62 – 73.
15. Canetti R., Krawczyk H. Analysis of key-exchange protocols and their use for building secure channels // EUROCRYPT. 2001. Vol. 2045. P. 453 – 474.
16. Hoffstein J., Pipher J., Silverman H. NTRU: a ring based public key cryptosystem // Algorithmic Nuber Theory. Third International Symposium. 1998. P. 267 – 288.
17. Hirschhorn P., Hoffstein J., Howgrave-Graham N. Choosing NTRUEncrypt Parameters in Light of Combined Lattice Reduction and MITM Approaches // Lecture Notes in Computer Science. Vol. 5536. P. 58 – 78.
18. Mol, P., Young M. Recovering NTRU Secret Key from Inversion Oracles // PKS. 2008. Vol. 4939. P. 18 – 36.
19. Micheli G., Heninger N., Shani B. Characterizing overstretched NTRU attacks // Journal of Mathematical Cryptology. 2020. Vol 14, Is 1. P. 110 – 119.
20. Bernstein D., Lange T. Non-randomness of S-unit lattices // [Electronic resource]. Online: https://s-unit.attacks.cr.yp.to/spherical.html
21. Eisenträger K., Hallgren S., Kitaev A. and Song F. A quantum algorithm for computing the unit group of an arbitrary degree number field // STOC. 2014. P. 293 – 302.
22. Fujioka A., Suzuki K., Xagawa K. and Yoneyama K. Strongly secure authenticated key exchange from factoring codes and lattices // PKC. 2012. Vol. 7293. P. 467 – 484.
23. Boyd C., Cliff Y., Gonzalez J. and Kenneth G. Efficient one-round key exchange in the standard model // ACISP. 2008. Vol. 5107. P. 69 – 83.

Надійшла до редколегії 04.10.2022

Відомості про автора:

Кандій Сергій Олегович – Харківський національний університет імені В. Н. Каразіна, аспірант кафедри безпеки інформаційних систем і технологій, факультет комп'ютерних наук, АТ «Інститут Інформаційних технологій», технік-конструктор, Україна; e-mail: sergeykandy@gmail.com