КРИПТОГРАФІЯ
КОМП'ЮТЕРНИЙ ПРАКТИКУМ №5

Вивчення криптосистеми RSA та алгоритму електронного підпису; ознайомлення з методами генерації параметрів для асиметричних криптосистем

Мета та основні завдання роботи

Ознайомлення з тестами перевірки чисел на простоту і методами генерації ключів для асиметричної криптосистеми типу RSA; практичне ознайомлення з системою захисту інформації на основі криптосхеми RSA, організація з використанням цієї системи засекреченого зв'язку й електронного підпису, вивчення протоколу розсилання ключів.

Основні теоретичні відомості

1. Властивість конгруенцій

Якщо у виразі цілої раціональної функції з цілими коефіцієнтами $F = \sum A_{\alpha_1,\dots,\alpha_k} x_1^{\alpha_1} \dots x_k^{\alpha_k}$ замінити $A_{\alpha_1,\dots,\alpha_k}, x_1, \dots, x_k$ числами $B_{\alpha_1,\dots,\alpha_k}, y_1, \dots, y_k$, конгруентними з колишніми за модулем m, то новий вираз F буде конгруентним з колишнім за модулем m.

2. Схема Горнера швидкого піднесення до степеня

Потрібно обчислити вираз $x^\alpha \pmod m, 0 \le \alpha,\ x < m,\ \alpha, x$ – цілі невід'ємні числа. Представимо α у двійковій формі: $\alpha = \sum_{i=0}^{k-1} \alpha_i 2^i,\ \alpha_i \in \{0,1\}$. Тоді піднесення до степеня α можна звести до не більш ніж $k-1$ піднесення в квадрат за модулем і не більш ніж $k-1$ множень за модулем за допомогою однієї з двох формул:

$$x^\alpha \pmod m = (\dots(\dots(x^{\alpha_{k-1}})^2 x^{\alpha_{k-2}})^2 \dots x^{\alpha_1})^2 x^{\alpha_0} \pmod m, \text{ або}$$
$$x^\alpha \pmod m = x^{\alpha_0} (x^2)^{\alpha_1} (x^{2^2})^{\alpha_2} \dots (x^{2^{k-1}})^{\alpha_{k-1}} \pmod m$$

При цьому всі проміжні обчислення (піднесення до квадрата й множення) виконуються за $\bmod m$ у відповідності з властивістю конгруенцій. Наприклад, першу формулу можна задати таким алгоритмом (у псевдокоді):

$y := 1$
`for` $i := k-1$ `to 0 do`
    $y := y^2 \pmod m$
    $y := y \cdot x^{\alpha_i} \pmod m$

3. Псевдопрості числа

Для побудови, формального опису та оцінки коректності та ефективності тестів на простоту використовуються так звані *псевдопрості числа* – складені числа, що зберігають ті чи інші властивості простих.

Непарне число p називається *псевдопростим за основою* $a \in N$, якщо $\gcd(a, p) = 1$[1] і $a^{p-1} \equiv 1 \pmod{p}$.

Непарне число p називається *псевдопростим числом Ойлера*[2] *за основою* $a \in N$, якщо $\gcd(a, p) = 1$ та $\left(\frac{a}{n}\right) \equiv a^{\frac{n-1}{2}} \pmod{n}$, де $\left(\frac{a}{n}\right)$ – символ Якобі.

Нехай p – непарне число, $p - 1 = d \cdot 2^s$, d – непарне. Число p називається *сильним псевдопростим числом за основою* $a \in N$, якщо $a^d \equiv 1 \pmod{p}$ або $a^{d \cdot 2^r} \equiv -1 \pmod{p}$ при якомусь $r, 0 \le r < s$.

4. Тести перевірки натуральних чисел на простоту

В наш час існує ряд ефективних методів тестування великих чисел на простоту – імовірнісних, детермінованих, а також гіпотетичних (що спираються на деяку математичну гіпотезу). На практиці в більшості випадків використовуються імовірнісні тести, до яких відносяться тест Ферма, тест Соловея-Штрассена та тест Міллера-Рабіна. Знайдений у 2000-му році детермінований тест перевірки на простоту (*тест трьох індусів* або *тест AKS*[3]) виявився більш ефективним за імовірнісні лише для чисел, довжина яких перевищує десятки тисяч біт.

Також слід зауважити, що зазвичай перед використанням імовірнісних тестів використовується так званий *тест пробних ділень* – перевірка, чи поділяється досліджуване число на набір малих простих множників (2, 3, 5, 7 тощо). Тест пробних ділень дозволяє швидко відкинути до 90% складених чисел під час перевірки.

Для побудови тесту пробних ділень на довільні малі множники використовують так звану *ознаку подільності Паскаля*. Нехай натуральне число N записане у системі числення із основою B:

$$N = \overline{a_t a_{t-1} \ldots a_1 a_0} = a_t B^t + a_{t-1} B^{t-1} + \ldots + a_1 B + a_0,$$

де $0 \le a_i \le B - 1$. Тоді для довільного (малого) m має місце рівність:

$$N \equiv \sum_{i=0}^{t} a_i r_i \pmod{m},$$

де $r_i = B^i \bmod m$. Сума в правій частині не перевищує величини tmB, що є значно меншим за оригінальне N. Послідовність r_i не залежить від N та може бути обчислена заздалегідь

---

[1] $\gcd(a, b)$ (від англ. greatest common divisor) – найбільший спільний дільник чисел a та b.

[2] В деяких джерелах такі числа називають *псевдопростими Ойлера-Якобі*.

[3] *Тест Агравала-Каяла-Саксени* – це, власне, ті самі три індуси

за рекурентною формулою $r_0 = 1$, $r_{i+1} = r_i \cdot B \bmod m$. Більш того, послідовність r_i є періодичною, що спрощує обчислення.

З ознаки подільності Паскаля випливають відомі нам зі школи ознаки подільності на 2, 3, 5, 7, 11 тощо в десятковій системі числення.

4.1. Імовірнісний тест Ферма

Оберемо деяке число $k \in N$.

Крок 0. Встановлюємо лічильник у 0.

Крок 1. Вибираємо випадкове число $x \in N$ з інтервалу $1 < x < p$ (незалежне від раніше обраних *x*). За допомогою алгоритму Евкліда знаходимо $\gcd(x, p)$. Якщо $\gcd(x, p) = 1$, то переходимо до кроку 2. Якщо $\gcd(x, p) > 1$, то *p* – складене число, алгоритм завершує свою роботу.

Крок 2. Перевіряємо, чи є *p* псевдопростим за основою *x*. Якщо *p* виявилось не псевдопростим за основою *x*, то *p* – складене число, алгоритм завершує свою роботу; інакше лічильник збільшується на 1.

Крок 3. Якщо лічильник менший за *k*, то переходимо до кроку 1, інакше алгоритм завершує роботу.

У результаті число *p* буде перевірено за тестом Ферма при *k* різних основах $a_1, ..., a_k$. Якщо *p* виявиться не псевдопростим хоча б за однією основою $a_i, i = 1, ..., k$, то *p* складене. Якщо *p* псевдопросте за всіма цими основами, то вважаємо *p* простим числом.

Існує клас складених чисел, що проходять тест Ферма із імовірністю 1 – так звані *числа Кармайкла*. Не дивлячись на те, що імовірність натрапити на число Кармайкла при випадковій генерації майже дорівнює нулю, така теоретична загроза вважається достатньою для того, щоб не використовувати тест Ферма у практичних застосуваннях.

4.2. Імовірнісний тест Соловея-Штрассена

Оберемо деяке число $k \in N$.

Крок 0. Встановлюємо лічильник у 0.

Крок 1. Вибираємо випадкове число $x \in N$ з інтервалу $1 < x < p$ (незалежне від раніше обраних *x*). За допомогою алгоритму Евкліда знаходимо $\gcd(x, p)$. Якщо $\gcd(x, p) = 1$, то переходимо до кроку 2. Якщо $\gcd(x, p) > 1$, то *p* – складене число, алгоритм завершує свою роботу.

Крок 2. Перевіряємо, чи є *p* псевдопростим Ойлера за основою *x*. Якщо *p* виявилось не псевдопростим за основою *x*, то *p* – складене число, алгоритм завершує свою роботу; інакше лічильник збільшується на 1.

Крок 3. Якщо лічильник менший за *k*, то переходимо до кроку 1, інакше алгоритм завершує роботу.

У результаті число *p* буде перевірено за тестом Соловея-Штрассена при *k* різних основах $a_1, ..., a_k$. Якщо *p* виявиться не псевдопростим хоча б за однією основою $a_i, i = 1, ..., k$, то *p* складене. Якщо *p* псевдопросте за всіма цими основами, то вважаємо *p* простим числом.

Тест Соловея-Штрассена є одностороннім, тобто в ньому відсутні помилки першого роду: якщо *p* – просте число, то тест завжди поверне, що воно просте. Однак якщо *p* – число складене, то для кожної можливої основи тест із імовірністю ½ поверне

помилкову відповідь (повідомлення, що p – просте). Використання k різних випадкових основ зменшує помилку тесту до 2^{-k}. Також слід зауважити, що цей тест є доволі простим та невибагливим з обчислювальної точки зору: кожен запуск вимагає лише одного піднесення у степінь за модулем та обчислення символу Якобі, складність якого невелика.

4.3. Імовірнісний тест Міллера-Рабіна

Оберемо деяке число $k \in N$.

Крок 0. Знаходимо розклад $p-1 = d \cdot 2^s$ та встановлюємо лічильник у 0.

Крок 1. Вибираємо випадкове число $x \in N$ з інтервалу $1 < x < p$ (незалежне від раніше обраних x). За допомогою алгоритму Евкліда знаходимо $\gcd(x, p)$. Якщо $\gcd(x, p) = 1$, то переходимо до кроку 2. Якщо $\gcd(x, p) > 1$, то p – складене число, алгоритм завершує свою роботу.

Крок 2. Перевіряємо, чи є p сильно псевдопростим за основою x:

2.1 Якщо $x^d \bmod p = \pm 1$, то p є сильно псевдопростим за основою x, інакше

2.2 Для всіх послідовних r від 1 до $s-1$ виконати такі дії:

Обчислити $x_r = x^{d \cdot 2^r} \bmod p$ (тобто $x_r = x_{r-1}^2 \bmod p$).

Якщо $x_r = -1$, то p є сильно псевдопростим за основою x,

інакше якщо $x_r = 1$, то p не є сильно псевдопростим за основою x,

інакше перейти до наступного значення r.

2.3 Якщо за кроки 2.1 та 2.2 не було встановлено, що p є сильно псевдопростим за основою x, то p не є сильно псевдопростим за основою x.

Якщо p виявилось не псевдопростим за основою x, то p – складене число, алгоритм завершує свою роботу; інакше лічильник збільшується на 1.

Крок 3. Якщо лічильник менший за k, то переходимо до кроку 1, інакше алгоритм завершує роботу.

У результаті число p буде перевірено по тесту Міллера-Рабіна при k різних основах $a_1, ..., a_k$. Якщо p виявиться не сильно псевдопростим хоча б за однією основою $a_i, i = 1, ..., k$, то p складене. Якщо p сильно псевдопросте за всіма цими основами, то вважаємо p простим числом.

Тест Міллера-Рабіна є одностороннім, тобто в ньому відсутні помилки першого роду: якщо p – просте число, то тест завжди поверне, що воно просте. Однак якщо p – число складене, то для кожної можливої основи тест із імовірністю ¼ поверне помилкову відповідь (повідомлення, що p – просте). Використання k різних випадкових основ зменшує помилку тесту до 4^{-k}.

Оскільки у тесті Соловея-Штрассена відповідна помилка складає 2^{-k}, він в ході жорсткої еволюційної боротьби майже всюди витіснений тестом Міллера-Рабіна.

Тест Міллера-Рабіна дозволяє будувати детерміновані тести для перевірки простоти натуральних чисел, якщо вони не перевищують деякої межі. Наприклад, наступний тест перевіряє простоту числа $p < 3 \cdot 10^9$**:**

Крок 1. Перевіряємо, чи є p сильним псевдопростим числом за основою 2. Якщо ні, то p складене й алгоритм зупиняється.

Крок 2. Те ж за основою 3.

Крок 3. Те ж за основою 5.

Крок 4. Те ж за основою 7.

Крок 5. Якщо не було виявлено, що p складене, то p є простим числом.

5. Вибір випадкового простого числа

Нехай необхідно вибрати велике випадкове число p таке, що $n_0 \le p \le n_1$, де $n_0, n_1 \in N$. Використовуючи датчик випадкових чисел, вибираємо випадкове число x з інтервалу $[n_0, n_1]$. За допомогою тестів перевірки чисел на простоту знаходимо перше просте число p у послідовності непарних чисел $m_0 + 2i, \quad i = 0, 1, 2, ..., \left[\frac{n_1 - m_0}{2}\right]$, де m_0 – найменше непарне число, більше чи рівне x, $[z]$ – ціла частина числа z. Тоді p – шукане випадкове просте число. Якщо серед тестованих чисел виду $m_0 + 2i$ число, що задовольняє тестам на простоту, не знайдено, то повторюємо процедуру, генеруючи нове випадкове число x і, можливо, розширивши інтервал $[n_0, n_1]$.

Згідно постулату Бертрана, для кожного натурального n в інтервалі $[n, 2n-2]$ обов'язково існує просте число[4], отже, наведена схема завершить свою роботу і поверне просте число за скінченну кількість кроків.

Для кращого протистояння криптоаналітичним атакам на ключ схеми RSA числа $p-1$ і $q-1$ (див. нижче) повинні містити великі прості дільники (найкраще, якщо $p-1 = 2p', \quad q-1 = 2q'$, де p', q' також як і p, q – прості[5]). Для забезпечення цієї умови генерування простих чисел p і q проводять у два етапи: спочатку генеруються прості числа p' і q', а потім тестуються на простоту числа виду $p = 2ip' + 1, \quad q = 2jq' + 1$ для $i, j = 1, 2, 3, ...$ доти поки p і q не будуть визнані простими. «Гарні» випадкові прості числа, що використовуються для ключів у RSA, повинні мати також деякі інші додаткові властивості. Також існують алгоритми гарантованої генерації простих чисел заданого розміру (метод Маурера, метод Діємітко тощо).

6. Побудова криптосистеми RSA

Побудова системи RSA відбувається таким чином.

Абонент A генерує два великих випадкових простих числа p і q, обчислює добуток $n = pq$ і функцію Ойлера $\varphi(n) = (p-1)(q-1)$. Потім A обирає випадкове число $e, \quad 2 \le e \le \varphi(n) - 1$ таке, що $\gcd(e, \varphi(n)) = 1$, і знаходить для e обернений за $\bmod \varphi(n)$ елемент d: $ed \equiv 1 (\bmod \varphi(n))$.

Зауваження: в сучасних криптографічних системах майже завжди обирається $e = 2^{16} + 1$.

Пара чисел n і e – це відкритий ключ A, а d – секретний (чи особистий) ключ A. При цьому числа p і q абонент A також тримає в таємниці. Не знаючи p і q, не можна обчислити $\varphi(n)$ і знайти секретний ключ d по відкритому e і n.

Зашифрування відкритого повідомлення $M, \quad 0 \le M \le n-1$, здійснюється за формулою

$$C = M^e \bmod n \ ,$$

---

[4] Постулат Бертрана неодноразово уточнювався та підсилювався. Зокрема, О.В. Вербицький приводить твердження, що для довільного n існує просте число в інтервалі $[n, n + n^{\frac{107}{200}}]$; в 2001 році було доведено, що просте число існуватиме в інтервалі $[n, n + O(n^{\frac{21}{40}})]$. У 2010 році П'єр Дузар довів, що для $n > 396738$ існує просте число в інтервалі $[n, n + \frac{n}{25 \ln^2 n}]$.

[5] Прості числа p такі, що число $2p + 1$ також є простим, називаються *числами Софі Жермен*.

де $C, 0 \le C \le n-1,$ – криптограма, (e,n) – відкритий ключ, тобто кожний може послати *A* шифроване повідомлення.

Розшифрування криптограми $0 \le C \le n-1$ здійснюється за допомогою особистого ключа *d* за формулою:

$$M = C^d \bmod n \ .$$

Цифровий підпис у системі RSA. Щоб підписати відкрите повідомлення *M,* абонент *A* зашифровує його на своєму секретному ключі $S = M^d \bmod n$ і додає *S* до повідомлення *M*. Підписане повідомлення (M,S) може передаватися як у відкритому, так і в зашифрованому виді. Перевіряється підпис за допомогою відкритого ключа: якщо виконується рівність $M = S^e \bmod n$, то повідомлення *M* не спотворене і підпис вірний. На практиці повідомлення *M*, що може бути набагато довше за n, попередньо піддається стисненню за допомогою геш-функції $H(M) = m$, і до повідомлення *M* додається підпис $S = m^d \bmod n$. Функція *H* повинна мати властивості *протидії колізіям*: *m* повинне істотно залежати від кожного біта *M* і за значенням *m* практично неможливо підібрати відповідне йому значення *M*.

7. Протокол конфіденційного розсилання ключів по відкритих каналах зв'язку з підтвердженням справжності відправника

Нехай абонент *A* має відкритий ключ (e,n), секретний *d* і відкритий ключ (e_1, n_1) абонента *B*, якому він має намір передати секретне значення $0 < k < n$ для створення спільного ключа симетричної чи асиметричної криптосистеми. Обов'язковою умовою роботи протоколу є $n_1 \ge n$; якщо $n_1 < n$, абонент А повинен згенерувати собі іншу пару ключів.

Абонент *A* формує повідомлення (k_1, S_1) і відправляє його *B*, де

$$k_1 = k^{e_1} \bmod n_1, \quad S_1 = S^{e_1} \bmod n_1, \quad S = k^d \bmod n \, .$$

Абонент *B* за допомогою свого секретного ключа d_1 знаходить (конфіденційність):

$$k = k_1^{d_1} \bmod n_1, \quad S = S_1^{d_1} \bmod n_1 \, ,$$

і за допомогою відкритого ключа *e* абонента *A* перевіряє підпис *A* (автентифікація):

$$k = S^e \bmod n \, .$$

Порядок і рекомендації щодо виконання роботи

1. Написати функцію пошуку випадкового простого числа з заданого інтервалу або заданої довжини, використовуючи датчик випадкових чисел та тести перевірки на простоту. В якості датчика випадкових чисел використовуйте вбудований генератор псевдовипадкових чисел вашої мови програмування. В якості тесту перевірки на простоту рекомендовано використовувати тест Міллера-Рабіна із попередніми пробними діленнями. Тести необхідно реалізовувати власноруч, використання готових реалізацій тестів не дозволяється.

2. За допомогою цієї функції згенерувати дві пари простих чисел p, q і p_1, q_1 довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб $pq \le p_1 q_1$; p і q – прості числа для побудови ключів абонента *A*, p_1 і q_1 – абонента *B*.

3. Написати функцію генерації ключових пар для RSA. Після генерування функція повинна повертати та/або зберігати секретний ключ (d, p, q) та відкритий ключ (n, e). За допомогою цієї функції побудувати схеми RSA для абонентів *A* і *B* – тобто, створити та зберегти для подальшого використання відкриті ключі (e, n), (e_1, n_1) та секретні d і d_1.

4. Написати програму шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів *A* і *B*. Кожна з операцій (шифрування, розшифрування, створення цифрового підпису, перевірка цифрового підпису) повинна бути реалізована окремою процедурою, на вхід до якої повинні подаватись лише ті ключові дані, які необхідні для її виконання.

За допомогою датчика випадкових чисел вибрати відкрите повідомлення *M* і знайти криптограму для абонентів *A* и *B*, перевірити правильність розшифрування. Скласти для *A* і *B* повідомлення з цифровим підписом і перевірити його.

5. За допомогою раніше написаних на попередніх етапах програм організувати роботу протоколу конфіденційного розсилання ключів з підтвердженням справжності по відкритому каналу за допомогою алгоритму RSA. Протоколи роботи кожного учасника (відправника та приймаючого) повинні бути реалізовані у вигляді окремих процедур, на вхід до яких повинні подаватись лише ті ключові дані, які необхідні для виконання. Перевірити роботу програм для випадково обраного ключа $0 < k < n$.

Кожна з наведених операцій повинна бути реалізована у вигляді окремої процедури, інтерфейс якої повинен приймати лише ті дані, які необхідні для її роботи; наприклад, функція Encrypt(), яка шифрує повідомлення для абонента, повинна приймати на вхід повідомлення та відкритий ключ адресата (і тільки його), повертаючи в якості результату шифротекст. Відповідно, програмний код повинен містити сім високорівневих процедур: GenerateKeyPair(), Encrypt(), Decrypt(), Sign(), Verify(), SendKey(), ReceiveKey().

Кожну операцію рекомендується перевіряти шляхом взаємодії із тестовим середовищем, розташованим за адресою

http://asymcryptwebservice.appspot.com/?section=rsa.

Наприклад, для перевірки коректності операції шифрування необхідно а) зашифрувати власною реалізацією повідомлення для серверу та розшифрувати його на сервері, б) зашифрувати на сервері повідомлення для вашої реалізації та розшифрувати його локально.

Оформлення звіту

Звіт до комп'ютерного практикуму оформлюється згідно зі стандартними правилами оформлення наукових робіт, за такими винятками:

- дозволяється використовувати шрифт Times New Roman 12pt та одинарний інтервал між рядками;
- для оформлення фрагментів текстів програм дозволяється використовувати шрифт Courier New 10pt та друкувати тексти в дві колонки;
- дозволяється не починати нові розділи з окремої сторінки.

До звіту можна не включати анотацію, перелік термінів та позначень та перелік використаних джерел. Також не обов'язково оформлювати зміст.

Звіт має містити:
- мету лабораторної роботи;
- постановку задачі та варіант завдання;
- хід роботи, опис труднощів, що виникали, та шляхів їх розв'язання;
- значення вибраних чисел p, q, p_1, q_1 із зазначенням кандидатів, що не пройшли тест перевірки простоти, і параметрів криптосистеми RSA для абонентів A і B;
- чисельні значення прикладів ВТ, ШТ, цифрового підпису для A і B;
- опис кроків протоколу конфіденційного розсилання ключів з підтвердженням справжності, чисельні значення характеристик на кожному кроці;
- висновки.

Тексти всіх програм здаються викладачеві в електронному вигляді для перевірки на плагіат. До захисту комп'ютерного практикуму допускаються студенти, які оформили звіт та пройшли перевірку програмного коду.

Контрольні питання

1) Сформулюйте малу теорему Ферма та теорему Ойлера.
2) Яке число називається псевдопростим, псевдопростим Ойлера та сильним псевдопростим?
3) Опишіть покроково тест Ферма, тест Соловея-Штрассена та тест Міллера-Рабіна. На яких математичних задачах вони ґрунтуються? Які помилки першого та другого роду вони мають?
4) Дайте визначення односторонньої функції та односторонньої функції з потаємним ходом. Наведіть приклади кандидатів у такі функції.
5) Наведіть приклад застосування схеми Горнера для піднесення до степеня по модулю.
6) Сформулюйте постулат Бертрана. Доведіть, що для довільного натурального m існує просте число довжини m біт.
7) Опишіть криптосистему RSA та її побудову.
8) Знайдіть імовірність того, що в системі RSA випадково обране повідомлення M матиме спільний дільник із модулем n.
9) Знайдіть імовірність того, що випадково вибраний відкритий ключ e в RSA буде взаємно простим з $\varphi(n)$, якщо $p = 2r + 1$, $q = 2s + 1$, а r та s – прості числа.
10) Чи можуть в схемі RSA співпадати числа p та q?
11) Які функції та задачі виконує цифровий підпис?
12) Які Ви знаєте обмеження на вибір параметрів криптосистеми RSA?
13) Що таке геш-функція та колізії геш-функцій?
14) Навіщо в протоколі конфіденційного розсилання ключів потрібно шифрування підпису S?
15) Навіщо в протоколі конфіденційного розсилання ключів потрібна умова $n_1 \geq n$? Чи можливо сформулювати аналогічний протокол, який би не вимагав зазначеного обмеження?

Оцінювання комп'ютерного практикуму

За виконання лабораторної роботи студент може одержати до 11 рейтингових балів; зокрема, оцінюються такі позиції:
- реалізація програм – до трьох балів (в залежності від правильності та швидкодії);

– теоретичний захист роботи – до п’яти балів;
– своєчасне виконання роботи – 1 бал;
– несвоєчасне виконання роботи – (-1) бал за кожен тиждень пропуску.

Програмний код, створений під час виконання комп’ютерного практикуму, перевіряється на наявність неправомірних запозичень (плагіату) за допомогою сервісу *Stanford MOSS Antiplagiarism*. У разі виявлення в програмному коді неправомірних запозичень реалізація програм оцінюється у 0 балів, а за виконання практикуму студент одержує штраф (–10) балів.

Студенти допускаються до теоретичного захисту тільки за умови оформленого звіту з виконання практикуму.