Autori: Marian Gușatu, Ruxandra F. Olimid
Departamentul de Informatică, Universitatea din București

- Laboratorul 8 -

Funcții hash

Disclaimer: Pe parcursul acestui curs/laborator vi se vor prezenta diverse noțiuni de securitate informatică, cu scopul de a învăța cum să securizați sistemele. Toate noțiunile și exercițiile sunt prezentate în scop didactic, chiar dacă uneori se presupune să gândiți ca un adversar. Nu folosiți aceste tehnici în scopuri malițioase! Acestea pot avea consecințe legale în cazul comiterii unor infracțiuni, pentru care **deveniți pe deplin răspunzători**!

1. *Noțiuni introductive*

Răspundeți cu *adevărat* sau *fals* pentru fiecare dintre următoarele afirmații. Căutați online informații despre funcțiile hash menționate.

a) *Amestecarea ingredientelor pentru realizarea unei prăjituri poate fi considerată one-way function.*
b) *Funcția hash MD5 este considerată sigură la coliziuni.*
c) *SHA256 este o funcție hash cu output pe 256 biți.*
d) *Valoarea hash SHA-1 pentru cuvantul „laborator” este 0x4bcc6eab9c4ecb9d12dcb0595e2aa5fbc27231f3.*
e) *Este corect să afirmăm că „o funcție hash criptează”.*
f) *O funcție hash folosită pentru stocarea parolelor trebuie să fie rapidă (i.e., să se calculeze rapid H(x) pentru x dat).*
g) *Hash-ul (fără salt) - 095b2626c9b6bad0eb89019ea6091bd9 – corespunde unei parole sigure, care nu ar fi susceptibilă spre exemplu la un atac de tip dicționar.*

2. *Securitatea funcției hash PHOTON-80/20/16*

Citiți despre familia de funcții hash PHOTON [1,2]. În continuare, problema se referă la varianta cu output pe 80 biți.

Urmați următorii pași:

- Descărcați *Reference-Implementation.zip* de pe Moodle; **Atenție**! Codul sursa este modificat fata de cel postat la [1] pentru a permite calculul funcțiilor hash pentru valori oarecare, citite din fișierul *input.txt;* Mai exact, se va calcula valoarea hash pentru fiecare linie din fisierul *input.txt*, si se va scrie in fișierul *output.txt*;

Ca exemplu de fișiere de input si output, vedeti *input_test.txt* si *output_test.txt*, postate pe Moodle;

- **Atentie!** Folosiți **implementarea bazată pe tabele** (*table-based implementation*), este mult mai rapidă, și versiunea functiei hash pe **80 de biti**. Mai exact:

```
icc / gcc / clang / etc. -D_PHOTON80_ -D_TABLE_ photon.c photondriver.c -o photon80 sha2.c timer.c -O3
```

- Ca valori estimative de execuție, pentru 1,6 GHz Intel Core i5, 2 cores - `231 cycles per byte` vs. -`1798 cycles per byte`. Pentru verificarea vitezei de execuție:

```
./photon80 -s
```

Atenție! Timpul de execuție depinde în mod direct de valoarea cycles per byte, deci o valoare cât mai mică vă permite să calculați mai repede un numar mai mare de hash-uri;

- Generați-vă un fișier *input.txt* care sa conțină **pe fiecare linie cate o valoare distinctă** (pentru aceasta se va calcula hash-ul). Un mod simplu de generare este sa porniți cu o valoare inițială (un cuvânt, o propoziție scurtă, etc.) și să adăugați la final un contor; orice alta modalitate de generare a fișierului este acceptată, cât timp valorile sunt **distincte și diferite de cele din *input_test.txt*;**
- Daca vreți să calculați și să stocați funcțiile hash pentru valorile din *input.txt* în *output.txt* folosiți:

```
./photon80 -f
```

- Ca valori extimative, pentru un fișier *output.txt* (rulat la 231 cycles per bye, cu input de tip *photon<contor>*):
  - 1 000 000 linii – 40 MB - timp de execuție: aprox. 10 sec
  - 10 000 000 linii – 410MB – timp de execuție: aprox. 1 min
  - 100 000 000 linii – 4.2GB – timp de execuție: 12 min

- Verificați pentru fiecare hash obținut egalitatea cu fiecare hash din *output_test.txt*; **Atenție!** Devine dificil să lucrați cu fișiere mari, deci puteți face verificarea direct la generarea valorii hash, fără stocarea acesteia în fișier. Analog, puteți face generarea string-ului de input direct, fără stocare si apoi citire din fișierul *input.txt;*

- Generați aprox. 10 000 000 valori de input, și verificați egalitatea valorilor hash obținute cu valorile hash din *output_test.txt*.

- Ați reușit să găsiți o coliziune? Vă așteptați la acest rezultat? De ce/de ce nu?

3. Stocarea parolelor

Studiați următoarele exemple de utilizare a parolelor, așa cum au fost definite / folosite de colegii din anii trecuți. Pentru fiecare, menționați posibile vulnerabilități, probleme sau greșeli de utilizare.

- *Exemplul 1*

```
}
@PostMapping("/url")
public ArrayList<String> crypt(@RequestBody Users json)
{

    final String secretKey = "bettyesuper";
    Iterable<Users> all_users = getAllUsers();
    for (Users user:all_users) {
        noncrypted_message = user.getPassword();
        encryptedmessage = enc.encrypt(noncrypted_message,secretKey);
        list_of_enc_passw.add(encryptedmessage);

    }
    return list_of_enc_passw;
}
```

- *Exemplul 2*

```
def creeazaCont():
    userName = input("Username: ")
    password = input("Password: ")
    hashedUsername = bcrypt.hashpw(userName.encode('utf-8'), bcrypt.gensalt())
    hashedPassword = bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt())
    existaDeja = 0
    for i in range(len(listaConturi)):
        if bcrypt.checkpw(userName.encode('utf-8'), listaConturi[i].userName):
            existaDeja = 1
            print("Acest user exista deja!")
            meniu()
    if existaDeja == 0:
        print("Contul a fost creat cu succes!")
        listaConturi.append(cont(hashedUsername, hashedPassword))
        meniu()
```

- *Exemplul 3*

```
    [HttpPost]
    [AllowAnonymous]
    [ValidateAntiForgeryToken]
    public async Task<ActionResult> Register(RegisterViewModel model)
    {
        if (ModelState.IsValid)
        {
```

```
                var user = new ApplicationUser { UserName = model.Name, Email = model.Em
ail, Name = model.Name };

                SHA256 sha = SHA256.Create();

                //transformam in binar si criptam
                byte[] binar = sha.ComputeHash(Encoding.UTF8.GetBytes(model.Password));

                //transformam textul criptat in hexadecimal
                StringBuilder b = new StringBuilder();
                for(int i=0; i<binar.Length; i++)
                {
                    b.Append(binar[i].ToString("x2"));
                }
                string pass = b.ToString();

                var result = await UserManager.CreateAsync(user, pass);
                if (result.Succeeded)
                {
                    // For more information on how to enable account confirmation and pa
ssword reset please visit https://go.microsoft.com/fwlink/?LinkID=320771
                    // Send an email with this link
                    // string code = await UserManager.GenerateEmailConfirmationTokenAsy
nc(user.Id);
                    // var callbackUrl = Url.Action("ConfirmEmail", "Account", new { use
rId = user.Id, code = code }, protocol: Request.Url.Scheme);
                    // await UserManager.SendEmailAsync(user.Id, "Confirm your account",
 "Please confirm your account by clicking <a href=\"" + callbackUrl + "\">here</a>");

                    return RedirectToAction("Index", "Home");
                }
                AddErrors(result);
            }

            // If we got this far, something failed, redisplay form
            return View(model);
        }
```

- *Exemplul 4*

```
    # Security settings
    app.config['SECURITY_CHANGEABLE'] = True
    app.config['SECURITY_CONFIRMABLE'] = False
    app.config['SECURITY_EMAIL_SENDER'] = ('ItoIT', 'contact@itoit.ro')
    app.config['SECURITY_RECOVERABLE'] = True
    app.config['SECURITY_REGISTERABLE'] = True
    app.config['SECURITY_PASSWORD_HASH'] = 'bcrypt'
    app.config['SECURITY_PASSWORD_SALT'] = '[redacted]'

    # Security email settings
    app.config['SECURITY_EMAIL_SUBJECT_REGISTER'] = 'Bine ai venit!'
    app.config['SECURITY_EMAIL_SUBJECT_PASSWORD_NOTICE'] = 'Parola ta a fost schimbată'
    app.config['SECURITY_EMAIL_SUBJECT_PASSWORD_RESET'] = 'Resetează-ți parola'
    app.config['SECURITY_EMAIL_SUBJECT_PASSWORD_CHANGE_NOTICE'] = 'Parola ta a fost schimbată'
```

- *Exemplul 5*

```
public static string MD5Hash(string parola)
    {

      //calculeaza hash
      MD5 md5 = new MD5CryptoServiceProvider();
      md5.ComputeHash(ASCIIEncoding.ASCII.GetBytes(parola));

      //stocheaza parola
      // ...
    }
```

Referințe bibliografice

1. The PHOTON Family of Lightweight Hash Functions. Accesibil la: https://sites.google.com/site/photonhashfunction/design
2. Guo, J., Peyrin, T., & Poschmann, A. (2011, August). The PHOTON family of lightweight hash functions. In *Annual Cryptology Conference* (pp. 222-239). Springer, Berlin, Heidelberg. Accesibil în varianta e-print la: https://eprint.iacr.org/2011/609.pdf