УДК 621.3.06

**А. В. Дудатьєв, к. т. н., доц.; П. В. Козлюк; Д. С. Оксимчук**

# РОЗРОБКА АЛГОРИТМУ ПРИХОВУВАННЯ ЦИФРОВИХ ВОДЯНИХ ЗНАКІВ У АУДІОФАЙЛАХ ФОРМАТУ WAV

*У статті проілюстровано процес розробки та аналізу стеганографічного алгоритму, призначеного для приховування текстових повідомлень або бінарних файлів даних в аудіофайлах формату wav.*

***Ключові слова:*** *ЦВЗ, стеганографія, захист медіафайлів, стеганографічні алгоритми, шифрування даних, програмування C#.*

## Вступ

Цифровий водяний знак (ЦВЗ) – технологія, створена для захисту авторських прав мультимедійних файлів. ЦВЗ здебільшого застосовують для захисту від копіювання й несанкціонованого використання [1]. У зв'язку з бурхливим розвитком технологій мультимедіа гостро постало питання захисту авторських прав та інтелектуальної власності, представленої у цифровому вигляді. Прикладами можуть бути фотографії, аудіо- й відеозаписи і т. д. У зв'язку із крадіжками або модифікацією повідомлень переваг вставки та передачі повідомлень у цифровому вигляді може не бути. Тому розробляють різні заходи захисту інформації організаційного й технічного характеру. Один з найбільш ефективних технічних засобів захисту мультимедійної інформації полягає у вбудовуванні в об'єкт невидимих міток – ЦВЗ. Розробки в цій області проводять найбільші фірми в усьому світі. Оскільки методи ЦВЗ почали розроблятися зовсім недавно, у цій галузі є багато проблем, що вимагають розгляду. Свою назву цей метод отримав від усім відомого способу захисту цінних паперів, у тому числі й грошей, від підробки (термін «digital watermarking») [5]. На відміну від звичайних водяних знаків ЦВЗ можуть бути не тільки видимими, але й (як правило) невидимими. Невидимі ЦВЗ аналізуються спеціальним декодером, який приймає рішення про їхню коректність. ЦВЗ можуть містити деякий автентичний код, інформацію про власника, або яку-небудь керуючу інформацію. Найбільш придатними об'єктами захисту за допомогою ЦВЗ є нерухливі зображення, файли аудіо- й відеоданих [4].

Метою цієї статті є розробка алгоритму та програмного забезпечення для приховування цифрових водяних знаків в аудіофайлах. В якості цільового формату аудіофайлів розглядається формат wav.

Розглянемо структуру файлів цього формату.

Найперші байти у файлі WAV - це ідентифікатори формату.

```
typedef struct {
     char id[4];//- ідентифікатор файлу = "RIFF" = 0x46464952
     long len;  // - довжина файлу без цього заголовка
} IDRiff;
```

Наведений код, описаний мовою C#, демонструє програмну реалізацію структури заголовку файлу. Так само як і у випадку інших контейнерів, розпізнавання відбувається саме за цими байтами, тому розширення імені файлу може бути будь-яким.

У найпростішому випадку після ідентифікаційного заголовка у WAV-файлі вказується розмір і формат даних (на що відведено 24 байта), в тому числі наводиться величина бітрейту (скільки відліків в секунду), кількість каналів (моно або стерео) і т. п.

```
typedef struct {
        int type;   - тип звукових даних, буває - !!!
                1 - просто вибірка;
```

```
                0x101 - IBM mu-law;
                        0x102 - IBM a-law;
                        0x103 - ADPCM.
          int channels; - число каналів 1/2 - !!!
          long SamplesPerSec; - частота вибірки - !!!
          long AvgBytesPerSec; - частота видачі байтів
          int align; - вирівнювання
          int bits; - число біт на вибірку  - !!!
  } IDWave;
```

Потім ключове слово data, після чого розташовуються дані.

```
typedef struct {
       char id[4]; - ідентифікатор ="data" =0x61746164
          long len;    - довжина вибірки ( кратно 2 )
  } IDSampleWave;
```

Якщо формат без стиснення, то ці дані можуть представляти 8-бітний (по байту на кожен відлік) або 16-бітний (по два байти на відлік) звук. Якщо число каналів більше одного, то відліки для кожного розташовуються один за одним, причому першим йде лівий канал, другим – правий. Настільки проста структура дозволяє використовувати WAV-файли для зберігання послідовностей оцифрованого сигналу не тільки для аудіопотреб, але й для інших (наприклад науково-технічних) цілей.

**Теоретична база алгоритму**

Для того щоб підійти до розробки нашого алгоритму, ми маємо визначити вимоги, які можуть бути і будуть пред'явлені до нашої стегосистеми. Саме ці вимоги і стануть критеріями для створення нашого методу, тому розглянемо їх детальніше:

– приховувана інформація має бути стійкою до наявності різних забарвлених шумів, стискування з втратами, фільтрування, аналогово-цифрових і цифро-аналогових перетворень;

– приховувана інформація не повинна вносити до сигналу спотворення, які сприймаються системою чуття людини;

– спроба видалення приховуваної інформації повинна приводити до явного пошкодження контейнера (для ЦВЗ);

– приховувана інформація не повинна вносити помітних змін до статистики контейнера;

Відповідно програмна реалізація алгоритму повинна задовільняти наведені вимоги, зберігаючи простоту використання та функціональність. Розглянемо теоретичну сутність алгоритму поблочної інтеграції ЦВЗ у цільовий файл шляхом заміни надлишкових бітів.

ЦВЗ впроваджується в аудіосигнали (послідовність 8- або 16-бітних відліків) шляхом незначної зміни амплітуди кожного відліку. Для виявлення ЦВЗ не вимагається вихідного аудіосигналу [2].

Нехай аудіосигнал складається з N відліків x (i), i = 1, ..., N, де значення N не менше 88200 (відповідно 1 секунда для стерео аудіосигналу, дискретизованого на частоті 44,1 кГц). Для того щоб вбудувати ЦВЗ, виконується функція f(x(i), w(i)), де w(i) – відлік ЦВЗ, що змінюється в межах [-$\alpha$; $\alpha$], $\alpha$ – деяка константа. Функція f повинна брати до уваги особливості системи слуху людини, щоб уникнути відчутних спотворень вихідного сигналу. Відлік результуючого сигналу отримується таким чином

$$(i) = x(i) + f(x\ (i), w(i)). \qquad (1.1)$$

Відношення сигнал-шум у цьому випадку обчислюється як

$$\mathrm{SNR} = 10 \log_{10} \frac{\sum_n x^2(n)}{\sum_n \left[x(n) - y(n)\right]^2}$$

Важливо зазначити, що застосовуваний у схемі генератор випадкових чисел повинен мати рівномірний розподіл. Стійкість ЦВЗ в загальному випадку підвищується зі збільшенням енергії ЦВЗ, але це збільшення обмежується зверху допустимим відношенням сигнал-шум.

Виявлення ЦВЗ відбувається наступним чином. Позначимо через S наступну суму

$$S = \sum_{i=1}^{N} y(i)w(i) . \tag{1.2}$$

Об'єднавши (1.1) та (1.2) отримуємо

$$S = \sum_{i=1}^{N} \left[x(i)w(i) + f(x(i), w(i))w(i)\right]. \tag{1.3}$$

Перша сума дорівнює нулю, якщо числа на виході ГСЧ розподілені рівномірно і математичне очікування значення сигналу дорівнює нулю. У більшості ж випадків спостерігається деяка відмінність, яка позначається $\Delta w$, що необхідно також враховувати.

Отже, (1.3) приймає вигляд

$$S = \sum_{i=1}^{N} \left[x(i)w(i) + f(x(i), w(i))w(i)\right].$$

Сума $\sum_{i=1}^{N-\Delta w} x(i)w(i)$ приблизно дорівнює нулю. Якщо до аудіосигналу не був впроваджений ЦВЗ, то S буде приблизно дорівнювати $\frac{\Delta w}{N}\sum_{i=1}^{N} x(i)w(i)$. З іншого боку, якщо в аудіосигнал був впроваджений ЦВЗ, то S буде приблизно дорівнювати $\frac{\Delta w}{N}\sum_{i=1}^{N} x(i)w(i) + \sum_{i=1}^{N} f(x(i), w(i))w(i)$. Але *x(i)* – це вихідний сигнал, що за умовою не може бути використаний у процесі виявлення ЦВЗ. Сигнал *x(i)* можна замінити на *y(i)*, що призведе до заміни $\sum_{i=1}^{\Delta w} x(i)w(i)$ на $\frac{\Delta w}{N} S$, а помилка при цьому буде незначною.

Отже, віднімаючи величину $\frac{\Delta w}{N} S$ з S, і поділивши результат на $\sum_{i=1}^{N} f(y(i), w(i))w(i)$, отримаємо результат r, нормований до 1. Детектор ЦВЗ, що використовується у цьому методі, обчислює величину r, що задається формулою

$$r \hat{=} \frac{S - \frac{\Delta w}{N}|S|}{\sum_{i=1}^{N} f(y(i), w(i))w(i)}.$$

Порогова величина виявлення теоретично лежить між 0 і 1, з урахуванням апроксимації цей інтервал зводиться до [0 - ε; 1 + ε]. Дослідним шляхом встановлено, що для того, щоб визначити, чи дійсно певний ЦВЗ знаходиться у сигналі, порогове значення ЦВЗ повинне бути вище 0,7 [6]. Якщо потрібна велика вірогідність у визначенні наявності ЦВЗ у сигналі, порогове значення необхідно збільшити.

На рис. 1 показана емпірична функція щільності ймовірності для аудіосигналу з ЦВЗ і без ЦВЗ.

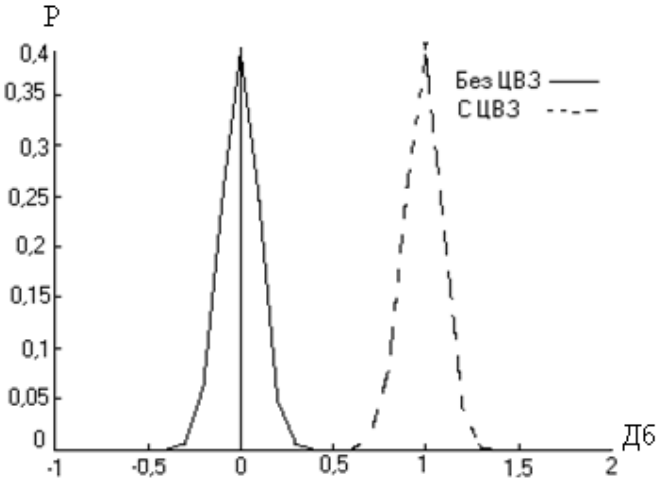


Рис. 1. Функція щільності розподілу величини виявлення для сигналів з ЦВЗ і без ЦВЗ

Емпірична функція щільності ймовірності аудіосигналу без ЦВЗ показана безперервною кривою, пунктирна крива описує емпіричну функцію щільності ймовірності аудіосигналу з вбудованим ЦВЗ. Обидва розподіли були обчислені з отриманням 1000 різних значень ЦВЗ при відношенні сигнал-шум 26 дб.

Впровадження в один аудіосигнал великої кількості різних ЦВЗ призводить до збільшення спотворень, які чуємо. Максимальне число ЦВЗ обмежено енергією кожного з них. Декодер здатний правильно відновити кожен ЦВЗ за умови використання кодера унікальних ключів. На рис. 2 показаний приклад виявлення ЦВЗ з використанням різних 1000-них ключів, з яких тільки один – вірний [1].

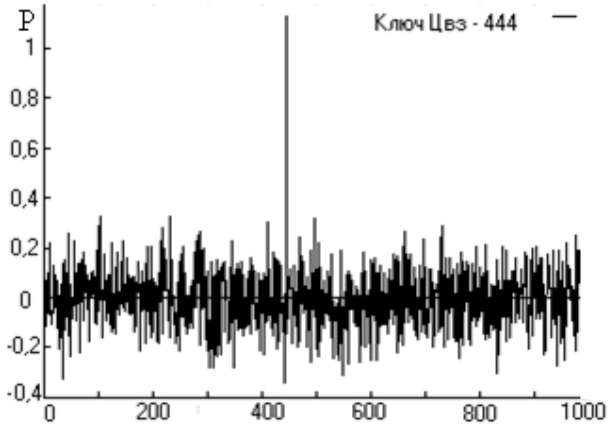


Рис. 2. Розпізнавання заданого ключа вбудовування ЦВЗ

Стійкість алгоритму вбудовування ЦВЗ до фільтрації перевірена застосуванням до нього ковзаючого фільтра середніх частот і фільтра нижніх частот. Аудіофайли з вбудованим ЦВЗ профільтровано ковзаючим фільтром середніх частот довжиною 20, який вносить до аудіоінформації значні спотворення.

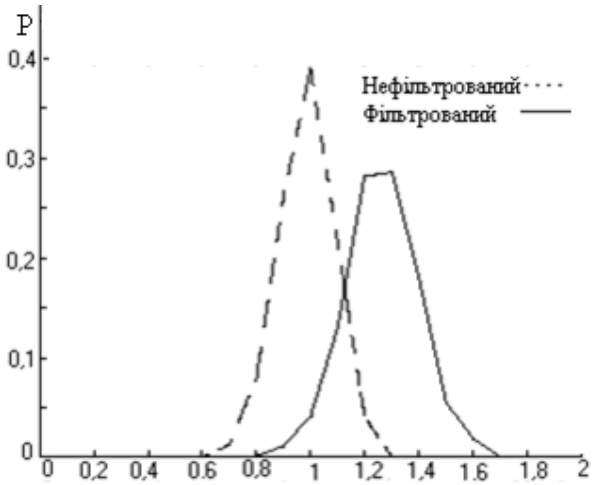


Рис. 3. Вплив на ЦВЗ застосування до аудіосигналів ковзкого фільтру середніх частот

На рис. 3 показано, як змінюється порогова величина виявлення при застосуванні вищеописаного фільтра. Поріг виявлення збільшується у відфільтрованих сигналах. Це відбувається через те, що функція щільності розподілу сигналів після фільтрації зсувається вправо у порівнянні з відносною функцією розподілу сигналів, що не піддавалися фільтрації.

ЦВЗ зберігається і при застосуванні до аудіосигналів фільтра нижніх частот. Однак, при фільтрації аудіосигналів з ЦВЗ фільтром нижніх частот Хеммінга 25-го порядку з частотою зрізу 2205 Гц мало місце зменшення ймовірності виявлення наявності ЦВЗ.

При переквантуванні аудіосигналу з 16-бітного у 8-бітний і назад упроваджений ЦВЗ зберігається, незважаючи на часткову втрату інформації. На рис. 4 показано наскільки добре ЦВЗ зберігається в 1000 аудіосігналах при їх переквантуванні у 8-бітні відліки і назад у 16-бітні.

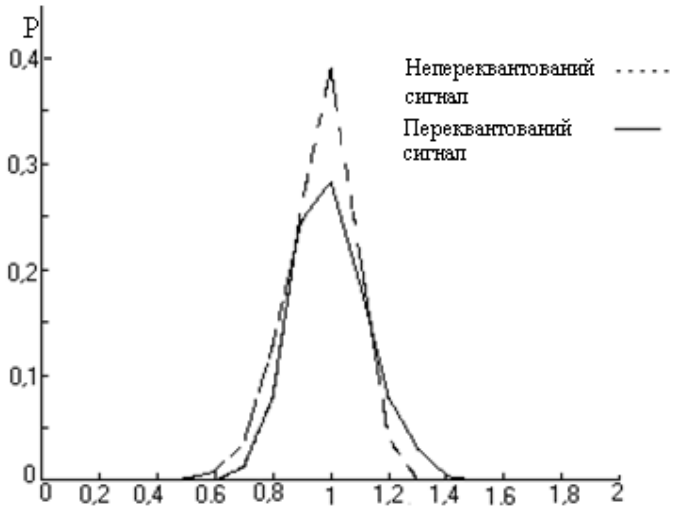


Рис. 4. Вплив переквантування сигналу на ЦВЗ

Девіація функції щільності розподілу переквантованого сигналу збільшується, як і у випадку застосування фільтра нижніх частот і має місце зменшення ефективності виявлення.

**Практична реалізація алгоритму[4]**

Розглянемо програмну реалізацію описаного вище алгоритму та фрагменти програмного коду,

описаного мовою C#.

Алгоритм побітного приховування ЦВЗ оснований на розбитті вхідного файлу на блоки однакового розміру та записі в останні байти даних блоків замаскованих бітів повідомлення. Запис повідомлення відбувається за рахунок збільшення або зменшення поточного байта вхідного файла на величину, що залежить від значення, отриманого в результаті логічного додавання байту повідомлення та індексу біту, що відповідає значенню ітератора основного циклу шифрування. Суть логічної операції полягає в наступному: в циклі for до байту повідомлення, що отримується з потоку файлу-повідомлення в циклі while, логічно додається значення двійкового представлення числа 1, зсунутого вліво на значення, відповідне поточній ітерації циклу.

```
  public void Hide(Stream messageStream, Stream keyStream)
  {
  byte[] waveBuffer = new byte[bytesPerSample];
  byte message, bit, waveByte;
  int messageBuffer;
  int keyByte;

  while( (messageBuffer=messageStream.ReadByte()) >= 0 ){
  //читаємо 1 байт з повідомлення
  message = (byte)messageBuffer;

  //а тепер для кожного біта
  for(int bitIndex=0; bitIndex<8; bitIndex++){

  //читаємо байт з ключа
  keyByte = GetKeyValue(keyStream);

  for(int n=0; n<keyByte-1; n++){
  //копіюємо одну частину
  sourceStream.Copy(waveBuffer,          0,          waveBuffer.Length,
destinationStream);
  }

  sourceStream.Read(waveBuffer, 0, waveBuffer.Length);
  waveByte = waveBuffer[bytesPerSample-1];

  //беремо наступний біт повідомлення
  bit = (byte)(((message & (byte)(1 << bitIndex)) > 0) ? 1 : 0);
```

Порівнюючи отримане значення зі значенням 0, тобто визначаючи знак цього числа, отримуємо значення, що буде необхідне для перестановки байту вхідного файлу. Виходячи з отриманого значення та результату операції mod 2 з поточним байтом, виконуємо збільшення або зменшення цього байту.

```
  if((bit == 1) && ((waveByte % 2) == 0))
  {
       waveByte += 1;
  }
  else if((bit == 0) && ((waveByte % 2) == 1))
  {
```

```
        waveByte -= 1;
    }

    waveBuffer[bytesPerSample-1] = waveByte;
```

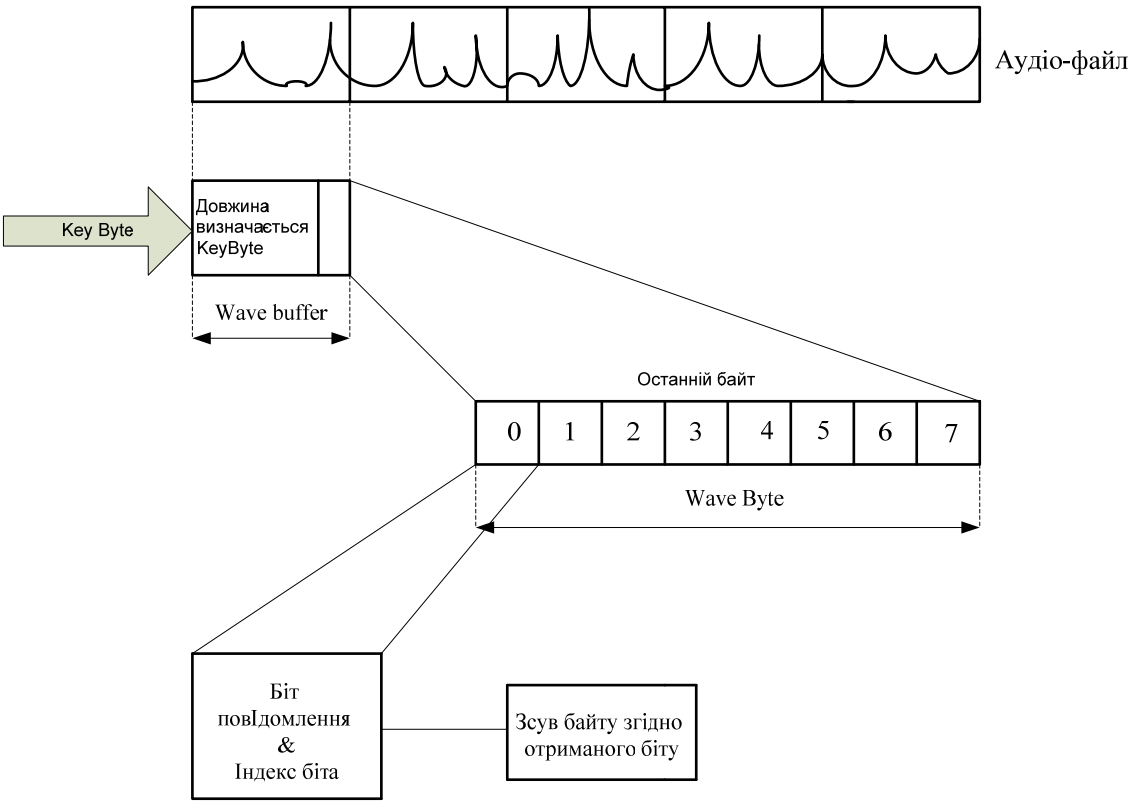


Рис. 5. Схема приховування повідомлення

Маючи зашифроване таким чином повідомлення, легко провести операцію зворотного дешифрування за наявності правильного ключа. Алгоритм дешифрування наступний: вхідний файл знову ділиться на блоки, що визначаються значенням ключа, для кожного блоку обирається останній байт в якості байта з прихованим повідомленням. Далі для отримання кожного з 8 біт, що входитимуть в байт повідомлення, виконуємо операцію mod 2 з поточним байтом. Отримане значення зсувається на величину, відповідну ітератору циклу, що буде визначати розряд цього біта. В цьому ж циклі до байту повідомлення додається отримане значення. Після виходу з циклу в потік повідомлення вноситься сформований байт.

Лістинг основного циклу отримання прихованого повідомлення наведений нижче:

```
for(int bitIndex=0; bitIndex<8; bitIndex++)
{
//читаємо байт ключа
keyByte = GetKeyValue(keyStream);
//Формуємо блок
for(int n=0; n<keyByte; n++)
{
sourceStream.Read(waveBuffer, 0, waveBuffer.Length);
}
//Отримуємо останій байт блоку
waveByte = waveBuffer[bytesPerSample-1];
//Отримуємо значення поточного біту
```

```
bit = (byte)(((waveByte % 2) == 0) ? 0 : 1);
//Записуємо в байт повідомлення отриманий біт у відповідний розряд
message += (byte)(bit << bitIndex);
}
```

**Висновки**

Наведений у статті алгоритм задовільняє всім поставленим вимогам, що визначають якісне приховування ЦВЗ та малий ризик їх виявлення. Розроблена на базі алгоритму тестова програма показала відсутність зміни розміру цільового файла, незначну відмінність вхідного та вихідного файлів як результат аналітичної оцінки файлів та візуального порівняння графіків амплітудно-часових характеристик файлів.

СПИСОК ЛІТЕРАТУРИ

1. Барсуков В. С. Компьютерная стеганография: вчера, сегодня, завтра / В. С. Барсуков // Специальная Техника, – 2000. – № 5 .– С. 35.

2. Хорошко В. О. Основи комп'ютерної стеганографії: Навчальний посібник / В. О. Хорошко, М. Є. Шелест, О. Д. Азаров, Ю. Є. Яремчук. – Вінниця: ВДТУ, 2003 – 143 с.

3. Ткаченко О. М. Об'єктно-орієнтоване програмування мовою Java: Навчальний посібник / О. М. Ткаченко, В. А. Каплун. – Вінниця: ВНТУ, 2006. – 106 с.

4. Інформаційний ресурс наукової групи «CNews Analytics» [Електронний ресурс]. // Режим доступу: http://www/cnews.ru.

5. Cox J., Miller M., McKellips A. Watermarking as communications with side information // Proceedings of the IEEE. 1999. Vol. 87. № 7. P. 1127-41 [Електронний ресурс] // Режим доступу: http://www.autex.spb.ru/wavelet/books/stego.zip.

6. Козлюк П. В. Розробка ефективного дискретного перетворення для потокової обробки / П. В. Козлюк // Прогресивні інформаційні технології в науці та освіті. Збірник наукових праць. – Вінниця: Вінницький соціально-економічний інститут Університету «Україна». – 2007.– С. 42 – 46.

***Дудатьєв Андрій Веніамінович –*** к. т. н., доцент кафедри захисту інформації.

***Козлюк Петро Володимирович –*** асистент кафедри захисту інформації.

***Оксимчук Дмитро Сергійович –*** студент гр. 1-ЗІ-06.
Вінницький національний технічний університет.