ქართულენოვანი ტექსტების შიფრაციისა და დეშიფრაციის პროგრამული უზრუნველყოფის შემუშავება BLOWFISH ალგორითმის საფუძველზე

ლელა გაჩეჩილაძე, გიორგი ჭოხონელიძე
საქართველოს ტექნიკური უნივერსიტეტი

რეზიუმე

წარმოდგენილია ქართულენოვანი ტექსტების შიფრაცია/დეშიფრაციის პროგრამული უზრუნველყოფა ობიექტზე ორიენტირებულ Java დაპროგრამების ენაზე. აღნიშნულ პროგრამულ პორდუქტში გამოყენებულია ცნობილი კრიპტოგრაფის ბრიუს შნეიერის მიერ შემუშავებული სიმეტრიული დაშიფვრის კრიპტოგრაფიული ალგორითმი სახელწოდებით BLOWFISH. ეს უკანასკნელი ცვლადი სიგრძის გასაღების მქონე 64-ბიტიან შიფრს წარმოადგენს, თუმა გასაღების გაფართოება 448 ბიტამდეა შესაძლებელი. ალგორითმი მაღალი საიმედოობით გამოირჩევა და იმ შემთხვევებში გამოიყენება, როდესაც გასაღების ხშირი ცვლილება საჭირო არ არის, ხოლო მონაცემების შიფრაცია/დეშიფრაციის პროცესი მაღალ სწრაფქმედებას მოითხოვს.

საკვანძო სიტყვები: კრიპტოგრაფია. ფეისტელის ქსელი. შიფრაცია. დეშიფრაცია.

1. შესავალი

კრიპტოგრაფიის ამოცანები მხოლოდ იმ ინფორმაციისთვის დაისმის, რომელიც დაცვას საჭიროებს. ასეთ შემთხვევაში ამბობენ, რომ ინფორმაცია წარმოადგენს პრივატულს (კერძოს), კონფიდენციალურს, და მოითხოვს დაცვას, უსაფრთხოებას.

კრიპტოგრაფია სამეცნიერო-ტექნიკური დარგია, რომელიც ინფორმაციის უსაფრთხოებას უზრუნველყოფს. მისი ძირითადი ამოცანაა გადაწყვიტოს ოთხი მთავარი პრობლემა: ინფორმაციის შიფრაცია-დეშიფრაცია, ანუ უსაფრთხოება- კონფიდენციალურობა, აუტენთიფიკაცია, მთლიანობა და მონაწილეთა (მომხმარებლების) ურთიერთობის კონტროლი [1].

შიფრაცია მონაცემების (ინფორმაციის) არაწაკითხვად ფორმაში გარდაქმნას გულისხმობს (ძირითადად მათემატიკური აპარატის მეშვეობით) შიფრაცია-დეშიფრაციის გასაღების საშუალებით.

კრიპტოსისტემის საფუძველს შეადგენს:

- გარკვეული მეთოდოლოგია, რომელიც შიფრაცია-დეშიფრაციის ალგორითმისა და ერთი ან მეტი გასაღებისაგან შედგება;
- გასაღებების განაწილების შესაბამისი სისტემა;
- ღია (დაუშიფრავი) ტექსტი;
- დაშიფრული ტექსტი (იგივე შიფროტექსტი).

გასაღები ალგორითმის რეალიზაციის ერთ-ერთი კონკრეტული სახეა. იგი თავდაპირველად ღია ტექსტის მიმართ გამოიყენება, რის შედეგადაც შიფროტექსტი მიიღება. შემდგომ ეს უკანასკნელი დანიშნულების მხარეს გადაეგზავნება, სადაც, ღია ტექსტის მისაღებად ფაქტიურად, იგივე გასაღები გამოიყენება დეშიფრაციისთვის. ამ მეთოდოლოგიით შიფრაციის ალგორითმში ღია ტექსტი გასაღებთან ერთად განიხილება, რაც შიფრო-

ტექსტს გვაძლევს. ასეთი კრიპტოსისტემის უსაფრთხოება გასაღების კონფიდენციალურობაზეა დამოკიდებული [2].

საზოგადოდ კრიპტოალგორითმი საყოველთაოდ ცნობილია (ღიაა) და ამასთან, კარგად აპრობირებული, მაგრამ გასაღები საიდუმლოა, რომელიც გარკვეული პერიოდის შემდეგ იცვლება.

2. ძირითადი ნაწილი

არსებობს ორი სახის კრიპტოსისტემა: სიმეტრიული (საიდუმლო გასაღებით), როდესაც გასაღების გაცვლა X და Y მხარეებს შორის საიდუმლო არხით ხდება და ასიმეტრიული (ღია გასაღებით). ყოველი სისტემა საკუთარ პროცედურას, გასაღებების ტიპს, მათი განაწილების მეთოდოლოგიასა და შიფრაციის ალგორითმს იყენებს.

სიმეტრიული კრიპტოსისტემის შემთხვევაში შიფრაციის და დეშიფრაციის გასაღები ერთი და იგივეა, ხოლო ასიმეტრიული კრიპტოსისტემის დროს შიფრაციის გასაღები დეშიფრაციის გასაღებისაგან განსხვავებულია.

ქართულენოვანი ტექსტის შიფრაცია-დეშიფრაციისთვის პროგრამული უზრუნველყოფის შემუშავების მიზნით ჩვენ სიმეტრიული შიფრაცია/დეშიფრაციის კრიპტოგრაფიულ ალგორთმს სახელწოდებით BLOWFISH-ს განვიხილავთ. მონაცემთა შიფრაციის ფუნქცია საკმაოდ მარტივია. იგი მიმდევრობით 16-ჯერ სრულდება. მასში 32-ბიტიანი სიტყვების შეკრების და ლოგიკური XOR (გამომრიცხავი „ან“) ოპერაციები გამოიყენება. ალგორითმი მრავალ ქვეგასაღებს მიმართავს. პრაქტიკულად, გასაღებების P მასივი თვრამეტი 32-ბიტიანი ქვეგასაღებისგან შედგება [3].

ყოველ ეტაპზე სრულდება გასაღებზე დამოკიდებული გადანაცვლება და გასაღებზე დამოკიდებული ჩანაცვლება. ამიტომ, სანამ ალგორითმი მონაცემთა დაშიფრვას დაიწყებს, მანამდე საიდუმლო გასაღების საშუალებით აუცილებელია საწყისი თექვსმეტი ეტაპისთვის საჭირო 18 ოცდათორმეტბიტიანი ქვეგასაღებების და ჩანაცვლების ოთხი S ბლოკის მომზადება, რომელთა საშუალებითაც ეტაპობრივი დაშიფრვა უნდა განხორციელდეს. ყოველი 32-ბიტიანი S ბლოკი 256 ელემენტს შეიცავს. თითოეულ ეტაპზე თითო ქვეგასაღები გამოიყენება, ანუ თექვსმეტი ეტაპისთვის თექვსმეტი ქვეგასაღებია საჭირო და კიდევ ორი ქვეგასაღები დამამთავრებელი გარდაქმნისთვის. ასე, რომ ქვეგასაღებების საერთო სიგრძე 576 ბიტს შეადგენს.

ალგორითმის მიხედვით, დასაშიფრი 64 ბიტი ორ, მარცხენა და მარჯვენა ქვებლოკებად იყოფა. მარცხენა ნახევარი (32 ბიტი) P1 ქვეგასაღებთან XOR ოპერაციით შეიკრიბება და შედის S ბლოკში. მისგან გამოსული 32 ბიტი ტექსტი მარჯვენა 32 ბიტთან იკრიბება. ამის შემდეგ მარჯვენა და მარცხენა ქვებლოკები ადგილებს ცვლიან და იგივე პროცედურა კიდევ თხუთმეტჯერ მეორდება. თექვსმეტი იტერაციის შემდეგ მიღებული 32-ბიტიანი მარჯვენა და მარცხენა ნახევრები P17 და P18 ქვეგასაღებებთან კიდევ ერთხელ შეიკრიბება, რის შემდეგაც მოხდება მათი კონკატაცია და 64 ბიტი დაშიფრული ტექსტი მიიღება [4].

დეშიფრაცია ზუსტად ისევე ხორციელდება, როგორც შიფრაცია (იმიტომ, რომ ალგორითმი ჰ. ფეისტელის სქემას იყენებს), ოღონდ გასაღებები შებრუნებული მიმდევრობით გამოიყენება [5].

ქვემოთ წარმოდენილია Java ენაზე შემუშავებული პროგრამა (ლისტინგი_1), რომელიც BLOWFISH ალგორითმის საფუძველზე ახდენს ქართულენოვანი ტექსტის შიფრაცია-დეშიფრაციას.

```
//--- ლისტინგი_1 --- BLOWFISH ----
public class blowfish {
      // გასაღები
        class blf_ctx {
           int S[][] = new int[4][256];
           int P[] = new int[18];
         }
          // მუდმივები  - pi რიცხვი
// მიმდინარე გასაღები
        private blf_ctx ctx;
          /**
          * Constructor
             * @param key
          */
        public blowfish(byte[] key) {
           this.ctx = new blf_ctx();
           // P მასივის ინიციალება
          System.arraycopy(ps, 0, this.ctx.P, 0, 18);
           // S მასივის ინიციალება
          System.arraycopy(ks0, 0, this.ctx.S[0], 0, 256);
           System.arraycopy(ks1, 0, this.ctx.S[1], 0, 256);
           System.arraycopy(ks2, 0, this.ctx.S[2], 0, 256);
           System.arraycopy(ks3, 0, this.ctx.S[3], 0, 256);
           // გასაღების გენერირება
          int data;
           int j = 0, i;
           for (i = 0; i < 18; ++i) {
             data = 0x00000000;
             for (int k = 0; k < 4; ++k) {
               data = (data << 8) | (key[j] & 0xFF);
               j++;
               if (j >= key.length) j = 0;
             }
             this.ctx.P[i] ^= data;
           }
           byte[] b = new byte[8];
           for (i = 0; i < 18; i += 2) {
             encipher(b);
             this.ctx.P[i] = b2d(b, 0);
             this.ctx.P[i + 1] = b2d(b, 4);
           }
           for (i = 0; i < 4; ++i) {
```

```
      for (j = 0; j < 256; j += 2) {
        encipher(b);
        ctx.S[i][j] = b2d(b, 0);
        ctx.S[i][j + 1] = b2d(b, 4);
      }
    }
  }
   /**
   * function F
   * @param x
   * @return
  */
private int F(int x) {
    int a, b, c, d;
    d = x & 0xFF;
    x >>= 8;
    c = x & 0xFF;
    x >>= 8;
    b = x & 0xFF;
    x >>= 8;
    a = x & 0xFF;
    int y = this.ctx.S[0][a] + this.ctx.S[1][b];
    y ^= this.ctx.S[2][c];
    y += this.ctx.S[3][d];
    return y;
  }
   /**
   * სიტყვის დაძვრა (32 ბიტი) 4 ბიტით
   * @param b
   * @param p
   * @return
  */
private int b2d(byte[] b, int p) {
    int r = 0;
    r |= b[p + 3] & 0xFF;
    r <<= 8;
    r |= b[p + 2] & 0xFF;
    r <<= 8;
    r |= b[p + 1] & 0xFF;
    r <<= 8;
    r |= b[p] & 0xFF;
    return r;
  }
   /**
   * სიტყვის დაძვრა (32 ბიტი) 4 ბაიტით
   * @param a
   * @param b
```

```
     * @param p
     */
    private void d2b(int a, byte[] b, int p) {
        b[p] = (byte) (a & 0xFF);
        a >>= 8;
        b[p + 1] = (byte) (a & 0xFF);
        a >>= 8;
        b[p + 2] = (byte) (a & 0xFF);
        a >>= 8;
        b[p + 3] = (byte) (a & 0xFF);
    }
     /**
     * ბაიტების მასივის დაშიფვრა
     * @param data მონაცემები (8-ის ჯერადი სიგრძით)
     */
    public void encipher(byte[] data) {
        int blocks = data.length >> 3;
        for (int k = 0, p; k < blocks; k++) {
            p = k << 3;
            int Xl = b2d(data, p);
            int Xr = b2d(data, p + 4);
            int tmp;
            for (int i = 0; i < 16; i++) {
                Xl = Xl ^ this.ctx.P[i];
                Xr = F(Xl) ^ Xr;
                tmp = Xl;
                Xl = Xr;
                Xr = tmp;
            }
            tmp = Xl;
            Xl = Xr;
            Xr = tmp;
            Xr ^= this.ctx.P[16];
            Xl ^= this.ctx.P[17];
            d2b(Xl, data, p);
            d2b(Xr, data, p + 4);
        }
    }
    /**
     * ბაიტების მასივის გაშიფვრა
     * @param data მონაცემები (8-ის ჯერადი სიგრძით)
     */
    public void decipher(byte[] data) {
        int blocks = data.length >> 3;
        for (int k = 0, p; k < blocks; k++) {
            p = k << 3;
            int Xl = b2d(data, p);
```

```
            int Xr = b2d(data, p + 4);
            int tmp;
            for (int i = 17; i > 1; i--) {
              Xl = Xl ^ this.ctx.P[i];
              Xr = F(Xl) ^ Xr;
              tmp = Xl;
              Xl = Xr;
              Xr = tmp;
            }
            tmp = Xl;
            Xl = Xr;
            Xr = tmp;
            Xr ^= this.ctx.P[1];
            Xl ^= this.ctx.P[0];
            d2b(Xl, data, p);
            d2b(Xr, data, p + 4);
          }
        }
         /**
         * @return
        */
       public byte[] padding(byte[] a, int p) {
          int l = (a.length | 7) + 1;
          byte[] b = new byte[l];
          for (int i = 0; i < b.length; i++) b[i] = (byte) p;
          System.arraycopy(a, 0, b, 0, a.length);
          return b;
        }
      }
public class Test {
     public static void main(String[] args) {
          // გასაღების შექმნა
        blowfish key = new blowfish("my password 8)".getBytes());
          // ტექსტური შეტყობინება
      String message = "მე ვარ ქართველი და მაშასადამე, ვარ ევროპელი!";
          // შეტყობინება
        byte[] msg = key.padding(message.getBytes(), ' ');
          // ტექსტის ორიგინალის გამოტანა
        System.out.println("ორიგინალი: "+new String(msg));
          // ტექსტის დაშიფვრა
        key.decipher(msg);
          // დაშიფრული ტექსტის გამოტანა
        System.out.println("დაშიფრული: "+new String(msg));
          // ტექსტის დეშიფრაცია
        key.encipher(msg);
```

```
            // გაშიფრული ტესქტის გამოტანა
            System.out.println("გაშიფრული: "+new String(msg));
          }
        }
```

პროგრამის შესრულების შედეგს შემდეგი სახე აქვს:

ორიგინალი: მე ვარ ქართველი და მაშასადამე, ვარ ევროპელი!

დაშიფრული:

f□)LU□□_Y_/□□□l=g□>□□□□□n□□_`□t□’.□z□]T□□□□□aA_□~U_l1
□-□<ClT□□□$□kt□
□:□/□G N□:□□^K]_□f□< r□8□_□p□
□)_
□□}□□b□h□□□h□

გაშიფრული: მე ვარ ქართველი და მაშასადამე, ვარ ევროპელი!

3. დასკვნა

ამრიგად, მიღებული შედეგების მიხედვით შეგვიძლია დავასკვნათ, რომ ბრიუს შნეიერის მიერ შექმნილი კრიპტოგრაფიული ალგორითმის საფუძველზე ჩვენ მიერ შემუშავებული პროგრამული უზრუნველყოფა წარმატებით ახორციელებს ქართულენოვანი ტექსტების შიფრაცია-დეშიფრაციას.

ლიტერატურა - References – Литература:

1. Buchmann J. (2004). Introduction to Cruptography Springer Science+Business Media, New York.

2. Kelsey J., Schneier B., Wagner D. (1996). Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES // Advances in Cryptology — CRYPTO '96, LNCS, v. 1109, Springer-Verlag, pp. 237–251

3. Dan Boneh (Stanford University), Online Cryptography Courses, (2013)
https://www.coursera.org/course/crypto

4. Шнайер Б. (2002). Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си — М.: Триумф,

5. Баричев С. Г., Гончаров В. В., Серов Р. Е. (2011). Основы современной криптографии. 3-е изд. -М.: Диалог-МИФИ.

DEVELOP A GEORGIAN ARTICLES ENCRYPTION AND DECRYPTION SOFTWARE BASED ON BLOWFISH ALGORITHM

Gachechiladze Lela, Chokhonelidze George
Georgian Technical University

Summary

The paper represented a software of Georgian articles encryption / decryption in object-oriented Java programming language. The software product used a symmetric encryption cryptographic algorithm created by famous cryptography Bruce Schneier's with called BLOWFISH. This algorithm is the key variable in length with a 64-bit cipher, however, key 448 bit resolution are available. The algorithm is distinguished by high reliability and used in those cases, when frequent key change is required, and process of data encryption / decryption requires high speed.

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ШИФРОВАНИЯ И ДЕШИФРОВАНИЯ ГРУЗИНСКИХ ТЕКСТОВ НА ОСНОВЕ АЛГОРИТМА BLOWFISH

Гачечиладзе Л., Чохонелидзе Г.
Грузинский Технический Университет

Резюме

Представлено программное обеспечение шифрования/дешифрования Грузинских текстов на объектно-ориентированном языке программирования Java. В программном продукте использован криптографический алгоритм симметричного шифрования разработанный известным криптографом Брюсом Шнайером под нозванием BLOWFISH. Алгоритм представляет собой 64-битовый блочный шифр с ключом переменной длины, однако, расширение ключа возможна длиной до 448 битов. Алгоритм отличается высокой надежностью и используется в тех случаях, когда не требуется частая смена ключа, а сам процесс шифрования/дешифрования данных требует высокого быстродействия.