УДК 621.3.06

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

РАЗРАБОТКА АЛГОРИТМА СКРЫТИЯ ЦИФРОВЫХ ВОДЯНЫХ ЗНАКОВ В АУДИОФАЙЛАХ ФОРМАТА WAV

В статье проиллюстрирован процесс разработки и анализа стеганографического алгоритма, предназначенного для сокрытия текстовых сообщений или бинарных данных в аудиофайлах формата wav.

Ключевые слова: *ЦВЗ, стеганография, защита медиафайлов, стеганографические алгоритмы, шифрование данных, программирование C #.*

Вступление

Цифровой водяной знак (ЦВЗ) – технология, созданная для защиты авторских прав мультимедийных файлов. ЦВЗ в основном применяются для защиты от копирования и несанкционированного использования [1]. В связи с бурным развитием технологий мультимедиа остро встал вопрос защиты авторских прав и интеллектуальной собственности, представленной в цифровом виде. Примерами могут быть фотографии, аудио- и видеозаписи и т. д. В связи с воровством или модификацией сообщения преимуществ вставки и передачи сообщений в цифровом виде может не быть. Поэтому разрабатываются различные меры защиты информации организационного и технического характера. Один из наиболее эффективных технических средств защиты мультимедийной информации заключается во встраивании в объект невидимым меток – ЦВЗ. Разработки в этой области ведут крупнейшие фирмы во всем мире. В связи с тем что методы ЦВЗ начали разрабатываться совсем недавно, существует много проблем, требующих рассмотрения. Свое название этот метод получил от всем известного способа защиты ценных бумаг, в том числе и денег, от подделки (термин «digital watermarking») [5]. В отличие от обычных водяных знаков ЦВЗ могут быть не только видимыми, но и (как правило) невидимыми. Невидимые ЦВЗ анализируются специальным декодером, который принимает решение об их корректности. ЦВЗ могут содержать некоторый аутентичный код, информацию о владельце, или какую-либо управляющую информацию. Наиболее подходящими объектами защиты при помощи ЦВЗ являются неподвижные изображения, файлы аудио- и видеоданных [4].

Целью данной статьи является разработка алгоритма и программного обеспечения для сокрытия цифровых водяных знаков в аудиофайлах. В качестве целевого формата аудиофайлов рассматривается формат wav.

Рассмотрим структуру файлов этого формата.

Первые байты в файле WAV – это идентификаторы формата.

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

Приведенный код, описанный на языке C #, демонстрирует программную реализацию структуры заголовка файла. Так же как и в случае других контейнеров, распознавание происходит именно по этим байтам, поэтому расширение файла может быть любым.

В простейшем случае после идентификационного заголовка в WAV-файле указывается размер и формат данных (на что отведено 24 байта), в том числе приводится величина битрейта (сколько отсчетов в секунду), количество каналов (моно или стерео) и 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;
```

Затем следует ключевое слово data, после чего располагаются данные.

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

Если формат без сжатия, то эти данные могут представлять 8-битный (по байту на каждый отсчет) или 16-битный (по два байта на отсчет) звук. Если число каналов больше одного, то отсчеты для каждого располагаются друг за другом, причем первым идет левый канал, вторым – правый. Столь простая структура позволяет использовать WAV-файлы для хранения последовательностей оцифрованного сигнала не только для аудиопотребностей, но и для других (например научно-технических) целей.

Теоретическая база алгоритма

Чтобы подойти к разработке нашего алгоритма, мы должны определить требования, которые могут быть и будут предъявлены к нашей стегосистеме. Именно эти требования и станут критериями для создания нашего метода, потому рассмотрим их более подробно:

– скрываемая информация должна быть стойкой к наличию различных окрашенных шумов, сжатию с потерями, фильтрованию, аналогово-цифровым и цифро-аналоговым преобразованиям;

– скрываемая информация не вносит в сигнал искажения, которые ощущаются системой чувств человека;

– попытка удаления скрываемой информации должна приводить к явному повреждению контейнера (для ЦВЗ);

– скрываемая информация не должна вносить заметных изменений в статистику контейнера;

Соответственно, программная реализация алгоритма должна удовлетворять приведенные требования, сохраняя простоту использования и функциональность.

Рассмотрим теоретическую сущность алгоритма поблочной интеграции ЦВЗ в целевой файл путем замены избыточных битов.

ЦВЗ внедряется в аудиосигналы (последовательность 8- или 16-битных отсчетов) путем незначительного изменения амплитуды каждого отсчета. Для выявления ЦВЗ не требуется выходного аудиосигнала [2].

Пусть аудиосигнал состоит из N отсчетов x (i), i = 1, ..., N, где значение N не менее 88200 (соответственно 1 секунда для стерео аудиосигнала, дискретизованого на частоте 44,1 кГц). Чтобы встроить ЦВЗ, выполняется функция f (x (i), w (i)), где w (i) – отсчет ЦВЗ, изменяющийся в пределах [- α; α], α – некоторая константа. Функция f должна учитывать особенности системы слуха человека, чтобы избежать ощутимых искажений выходного сигнала. Отсчет результирующего сигнала получается таким образом

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

Отношение сигнал-шум в этом случае вычисляется как

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

Важно отметить, что применяемый в схеме генератор случайных чисел должен иметь равномерное распределение. Устойчивость ЦВЗ в общем случае повышается с увеличением энергии ЦВЗ, но это увеличение ограничивается сверху допустимым отношением сигнал-шум.

Обнаружение ЦВЗ происходит следующим образом. Обозначим через S следующую сумму

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

Объединив (1.1) и (1.2) получаем:

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

Первая сумма равна нулю, если числа на выходе ГСЧ распределены равномерно и математическое ожидание значения сигнала равно нулю. В большинстве же случаев наблюдается некоторое отличие, которое обозначается Δw, что необходимо также учитывать.

Следовательно, (1.3) принимает вид

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

Сумма $\sum_{i=1}^{N-\Delta w} x(i)w(i)$ примерно равна нулю. Если в аудиосигнал не был внедрен ЦВЗ, то S будет примерно равна $\frac{\Delta w}{N}\sum_{i=1}^{N} x(i)w(i)$. С другой стороны, если в аудиосигнал был внедрен ЦВЗ, то S будет примерно равна $\frac{\Delta w}{N}\sum_{i=1}^{N} x(i)w(i) + \sum_{i=1}^{N} f(x(i), w(i))w(i)$. Но *x (i)* – это выходной сигнал, который по условию не может быть использован в процессе выявления ЦВЗ. Сигнал *x (i)* можно заменить на *y (i)*, это приведет к замене $\sum_{i=1}^{\Delta w} x(i)w(i)$ на $\frac{\Delta w}{N} S$, ошибка при этом будет незначительной.

Следовательно, вычитая величину $\frac{\Delta w}{N} S$ с S, и разделив результат на $\sum_{i=1}^{N} f(y(i), w(i))w(i)$, получим результат r, нормированный к 1. Детектор ЦВЗ, используемый в этом методе, вычисляет величину r, задаваемую формулой

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

Пороговая величина выявления теоретически лежит между 0 и 1, с учетом аппроксимации этот интервал сводится к [0 - ε; 1 + ε]. Опытным путем установлено, что для того, чтобы определить действительно ли определенный ЦВЗ находится в сигнале, пороговое

значение ЦВЗ должно быть выше 0,7 [6]. Если нужна большая вероятность в определении наличия ЦВЗ в сигнале, пороговое значение необходимо увеличить.

На рис. 1 показана эмпирическая функция плотности вероятности для аудиосигнала с ЦВЗ и без ЦВЗ.

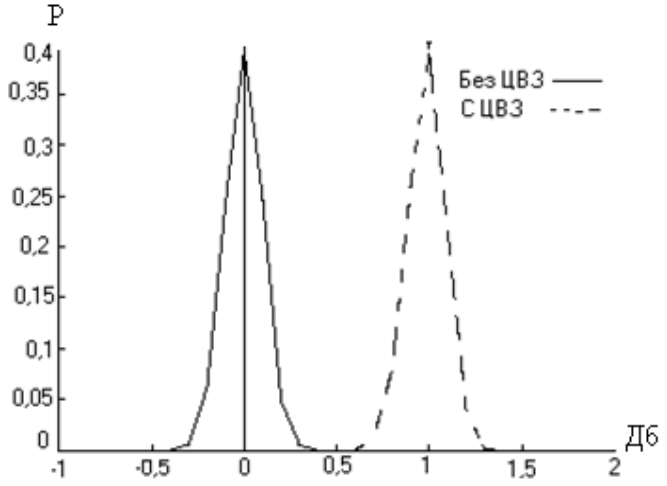

Рис. 1. Функция плотности распределения величины обнаружения для сигналов с ЦВЗ и без ЦВЗ

Эмпирическая функция плотности вероятности аудиосигнала без ЦВЗ показана непрерывной кривой, пунктирная кривая описывает эмпирическую функцию плотности вероятности аудиосигнала со встроенным ЦВЗ. Оба распределения были вычислены с получением 1000 различных значений ЦВЗ при отношении сигнал-шум 26 дБ.

Внедрение в один аудиосигнал большого количества различных ЦВЗ приводит к увеличению слышимых искажений. Максимальное число ЦВЗ ограничено энергией каждого из них. Декодер способен правильно восстановить каждый ЦВЗ при использовании кодера уникальных ключей. На рис. 2 показан пример выявления ЦВЗ с использованием различных 1000-ных ключей, из которых только один верный [1].

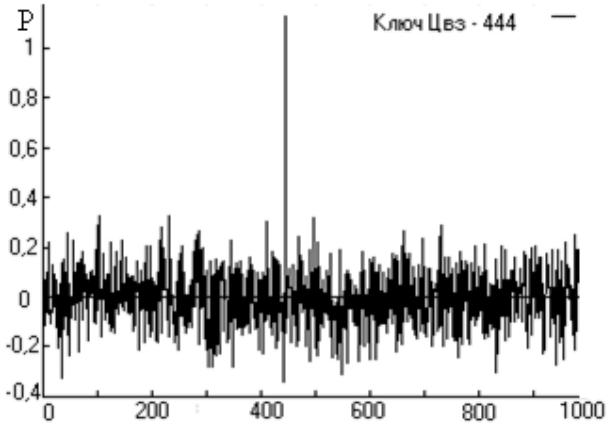

Рис. 2. Распознавание заданного ключа встраивания ЦВЗ

Стойкость алгоритма встраивания ЦВЗ к фильтрации проверена применением к нему скользящего фильтра средних частот и фильтра нижних частот. Аудиофайлы с встроенным ЦВЗ отфильтрованы скользящим фильтром средних частот длиной 20, который вносит в аудиоинформацию значительные искажения.

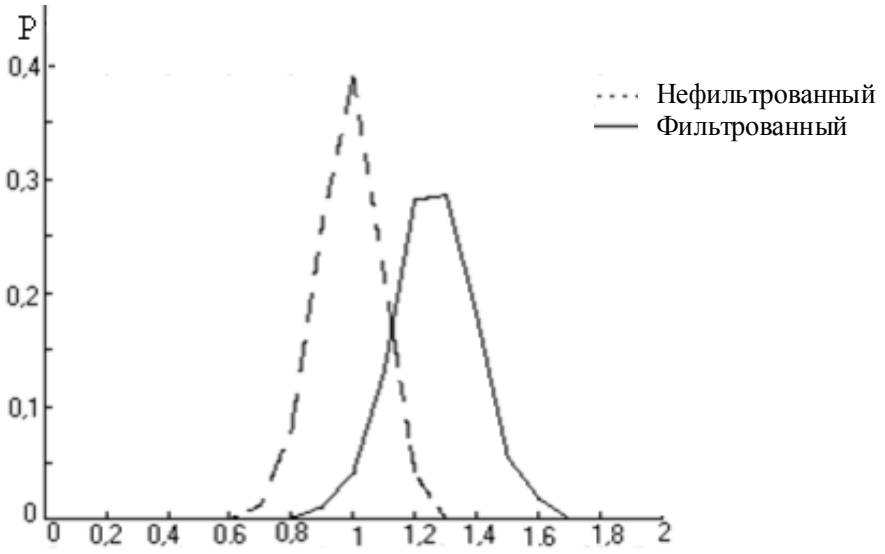

Рис. 3. Влияние на ЦВЗ применения к аудиосигналам скользящего фильтра средних частот

На рис. 3 показано, как изменяется пороговая величина обнаружения при применении вышеописанного фильтра. Порог обнаружения увеличивается в отфильтрованных сигналах. Это происходит потому, что функция плотности распределения сигналов после фильтрации сдвигается вправо по сравнению с относительной функцией распределения сигналов, не подвергавшихся фильтрации.

ЦВЗ сохраняется и при применении к аудиосигналам фильтра нижних частот. Однако при фильтрации аудиосигналов с ЦВЗ фильтром нижних частот Хемминга 25-го порядка с частотой среза 2205 Гц имело место снижение вероятности выявления наличия ЦВЗ.

При переквантовании аудиосигнала с 16-битного в 8-битный и обратно внедренный ЦВЗ сохраняется, несмотря на частичную потерю информации. На рис. 4 показано, насколько хорошо ЦВЗ хранится в 1000 аудиосигналах при их переквантовании в 8-битные отсчеты и обратно в 16-битные.

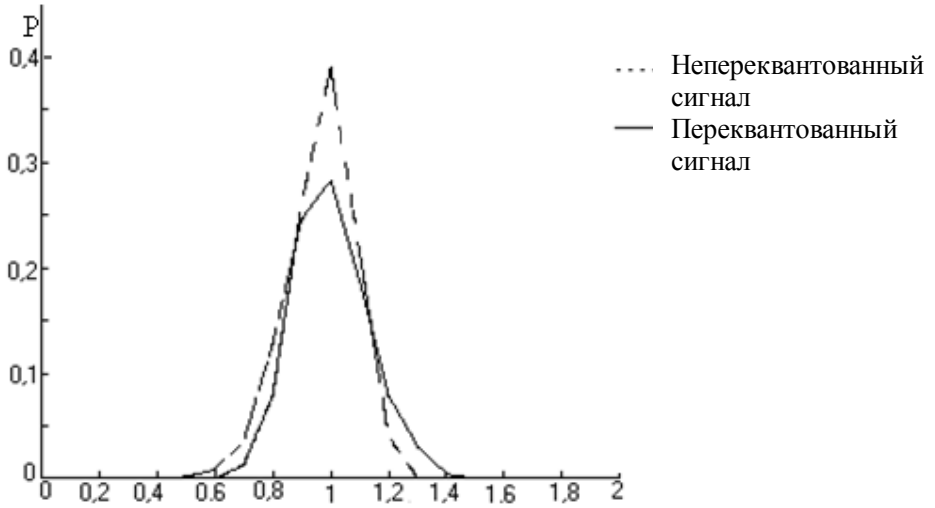

Рис. 4. Влияние переквантования сигнала на ЦВЗ

Девиация функции плотности распределения переквантованого сигнала увеличивается, как и в случае применения фильтра нижних частот, и имеет место снижение эффективности обнаружения.

Практическая реализация алгоритма [4]

Рассмотрим программную реализацию описанного выше алгоритма и фрагменты кода, описанного на языке C #.

Алгоритм побитного сокрытия ЦВЗ, основанный на разбиении входного файла на блоки одинакового размера и записи в последние байты данных блоков замаскированных битов сообщения. Запись сообщения происходит за счет увеличения или уменьшения текущего байта входного файла на величину, зависящую от значения, полученного в результате логического добавления байта сообщения и индекса бита, что соответствует значению итератора основного цикла шифрования. Суть логической операции заключается в следующем: в цикле for к байту сообщения, получаемого из потока файла-сообщения в цикле while, логично добавляется значение двоичного представления числа 1, сдвинутого влево на значение, соответствующее текущей итерации цикла.

```
  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);
```

Сравнивая полученное значение со значением 0, т. е. определяя знак данного числа, получим значение, что будет необходимо для перестановки байта входного файла. Исходя из полученного значения и результата операции mod 2 с текущим байтом, выполняем увеличение или уменьшение данного байта.

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

```
    waveByte -= 1;
}

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

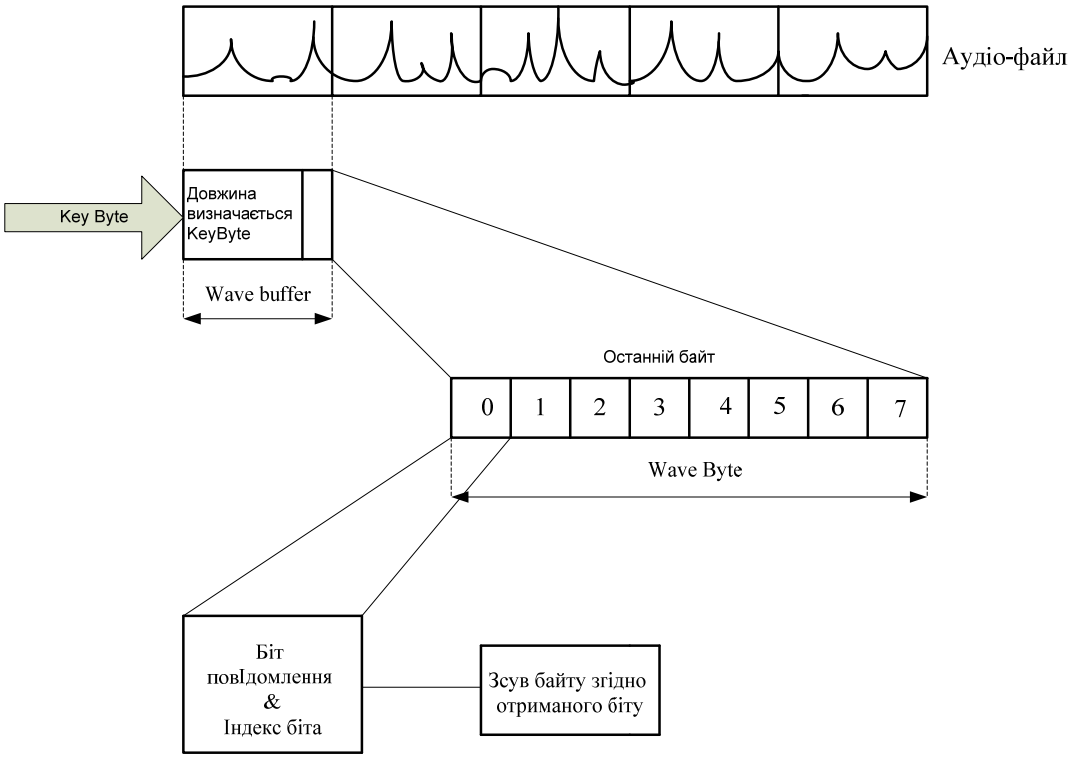

Рис. 5. Схема сокрытия сообщения

Имея зашифрованное таким образом сообщение, легко провести операцию обратного дешифрирования при наличии верного ключа. Алгоритм дешифровки следующий: входной файл снова делится на блоки, которые определяются значением ключа; для каждого блока выбирается последний байт в качестве байта со скрытым сообщением. Далее, для получения каждого из 8 бит, которые будут входить в байт сообщения, выполняем операцию mod 2 с текущим байтом. Полученное значение сдвигается на величину, соответствующую итератору цикла, и будет определять разряд этого бита. В этом же цикле к байту сообщения прилагается полученное значение. По выходу из цикла в поток сообщения вносится сформированный байт.

Листинг основного цикла получения скрытого сообщения приведен ниже:

```
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);
  }
```

Выводы

Приведенный в статье алгоритм удовлетворяет всем поставленным требованиям, определяющим качественное сокрытие ЦВЗ и малый риск их обнаружения. Разработанная на базе алгоритма тестовая программа показала отсутствие изменения размера целевого файла, незначительное отличие входящего и исходящего файлов как результат аналитической оценки файлов и визуального сравнения графиков амплитудно-временных характеристик файлов.

СПИСОК ЛИТЕРАТУРЫ

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.

Дудатьев Андрей Вениаминович – к. т. н., доцент кафедры защиты информации.

Козлюк Петр Владимирович – ассистент кафедры защиты информации.

Оксимчук Дмитрий Сергеевич – студент гр. 1-С-06.

Винницкий национальный технический университет.