

# Лабораторна робота № 4 Використання мнемонічних фраз для формування ключів шифрування

Група КН-М922в

Автор Савичев О.В.

## Мета

Дослідити і реалізувати механізм мнемонічних фраз для формування ключів шифрування

## Завдання

- Використовуючи алгоритм bip39, створити seed генератора псевдовипадкових чисел за допомогою мнемонічної фрази та стосовні ключі шифрування.
  - Зашифрувати текст
  - Використовуючи раніше створену мнемонічну фразу, відновити ключі шифрування на дешифрувати текст. Вдосконалитись, що оригінальний та дешифрований тексти однакові.
- З.І., для додаткових балів необхідно реалізувати підтримку україномовних мнемонічних фраз.

## Хід роботи

Мнемонічна фраза – це група слів, яка забезпечує доступ до ваших активів і є резервною копією для вашого крипто-гаманця (її часто також називають Seed-фразою або BIP39). Фраза може містити 12, 18, 24 слова. Найчастіше зустрічаються фрази на 12 та 24 слова.

BIP39 описує кроки, які потрібно зробити, щоб перетворити приватний ключ на мнемонічну фразу. Ця пропозиція стала стандартом для гаманців і в тому числі використовується безліччю інших криптовалютних проектів.

## Генерація випадкового ключа заданої довжини, та перетворення його у мнемонічну фразу

```
public string GenerateMnemonic(int strength)
{
    if (strength % 32 != 0)
        throw new NotSupportedException(InvalidEntropy);

    var rngCryptoServiceProvider = new RNGCryptoServiceProvider();

    byte[] buffer = new byte[strength / 8];
    rngCryptoServiceProvider.GetBytes(buffer);

    var entropyHex = BitConverter.ToString(buffer).Replace("-", "");

    return EntropyToMnemonic(entropyHex);
}
```

## Перетворення ключа у мнемонічну фразу

```
public string EntropyToMnemonic(string entropy)
{
    var wordlist = GetWordlist();

    var entropyBytes = Enumerable.Range(0, entropy.Length / 2)
        .Select(x => Convert.ToByte(entropy.Substring(x * 2, 2), 16))
        .ToArray();

    _CheckValidEntropy(entropyBytes);

    var entropyBits = BytesToBinary(entropyBytes);
    var checksumBits = _CalcChecksumBits(entropyBits);

    var bits = entropyBits + checksumBits;

    var chunks = Regex.Matches(bits, "({1,11})")
        .OfType<Match>()
        .Select(m => m.Groups[0].Value)
        .ToArray();

    var words = chunks.Select(binary =>
    {
        var index = Convert.ToInt32(binary, 2);
        return wordlist[index];
    });

    return string.Join(" ", words);
}
```

## Перетворення мнемонічної фрази у ключ

```
public byte[] MnemonicToEntropy(string mnemonic)
{
    var wordlist = GetWordlist();
    var words = mnemonic.Normalize(NormalizationForm.FormKD).Split(new[] { ' ' },
        StringSplitOptions.RemoveEmptyEntries);

    if (words.Length % 3 != 0)
        throw new FormatException(InvalidMnemonic);

    var bits = string.Join("", words.Select(word =>
    {
        var index = Array.IndexOf(wordlist, word);
        if (index == -1)
            throw new FormatException(InvalidMnemonic);

        return Convert.ToString(index, 2).PadLeft(11, '0');
    }));
}
```

```

// Виділення ENT (ентропії) / CS (контрольної суми)
var dividerIndex = (int)Math.Floor((double)bits.Length / 33) * 32;
var entropyBits = bits.Substring(0, dividerIndex);
var checksumBits = bits.Substring(dividerIndex);

// Розрахунок контрольної суми та порівняння
var entropyBytesMatch = Regex.Matches(entropyBits, "{1,8}")
    .OfType<Match>()
    .Select(m => m.Groups[0].Value)
    .ToArray();

var entropyBytes = entropyBytesMatch
    .Select(bytes => Convert.ToByte(bytes, 2)).ToArray();

_CheckValidEntropy(entropyBytes);

var newChecksum = _CalcChecksumBits(entropyBytes);

if (newChecksum != checksumBits)
    throw new Exception(InvalidChecksum);

return entropyBytes;
}

```

### Розрахунок бітів контрольної суми

```

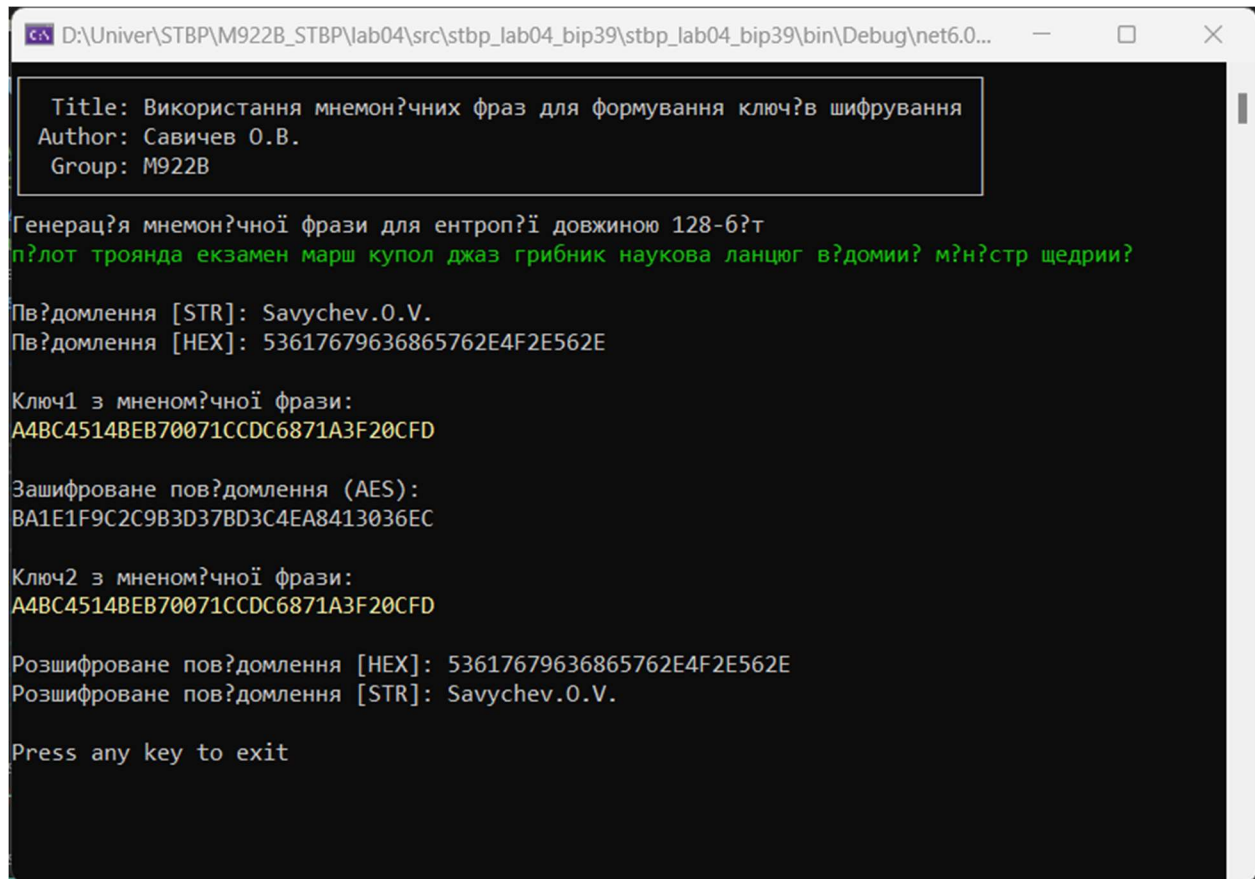
string _CalcChecksumBits(byte[] checksum)
{
    var ent = checksum.Length * 8;
    var cs = (int)ent / 32;

    using (var sha256Provider = SHA256CryptoServiceProvider.Create())
    {
        var hash = sha256Provider.ComputeHash(checksum);
        string result = BytesToBinary(hash);
        return result.Substring(0, cs);
    }
}

```

## Результати роботи

Результати роботи моєї реалізації алгоритму bip39



```
D:\Univer\STBP\M922B_STBP\lab04\src\stbp_lab04_bip39\stbp_lab04_bip39\bin\Debug\net6.0...

Title: Використання мнемонічних фраз для формування ключів шифрування
Author: Савичев О.В.
Group: M922B

Генерація мнемонічної фрази для ентропії довжиною 128-біт:
п'лот троянда екзамен марш купол джаз грибник наукова ланцюг в'дощи м'н'стр щедрий?

Повідомлення [STR]: Savychev.O.V.
Повідомлення [HEX]: 53617679636865762E4F2E562E

Ключ1 з мнемонічної фрази:
A4BC4514BEB70071CCDC6871A3F20CFD

Зашифроване повідомлення (AES):
BA1E1F9C2C9B3D37BD3C4EA8413036EC

Ключ2 з мнемонічної фрази:
A4BC4514BEB70071CCDC6871A3F20CFD

Розшифроване повідомлення [HEX]: 53617679636865762E4F2E562E
Розшифроване повідомлення [STR]: Savychev.O.V.

Press any key to exit
```

## Висновок

Дослідив і реалізував механізм асиметричного алгоритму шифрування RSA.