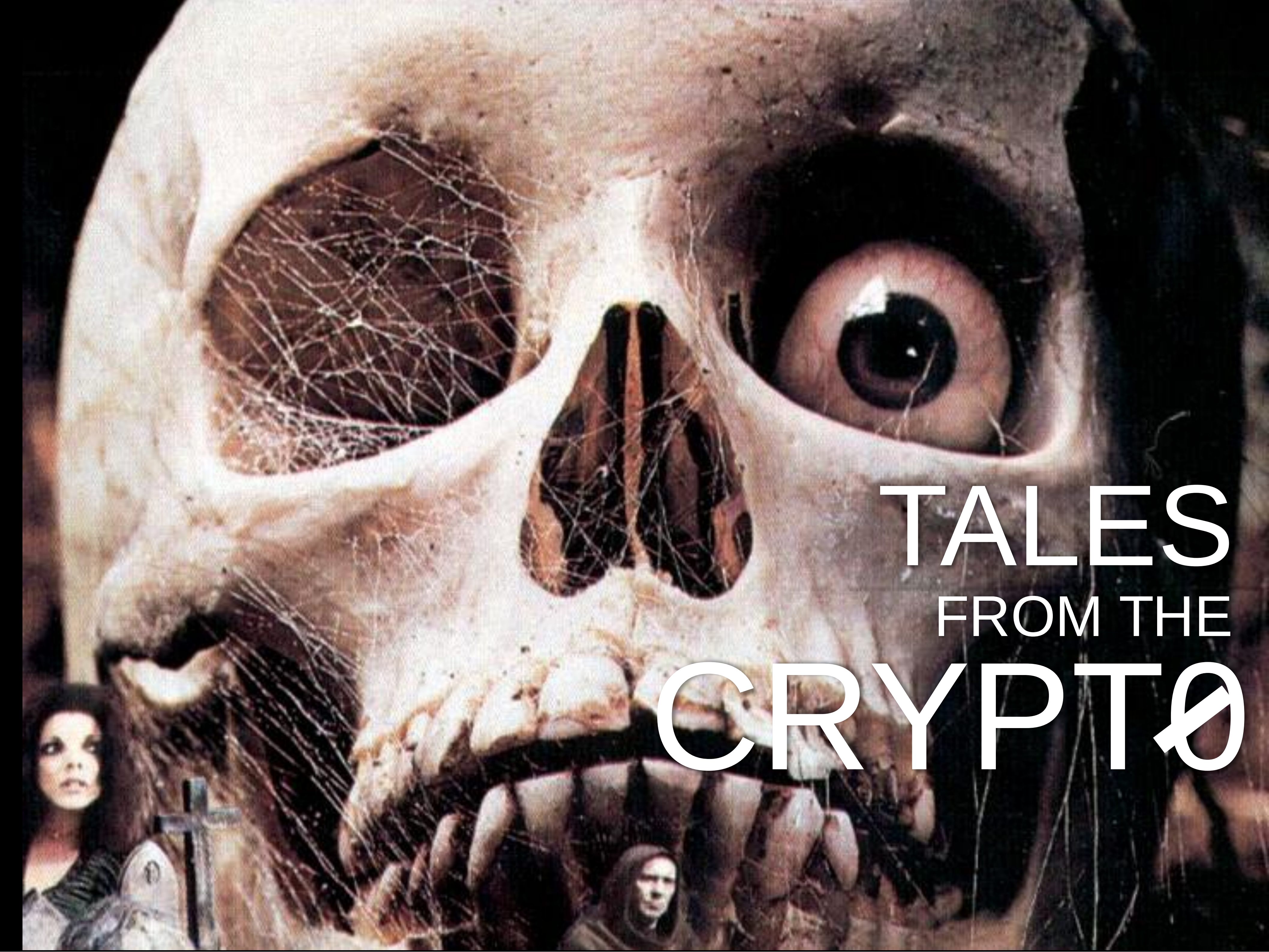


TALES
FROM THE
CRYPTØ

INTRODUCTION

Kirk Jackson
Xero
http://www.xero.com
http://www.codecamp.co.nz
http://www.ignitewellington.co.nz
http://pageofwords.com

Graeme Neilson
Aura Software Security
Security Researcher
http://www.aurasoftwaresecurity.co.nz

CRYPTOGRAPHY

What is it good for...?

◆ Confidentiality	...sssh it might hear you!
◆ Authentication	who's there...?
◆ Integrity	...I have not been possessed!
• Non-repudiation	who said that...?

When do I use...?

- ◆ Hashing
- ◆ Symmetric encryption
- ◆ Asymmetric cryptography

BEWARE OF IMPLEMENTATIONS NOT ALGORITHMS

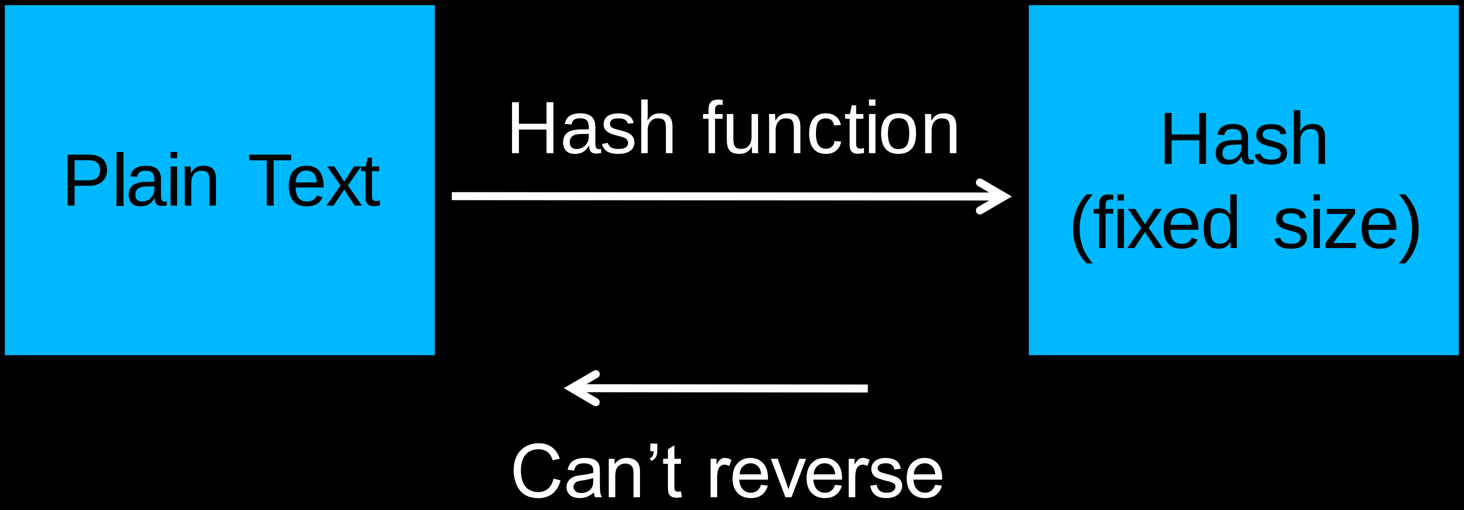
KERCKHOFF'S PRINCIPLE

THE SECURITY OF A SYSTEM SHOULD RESIDE ONLY IN THE KEY

DISCO PRINCIPLE

DON'T **I**NVENT **S**UPER **C**RYPTO OF YOUR **O**WN

Hashing



Plain Text

Hash function

Hash
(fixed size)

Can’t reverse

HASHING

One way functions for:

- Integrity checks
- Password storage

Using hash algorithms

- Use SHA family not MD5
- NIST hash competition

TALES FOM THE CRYPT0

- Juniper Netscreen password hash algorithm

JUNIPER NETSCREEN PASSWORD HASH

FIPS140-2 Security Policy for Netscreen 5400 and Netscreen-ISG 2000 states

“***The following non-approved algorithms/protocols are disabled in FIPS mode: RSA encryption/decryption, DES, MD5, SNMPv3***”

Algorithm:

- MD5 hash of username + “:Administration Tools:” + password
- Base64 encode the hash
- Insert the characters 'n' 'r' 'c' 's' 't' 'n' at fixed positions

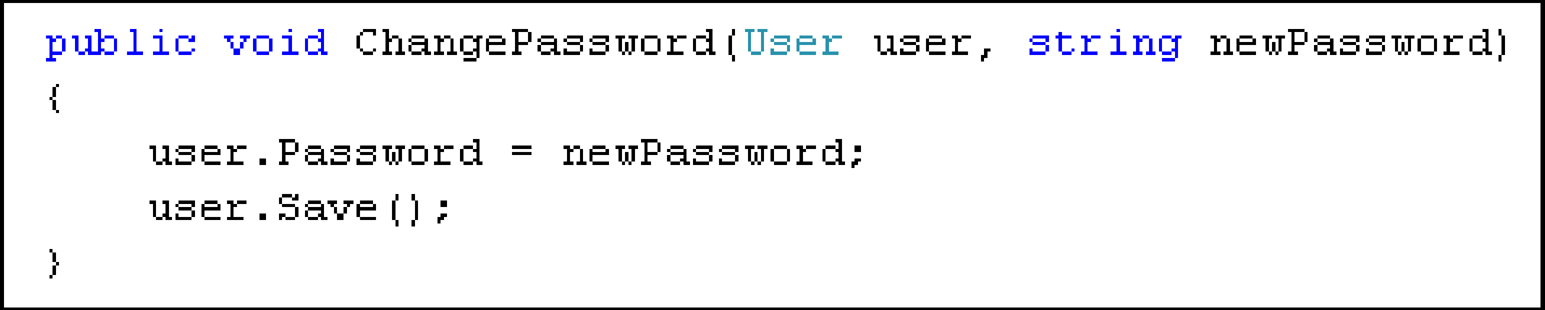
(netscreen backwards excluding the letter 'e')

```
nJ8aK7rVOo1Ico6CbsQFKNCtviAjTn
nPZmEerYEtdHcanJhsHGsSBtkrAV4n
nKqqMDroCJPBc8lF2smLmCMtnNCHRn
nNtMGWrpGPFJcNuMTsJKyPEtPhHVln
nKfNBWrbFpzNcaZAJs6M18HteGPUmn
nGH8EvrtD3/Dc4JDrsZEzyMtiFKLtn
```

Weaknesses

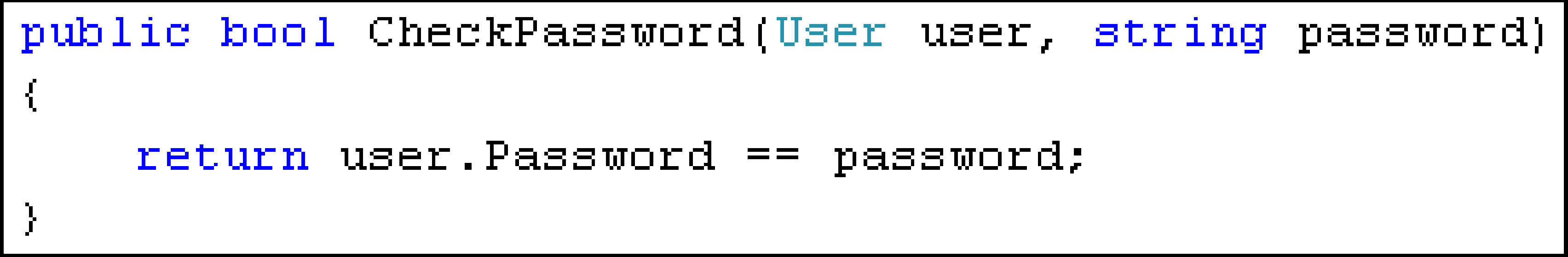
- It's MD5!
- Salt is the username and a constant string so susceptible to Rainbow Table attack

The password problem



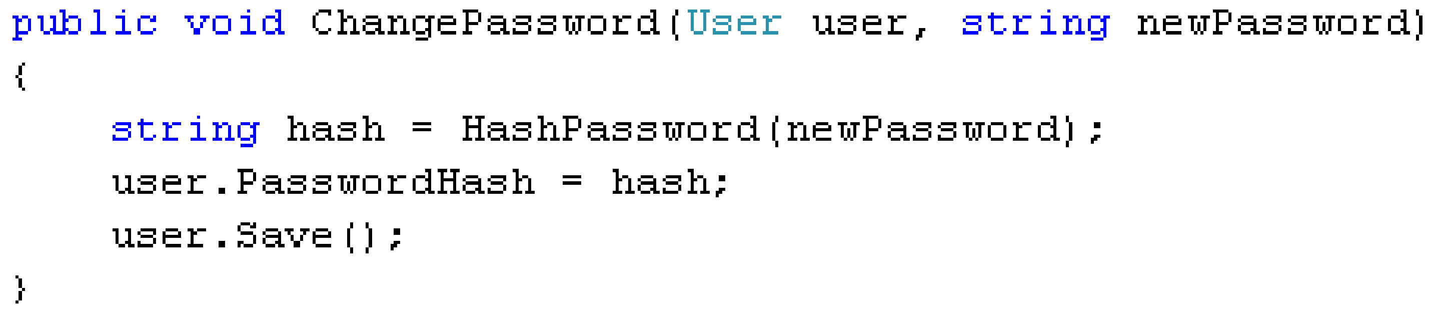
```
public void ChangePassword(User user, string newPassword)
{
    user.Password = newPassword;
    user.Save();
}
```

- Storing plain-text password in the DB
- Beware the ‘forgot password’ email with the password sent in plain-text



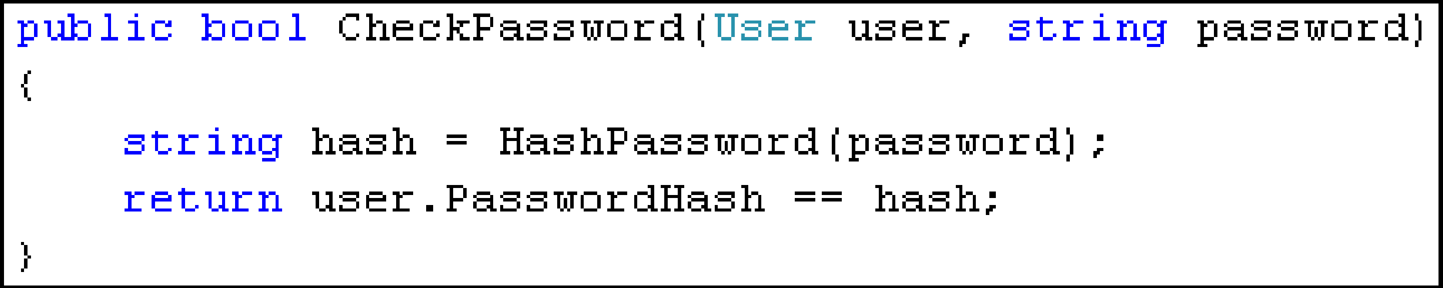
```
public bool CheckPassword(User user, string password)
{
    return user.Password == password;
}
```

Store and compare a hash



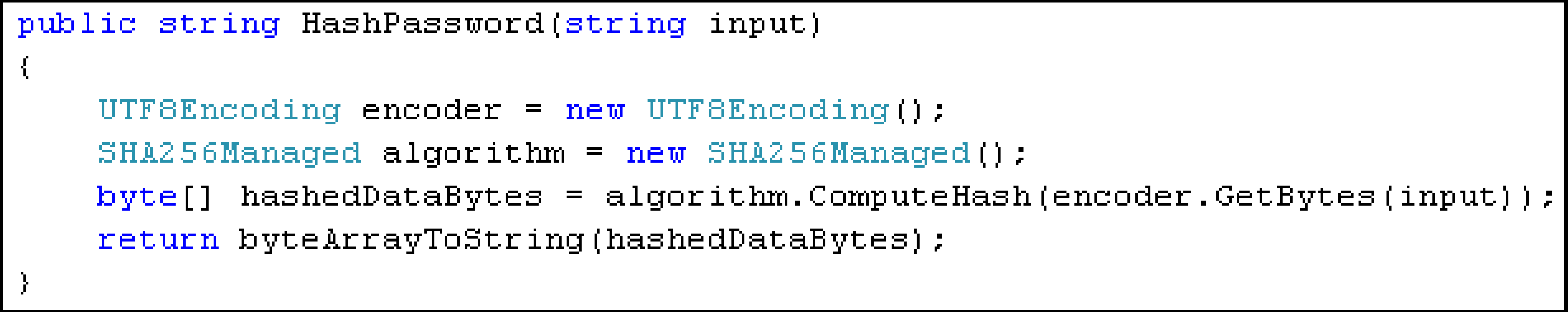
```
public void ChangePassword(User user, string newPassword)
{
    string hash = HashPassword(newPassword);
    user.PasswordHash = hash;
    user.Save();
}
```

- Storing a hash of the password in the DB
- Now we have to compare hashes, rather than plain-text



```
public bool CheckPassword(User user, string password)
{
    string hash = HashPassword(password);
    return user.PasswordHash == hash;
}
```

Computing a hash

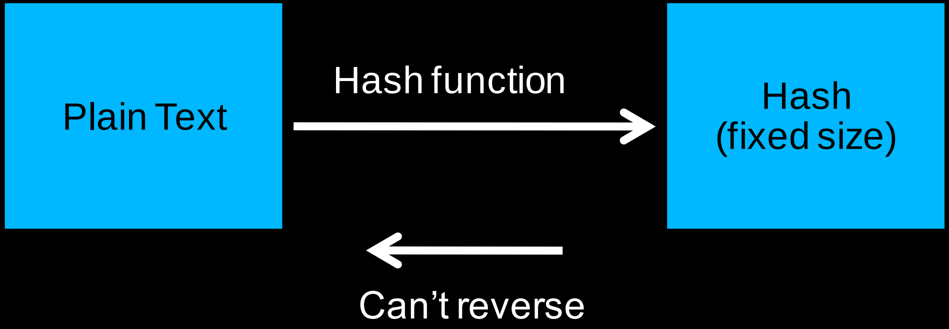


```
public string HashPassword(string input)
{
    UTF8Encoding encoder = new UTF8Encoding();
    SHA256Managed algorithm = new SHA256Managed();
    byte[] hashedDataBytes = algorithm.ComputeHash(encoder.GetBytes(input));
    return byteArrayToString(hashedDataBytes);
}
```

- Most libraries have decent hash functions
- A given input always gives the same output

Lookup Tables

Lookup a potential password given a hash


Plain Text

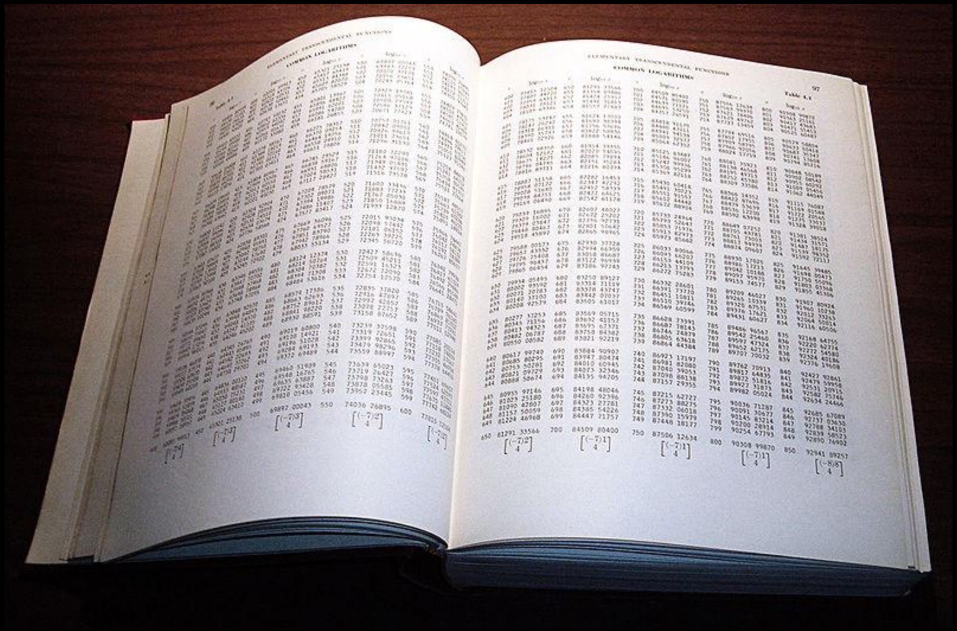
Hash function

Hash
(fixed size)

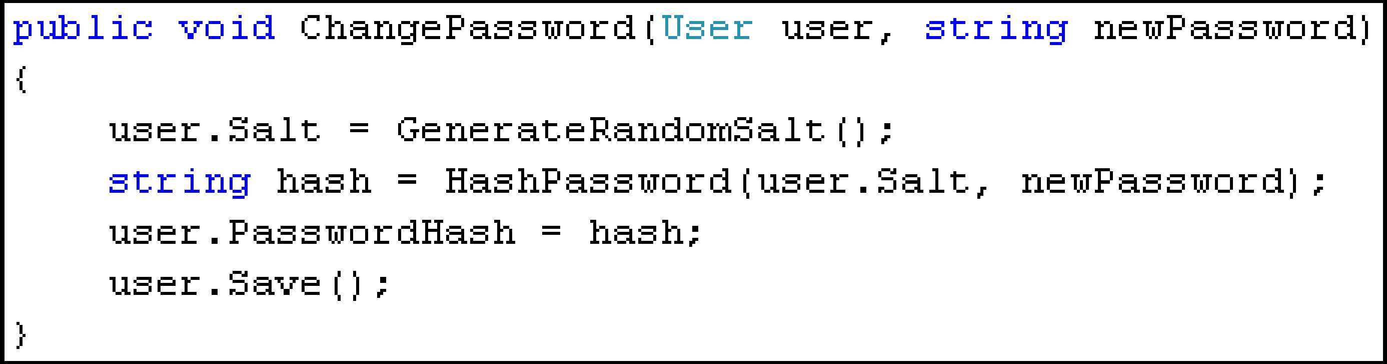
Can’t reverse

Rainbow table:

A faster algorithm for looking up

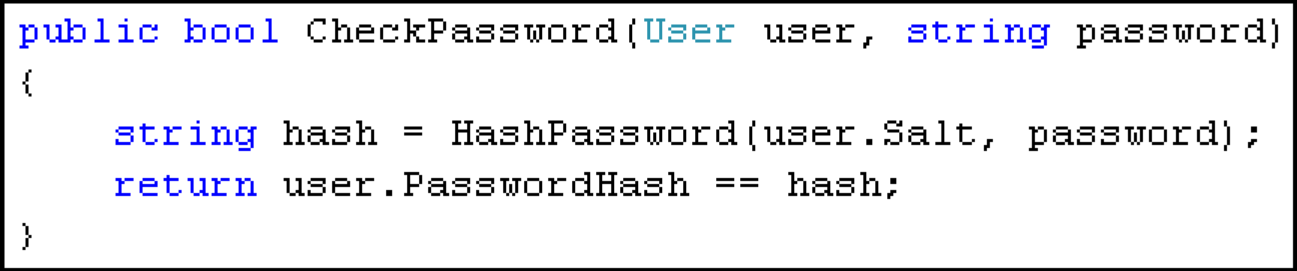


Use the salt



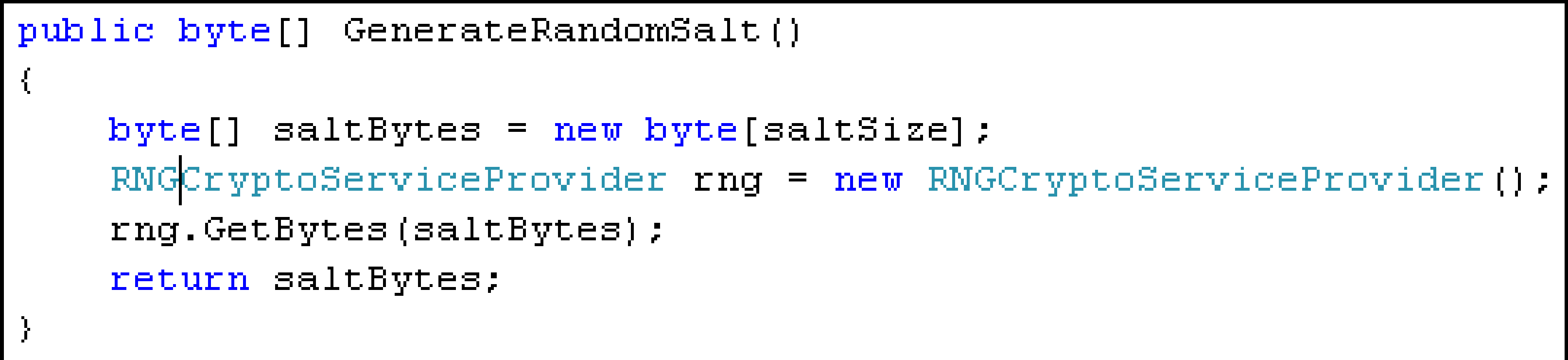
```
public void ChangePassword(User user, string newPassword)
{
    user.Salt = GenerateRandomSalt();
    string hash = HashPassword(user.Salt, newPassword);
    user.PasswordHash = hash;
    user.Save();
}
```

- Store a different random salt for each user
- Append the salt when hashing the password



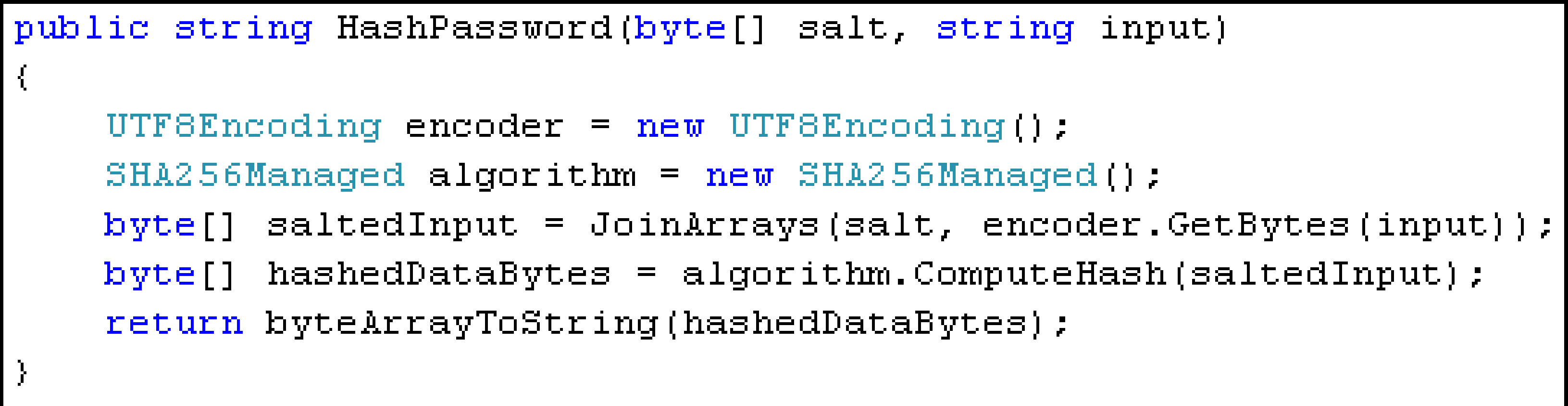
```
public bool CheckPassword(User user, string password)
{
    string hash = HashPassword(user.Salt, password);
    return user.PasswordHash == hash;
}
```

Generate a salt



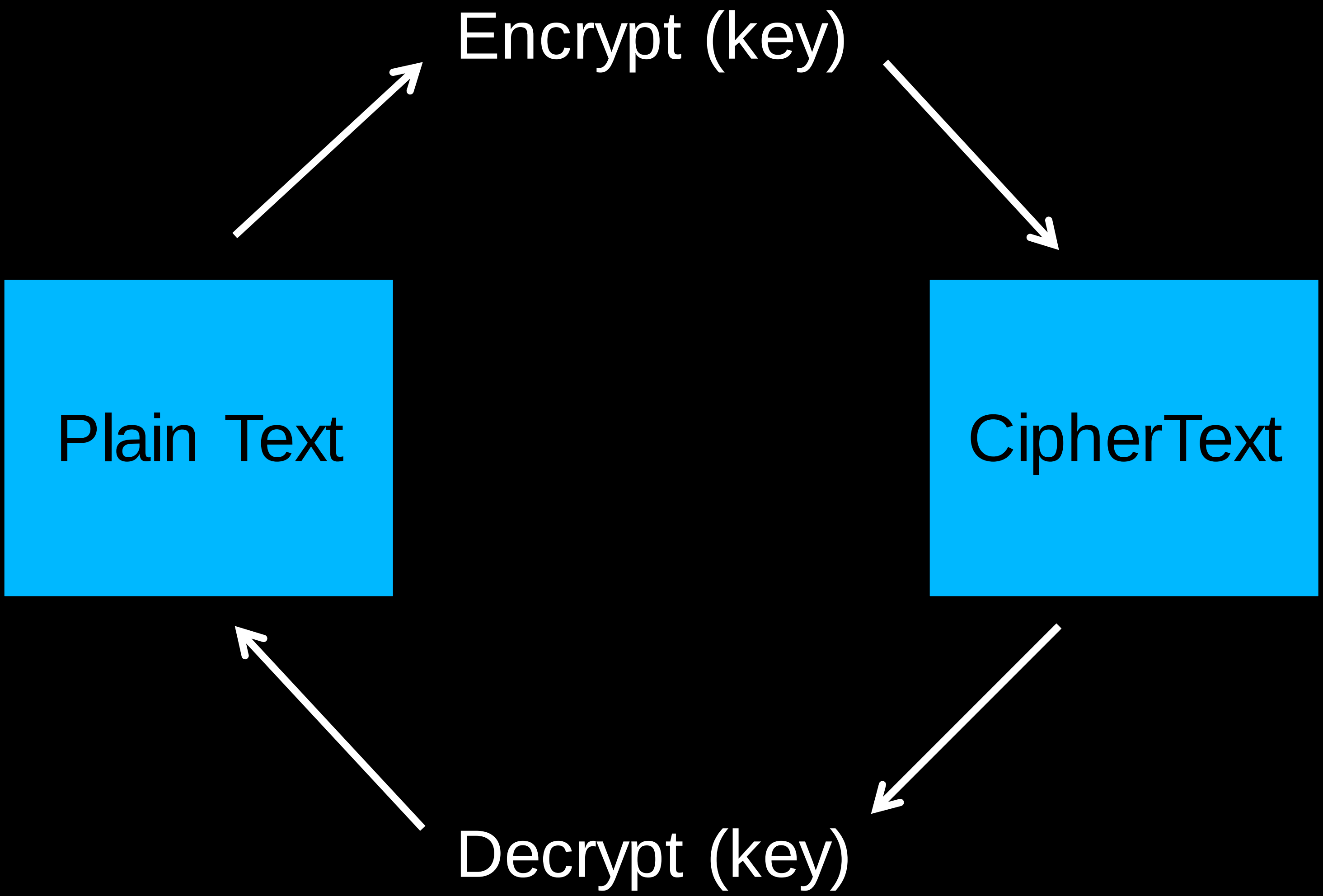
```
public byte[] GenerateRandomSalt()
{
    byte[] saltBytes = new byte[saltSize];
    RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider();
    rng.GetBytes(saltBytes);
    return saltBytes;
}
```

- Use a cryptographically secure pseudorandom number generator
  - System.Random isn’t



```
public string HashPassword(byte[] salt, string input)
{
    UTF8Encoding encoder = new UTF8Encoding();
    SHA256Managed algorithm = new SHA256Managed();
    byte[] saltedInput = JoinArrays(salt, encoder.GetBytes(input));
    byte[] hashedDataBytes = algorithm.ComputeHash(saltedInput);
    return byteArrayToString(hashedDataBytes);
}
```

Symmetric Encryption



Encrypt (key)

Plain Text

CipherText

Decrypt (key)

SYMMETRIC ENCRYPTION

Many encryption algorithms

Rijndael won the NIST competition to replace DES
Rijndael == AES
AES Finalists: Rijndael, Serpent, Twofish, RC6, MARS
Not recommended: DES, 3DES, IDEA, RC4

Birthday Attack:

- System of N elements
- Collision after square root of N
- Need 23 people to have a collision of birthdays
- 256 bit key provides 128 bit encryption strength

TALES FrOM THE CRYPT0
FORTIGATE HARD DISK ENCRYPTION

FORTIGATE DISK ENCRYPTION

Uses AES_ECB

Watermarking is visible on the disk – blocks of identical ciphertext

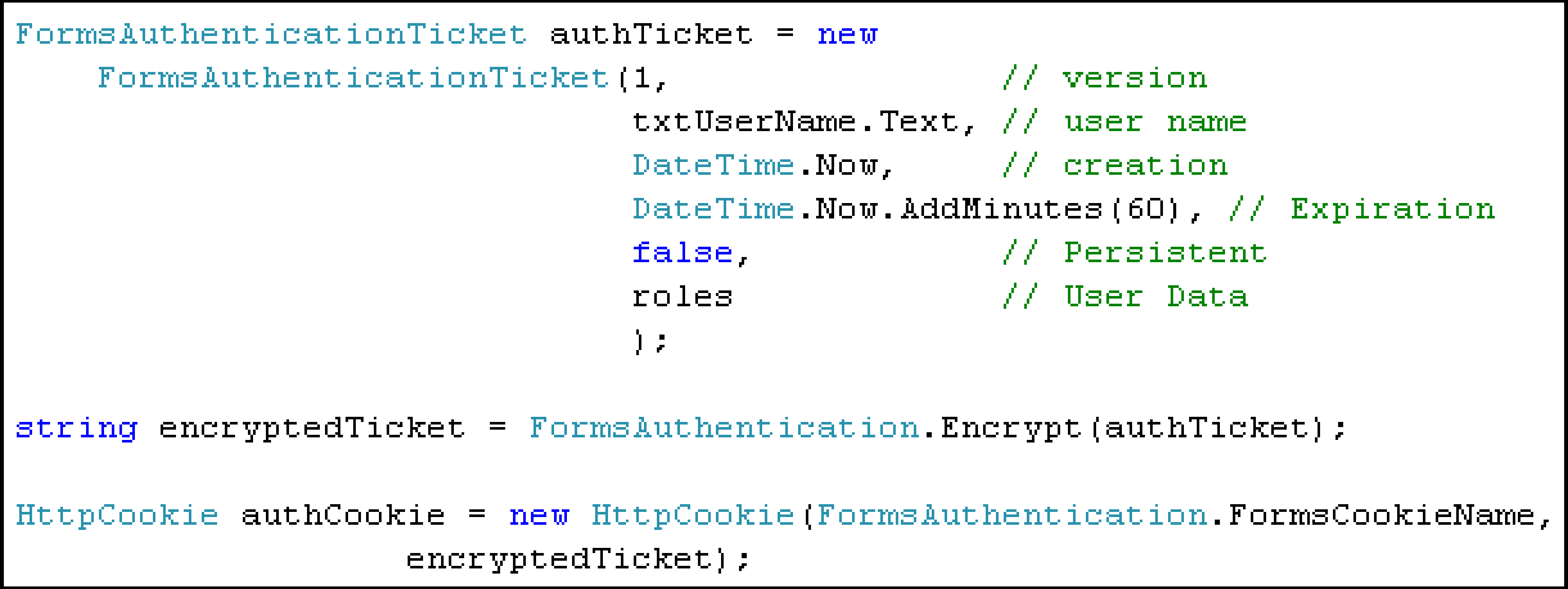
The same plain text encrypts to the same ciphertext

Disk contains known plain text

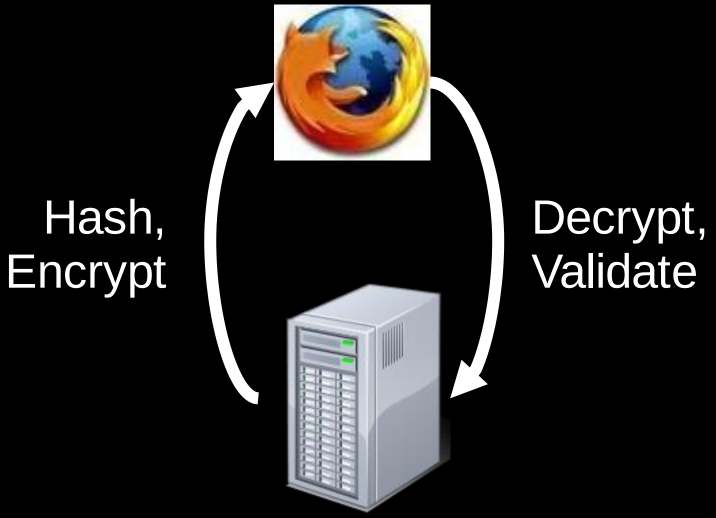
We can subvert the system to carry out chosen plaintext attacks
Write attacker specified plaintext to the disk

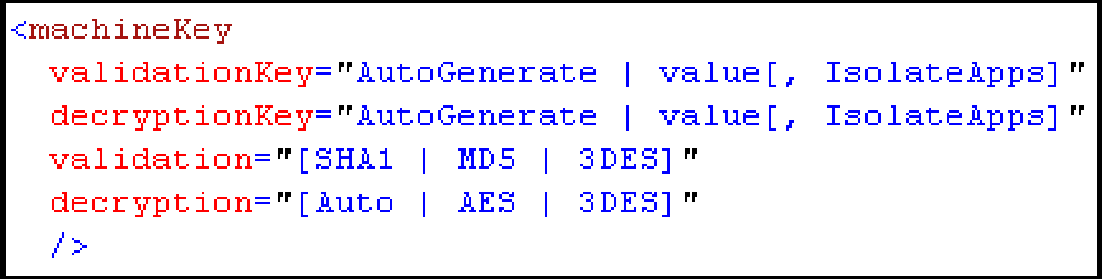
Symmetric key is on the system

Forms Auth Cookies



```
FormsAuthenticationTicket authTicket = new
    FormsAuthenticationTicket(1,                // version
                              txtUserName.Text, // user name
                              DateTime.Now,     // creation
                              DateTime.Now.AddMinutes(60), // Expiration
                              false,            // Persistent
                              roles             // User Data
                              );

string encryptedTicket = FormsAuthentication.Encrypt(authTicket);

HttpCookie authCookie = new HttpCookie(FormsAuthentication.FormsCookieName,
                   encryptedTicket);
```

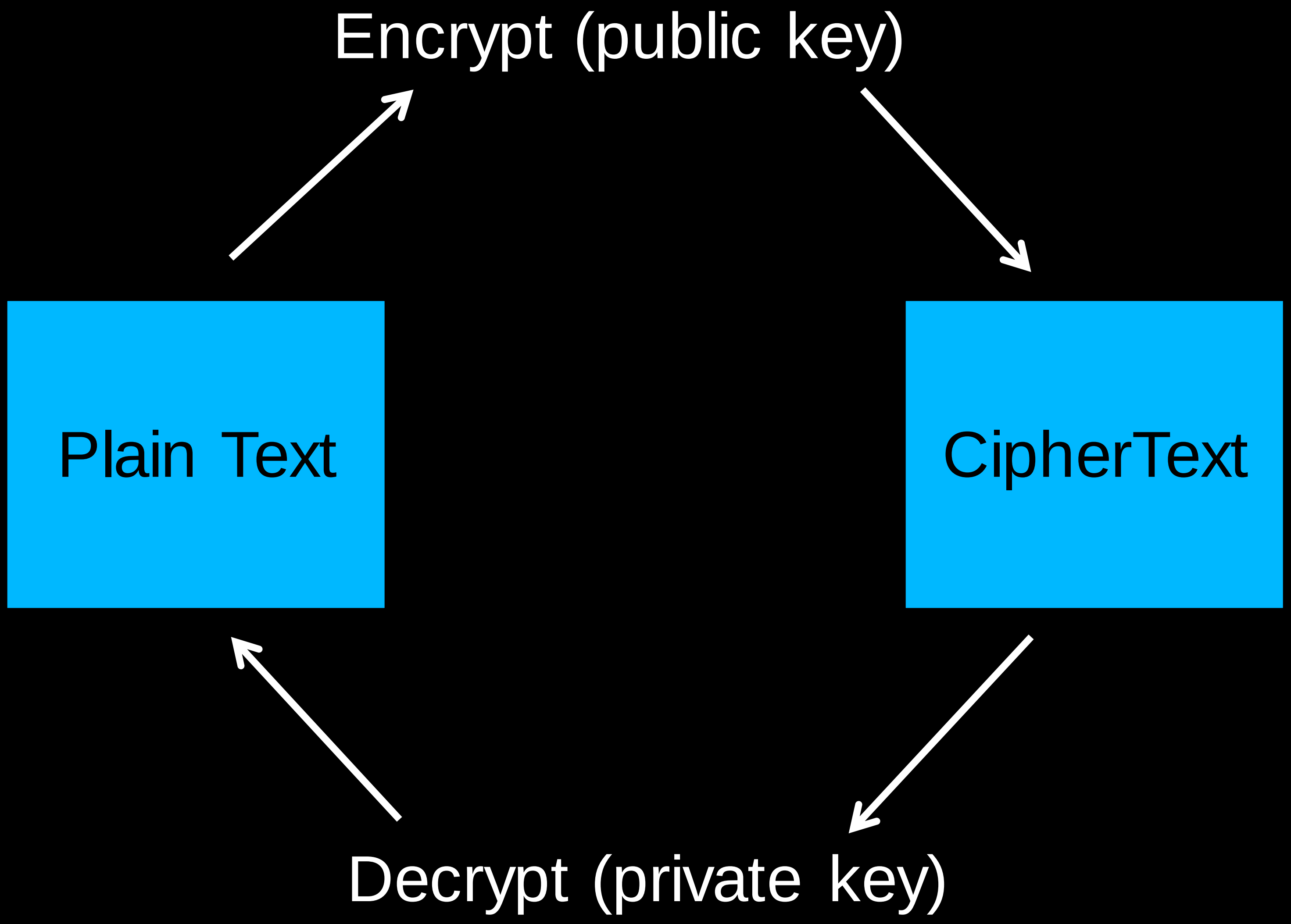




```
<machineKey
  validationKey="AutoGenerate | value[, IsolateApps]"
  decryptionKey="AutoGenerate | value[, IsolateApps]"
  validation="[SHA1 | MD5 | 3DES]"
  decryption="[Auto | AES | 3DES]"
  />
```

Keep the machine keys secure

Asymmetric Encryption



Encrypt (public key)

Plain Text

CipherText

Decrypt (private key)

ASYMMETRIC ENCRYPTION

Here comes a massive over simplification:
Exploits mathematical operations that are easy but whose inverse operations are hard^{TM}

For example:
Multiplying two primes is easy.
Finding the prime factors of a BIG integer is hard.

Certificate Revocation / PKI

- Brute force key search so choose an appropriate key size
- Side channel attacks
- Man in the middle
- Source of randomness must be random – really random

TALES FrOM THE CRYPT0

- Debian versus OpenSSL

DEBIAN VERSUS OPENSSL

All keys generated on Linux Debian based systems SEP 2006 – MAY 2008 affected

To fix a Vlgrind warning regarding uninitialised variables a maintainer of Debian patched OpenSSL and broke the random number generator.

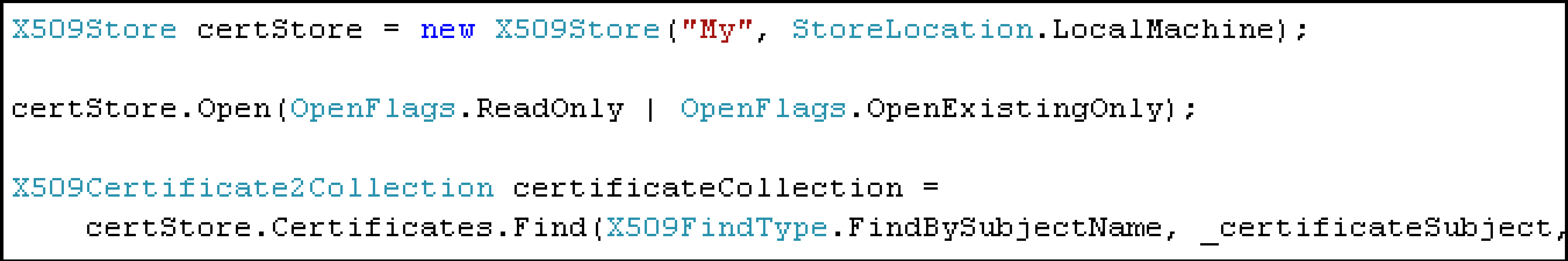
Only seed for the random number generator became the process ID: 1 - 32768

For each algorithm and key size only 32767 possible key values.

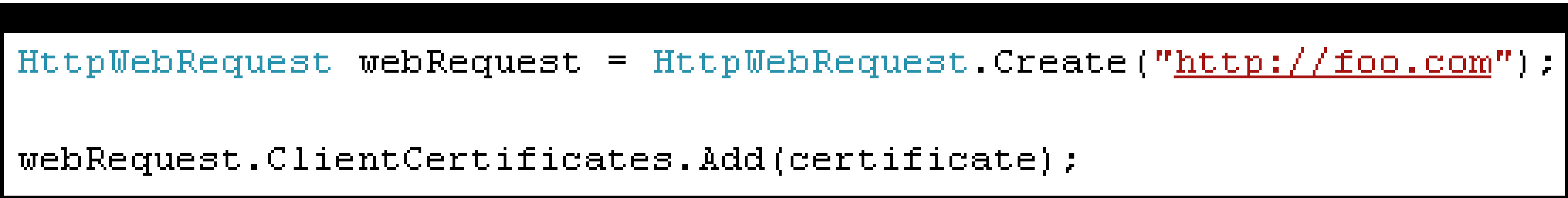
In practice:

- Keys generated at boot time will have a PID less than 500
- User generated keys PID between 500 and10,000
- Most keys will have a PID between1 and 3000

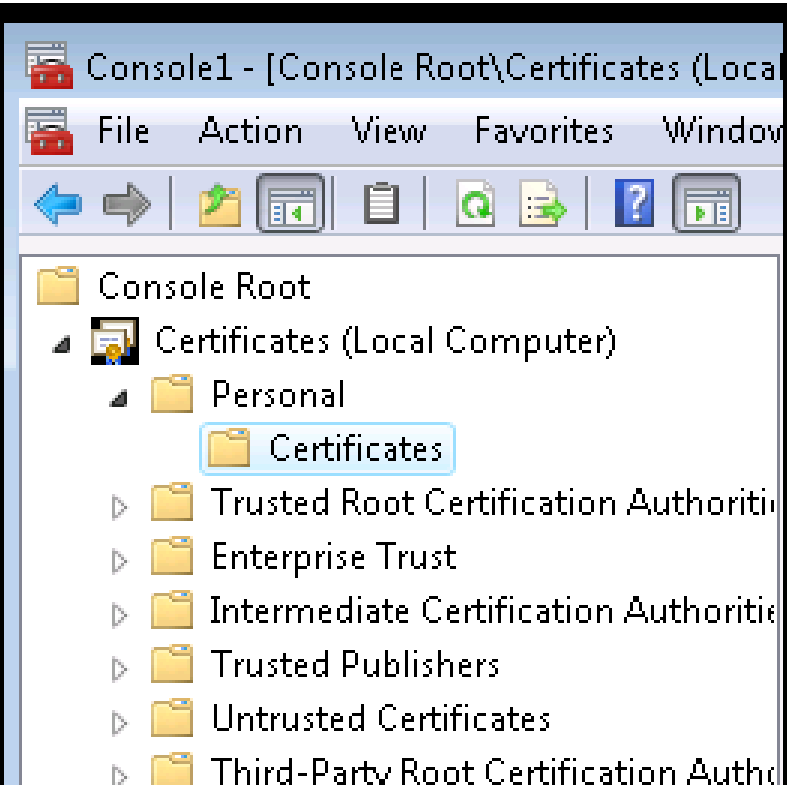
Asymmetric Example



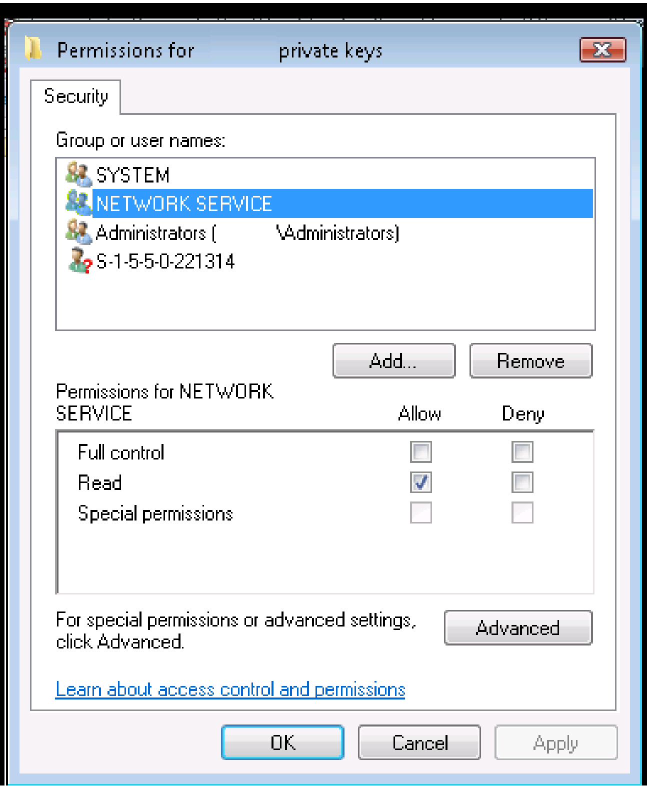
```
X509Store certStore = new X509Store("My", StoreLocation.LocalMachine);

certStore.Open(OpenFlags.ReadOnly | OpenFlags.OpenExistingOnly);

X509Certificate2Collection certificateCollection =
    certStore.Certificates.Find(X509FindType.FindBySubjectName, _certificateSubject,
```



```
HttpWebRequest webRequest = HttpWebRequest.Create("http://foo.com");

webRequest.ClientCertificates.Add(certificate);
```



- Console Root
  - Certificates (Local Computer)
    - Personal
      - Certificates
    - Trusted Root Certification Authoriti
    - Enterprise Trust
    - Intermediate Certification Authoritie
    - Trusted Publishers
    - Untrusted Certificates
    - Third-Party Root Certification Auth



Group or user names:

- SYSTEM
- NETWORK SERVICE
- Administrators (\Administrators)
- S-1-5-5-0-221314

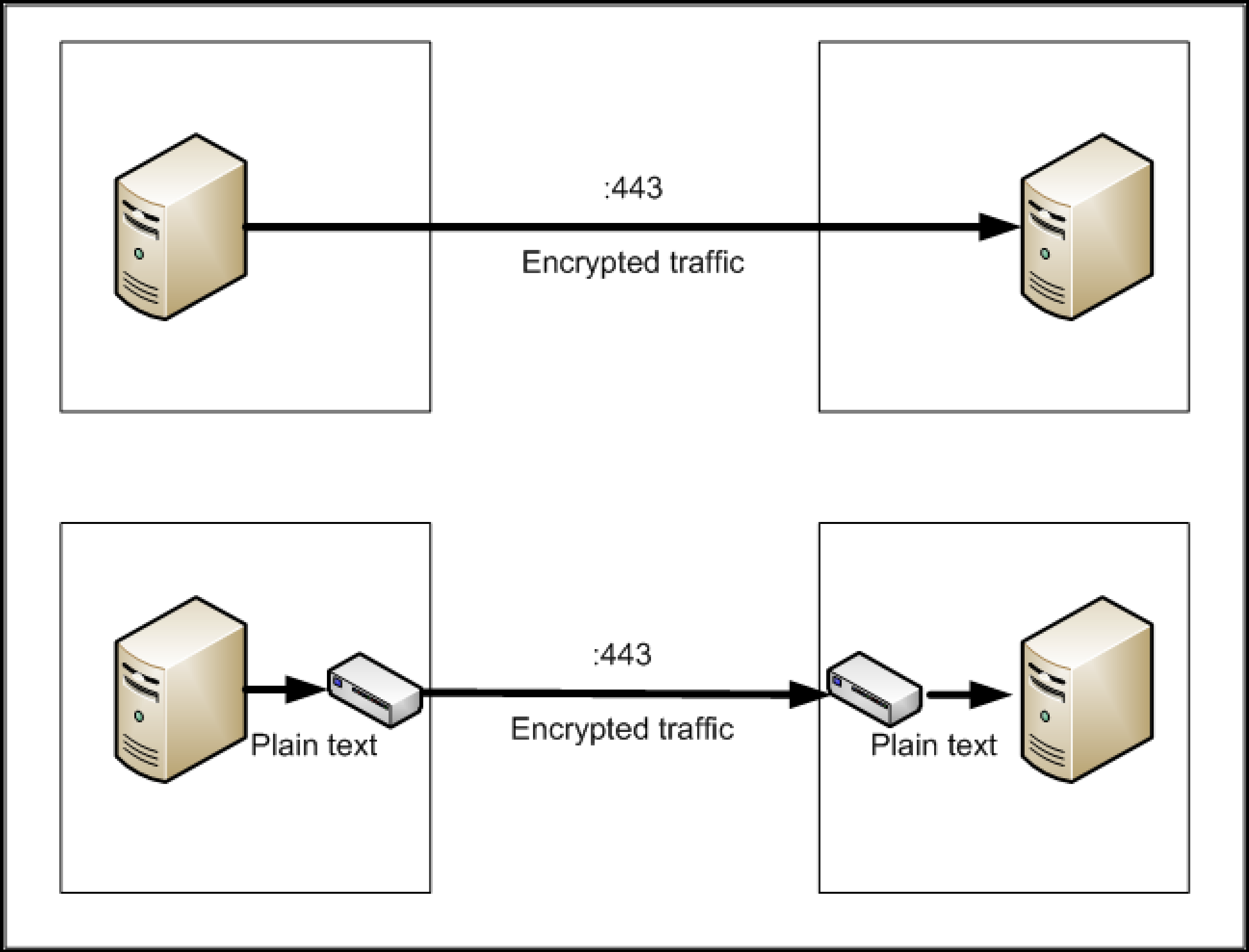
Permissions for NETWORK SERVICE

	Allow	Deny
Full control	□	□
Read	☑	□
Special permissions	□	□

For special permissions or advanced settings, click Advanced.

Learn about access control and permissions

SSL Proxy


:443
Encrypted traffic
:443
Plain text
Encrypted traffic
Plain text

SECURE SOCKETS LAYER

Hashing for Message Authenticatior

Symmetric encryption for confiden

Asymmetric encryption for

- authentication
- symmetric encryption key exchange.

Cypher suite: protocol, authentication, encryption, message authentication code

e.g. TLS_RSA_WITH_AES_128_CBC_SHA256

Certificates

- Use strong certificates (key size of 2048 bit from 2010)
- Protect the private keys
- Get certificates for all used domain names
  - Conditioning users to certificate errors is not acceptable
  - Wildcard (*.example.com) or enhanced wildcard (a.example.com, b.example.com) are options

Use Strong SSL Versions

History:

- SSL v1, v2 (1995), v3 (1996)
- TLS v1.0 (1999), v1.1 (2006), v1.2 (2008)

Best practice: use TLS v1.0 and above

- IE7, Firefox 2.0 and newer (IE6 via patch)

Never use SSL v2. Use SSL v3 under duress.

Only Support Strong Cryptographic Cyphers

- Use AES for encryption
- Use CBC mode
- Use SHA for digest
- MD5 may be used within the TLS protocol
- Do not provide support for NULL ciphersuites

SSL Cipher Check, How to disable

SSL / TLS Cheat Sheet

- Use TLS for All Login Pages and All Authenticated Pages
- Use TLS on Any Networks (External and Internal) Transmitting Sensitive Data
- Do Not Provide Non-TLS Pages for Secure Content
- Do Not Mix TLS and Non-TLS Content
- Use "Secure" Cookie Flag
- Keep Sensitive Data Out of the URL
- Do Not Perform Redirects from Non-TLS Page to TLS Login Page

http://www.owasp.org/index.php?title=Transport_Layer_Protection_Cheat_Sheet

DISABLING SSLV2

Apache

```
SSLProtocol -ALL +SSLv3 +TLSv1
SSLCipherSuite ALL:!aNULL:!ADH:!eNULL:!LOW:!EXP:RC4+RSA:+HIGH:+MEDIUM
```

IIS

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\PCT 1.0\Server]
"Enabled"=dword:00000000
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\SSL 2.0\Server]
"Enabled"=dword:00000000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\DES 56/56]
"Enabled"=dword:00000000
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\NULL]
"Enabled"=dword:00000000
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\RC2 40/128]
"Enabled"=dword:00000000
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\RC2 56/128]
"Enabled"=dword:00000000
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\RC4 40/128]
"Enabled"=dword:00000000
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\RC4 56/128]
"Enabled"=dword:00000000
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\RC4 64/128]
"Enabled"=dword:0000000
```

CRYPT0KEEPER SAYS

- Use cryptography
- Remember the Hoff’s
- Hash with SHA
- Encrypt with AES
- Protect your keys
- Randomness is vital