Bogenberger - Damsell, O., Cleary E., Crowley, T., Kocik, D., McMorrow, R. (2020) Report on Cryptography, University of Limerick, unpublished

**Abstract:**

**Introduction**

**Early History**

Caesar Cypher first actual use of cryptography to hide messages.

"Caesar used it in order to   
if there was occasion for secrecy, he wrote in cyphers; that is, he used the alphabet in such a manner, that not a single word could be made out. The way to decipher those epistles was to substitute the fourth for the first letter, as d for a, and so for the other letters respectively." The Twelve Caesars 56.Gaius Suetonius Tranquillus. Translator:Alexander Thomson

Due to the Power and enemies he had aquired in Rome Caesar needed a good way to keep his messages secret.The Caesar is of course a simple form of cryptography but as it was close to the first use of cryptography it didnt need to be too complex, and the caesar cypher was also suprisingly used in modern computing in the ROT13 method as a low security way of hiding information. this method was used more to protect the viewer from potentially offensive language or information.Another interesting adaption of the Caesar Cypehr in history was the Vigenère cipher.

Vignere cipher first use of encryption key. Vignere Cypher was a Cypher first described in the book La cifra del, by Giovan Battista Bellaso in 1553 and then later misattributed to Blaise de Vigenère. The cypher was known to be extremely secure in the era of pen and paper cryptography and so earned the name "le chiffre indéchiffrable".

The Cypher worked using a square grid of alphabets, as shown here.

A close up of a newspaper

Description automatically generated

and a keyword as the encryption key. The encoder would write their message in plain text, and then repeat the keyword inline underneath until the lengths were equal.

|  |
| --- |
| thisisademonstration |
| pizzapizzapizzapizza |

The encoder would then check on the grid where the letter in the plaintext and in the keyword overlapped in the grid, and write down that letter in the encoded message. For example letter b in plaintext and letter p in the keyword would make q in the encoded message.

And the example shown above becomes.

IPHRIHICDMDVRSRPBHNN

**Cryptography and early computing**

Hebern rotor machine.

Enigma.and Bombe machine.

**Early modern history**

Ibm crypto group and Lucifer(D.E.S. Data Encryption Standard)

1997 NIST encryption (Advanced Encryption standard).

**Cryptography and modern computing**

//also discuss key terms, mention cryptography and encryption were synonymous

//Terms to mention: key(randomness),

**Symmetric cryptography**

## Introduction

The formal definition of symmetric encryption is “an algorithmic tool that allows a pair of parties to communicate secret information over open communication media that are accessible to eavesdroppers.” (Theory of Cryptography Conference Corporate 2010) This is a classic model of encryption, where the both parties share a secret key. The key is assumed to be random, single-purpose and not dependant on the message. The security is ensured by the fact that encryption and decryption happen in safe environments and the adversary cannot intercept the key.

According to (Buchanan 2017, p.55), there are two main types of symmetric (a.k.a. “secret key”) encryption: stream cipher and block cipher. Buchanan also mentions that symmetric key decryption is faster than the asymmetric one with two keys, and so it is more suitable to use where data has to be transmitted in real-time e.g. secure voice communication online.

## Stream cipher

Stream encryption works on the basis of operating on a continuous data stream, where “the message is broken into successive bits or characters and then the string of characters is encrypted using a key stream” (Nandi et al. 1994). In order to create a ciphertext, a pseudorandom, pseudo-infinite key must be generated in binary form. This is generated from initialisation vector, which acts as a random seed. This key is then XOR-ed with the plaintext (here the data stream) in binary form. (Buchanan 2017, p.58)

According to (Paar 2010) traditionally, modulo-2 addition performed by the XOR gates was used for stream encryption, since the key used by this form of encryption is random. This is because after XOR operation the resulting bit has 50% chances of being 1 and 50% chances of being 0, unlike if we were to use other gates (e.g. AND gate).

## RC4

A popular example of a stream cipher encryption algorithm is RC4. It is used by SSL (Secure Socket Layer) and WEP (Wireless Encryption Protocol). RC4 is made up of key scheduling (KSA) and pseudo-random number generation (PRGA) algorithms (Isobe et al. 2014, pp.1-2). The first algorithm creates an initial permutation of S, which is an array of 256 bytes. Element i of S is swapped with element j of S, where j is a sum of its previous value, value of S[i] and the ith byte of the key. Since our key is shorter than 256 bytes, we use modulo operation and traverse through the same key again. After this is completed, we run the PRGA which traverses through the vector S and alters it again, as i is incremented by 1 and j is increased by the ith byte of S. The ith and jth bytes are then swapped and the remainder of their sum divided by 256 is the index of S that points to the byte that will be output in the keystream. As we keep getting the mod 256 of i and j as those values increase, we get pseudo-infinite key. The key is then X-ORed with our plaintext data bit-by-bit, resulting in a ciphertext.

This method, however, although very fast, is no longer considered safe. RC4 is subject to “plaintext recovery attacks” that work for the “initial bytes of the keystream”, which are actively used by SSL/TLS (Isobe et al. 2014, p.15).

**Block cipher**

Block ciphers work on the basis of splitting the data into multiple blocks. Although most blocks are fixed size, the last block could be shorter, since data does not have to be a multiple of block size. For this reason, block encryption requires padding, which would fill the space needed for the block to be encrypted, send and decrypted correctly.

Popular ways of padding include adding NULL characters, or 0x80 and then NULL characters (Bits), or “the same value as the number of the padding bytes” (CMS standard), filling with zeros until the last 8 bits, which are set to length of the added bytes (“ZeroLength”) (Buchanan 2017, p.59).

There are many block ciphers in use today, such as Blowfish, AES and RC5.

**AES (Advanced Encryption Standard)**

AES, a.k.a. Rijndael, is a popular implementation of the block cipher. The block size is 128 bits / 16 bytes and the key used for encryption can have 128, 192 or 256 bits (A.E.S. Corporate 2005). The easiest way to represent AES block would be to by splitting the block into bytes and forming a square matrix of length 4. The operations performed on a block consist of 10-14 rounds (depending on key length). Each round performs operations described as “AddRoundKey”, “SubBytes”, “ShiftRows” and “MixColumns”, while the columns are not mixed during the final round (A.E.S. Corporate 2005, p.2).

There are two types of AES: Galois/Counter Mode (GCM) that XORs each block with the next, which adds on the mechanics from the stream cipher, and Cipher Block Chaining, where the initialisation vector is used encryption of the first block and the vector has to be sent to the receiver, who can then use it to decrypt that block, while all the consecutive blocks are XORed with the previous block to decrypt the sequence (Buchanan 2017, pp.65-66).

**Common attacks**

**Public key cryptography**

**Introduction**

(Buchanan, William J. Cryptography, pg 144/ 155)

Public key encryption or asymmetric key encryption uses both a public and private key to secure a communication between two entities. The public key is shared and distributed while the private key is kept private and secure. The mathematics of the encryption make it really difficult to determine one key when given the other key because of how difficult it is to factor a value for its prime number factors, it is so difficult that it should take conventional computers thousands of years of computing time to decrypt. This is achieved using the following methods. **Integer factorization** (RSA method) **discrete logarithms** (ElGamal)and **elliptic curve relationships** (Elliptic Curve).

Public key encryption is often used for identity checking. It checks the identity of the entity by using its public key to decrypt a message that was encrypted using the entities private key. Since the two keys are mathematically linked then they can only be decrypted using the other so if it can be decrypted by the public key then it proves its identity.

It can also be used for protection of a symmetric key. This tends to be used in disc encryption where the symmetric key that is used to encrypt a file is protected with the public key of an entity so then the only key with access to the symmetric key is the private key.

Of course it’s most common use is to establish secure communications. When two entities want to communicate securely they simply exchange their public keys and use them to encrypt the data they want to send to the other. That way the only entity that can decrypt the message is its intended recipient who holds the private key.

The key is usually stored in an XML format or on a digital certificate which allows it to be stored processed and transmitted this is known as the (PKI) or the Public Key Infrastructure. Where key pairs are generated by trusted entities the private key is kept secret while the public key is distributed using the PKI. The most important part of the PKI is the keeping of the private key secret. If the private key loses its secrecy then the security and identity of the entity could be breached as well as the any encryption keys that are protected by the key pair.

## RSA

The public and private keys are generated from very large prime numbers as a value which is the product of another two large prime numbers that is extremely to factorize. The public key is then passed where it is used to encrypt data intended to be sent to the entity.

The most used and well known algorithm in RSA selects two large prime numbers that are usually around 256 bits in length *a* and *b* and are also to the order of 10100, these factors are then kept secret and you make the modulus N by multiplying them together. Then to make the second public key you choose another value *e* so that *e* and (*a* – 1) × (*b* - 1) are relatively prime as in they do not share a common factor bigger than 1 so GCD (*x, y*) = 1. The public key is then <*d, N*> and this makes a key that is at least 512 bits long.

The private key for decryption *d* is computed so that:

*d = e-1 mod [(a – 1) × (b - 1)]*

*Or*

*(d × e) mod [(a-1)(q-1)] = 1*

PHI is defined as *(p -1) × (b-1).* The encryption process to the *c* is *= me mod N,* and the message *m* is then decrypted with the formula *m = cd mod N.*

**EIGamal**

The keys are generated using a description of a cyclic group *G* of order *q* with the generator *g*. *e* is represents the unit element of *G*. A random integer *x* is then chosen from *{1,…, q - 1} h := gx.* The public key is made from the values *(G, q, g, h)* and *x* is kept as the private key.

The message is encrypted by mapping it to an element *m* of *G* with a reversible mapping function. You choose another random integer *y* from *{1, …, q -1}*. Compute the shared secret *s := hy* as well as *c1 := gy* and *c2 := hy c2 :m × s* and send the cipher text *(c1, c2)* to the intended recipient of the message.

*(c1, c2)* is decrypted with the private key *x* by computing *s := c1x* because *c1 := gy,* *c1x* *= gxy* = *hy* which is the same shared secret used. *s-1*, the inverse of *s* is then computed in the group *G*. You then compute *s-1* as *c1q-x* because of Laranges theorem which states *s × c = gxy × g(q - x)y = (gq)y = ey = e*. *m := c2 × s-1* is then computed producing the original message because

*c2 = m×s* so *c2 × s-1 = (m×s) s-1* *= m×e = m*

the result *m* is then mapped onto the message

**Common attacks**

**Hash functions**

**Introduction**

A hash function is a function that can be used to map data of any size to a number of fixed size. The values returned by a hash function are called hash values, hash codes, digests, or hashes. The values are used to index a fixed-size table called a hash table. Using hash functions to index a hash table is called hashing.

Hashes do not need to be able to be decrypted. If 2 pieces of data have the same hash then it is (probably) the same file.

A hash function takes an input as a key, which is associated with a record and used to map it to the data storage and retrieval application. The keys may be fixed length, like an integer, or variable length, like a name. In some cases, the key is the data itself. The output is a hash code used to index a hash table holding the data or records, or pointers to them.

A hash function has three requirements:

it has to be fast but it cant be too fast or else it becomes too easy to crack.

It has to have an avalanche effect. Changing one bit ANYWHERE in the file should change the entire Hash.

Have to be able to avoid collisions. Think of an analogy with cables. If there is a collision with Hashes then its like trying to plug 2 cables into the 1 socket.

A Hash function is considered broken if its possible to create collisions deliberately. MD5 was previously the most widely used but now it is considered broken for this exact reason. People found out how to deliberately create collisions.

The problem with this is that if you are able to intercept a file and edit it and have the hash stay the exact same then you can put something malicious into the file and cause problems.

MD5 is so broken now that you can figure out what its hash’s represent by typing them into google. People used to store passwords this way and if anyone were to use MD5 for a password nowadays someone could crack it by just looking it up on google.

**SHA1,2,3,**

People then began to move to SHA-1 which was created by the NSA. However now its thought that this might start to become broken as well as computers get faster and faster.

Some moved to SHA-2, which for the time being is secure.

SHA-3 is being checked by agencies. In a few years this will become the standard.

None of these should ever be used for storing passwords. They can become broken too easily. They should only be used for file transfer.

## Hash Tables

Hash functions are used with Hash table to store and retrieve data items or data records. The hash function translates the key associated with each piece of data into a hash code which is used to index the hash table. When an item is added to the table, the hash code might index an empty slot (bucket), in which case the item is added to the table there. If the hash code indexes a full slot, some kind of collision resolution is required:

the new item may:

not be added to the table

replace the old item

added to the table according to some other rules.

REFRENCES

Knuth, D. 1973, The Art of Computer Science, Vol. 3, Sorting and Searching, p.527. Addison-Wesley, Reading, MA., United States

https://youtu.be/b4b8ktEV4Bg

https://youtu.be/DMtFhACPnTY

**Message Authentication**

**Common attacks**

**Crypto cracking**

**Other technologies that rely on cryptography**

**Blockchain**

**VPN**

A VPN or Virtual Private Network.

**Short note on future of cryptography(quantum)**

## Quantum Computing

Computers are constantly getting faster as we develop new technology to do tasks in fractions of the time. This means we need to make the parts smaller and smaller so we can fit more power into the components. Computer parts are starting to approach the same size as an atom. At this scale physics as we know it completely breaks apart and a new rule set appears. Transistors block the passage of these electrons and this is the basis of all computing. At these incredibly small scales however electrons can just pass through the transistors using a process called quantum tunnelling. The next step in computing which people are currently working on is Quantum Computing. Quantum computers use Qbits rather than bits. Qbits can be set to the same values of 0 and 1. However the Qbit doesn't have to be in one state or the other. It can be either but once you measure the Qbit it has to pick 0 or 1. Let’s say you have 4 bits and 4 Qbits. The 4 bits can only be in 1 of the 16 possible combinations at any given time. Qbits however can be in every single one of those 16 combinations at the same time.

## Impact on Cryptography

A Quantum Computer will be vastly superior to a regular computer when it comes to things that means dealing with very large amounts of data. Certain things like searching through the data or checking every possible combination of data become so much faster when a Quantum Computer is used. Quantum Computers only require the square root of the time a normal computer would need to do a task like this. In banking, you give people a public key which is used to encode messages only you can decode using your private key. This public key can be used to calculate the private key. With a regular computer this would take way too long. It would be constant trial and error until it figures all the maths out. Quantum Computers could do the maths for it so much faster. This makes current cryptography next to useless. People will need to start working on new and different ways of security and encryption of data. Who knows, maybe in the future people will be using Quantum computers to try break through security measures ran by another Quantum Computer. Fighting fire with fire if you will. Right now we aren’t anywhere near the stage where Quantum Computers will be widely used. They will probably never replace the household PC but their uses in society are definitely going to be helpful in progressing our knowledge about the Universe.

**Conclusion**

**List of references**

# ***Cryptography and Public Key Infrastructure on the Internet (2006)***

# ***By Klaus Schmeh***

*Mjølsnes, S. F. et al. (2008) Public Key Infrastructure 5th European PKI Workshop: Theory and Practice, EuroPKI 2008 Trondheim, Norway, June 16-17, 2008, Proceedings . 1st ed. 2008. [Online]. Berlin, Heidelberg: Springer Berlin Heidelberg.*

A.E.S. Corporate, A. (2005) 'Advanced Encryption Standard - AES 4th International Conference, AES 2004, Bonn, Germany, May 10-12, 2004, Revised Selected and Invited Papers', available: <http://dx.doi.org/10.1007/b137765>.

Buchanan, W.J. (2017) *Cryptography*, Gistrup, Denmark ;:River Publishers.

Isobe, T., Ohigashi, T., Watanabe, Y. and Morii, M. (2014) *Full Plaintext Recovery Attack on Broadcast RC4*.

Nandi, S., Kar, B.K. and Pal Chaudhuri, P. (1994) 'Theory and applications of cellular automata in cryptography', *IEEE Transactions on Computers*, 43(12), 1346-1357, available: <http://dx.doi.org/10.1109/12.338094>.

Paar, C. (2010) *Understanding Cryptography: A Textbook for Students and Practitioners*.

Theory of Cryptography Conference Corporate, A. (2010) *Theory of Cryptography 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010, Proceedings*,1st ed. 2010. ed., Berlin, Heidelberg:Springer Berlin Heidelberg.

**Bibliography**

Canetti R., Tauman Kalai, Y., Varia, M., and Wich, D. (<https://link-springer-com.proxy.lib.ul.ie/book/10.1007%2F978-3-642-11799-2>)

Kurzgesagt – In a Nutshell (2015) 'Quantum Computers Explained – Limits of Human Technology', *Futurism* [video], available: https://www.youtube.com/watch?v=JhHMJCUmq28 [accessed 10 Apr 2020].