19AD713 **BIG DATA SECURITY**

Security based analysis and Implementation on a dataset [Global Temperatures]

WORK GUIDED BY :

Dr. HARINI

WORK DONE BY:

RAAKESH . M

CB.EN.P2AID19023

Introduction:

Big data security is the collective term for all the measures and tools used to guard both the data and analytics processes from attacks, theft, or other malicious activities that could harm or negatively affect them. Much like other forms of cyber-security, the big data variant is concerned with attacks that originate either from the online or offline spheres. The security implementation in big data is now becoming a necessary precautionary measure so that the data theft and cyber misuse can be prevented by being safe when the data transfer between two users takes place.

Current big data security measures involves;

- Hadoop security - security and privacy of hadoop, HDFS security
- Cloud security - secured data storage platform (believed to be)
- Monitoring - intrusion detection architecture, detect anomalies or the behavioural pattern
- Auditing - detect abnormal behaviour of the user and act as storage [Audition Dynamic Big Data Storage]
- Key Management - generation of strong keys and authentication of data centers, secure group key transfer, secure group data transfer, unstructured data security
- Data Security (Anonymization) - scalable privacy preservation of the big data , data mining privacy preservation and sensitive field Anonymization

So the data security measures are involved in all fields of work that we are involved from day till dusk.

The reason for the usage of big data is

- ➢ The big data benefits and expectations being high (works for real time models)
- ➢ Most commonly used data among the companies and organisations
- ➢ Analysis of a complete flow of work with the risk done easy

The analysis and implementation of security towards a sensitive data before transfer to prevent the data misuse in case of a global data made available

Current state of data is much better compared to when the process of data sharing and cloud was formed initially there was a large volume of data that was dumped wherever possible which made the hackers connect to the dump and steal the data and also allowed them free reign in analysing the sensitivity involved .Later as the process of refinement the storages of data were made secure with a lot of algorithms ,keys, anonymous passcodes etc

Breach :

Lot of climate data is being analysed and processed but in relation to security there are no measures taken which have been found through the climategate scandal where the climatic data of a university was being stolen.

Incident: The **Climatic Research Unit email controversy** (also known as "**Climategate**") began in November 2009 with the hacking of a server at the Climatic Research Unit (CRU) at the University of East Anglia (UEA) by an external attacker,copying thousands of emails and computer files (the Climatic Research Unit documents) to various internet locations several weeks before the Copenhagen Summit on climate change.

We may think it is simple and there can be no damages caused by it , but if there is a breach the regular monitoring may get affected and any emergency information can be blocked or miscommunicated. Hence the security requirement .

Impact:

The impact of climate change is not restricted to a single domain , it also involves a lot of other domains and has an involvement in their changes or sometimes indirectly has a share in the damages.The other domains include Healthcare,Telecom services, Traffic control, Train-Delay, etc which are directly or indirectly involved with change in climate .

Problem statement :

Prevention of unwanted/ sensitive data being shared from the global temperatures dataset that could be used to determine and detect the area sensitive to be shared.

[Understanding of the relationship among the data,Sensitive data identification, Data analysis, False Data Analysis,Key allotment,Encryption and Decryption (comparative analysis), Hashing, Digital signatures]

For example some patterns show the occurrence of a landslide and the possibility of its occurrence pattern for the next few decades which in the hands of bad people may cause a huge disaster. Such was the occurrence of the global temperature results causing the sea level increase and greenhouse gas emission reports being shared in public domains to cause a panic among the people of the world.

We usually find a lot of unpredictable things in our day to day life especially in case of weather and climate. Lot of unpredictables are now being processed and predicted to prevent massive surges or in case of this dataset sudden rise in land ocean temperature causing the rise of sea level or glacial melting in the poles.

Temperature is something that is common to all but here the bearable heat of land is considered as the temperature range along with the maximum and minimum values (in centigrade).the record starts from 1750 the period of renaissance till 2015 with update every month.

This data is used for various purposes like .,

- Comparing the similarity in the records to check for relativity in occurances
- Prediction analysis
- Land/ocean temperature increase/decrease
- Uncertainty level
- The global warming in relation to this analysis

The implementation of security in this case is to prevent the knowledge of some data becoming common and causing a panic among the people.

For example some patterns show the occurrence of a landslide and the possibility of its occurrence pattern for the next few decades which in the hands of bad people may cause a huge disaster. Such was the occurrence of the global temperature results causing the sea level increase and greenhouse gas emission reports being shared in public domains to cause a panic among the people of the world.

Dataset analysis :

Global Land and Ocean-and-Land Temperatures (**GlobalTemperatures.csv**):

- Date: starts in 1750 for average land temperature and 1850 for max and min land temperatures and global ocean and land temperatures
- LandAverageTemperature: global average land temperature in celsius
- LandAverageTemperatureUncertainty: the 95% confidence interval around the average
- LandMaxTemperature: global average maximum land temperature in celsius
- LandMaxTemperatureUncertainty: the 95% confidence interval around the maximum land temperature
- LandMinTemperature: global average minimum land temperature in celsius
- LandMinTemperatureUncertainty: the 95% confidence interval around the minimum land temperature
- LandAndOceanAverageTemperature: global average land and ocean temperature in celsius
- LandAndOceanAverageTemperatureUncertainty: the 95% confidence interval around the global average land and ocean temperature

more than 2 datasets of similar origin is also taken into account such as .,

- Global Land Temperatures By Major City (**GlobalLandTemperaturesByMajorCity.csv**)
- Global Average Land Temperature by Country (**GlobalLandTemperaturesByCountry.csv**)

Confidentiality :

Confidentiality is the prevention of data being accessed by unauthorized parties. Data accessibility being denied to unauthorized personnel

Plain Text - The plain text message is the text which is readable and can be understood by all users. The plain text is the message which undergoes cryptography.

Cipher Text - Cipher text is the message obtained after applying cryptography on plain text.

Encryption- The process of converting plaintext to cipher text is called encryption. It is also called encoding.

Decryption-The process of converting ciphertext to plain text is called decryption. It is also termed as decoding.

Integrity :

Integrity refers to ensuring the authenticity of information—that information is not altered, and that the source of the information is genuine. An example of a website selling home products suddenly increasing prices of products once we log in or us being redirected to some malicious sites is the loss of Integrity.

- Reliability and relationship between the data
- Comparative analysis of the data
- Predictive analysis of the data and its sensitivity

Authentication:

Data authentication is the process of confirming the origin and integrity of data. The term is typically related to communication, messaging and integration. Data authentication has two elements: authenticating that you're getting data from the correct entity and validating the integrity of that data.

Confidentiality:

When implementing confidentiality only the sender and the receiver should be able to read the message, that means the contents have to be kept secret from every other person, except for those two.

Diffie-Hellman Algorithm:

Step 1: Alice and Bob get public numbers P = 23, G = 9

Step 2: Alice selected a private key a = 4 and Bob selected a private key b = 3

Step 3: Alice and Bob compute public values

Alice: x =(9^4 mod 23) = (6561 mod 23) = 6

Bob: y = (9^3 mod 23) = (729 mod 23) = 16

Step 4: Alice and Bob exchange public numbers

Step 5: Alice receives public key y =16 and Bob receives public key x = 6

Step 6: Alice and Bob compute symmetric keys

Alice: ka = y^a mod p = 65536 mod 23 = 9

Bob: kb = x^b mod p = 216 mod 23 = 9

Step 7: 9 is the shared secret.

RSA:

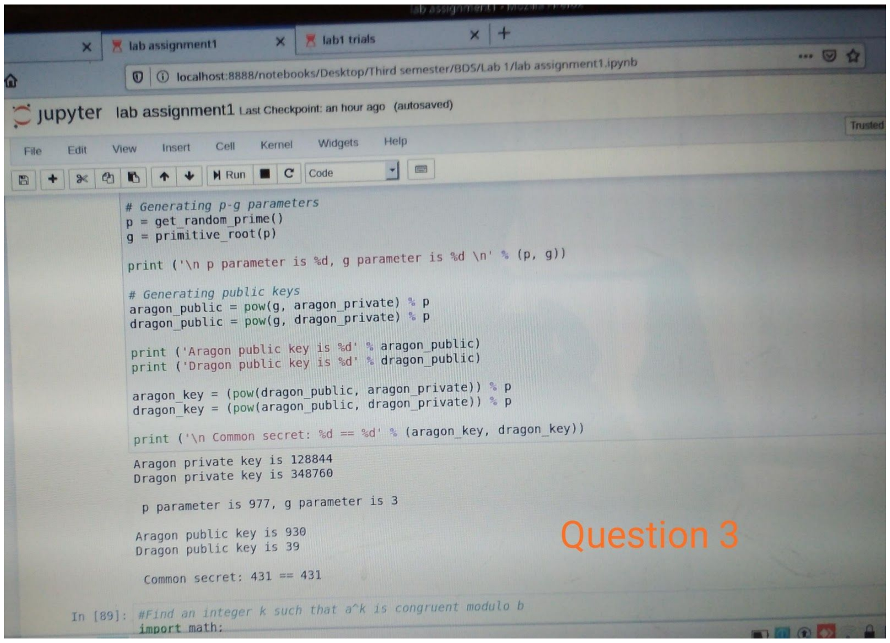
RSA algorithm is an asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. **Public Key** and **Private Key.** As the name describes that the Public Key is given to everyone and the Private key is kept private.

1. A client (for example browser) sends its public key to the server and requests for some data.
2. The server encrypts the data using the client's public key and sends the encrypted data.
3. Client receives this data and decrypts it.

4. Since this is asymmetric, nobody else except the browser can decrypt the data even if a third party has the public key of the browser.

RSA key generation:

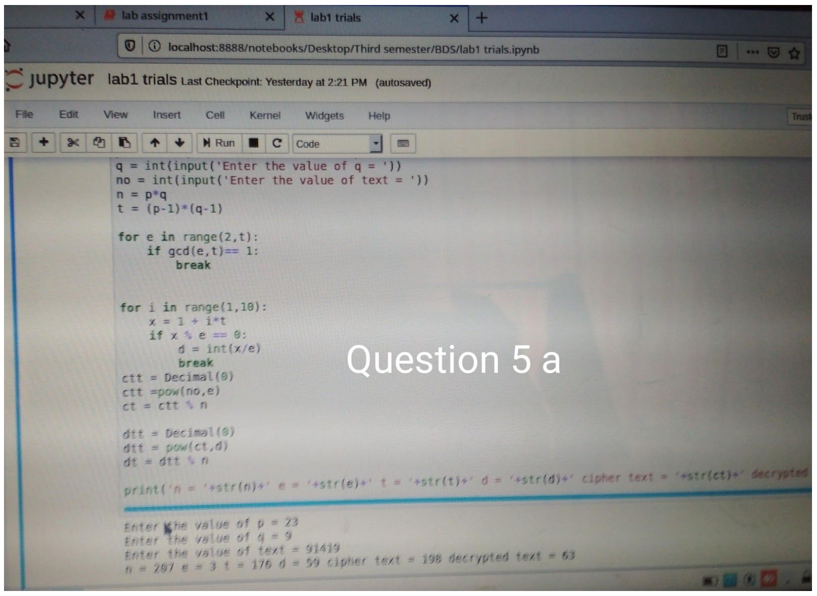
- Choose two large prime numbers p and q
- Calculate n=p*q
- Select public key e such that it is not a factor of (p-1)*(q-1)
- Select private key d such that the following equation is true (d*e)mod(p-1)(q-1)=1 or d is inverse of E in modulo (p-1)*(q-1)


lab assignment1
lab1 trials
localhost:8888/notebooks/Desktop/Third semester/BDS/Lab 1/lab assignment1.ipynb
jupyter lab assignment1 Last Checkpoint: an hour ago (autosaved)
Trusted
File Edit View Insert Cell Kernel Widgets Help
Run
Code
Question 3

```
# Generating p-g parameters
p = get_random_prime()
g = primitive_root(p)

print ('\n p parameter is %d, g parameter is %d \n' % (p, g))

# Generating public keys
aragon_public = pow(g, aragon_private) % p
dragon_public = pow(g, dragon_private) % p

print ('Aragon public key is %d' % aragon_public)
print ('Dragon public key is %d' % dragon_public)

aragon_key = (pow(dragon_public, aragon_private)) % p
dragon_key = (pow(aragon_public, dragon_private)) % p

print ('\n Common secret: %d == %d' % (aragon_key, dragon_key))
```

```
Aragon private key is 128844
Dragon private key is 348760

 p parameter is 977, g parameter is 3

Aragon public key is 930
Dragon public key is 39

 Common secret: 431 == 431
```

```
In [89]: #Find an integer k such that a^k is congruent modulo b
         import math;
```


lab assignment1
lab1 trials
localhost:8888/notebooks/Desktop/Third semester/BDS/lab1 trials.ipynb
jupyter lab1 trials Last Checkpoint: Yesterday at 2:21 PM (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run
Code
Question 5 a

```
q = int(input('Enter the value of q = '))
no = int(input('Enter the value of text = '))
n = p*q
t = (p-1)*(q-1)

for e in range(2,t):
    if gcd(e,t)== 1:
        break


for i in range(1,10):
    x = 1 + i*t
    if x % e == 0:
        d = int(x/e)
        break
ctt = Decimal(0)
ctt =pow(no,e)
ct = ctt % n

dtt = Decimal(0)
dtt = pow(ct,d)
dt = dtt % n

print('n = '+str(n)+' e = '+str(e)+' t = '+str(t)+' d = '+str(d)+' cipher text = '+str(ct)+' decrypted
```

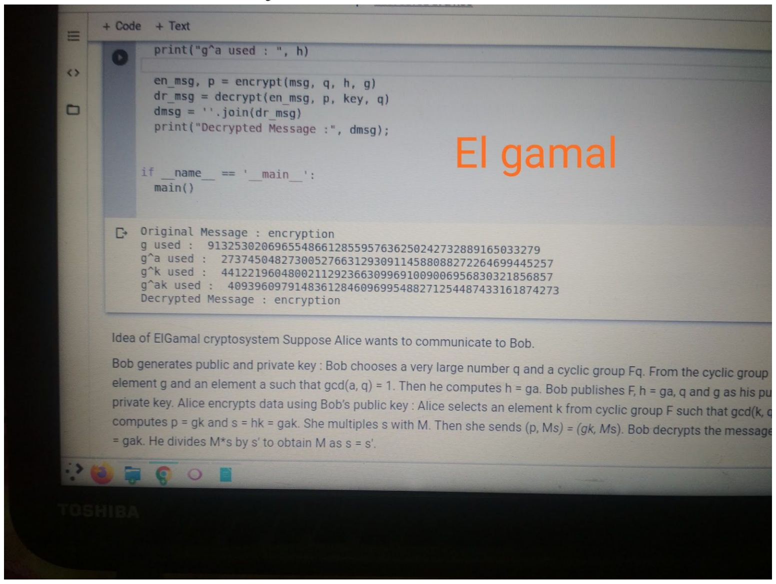
```
Enter the value of p = 23
Enter the value of q = 9
Enter the value of text = 91419
n = 207 e = 3 t = 176 d = 59 cipher text = 198 decrypted text = 63
```

ElGamal encryption is a public-key cryptosystem. It uses asymmetric key encryption for communicating between two parties and encrypting the message.This cryptosystem is based on the difficulty of finding **discrete logarithm** in a cyclic group that is even if we know ga and gk, it is extremely difficult to compute gak.

Idea of ElGamal cryptosystem

Suppose Alice wants to communicate to Bob.

1. Bob generates public and private key :
    - Bob chooses a very large number **q** and a cyclic group **F**q.
    - From the cyclic group **F**q, he choose any element **g** and an element **a** such that gcd(a, q) = 1.
    - Then he computes h = ga.
    - Bob publishes **F**, **h = ga**, **q** and **g** as his public key and retains **a** as private key.
2. Alice encrypts data using Bob's public key :
    - Alice selects an element **k** from cyclic group **F** such that gcd(k, q) = 1.
    - Then she computes p = gk and s = hk = gak.
    - She multiples s with M.
    - Then she sends (p, M*s) = (gk, M*s).

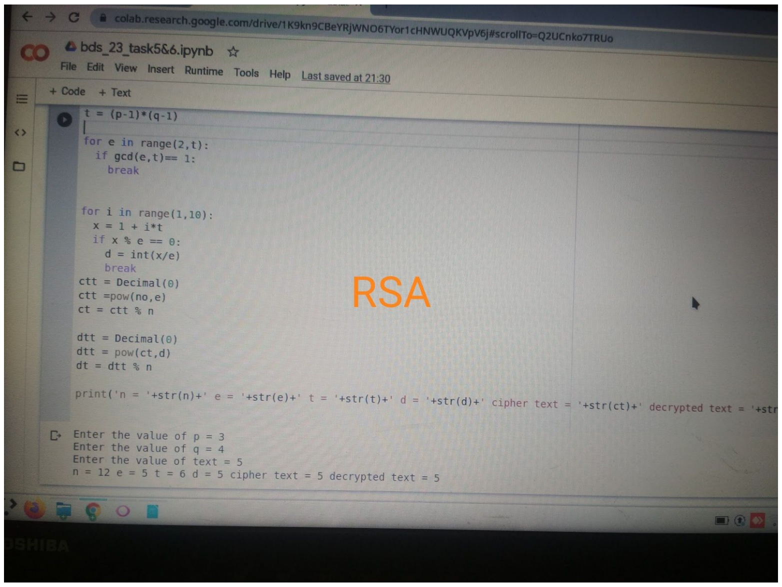
3. Bob decrypts the message :
    - Bob calculates s′ = pa = gak.
    - He divides M*s by s′ to obtain M as s = s′.
    - 

```
    print("g^a used : ", h)

    en_msg, p = encrypt(msg, q, h, g)
    dr_msg = decrypt(en_msg, p, key, q)
    dmsg = ''.join(dr_msg)
    print("Decrypted Message :", dmsg);


if __name__ == '__main__':
    main()
```

```
Original Message : encryption
g used :  91325302069655486612855957636250242732889165033279
g^a used :  27374504827300527663129309114588088272264699445257
g^k used :  44122196048002112923663099691009006956830321856857
g^ak used :  40939609791483612846096995488271254487433161874273
Decrypted Message : encryption
```

Idea of ElGamal cryptosystem Suppose Alice wants to communicate to Bob.

Bob generates public and private key : Bob chooses a very large number q and a cyclic group Fq. From the cyclic group element g and an element a such that gcd(a, q) = 1. Then he computes h = ga. Bob publishes F, h = ga, q and g as his pu private key. Alice encrypts data using Bob's public key : Alice selects an element k from cyclic group F such that gcd(k, c computes p = gk and s = hk = gak. She multiples s with M. Then she sends (p, Ms) = (gk, Ms). Bob decrypts the message = gak. He divides M*s by s' to obtain M as s = s'.

    - 
colab.research.google.com/drive/1K9kn9CBeYRjWNO6TYor1cHNWUQKVpV6j#scrollTo=Q2UCnko7TRUo
bds_23_task5&6.ipynb
File Edit View Insert Runtime Tools Help Last saved at 21:30
+ Code
+ Text
RSA

```
t = (p-1)*(q-1)

for e in range(2,t):
  if gcd(e,t)== 1:
    break


for i in range(1,10):
  x = 1 + i*t
  if x % e == 0:
    d = int(x/e)
    break
ctt = Decimal(0)
ctt =pow(no,e)
ct = ctt % n

dtt = Decimal(0)
dtt = pow(ct,d)
dt = dtt % n

print('n = '+str(n)+' e = '+str(e)+' t = '+str(t)+' d = '+str(d)+' cipher text = '+str(ct)+' decrypted text = '+str
```

```
Enter the value of p = 3
Enter the value of q = 4
Enter the value of text = 5
n = 12 e = 5 t = 6 d = 5 cipher text = 5 decrypted text = 5
```

Integrity:

Integrity models are used to describe what needs to be done to enforce the information integrity policy. There are three goals of integrity, which the models address in various ways:

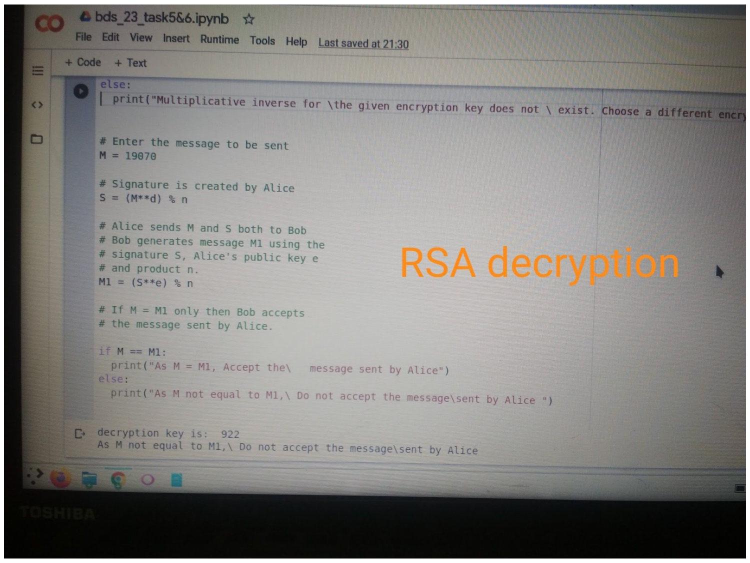
1. Preventing unauthorized users from making modifications to data or programs.
2. Preventing authorized users from making improper or unauthorized modifications.
3. Maintaining internal and external consistency of data and programs.

Digital signatures are used to verify the authenticity of the message sent electronically. A digital signature algorithm uses a public key system. The intended transmitter signs his/her message with his/her private key and the intended receiver verifies it with the transmitter's public key. A digital signature can provide message authentication, message integrity and non-repudiation services.

Algorithms:

RSA Digital Signature Scheme: In RSA, d is private; e and n are public.

- Alice creates her digital signature using S=M^d mod n where M is the message
- Alice sends Message M and Signature S to Bob
- Bob computes M1=S^e mod nIf M1=M then Bob accepts the data sent by Alice.


bds_23_task5&6.ipynb
File Edit View Insert Runtime Tools Help Last saved at 21:30
+ Code + Text
RSA decryption

```
else:
  print("Multiplicative inverse for \the given encryption key does not \ exist. Choose a different encry


# Enter the message to be sent
M = 19070

# Signature is created by Alice
S = (M**d) % n

# Alice sends M and S both to Bob
# Bob generates message M1 using the
# signature S, Alice's public key e
# and product n.
M1 = (S**e) % n

# If M = M1 only then Bob accepts
# the message sent by Alice.

if M == M1:
  print("As M = M1, Accept the\   message sent by Alice")
else:
  print("As M not equal to M1,\ Do not accept the message\sent by Alice ")
```

```
decryption key is:  922
As M not equal to M1,\ Do not accept the message\sent by Alice
```

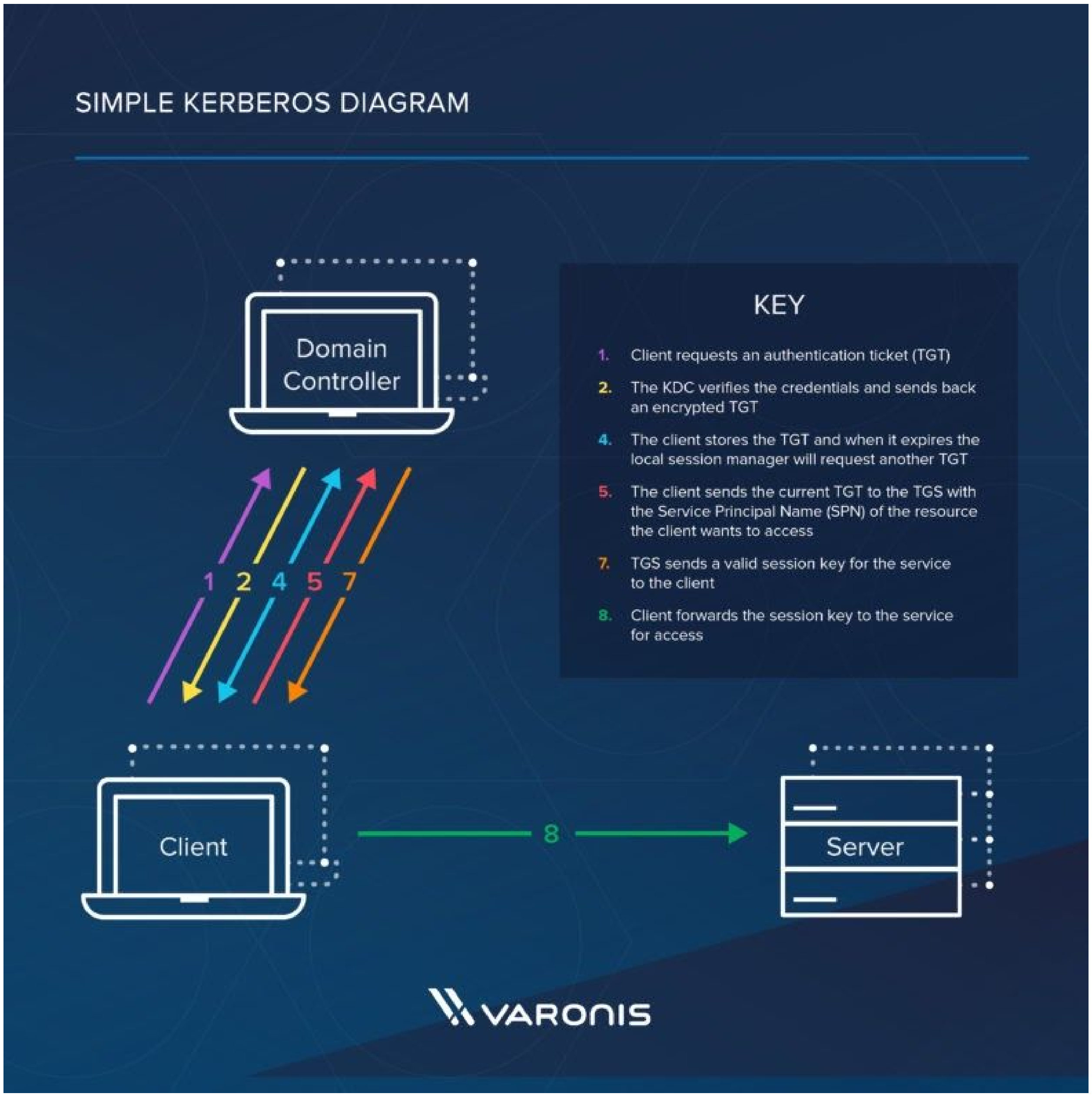
Authentication and Authorization :

Kerberos model 5:

Kerberos V5 is based on the Kerberos authentication system developed at MIT. Under Kerberos, a client (generally either a user or a service) sends a request for a ticket to the Key Distribution Center (KDC). The KDC creates a ticket-granting ticket (TGT) for the client, encrypts it using the client's password as the key, and sends the encrypted TGT back to the client. The client then attempts to decrypt the TGT, using its password. If the client successfully decrypts the TGT (*i.e.*, if the client gave the correct password), it keeps the decrypted TGT, which indicates proof of the client's identity.

The TGT, which expires at a specified time, permits the client to obtain additional tickets, which give permission for specific services. The requesting and granting of these additional tickets is user-transparent.

Kerberos is a vast improvement on previous authorization technologies. The strong cryptography and third-party ticket authorization make it much more difficult for cybercriminals to infiltrate your network. It is not totally without flaws, and in order to defend against those flaws, you need to first understand them.Kerberos has made the internet and its denizens more secure, and enables users to do more work on the Internet and in the office without compromising safety.



KEY

1. Client requests an authentication ticket (TGT)
2. The KDC verifies the credentials and sends back an encrypted TGT
4. The client stores the TGT and when it expires the local session manager will request another TGT
5. The client sends the current TGT to the TGS with the Service Principal Name (SPN) of the resource the client wants to access
7. TGS sends a valid session key for the service to the client
8. Client forwards the session key to the service for access

steps taken to authenticate in a Kerberized environment.

1. Client requests an authentication ticket (TGT) from the Key Distribution Center (KDC)
2. The KDC verifies the credentials and sends back an encrypted TGT and session key
3. The TGT is encrypted using the Ticket Granting Service (TGS) secret key
4. The client stores the TGT and when it expires the local session manager will request another TGT (this process is transparent to the user)

If the Client is requesting access to a service or other resource on the network, this is the process:

5. The client sends the current TGT to the TGS with the Service Principal Name (SPN) of the resource the client wants to access
6. The KDC verifies the TGT of the user and that the user has access to the service
7. TGS sends a valid session key for the service to the client
8. Client forwards the session key to the service to prove the user has access, and the service grants access.

Verification :

Use the function krb5_verify_init_creds() to verify initial credentials. It takes an options structure (which can be a null pointer). Use krb5_verify_init_creds_opt_init() to initialize the caller-allocated options structure, and krb5_verify_init_creds_opt_set_ap_req_nofail() to set the "nofail" option.

References:

https://www.rd-alliance.org/group/big-data-ig-data-security-and-trust-wg/wiki/big-data-security-issues-challenges-tech-concerns

https://docs.microsoft.com/en-us/windows-server/security/kerberos/kerberos-authentication-overview

https://www.varonis.com/blog/kerberos-authentication-explained/

https://web.mit.edu/kerberos/krb5-current/doc/appdev/init_creds.html

https://www.vanimpe.eu/2017/05/26/kerberos-made-easy/