CRYPTOLOGY AND INFORMATION SECURITY - PAST, PRESENT, AND FUTURE ROLE IN SOCIETY

Soumitra Bhattacharya
Life Member ISTAM-(IIT-Kharagpur), & CRSI-Kolkata, India

ABSTRACT

This article provides a general introduction to the subject of Cryptology, Cryptography and Cryptoanalysis and explains the terminology and the practical application of security techniques... In recorded History, technological innovations have revolutionized societies. The printing press is an often-cited example of the great impact one humble person’s invention can have on ruling dynasties, world religions, and personal life.. Quantum encryption could rival Gutenberg’s printing press in its impact. Cryptology has a fascinating History. In Warfare Cryptography is a broad, sticky, and mathematically complex, but interesting subject and an integral part of the evolution of warfare. So let’s get some definitions out of the way first. Cryptology is the study of codes, both creating and solving them. Cryptography is the art of creating codes. Cryptanalysis is the art of surreptitiously revealing the contents of coded messages, breaking codes, that were not intended for you as a recipient. Secondly, there are nomenclators and enciphers. Nomenclators are letters or numbers that represent words or phrases, like 103A means “meet me at 4PM”. Ciphers are alphabetical letters or numbers that are encrypted using some sequential coding process and a key. For this essay, we will refer to both as codes. Also, enciphered, encrypted and encoded mean the same thing....Additionally, there is plain text. This is the original message that is readable and understandable, uncoded or unencrypted. Once it goes through the coding process and is encrypted, the output is readable but not understandable. There are a bunch of other terms like steganography, homophones, polyphones, digraphs, bigrams, and polygrams, but they are just variations of coding and decoding techniques.

KEYWORDS

Network Protocols, Mathematical Cryptology, encryption, decryption

1. CRYPTOLOGY

The study of cryptography and cryptanalysis. Cryptology is the mathematics, such as number theory, and the application of formulas and algorithms, that underpin cryptography and cryptanalysis. Since the cryptanalysis concepts are highly specialized and complex, we concentrate here only on some of the key mathematical concepts behind cryptography. cryptology is made up of cryptography and cryptanalysis.

1) Cryptography- The art and science of making ciphers. Which concerns with the design of the cryptosystems and cryptography, is the actual securing, control, and identification of digital data.

2) Cryptoanalysis— The art and science of breaking ciphers. In other words, the extraction of mm given cc, EE, DD, and possibly keke. which aims at studies relating to breaking of cryptosystems. cryptanalysis, is made up of all the attempts one might develop to undermine, circumvent, and/or break what the first part, cryptography, is attempting to accomplish.

3) Cryptosystems:-A particular suite of algorithms and protocols for encryption, decryption, and key generation. Examples: Cramer-Shoup cryptosystem, Rabin cryptosystem, Benaloh cryptosystem, RSA cryptosystem.
4) **Cryptographic System:** Any system that uses cryptography.

5) **Cipher:** An algorithm used in a cryptosystem.

6) **Confusion:** The property of having the relationship between the plaintext, ciphertext, and key so complicated as to be useless to the cryptanalyst.

7) **Diffusion:** The property of having statistical patterns in the plaintext spread widely throughout the ciphertext.

8) **Frequency Analysis**
   - Known plaintext attack
   - Known ciphertext attack
   - Chosen plaintext attack
   - Chosen key attack
   - Linear cryptanalysis
   - Differential cryptanalysis
   - Theft
   - Bribery Blackmail

**What tools use cryptography?** Some form of cryptography can be found nearly everywhere in computer technology. Popular standalone programs, like PGP and GPG, aid in securing communications. Web browsers and other programs implement cryptographic layers in their channels. Drivers and programs exist to secure files on disk and control access thereto. Some commercial programs use cryptographic mechanisms to limit where their installation and use may occur. Basically, every time you find a need to control the access and usage of computer programs or digital data, you’ll find that cryptographic algorithms constitute important parts of the protocol for use of these programs/data.

2. **A LITTLE HISTORY**

**Cryptology,** the study of coded messages, dates back to Egypt about 1,900 BC, when a scribe carved some hieroglyphic symbols into a rock at the tomb of Khnumhotep II. Cryptology wasn’t that hard back then, since most of the people were illiterate and only the elite could read any written language. Pharaohs and potentates, kings and queens, presidents and dictators, and military commanders have used cryptology to hide their communications from their enemies ever since. An interesting example is Mary, Queen of Scots. In 1577, Don Juan, ruler of Austria, had worked out a plan to invade England, dethrone and kill Queen Elizabeth, put Mary on the throne, and marry her. But, Elizabeth’s minister, Francis Walsingham, smelled a rat and had Mary confined to castles away from Elizabeth Mary’s page, Antony Babington, began plotting to have Elizabeth assassinated 1586, with the help of Don Juan and Phillip, the king of the Netherlands. Babington employed Gilbert Gifford to smuggle encrypted communications between him and Mary out of the castle in beer barrels. As it turns out, Gifford was an agent working for Walsingham, so those messages got delivered to him immediately. He employed Thomas Phelippes to decrypt those messages which thoroughly incriminated Mary of plotting to kill Elizabeth, and she was convicted of high treason. So, at 8AM on the morning of 8 February 1587, the executioner chopped off Mary’s head with an axe. Such is the importance of insuring your code is hard to break. Thomas Philpypes became England’s first cryptanalyst, establishing the role of code breakers in the future.
3. CODE TYPES

There are various types of codes in history: mono-alphabetic, poly-alphabetic, columnar transposition, S-boxes and P-boxes, Polybius squares, etc. Many primitive machines were made to encrypt/decrypt messages. One of the first used a wooden dowel about an inch in diameter and two feet long. A 1/4 inch-wide piece of parchment, about 3 feet long, was wrapped around the dowel, edge to edge like a barber pole. The plaintext message was then written vertically on the dowel in columns. When unraveled from the pole and laid out flat, it would look like a series of letters on a strip of paper that made no sense. Only when the intended recipient wrapped the 1/4 inch strip of parchment around his 1-inch wooden dowel could the characters be read in the proper order, vertically. Other early mechanisms used were identical cipher wheels and cipher drums held by both the sender and receiver. Other codes used simple key words shared by two people, and an agreement on how the code would be structured. That could be worked out mentally on paper, without the use of a mechanical mechanism. As you can see, many early codes and code machines were cumbersome and it took a lot of time to encrypt or decrypt a message...Although the early codes were unsophisticated and primitive, they taught many lessons. The first is that the only people with the knowledge to create secure codes are the cryptanalysts: they know the weaknesses of the codes they have already broken. The second lesson is that the most important element of a code is the key: it does not matter if your enemy knows how your code is structured if you have complex keys. Third, the longer your key, the more difficult your code is to break. If your key is the same length as your message, your message is basically unbreakable. Fourth, the longer your message, the easier it is to break. And finally, always send the message in five-letter groups, to hide the number and length of words.

3.1 Externals and internals

There are two parts to any encrypted message: the externals, and the internals. Cryptanalysis should discover the internals, the process by which the encrypted message was created, by using the external characteristics of the message. The first technique is character frequency analysis. The National Security Agency (NSA) has run millions of pages of newspapers, books, periodicals, and magazines through their computers, establishing the letter frequency charts of the English language: these are the percentages of each letter in a page of text: a= 8.167 percent, e= 12.702 percent, i=6.966 percent, o=7.507 percent, u=2.758 percent, and so on. With an encrypted message using a simple substitution code where x=e, z=a, etc, it’s child’s play to decrypt it just using the character frequency tables. Our intelligence people have such tables for all the significant languages on the planet.

4. NSA EFFORTS

In 1969, there were 96,000 people working for the NSA, including the old Army Security Agency (ASA) unit, that were intercepting enemy messages, running the frequency/proximity/symmetry charts against those messages, decrypting them, analyzing them, cataloging them, and disseminating the intelligence to our military commanders. Today, that process is done with acres of NSA supercomputers discussed in the previous installment of this warfare series. Now you know why NSA has the most computing power on the planet. With massive amounts of computer power, relatively sophisticated codes can be broken in a short time with these brute-force methods. Today’s advanced codes, like Logarithmic and Elliptical coding schemes, are very secure but not unbreakable. It takes millions of years for our computers to discover the two prime numbers multiplied together to create a 200-digit key used for encrypting a message. A quantum
computer can do it in an hour, according to cryptanalysts, which is why NSA has been funding quantum computer research and development since the 90’s. Turn that decryption concept around, and quantum computers could create massive keys, as long as the message itself, that make them invulnerable to cryptanalysis techniques and decoding by our enemies. Auguste Kerckhoffs, the author of La Cryptographie militaire, once said, “...secrecy must reside solely in the keys.” As you can see from the examples of frequency analysis above, all encrypting techniques and their resulting coded messages, have certain idiosyncrasies or patterns that offer a good cryptanalyst a crack to get his fingers into. With a bit of work, by him or computers, he can widen that crack and get inside, recover the keys and the coding scheme, and break it.

4.1 Time shifting

The scientists at Purdue University have figured out how hide data in a light beam. They take two beams and modulate them slightly out of phase with each other on an optical fiber. That creates “holes in time”, between the two beam phases. Into that hole, they drop the plaintext data. If you look at the modulated beam, you see a flat line, no light, where the data is now located. The receiver, knowing the modulation patterns, can simply demodulate the beam and recover the data. The modulation patterns are the key to the encoding scheme. For many years, computers have been doing the in-stream encryption and decryption for our military and government communications. That makes the process fast and secure. When the Pueblo was captured by the North Koreans in January 1968, the President of the United States knew about it within a few minutes via secure encrypted communications from ASA intelligence people in the Pacific to NSA at Fort Meade…. The Pueblo had top secret crypto machines onboard, the same models used by NSA and ASA, and the North Koreans grabbed them intact (they shared them with the Russians too). Did the U.S. intelligence groups have to throw away all the crypto machines and replace them with new ones? No. All we had to do was change the keys and the captured machines were basically worthless to the North Koreans and the Russians. So, Kerckhoffs was correct: the keys are the key. If you want to dig deeper into this topic, the best book on the history of cryptology is “The Codebreakers” by David Kahn (if you can find a copy). It’s over 1,000 pages long but worth a read. The chapters about breaking the German Enigma machines and the Japanese Purple code in WWII are astonishingly detailed. If you want to understand basic encryption techniques and more history, read “The Code Book” by Simon Singh. If you are not mathematically challenged, read “Cryptanalysis” by Helen Fouche Gaines. Same goes for “Decrypted Secrets” by F. L. Bauer, very heavy on formulas. And finally, there’s the ever-popular “Modern Cryptanalysis” by Christopher Swenson (this is a text book with exam questions, problems to solve, and software examples in Python, so you can program your analysis techniques into your computer). I’ll warn you now: these books are not light reading. For our next installment, we will take a look at the military capabilities of our closest enemies, through the eyes of our intelligence folks. That will give you an informed idea of where we stand in a war with any of them.
5. **BACKGROUND**

**Goals:**
- **Confidentiality:** It should cost Eve more to recover \( mm \) than \( mm \) is worth.
- **Authentication:** Bob should be able to verify that it was Alice that sent \( mm \).
- **Integrity:** Bob should be able to verify that \( mm \) was not tampered with.
- **Nonrepudiation:** Alice should not be able to deny sending \( mm \).

The best cryptosystems assume that Eve and Mallory know \( EE, DD, cc \), and, if \( ke \neq kd \), then \( ke \) as well. Most cryptosystems do **not** rely on their algorithms being kept secret, because:

If your secret algorithm is compromised (someone could leave your group), you have to change it, and that is way harder than just changing a key!

Public algorithms can be subjected to thousands of experts looking for flaws, so you have some degree of confidence in those that withstand scrutiny.

The study of cryptology includes the design of various ciphers, cryptanalysis methods (attacks), key exchange, key authentication, cryptographic hashing, digital signing, and social issues (legal, political, etc.).

**Protocols and algorithms**

When considering cryptology, it is important to make the distinction between protocols and algorithms. This is especially important in light of the misleading claims sometimes made by companies that produce cryptographic products (either out of carelessness or misrepresentation). For example, a company might claim: "If you use our product, your data is secure because it would take a million years for the fastest computers to break our encryption!" The claim can be true, but still not make for a very good product. A true claim about the strength of an algorithm by itself does not necessarily mean that a whole protocol that uses that algorithm as one of its steps does not have other weaknesses.
A protocol is a specification of the complete set of steps involved in carrying out a cryptographic activity, including explicit specification of how to proceed in every contingency. An algorithm is the much more narrow procedure involved in transforming some digital data into some other digital data. Cryptographic protocols inevitably involve using one or more cryptographic algorithms, but security (and other cryptographic goals) is the product of a total protocol.

6. TECHNOLOGICAL BACKGROUND

There are two basic types of encryption commonly used today, symmetric key and asymmetric key encryption. Although the two methods are very different in theory and application, similar terminology is used to describe the processes:

**Plaintext:** The data or message to be sent, in a clear form anyone can read.

**Cipher text:** The data in encrypted form.

**Bit:** Binary digit, the basic unit of information stored by a computer. Any letter or number can be encoded as a string of 8 bits.

**Algorithm:** The method used to encrypt and decrypt data, also called a “Cipher.”

**Key:** A crucial parameter in the algorithm.

**Hash:** A fingerprint for a digital file.

**Alice and Bob:** Alice is trying to send Bob a message over an insecure channel. Eve wants to eavesdrop.

**Attack:** A method that can decrypt the message for an interceptor.

**Shannon’s Maxim:** The enemy knows the system! A secure algorithm must assume the enemy knows everything about the system except the key.

The goal of this section is to provide a brief overview of how ciphers work and the history of cryptography. The scope includes everything from World War I and excludes the field of classical cryptography.

Symmetric Key Encryption:- Symmetric key encryption is the older and better-known technique. At its most primitive, the algorithm could be “shift each letter alphabetically” and the key could be “+2.” Therefore, the Alice will simply shift each letter by 2 spaces to convert plaintext to cipher text, and Bob will simply shift back 2 spaces to decrypt the message.

For example: Plaintext: MARK IS A SPY

Alice shifts each letter +2: OCTM KU C URA
Bob shifts each letter -2: MARK IS A SPY

There are three characteristics of this simple exercise that also hold true for even the most complex symmetric key algorithms:

Alice and Bob use the same key to both encrypt and decrypt.
The method is useless if they key is not kept privately between Alice and Bob, which is why this method is sometimes referred to as private key encryption.

Alice must first securely notify Bob of her key.

The last characteristic is the method’s greatest limitation. The key, which must be sent in plaintext, can be intercepted. Overcoming or exploiting this weakness is a reoccurring theme in this report and also a focus of cryptographic research.

Examples of Symmetric Key

I present four examples to illustrate symmetric key cryptography: Enigma: Historical yet fascinating example

- One-time Pad: Unbreakable but hard to implement
- Stream Cipher: Vulnerable yet still foolishly used.
- Block Cipher: The current industry standard in security.

7. **Enigma**

The Enigma machines are a series of electro-mechanical rotor cipher machines, mainly developed and used in the early- to mid-20th century to protect commercial, diplomatic and military communication. Enigma was invented by the German engineer Arthur Scherbius at the end of World War I. Early models were used commercially from the early 1920s, and adopted by military and government services of several countries, most notably Nazi Germany before and during World War II. Several different Enigma models were produced, but the German military models, having a plugboard, were the most complex. Japanese and Italian models were in use:

- Kinds of Ciphers
- Here are some useful categories of ciphers. Note that a particular cipher may belong to more than one of these categories.
  - Classical (a.k.a. manual): A cipher easy enough to be performed by hand, usually character-based
  - Modern: Pretty much any cipher that isn’t classical
  - Substitution: Each character of the plaintext is replaced with one or more characters to make the ciphertext
Transposition: Characters in the plaintext are rearranged to form the ciphertext
Monoalphabetic: A substitution cipher in which a character of the plaintext is always replaced by the same character
Polyalphabetic: A substitution cipher that essentially uses multiple monoalphabetic substitution mappings
Homophonic: A substitution in which one character can map to one of a set of characters
Polygraphic: A substitution of blocks of characters for blocks of characters
Periodic: A polyalphabetic cipher in which the replacement scheme repeats
Non-periodic: (Self-explanatory if you understand periodic)
Block: Encryption takes place not per character but per blocks of characters.
Stream: A cipher operating on a data stream of unknown length, usually incorporating feedback.
Secret Key: A cipher in which keke and kdkd are the same or trivially derivable from one another; requires the parties to meet in secret to exchange the keys they’ll be using. Also called symmetric.
Public Key: A scheme in which everyone’s encryption key is publicly known but their decryption key is kept secret. Also called asymmetric.

NOTE: In the character-based examples below, we’ll assume (without any loss of generality) a 26 symbol alphabet (A..Z).

**Homophonic Substitution:** Each plaintext letter maps to one or more symbols in the ciphertext. The number of targets should be proportional to its frequency (to defeat frequency analysis).

Example:

<table>
<thead>
<tr>
<th>Letter</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>12 15 36 50 56 70 81 95</td>
</tr>
<tr>
<td>B</td>
<td>51 84</td>
</tr>
<tr>
<td>C</td>
<td>16 44 65</td>
</tr>
<tr>
<td>D</td>
<td>04 06 48 82</td>
</tr>
<tr>
<td>E</td>
<td>01 17 19 34 47 49 58 60 67 77 85 90</td>
</tr>
<tr>
<td>F</td>
<td>13 27</td>
</tr>
<tr>
<td>G</td>
<td>09 28</td>
</tr>
<tr>
<td>H</td>
<td>26 42 53 59 68 71</td>
</tr>
<tr>
<td>I</td>
<td>35 73 76 86 91 96</td>
</tr>
<tr>
<td>J</td>
<td>18</td>
</tr>
<tr>
<td>K</td>
<td>07</td>
</tr>
<tr>
<td>L</td>
<td>29 40 54 87</td>
</tr>
<tr>
<td>M</td>
<td>25 30</td>
</tr>
<tr>
<td>N</td>
<td>21 61 62 69 74 94</td>
</tr>
<tr>
<td>O</td>
<td>02 03 08 10 57 75 93</td>
</tr>
<tr>
<td>P</td>
<td>41 98</td>
</tr>
<tr>
<td>Q</td>
<td>97</td>
</tr>
<tr>
<td>R</td>
<td>32 38 43 45 80 83</td>
</tr>
<tr>
<td>S</td>
<td>14 22 39 79 89 99</td>
</tr>
<tr>
<td>T</td>
<td>00 20 23 33 46 52 72 78 88</td>
</tr>
<tr>
<td>U</td>
<td>11 64 66</td>
</tr>
<tr>
<td>V</td>
<td>37</td>
</tr>
<tr>
<td>W</td>
<td>63 92</td>
</tr>
<tr>
<td>X</td>
<td>31</td>
</tr>
<tr>
<td>Y</td>
<td>24 55</td>
</tr>
<tr>
<td>Z</td>
<td>05</td>
</tr>
</tbody>
</table>
To encrypt, choose randomly among possibilities. Example, one possible encryption of ATTACKATDAWN is

56 78 20 95 65 07 12 72 06 50 92 61

Secret Key Cryptography

Secret key (a.k.a. symmetric key) ciphers are much faster than public key ciphers, but key management can be a huge problem. If n people in a group need to communicate, they need n(n−1) /2 keys. Keys must be distributed securely (in secret). Keys must be kept safe. Keys should be changed frequently, which feeds back into the distribution headache.

8. MODERN CRYPTOGRAPHIC METHODS

Now that we have Shannon’s information theory, very powerful computers, and centuries of theory and practice behind us, most modern techniques operate on bit strings, not character strings are careful to completely mask patterns and redundancies in the plaintext use random keys (that can be reused, too) ensure that very slight changes in the plaintext affect a large portion of the ciphertext (and vice versa). This is called the Avalanche Effect. In addition, it’s nice if the cipher is

- efficient
- fault-tolerant
- Most ciphers are either block ciphers or stream ciphers. Block ciphers require padding and can operate in different modes (See Schnier’s book or the Wikipedia article:
  - ECB — Electronic Codebook
  - CBC — Cipher Block Chaining
  - CFB — Cipher Feedback
  - OFB — Output Feedback
  - CTR — Counter
  - BC — Block Chaining
  - PCBC — Propagating Cipher Block Chaining
  - CBCC — Cipher Block Chaining with Checksum
  - DES
  - IDEA
  - RC4
  - RC6
  - Blowfish
  - Twofish

AES

AES is the new standard, replacing DES. It was the winner of the competition (in 2001), where is was submitted under the name Rijndael, beating out RC6, Serpent, MARS, and Twofish.

Key Exchange
**Diffie and Hellman** (the 2015 Turing Award Winners) and their friend Merkle showed in 1976 that it was possible for two people to exchange a secret key *without having to actually meet in secret*:

- Alice picks a prime \( n \) and sends this in the clear to Bob
- Alice picks a primitive root \( g \mod n \) (how to find), called \( g \), and sends this in the clear to Bob
- Alice picks a secret integer \( a \), and sends \( g^a \mod n \) in the clear to Bob
- Bob picks a secret integer \( b \), and sends \( g^b \mod n \) in the clear to Alice
- Alice computes \((g^b \mod n)^a\) and Bob computes \((g^a \mod n)\). This is the key! (It’s \( g^{ab} \mod n \))

This is probably secure, provided \( n \) is very large and \( n-1 \) is also prime, because although Eve knows \( g \), \( n \), \( g^a \mod n \), and \( g^b \mod n \), there’s no known efficient way to get \( a \) or \( b \) from these. That’s the discrete logarithm problem, remember?

**Example with small \( n \):**

- Alice picks \( n=208799 \) and \( g=13 \) and sends these to Bob
- Alice picks \( a=152335 \) and Bob picks \( b=98113 \)
- Alice sends Bob \( 13^{152335} \mod 208799 = 73033 \)
- Bob sends Alice \( 13^{98133} \mod 208799 = 147540 \)
- Alice computes \( 147540^{152335} \mod 208799 = 8435 \)
- Bob computes \( 73033^{98133} \mod 208799 = 8435 \)

The secret key is **84358435**.

**Public key cryptography** is a cryptographic technique that enables entities to securely communicate on an insecure **public network**, and reliably verify the identity of an entity via digital signatures…Public key ciphers solve the key management nightmare of secret key ciphers, at the cost of speed. In a group of \( n \) people one needs only \( n \) public keys and \( n \) private keys…The Public key infrastructure (PKI) is the set of hardware, software, policies, processes, and procedures required to create, manage, distribute, use, store, and revoke digital certificates and public-keys. The PKI is the foundation that enables the use of technologies, such as digital signatures and encryption, across large user populations. PKIs deliver the elements essential for a secure and trusted business environment for e-commerce and the growing Internet of Things (IoT)…PKIs help establish the identity of people, devices, and services – enabling controlled access to systems and resources, protection of data, and accountability in transactions. Next generation business applications are becoming more reliant on public key infrastructure (PKI) technology to guarantee high assurance as evolving business models are becoming more dependent on electronic interaction requiring online authentication and compliance with stricter data security regulations.

### 9. Cryptographic Security

Using the principles of asymmetric and symmetric cryptography, PKIs facilitate the establishment of a secure exchange of data between users and devices – ensuring authenticity, confidentiality, and integrity of transactions. Users (also known as “Subscribers” in PKI parlance) can be individual end users, web servers, embedded systems, connected devices, or programs/applications that are executing business processes. Asymmetric cryptography provides the users, devices or services within an ecosystem with a key pair composed of a public and a private key component. A public key is available to anyone in the group for encryption or for
verification of a digital signature. The private key on the other hand, must be kept secret and is only used by the entity to which it belongs, typically for tasks such as decryption or for the creation of digital signatures.

The Increasing Importance of PKIs

With evolving business models becoming more dependent on electronic transactions and digital documents, and with more Internet-aware devices connected to corporate networks, the role of a public key infrastructure is no longer limited to isolated systems such as secure email, smart cards for physical access or encrypted web traffic. PKIs today are expected to support larger numbers of applications, users and devices across complex ecosystems. And with stricter government and industry data security regulations, mainstream operating systems and business applications are becoming more reliant than ever on an organizational PKI to guarantee trust.

RSA Cryptosystem

Diffie-Hellman doesn’t do encryption; it just exchanges a key. RSA can encrypt and decrypt. Here’s how. ..

Each person generates two large primes, p and q. Publishes his or her public key (N,e) where N=pq. Computes d = modular inverse of e relative to (p−1)(q−1), keeping it secret. Destroys p and q.

Now check this out:

For Alice to send a message m to Bob, she sends c=me mod N. Bob decodes this easily because m=cd mod N.

RSA algorithm is asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. Public Key and Private Key. As the name describes that the Public Key is given to everyone and Private key is kept private. Let us learn the mechanism behind RSA algorithm:

Generating Public Key:
Select two prime no's. Suppose P = 53 and Q = 59.
Now First part of the Public key : n = P*Q = 3127.
We also need a small exponent say e:
But Must be
• An integer.
• Not a factor of n.

Generating Private Key: We need to calculate \( \Phi(n) \):

• Such that \( \Phi(n) = (P-1)(Q-1) \)
• so, \( \Phi(n) = 3016 \)
• Now calculate Private Key, d:
• \( d = (k*\Phi(n) + 1) / e \) for some integer k
• For k = 2, value of d is 2011.
• Now we are ready with our – Public Key ( n = 3127 and e = 3) and Private Key(d = 2011)
• Now we will encrypt “HI”:
• Convert letters to numbers: H = 8 and I = 9
• Thus Encrypted Data c = 89e mod n.
Thus our Encrypted Data comes out to be 1394
Now we will decrypt 1394:
\[ \text{Decrypted Data} = cd \mod n \]
Thus our Encrypted Data comes out to be 89
\[ 8 = H \text{ and } I = 9 \text{ i.e. } "HI". \]

10. CRYPTOGRAPHIC HASHING

Cryptography hash functions play a fundamental role in modern cryptography. Hash function takes a message as input and produce an output referred to as a hash code, hash-result, hash-value or a simple hash. A hash, a.k.a. fingerprint, checksum, message digest is a bit pattern (usually around 160 bits or so), generated from a message by a cryptographic hash function. For the hash to be secure, or cryptographic, it must be computationally infeasible to find a message that hashes to a given value (one wayness) be computationally infeasible to find two messages that hash to the same value. Usually the change of just a single bit in the message will cause the digest to look completely and totally different.

11. BASIC MATHEMATICAL CRYPTOLOGY

An original plain message is called plaintext and an encrypted one is called cryptotext.

**Encryption** is the process of encoding a message or an information in such a way that only authorized recipient can access it and any other unauthorized cannot... Encryption is the key concept of cryptography – It is the process where a message is encoded in such a format that cannot be read or understood by an eavesdropper... An algorithm is basically a procedure or a formula for solving a data snooping problem. An encryption algorithm is a set of mathematical procedure for performing encryption on data... when encrypting you need to have a so-called key, a usually quite complicated parameter that you can use to change the encryption.
The authorized receiver of the message decrypts it. Decryption is the process of converting the encrypted message / text back to its original (readable) format / Plain Text.. An authorized recipient can easily decrypt the message for which he/she needs to have his/her own key. An encrypting procedure can encrypt a continuous stream of symbols (stream encryption) or divide it into blocks (block encryption). Sometimes in the block encryption the sizes of blocks can vary, but a certain maximum size of block should not be exceeded. However, blocks are usually of the same size. In what follows block encryption, where it’s sufficient to consider encrypting and decrypting of an arbitrary message block, and one arbitrary message block may be considered as the plaintext and its encrypted version as the cryptotext.

An encryption procedure is symmetric, if the encrypting and decrypting keys are the same, or **Symmetric encryption** is a type of encryption where only one key (a secret key) is used to both encrypt and decrypt the information /message. It’s easy to derive one from the other. In nonsymmetric encryption or Asymmetric Encryption also known as public key cryptography, the decrypting key can’t be derived from the encrypting key with any small amount of work. In that case the encrypting key can be public while the decrypting key stays classified. This kind of encryption procedure is known as public-key cryptography, correspondingly symmetric encrypting is called secretkey cryptography. The problem with symmetric encrypting is the secret key distribution to all parties, as keys must also be updated every now and then. Symmetric encryption can be characterized as a so called cryptosystem which is an ordered quintet \((P, C, K, E, D)\), where

- **P** is the finite message space (plaintexts).
- **C** is the finite cryptotext space (cryptotexts).
- **K** is the finite key space for every key \(k \in K\) there is an encrypting function \(e_k \in E\) and a decrypting function \(d_k \in D\). \(E\) is called the encrypting function space which includes every possible encrypting function and \(D\) is called the decrypting function space which includes every possible decrypting function.
- \(d_k(e_k(w)) = w\) holds for every message (block) \(w\) and key \(k\).

It would seem that an encrypting function must be injective, so that it won’t encrypt two different plaintexts to the same cryptotext. Encryption can still be random, and an encrypting function can encrypt the same plaintext to several different cryptotexts, so an encrypting function is not actually a mathematical function. On the other hand, encrypting functions don’t always have to be injective functions, if there’s a limited amount of plaintexts which correspond to the same cryptotext and it’s easy to find the right one of them.
12. NUMBER THEORY.

Divisibility. Factors. Primes

Certain concepts and results of number theory come up often in cryptology, even though the procedure itself doesn’t have anything to do with number theory. The set of all integers is denoted by $\mathbb{Z}$. The set of nonnegative integers $\{0, 1, 2, \ldots \}$ is called the set of natural numbers and it’s denoted by $\mathbb{N}$.

Addition and multiplication of integers are familiar commutative and associative operations, with identity elements 0 and 1 respectively. Also recall the distributive law $x(y + z) = xy + xz$ and the definitions of opposite number $-x = (-1)x$ and subtraction $x - y = x + (-1)y$. Division of integers means the following operation: When dividing an integer $x$ (dividend) by an integer $y \not= 0$ (divisor), $x$ is to be given in the form

$$x = qy + r,$$

where the integer $r$ is called remainder and fulfills the condition $0 \leq r < |y|$. The integer $q$ is called quotient. Adding repeatedly $-y$ or $y$ to $x$ we see that it’s possible to write $x$ in the desired form. If it’s possible to give $x$ in the form $x = qy$, where $q$ is an integer then it’s said that $x$ is divisible by $y$ or that $y$ divides $x$ or that $y$ is a factor of $x$ or that $x$ is a multiple of $y$, and this is denoted by $y | x$. The so-called trivial factors of an integer $x$ are $\pm 1$ and $\pm x$. Possible other factors are nontrivial.

The following properties of divisibility are quite obvious:

1. $0$ is divisible by any integer, but divides only itself.
2. $1$ and $-1$ divide all integers, but are divisible only by themselves and by one another.
3. If $y | x$ and $x \neq 0$ then $|y| \leq |x|$. (4) If $x | y$ and $y | z$ then also $x | z$ (in other words, divisibility is transitive).
4. If $x | y$ and $y | z$ then also $x | z$ (in other words, divisibility is transitive).
5. If $x | y$ and $z$ then also $x | y \pm z$.
6. If $x | y$ and $z$ is an integer then $x | yz$.

The result of division is unique since, if $x = q_1x + r_1 = q_2y + r_2$, where $q_1$, $q_2$, $r_1$, $r_2$ are integers and $0 \leq r_1$, $r_2 < |y|$, then $y$ divides $r_1 - r_2$. From the fact that $|r_1 - r_2| < |y|$ it then follows that $r_1 = r_2$ and further that $q_1 = q_2$.

An integer that has only trivial factors is called indivisible. An indivisible integer is a prime number or just a prime, if it is $\geq 2$. The first few primes are 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, $\ldots$ 2 is the only even prime. One basic task is to test whether or not a natural number is a prime. An integer, which is $\geq 2$ and is not a prime, is called composite.

Representations of Integers in Different Bases:- The most common way to represent an integer is to use the familiar decimal representation or in other words base-10 representation. Base-2 representation, called the binary representation, is also often used and so is base-8 octal representation and base-16 hexadecimal representation. The general base representation is given by

**Theorem-1 :-** If $k \geq 2$ then every positive integer $x$ can be represented uniquely in the form

$$x = a_nk^n + a_{n-1}k^{n-1} + \cdots + a_1k + a_0$$

where $0 \leq a_0, a_1, \ldots, a_n \leq k - 1$ and $a_n > 0$. This is called base-$k$ representation of $x$, where $k$ is the base (number) or radix and $n + 1$ is the length of the representation.
**Proof.** The representation, i.e. the sequence \( a_n, a_{n-1}, \ldots, a_0 \), is obtained by the following algorithm:

1. Set \( K \leftarrow \text{NULL} \) (the empty sequence).
2. Divide \( x \) by the radix \( K \):
   \[ x = qk + r \quad \text{(quotient } q, \text{ remainder } r). \]
   Set \( K \leftarrow r, K \) and \( x \leftarrow q \).
3. If \( x = 0 \) then return \( K \) and quit. Else repeat #2. \( x \) gets smaller and smaller in #2 with each iteration and so the procedure stops eventually in #3.

The base-k representation is unique because if
\[
x = a_nk^n + a_{n-1}k^{n-1} + \cdots + a_1k + a_0 = b_mk^m + b_{m-1}k^{m-1} + \cdots + b_1k + b_0,
\]
where \( 0 \leq a_0, a_1, \ldots, a_n, b_1, \ldots, b_m \leq k-1 \) and \( a_n, b_m > 0 \) and \( n \geq m \), then we first conclude that \( n = m \). Indeed, if \( n > m \) then we also have
\[
b_mk^m + b_{m-1}k^{m-1} + \cdots + b_1k + b_0 \leq (k-1)k^m + (k-1)k^{m-1} + \cdots + (k-1)k + k - 1 = k^{m-1} - 1.
\]

If \( k^m \leq a_nk^n + a_{n-1}k^{n-1} + \cdots + a_1k + a_0 \), which is a contradiction. So \( n = m \), that is, the length of the representation must be unique. Similarly we can conclude that \( an = bn \), because if
\[
a_n > b_n\text{ then }
b_k^n + b_{n-1}k^{n-1} + \cdots + b_1k + b_0 \leq (an-1)k^n + (k-1)k^{n-1} + \cdots + (k-1)k + k - 1 = ak^n - a_nk^n + a_{n-1}k^{n-1} + \cdots + a_1k + a_0,
\]
Which is also a contradiction. Again in the same way we can conclude that \( an-1 = bn-1 \) and so on. Representation of the number 0 is basically an empty sequence in every base. This of course creates problems and so we agree on the convention that the representation of 0 is 0. Conversion between base representations, the so-called change of base or radix transformation, is a basic task concerning integers.

**Theorem 2:** The length of the base-k representation of a positive integer \( x \) is
\[
[\log_k x] + 1 = [\log_k (x + 1)] \quad \text{where } \log_k \text{ is the base-k logarithm}^3
\]

**Proof:** If the base-k representation of \( x \) is \( x = a_nk^n + a_{n-1}k^{n-1} + \cdots + a_1k + a_0 \) then its length is \( s = n + 1 \).

It is apparent that \( x \geq k^n \), and on the other hand that
\[
x \leq (k-1)k^n + (k-1)k^{n-1} + \cdots + (k-1)k + k - 1 = k^{n-1} - 1 < k^n.
\]
Since \( k^{n-1} \leq x < k^n \), then \( s = [\log_k x] + 1 \). Then again \( k^{s-1} \leq x + 1 \leq k^s \), whence \( s - 1 < \log_k (x + 1) \leq s \) and so
\[
s = [\log_k (x + 1)].
\]

13. **GREATEST COMMON DIVISOR AND LEAST COMMON MULTIPLE**

The greatest common divisor (g.c.d.) of the integers \( x \) and \( y \) is the largest integer \( d \) which divides both integers, denoted \( d = \gcd(x, y) \). The g.c.d. exists if at least one of the integers \( x \) and \( y \) is \( 0 \). Note that the g.c.d. is positive. (It’s often agreed, however, that \( \gcd(0, 0) = 0 \).) If \( \gcd(x, y) = 1 \) then we say that \( x \) and \( y \) have no common divisors or that they are coprime.

**Theorem 3:** (Bézout’s theorem) - The g.c.d. \( d \) of the integers \( x \) and \( y \), at least one of which is \( 0 \), can be written in the form \( d = cx + cy \) (the so-called Bézout form) where \( c_1 \) and \( c_2 \) are integers, the so-called Bézout coefficients. Also, if \( x, y \neq 0 \), then we may assume that \( |c_1| \leq |y| \) and \( |c_2| \leq |x| \).

**Proof:** Bézout’s form and the g.c.d. \( d \) are produced by the following so-called (Generalized) Euclidean algorithm. Here we may assume that \( 0 \leq x \leq y \), without loss of generality.

Denote \( \gcd(x, y) = (d, c_1, c_2) \).
(Generalized) Euclidean algorithm.

1. If \( x = 0 \) then we come out of the algorithm with \( \text{GCD}(x, y) = (y, 0, 1) \) and quit.

2. If \( x > 0 \) then first we divide \( y \) with \( x \): \( y = qx + r \), where \( 0 \leq r < x \). Next we find \( \text{GCD}(r, x) = (d, e_1, e_2) \). Now \( d = e_1r + e_2x = e_1(y - qx) + e_2x = (e_2 - e_1q)x + e_1y \). We end the algorithm by returning \( \text{GCD}(x, y) = (d, e_2 - e_1q, e_1) \) and quit. Since \( r = y - qx \), \( \text{gcd}(x, y) \) divides \( r \) and hence \( \text{gcd}(x, y) \leq \text{gcd}(x, r) \). Similarly \( \text{gcd}(x, r) \) divides \( y \) and thus \( \text{gcd}(x, r) \leq \text{gcd}(x, y) \), so \( \text{gcd}(x, r) = \text{gcd}(x, y) \). Hence \#2 produces the correct result. The recursion ends after a finite number of iterations because \( \min(r, x) < \min(x, y) \), and so every time we call \( \text{GCD} \) (iterate) the minimum value gets smaller and is eventually = 0. If \( x, y \neq 0 \) then apparently right before stopping in \#1 in the recursion we have \( y = qx \) and \( r = 0 \) and \( d = x \), whence at that point \( c_1 = 1 \leq y \) and \( c_2 = 0 \leq x \). On the other hand, every time when in \#2 we have \( y = qx + r \) and \( d = e_1r + e_2x \), where \( |e_1| \leq x \) and \( |e_2| \leq r \), then \( e_1 \) and \( e_2 \) have opposite signs and thus \( |e_2 - e_1q| = |e_2| + |e_1|q \leq r + xq = y \). So, the new coefficients \( c_1 = e_2 - e_1q \) and \( c_2 = e_1 \) will then also satisfy the claimed conditions.

**Theorem 4:** Factorization of an integer \( x \neq 0 \) is unique.

**Proof:** Assume the contrary: There exists an integer \( x \), which has (at least) two different factorizations. We may assume that \( x \) is positive and that \( x \) is the smallest positive integer that has this property. Thus \( x \geq 2 \), since the only factorization of 1 is the empty product. Now we can write \( x \) as a product of primes with respect to two different factorizations:

\[
x = p_1^{i_1}p_2^{i_2} \cdots p_n^{i_n} = q_1^{j_1}q_2^{j_2} \cdots q_m^{j_m}
\]

where \( p_1, \ldots, p_n \) are different primes and likewise \( q_1, \ldots, q_m \) are different primes and \( i_1, \ldots, i_n \), as well as \( j_1, \ldots, j_m \) are positive integers. In fact, we also know that the primes \( p_1, \ldots, p_n \) differ from the primes \( q_1, \ldots, q_m \). If, for example, \( p_1 = q_1 \), then the integer \( x/p_1 \) would have two different factorizations and \( x/p_1 < x \), a contradiction. So we know that \( \text{gcd}(p_1, q_1) = 1 \), in Bézout’s form

\[
1 = c_1p_1 + c_2q_1.
\]

But it follows from this that \( q_1^{j_1-1}q_2^{j_2} \cdots q_m^{j_m} = (c_1p_1 + c_2q_1)q_1^{j_1-1}q_2^{j_2} \cdots q_m^{j_m} = c_1p_1q_1^{j_1-1}q_2^{j_2} \cdots q_m^{j_m} + c_2x, \ldots \) from which we see further that \( p_1 \) divides the product \( q_1^{j_1-1}q_2^{j_2} \cdots q_m^{j_m} \), in other words,

\[
q_1^{j_1-1}q_2^{j_2} \cdots q_m^{j_m} = p_1z.
\]

Because \( z \) and \( q_1^{j_1-1} q_2^{j_2} \cdots q_m^{j_m} \) have unique factorizations (they are both smaller than \( x \)), it follows from this that \( p_1 \) is one of the primes \( q_1, \ldots, q_m \) which is a contradiction. So the contrary is false and factorization is unique. When giving a rational number in the form \( x/y \), it is usually assumed that \( \text{gcd}(x, y) = 1 \), in other words, that the number is with the smallest terms. This is very important when calculating with large numbers, to prevent numerators and denominators from growing too large. Such reduced form is naturally obtained by dividing \( x \) and \( y \) by \( \text{gcd}(x, y) \), so in long calculations the g.c.d. must be determined repeatedly. It’s important to notice that the bounds of the coefficients mentioned in Bézout’s theorem, i.e. \( |c_1| \leq |y| \) and \( |c_2| \leq |x| \), are valid in every step of the Euclidean algorithm. This way intermediate results won’t get too large. On the other hand, the Euclidean algorithm does not even take too many steps.

**Theorem 5:** When computing \( \text{gcd}(x, y) \), where \( 0 \leq x \leq y \), the Euclidean algorithm needs no more than \( \lfloor 2 \log_2 y + 1 \rfloor \) divisions.
**Proof** :- If \( x = 0 \) (no divisions) or \( x = y \) (one division) there’s nothing to prove, so we can concentrate on the case \( 0 < x < y \). The proof is based on the following simple observation concerning division: Every time we divide integers \( a \) and \( b \), where \( 0 < a < b \), and write \( b = qa + r \) (quotient \( q \), remainder \( r \)), we have

\[
b = qa + r \geq a + r > 2r
\]

When computing \( \gcd(x, y) \) using the Euclidean algorithm we get a sequence

\[
y = q_1x + r_1 \quad (0 < r_1 < x),
x = q_2r_1 + r_2 \quad (0 < r_2 < r_1),
r_1 = q_3r_2 + r_3 \quad (0 < r_3 < r_2),

\]

with \( 1 + 1 \) divisions. ... If \( l = 2k + 1 \) is odd then by our observation above \( 1 \leq r_l < 2 \cdot 2^{-l} \cdot 2^{-l-2} < \cdots < 2^{-l} \cdot 2^{-i} \)

\[
l-2i < \cdots < 2^{-k-1} \cdot x = 2^{-l-1/2} < 2^{-l-1} \cdot y < 2^{-l-1/2},
\]

and if \( l = 2k \) is even then

\[
1 \leq r_l < 2^{-l-1} \cdot r_l-2 < 2^{-l-2} \cdot r_l-4 < \cdots < 2^{-k+1} \cdot r_2 < 2^{-k} \cdot x < 2^{-k+1} \cdot y < 2^{-k-1/2}.
\]

Thus we see that applying the Euclidean algorithm is not very laborious, \([2 \log_2 y + 1]\) is proportional to the length of the binary representation of \( y \) (Theorem 2). If you want to know more about the computational efficiency of the Euclidean algorithm, see e.g. KNUTH. The greatest common divisor of more than two integers \( x_1, x_2, \ldots, x_N \)

\[
d = \gcd(x_1, x_2, \ldots, x_n) \text{ is defined in same way as for two integers, so it’s the largest integer which divides all the numbers in the sequence } x_1, x_2, \ldots, x_N.
\]

Again we require that at least one of the numbers is \( \neq 0 \). We may agree that \( x_N \neq 0 \). This kind of g.c.d. can be computed by applying the Euclidean algorithm \( N - 1 \) times, since...

**RSA Cryptosystem**

Diffie-Hellman doesn’t do encryption; it just exchanges a key. RSA can encrypt and decrypt. Here’s how. ..Each personGenerates two large primes, \( p \) and \( q \).

Publishes his or her public key \((N,e)\) where \( N=pq \).

Computes \( d = \text{modular inverse of } e \text{ relative to } (p-1)(q-1) \), keeping it secret.

Destroys \( p \) and \( q \).

Now check this out:

For Alice to send a message \( m \) to Bob, she sends \( c=m^e \mod N \).

Bob decodes this easily because \( m=c^d \mod N \).

RSA algorithm is asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. **Public Key** and **Private Key**. As the name describes that the Public Key is given to everyone and Private key is kept private. **Let us learn the mechanism behind**
RSA algorithm

Generating Public Key:

Select two prime no's. Suppose \( P = 53 \) and \( Q = 59 \).
Now First part of the Public key : \( n = P \times Q = 3127 \).
We also need a small exponent say \( e \):
But \( e \) Must be
  - An integer.
  - Not be a factor of \( n \).

Generating Private Key:
We need to calculate \( \Phi(n) \):
Such that \( \Phi(n) = (P-1)(Q-1) \)
so, \( \Phi(n) = 3016 \)
Now calculate Private Key, \( d \):
\( d = (k \times \Phi(n) + 1) / e \) for some integer \( k \)
For \( k = 2 \), value of \( d \) is 2011.
Now we are ready with our – Public Key ( \( n = 3127 \) and \( e = 3 \)) and Private Key(\( d = 2011 \))

Now we will encrypt “HI”:
- Convert letters to numbers : \( H = 8 \) and \( I = 9 \)
- Thus Encrypted Data \( c = 89^e \mod n \).
  Thus our Encrypted Data comes out to be 1394
  Now we will decrypt 1394:
- Decrypted Data \( c^d \mod n \).
  Thus our Encrypted Data comes out to be 89
  \( 8 = H \) and \( I = 9 \) i.e. "HI".

14. DIGITAL SIGNATURES

A digital certificate is an electronic document issued by a Certificate Authority (CA). It contains the public key for a digital signature and specifies the identity associated with the key, such as the name of an organization. Digital signatures are the public-key primitives of message authentication. In the physical world, it is common to use handwritten signatures on handwritten or typed messages. They are used to bind signatory to the message…Similarly, a digital signature is a technique that binds a person/entity to the digital data. This binding can be independently verified by receiver as well as any third party.

Digital signature is a cryptographic value that is calculated from the data and a secret key known only by the signer….In real world, the receiver of message needs assurance that the message belongs to the sender and he should not be able to repudiate the origination of that message. This requirement is very crucial in business applications, since likelihood of a dispute over exchanged data is very high.

15. MODEL OF DIGITAL SIGNATURE

As mentioned earlier, the digital signature scheme is based on public key cryptography. The model of digital signature scheme is depicted in the following illustration
The following points explain the entire process in detail – Each person adopting this scheme has a public-private key pair…Generally, the key pairs used for encryption/decryption and signing/verifying are different. The private key used for signing is referred to as the signature key and the public key as the verification key.

**Signer feeds data to the hash function and generates hash of data.**

- Hash value and signature key are then fed to the signature algorithm which produces the digital signature on given hash. Signature is appended to the data and then both are sent to the verifier.
- Verifier feeds the digital signature and the verification key into the verification algorithm. The verification algorithm gives some value as output.
- Verifier also runs same hash function on received data to generate hash value.
- For verification, this hash value and output of verification algorithm are compared. Based on the comparison result, verifier decides whether the digital signature is valid.

Since digital signature is created by ‘private’ key of signer and no one else can have this key; the signer cannot repudiate signing the data in future. It should be noticed that instead of signing data directly by signing algorithm usually a hash of data is created. Since the hash of data is a unique representation of data, it is sufficient to sign the hash in place of data. The most important reason of using hash instead of data directly for signing is efficiency of the scheme. Let us assume RSA is used as the signing algorithm. As discussed in public key encryption chapter, the encryption/signing process using RSA involves modular exponentiation. Signing large data through modular exponentiation is computationally expensive and time consuming. The hash of the data is a relatively small digest of the data, hence **signing a hash is more efficient than signing the entire data.**

### 15.1 Importance of Digital Signature

Out of all cryptographic primitives, the digital signature using public key cryptography is considered as very important and useful tool to achieve information security. Apart from ability to provide non-repudiation of message, the digital signature also provides message authentication and data integrity. Let us briefly see how this is achieved by the digital signature –
**Message authentication**

When the verifier validates the digital signature using public key of a sender, he is assured that signature has been created only by sender who possess the corresponding secret private key and no one else.

**Data Integrity**

In case an attacker has access to the data and modifies it, the digital signature verification at receiver end fails. The hash of modified data and the output provided by the verification algorithm will not match. Hence, receiver can safely deny the message assuming that data integrity has been breached.

**Non-repudiation**

Since it is assumed that only the signer has the knowledge of the signature key, he can only create unique signature on a given data. Thus the receiver can present data and the digital signature to a third party as evidence if any dispute arises in the future. By adding public-key encryption to digital signature scheme, we can create a cryptosystem that can provide the four essential elements of security namely- Privacy, Authentication, Integrity, and Non-repudiation.

**16. Encryption with Digital Signature**

In many digital communications, it is desirable to exchange an encrypted messages than plaintext to achieve confidentiality. In public key encryption scheme, a public (encryption) key of sender is available in open domain, and hence anyone can spoof his identity and send any encrypted message to the receiver….This makes it essential for users employing PKC for encryption to seek digital signatures along with encrypted data to be assured of message authentication and non-repudiation. This can archived by combining digital signatures with encryption scheme. Let us briefly discuss how to achieve this requirement. There are two possibilities, sign-then-encrypt and encrypt-then-sign….However, the crypto system based on sign-then-encrypt can be exploited by receiver to spoof identity of sender and sent that data to third party. Hence, this method is not preferred. The process of encrypt-then-sign is more reliable and widely adopted.

**This is depicted in the following illustrations:**
The receiver after receiving the encrypted data and signature on it, first verifies the signature using sender’s public key. After ensuring the validity of the signature, he then retrieves the data through decryption using his private key.

RSA for Digital Signatures

- For Alice to send a message to Bob,
- Alice encrypts m with her private key
- Alice encrypts with Bob’s public key
- Bob decrypts with his private key
- Bob decrypts with Alice’s public key
- \( m = A'(B'(B(A'(m))) \)

Cryptography | One Time Password (OTP) algorithm

Authentication, the process of identifying and validating an individual is the rudimentary step before granting access to any protected service (such as a personal account). Authentication has been built into the cyber security standards and offers to prevent unauthorized access to safeguarded resources. Authentication mechanisms today create a double layer gateway prior to unlocking any protected information. This double layer of security, termed as two factor authentication, creates a pathway that requires validation of credentials (username/email and password) followed by creation and validation of the One Time Password (OTP). The OTP is a numeric code that is randomly and uniquely generated during each authentication event. This adds an additional layer of security, as the password generated is fresh set of digits each time an authentication is attempted and it offers the quality of being unpredictable for the next created session.

The two main methods for delivery of the OTP is

1. SMS-Based:

This is quite straightforward. It is the standard procedure for delivering the OTP via a text message after regular authentication is successful. Here, the OTP is generated on the server side and delivered to the authenticator via text message. It is the most common method of OTP delivery that is encountered across services.

2. Application-Based: This method of OTP generation is done on the user side using a specific smartphone application that scans a QR code on the screen. The application is responsible for the unique OTP digits. This reduces wait time for the OTP as well as reduces security risk as compared to the SMS based delivery. The most common way for the generation of OTP defined by The Initiative For Open Authentication (OATH) is the Time Based One Time Password (TOTP), which is a Time Synchronized OTP. In these OTP systems, time is the cardinal factor to generate the unique password. The password generated is created using the current time and it also factors in a secret key. An example of this OTP generation is the Time Based OTP Algorithm (TOTP) described as follows:

Backend server generates the secret key. The server shares secret key with the service generating the OTP.
A hash based message authentication code (HMAC) is generated using the obtained secret key and time. This is done using the cryptographic SHA-1 algorithm. Since both the server and the device requesting the OTP have access to time, which is obviously dynamic, it is taken as a parameter in the algorithm. Here, the Unix timestamp is considered which is independent of time zone i.e. time is calculated in seconds starting from January First 1970. Let us consider “0215a7d8c15b492eb2116482b6d34fe4e1a9f6ba” as the generated string from the HMAC-SHA 1 algorithm. The code generated is 20 bytes long and is thus truncated to the desired length suitable for the user to enter. Here dynamic truncation is used. For the 20-byte code “0215a7d8c15b492eb2116482b6d34fe4e1a9f6ba”, each character occupies 4 bits.

The entire string is taken as 20 individual one byte sting.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 1 | 1 | 1 |
| 12 | 15 | a7 | d8 | c1 | 5b | 49 | 2e | 21 | 11 | 64 | 82 | b6 | d3 | 4f | c4 | el | a | f | b | 6 | a |

We look at the last character, here a. The decimal value of which is taken to determine the offset from which to begin truncation. Starting from the offset value, 10 the next 31 bits are read to obtain the string “6482b6d3”. The last thing left to do, is to take our hexadecimal numerical value, and convert it to decimal, which gives 1686288083.

All we need now are the last desired length of OTP digits of the obtained decimal string, zero-padded if necessary. This is easily accomplished by taking the decimal string, modulo 10 ^ number of digits required in OTP. We end up with “288083” as our TOTP code… A counter is used to keep track of the time elapsed and generate a new code after a set interval of time. Then OTP generated is delivered to user by the methods described above. Apart from the time-based method described above, there also exist certain mathematical algorithms for OTP generation for example a one-way function that creates a subsequent OTP from the previously created OTP. The two factor authentication system is an effective strategy that exploits the authentication principles of “something that you know” and “something that you have”. The dynamic nature of the latter principle implemented by the One Time Password Algorithm is crucial to security and offers an effective layer of protection against malicious attackers. The unpredictability of the OTP presents a hindrance in peeling off the layers that this method of cryptography has to offer.

**Computer Network | Password authentication protocol (PAP)**

There are simply two methods to authenticate PPP links namely Password Authentication Protocol (PAP) and Challenge Handshake Authentication Protocol (CHAP).

From these two authentication protocols, PAP is less secured as the password is sent in clear text and is performed only at the initial link establishment. Password-Authentication-Protocol-(PAP) – PAP is a password Authentication Protocol used by PPP links to validate users. PAP authentication requires the calling device to enter the username and password. If the credentials match with the local database of the called device or in the remote AAA database then it is allowed to access otherwise denied.
Features

Some of the features of PAP are:

- The password is sent in clear text.
- All network operating system support PAP.
- It uses two-way Handshake Protocol.
- It is non-interactive.
- PAP supports both one-way authentication (unidirectional) and two-way authentication (bidirectional).

Configuration

There is a small topology in which there are 2 routers namely R1 and R2. R1 having ip address 10.1.1.1/30 on s0/0 and R2 have ip address 10.1.1.2/30 on s0/0.

1) First, we will create local database on R1 by providing username and password:
   R1(config)#username Router1 password GeeksforGeeks

2) Configuring local database on R2:
   R2(config)#username Router2 password GeeksforGeeks

3) Remember, by default HDLC is configured on Cisco routers therefore first we have to change the encapsulation to PPP and enable PAP.
   R2(config)# int s0/0
   R2(config-if)#encapsulation ppp
   R2(config-if)#ppp authentication pap
   R2(config-if)#ppp pap sent-username Router1 password GeeksforGeeks

4) Enabling PAP on R2:
   R2(config)# int s0/0
   R2(config-if)#encapsulation ppp
   R2(config-if)#ppp authentication pap
   R2(config-if)#ppp pap sent-username Router1 password GeeksforGeeks

Here, notice that username and password are case-sensitive. Also, on router R1 we have to give a username and password.

When to use PAP – PAP is usually used in following scenarios:

- When the application doesn’t support CHAP.
- Circumstances where it is necessary to sent a plain text password to stimulate a login at the called device (remote host).
- When there is occurrence of incompatibilities between different vendors of CHAP.
Advantage of CHAP over PAP

Some of the advantages are:

- CHAP is more secured than PAP.
- CHAP can provide authentication periodically to recognise that the user accessing the PPP link is same or not.
- In CHAP, the real passwords are never shared on the link instead a hash value of it is calculated and transferred.

Advantage of PAP over CHAP

The only advantage PAP holds over CHAP is that it is supported by the all the network operating system vendors therefore it can be said that PAP is used where CHAP is not supported. But if CHAP is supported then it is recommended to use CHAP as it is more secured…..

REFERENCES

[3] Cryptography’s Past, Present, and Future Role in Society – By-Franck Lin