Computer Forensics and Security Fundamentals

Portfolio 2

Karan Nihalani, ID: 17023122

Contents

Introduction

What is hashing?

In this Computer Forensics portfolio, we were instructed to develop a hashing program, using Java programming language, but first, the definition of hashing must be made clear.

Hashing or a hash algorithm is a method of converting a long or short string of data into “a numeric string output of fixed length” (Federal Agencies Digitization Guidelines Initiative, 2017), which will be a hexadecimal output in the hashing program. There are different types of hash algorithms, such as, MD5 (Message Digest algorithm 5), SHA-1 (Secure Hash Algorithm 1) and SHA-256 (Secure Hash Algorithm 256). These are going to be the methods that are displayed in the program and the walkthrough.

Methods and Demo Walkthrough

In this program, text or files are hashed using the MD5, SHA-1 and SHA-256 algorithms, with the algorithms being imported at the very beginning of the code and further on, the text is stored in another file, which can be opened and referred to. You can see in **Fig.1** and **Fig.2**, how it is seen on the console and the code for the Main class, which prompts the user to input their choice of the hashing algorithm.

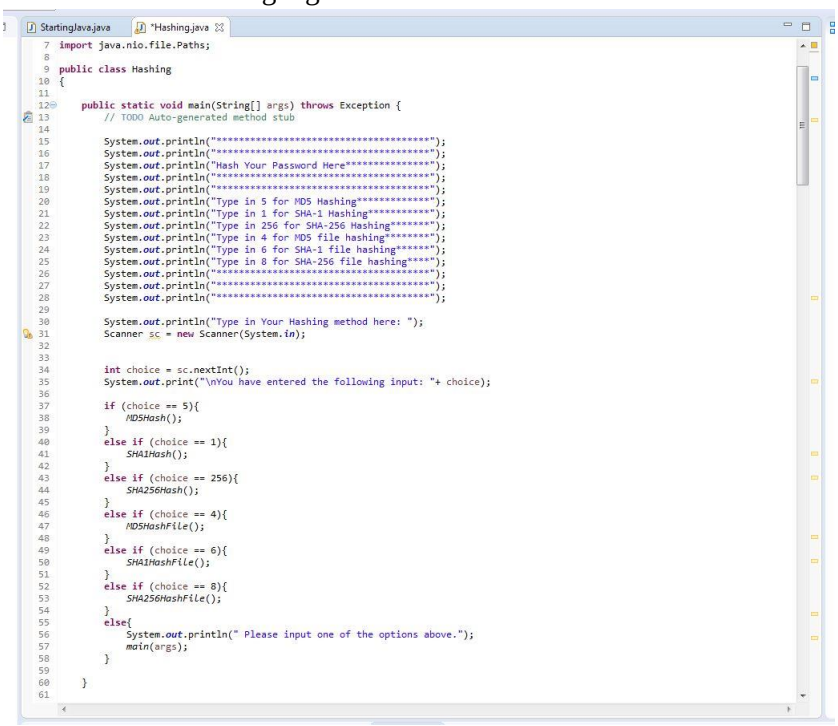

Fig.1

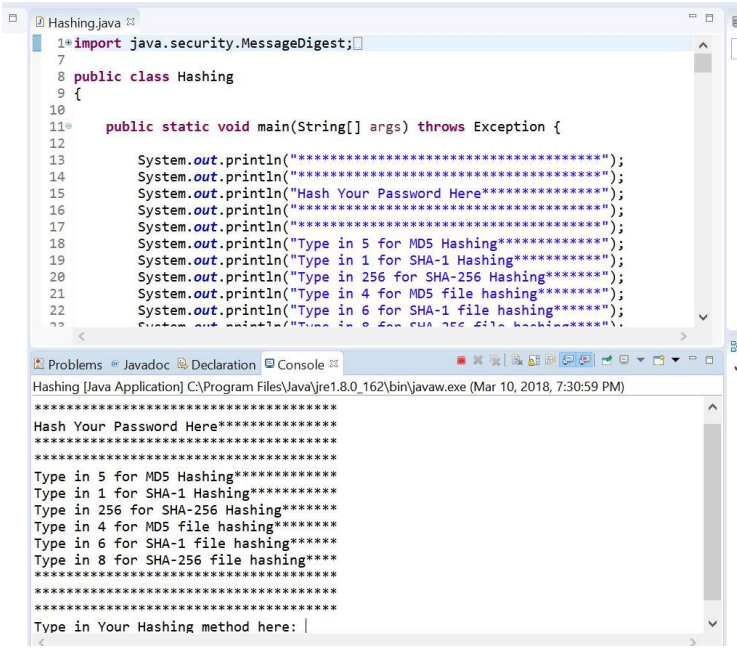

Fig.2

In the code, different if statements were used, so that depending on the choice, it calls the code for that hashing algorithm.

Plain Text Hashing (MD5, SHA-1, SHA-256)

To be able to hash plain text, the name for the input string is called ‘password’, it being the string the user wants to hash. Furthermore, we can select the algorithm we need for each method, using ‘MessageDigest.*getInstance*’ and ‘**byte** byteData[] = md.digest(),’ (**Fig.3**) is used to get the array of bytes from the string, outputting the hash value in hexadecimal (**Fig.4**). For all the hashes, MD5 “produces a 128-bit hash value, SHA-1 (160-bit) and SHA-256 (256 bit),” (codelatte, 2017).

```java
public static void SHA1Hash() throws Exception
{

    String password;

    MessageDigest md = MessageDigest.getInstance("SHA-1");

    System.out.println("Please Enter Your Password here: ");
    Scanner sc = new Scanner(System.in);

    password = sc.next();

    md.update(password.getBytes());

    byte byteData[] = md.digest();

    StringBuffer hexString = new StringBuffer();

    for (int i=0;i<byteData.length;i++) {
        String hex=Integer.toHexString( byteData[i] & 0xff );
        if(hex.length()==1) hexString.append('0');
        hexString.append(hex);
    }
```

Fig.3

```
Scanner sc = new Scanner(System.in);
```

```
Problems  Javadoc  Declaration  Console
erminated> MD5HashingExample [Java Application] C:\Program Files\Java\jre1.8.0_162\bin\javaw.exe (Mar 10
ease Enter Your Password here:
ran
gest(in hex format) for: karan - db068ce9f744fbb35eedc9a883f91085
```

Fig.4

File Hashing (MD5, SHA-1, SHA-256)

Moreover, a more advanced method used is file hashing, being able to hash more than one string at once. Similar syntax to plain text hashing is written, prompting the user to input the name of the file, at the beginning of the method and presenting an output message (Fig.5) when the file is hashed (Fig.6). The difference is that a 'FileInputStream' is used, for the file to be read and produce the hash value.

```java
}

public static void SHA1HashFile() throws Exception
{
    System.out.println(" Please enter the name of the file with extension (e.g. 123.txt) you want to hash? ");
    Scanner input = new Scanner(System.in);
    String inputstring = input.next();

    System.out.println("You have Entered: "+inputstring);

    MessageDigest md = MessageDigest.getInstance("SHA-1");
    String currentdir = Paths.get(".").toAbsolutePath().normalize().toString();
    String filepath;
    filepath= currentdir+"\\"+inputstring;
    System.out.println("file path is: "+filepath);

    FileInputStream fis = new FileInputStream(filepath);
       byte[] dataBytes = new byte[1024];
       int nread = 0;
       while ((nread = fis.read(dataBytes)) != -1) {
         // md is the MessageDigest instance
         md.update(dataBytes, 0, nread);
       };

     byte byteData[] = md.digest();

     StringBuffer hexString = new StringBuffer();
     for (int i=0;i<byteData.length;i++) {
          String hex=Integer.toHexString( byteData[i] & 0xff );
          if(hex.length()==1) hexString.append('0');
           hexString.append(hex);
                                        }

   System.out.println("The File Hash is: " + hexString.toString());
   main(null);
}

public static void SHA256HashFile() throws Exception
```

Fig.5

```java
System.out.println("The File Hash is: " + hexString.toString());
main(null);
}
```

```
Markers  Properties  Servers  Data Source Explorer  Snippets  Console  Debug
Hashing [Java Application] C:\Program Files\Java\jdk1.8.0_121\bin\javaw.exe (9 Mar 2018, 11:31:35)

You have entered the following input: 8 Please enter the name of the file with extension (e.g. 123.txt) you want to hash?
test.txt
You have Entered: test.txt
file path is: H:\Documents\Forensics Java\JavaHashingExercise\JavaHashing\test.txt
The File Hash is: 8bd07455a35f8f4fefb8a5a48701f4796659313810d4b53f466c790f5580e7ee
************************************
```

Fig.6

Storing hashes to a file

It's required to find a way to store the hashes that have been output into a text file within the directory. The classes 'PrintWriter' and 'FileWriter' are utilised. Here, 'FileWriter' has the ability to "write streams of characters" into a new file (Oracle, 2016). As shown in Fig.7, the word 'true' is added at the end of this section of the code to prevent the hashes from overwriting.

```java
public static void SHA256HashFile() throws Exception
{
     System.out.println(" Please enter the name of the file with extension (e.g. 123.txt) you want to hash? ");
     Scanner input = new Scanner(System.in);
     String inputstring = input.next();

     System.out.println("You have Entered: "+inputstring);

     MessageDigest md = MessageDigest.getInstance("SHA-256");
     String currentdir = Paths.get(".").toAbsolutePath().normalize().toString();
     String filepath;
     filepath= currentdir+"\\"+inputstring;
     System.out.println("file path is: "+filepath);

     FileInputStream fis = new FileInputStream(filepath);
        byte[] dataBytes = new byte[1024];
        int nread = 0;
        while ((nread = fis.read(dataBytes)) != -1) {
          // md is the MessageDigest instance
          md.update(dataBytes, 0, nread);
        };

      byte byteData[] = md.digest();

      StringBuffer hexString = new StringBuffer();
      for (int i=0;i<byteData.length;i++) {
           String hex=Integer.toHexString( byteData[i] & 0xff );
           if(hex.length()==1) hexString.append('0');
            hexString.append(hex);
                                        }

    PrintWriter outfile = new PrintWriter((new FileWriter ("H:/Documents/Forensics Java/JavaHashingExercise/JavaHashing/Hashes.txt", true)));

    outfile.println(hexString.toString());
    outfile.flush();
    outfile.close();

    System.out.println("The File Hash is: " + hexString.toString());
    main(null);
}
```

```
Hashing [Java Application] C:\Program Files\Java\jdk1.8.0_121\bin\javaw.exe (9 Mar 2018, 11:31:35)

You have entered the following input: 8 Please enter the name of the file with extension (e.g. 123.txt) you want to hash?
test.txt
You have Entered: test.txt
file path is: H:\Documents\Forensics Java\JavaHashingExercise\JavaHashing\test.txt
The File Hash is: 8bd07455a35f8f4fefb8a5a48701f4796659313810d4b53f466c790f5580e7ee
************************************
```

Fig.7

Reflection and Conclusion

At the beginning of this portfolio, I set out to accomplish a fully functional Java application of a hashing system that allowed a user to hash plain text and files or text documents, using the MD5, SHA-1 and SHA-256 algorithms. Further on, I even proceeded to develop the code more, so that the hash values that were output would be successfully stored in a separate file with the purpose of the values being viewed.

In conclusion, it is clear that hashing is useful to us as it has been used in cryptography, where these hashing algorithms can be "combined with standard cryptographic methods" to be able to check or "verify the source of data." (J.E. Silva-SANS Institute, 2003), which will definitely be extremely useful for the future of Cryptography and Computer Science.

References

Federal Agencies Digitization Guidelines Initiative. (2017) Hash Algorithm – Glossary. [Online][Accessed on 7th March 2018]
http://www.digitizationguidelines.gov/term.php?term=hashalgorithm

Codelatte. (2017) How to calculate MD5 hash in Java? [Online][Accessed on 10th March 2018]
http://www.codejava.net/coding/how-to-calculate-md5-and-sha-hash-values-in-java

Oracle. (2016) PriorityQueue (Java Platform SE 7). [Online][Accessed on 10th March 2018]
https://docs.oracle.com/javase/7/docs/api/java/io/FileWriter.html

SANS Institute – J.E. Silva. (2003) An Overview of Cryptographic Hash Functions and Their Uses. [Online][Accessed on 14th March 2018]
https://www.sans.org/reading-room/whitepapers/vpns/overview-cryptographic-hash-functions-879

Appendix

```
import java.security.MessageDigest;
import java.util.Scanner;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.nio.file.Paths;

public class Hashing
{

	public static void main(String[] args) throws Exception {

		System.out.println("*************************************");
		System.out.println("*************************************");
		System.out.println("Hash Your Password Here***************");
		System.out.println("*************************************");
		System.out.println("*************************************");
		System.out.println("Type in 5 for MD5 Hashing*************");
		System.out.println("Type in 1 for SHA-1 Hashing***********");
		System.out.println("Type in 256 for SHA-256 Hashing*******");
		System.out.println("Type in 4 for MD5 file hashing********");
		System.out.println("Type in 6 for SHA-1 file hashing******");
		System.out.println("Type in 8 for SHA-256 file hashing****");
		System.out.println("*************************************");
		System.out.println("*************************************");
		System.out.println("*************************************");

		System.out.println("Type in Your Hashing method here: ");
		@SuppressWarnings("resource")
		Scanner sc = new Scanner(System.in);


		int choice = sc.nextInt();
		System.out.print("\nYou have entered the following input: "+
choice);

		if (choice == 5){
			MD5Hash();
		}
		else if (choice == 1){
			SHA1Hash();
		}
		else if (choice == 256){
			SHA256Hash();
		}
		else if (choice == 4){
			MD5HashFile();
		}
		else if (choice == 6){
			SHA1HashFile();
		}
		else if (choice == 8){
			SHA256HashFile();
		}
```

```
        else{
                System.out.println(" Please input one of the options above.");
                main(args);
            }

    }



    @SuppressWarnings("resource")
    public static void MD5Hash() throws Exception
  {
    String password;

     MessageDigest md = MessageDigest.getInstance("MD5");

     System.out.print(" Please Enter Your Password here: ");
     Scanner sc = new Scanner(System.in);

     password = sc.next();

     md.update(password.getBytes());

     byte byteData[] = md.digest();

     StringBuffer hexString = new StringBuffer();

    for (int i=0;i<byteData.length;i++) {
          String hex=Integer.toHexString( byteData[i] & 0xff );
          if(hex.length()==1) hexString.append('0');
          hexString.append(hex);
    }

    PrintWriter outfile = new PrintWriter((new FileWriter
("H:/Documents/Forensics Java/JavaHashingExercise/JavaHashing/Hashes.txt",
true)));

    outfile.println(hexString.toString());
    outfile.flush();
          outfile.close();

    System.out.println("The hash has been added to 'Hashes.txt'");


    System.out.println("Digest(in hex format of MD5) for: "+password +" - "+
hexString.toString());
    main(null);
  }



    public static void SHA1Hash() throws Exception
  {

    String password;

     MessageDigest md = MessageDigest.getInstance("SHA-1");

     System.out.println("Please Enter Your Password here: ");
```

```
        @SuppressWarnings("resource")
        Scanner sc = new Scanner(System.in);

        password = sc.next();

        md.update(password.getBytes());

        byte byteData[] = md.digest();

        StringBuffer hexString = new StringBuffer();

       for (int i=0;i<byteData.length;i++) {
              String hex=Integer.toHexString( byteData[i] & 0xff );
              if(hex.length()==1) hexString.append('0');
              hexString.append(hex);
       }

        PrintWriter outfile = new PrintWriter((new FileWriter
("H:/Documents/Forensics Java/JavaHashingExercise/JavaHashing/Hashes.txt",
true)));

       outfile.println(hexString.toString());
       outfile.flush();
              outfile.close();

              System.out.println("The hash has been added to 'Hashes.txt'");

       System.out.println("Digest(in hex format of SHA-1) for: "+password +" - "+
hexString.toString());
       main(null);
    }


    public static void SHA256Hash() throws Exception
    {

       String password;

        MessageDigest md = MessageDigest.getInstance("SHA-256");

        System.out.println("Please Enter Your Password here: ");
        @SuppressWarnings("resource")
              Scanner sc = new Scanner(System.in);

        password = sc.next();

        md.update(password.getBytes());

        byte byteData[] = md.digest();

        StringBuffer hexString = new StringBuffer();

       for (int i=0;i<byteData.length;i++) {
              String hex=Integer.toHexString( byteData[i] & 0xff );
              if(hex.length()==1) hexString.append('0');
              hexString.append(hex);
       }
```

```
        PrintWriter outfile = new PrintWriter((new FileWriter
("H:/Documents/Forensics Java/JavaHashingExercise/JavaHashing/Hashes.txt",
true)));

      outfile.println(hexString.toString());
      outfile.flush();
            outfile.close();
            System.out.println("The hash has been added to 'Hashes.txt'");



      System.out.println("Digest(in hex format of SHA-256) for: "+password +" -
"+ hexString.toString());
      main(null);
    }

    public static void MD5HashFile() throws Exception
      {
             System.out.println(" Please enter the name of the file with
extension (e.g. 123.txt) you want to hash? ");
           @SuppressWarnings("resource")
            Scanner input = new Scanner(System.in);
           String inputstring = input.next();

             System.out.println("You have Entered: "+inputstring);

             MessageDigest md = MessageDigest.getInstance("MD5");
             String currentdir =
Paths.get(".").toAbsolutePath().normalize().toString();
             String filepath;
             filepath= currentdir+"\\"+inputstring;
             System.out.println("file path is: "+filepath);

             @SuppressWarnings("resource")
            FileInputStream fis = new FileInputStream(filepath);
             byte[] dataBytes = new byte[1024];
             int nread = 0;
             while ((nread = fis.read(dataBytes)) != -1) {
               // md is the MessageDigest instance
               md.update(dataBytes, 0, nread);
             };

           byte byteData[] = md.digest();

           StringBuffer hexString = new StringBuffer();
              for (int i=0;i<byteData.length;i++) {
                      String hex=Integer.toHexString( byteData[i] & 0xff );
                   if(hex.length()==1) hexString.append('0');
                  hexString.append(hex);
                                                 }

        PrintWriter outfile = new PrintWriter((new FileWriter
("H:/Documents/Forensics Java/JavaHashingExercise/JavaHashing/Hashes.txt",
true)));

      outfile.println(hexString.toString());
      outfile.flush();
            outfile.close();
```

```
            System.out.println("The File Hash is: " + hexString.toString());
            main(null);
        }

    public static void SHA1HashFile() throws Exception
        {
             System.out.println(" Please enter the name of the file with
extension (e.g. 123.txt) you want to hash? ");
           @SuppressWarnings("resource")
            Scanner input = new Scanner(System.in);
           String inputstring = input.next();

             System.out.println("You have Entered: "+inputstring);

             MessageDigest md = MessageDigest.getInstance("SHA-1");
             String currentdir =
Paths.get(".").toAbsolutePath().normalize().toString();
             String filepath;
             filepath= currentdir+"\\"+inputstring;
             System.out.println("file path is: "+filepath);

             @SuppressWarnings("resource")
            FileInputStream fis = new FileInputStream(filepath);
             byte[] dataBytes = new byte[1024];
             int nread = 0;
             while ((nread = fis.read(dataBytes)) != -1) {
               // md is the MessageDigest instance
               md.update(dataBytes, 0, nread);
             };

            byte byteData[] = md.digest();

            StringBuffer hexString = new StringBuffer();
              for (int i=0;i<byteData.length;i++) {
                      String hex=Integer.toHexString( byteData[i] & 0xff );
                   if(hex.length()==1) hexString.append('0');
                  hexString.append(hex);
                                                  }

        PrintWriter outfile = new PrintWriter((new FileWriter
("H:/Documents/Forensics Java/JavaHashingExercise/JavaHashing/Hashes.txt",
true)));

       outfile.println(hexString.toString());
       outfile.flush();
             outfile.close();

             System.out.println("The File Hash is: " + hexString.toString());
             main(null);
        }

    public static void SHA256HashFile() throws Exception
        {
             System.out.println(" Please enter the name of the file with
extension (e.g. 123.txt) you want to hash? ");
           @SuppressWarnings("resource")
            Scanner input = new Scanner(System.in);
           String inputstring = input.next();
```

```
                System.out.println("You have Entered: "+inputstring);

                MessageDigest md = MessageDigest.getInstance("SHA-256");
                String currentdir =
Paths.get(".").toAbsolutePath().normalize().toString();
                String filepath;
                filepath= currentdir+"\\"+inputstring;
                System.out.println("file path is: "+filepath);

                @SuppressWarnings("resource")
               FileInputStream fis = new FileInputStream(filepath);
                byte[] dataBytes = new byte[1024];
                int nread = 0;
                while ((nread = fis.read(dataBytes)) != -1) {
                  // md is the MessageDigest instance
                  md.update(dataBytes, 0, nread);
                };

              byte byteData[] = md.digest();

              StringBuffer hexString = new StringBuffer();
                 for (int i=0;i<byteData.length;i++) {
                         String hex=Integer.toHexString( byteData[i] & 0xff );
                       if(hex.length()==1) hexString.append('0');
                      hexString.append(hex);
                                                    }

         PrintWriter outfile = new PrintWriter((new FileWriter
("H:/Documents/Forensics Java/JavaHashingExercise/JavaHashing/Hashes.txt",
true)));

        outfile.println(hexString.toString());
        outfile.flush();
               outfile.close();

               System.out.println("The File Hash is: " + hexString.toString());
               main(null);
        }

}
```