**INTRODUCTION TO COMPUTER SECURITY(CSCE 5550)**

**PROJECT REPORT**

**12/6/2016**

(Harish Tirunagari , Abhishek Mutyampeta)

UNIVERSITY OF NORTH TEXAS

1. **Message Digest**

**MD5** : The MD5 algorithm is a widely used [hash function](https://en.wikipedia.org/wiki/Hash_function) producing a 128-[bit](https://en.wikipedia.org/wiki/Bit) hash value. Although MD5 was initially designed to be used as a [cryptographic hash function](https://en.wikipedia.org/wiki/Cryptographic_hash_function), it has been found to suffer from extensive vulnerabilities. It can still be used as a [checksum](https://en.wikipedia.org/wiki/Checksum) to verify [data integrity](https://en.wikipedia.org/wiki/Data_integrity), but only against unintentional corruption.Like most hash functions, MD5 is neither encryption nor encoding.

**SHA :**

**Message Digest:** This MessageDigest class provides applications, the functionality of a message digest algorithm, such as SHA-1 or SHA-256. Message digests are secure one-way hash functions that take arbitrary-sized data and output a fixed-length hash value.A MessageDigest object starts out initialized. The data is processed through it using the [update](https://docs.oracle.com/javase/7/docs/api/java/security/MessageDigest.html#update(byte)) methods. At any point, [reset](https://docs.oracle.com/javase/7/docs/api/java/security/MessageDigest.html#reset()) can be called to reset the digest. Once all the data to be updated has been updated, one of the [digest](https://docs.oracle.com/javase/7/docs/api/java/security/MessageDigest.html#digest()) methods should be called to complete the hash computation.The digest method can be called once for a given number of updates. After digest has been called, the MessageDigest object is reset to its initialized state.

**2. Various Crypto Techniques**

**Keygenerator class :** This class provides the functionality of a secret (symmetric) key generator Key generators are constructed using one of the getInstance class methods of this class. KeyGenerator objects are reusable, i.e., after a key has been generated, the same KeyGenerator object can be re-used to generate further keys.

**Key pair generator:** The KeyPairGenerator class is used to generate pairs of public and private keys. Key pair generators are constructed using the getInstance factory methods (static methods that return instances of a given class).A Key pair generator for a particular algorithm creates a public/private key pair that can be used with this algorithm. It also associates algorithm-specific parameters with each of the generated keys.

**Public and Private keys :** Public key cryptography, or asymmetric cryptography, is any cryptographic system that uses pairs of [keys](https://en.wikipedia.org/wiki/Cryptographic_key): public keys which may be disseminated widely, and private keys which are known only to the owner. This accomplishes two functions: [authentication](https://en.wikipedia.org/wiki/Authentication_protocol), which is when the public key is used to verify that a holder of the paired private key sent the message, and [encryption](https://en.wikipedia.org/wiki/Encryption), whereby only the holder of the paired private key can decrypt the message encrypted with the public key.

**Secure Random :** This class provides a cryptographically strong random number generator (RNG).A cryptographically strong random number minimally complies with the statistical random number generator tests specified in [FIPS 140-2, Security Requirements for Cryptographic Modules](http://csrc.nist.gov/cryptval/140-2.htm), section 4.9.1. Additionally, SecureRandom must produce non-deterministic output. Therefore any seed material passed to a SecureRandom object must be unpredictable, and all SecureRandom output sequences must be cryptographically strong, as described in [RFC 1750: Randomness Recommendations for Security](http://www.ietf.org/rfc/rfc1750.txt).

**Cipheroutputstream :** A CipherOutputStream is composed of an OutputStream and a Cipher so that write() methods first process the data before writing them out to the underlying OutputStream. The cipher must be fully initialized before being used by a CipherOutputStream.

For example, if the cipher is initialized for encryption, the CipherOutputStream will attempt to encrypt data before writing out the encrypted data.

1. **Authentication:**

In the following, we implemented double strength password authentication using message digest. In version -1 of creating message digest we create digest by taking following inputs : user,password,timestamp1,timestamp2.random1,random2.

Then we create double strength by creating a digest again on the above creating digest.

1. **Signature:** This is implemented by using ELGAMAL Signature scheme , which is very similar to Diffie hellman scheme. So the program is implemented in the following way.

We have first Choose a prime *p* and two random numbers *g* and *d* both less than *p* and then computed *y* = *gd* mod *p*. The public key is the triplet (*y*, *g*, *p*); the private key is *d*.

Suppose Alice wants to send Bob a signed contract *m*. She chooses a number *k* that is relatively prime to *p – 1* and has not been used before. She computes *a* = *mk* mod *p* and then uses the Extended Euclidean Algorithm to find *b* such that

*m* = (*da* + *kb*) mod *p* – 1

The pair (*a*, *b*) is the signature.

To verify the signature, check that *a* = *mk* mod *p*

**D) Public-Key System:**

RSA is one of the first practical [public-key cryptosystems](https://en.wikipedia.org/wiki/Public-key_cryptography) and is widely used for secure data transmission. In such a [cryptosystem](https://en.wikipedia.org/wiki/Cryptosystem), the [encryption key](https://en.wikipedia.org/wiki/Encryption_key) is public and differs from the [decryption key](https://en.wikipedia.org/wiki/Decryption_key) which is kept secret. In RSA, this asymmetry is based on the practical difficulty of [factoring](https://en.wikipedia.org/wiki/Factorization) the product of two large [prime numbers](https://en.wikipedia.org/wiki/Prime_number), the [factoring problem](https://en.wikipedia.org/wiki/Factoring_problem). RSA is made of the initial letters of the surnames of [Ron Rivest](https://en.wikipedia.org/wiki/Ron_Rivest), [Adi Shamir](https://en.wikipedia.org/wiki/Adi_Shamir), and [Leonard Adleman](https://en.wikipedia.org/wiki/Leonard_Adleman), who first publicly described the algorithm in 1977.

Choose two large prime numbers p, q

Let n = pq; then f(n) = (p–1)(q–1)

Choose e < n relatively prime to f(n).

Compute d such that ed mod f(n) = 1

Public key: (e, n);

private key: d (or (d, n))

Encipher: c = *me* mod *n*

Decipher: m = c*d* mod *p*

**E) X.509 certificates**

An X.509 certificate contains information about the identity of which a certificate is issued and the identity that issued it. Standard information in an X.509 certificate includes:

* Version – which X.509 version applies to the certificate (which indicates what data the certificate must include)
* Serial number – the identity creating the certificate must assign it a serial number that distinguishes it from other certificates
* Algorithm information – the algorithm used by the issuer to sign the certificate
* Issuer distinguished name – the name of the entity issuing the certificate (usually a [certificate authority](http://searchsecurity.techtarget.com/definition/certificate-authority))
* Validity period of the certificate – start/end date and time
* Subject distinguished name – the name of the identity the certificate is issued to
* Subject public key information – the public key associated with the identity

**Q) What are the limitations of using self-signed certificates? What are they useful for?**

1. The certificates aren’t trusted by other applications/operating systems. This may lead to authentications errors etc.

Note: To overcome this limitation, some IT staff add the self-signed certificates to the Trusted Roots Certificate Authorities. However, using this workaround may to additional time that needed for management and troubleshooting.

2. Self-signed certificates life time is usually 1 years. Before the year is ended, the certificate may need to renew/replace.

3. Self-signed certificates may use low hash and cipher technologies. Due to this, the security level that implemented by self-signed certificates may not satisfy the current Security Policy etc.

4. No support for advanced PKI (Public Key Infrastructure) functions (e.g. Online checking of the revocation list etc.).

5. Most of the advanced feathers of the server side applications required to impended a PKI (Public Key Infrastructure). By this, self-signed certificates advantages cant be used.

what are they used for

1. No PKI (Public Key Infrastructure) is needed.

2. Automatic deployment (Usually Self-signed certificates created automatic during the installation process of the server-side applications).

self-signed certificates are free of charge