Hashing Functions (MD5, SHA and so on) in Microsoft .NET

William J. Buchanan

Centre for Distributed Computing and Security,
Edinburgh Napier University
{w.buchanan}@napier.ac.uk
http://cdcs.napier.ac.uk

Abstract. This paper outlines the usage of hash functions, such as MD5 and SHA-1 in Microsoft .NET. It provides a basic overview of the encryption method, along with a review of other popular encryption methods and some sample code which can be used to implement hash methods.

1 Introduction

The future of the Internet, especially in expanding the range of applications, involves a much deeper degree of privacy, and authentication. Without these the Internet cannot be properly used to replace existing applications such as in voting, finance, and so on [3][2][1]. The future is thus towards data encryption which is the science of cryptographics, and provides a mechanism for two entities to communicate without any other entity being able to read their messages. In a secret communications system, Bob and Alice should be able to communicate securely, without Eve finding out the contents of their messages, or in keeping other details secure, such as their location, or the date that their messages are sent (Figure 1).

The two main methods used are to either use a unique algorithm which both Bob and Alice know, and do not tell Eve, or they use a well-known algorithm, which Eve also knows, and use some special electronic key to uniquely define how the message is converted into cipertext, and back again. A particular problem in any type of encryption is the passing of the secret algorithm or the key in a secure way, as Bob or Alice does not know if Eve is listening to their communications. If Eve finds-out the algorithm or the key, neither Bob nor Alice is able to detect this. This chapter looks at some of the basic principles of encryption, including the usage of private-key and public-key methods. As we will find public and private key methods work together in perfect harmony, with, typically, private key methods providing in the actual core encryption, and public key methods providing ways to authenticate, and pass keys.

There are many ways that encryption can be used in modern application, including encrypting data buckets in an e-Health Cloud [6], in digital forensics [7] and in information sharing [9]. New methods have also been created related to new ways to encrypting data using cumulative encryption [8], which supports

the usage of the encryption keys being added to the data so that it does not matter the one round that the ciper text is decrypted. For example if Alice encrypts with her key, and the Bob encrypts with his, then Alice or Bob can then apply their keys in any order, so that the data can be decrypted.

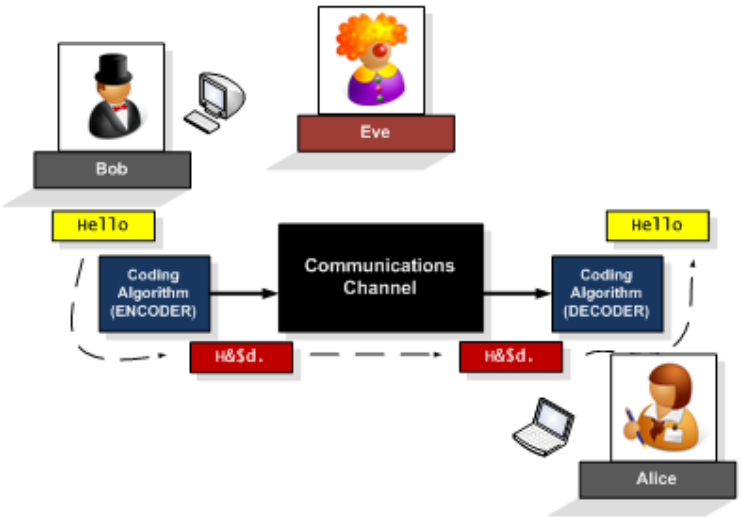

Fig. 1. Private key encryption

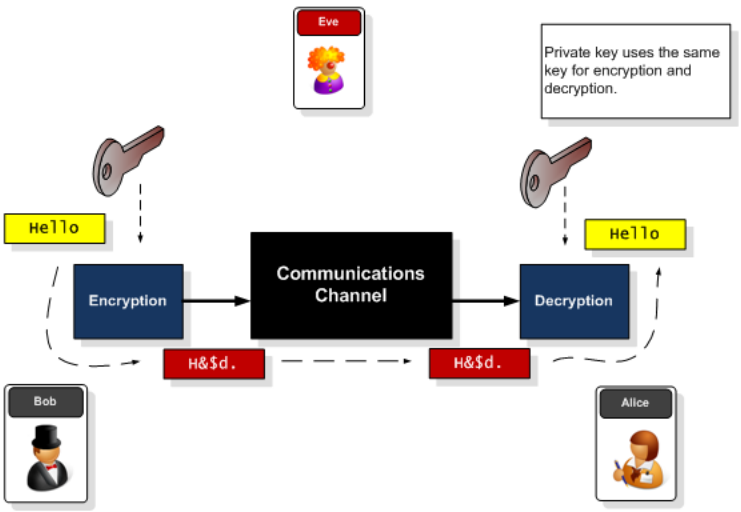

Fig. 2. Private key encryption

2 Hash Methods

One-way hashes are used for digital fingerprints and for secure password storage. Typical methods are NT hash, MD4, MD5, and SHA-1, and are used to convert plaintext into a hash value (Figure 3). It has applications in storing passwords, such as in Unix/Windows and on Cisco devices (Figure 4). A weakness of one-way hashing is that the same piece of plaintext will result in the same ciphertext (unless some salt is applied). Thus it is possible for an intruder to generate a list of hash values for a standard dictionary (Figure 5), and possibly determine the plaintext which makes the one-way hash. A major factor with hash signatures is:

- Collision. This is where another match is found, no matter the similarity of the original message. This can be defined as a Collision attack.
- Similar context. This is where part of the message has some significance to the original, and generates the same hash signature. The can be defined as a Pre-image attack.
- Full context. This is where an alternative message is created with the same hash signature, and has a direct relation to the original message. This is an extension to a Pre-image attack.

In 2006, for example, it was shown that MD5 can produce a collision within one minute, whereas it was 18 hours for SHA-1.

- Web link: http://buchananweb.co.uk/security03.aspx [MD5/SHA-1]
- Web link: http://buchananweb.co.uk/security03a.aspx [MD5/SHA-1]
- Web link: http://buchananweb.co.uk/security03b.aspx [MD5/SHA-1 with salt]

3 Coding

MD5 and SHA-1 methods produces a hash signature, and are the two of the most widely used methods. The MD5 algorithm has been show to have weaknesses, and a collision of message hashes has been shown to occur in less than one day. An MD5 signature has 128 bits, an SHA-1 signature has 160 bits, and an SHA-256 signature has 256 bits [4].

Listing 1.1. MD5 Code.

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
```

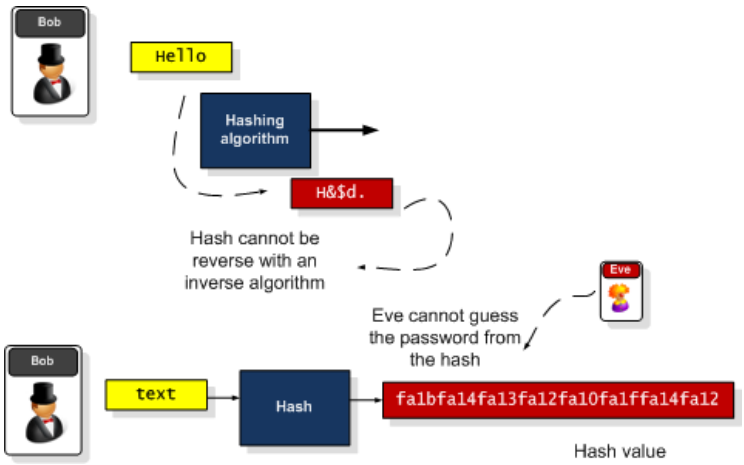

Fig. 3. One-way hashing

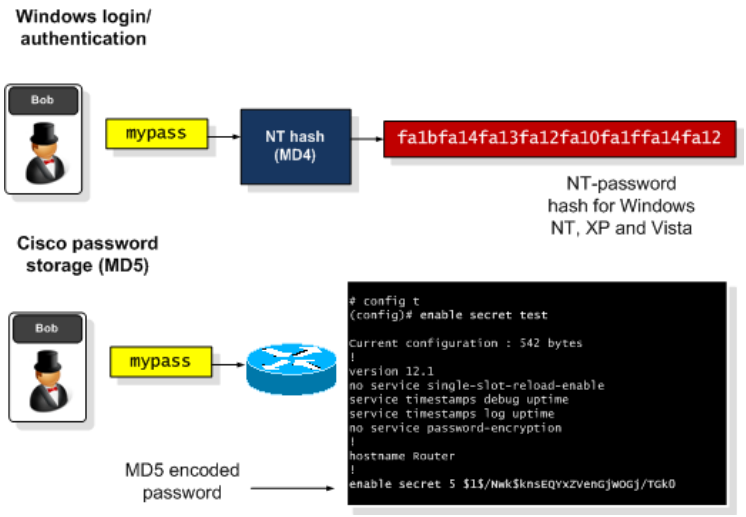

Fig. 4. Application of one-way hashing

```
using System.Web.UI.HtmlControls;
using System.Collections;
using System.Security.Cryptography;

public partial class _Default3 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

```

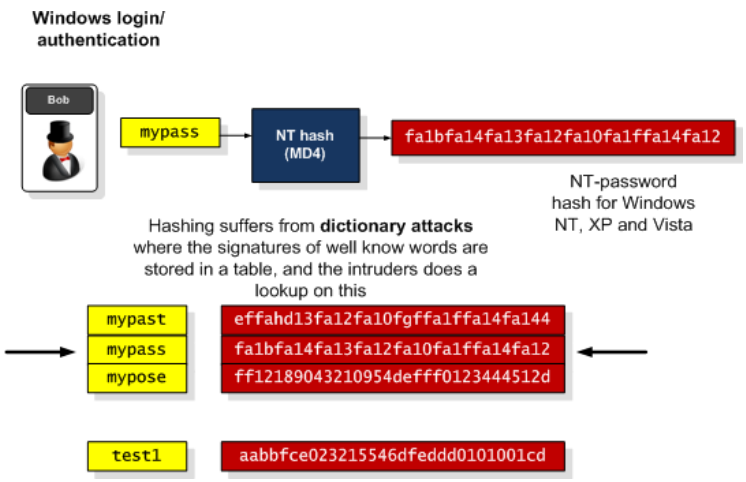

Fig. 5. Application of one-way hashing

```
    }
    protected void Button3_Click(object sender, EventArgs
        e)
    {
        string message;

        message = this.tbMessage.Text;

        System.Text.ASCIIEncoding encoding = new System.
            Text.ASCIIEncoding();

        MD5 md5 = new MD5CryptoServiceProvider();
        SHA1 sha1 = new SHA1CryptoServiceProvider();
        SHA256Managed sha256 = new SHA256Managed();
        SHA384Managed sha384 = new SHA384Managed();
        SHA512Managed sha512 = new SHA512Managed();

        byte[] messageBytes = encoding.GetBytes(message);

        byte[] hashmessage = md5.ComputeHash(messageBytes
            );


        this.tbMD5.Text = ByteToString(hashmessage);

        hashmessage = sha1.ComputeHash(messageBytes);

```

```
            this.tbSHA1.Text = ByteToString(hashmessage);

            hashmessage = sha256.ComputeHash(messageBytes);

            this.tbSHA256.Text = ByteToString(hashmessage);

            hashmessage = sha384.ComputeHash(messageBytes);

            this.tbSHA384.Text = ByteToString(hashmessage);

            hashmessage = sha512.ComputeHash(messageBytes);

            this.tbSHA512.Text = ByteToString(hashmessage);

        }
        public static string ByteToString(byte[] buff)
        {
            string sbinary = "";

            for (int i = 0; i < buff.Length; i++)
            {
                sbinary += buff[i].ToString("X2"); // hex
                    format
            }
            return (sbinary);
        }
}
```

4 Testing

As a test: "Hello" should give: 8b1a9953c4611296a827abf8c47804d7
and "hello" should give: 5d41402abc4b2a76b9719d911017c592

References

1. William J Buchanan. *Distributed Systems And Networks.* McGraw-Hill Higher Education, 2001.
2. William J Buchanan. *The complete handbook of the Internet.* Springer, 2002.
3. William J Buchanan. *Handbook of Data Communications and Networks.* Kluwer Academic Publishers, 2005.
4. William J Buchanan. Hashing encryption in .net. `http://buchananweb.co.uk/security03.aspx`, April 2011.
5. William J Buchanan. Security and forensic computing: Encryption. `http://buchananweb.co.uk/index_sfc_napier.html`, March 2011.

6. L Fan, W Buchanan, C. Thuemmler, O. Lo, A. Khedim, Uthmani O., A. Lawson, and D. Bell. Dacar platform for ehealth services cloud. *IEEE Cloud 2011*, 2011.
7. Z. Kwecka, W. Buchanan, and D. Spiers. Minimising collateral damage: Privacy-preserving investigative data acquisition platform. *International Journal of Information Technologies and Systems Approach (IJITSA) : Special issue on Privacy and Security Issues in IT*, 4(2), 2010.
8. Z. Kwecka, W. Buchanan, and D. Spiers. Privacy-preserving data acquisition protocol. *IEEE International Conference of Computational Methods in Electrical and Electronics Engineering*, pages 131 – 136, 2010.
9. O. Uthmani, W. Buchanan, A. Lawson, and L. Fan. Novel information sharing syntax for data sharing between police and community partners, using role-based security. *Proceedings of the 9th European Conference on Information Warfare and Security*, pages 394–402, 2010.