***Список літератури***

1. Завадський І. О. Префіксний стискальний код на основі нижнього (2,3)-подання чисел / І. О. Завадський // Вісник КНУ ім. Т. Шевченка. Серія фіз.-мат. науки. – 2015. – № 1. – С. 124–129.
2. Apostolico A. Robust transmission of unbounded strings using Fibonacci representations / A. Apostolico, A. S. Fraenkel // IEEE Trans. Inform. Theory. – 1987. – Vol. 33. – P. 238–245.
3. Brisaboa N. An efficient compression code for text databases / N. R. Brisaboa, E. L. Iglesias, G. Navarro, J. R. Parama // Proc. European Conference on Information Retrieval ECIR'03, Pisa, Italy. – 2003. – LNCS 2633. – P. 468–481.
4. Brisaboa N. (S,C)-dense coding: an optimized compression code for natural language text databases / N. R. Brisaboa, A. Farina, G. Navarro, M. F. Esteller // Proc. Symposium on String Processing and Information Retrieval SPIRE'03, Manaus, Brazil. – 2003. – LNCS 2857. – P. 122–136.
5. Klein S. On the usefulness of Fibonacci compression codes / S. T. Klein, M. K. Ben-Nissan // Computer Journal. – 2010. – Vol. 53, № 6. – P. 701–716.

*I. Zavadskyi*

## MULTIDELIMITER CODES

*The new family of prefix codes is introduced and investigated. They can be efficiently applied for compressing of texts. Some of codes in this family are 10–30 % closer to Shannon entropy limit, they are faster by 20 % and use memory more efficiently in times than the best known code in Fibonacci codes family.*

**Keywords:** data compressing, encoding, delimiter, prefix code, Fibonacci code, multidelimiter code.

*Матеріал надійшов 06.04.2015*

**УДК 004.056.5**

*Гончар С. А.*

# КРИПТОГРАФІЯ З ЧАСОВИМ РОЗКРИТТЯМ: ПЕРСПЕКТИВИ РОЗВИТКУ ДЛЯ БАГАТОЯДЕРНИХ СИСТЕМ

*У статті розглянуто можливість використання такого напряму криптографії з часовим розкриттям, як часові замки, для сучасних багатоядерних систем та наведено результати експериментальних досліджень, які показали, що кількість часових замків, процеси розкриття яких проходять одночасно, має бути на одиницю меншою за кількість ядер у процесорі. Новизною цього дослідження також є те, що разом з портативними системами дослідження проводились і на мобільній системі (смартфон).*

**Ключові слова:** криптографія з часовим розкриттям, часові замки, багатоядерні процесори.

## Вступ

Комп'ютерні системи з плином часу набувають усе більшого розвитку. Нині в них використовують високопродуктивні складові, про які ще зовсім недавно можна було тільки мріяти: багатоядерні процесори, оперативну пам'ять, яка забезпечує більш швидке оброблення операцій читання/запису процесором, а також більш продуктивно обробляє виконуваний машинний код, графічні

© *Гончар С. А., 2015*

карти, які можна використовувати не тільки для графічних, а й для обчислювальних задач. Одним із нововведень сучасності є впровадження повноцінних операційних систем для мобільних пристроїв, а також використання багатоядерних систем для них. Усе це дає змогу значно розширити та пришвидшити виконання старих задач, що може бути досягнуто шляхом їх розпаралелювання або ж шляхом модифікації схем виконання відповідно до нової архітектури систем. У випадку з мобільними пристроями всі задачі, які дотепер не могли виконуватись на мобільних пристроях, можуть бути перенесені на них.

Криптографія з часовим розкриттям не є винятком. Нині вона є цікавим та перспективним напрямом розвитку технології захищеної передачі даних, а з використанням сучасних технологій у нас є можливість вдосконалити наявні схеми та досягти більш високої продуктивності в порівнянні з наявною. Сам підхід бере свій початок ще з першої найвизначнішої роботи за цим напрямом Рівеста та співавторів, де ними було висунуто концепцію так званого «time-released crypto», або криптографії з часовим розкриттям, яка базувалась на тому, що одержувач до певного моменту часу не зможе прочитати повідомлення, яке йому відправили. Практичне впровадження цієї концепції має сенс для таких напрямів:

- ставки на аукціонах;
- сплата рахунків у певний час;
- депонування ключів;
- ставки в лотереях та інше.

Рівест та співавтори сформували два підходи реалізації криптографії з часовим розкриттям, які, розвиваючись, залишились незмінними і до сьогодні [4, с. 1]:

- використовувати «математичні замки з часовим механізмом» – обчислювальні задачі, які не можуть бути розв'язані без обчислювання на комп'ютері протягом певного проміжку часу;
- використовувати третю сторону для зберігання інформації, за допомогою якої надалі буде розшифроване повідомлення через певний проміжок часу.

На жаль, перший з цих підходів на той час мав досить суттєві недоліки для використання на тогочасних системах, одним з яких було те, що обчислення часового замка практично унеможливлювало роботу інших процесів.

На сьогодні багатоядерні системи дають нам можливість вирішити цю проблему та використовувати часові замки без істотного впливу на роботоздатність комп'ютера, що, своєю чергою, є актуальною задачею. Також багатоядерні системи дають нам можливість розширити схему часових замків та використовувати цей підхід більш ефективно, а використання повноцінних операційних систем у мобільних пристроях дає нам можливість розширити цю концепцію і на них.

## Часові замки. Основи методу

Ще від самого початку для створення часових замків було використано підхід, який базувався на обчислювальній складності. Потрібно було протягом точно визначеного часу виконувати обчислення, щоб отримати ключ для розшифрування повідомлення.

Рівест та співавтори [4, с. 2] запропонували метод створення часових замків, що ґрунтується на повторному піднесенні у квадрат. Цей підхід можна також розглядати як застосування Blum-Blum-Shub «$x^2 \bmod n$» псевдовипадкового генератора чисел.

Розглянемо принцип роботи такого підходу. Якщо потрібно передати повідомлення $M$, яке зашифроване на час $T$, відправник має зробити такі кроки:

1. Генерується складена величина

$$n = pq,$$

в якій $p$ і $q$ – два випадково обраних великих числа.

2. Обраховується

$$\varphi(n) = (p-1)(q-1).$$

3. Обраховується $t = TS$, де $S$ – кількість піднесень у квадрат за модулем $n$ за секунду, яку може забезпечити техніка одержувача.

4. Генерується випадковий ключ $K$ для загальноприйнятої криптосистеми (наприклад, $RC5$). Довжина ключа має бути такою, щоб його підбір був недоцільним у порівнянні з часом обчислення замка.

5. Ключем $K$ зашифровується повідомлення $M$, таким чином отримуємо зашифрований текст $C_M = RC5(K, M)$.

6. Випадковим чином обирається $a$ за модулем $n$ ($1 < a < n$) та зашифровується $K$ методом обчислення

$$C_K = K + a^{2^t} \pmod n.$$

Для більшої ефективності спочатку обчислюється

$$e = 2^t \pmod{\varphi(n)},$$

а потім

$$b = a^e \pmod n. \qquad (1)$$

7. У результаті отримуємо замок ($n$, $a$, $t$, $C_K$, $C_M$) та видаляємо всі змінні, що залишилися (наприклад, $p$, $q$) від попередніх розрахунків.

Виходячи з наведених вище кроків, найшвидший шлях для обрахування замка полягає в тому,

що нам потрібно певним чином визначити $b = a^{2^t} \pmod n$. Якщо відома функція $\varphi(n)$, то $b$ можна обчислити за рівнянням (1).

Для $t << n$ (тобто $n > 2^{1024}$ та $t < 2^{100}$) можна припустити, що розклад на множники за $n$ (і таким чином обчислення $\varphi(n)$ для проведення згаданих вище кроків) буде значно важчим процесом, ніж пряме піднесення до степеня $t$ разів. Відповідно до цієї умови в нас немає іншого виходу, як розкладати на множники за $n$, за якими можна обчислити $a^{2^t} \pmod n$ за час менший, ніж час піднесення у степінь $t$ разів. Більше того, через те що кожне піднесення можна провести на основі попереднього результату піднесення, спосіб, за яким можна підвищити швидкість обрахування за допомогою паралелізації процесу, не може мати місце [1, с. 2].

Таким чином, для розшифрування ключа (повідомлення) не залишається нічого іншого, як отримати $b$ шляхом послідовного піднесення у квадрат $t$ разів.

У роботі [3, с. 3] припустили, що вираз

$$L(a,t,n) = \{(a,t,a^{2^t} \bmod n) \mid t < n, \gcd(a,n) = 1\}$$

є гарним кандидатом для криптографії з часовим розкриттям.

Рівест та співавтори запропонували та запустили подібний часовий замок у 1999 р. на честь 35-річчя «MIT Laboratory for Computer Science», він повинен розкритися через 35 років.

Описаний вище спосіб реалізації часових замків з плином часу майже не змінився, і подальший розвиток цього методу полягав лише у спробах створення протоколу з нульовим розголошенням [3, с. 2].

Але еволюція технічних можливостей комп'ютерних систем (зокрема й настільних) дає змогу використовувати часові замки навіть у побуті, а використання багатоядерних процесорів може дати нам можливість розкривати одночасно більше одного часового замка і, своєю чергою, не буде повністю завантажувати систему. Отже, дослідження характеристик продуктивності процесу розкриття часового замка на сучасних персональних багатоядерних комп'ютерах є актуальною задачею, що на сьогодні взагалі не досліджена.

## Дослідження характеристик продуктивності процесу розкриття часового замка

Використання сучасних багатоядерних комп'ютерних систем для розкриття часового замка дає нам можливість модифікувати схему обрахувань і працювати одночасно не з одним замком, як про це говорилось раніше, а з декількома. Використовуючи ядра процесора, ми можемо паралельно обраховувати їх на різних ядрах. Це дасть нам можливість розширити схему використання часових замків та зробити її більш продуктивною та універсальною. Це розширення, своєю чергою, ніяк не вплине на швидкість обрахунків і не порушить попередньо описану схему розкриття часових замків.

Щоб довести коректність роботи запропонованої концепції, було проведено експериментальні дослідження на сучасних багатоядерних системах. Окрім настільних та портативних систем, дослідження було розширено на мобільні пристрої, зокрема на смартфони на базі Android-ОС, які, своєю чергою, теж використовують багатоядерні процесори у своїх модифікаціях.

Оскільки в розкритті часового замка основоположними є процесорні потужності системи, найбільша увага в дослідженнях приділялась диференціації результатів за типом і тактовою частотою процесора. Дослідження проводились на трьох різних системах із такими конфігураціями:

1) настільний комп'ютер – Intel Core I5 750 – 4 ядра з тактовою частотою процесора 3,71 GHz;

2) портативний комп'ютер – Intel Core 2 Duo P8600 – 2 ядра з тактовою частотою процесора 2,4 GHz;

3) мобільний телефон – Qualcomm Snapdragon MSM8064T – 4 ядра з тактовою частотою процесора 1,89 GHz.

Остання конфігурація належить смартфону Samsung Galaxy S4 Active. Незважаючи на те, що такі портативні системи вочевидь програватимуть у роботоздатності настільним рішенням, включення їх у дослідження є актуальним через те, що такі системи нині набувають усе більшого поширення і мають попит серед користувачів. Власне, сама ідея криптографії з часовим розкриттям є дуже актуальною для подібних систем і практично не досліджена.

Для проведення дослідження було задіяно неповну схему часових замків. Як було зазначено вище, основою отримання замка є піднесення у степінь, тому фактично система обчислювала не повноцінний замок, а лише проводила піднесення у степінь заздалегідь обраного числа протягом певного часу. Тобто було реалізовано таку операцію:

$$b = a^{2^t} \pmod n,$$

де $a$ – велике випадкове число; $n$ – велике просте випадкове число; $t$ – час виконання обчислень.

Програмний код було написано на Java з використанням середовища розробки Eclipse. При цьому було використано бібліотеку довгої арифметики BigInteger, яка дає змогу працювати

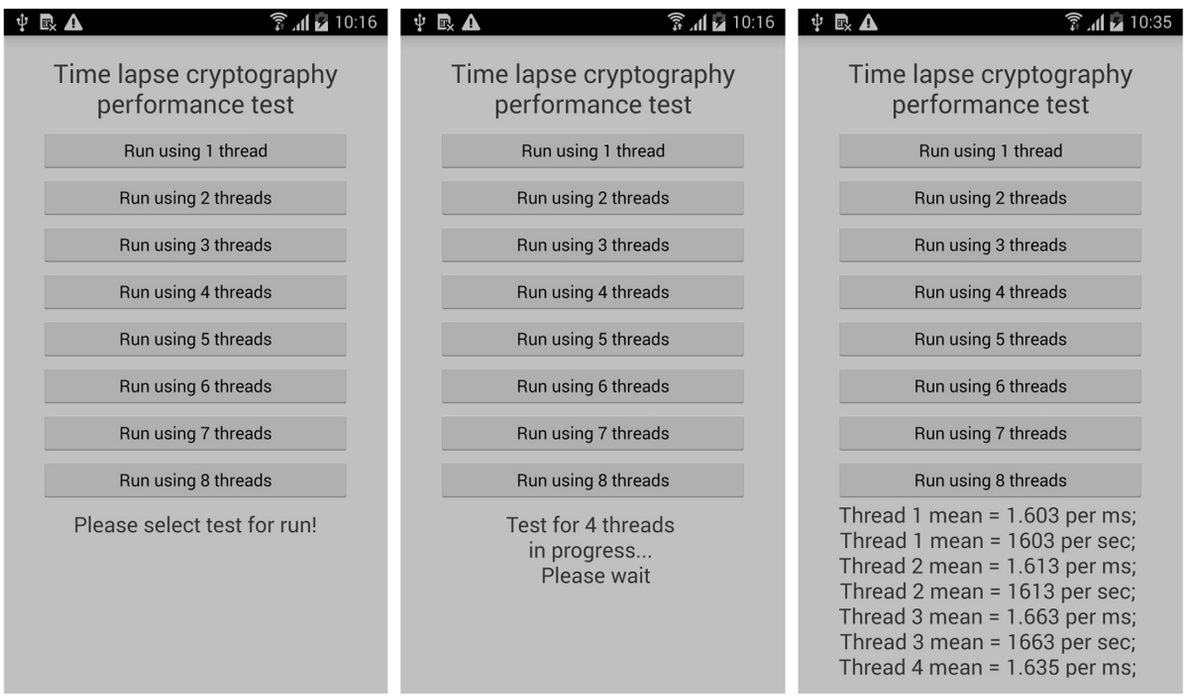


**Рис. 1.** Зовнішній вигляд графічної оболонки для запуску обчислень

з довгими числами. У нашому випадку обиралися числа актуальної на сьогодні довжини, а саме 1024 біти.

У випадку з настільними системами програма працювала протягом певного часу, після чого проводився запис отриманих даних у текстовий файл. На відміну від настільних систем, програма для мобільного пристрою мала власну оболонку, яка, окрім запису отриманих даних у файл, мала змогу відображати на екрані середню кількість операцій на цьому пристрої, виконану за мілісекунду і секунду, тим самим даючи нам можливість практично одразу оцінити швидкість виконання обчислень на цьому пристрої.

На рис. 1 наведено приклад графічної оболонки для запуску обчислень на смартфоні.

Під час експерименту на кожній системі проводилася емуляція розкриття від одного до восьми (для 4-ядерних систем) та до чотирьох (для 2-ядерної системи) замків. Розкриття кожного замка проводилось в окремому потоці, тому в подальшому для спрощення емуляцію одного замка будемо називати потоком. Мова програмування Java автоматично розподіляє виконання задач потоків по різних ядрах системи. У випадку, коли кількість потоків збігається з кількістю ядер, кожен потік буде виконуватись на різних ядрах. У випадку, коли кількість потоків більша за кількість ядер, виконання операцій буде проводитись паралельно, але при цьому потоки будуть час від часу призупинятись, щоб дати змогу

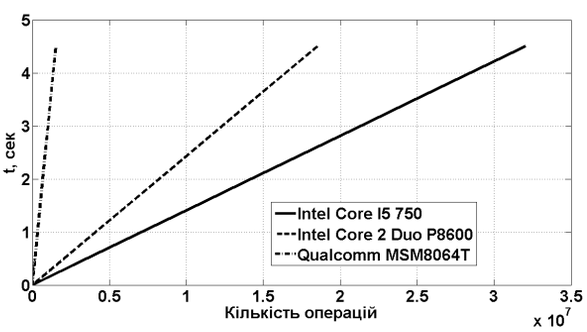


**Рис. 2.** Процесорний час, необхідний для модулярного піднесення числа у степінь

іншому потоку теж виконувати свої задачі. Тим самим зі збільшенням кількості потоків збільшуватиметься час простою кожного потоку і власне для отримання певних результатів витрачатиметься більше часу. Виходячи з цих умов, кількість потоків, обрану для дослідження, було збільшено у 2 рази від кількості ядер саме для того, щоб показати, як поводитиме себе ця задача, використовуючи одночасне розкриття більшої кількості замків, ніж у нас є ядер у системі.

На рис. 2 показано приклад зміни величини кількості операцій піднесення у степінь відповідно до часу в режимі дослідження емуляції обрахування одного замка на всіх вказаних вище системах. Дослідження проводилось протягом 5 хвилин. Як видно з графіка на рис. 2, кількість проведених операцій досить прогнозовано зростає лінійно за часом. Разом з тим досить чітко видно, що

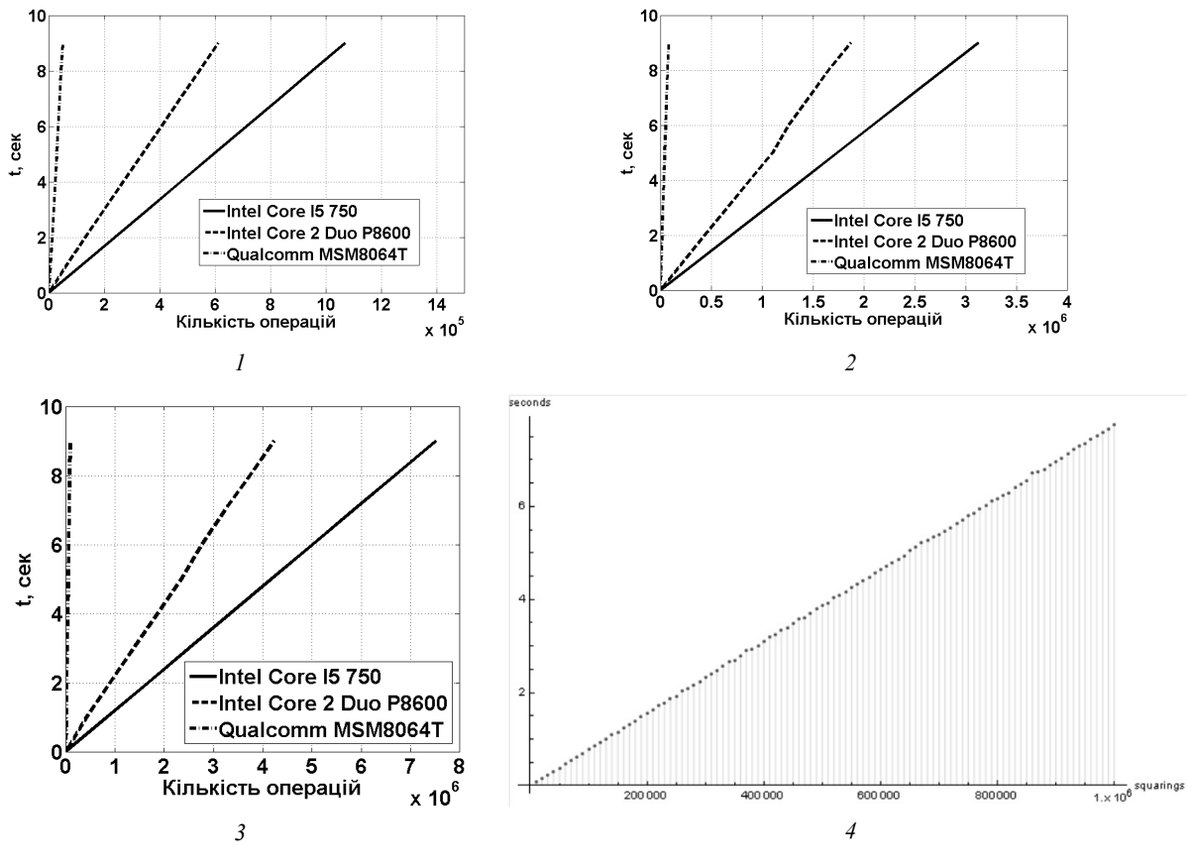


**Рис. 3.** Процесорний час, необхідний для модулярного піднесення числа у степінь: *1* – дані власного експерименту (довжина числа – 1024 біти); *2* – дані власного експерименту (довжина числа – 512 бітів); *3* – дані власного експерименту (довжина числа – 256 бітів); *4* – дані Ебрінгера Intel Core2Duo 2,4 GHz [2, с. 3]

потужність мобільного пристрою в порівнянні з повноцінними комп'ютерами є значно меншою.

На рис. 3 представлено результати досліджень, які розширюють ті, що наведено на рис. 2.

Дані було отримано для ключів завдовжки 1024, 512 та 256 бітів за проміжок часу 10 секунд для того, щоб їх можна було порівняти з даними, отриманими Ебрінгером [2, с. 3]. На жаль, у своїй роботі [2, с. 3] автор не вказує, якої довжини числа було обрано для його розрахунків, але, взявши до уваги час написання його статті, його процесор, а також порівнявши власні дані (рис. 3.*2*) з даними Ебрінгера (рис. 3.*3*), можна припустити, що це були числа завдовжки 512 бітів.

З отриманих результатів видно, що апаратна частина більш сучасних систем дає більш високі результати обчислень за секунду, тим самим можна зробити прогнозований висновок, що, незважаючи на те, що в нових системах використовується багатоядерність, окрім неї, швидкість обрахувань кожного з ядер теж зростає в порівнянні з минулим поколінням процесорів.

Порівняння процесорного часу та довжини ключа для всіх комп'ютерних систем, що

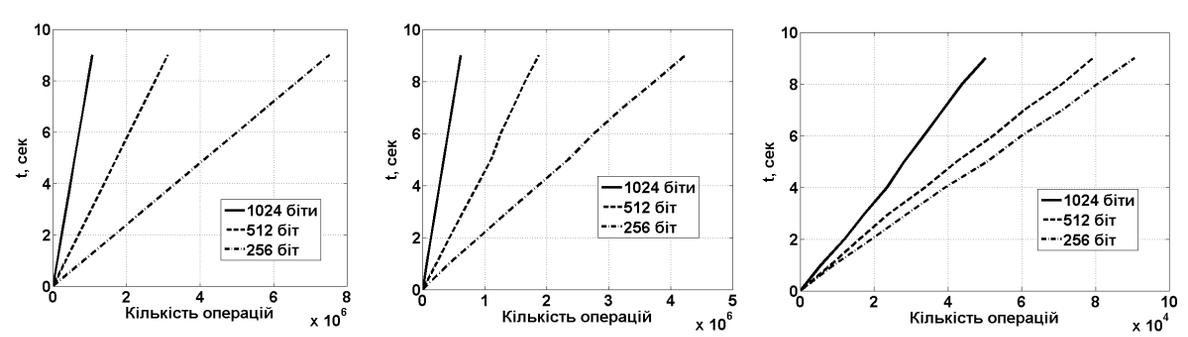


**Рис. 4.** Процесорний час, необхідний для модулярного піднесення чисел різних довжин у степінь: *1* – Intel Core I5 750; *2* – Intel Core 2 Duo P8600; *3* – Qualcomm Snapdragon MSM8064T

досліджувалися, показує, що зростання кількості операцій при зменшенні довжини ключа для смартфону проходить з меншою швидкістю в порівнянні з портативною та настільною комп'ютерними системами (рис. 4). Особливо це помітно для довжини ключа 512 та 256 бітів (рис. 4.*3*).

Під час емуляції більше ніж одного замка кількість операцій за мілісекунду в комп'ютерних системах була приблизно однакова, якщо використовувалась кількість потоків менша або рівна кількості ядер, для кожного потоку, що можна побачити з даних, наведених у таблиці.

Різкий спад операцій по потоках можна побачити на процесорі Intel Core 2 Duo P8600. Згідно з таблицею в нас деякі потоки простоюють більше, ніж інші в цьому процесорі, тим самим отримуємо різну кількість операцій за мілісекунду. На інших процесорах зі зростанням потоків зменшується кількість операцій, але приблизно по кожному потоку кількість операцій збігається. Найвірогідніше, різниця в отриманих даних теж зумовлена особливостями архітектури процесорів.

Більш наочно збільшення потоків та зменшення кількості операцій можна побачити на

*Таблиця*. **Величини значень операцій за мілісекунду (оп/мс)**

| Процесор | 1 потік | 2 потік | 3 потік | 4 потік | 5 потік | 6 потік | 7 потік | 8 потік |
|---|---|---|---|---|---|---|---|---|
| Core I5 750 | 118 | | | | | | | |
| Core I5 750 | 120 | 117 | | | | | | |
| Core I5 750 | 119 | 117 | 119 | | | | | |
| Core I5 750 | 99 | 100 | 99 | 98 | | | | |
| Core I5 750 | 95 | 95 | 94 | 95 | 93 | | | |
| Core I5 750 | 77 | 78 | 77 | 77 | 76 | 79 | | |
| Core I5 750 | 66 | 65 | 66 | 68 | 67 | 68 | 69 | |
| Core I5 750 | 49 | 50 | 50 | 49 | 50 | 50 | 50 | 49 |
| Intel P8600 | 68 | | | | | | | |
| Intel P8600 | 65 | 64 | | | | | | |
| Intel P8600 | 40 | 50 | 40 | | | | | |
| Intel P8600 | 29 | 36 | 29 | 37 | | | | |
| Qualcomm | 5,62 | | | | | | | |
| Qualcomm | 3,64 | 3,73 | | | | | | |
| Qualcomm | 2,39 | 2,37 | 2,37 | | | | | |
| Qualcomm | 1,62 | 1,59 | 1,62 | 1,62 | | | | |
| Qualcomm | 1,42 | 1,43 | 1,49 | 1,49 | 1,49 | | | |
| Qualcomm | 1,07 | 1,07 | 1,07 | 1,07 | 1,07 | 1,11 | | |
| Qualcomm | 1,00 | 0,99 | 0,98 | 0,98 | 1,01 | 0,98 | 0,98 | |
| Qualcomm | 0,82 | 0,84 | 0,86 | 0,83 | 0,82 | 0,85 | 0,82 | 0,83 |

Невеликий спад спостерігався лише тоді, коли ми використовували кількість потоків, рівну кількості ядер, це добре видно для процесора Core I5 750. А от для процесора мобільного пристрою Qualcomm Snapdragon MSM8064T характерні досить суттєві коливання значень, навіть при кількості потоків меншій або рівній кількості ядер. Це, вірогідно, зумовлено відмінностями в архітектурі мобільного та повноцінного процесора.

Якщо ж взяти кількість потоків більшу, ніж кількість ядер (одночасно обраховувати більше замків, ніж ядер у процесорі), то за даними, наведеними в таблиці, чітко простежується різкий спад швидкодії зі зростанням кількості потоків. Як уже було описано раніше, це зумовлено тим, що зі збільшенням кількості потоків зростає час простою кожного потоку, тим самим зменшується кількість операцій по кожному з них (по кожному ядру кількість операцій залишається такою ж). графіку, наведеному на рис. 5. Для зручності представлення результатів було проведено осереднення по потоках.

Для систем з Intel Core I5 750 та Intel P8600 продуктивність різко починає знижуватися після того, як кількість потоків стає рівною кількості ядер. Тому для цих процесорів є проміжок сталих значень, який збігається з кількістю ядер.

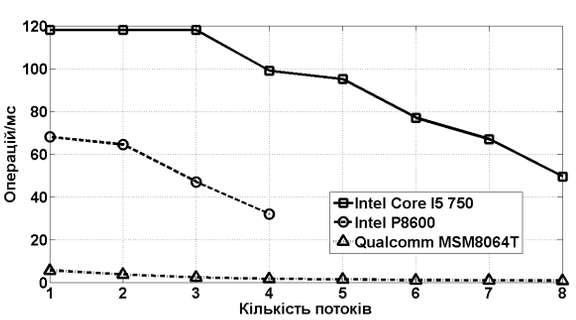


**Рис. 5.** Залежність кількості операцій за мілісекунду від кількості потоків, запущених одночасно

Зовсім інша схема зниження кількості операцій спостерігається для смартфону: незважаючи на наявність чотирьох ядер, продуктивність починає спадати одразу. Такі результати можуть бути зумовлені особливостями взаємодії операційної системи з апаратною частиною або архітектурою мобільного процесора.

**Висновки**

Результати проведених експериментів показують, що сучасні настільні та портативні багатоядерні системи можуть бути використані для розкриття часових замків навіть у побутових умовах. Причому кількість часових замків, процеси розкриття яких проходять одночасно, має бути на одиницю меншою за кількість ядер у процесорі, оскільки паралельно можуть виконуватись інші процеси в комп'ютері і буде некоректно займати всі ядра, тому що через ці процеси може сповільнитись обчислення одного із замків.

На жаль, смартфони не можуть використовуватися для обчислення декількох замків через недостатню потужність апаратної частини.

Незважаючи на отримані позитивні результати досліджень, на шляху повноцінного впровадження цього способу безпечної передачі даних стоїть технічна перепона: для збільшення точності часу розпакування потрібно повністю виділяти під один процес розкриття одне ядро процесора, що на сучасному етапі досліджень неможливо зробити, оскільки, окрім обрахування часового замка, у нас можуть виконуватись інші процеси в операційній системі, а це своєю чергою даватиме певні неточності щодо часу розкриття, що може бути критично в окремих випадках.

Разом з тим з отриманих результатів видно, що побутові багатоядерні системи можуть ефективно використовуватись у криптографії з часовим розкриттям для обрахування часових замків. Також дослідження показали, що ці системи здатні більш ефективно проводити обрахування часових замків, що своєю чергою більш позитивно впливатиме на процес отримання ключа та дасть змогу отримувати ключ з меншою похибкою щодо часу.

***Список літератури***

1. Boneh D. Timed commitments / Dan Boneh, Mani Naor // Lecture Notes in Computer Science. – 2000. – Vol. 1880. – P. 236–254.
2. Ebringer T. Anti-emulation through time-lock puzzles / T. Ebringer // Second International CARO Workshop. – Hoofddorp, Netherlands, 2008. – P. 1–10.
3. Mao W. Timed-release cryptography / W. Mao // Selected Areas in Cryptography VIII SAC'01. – 2001. – P. 1–18.
4. Rivest R. L. Time-lock puzzles and timed-released crypto / R. L. Rivest, A. Shamir, D. A. Wagner. – Cambridge : MA, USA, 1996. – 9 p.

*S. Gonchar*

## TIMED-RELEASE CRYPTOGRAPHY: PROSPECTS OF DEVELOPMENT FOR MULTI-CORE PROCESSORS

*The article reviewed the use of such direction of time-released cryptography as time-lock puzzles for modern multi-core systems and results of experimental studies that have shown that the amount of time-locks opening processes which take place at the same time, must be one less than the number of cores in the processor. The novelty of this study also lies in the fact that along with portable systems research was conducted for mobile systems (smartphone).*

**Keywords:** time-released cryptography, time-lock puzzles, multi-core processors.

*Матеріал надійшов 09.12.2014*