**УДК 004.056.5** **DOI:10.30837/rt.2023.2.213.01**

*Я.А. ДЕРЕВ'ЯНКО, О.Г.КАЧКО, канд. техн. наук, І.Д. ГОРБЕНКО, д-р техн. наук*

# КРИПТОГРАФІЯ НА ОСНОВІ ГЕШУ, ЇЇ ЗАХИЩЕНІСТЬ ТА ДОЦІЛЬНІСТЬ ЗАСТОСУВАННЯ У СУЧАСНИХ КРИПТОСИСТЕМАХ

## Вступ

ЕП на основі гешу є одним із найбільш перспективних класів криптографічних схем, які вважаються квантово стійкими. Стійкість криптографічних геш функцій є одним з найважливіших аспектів забезпечення захищеності схем на основі гешу, тому для таких функцій вимагається наявність стійкості до відновлення першого та другого прообразу, а також стійкість до колізій.

Зазвичай виділяють три категорії цифрових підписів на основі гешу: одноразові підписи (OTS), де будь-який ключ підпису слід використовувати не більше одного разу; багаторазові підписи (MTS), які поєднують кілька екземплярів OTS, щоб забезпечити обмежену кількість підписів; декількаразові підписи (FTS), у яких безпека однієї пари ключів повільно знижується з кількістю використань.

Оскільки класичні ЕП на основі гешу вимагають відстеження кількості використаних підписів, довгий час вони вважалися такими, що мають стан. Це обмеження було подолано схемою SPHINCS, яку згодом було вдосконалено до SPHINCS+.

У роботі наводиться оцінка захищеності ЕП на геш функціях відносно атак бічними каналами. Також у роботі проводиться аналіз рекомендацій щодо використання одного з кандидатів конкурсу NIST, який базується на геш криптографії – SPHINCS+, і робляться висновки щодо доцільності його використання.

## 1. Оцінка захищеності ЕП на основі гешу відносно атак бічними каналами

### 1.1. Припущення при аналізі бічних каналів

Для забезпечення аналізу бічних каналів незалежно від фактичної реалізації робляться наступні припущення [1]:

1. Припускається, що атакована реалізація використовує PRNG для створення ключів підпису W-OTS+ на льоту під час створення підпису та обчислення шляху автентифікації.

2. Припускається, що як реалізація PRNG, так і односпрямована функція взагалі не мають витоку бічним каналом (у наступних пунктах буде розглянуто, де саме застосовуються ВЧ у реалізації геш алгоритму SPHINCS+).

### 1.2. W-OTS+

*Часові бічні канали.*

Єдиними секретними даними, які обробляються в W-OTS+, є частини приватного ключа $x_i$. На рис. 1 показані актуальні для аналізу бічних каналів частини W-OTS+.

$x_i$ використовуються лише як вхідні дані для ланцюгової функції $c_k$. Ланцюгова функція застосовує геш функцію $f_k$ кілька разів до частин приватного ключа $x_i$. Під час генерації ключа кількість геш викликів є фіксованою ($w-1$), а під час генерації підпису кількість геш викликів залежить виключно від блоків повідомлень ($b_i$) [2]. Блоки $b_i$ залежать лише від геш значення, яке не представляє інтересу для потенційного зловмисника. Згідно з припущен-

ням 1, сама геш функція не містить часового бічного каналу, отже, оцінка $c_k(x_i, r)$ не може спричинити витік по часу інформації щодо $x_i$.

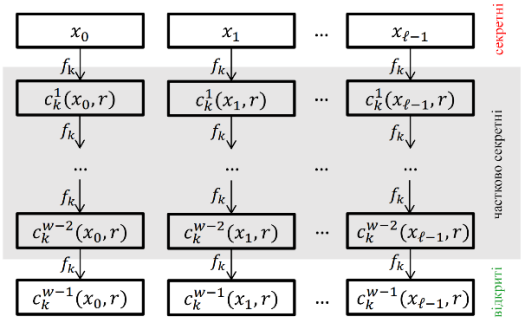


Рис. 1. Частини W-OTS, актуальні для аналізу бічних каналів

Якщо $b_i = 0$, це означає, що зловмисник знає частину приватного ключа, проте навіть якщо все геш значення повідомлення, яке підписується, дорівнюватиме 0, контрольна сума все одно гарантуватиме, що деякі $b_i$ більші за нуль.

Отже, часовий бічний канал в W-OTS+ можна використовувати лише для вилучення інформації про $b_i$-ті, яку зловмисник або вже знає (атака з відомим повідомленням), або обирає сам (атака з обраним повідомленням).

*Бічні канали потужності.*

Цікавою функцією в W-OTS+ є ланцюгова функція:

$$00 \tag{1}$$

Така функція здається ідеальним варіантом для аналізу потужності, тому що для $i = 1$ підписувач обчислює $x \oplus r$, де $x$ – деякий блок приватного ключа, а $r_i$ – бітова маска рандомізації, відома верифікатору та зловмиснику. Однак ця функція викликається лише двічі: один раз під час генерації ключа та один раз під час генерації підпису. Крім того, $r_i$ є однаковим для обох оцінок. Це запобігає диференціальним атакам, таким як DPA, які покладаються на різні вхідні дані атакованого примітиву. Крім того, атаки SPA можуть у найкращому випадку відновити HW (Вагу Хемінга) оброблених значень. Формула середнього витоку таким чином матиме наступний вигляд [1]:

$$leak_{avg} = \sum_{i=0}^{8} \frac{1}{256} \binom{8}{i} \left( 8 - \log_2 \binom{8}{i} \right) = 2.5 bits \,. \tag{2}$$

Крім того, якщо станеться витік ваги Хеммінга лише 32-розрядних слів, що більш імовірно, витік зменшиться приблизно до 3,5 бітів на слово, тобто 28 бітів на 256-бітний ключ.

### 1.3. XMSS

*Часові бічні канали.*

В попередньому пункті зроблено висновок, що W-OTS+ не спричиняє витоку жодної інформації про частини приватного ключа через часові бічні канали. Оскільки XMSS побудовано з використанням багатьох ключів W-OTS+, XMSS також забезпечує цю стійкість.

Актуальні для аналізу частини показано на рис. 2, до них відносяться початкове значення (seed), яке використовується для псевдовипадкової генерації приватного ключа W-OTS+, і сам приватний ключ W-OTS+.

Таким чином, приватний ключ у XMSS або складається з багатьох приватних ключів W-OTS+, або з випадкового початкового числа, яке використовується для їх створення [3]. При використанні першого варіанту стійкість часового каналу XMSS безпосередньо випли-

ває зі стійкості W-OTS+. При використанні останнього варіанту також потрібно зробити припущення, що PRNG не спричиняє витоку про випадкове початкове число [1]. Це гарантує, що принаймні W-OTS+ частина генерації підпису і обчислення відкритого ключа XMSS не має часового бічного каналу, який можна використовувати.

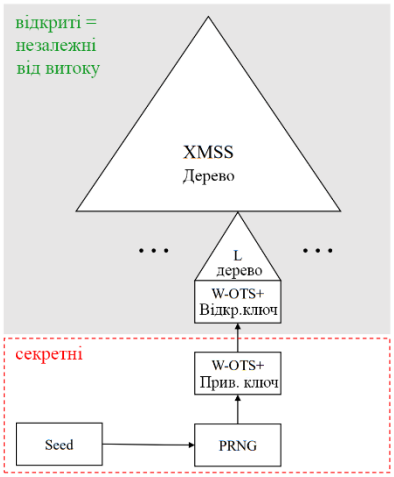


Рис. 2. Частини XMSS, які підходять для аналізу бічних каналів

*Бічні канали потужності.*

Стійкість W-OTS+ до бічних каналів потужності не означає стійкість XMSS, оскільки у XMSS генерація ключа W-OTS+ викликається набагато частіше під час обчислення шляху автентифікації.

Нехай дерево XMSS має висоту $H$, тобто має $2^H$ листів. Для підпису з використанням приватного ключа W-OTS+ з індексом $s$ підписувачу потрібно спочатку обчислити підпис W-OTS+ за допомогою $sk_{OTS,s}$, а потім – шлях автентифікації для $v_0[s]$. Якщо припустити, що підписувач взагалі не використовує вузли повторно, знаємо, що під час генерації підпису для індексу $s$, $sk_{OTS,s}$ вже витікав кілька разів раніше. Витік відбувся один раз під час початкової генерації ключа та один раз для кожного підпису, який був створений раніше, що означає, що включно з поточним витоком він витік $s+2$ рази, а також витік буде відбуватися для всіх генерацій підпису в майбутньому ($2^H+1$ разів, якщо використовуються всі ключі).

У випадку реальної моделі витоку атака стає нездійсненною: під час кожної генерації підпису ланцюгова функція W-OTS+ викликається з абсолютно однаковими вхідними даними для створення однакових відкритих ключів W-OTS+. Зловмисник не може сподіватися відновити більше, ніж вагу Хеммінга кожного проміжного результату [1].

Згідно з представленими припущеннями, W-OTS+ і XMSS забезпечують сильну стійкість щодо атак бічними каналами. Показано, що W-OTS+ стійкий до більшості атак через його одноразовий характер, що обмежує кількість слідів, які можна отримати. Ця стійкість послаблюється генерацією ключів на льоту в XMSS. Отже алгоритми є стійкими, якщо використовувана геш функція та PRNG є стійкими до витоку [4].

### 1.4. LD-OTS, W-OTS і MSS

*Часові бічні канали.*

Обчислення відкритого ключа як LD-OTS, так і W-OTS мають постійну кількість обчислень геш функції. Тому часовий бічний канал у даному випадку відсутній. Генерація підпису для LD-OTS має фіксовану кількість геш викликів (половина обчислення відкритого ключа), що також означає, що жодна атака за часом не може бути проведена [1]. Час виконання генерації підпису W-OTS залежить лише від повідомлення, яке потрібно підписати, і тому не

призводить до витоку будь-якої цінної інформації для зловмисника. Для MSS можна використовувати ті ж міркування, що й для XMSS.

*Бічні канали потужності.*

LD-OTS і W-OTS не роблять нічого, крім застосування геш функції до частин приватного ключа. Бітові маски в W-OTS+, які спричиняють незначний бічний канал потужності, відсутні в попередніх схемах. Таким чином, геш функція стійка до бічних каналів, є достатньою для гарантування стійкості щодо атак бічними каналами LD-OTS і W-OTS. MSS не виконує жодних обчислень із секретними даними, крім базового OTS. Генерація ключа на льоту може бути корисною для зловмисника, щоб зменшити шум, однак атака виглядає так само малоймовірною, як і для XMSS [1]. Очевидно, що якщо використовується PRNG, він має бути стійким до атак бічними каналами.

Таким чином, можна зробити висновок, що представлені схеми також природним чином протистоять розглянутим атакам бічними каналами.

### 1.5. LMS

*Часові бічні канали.*

Дивлячись на ланцюгову функцію $c^j(S,x,i)$, бачимо, що вона застосовує лише геш функцію $H$. Очевидно, що час цієї ланцюгової функції не спричиняє витоку цінної інформації, оскільки він залежить лише від значення, яке не є секретним. Конструкція LMS подібна до XMSS лише з невеликими варіаціями у використанні констант і елементів рандомізації. Оскільки всі вони загальнодоступні, додаткові часові бічні канали не можуть виникати [1].

*Бічні канали потужності.*

Включені $S$ (постійний рядок рандомізації), $i$ (індекс геш ланцюжка), $j$ (індекс у геш ланцюжку) і D_IT ER (константа 0x00), на перший погляд, надають додатковий бічний канал потужності, коли відкриті ключі LM-OTS обчислюються на льоту та кілька разів. Однак значення є фіксованими і тому не підходять для атаки з аналізом потужності [1]. Крім того, вони передаються лише як вхідні дані для геш функції, яка в представленому аналізі вважається стійкою до витоку. Те саме міркування стосується LMS: усі проміжні значення в деревах Merkle є загальнодоступними і, таким чином, не залежать від витоку.

Можна зробити висновок, що LMS за своєю суттю захищена від атак бічними каналами за часом та атак аналізу потужності, якщо реалізації PRNG і геш функції є стійкими до атак бічними каналами.

### 1.6. SPHINCS

*Часові бічні канали.*

Кількість обчислень геш функції як для генерації ключа HORST, так і для генерації підпису HORST не залежить від приватного ключа. Таким чином, якщо геш функція не містить часового бічного каналу, можна з упевненістю зробити висновок, що HORST стійкий до часового бічного каналу. Решта конструкції SPHINCS використовує лише W-OTS+ і XMSS і, таким чином, не може містити часовий бічний канал [1].

*Бічні канали потужності.*

Листя дерева HORST будуються з випадково згенерованих частин секретного ключа $sk_i$ шляхом застосування геш функції $F$. Тому HORST є стійким до бічних каналів, якщо $F$ є стійкою до бічних каналів, оскільки все інше можна вважати відкритим. Поєднуючи цю стійкість із міркуваннями щодо W-OTS+ і XMSS, можна бути впевненим, що SPHINCS має лише незначний витік бічними каналами, якщо використано стійкі геш функції та PRNG [1].

## 2. Атаки бічними каналами та оцінка захищеності SPHINCS+

SPHINCS+ – це схема електронного підпису на основі гешу, обрана NIST у процесі стандартизації постквантової криптографії. У даному пункті буде проаналізовано можливі атаки бічними каналами на даний алгоритм, а також надано оцінку контрзаходів від таких атак.

### 2.1. Захист SPHINCS+ від атак помилками

Перша подібна атака на SPHINCS+ розглядається у роботі Castelnovi та ін. [5]. У ній пропонується атака помилками на фреймворк, що лежить в основі SPHINCS, GRAVITY_SPHINCS та SPHINCS+. Представлена атака дозволяє підробити будь-який підпис повідомлення ціною одного помилкового повідомлення. Атака є загальною в тому сенсі, що вона не залежить від використовуваних геш функцій.

Автори стверджують, що атака дозволяє підробити підписи для будь-якого одного помилкового повідомлення за вартістю $2^{34}$ гешу за повідомлення. А для будь-якої цільової схеми можна підробити будь-яке повідомлення вартістю приблизно $2^{20}$ гешів, знаючи лише три помилкові повідомлення. Як показано в цій роботі, детермінована природа підписів на основі гешу та їхнє внутрішнє використання OTS може бути слабкою стороною проти атак помилками.

У роботі [5] представлено атаку помилками, яка змушує пару ключів W-OTS+ підписати пошкоджене повідомлення шляхом введення помилки під час побудови будь-якого неверхнього піддерева. Разом із дійсним (тобто безпомилковим) підписом піддерева, отриманий помилковий підпис W-OTS+ використовується для компрометації відповідної пари ключів W-OTS+ під час атаки з двома повідомленнями та надання дійсного підпису для іншого піддерева, для якого секрети відомі. Цей процес, подібний до щеплення дерева (живцювання), дає змогу підробити загальний підпис для будь-якого повідомлення.

Представлена у роботі атака має такі кроки:

1. Збирання підписів. На першому етапі зловмиснику необхідно зібрати як дійсні, так і помилкові підписи SPHINCS+ з цільового пристрою:

2. Обробка помилкових підписів. Наступним кроком є обробка помилкових підписів SPHINCS+, щоб отримати інформацію, яка уможливлює універсальну підробку.

3. «Щеплення» (живцювання) дерев. Після того, як найбільш секретні значення скомпрометованої пари ключів W-OTS+ були успішно вилучені з помилкових підписів, зловмисник прагне «прищепити» піддерево (або «ліс») до вилученої верхньої частини, тобто знайти інший XMSS (або FORS), для якого дійсний підпис W-OTS+ можна підробити, щоб підмінити скомпрометований екземпляр за його власною адресою.

4. Пошук шляху. Процедура підпису SPHINCS+ слідує шляху в гіпердереві залежно від повідомлення та значення $R$. У результаті зловмиснику необхідно знайти адекватне значення $R$, яке змусило б підроблений підпис відвідувати скомпрометоване піддерево.

Отримані у роботі результати вказують на те, що атака помилками можлива в усіх сценаріях, хоча кількість необхідних гешів значно змінюється залежно від конкретного рівня, на який спрямована атака. Однак, незважаючи на те, що надані цифри здаються високими, загальна кількість необхідних гешів може бути досягнута на практиці. Цей результат особливо важливий, оскільки атака помилками може бути успішною, навіть якщо помилка неконтрольована.

Робота Castelnovi та ін. [5] була додатково покращена та допрацьована Aymeric Genêt [6]. У цій роботі надано адаптовану оригінальну атаку на SPHINCS+, посилену рандомізованим підписом, і розширено застосовність атаки до будь-якої комбінації помилкових і дійсних підписів [6].

Припустимо, що помилка може вразити будь-який виклик геш функції рівномірно випадковим чином. Наведені вище перерахування призводять до наступних ймовірностей:

можливість використання помилки:

$$P(Expl.) = \frac{\#Total^F + (d-1)\cdot \#Total^X}{\#Total} = \frac{k(3t-1)+1+(d-1)(2^{h'}(lW+2)-1)}{3+k(3t-1)+d(2^{h'}(lW+2)-1)}; \quad (3)$$

можливість перевірки помилки:

$$P(Verif.) = \frac{1+\#Verif^F + (d-1)\cdot \#Verif^X}{\#Total} = \frac{1+k(3t-a-3)+(d-1)((2^{h'}-1)(lW-1)+2^{h'}-h'-1)}{3+k(3t-1)+d(2^{h'}(lW+2)-1)} \quad (4)$$

Влучання в шар:

$$P(L=l^*) = \begin{cases} \dfrac{\#Total^F}{\#Total} = \dfrac{k(3t-1)+1}{3+k(3t-1)+d(2^{h'}(lW+2)-1)} & \text{if } l^*=0 \\ \dfrac{\#Total^X}{\#Total} = \dfrac{2^{h'}(lW+2)-1}{3+k(3t-1)+d(2^{h'}(lW+2)-1)} & \text{if } 1\le l^*\le d \end{cases} \quad (5)$$

У табл. 1 надано ймовірності з урахуванням усіх наборів параметрів SPHINCS+. Ця таблиця показує високу ймовірність того, що випадкова помилка призведе до помилкового підпису, який можна використовувати та підтвердити:

Таблиця 1

Результати аналізу помилок для всіх параметрів SPHINCS+

| | $P(Expl.)$ | $P(Verif.)$ | $P(L=l^*)$ | | | | |
|---|---|---|---|---|---|---|---|
| Набір | | | $l^*=0$ | 1 | … | $d-1$ | $d$ |
| 128s | 0.9326 | 0.9306 | 0.4607 | 0.0674 | … | 0.0674 | 0.0674 |
| 128f | 0.9669 | 0.8857 | 0.3387 | 0.0331 | … | 0.0331 | 0.0331 |
| 192s | 0.9527 | 0.9513 | 0.6216 | 0.0473 | … | 0.0473 | 0.0473 |
| 192f | 0.9613 | 0.8576 | 0.1495 | 0.0387 | … | 0.0387 | 0.0387 |
| 256s | 0.9162 | 0.9138 | 0.3296 | 0.0838 | … | 0.0838 | 0.0838 |
| 256f | 0.9553 | 0.9095 | 0.2398 | 0.0447 | … | 0.0447 | 0.0447 |

### 2.2. Контрзаходи проти атаки помилками

*Кешування шарів.*

Ця стратегія, спочатку запропонована в Gravity-SPHINCS [7], полягає в кешуванні всіх W-OTS+ в межах одного або кількох рівнів (шарів) (починаючи з верхнього рівня). Оскільки кеш не оновлюється новими запитами на підпис, кеш є статичним і тому його можна додати до відкритого ключа.

У табл. 2 показано, як імовірність того, що одна випадкова помилка є придатною для використання, зменшується зі збільшенням $c$ (кількість рівнів, що кешується) для всіх наборів параметрів SPHINCS+ [6]:

Таблиця 2

Аналіз контрзаходу кешування шарів

| Набір | $P(Expl.)$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | $c=1$ | 2 | 3 | 4 | … | $d-1$ | $d$ |
| 128s | 0.8972 | 0.8591 | 0.8179 | 0.7733 | … | 0.6141 | 0.0000 |
| 128f | 0.9505 | 0.9335 | 0.9158 | 0.8975 | … | 0.5076 | 0.0000 |
| 192s | 0.9287 | 0.9034 | 0.8767 | 0.8486 | … | 0.7539 | 0.0000 |
| 192f | 0.9420 | 0.9218 | 0.9007 | 0.8787 | … | 0.2625 | 0.0000 |
| 256s | 0.8711 | 0.8216 | 0.7670 | 0.7066 | … | 0.4784 | 0.0000 |
| 256f | 0.9327 | 0.9090 | 0.8840 | 0.8578 | … | 0.3864 | 0.0000 |

*Кешування гілок.*

Ця стратегія полягає в кешуванні всіх підписів W-OTS+ і відкритих ключів у шляху під час процедури підписання [6]. Кеш є динамічним і може вимагати оновлення для кожного нового підпису.

У табл. 3 показано, як імовірність того, що випадкову помилку можна використати, зменшується з $b$ для всіх наборів параметрів SPHINCS+, припускаючи, що всі кеші заповнені до необхідної ємності:

Таблиця 3

Аналіз контрзаходу кешування гілок

| Набір | $P(Expl.)$ | | | | | |
|---|---|---|---|---|---|---|
| | $b = (2/3)2^{h'}$ | $(2/3)2^{2h'}$ | $(2/3)2^{3h'}$ | $(2/3)2^{4h'}$ | … | $(2/3)2^{dh'}$ |
| 128s | 0.9292 | 0.9238 | 0.9174 | 0.9098 | … | 0.3172 |
| 128f | 0.9647 | 0.9634 | 0.9620 | 0.9605 | … | 0.3219 |
| 192s | 0.9511 | 0.9485 | 0.9457 | 0.9425 | … | 0.3249 |
| 192f | 0.9585 | 0.9568 | 0.9549 | 0.9528 | … | 0.3052 |
| 256s | 0.9111 | 0.9023 | 0.8917 | 0.8785 | … | 0.3068 |
| 256f | 0.9530 | 0.9507 | 0.9481 | 0.9453 | … | 0.3130 |

Результати експериментів показали, що, не зважаючи на контрзаход, лише $2^9$ запитів підпису з імовірністю помилки $\approx 1/3$ достатньо, щоб скомпрометувати принаймні один W-OTS+ і, отже, виконати універсальну підробку SPHINCS+ [6].

Основний висновок аналізу полягає в тому, що SPHINCS+ є надзвичайно вразливим до помилок. Єдине необмежене пошкодження майже будь-якого обчислення має катастрофічний вплив на гарантії безпеки всіх наборів параметрів SPHINCS+.

### 2.3. Аналіз атаки Антонова на DM-SPR

У 2022 р. Сідней Антонов на форумі NIST PQC описав атаку на властивість DM-SPR геш функцій на основі SHA-256, які використовуються в SPHINCS+ [8]. Атака використовує конструкцію Merkle-Damgard SHA-256, використовуючи серію колізійних атак проти основної функції стиснення, щоб перетворити різнофункціональну багатоцільову атаку другого прообразу в однофункціональну багатоцільову атаку другого прообразу. На рис. 3 показано приклад того, як працює атака:

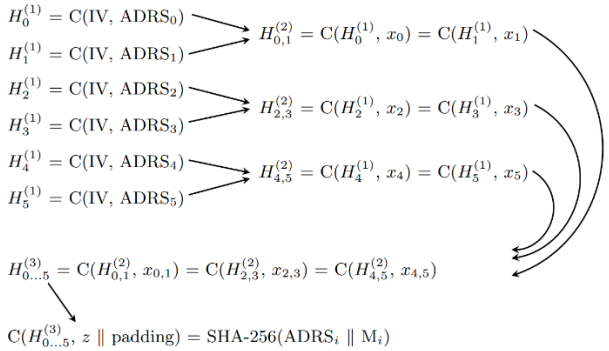


Рис. 3. Приклад роботи атаки Антонова

У відповіді йдеться про те, що автори вирішили замінити функцію SHA-256 на SHA-512 для рівнів безпеки NIST 3 та 5.

Автори не вважають, що ця атака ставить під сумнів загальну надійність конструкції SPHINCS+. У поєднанні з попередніми спостереженнями на форумі PQC щодо слабких місць у безпеці 5 рівня в гешуванні повідомлень [9] стає очевидним, що і атака Антонова, і атака з

роботи [10] стали можливими завдяки спробі дизайнерів SPHINCS+ використати 256-бітний Merkle-Damgard геш, як-от SHA-256, щоб загалом отримати 256 біт безпеки.

Оскільки після реагування на зауваження щодо безпеки для 3 та 5 рівнів безпеки ця функція тепер використовує SHA-512 замість SHA-256. Ця зміна означає, що подібна атака тепер потребує щонайменше $2^{256}$ обчислень геш функції, що ефективно блокує як атаку Антонова, так і атаку з роботи [10].

### 3. Аналіз рекомендацій щодо геш криптографії

Як описано раніше, у більшості випадків використання електронних підписів OTS не підходить, оскільки потрібно підписати та перевірити багато різних повідомлень. Для подолання цього недоліку були запропоновані схеми для створення багаторазових підписів з використанням OTS як будівельного блоку. Далі буде надано аналіз існуючих рекомендацій NIST, аналіз алгоритму, що пропонується до реалізації – SPHINCS+ – на основі відповідності до вимог NIST, а також аналіз швидкодії та захищеності різних варіантів SPHINCS+ [11]. На основі проведеного аналізу будуть представлені рекомендації щодо вибору реалізації.

#### 3.1. Аналіз існуючих рекомендації NIST

Рекомендація NIST (SP 800-208) [12] визначає 2 алгоритми, що можуть застосовуватися для створення ЕП:

- the Leighton-Micali Signature (LMS);
- the eXtended Merkle Signature Scheme (XMSS).

Фактично стандарт встановлює значення параметрів і які геш функції можна застосовувати.

Дослідження NIST 800-208 [11] показало наступне:

1. Публікація виступає у ролі рекомендації з реалізації алгоритмів ЕП «зі станом», серед яких LMS та XMSS разом з їх варіантами з кількома деревами.

2. У публікації описується застосування WOTS+ та дерев Меркла, які використовуються у реалізації SPHINCS+, що робить SPHINCS сумісним та таким, що відповідає усім критеріям даної публікації.

3. У публікації разом з XMSS та $XMSS^{MT}$ схвалюється застосування 4 різних геш функцій SHA-256, SHA-256/192, SHAKE256/256 та SHAKE256/192. В той час SHA-256 та SHAKE256 застосовуються у SPHINCS+, що робить його сумісним.

4. Для SPHINCS+ (sphincs-sha256-256f) ці параметри мають наступні значення: $n=32$, $w=16$, $len=67$, $h=68$ та $d=17$. Отже такий набір параметрів буде відповідати вимогам NIST (SP 800-208).

5. 192-розрядні геш функції, описані в цій рекомендації, SHA-256/192 та SHAKE256/192, пропонують значно меншу стійкість до загальних пошуків колізій, ніж їх 256-розрядні аналоги. Тому, реалізація SPHINCS+ із застосуванням SHA-256 є виправданою.

#### 3.2. Порівняння варіантів оптимізації ЕП

Далі буде надано порівняння різних варіантів оптимізації (закінчується на «s» – варіант з оптимізацією за пам'яттю, закінчується на «f» – варіант з оптимізацією за часом) [13]. Оцінка буде проводитися шляхом порівняння спочатку розмірів ЕП обох реалізацій, а потім швидкістю виконання підписання та перевірки, тобто всіх необхідних процедур алгоритму. У таблиці 4 предсталено порівняння розмірів ЕП для sphincs-256f та sphincs-256s:

Таблиця 4

Порівняння розмірів ЕП для 5 рівня безпеки NIST

| Варіант реалізації | Розмір PK | Розмір SK | Розмір підпису |
|---|---|---|---|
| sphincs-256f | 64 | 128 | 49856 |
| sphincs-256s | 64 | 128 | 29792 |

У табл. 5 надається порівняння швидкодії виконання різних варіантів. Тестування проводилося на процесорі i5-13600KF [14] із частотою 5.1ГГц і 32ГБ ОЗП.

Таблиця 5

Порівняння швидкодії для 5 рівня безпеки NIST

| Процес | Підпроцес | Кількість циклів на одне виконання | Кількість виконань | Загальна кількість циклів |
|---|---|---|---|---|
| sphincs-sha256-256f-robust | | | | |
| Генерація ключової пари | | 29,311,292 | 1x | 29,311,292 |
| - | Генерація WOTS pk | 1,612,542 | 16x | 25,800,672 |
| Підписання | | 701,347,571 | 1x | 701,347,571 |
| - | Підписання FORS | 108,020,496 | 1x | 108,020,496 |
| - | Підписання WOTS | 58,969 | 17x | 1,002,473 |
| - | Генерація WOTS pk | 1,594,614 | 272x | 433,735,008 |
| Перевірка | | 17,030,357 | 1x | 17,030,357 |
| sphincs-sha256-256s-robust | | | | |
| Генерація ключової пари | | 522,514,794 | 1x | 522,514,794 |
| - | Генерація WOTS pk | 1,626,626 | 256x | 416,416,256 |
| Підписання | | 6,736,001,820 | 1x | 6,736,001,820 |
| - | Підписання FORS | 2,282,116,657 | 1x | 2,282,116,657 |
| - | Підписання WOTS | 56,918 | 8x | 455,344 |
| - | Генерація WOTS pk | 1,611,667 | 2048x | 3,300,694,016 |
| Перевірка | | 9,143,489 | 1x | 9,143,489 |

З результатів, наведених у таблицях та на рисунках, видно, що хоча і варіант sphincs-256s має майже вдвічі менший розмір ЕП, проте різниця у швидкості роботи алгоритмів є дуже значною. Це вказує на те, що покращення розміру є не дуже доцільним, оскільки, зважаючи на сучасний стан розвитку технологій, час є більш цінним ресурсом ніж обсяг пам’яті.

### 3.3. Порівняння режимів роботи з різними параметрами

Варіант robust застосовує бітову маску, в той час як simple не застосовує. Оскільки представлені у NIST 800-208 алгоритми (LMS та XMSS) її застосовують, то з точки зору узгодження режим robust є більш підходящим.

### 3.4. Застосування псевдовипадкових чисел у реалізації

Детальний аналіз програмних реалізацій та документації різних версії SPHINCS+ показав, що (псевдо)випадкові числа застосовуються для SPHINCS+:

1. При початковій ініціалізації (заповненні) генератора псевдовипадкових чисел, що застосовується у реалізації (AES256_CTR_DRBG), за допомогою функції randombytes_init(entropy_input, NULL, 256). Тут до функції передається початкова ентропія (в даному випадку масив entropy_input, заповнений числами від 0 до 47). У самій функції, якщо присутній рядок персоналізація (у даному випадку рядок персоналізації = NULL), то цей рядок поелементно поєднується з seed_material (який і буде вхідною ентропією) шляхом XOR. Далі з використання функції AES256_CTR_DRBG_Update оновлюються (або ініціалізуються при початковому використанні) значення Key та V для DRBG_ctx.

2. При генерації початкового значення seed, а також при генерації повідомлення для підписання з використанням функції randombytes. Функція генерує seed довжини 48 байтів і

повідомлення довжини mlen. Також у функції AES256_CTR_DRBG_Update оновлюються значення Key та V для DRBG_ctx.

3. При оновленні (повторному заповненні) генератора псевдовипадкових чисел з використанням функції randombytes_init та згенерованого раніше seed. Всередині функції з використанням переданого seed оновлюються значення Key та V для DRBG_ctx із застосуванням AES256_CTR_DRBG_Update для подальшої генерації seed для застосування в процесі створення ключової пари.

4. Всередині функції crypto_sign_keypair при генерації seed із застосуванням функції randombytes для використання в процесі створення ключової пари. На цьому кроці генерується seed довжини CRYPTO_SEEDBYTES, яка залежить від вихідної довжини гешу у байтах SPX_N. SPX_N в свою чергу залежить від криптостійкості обраного варіанту реалізації алгоритму. Згенерований seed застосовується для формування ключів (до складу ключів входять частини seed, а також обчислений корінь геш дерева).

5. У процедурі підписання при ініціалізації випадкового значення optrand з використанням randombytes, якщо необхідно зробити процедуру підписання недетерміністичною. Довжина optrand дорівнює SPX_N і залежить від криптостійкості обраного варіанту реалізації SK_SEED, SK_PRF та PUB_SEED у процедурі генерації підпису не генеруються повторно.

Переглядаючи .req файли реалізацій, де генерується початкове значення seed, можна бачити, що для різних варіантів оптимізації генеруються однакові значення seed на однакових кроках. Це означає, що, з точки зору випадковості, дані реалізації є абсолютно ідентичними.

Таким чином, можна переконатися що за однакової ініціалізації (початкового заповнення) генератор буде поводитися ідентичним чином, а отже реалізація PRNG є однаковою для різних версій і варіантів оптимізації.

**Висновки**

Розглянуто атаки бічними каналами і атаки помилками на криптографію на основі гешу, а також проаналізовано рекомендації щодо застосування одного з кандидатів конкурсу NIST– SPHINCS+.

Згідно з представленими припущеннями W-OTS+ і XMSS забезпечують сильну стійкість щодо атак бічними каналами. Показано, що W-OTS+ стійкий до більшості атак через його одноразовий характер, що обмежує кількість слідів, які можна отримати. Ця стійкість послаблюється генерацією ключів на льоту в XMSS. Алгоритми є стійкими, якщо використовувана геш функція та PRNG є стійкими до витоку.

Не зважаючи на це, SPHINCS+ є надзвичайно вразливим до помилок. Єдине необмежене пошкодження майже будь-якого обчислення має катастрофічний вплив на гарантії безпеки всіх наборів параметрів SPHINCS+.

До такої вразливості слід ставитися серйозно, оскільки помилки природно трапляються у звичайному обладнанні, наприклад у DRAM. Оскільки загрозу помилки не можна повністю усунути, найкращим рішенням для захисту схеми ЕП від випадкових і навмисних помилок є надлишковість. Результати, надані у роботі, показують, що усі реальні застосування SPHINCS+ необхідно виконувати з перевірками шляхом надмірності, навіть якщо сценарій використання не є схильним до помилок.

Для відповідності вимогам NIST 800-208 слід застосовувати SPHINCS+ з параметром розміру вихідного гешу у 32 байти – 256 біт разом із функцією SHA-256 або SHAKE256 для кращого забезпечення стійкості до колізій.

Гарним рішенням було б використання середовища виконання з мінімально можливою ймовірність помилки.

Щодо варіанту реалізації SPHINCS+, з точки зору інших параметрів, найкращими варіантами для реалізації будуть: для 1 рівня захисту NIST: sphincs-128f-robust та sphincs-128s-robust. Для 3 рівня захисту NIST: sphincs-192f-robust та sphincs-192s-robust. Для 5 рівня захи-

сту NIST: sphincs-256f-robust та sphincs-256s-robust. Також доцільним є забезпечення переходу від одного варіанту до іншого за допомогою файлу параметрів.

**Список літератури:**


1. Denis Butin Physical Attack Vulnerability of Hash-Based Signature Schemes. 2017. URL: https://kannwischer.eu/theses/MasterThesisMatthiasKannwischerFINAL.pdf.
2. A. Hülsing W-OTS+ – Shorter Signatures for Hash-Based Signature Schemes. 2013. URL: https://eprint.iacr.org/2017/965.pdf.
3. A. Hülsing, D. Butin, S.-L. Gazdag, A. Mohaisen XMSS: Extended Hash-based Signatures. 2020. URL: https://datatracker.ietf.org/doc/rfc8391
4. T. Eisenbarth, I. von Maurich, and X. Ye. Faster Hash-Based Signatures with Bounded Leakage. 2014. URL: https://www.researchgate.net/publication/290110020_Faster_Hash-Based_Signatures_with_Bounded_Leakage.
5. Laurent Castelnovi, Ange Martinelli, Thomas Prest Grafting trees: A fault attack against the SPHINCS framework. 2018. URL: https://eprint.iacr.org/2018/102.pdf.
6. Aymeric Genêt On Protecting SPHINCS+ Against Fault Attacks. 2023. URL: https://eprint.iacr.org/2023/042.pdf.
7. Jean-Phillippe Aumasson and Guillaume Endignoux. Gravity-SPHINCS. 2017. URL: https://github.com/gravity-postquantum/gravity-sphincs
8. Antonov S. Round 3 official comment: SPHINCS+. 2022. URL: https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/FVItvyRea28/m/mGaRi5iZBwAJ
9. Stern M. Diversity of signature schemes. 2021. URL: https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/2LEoSpskELs/m/LkUdQ5mKAwA
10. Ray Perlner, John Kelsey, David Cooper Breaking Category Five SPHINCS+ with SHA-256. 2022. URL: https://eprint.iacr.org/2022/1061.pdf
11. J. Aumasson, D. J. Bernstein, et al. SPHINCS+. Submission to the NIST post-quantum project, v.3.1. 2022. URL: https://sphincs.org/data/sphincs+-r3.1-specification.pdf
12. NIST SP 800-208. Recommendation for Stateful Hash-Based Signature Schemes. 2020. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-208.pdf
13. SPHINCS+ official web-site. NIST 3-rd Round Package. URL: https://sphincs.org/data/sphincs+-round3-submission-nist.zip
14. Офіційний сайт Intel. Процессор Intel® Core™ i5-13600KF. URL: https://www.intel.com/content/www/us/en/products/sku/230494/intel-core-i513600kf-processor-24m-cache-up-to-5-10-ghz/specifications.html


*Надійшла до редколегії 04.06.2023*


*Відомості про авторів:*

**Дерев'янко Ярослав Андрійович** - науковий співробітник-консультант АТ «Інститут інформаційних технологій», Україна; e-mail: yarik0009258@gmail.com; ORCID: https://orcid.org/0000-0002-3290-3373

**Качко Олена Григорівна** – канд. техн. наук, Харківський національний університет радіоелектроніки, професор кафедри програмної інженерії, факультет комп'ютерних наук, АТ «Інститут інформаційних технологій», начальник відділу програмування; Україна; e-mail: iit@iit.kharkov.ua; ORCID: https://orcid.org/0000-0001-9249-0497

**Горбенко Іван Дмитрович** – д-р техн. наук, професор, Харківський національний університет імені В. Н. Каразіна, професор кафедри безпеки інформаційних систем і технологій, факультет комп'ютерних наук, АТ "Інститут Інформаційних Технологій", головний конструктор, Україна; e-mail: gorbenkoi@iit.kharkov.ua; ORCID: https://orcid.org/0000-0003-4616-3449