УДК 003.26

ПОБУДОВА МОДИФІКАЦІЇ ПОСТКВАНТОВОГО МЕХАНІЗМУ ІНКАПСУЛЯЦІЇ КЛЮЧА НА ОСНОВІ КРИПТОСИСТЕМИ AJPS

А. В. Ковальчук[1,a], Д. В. Ядуха[1]

[1] Навчально-науковий Фізико-технічний інститут

Анотація

У роботі виконано дослідження постквантового механізму інкапсуляції ключа AJPS-KEM$_1$ в контексті його перенесення з класу чисел Мерсенна на клас узагальнених чисел Мерсенна як модуля. Виконано адаптацію криптопримітиву AJPS-KEM$_1$ для арифметики за новим модулем та побудовано функцію Solve, за допомогою якої здійснюється декапсуляція ключа. Результати дослідження підтверджують можливість збільшення варіативності параметрів за рахунок застосування арифметики за модулем узагальненого числа Мерсенна у криптопримітиві.

Ключові слова: механізм інкапсуляції ключа AJPS-KEM$_1$, сімейство криптосистем AJPS, постквантові криптопримітиви, числа Мерсенна, узагальнені числа Мерсенна

Вступ

Основною проблемою симетричної криптографії є необхідність побудови захищеного каналу зв'язку для формування спільного секретного ключа. В 1976 році В. Діффі та М. Геллман заклали фундаментальні ідеї асиметричної криптографії, опублікувавши статтю «Нові напрямки в криптографії» [1], в якій представили перший асиметричний криптопримітив, який забезпечував вироблення спільного секрету без створення захищеного каналу зв'язку. Станом на сьогодні найбільш поширеними є асиметричні криптопримітиви, стійкість яких ґрунтується на складності розв'язання таких задач, як факторизація та дискретне логарифмування, наприклад криптосистеми RSA та ЕльГамаля.

Однак П. Шор в 1997 році представив алгоритм, користуючись яким квантові комп'ютери зможуть ефективно розв'язувати задачі факторизації та дискретного логарифмування [2]. Як наслідок, використання асиметричних криптопримітивів, стійкість яких ґрунтується на розв'язанні вище згаданих задач, залишається захищеним лише з огляду на те, що науковцям досі не вдалося побудувати масштабований квантовий комп'ютер.

Виходячи з таких міркувань, у 2017 році Національний інститут стандартів і технологій США (NIST) оголосив конкурс постквантових асиметричних криптопримітивів, які, як передбачається, будуть стійкими до атак з використанням як класичних, так і квантових комп'ютерів [3]. Оцінювання та відбір учасників відбувалися протягом трьох раундів, а також було проведено додатковий четвертий раунд для визначення альтернативного переможця. Чотирьох переможців третього раунду NIST оголосив 5 липня 2022 року, якими стали схеми цифрового підпису CRYSTALS-Dilithium, FALCON та SPHINCS+, а також механізм інкапсуляції ключа CRYSTALS-Kyber. 11 березня 2025 року переможцем додаткового раунду було оголошено механізм інкапсуляції ключа HQC, який в подальшому NIST планує стандартизувати як альтернативу CRYSTALS-Kyber.

Серед учасників першого раунду конкурсу є механізм інкапсуляції ключа Mersenne-756839 з сімейства криптосистем AJPS, особливістю якого є використання арифметики за модулем числа Мерсенна $M_n = 2^n - 1$, $n \in \mathbb{N}$. Одним із представників цього сімейства є схема побітового шифрування AJPS-1 [4], на основі якої згодом було побудовано механізм інкапсуляції ключа AJPS-KEM$_1$ [5]. Перевагою запропонованого механізму інкапсуляції ключа є підвищення пропускної здатності та ефективності схеми. Пізніше було побудовано модифікацію схеми шифрування AJPS-1 шляхом зміни класу чисел Мерсенна M_n на клас узагальнених чисел Мерсенна $GM_{n,m} = 2^n - 2^m - 1$, де n, $m \in \mathbb{N}$, $n > m$ [6].

У цій роботі зосередимось на побудові модифікації механізму інкапсуляції ключа AJPS-KEM$_1$ для класу узагальнених чисел Мерсенна $GM_{n,m}$.

1. Побудова модифікації AJPS-KEM$_1$

Нехай **AJPSGM-KEM$_1$** – механізм інкапсуляції ключа, що є модифікацією AJPS-KEM$_1$, запропонованого Х. Ферраді та Д. Наккаш у [5], з використанням арифметики за модулем узагальненого числа Мерсенна. Механізм інкапсуляції ключа визначається трьома процедурами: генерацією ключів (KeyGen), інкапсуляцією ключа (Encap) та декапсуляцією ключа (Decap). Детальний опис процедур модифікації

[a]anastkova-ipt27@lll.kpi.ua

механізму інкапсуляції ключа AJPSGM-KEM$_1$ наведено нижче.

Процедура KeyGen:

1. Враховуючи значення параметра захищеності λ, обирається просте узагальнене число Мерсенна $GM_{n,m} = 2^n - 2^m - 1$, де n, $m \in \mathbb{N}$, $n > m$, та додатне ціле число h, що задовольняють умовам:
   (а) $C_n^h \geq 2^\lambda$;
   (б) $4h^2 < n \leq 16h^2$.
2. Рівноймовірно та незалежно обираються два числа F та G з множини лишків за модулем узагальненого числа Мерсенна, що мають вагу Геммінга h, тобто $Ham(F) = Ham(G) = h$.
3. Відкритий ключ визначається як значення $\text{pk} := H = F \cdot G^{-1} \pmod{GM_{n,m}}$, особистим ключем є число $\text{sk} := G$, а значення F – секретний параметр криптопримітиву.

Процедура Encap: У відправника є відкритий ключ отримувача $\text{pk} = H$.

1. Рівноймовірно та незалежно обираються числа A та B з множини лишків за модулем $GM_{n,m}$, причому $Ham(A) = Ham(B) = h$.
2. Обчислюється $C = A \cdot H + B \pmod{GM_{n,m}}$.
3. Отримувачу надсилається шифротекст C, в який вбудовані значення A та B як інкапсульований ключ.

Процедура Decap: У отримувача є особистий ключ $\text{sk} = G$ та шифротекст C.

1. Обчислюється $W = G \cdot C \pmod{GM_{n,m}}$.
2. Отримувач намагається розв'язати рівняння
$$W = Fx + Gy \pmod{GM_{n,m}},$$
використовуючи функцію **Solve**.
3. Якщо розв'язок (x, y) знайдено, то повертається пара (A, B), яка і є інкапсульованим ключем, де $A = x$, $B = y$. Інакше результатом є символ $\perp$, що позначає помилку при декапсуляції.

Для аналізу стійкості побудованої модифікації механізму інкапсуляції ключа AJPSGM-KEM$_1$ знадобиться означення задачі GMLHRSP (англ. *Generalized Mersenne Low Hamming Ratio Search Problem*), що наведене нижче.

Означення 1 (GMLHRSP [6]). *Нехай є просте узагальнене число Мерсенна $GM_{n,m} = 2^n - 2^m - 1$, де n, $m \in \mathbb{N}$, $n > m$, число H таке, що $H < GM_{n,m}$, та h – додатне ціле число, яке визначає значення ваги Геммінга. Задача GMLHRSP полягає у пошуку за відомими значеннями $GM_{n,m}$, H та h двох чисел F та G з множини лишків за модулем узагальненого числа Мерсенна $GM_{n,m}$, вага Геммінга кожного з яких дорівнює значенню h, і для яких виконується:*
$$H = F \cdot G^{-1} \pmod{GM_{n,m}}.$$

Стійкість модифікації механізму інкапсуляції ключа AJPSGM-KEM$_1$ ґрунтується на складності розв'язання задачі GMLHRSP. Наступне твердження встановлює зв'язок між відновленням інкапсульованого ключа AJPSGM-KEM$_1$ та розв'язанням задачі GMLHRSP.

Твердження 1. *Злам інкапсульованого ключа, створеного AJPSGM-KEM$_1$, зводиться до розв'язання задачі GMLHRSP.*

Доведення. Нехай існує алгоритм $\mathcal{A}$, на вхід якому подається шифротекст C, значення H, h та число $GM_{n,m}$ як модуль, і алгоритм $\mathcal{A}$ відновлює інкапсульований ключ (A, B), створений AJPSGM-KEM$_1$:
$$\mathcal{A}(C, H, h, GM_{n,m}) = \{A, B\}.$$

Покажемо, що за допомогою цього алгоритму $\mathcal{A}$ можна ефективно розв'язати задачу GMLHRSP.

Нехай алгоритм $\mathcal{A}(C, H, h, GM_{n,m})$ відновлює інкапсульований ключ (A, B) шляхом розв'язання рівняння
$$C = A \cdot H + B \pmod{GM_{n,m}}$$
без використання особистого ключа G та секретного параметра F.

Застосуємо алгоритм для параметрів $\tilde{C}$, $\tilde{H}$, $\tilde{h}$ і $GM_{n,m}$, причому $\tilde{C} = 0$, $\tilde{H} = -H \pmod{GM_{n,m}}$, $\tilde{h} = h$. Тоді маємо:
$$\mathcal{A}(0, -H \pmod{GM_{n,m}}, h, GM_{n,m}).$$

Таким чином, алгоритм $\mathcal{A}$ розв'язує рівняння
$$0 = -A \cdot H + B \pmod{GM_{n,m}},$$
звідки слідує, що
$$H = B \cdot A^{-1} \pmod{GM_{n,m}},$$
причому $Ham(A) = Ham(B) = h$. Отже, шуканими значеннями задачі GMLHRSP є $F = B$ та $G = A$, що і треба було довести.

□

Для виконання процедури декапсуляції ключа в AJPSGM-KEM$_1$ необхідно застосувати функцію Solve для розв'язку рівняння
$$W = Fx + Gy \pmod{GM_{n,m}}.$$

Опис реалізації та особливостей функції Solve для класу чисел Мерсенна та класу узагальнених чисел Мерсенна як модуля наведено в наступному розділі.

2. Особливості реалізації функції Solve

Розглянемо структуру та принцип роботи оригінальної функції Solve для механізму інкапсуляції ключа AJPS-KEM$_1$ перед визначенням функції Solve для його модифікації AJPSGM-KEM$_1$. Нехай маємо
$$W := C \cdot G = A \cdot F + B \cdot G \pmod{M_n},$$
причому вага Геммінга числа W може бути оцінена як $Ham(W) \approx 2h^2$ за допомогою співвідношень для ваги Геммінга чисел за модулем числа Мерсенна, що наведені у [4].

Принцип роботи оригінальної функції Solve для класу чисел Мерсенна полягає в ітеративному відновленні невідомого числа A або B шляхом встановлення значення i-го біта шуканого числа. Коректність

оригінальної реалізації ґрунтується на властивості множення на степінь двійки за модулем числа Мерсенна M_n, що описана в наступному твердженні.

Твердження 1 [4]. *При множенні числа $x \pmod{M_n}$ на довільне число $y = 2^z \pmod{M_n}$, де $z \in \overline{\mathbb{0}, n-1}$ добуток $x \cdot y \pmod{M_n}$ може бути обчислений як циклічний зсув числа x на z бітів вліво.*

Використовуючи властивість з твердження 1, можна оцінити зміну ваги Геммінга чисел при множенні на степінь двійки за модулем числа Мерсенна M_n.

Наслідок 1 [4]. *При множенні $x \pmod{M_n}$ на степінь двійки за модулем числа Мерсенна вага Геммінга не змінюється:*

$$Ham(2^z \cdot x \pmod{M_n}) = Ham(x \pmod{M_n}).$$

Реалізація функції Solve для класу чисел Мерсенна полягає в побудові та перевірці гіпотези про те, що i-ий біт числа A дорівнює 1, базуючись на оцінці ваги Геммінга з наслідку 1. Розглянемо число Δ, яке обчислюється як:

$$\Delta = Ham(W) - Ham(W - 2^i \cdot F \pmod{M_n}).$$

Якщо гіпотеза справедлива й i-ий біт числа A дійсно дорівнює 1, то

$$\Delta \approx 2h^2 - (2h^2 - h) = h, \tag{1}$$

оскільки $Ham(2^i \cdot F \pmod{M_n}) = Ham(F) = h$ за наслідком 1. Якщо ж i-ий біт невідомого числа A дорівнює 0 і гіпотеза відхиляється, то при обчисленні значення Δ віднімається доданок $2^i \cdot F \pmod{M_n}$, якого не було в сумі W. Як наслідок, в такому випадку значення Δ є близьким до 0 і не дорівнює h. Отже, в механізмі інкапсуляції ключа AJPS-KEM$_1$ функція Solve відновлює i-й біт невідомого числа A за таким співвідношенням:

$$\begin{cases} 1, \text{ якщо } \Delta = h, \\ \mathbb{0}, \text{ якщо } \Delta \approx \mathbb{0}\ (\Delta \neq h). \end{cases}$$

Однак в ході дослідження можливості застосування такого підходу побудови функції Solve для класу узагальнених чисел Мерсенна встановлено таке:

- при множенні числа $x \pmod{GM_{n,m}}$ на довільне число $y = 2^z \pmod{GM_{n,m}}$, де $z \in \overline{\mathbb{0}, n-1}$ добуток $x \cdot y \pmod{GM_{n,m}}$ не є циклічним зсувом числа x на z бітів;
- при множенні $x \pmod{GM_{n,m}}$ на степінь двійки за модулем узагальненого числа Мерсенна вага Геммінга добутку може бути більшою за $Ham(x \pmod{GM_{n,m}})$, меншою або рівною.

Таким чином, співвідношення (1) не є істинним для класу узагальнених чисел Мерсенна і, як наслідок, реалізація функції Solve для класу узагальнених чисел Мерсенна аналогічно до оригінальної реалізації для класу чисел Мерсенна не є коректною та не може бути застосована в процедурі декапсуляції ключа. В наступному розділі представлено один із можливих підходів визначення функції Solve для класу узагальнених чисел Мерсенна.

3. Реалізація функції Solve для класу узагальнених чисел Мерсенна

Можлива реалізація функції Solve для класу узагальнених чисел Мерсенна як модуля ґрунтується на тому, що вага Геммінга чисел x та y є відомою:

$$Ham(x) = Ham(y) = h,$$

а також на тому, що число W – це сума доданків $2^i \cdot F \pmod{GM_{n,m}}$ та $2^j \cdot G \pmod{GM_{n,m}}$. Розглянемо співвідношення для параметра W:

$$W = Fx + Gy \pmod{GM_{n,m}}$$

та розпишемо числа x та y через їх двійкові представлення:

$$W = \sum_{i=\mathbb{0}}^{n-1} x_i \cdot (2^i \cdot F) + \sum_{j=\mathbb{0}}^{n-1} y_j \cdot (2^j \cdot G) \pmod{GM_{n,m}},$$

де $x_i, y_j \in \{\mathbb{0}, 1\}$, причому $\sum_{i=\mathbb{0}}^{n-1} x_i = h$, $\sum_{j=\mathbb{0}}^{n-1} y_j = h$. Звідси слідує, що можна спробувати відновити невідомі числа x та y через віднімання доданків $2^i \cdot F \pmod{GM_{n,m}}$ та $2^j \cdot G \pmod{GM_{n,m}}$ від W. Ідея такої реалізації функції Solve полягає в тому, що віднімання доданка $2^i \cdot F \pmod{GM_{n,m}}$ або $2^j \cdot G \pmod{GM_{n,m}}$ від W, якщо він дійсно є в сумі W, призводить до того, що значення Δ наближено дорівнюватиме $Ham(2^i \cdot F \pmod{GM_{n,m}})$ або $Ham(2^j \cdot G \pmod{GM_{n,m}})$ відповідно. У випадку, коли такого доданка немає в сумі W, значення різниці Δ буде або близьким до 0, або дорівнюватиме 0, або стане від'ємним.

На відміну від оригінальної реалізації Solve для класу чисел Мерсенна, для якого виконується співвідношення $\Delta \approx h$ і кожен біт може бути визначений незалежно, у випадку арифметики за модулем узагальненого числа Мерсенна таке співвідношення, як правило, не виконується. Як наслідок, у реалізації функції Solve для класу узагальнених чисел Мерсенна значення Δ обчислюється для всіх можливих i-их та j-их бітів чисел x та y відповідно, після чого обчислені значення порівнюються між собою, і для відновлення i-го або j-го біта обирається максимальне з них. Таким чином, функція використовує жадібну стратегію, приймаючи найкраще рішення на кожному кроці задля знаходження оптимального розв'язку в кінцевому результаті. Для цього встановлюється початкова змінна $del := W$ і для всіх i та j обчислюються

$$\Delta_{i,F} = Ham(del) - Ham(del - 2^i \cdot F \pmod{GM_{n,m}}),$$
$$\Delta_{j,G} = Ham(del) - Ham(del - 2^j \cdot G \pmod{GM_{n,m}}).$$

Далі обирається максимальне значення з усіх $\Delta_{i,F}$ та $\Delta_{j,G}$, виокремлюється відповідний йому доданок $2^i \cdot F \pmod{GM_{n,m}}$ або $2^j \cdot G \pmod{GM_{n,m}}$, припускається, що біт x_i або y_j дорівнює одиниці та оновлюється значення змінної del:

$$del := (del - \text{обраний доданок}).$$

Таблиця 1. Експериментальні результати ефективності реалізації функції Solve

Параметри	Критерії	Розмір вибірки					
		10		**100**		**1000**	
		x	y	x	y	x	y
$n = 1279$, $m = 512$, $h = 17$.	кількість повністю відновлених чисел	2	2	5	5	53	55
	ймовірність успіху	0.2	0.2	0.05	0.05	0.053	0.055
	середня кількість правильно відновлених бітів	1261.5	1260.1	1253.65	1253.15	1254.268	1253.822
	частка відновлених зайвих 1	0.514	0.549	0.755	0.747	0.730	0.732
	частка пропущених 1	0.529	0.541	0.761	0.745	0.736	0.732

Аналогічно на кожній новій ітерації циклу обчислюються $\Delta_{i,F}$ та $\Delta_{j,G}$ для всіх i, j, які ще не були використані в del, обирається найбільше значення, після чого припускається, що відповідний i-й або j-й біт в x або y дорівнює одиниці, оновлюється del, і так доти, доки del не стане рівним нулю, або доки не знайдеться потрібна кількість одиниць. В результаті поступово відновлюються біти x_i та y_j, а тому, відповідно, і шукані числа x, y. Зауважимо, що якщо максимум з усіх значень $\Delta_{i,F}$ дорівнює максимуму з усіх значень $\Delta_{j,G}$ під час однієї ітерації циклу відновлення невідомого біта, то одне з цих максимальних значень обирається випадковим чином, після чого виконується стандартне оновлення del та встановлення невідомого біта числа x або y відповідно до обраного максимума.

Для оцінки ефективності описаної вище реалізації функції Solve було проведено серію експериментів на випадково згенерованих даних. Тестування виконувалось для різних параметрів n, m та h, причому для кожного набору параметрів було згенеровано вибірки, які містять 10, 100 та 1000 тестових наборів даних. Для дослідження були використані такі метрики, як кількість повністю відновлених значень x та y, ймовірність повного відновлення числа, середня кількість правильно відновлених бітів, а також частка зайвих і пропущених одиниць. Отримані результати експерименту для параметрів $n = 1279$, $m = 512$ та $h = 17$ наведені в Таблиці 1.

При аналізі отриманих результатів встановлено, що така реалізація функції Solve в середньому забезпечує близько 98% правильно відновлених бітів, однак ймовірність повного відновлення шуканого значення x або y залишається відносно невисокою та зменшується зі збільшенням значень вхідних параметрів n, m та h. Основна проблема побудованої функції Solve пов'язана зі встановленням позицій одиничних бітів, що призводить до появи значної частки пропущених одиниць у відновлених бітових рядках.

Таким чином, результати експерименту підтверджують ефективність наближеного відновлення шуканих значень, однак наведена реалізація потребує подальшого дослідження з використанням різних значень вхідних параметрів та впровадженням технології бектрекінгу для підвищення кількості повністю відновлених значень.

Висновки

У роботі виконано дослідження постквантового механізму інкапсуляції ключа AJPS-KEM$_1$, що ґрунтується на використанні класу чисел Мерсенна як модуля. Задля збільшення пропускної здатності в AJPS-KEM$_1$ при декапсуляції використовується функція Solve. У роботі побудовано модифікацію механізму інкапсуляції ключа AJPSGM-KEM$_1$ шляхом заміни класу чисел Мерсенна для арифметики за модулем на клас узагальнених чисел Мерсенна та розроблено реалізацію функції Solve для узагальненого числа Мерсенна. Виконано експериментальне дослідження ефективності побудованої функції Solve. Отримані результати підтверджують можливість коректного перенесення схеми AJPS-KEM$_1$ на клас узагальнених чисел Мерсенна, що дозволяє збільшити варіативність параметрів криптопримітиву.

Перелік використаних джерел

1. *Diffie W.*, *Hellman M.* New Directions in Cryptography. — IEEE Transactions on Information Theory, 1976. — URL: https://ee.stanford.edu/~hellman/publications/24.pdf.
2. *Shor P.* Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. — SIAM Journal on Scientific, Statistical Computing, 1997. — URL: https://arxiv.org/abs/quant-ph/9508027v2.
3. Post-Quantum Cryptography Standardization. — National Institute of Standards and Technology. — URL: https://csrc.nist.gov/pqc-standardization.
4. A New Public-Key Cryptosystem via Mersenne Numbers / D. Aggarwal, A. Joux, A. Prakash, M. Santha. — IACR Cryptology ePrint Archive, 2017. — URL: https://eprint.iacr.org/2017/481.
5. *Ferradi H.*, *Naccache D.* Integer Reconstruction Public-Key Encryption. — IACR Cryptology ePrint Archive, 2017. — URL: https://eprint.iacr.org/2017/1231.
6. *Yadukha D.* The Modification of the Quantum-Resistant AJPS-1 Cryptographic Primitive. T. 4. — Theoretical and Applied Cybersecurity, 2022. — URL: https://tacs.ipt.kpi.ua/article/view/274116.