TWEAKED BLOCK CHAIN WITH RSA AND AES

Name:- PRATEEK KUMAR
Roll No:-2019B010
College:- IMA,Andharua,Bhubaneswar.
Year:- 2019-22

Institute of Mathematics & Applications
Andharua, Bhubaneswar- 751 029
Odisha

TWEAKED BLOCK CHAIN WITH RSA AND AES

A Project Report submitted to the

Institute of Mathematics & Applications, Bhubaneswar

in partial fulfilment of the requirements

of the degree of

B. Sc. (Hons)

in

Mathematics & Computing

by

Prateek Kumar.

(University Roll No.- 7473U196014)

under the supervision of

Dr. Debasis Gountia

Department of Computer Science & Engineering,

B.Tech(UCE Burla),M.Tech(IIT Kharagpur),Ph.D(IIT Roorkee)

MIEEE,MICSES,LMFSIESRP,PMIFERP

Associate Professor, PIC-IT , Odisha University of Technology & Research,
Techno Campus,Kalinga Nagar,Bhubaneswar,Odisha,India-751029

Institute of Mathematics & Applications
Andharua, Bhubaneswar -751 029
Odisha.

Institute of Mathematics & Applications
Andharua, Bhubaneswar- 751029, Odisha.

Dr. Debasis Gountia
Department of Computer Science Engineering, B.Tech(UCE Burla),M.Tech(IIT Kharagpur),Ph.D(IIT Roorkee) MIEEE,MICSES,LMFSIESRP,PMIFERP Associate Professor, PIC-IT

08/06/22

Supervisor’s Certificate

This is to certify that the work presented in this project entitled *“TWEAKED BLOCK CHAIN WITH RSA AND AES*" by *Prateek Kumar*, University Roll Number:- 7473U196014, is a record of review work carried out by him under my supervision and guidance in partial fulfilment of the requirements of the B. Sc. (Hons) in Mathematics and Computing.

08/06/2022

- (Dr. Debasis Gountia)
Associate Professor,
OUTR, BBSR.

Dr. Debasis Gountia.

Dedicated

to Family of IMA.

Declaration

I, *Prateek Kumar*, Roll Number:-2019B010 hereby declare that this project entitled *“TWEAKED BLOCK CHAIN WITH RSA AND AES*" represents my review work carried out as a B. Sc. (Hons) of IMA Bhubaneswar and, to the best of my knowledge, it is not a complete copy of previously published or written by another person, nor any material presented for award of any other degree or diploma of IMA Bhubaneswar or any other institution. Any contribution made to this research by others, with whom I have worked at IMA Bhubaneswar or elsewhere, is explicitly acknowledged in the dissertation.

08/06/22 **Prateek Kumar.**

Acknowledgment

I express my deep sense of gratitude to all those who have been instrumental in preparation of this project.

I thankful to Dr.Debasis Gountia,Associate Professor,OUTR,BBSR for his constant support and encouragement.

I acknowledge the kind of support,efforts and timely guidance provided by Debasis sir along with his associates.This project report helps in better understanding of the subject matter. I couldn’t forget the hard work and curiosity which provided me with this productive result.

Date- 08/06/22
IMA Bhubaneswar

Prateek Kumar
Roll N0- 2019B010

Abstract

In this paper,an approach to encryption algorithms along with Block-Chain has been proposed.There are two types of encryption i.e. Symmetric and Asymmetric.The process of converting ordinary and plain text into unintelligible text and vice versa is known as cryptography. In this method, data is stored and transmitted in a specific form in order to make it available for only particular people to read and process. Its importance lies in the fact that it protects data from hacking and alteration, while making it useful for user authentication.Here you will get to know the information about the types of encryption and also the simulation of algorithms using **Python code** .Also you will get to know about **Block-Chain** inclusive of encryption algorithms i.e **TBC(Tweaked Block-Chain.)**

Keywords:Cryptography,Algorithms,RSA,AES,Block-Chain,TBC

Contents

Chapter 1

Introduction

Cryptography is the process of encrypting messages and other data in order to transmit them in a form that can only be accessed by the intended recipients. It was initially applied to written messages. With the introduction of modern computers, cryptography became an important tool for securing many types of digital data.

WHAT IS CRYPTOGRAPHY?

The word “cryptography” comes from the Greek words kryptos (“hidden,” “secret”) and graphein (“writing”). Early cryptography focused on ensuring that written messages could be sent to their intended recipients without being intercepted and read by other parties. This was achieved through various encryption techniques. Encryption is based on a simple principle: the message is transformed in such a way that it becomes unreadable. The encrypted message is then transmitted to the recipient, who reads it by transforming (decrypting) it back into its original form.

CRYPTOGRAPHY IN THE COMPUTER AGE: -

With the introduction of digital computers, the focus of cryptography shifted from just written language to any data that could be expressed in binary format. The encryption of binary data is accomplished through the use of keys. A key is a string of data that determines the output of a cryptographic algorithm. While there are many different types of cryptographic algorithms, they are usually divided into two categories. Symmetric-key cryptography uses a single key to both encrypt and decrypt the data. Public-key cryptography, also called “asymmetric-key cryptography,” uses two keys, one public and one private. Usually, the public key is used to encrypt the data, and the private key is used to decrypt it.

When using symmetric-key cryptography, both the sender and the recipient of the encrypted message must have access to the same key. This key must be exchanged between parties using a secure channel, or else it may be compromised. Public-key cryptography does not require such an exchange. This is one reason that public-key cryptography is considered more secure.

Another cryptographic technique developed for use with computers is the digital signature. A **digital signature** is used to confirm the identity of the sender of a digital message and to ensure that no one has tampered with its contents. Digital signatures use public-key encryption. First, a hash function is used to compute a unique value based on the data contained in the message. This unique value is called a “message digest,” or just “digest.” The signer’s private key is then used to encrypt the digest. The combination of the digest and the private key creates the signature. To verify the digital signature, the recipient uses the signer’s public key to decrypt the digest. The same hash function is then applied to the data in the message. If the new digest matches the decrypted digest, the message is intact.

WHY IS CRYPTOGRAPHY IMPORTANT?

The ability to secure communications against interception and decryption has long been an important part of military and international affairs. In the modern age, the development of new computer-based methods of encryption has had a major impact on many areas of society, including law enforcement, international affairs, military strategy, and business. It has also led to widespread debate over how to balance the privacy rights of organizations and individuals with the needs of law enforcement and government agencies. Businesses, governments, and consumers must deal with the challenges of securing digital communications for commerce and banking on a daily basis. The impact of cryptography on society is likely to increase as computers grow more powerful, cryptographic techniques improve, and digital technologies become ever more important.

Chapter 2

RSA

What is the RSA algorithm?

The **RSA algorithm** is an asymmetric cryptography algorithm; this means that it uses a public key and a private key (i.e two different, mathematically linked keys). As their names suggest, a public key is shared publicly, while a private key is secret and must not be shared with anyone.

The RSA algorithm is named after those who invented it in 1978: Ron Rivest, Adi Shamir, and Leonard Adleman.

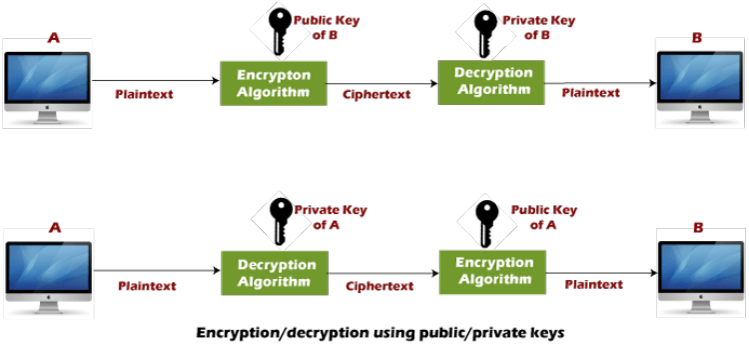


Figure 2.1: RSA Encryption Algorithm

Where is RSA encryption used?

RSA encryption is often used in combination with other encryption schemes, or for digital signatures which can prove the authenticity and integrity of a message. It isn't generally used to encrypt entire messages or files, because it is less efficient and more resource-heavy than symmetric-key encryption.

To make things more efficient, a file will generally be encrypted with a symmetric-key algorithm, and then the symmetric key will be encrypted with RSA encryption. Under this process, only an entity that has access to the RSA private key will be able to decrypt the symmetric key.

Without being able to access the symmetric key, the original file can't be decrypted. This method can be used to keep messages and files secure, without taking too long or consuming too many computational resources.

RSA encryption can be used in a number of different systems. It can be implemented in OpenSSL, wolfCrypt, cryptlib and a number of other cryptographic libraries.

As one of the first widely used public-key encryption schemes, RSA laid the foundations for much of our secure communications. It was traditionally used in TLS and was also the original algorithm used in PGP encryption. RSA is still seen in a range of web browsers, email, VPNs, chat and other communication channels.

RSA is also often used to make secure connections between VPN clients and VPN servers. Under protocols like OpenVPN, TLS handshakes can use the RSA algorithm to exchange keys and establish a secure channel.

The background of RSA encryption: -

As we mentioned at the start of this article, before public-key encryption, it was a challenge to communicate securely if there hadn't been a chance to safely exchange keys beforehand. If there wasn't an opportunity to share the code ahead of time, or a secure channel through which the keys could be distributed, there was no way to communicate without the threat of enemies being able to intercept and access the message contents.

It wasn't until the 1970s that things really began to change. The first major development towards what we now call public-key cryptography was published at the start of the decade by James H. Ellis. Ellis couldn't find a way to implement his work, but it was expanded upon by his colleague Clifford Cocks to become what we now know as RSA encryption.

The final piece of the puzzle is what we now call the Diffie-Hellman key exchange. Malcolm J. Williamson, another coworker, figured out a scheme that allowed two parties to share an encryption key, even if the channel was being monitored by adversaries.

All of this work was undertaken at the UK intelligence agency, the Government Communications Headquarters (GCHQ), which kept the discovery classified. Partly due to technological limitations, the GCHQ couldn't see a use for public-key cryptography at the time, so the development sat idly on the shelf gathering dust. It wasn't until 1997 that the work was declassified and the original inventors of RSA were acknowledged.

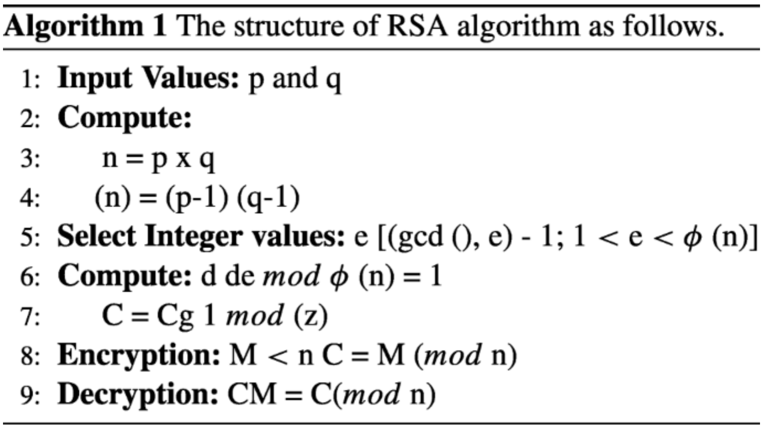
Several years later, similar concepts were beginning to develop in the public sphere. Ralph Merkle created an early form of public-key cryptography, which influenced Whitfield Diffie and Martin Hellman in the design of the Diffie-Hellman key exchange.

Diffie and Hellman's ideas were missing one important aspect that would make their work a foundation of public key cryptography. This was a one-way function that would be difficult to invert. In 1977, Ron Rivest, Adi Shamir and Leonard Adleman, whose last names form the RSA acronym, came up with a solution after a year of laboring on the problem.

How does RSA encryption work?

RSA encryption uses prime numbers that are much larger in magnitude and there are a few other complexities.There are several different concepts you will have to get your head around before we can explain how it all fits together. These include trapdoor functions, generating primes, Carmichael's totient function and the separate processes involved in computing the public and private keys used in the encryption and decryption processes.

Trap door functions



```
Algorithm 1 The structure of RSA algorithm as follows.
Input Values: p and q
Compute:
    n = p x q
    (n) = (p-1) (q-1)
Select Integer values: e [(gcd (), e) - 1; 1 < e < $\phi$ (n)]
Compute: d de $mod$ $\phi$ (n) = 1
    C = Cg 1 $mod$ (z)
Encryption: M < n C = M ($mod$ n)
Decryption: CM = C($mod$ n)
```

Figure 2.2: Working of RSA in the aspect of Number theory.

RSA encryption works under the premise that the algorithm is easy to compute in one direction, but almost impossible in reverse. As an example, if you were told that 701,111 is a product of two prime numbers, would you be able to figure out what those two numbers are? Even with a calculator or a computer, most of us wouldn't have any idea of where to start, let alone be able to figure out the answer. But if we flip things around, it becomes much easier.It is simple to figure out one of the prime numbers if you already have the other one, as well as the product. If you are told that 701,111 is the result of 907 multiplied by another prime number, you can figure it out the other prime with the following equation:

701,111 ÷ 907 = 773

Since the relationship between these numbers is simple to compute in one direction, but incredibly hard in reverse, the equation is known as a trap door function.

Because of this, RSA uses much larger numbers. The size of the primes in a real RSA implementation varies, but in 2048-bit RSA, they would come together to make keys that are 617 digits long.

Generating primes: -

The trap door functions mentioned above form the basis for how public and private-key encryption schemes work. Their properties allow public keys to be shared without endangering the message or revealing the private key. They also allow data to be encrypted with one key in a way that can only be decrypted by the other key from the pair.

The first step of encrypting a message with RSA is to generate the keys. To do this, we need two prime numbers (p and q) which are selected with a primality test. A primality test is an algorithm that efficiently finds prime numbers, such as the **Rabin-Miller primality test**.

The prime numbers in RSA need to be very large, and also relatively far apart. Numbers that are small or closer together are much easier to crack.

Carmichael's totient function: -

Once we have n, we use Carmichael's totient function: -

$\lambda(n)$ = lcm(p-1, q-1)

$\lambda(n)$ represents Carmichael's totient for n, while lcm means the lowest common multiple, which is the lowest number that both p and q can divide into.

Generating the public key: -

Now that we have Carmichael's totient of our prime numbers, it's time to figure out our public key. Under RSA, public keys are made up of a prime number e, as well as modulus n .Because the public key is shared openly, it's not so important for e to be a random number.Our final encrypted data is called the **ciphertext** (c).

We derive it from our plaintext message (m), by applying the public key with the following formula:

$c = m^e$ mod n

To keep things simple, let's say that the message (m) that we want to encrypt and keep secret.

Here,we can send the required message, **encrypting with cipher-text**.

Generating the private key: -

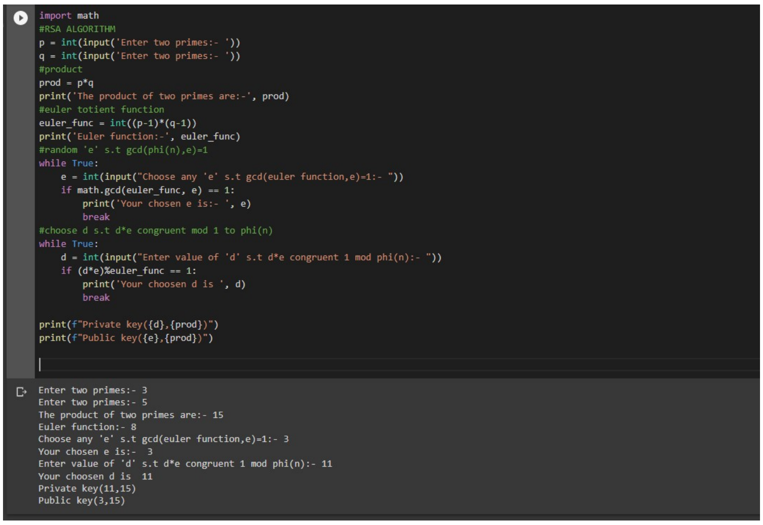
In RSA encryption, once data or a message has been turned into **ciphertext with a public key**, it can only be decrypted by the private key from the same key pair. Private keys are comprised of **d and n**. We already know **n**, and the following equation is used to find d:

$d = (1/e) mod \lambda(n)$
This essentially means that instead of performing a standard modulo operation, we will be using the inverse instead. This is normally found with the Extended Euclidean Algorithm.
Now that we have the value for d, we can decrypt messages that were encrypted with our public key using the following formula:

$$m = c^d \text{ mod n}$$

We can now go back to the ciphertext that we encrypted under the Generating the private key section. When we encrypted the message with the public key, it gave us a value for c.

How RSA encryption works in practice: -



```
import math
#RSA ALGORITHM
p = int(input('Enter two primes:- '))
q = int(input('Enter two primes:- '))
#product
prod = p*q
print('The product of two primes are:-', prod)
#euler totient function
euler_func = int((p-1)*(q-1))
print('Euler function:-', euler_func)
#random 'e' s.t gcd(phi(n),e)=1
while True:
    e = int(input("Choose any 'e' s.t gcd(euler function,e)=1:- "))
    if math.gcd(euler_func, e) == 1:
        print('Your chosen e is:- ', e)
        break
#choose d s.t d*e congruent mod 1 to phi(n)
while True:
    d = int(input("Enter value of 'd' s.t d*e congruent 1 mod phi(n):- "))
    if (d*e)%euler_func == 1:
        print('Your choosen d is ', d)
        break

print(f"Private key({d},{prod})")
print(f"Public key({e},{prod})")
```

```
Enter two primes:- 3
Enter two primes:- 5
The product of two primes are:- 15
Euler function:- 8
Choose any 'e' s.t gcd(euler function,e)=1:- 3
Your chosen e is:-  3
Enter value of 'd' s.t d*e congruent 1 mod phi(n):- 11
Your choosen d is  11
Private key(11,15)
Public key(3,15)
```

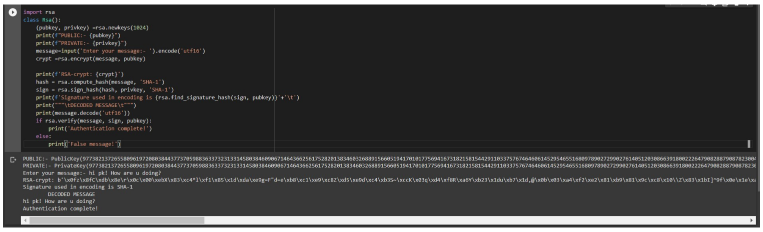
Figure 2.3: RSA python programming in the aspect of Number theory.

In the steps listed above, we have shown how two entities can securely communicate without having previously shared a code beforehand. First, they each need to set up their own key pairs and share the public key with one another. The two entities need to keep their private keys secret in order for their communications to remain secure.

Once the sender has the public key of their recipient, they can use it to encrypt the data that they want to keep secure. Once it has been encrypted with a public key, it can only be decrypted by the private key from the same key pair. Even the same public key can't be used to decrypt the data. This is due to the properties of trap door functions that we mentioned above.

When the recipient receives the encrypted message, they use their private key to access the data. If the recipient wants to return communications in a secure way, they can then encrypt their message with the public key of the party they are communicating with. Again, once it has been encrypted with the public key, the only way that the information can be accessed is through the matching private key.

In this way, RSA encryption can be used by previously unknown parties to securely send data between themselves. Significant parts of the communication channels that we use in our online lives were built up from this foundation.



```
import rsa
class Rsa():
    (pubkey, privkey) =rsa.newkeys(1024)
    print(f"PUBLIC:- {pubkey}")
    print(f"PRIVATE:- {privkey}")
    message=input('Enter your message:- ').encode('utf16')
    crypt =rsa.encrypt(message, pubkey)

    print(f'RSA-crypt: {crypt}')
    hash = rsa.compute_hash(message, 'SHA-1')
    sign = rsa.sign_hash(hash, privkey, 'SHA-1')
    print(f'Signature used in encoding is {rsa.find_signature_hash(sign, pubkey)}'+'\t')
    print("""\tDECODED MESSAGE\t""")
    print(message.decode('utf16'))
    if rsa.verify(message, sign, pubkey):
        print('Authentication complete!')
    else:
        print('False message!')
```

```
PUBLIC:- PublicKey(97738213726558096197208038443773705988363373231331458038460906714643662561752820138346032688915660519417010177569416731821581544291103375767464606145295465516809789027299027614051203086639180022264790828879087823004
PRIVATE:- PrivateKey(9773821372655809619720803844377370598836337323133145803846090671464366256175282013834603268891566051941701017756941673182158154429110337576746460614529546551680978902729902761405120308663918002226479082887908782300
Enter your message:- hi pk! How are u doing?
RSA-crypt: b'\x0fz\x8fC\xdb\x8e\r\x0c\x00\xebX\x83\xc4*l\xf1\x85\x1d\xda\xe9g=F"d=e\xb8\xc1\xe9\xc8Z\xd5\xe9d\xc4\xb3S~\xccK\x03q\xd4\xf8R\xa6Y\xb23\x1du\xb7\x1d,@\x0b\x03\xa4\xf2\xe2\x81\xb9\x81\x9c\xc8\x10\\Z\x83\x1bI]^9f\x0e\x1e\xa
Signature used in encoding is SHA-1
        DECODED MESSAGE
hi pk! How are u doing?
Authentication complete!
```

Figure 2.4: RSA python programming using modules.

How are more complicated messages encrypted with RSA?

Some people may be perplexed at how a key like $n38cb29fkbjh138g7fq$ or a message like "buy me a sandwich" can be encrypted by an algorithm like RSA, which deals with numbers and not letters. The reality is that all of the information that our computers process is stored in binary (1s and 0s) and we use encoding standards like ASCII or Unicode to represent them in ways that humans can understand (letters).

This means that keys like “n38cb29fkbjh138g7fqijnf3kaj84f8b9f…” and messages like “buy me a sandwich” already exist as numbers, which can easily be computed in the RSA algorithm. The numbers that they are represented by are much larger and harder for us to manage, which is why we prefer to deal with alphanumeric characters rather than a jumble of binary.

If you wanted to encrypt a longer session key or a more complex message with RSA, it would simply involve a much larger number.

Padding: -

When RSA is implemented, it uses something called padding to help prevent a number of attacks. To explain how this works, we’ll start with an example. Let’s say you were sending a coded message to a friend:

Dear Karen,

I hope you are well. Are we still having dinner tomorrow?

Yours sincerely,

James

Let’s say that you coded the message in a simple way, by changing each letter to the one that follows it in the alphabet. This would change the message to:

Efbs Lbsfo,

J ipqf zpv bsf xfmm. Bsf xf tujmm ibwjoh ejoofs upnpsspx?

Zpvst tjodfsfmz,

Kbnft

If your enemies intercepted this letter, there is a trick that they could use to try and crack the code. They could look at the format of your letter and try to guess what the message might be saying. They know that people normally begin their letters with “Hi”, “Hello”, “Dear” or a number of other conventions.

If they tried to apply “Hi” or “Hello” as the first word, they would see that it wouldn’t fit the number of characters. They could then try “Dear”. It fits, but that doesn’t necessarily mean anything. The attackers would just try it and see where it led them. So they would change the letters “e”, “f”, “b”, and “s” with “d”, “e”, “a”, and “r” respectively. This would give them:

Dear Laseo,

J ipqe zpv are xemm. Are xe tujmm iawjoh djooes upnpsspx?

Zpvrt tjoderemz,

Kanet

It still looks pretty confusing, so the attackers might try looking at some other conventions, like how we conclude our letters. People often add “From” or “Kind regards” at the end, but neither of these fit the format. Instead, the attackers might try “Yours sincerely” and replace the other letters to see where it gets them. By changing “z”, “p”, “v”, “t”, “j” “o”, “d” and “m” with “y”, “o”, “u”, “s”, “i”, “n”, “c” and “l” respectively, they would get:

Dear Lasen,

I ioqe you are xell. Are xe tuill iawinh dinnes uonossox?

Yours sincerely,

Kanet

After that modification, it looks like the attackers are starting to get somewhere. They have found the words “I”, “you” and “are”, in addition to the words that made up their initial guesses.

Seeing as the words are in correct grammatical order, the attackers can be pretty confident that they are heading in the right direction. By now, they have probably also realized that the code involved each letter being changed to the one that follows it in the alphabet. Once they realize this, it makes it easy to translate the rest and read the original message.

The above example was just a simple code, but as you can see, the structure of a message can give attackers clues about its content. Sure, it was difficult to figure out the message from just its structure and it took some educated guesswork, but you need to keep in mind that computers are much better at doing this than we are. This means that they can be used to figure out far more complex codes in a much shorter time, based on clues that come from the structure and other elements.

If the structure can lead to a code being cracked and reveal the contents of a message, then we need some way to hide the structure in order to keep the message secure. This brings us to padding.

When a message is padded, randomized data is added to hide the original formatting clues that could lead to an encrypted message being broken. With RSA, things are a little bit more complicated, because an encrypted key doesn't have the obvious formatting of a letter that helped to give us clues in our above example.

Despite this, adversaries can use a number of attacks to exploit the mathematical properties of a code and break encrypted data. Due to this threat, implementations of RSA use padding schemes like OAEP to embed extra data into the message. Adding this padding before the message is encrypted makes RSA much more secure.

Signing messages: -

RSA can be used for more than just encrypting data. Its properties also make it a useful system for confirming that a message has been sent by the entity who claims to have sent it, as well as proving that a message hasn't been altered or tampered with.

When someone wants to prove the authenticity of their message, they can compute a hash (a function that takes data of an arbitrary size and turns it into a fixed-length value) of the plaintext, then sign it with their private key. They sign the hash by applying the same formula that is used in decryption (m = cd mod n). Once the message has been signed, they send this digital signature to the recipient alongside the message.

If a recipient receives a message with a digital signature, they can use the signature to check whether the message was authentically signed by the private key of the person who claims to have sent it. They can also see whether the message has been changed by attackers after it was sent.

To check the digital signature, the recipient first uses the same hash function to find the hash value of the message they received. The recipient then applies the sender's public key to the digital signature, using the encryption formula (c = me mod n), to give them the hash of the digital signature.

By comparing the hash of the message that was received alongside the hash from the encrypted digital signature, the recipient can tell whether the message is authentic. If the two values are the same, the message has not been changed since it was signed by the original sender. If the message had been altered by even a single character, the hash value would be completely different.

RSA security attacks: -

The larger the number of bits in a key (essentially how long the key is), the more difficult it is to crack through attacks such as brute-forcing and factoring.Since asymmetric-key algorithms such as RSA can be broken by integer factorization, while symmetric-key algorithms like AES cannot, RSA keys need to be much longer to achieve the same level of security.

Currently, the largest key size that has been factored is 768 bits long. The **National Institute of Standards and Technology** recommends a minimum key size of 2048-bit, but 4096-bit keys are also used in some situations where the threat level is higher.

A number of other attacks have the potential to break the encryption with a smaller amount of resources, but these depend on the implementation and other factors, not necessarily RSA itself. Some of these include:

Are the primes really random?

Some implementations of RSA use weak random number generators to come up with the primes. If these numbers aren't sufficiently random, it makes it much easier for attackers to factor them and break the encryption. This problem can be avoided by using a cryptographically secure pseudo-random number generator.

Poor key generation: -

RSA keys need to fall within certain parameters in order for them to be secure. If the primes p and q are too close together, the key can easily be discovered. Likewise, the number d that makes up part of the private key cannot be too small. A low value makes it easy to solve. It's important that these numbers are of adequate length to keep your key safe.

Side channel attacks: -

These are a type of attack that don't break RSA directly, but instead use information from its implementation to give attackers hints about the encryption process. These attacks can include things like analyzing the amount of power that is being used, or branch prediction analysis, which uses execution-time measurements to discover the private key.

Another type of side channel attack is known as a timing attack. If an attacker has the ability to measure the decryption time on their target's computer for a number of different encrypted messages, this information can make it possible for the attacker to ascertain the target's private key.

Most implementations of RSA avoid this attack by adding a one-off value during the encryption process, which removes this correlation. This process is called cryptographic blinding.

Is RSA encryption safe for the future?

The good news is that RSA is currently considered safe to use, despite these possible attacks. The caveat is that it needs to be implemented correctly and use a key that falls within the correct parameters. As we have just discussed, implementations that don't use padding, use inadequately sized primes or have other vulnerabilities can not be considered safe.

If you want to use RSA encryption, make sure that you are using a key of at least 1024 bits. Those with higher threat models should stick to keys of 2048 or 4096 bits if they want to use RSA with confidence.

As long as you are conscious of the weaknesses that RSA has and use it correctly, you should feel safe to use RSA for key sharing and other similar tasks that require public key encryption.

Chapter 3

AES

What is AES encryption?

The AES algorithm (also known as the Rijndael algorithm) is a symmetrical block cipher algorithm that takes plain text in blocks of 128 bits and converts them to ciphertext using keys of 128, 192, and 256 bits. Since the AES algorithm is considered secure, it is in the worldwide standard.

In short, AES is a symmetric type of encryption, as it uses the same key to both encrypt and decrypt data.

It also uses the SPN (substitution permutation network) algorithm, applying multiple rounds to encrypt data. These encryption rounds are the reason behind the impenetrability of AES, as there are far too many rounds to break through.

There are three lengths of AES encryption keys. Each key length has a different number of possible key combinations:

128-bit key length: 3.4 x 1038

192-bit key length: 6.2 x 1057

256-bit key length: 1.1 x 1077

Even though the key length of this encryption method varies, its block size - 128-bits (or 16 bytes) - stays fixed.

Why are there multiple key lengths?

And, if the 256-bit key is the strongest of the bunch (even referred to as "military-grade" encryption), why don't we just always use it?

Well, it all comes down to resources. For example, an app that uses AES-256 instead of AES-128 might drain your phone battery a bit faster.

Luckily, current technology makes the resource difference so minuscule that there is simply no reason not to use 256-bit AES encryption.

How does AES work?

The AES algorithm uses a substitution-permutation, or SP network, with multiple rounds to produce ciphertext. The number of rounds depends on the key size being used. A 128-bit key size dictates ten rounds, a 192-bit key size dictates 12 rounds, and a 256-bit key size has 14 rounds. Each of these rounds requires a round key, but since only one key is inputted into the algorithm, this key needs to be expanded to get keys for each round, including round 0.

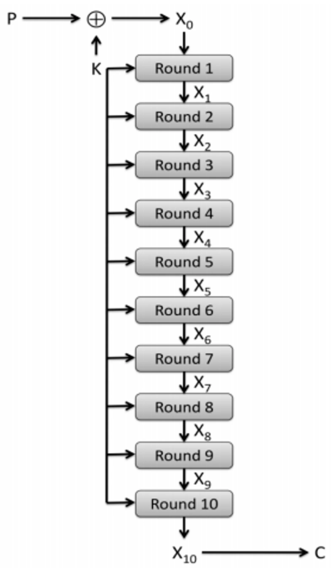


Figure 3.1: AES(FLOWCHART PROCESS)

Steps in each round: -

Each round in the algorithm consists of four steps: -

1. Substitution of the bytes: -

In the first step, the bytes of the block text are substituted based on rules dictated by predefined S-boxes (short for substitution boxes).

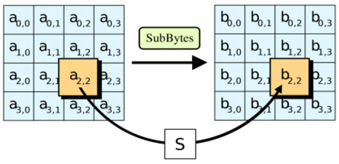


Figure 3.2: SUBSTITUTION OF BYTES

2. Shifting the rows: -

Next comes the permutation step. In this step, all rows except the first are shifted by one, as shown below.

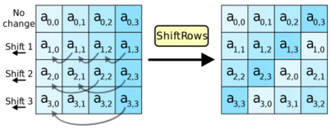


Figure 3.3: SHIFTING OF ROWS

3. Mixing the columns: -

In the third step, the Hill cipher is used to jumble up the message more by mixing the block's columns.

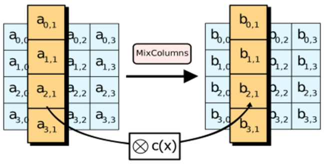


Figure 3.4: MIXING THE COLUMNS

4. Adding the round key: -

The advantages of AES: -

Safety aside, AES encryption is very appealing to those who work with it. Why? Because the encryption process of AES is relatively easy to

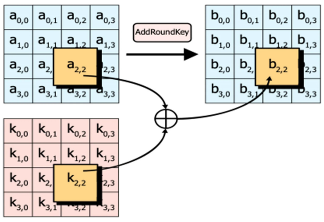
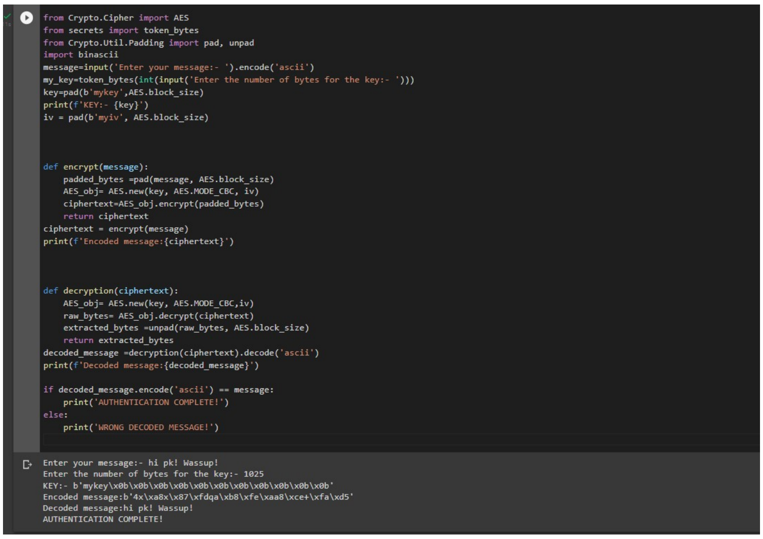


Figure 3.5: ADDING THE ROUND KEY

understand. This allows for easy implementation, as well as really fast encryption and decryption times.

In addition, AES requires less memory than many other types of encryption (like DES), which makes it a true winner when it comes to choosing your preferred encryption method.

Finally, when an action requires an extra layer of safety, you can combine AES with various security protocols like WPA2 or even other types of encryption like SSL.
When done repeatedly, these steps ensure that the final ciphertext is secure.



```
from Crypto.Cipher import AES
from secrets import token_bytes
from Crypto.Util.Padding import pad, unpad
import binascii
message=input('Enter your message:- ').encode('ascii')
my_key=token_bytes(int(input('Enter the number of bytes for the key:- ')))
key=pad(b'mykey',AES.block_size)
print(f'KEY:- {key}')
iv = pad(b'myiv', AES.block_size)



def encrypt(message):
    padded_bytes =pad(message, AES.block_size)
    AES_obj= AES.new(key, AES.MODE_CBC, iv)
    ciphertext=AES_obj.encrypt(padded_bytes)
    return ciphertext
ciphertext = encrypt(message)
print(f'Encoded message:{ciphertext}')



def decryption(ciphertext):
    AES_obj= AES.new(key, AES.MODE_CBC,iv)
    raw_bytes= AES_obj.decrypt(ciphertext)
    extracted_bytes =unpad(raw_bytes, AES.block_size)
    return extracted_bytes
decoded_message =decryption(ciphertext).decode('ascii')
print(f'Decoded message:{decoded_message}')

if decoded_message.encode('ascii') == message:
    print('AUTHENTICATION COMPLETE!')
else:
    print('WRONG DECODED MESSAGE!')
```

```
Enter your message:- hi pk! Wassup!
Enter the number of bytes for the key:- 1025
KEY:- b'mykey\x0b\x0b\x0b\x0b\x0b\x0b\x0b\x0b\x0b\x0b\x0b'
Encoded message:b'4x\xa8x\x87\xfdqa\xb8\xfe\xaa8\xce+\xfa\xd5'
Decoded message:hi pk! Wassup!
AUTHENTICATION COMPLETE!
```

Figure 3.6: AES PYTHON PROGRAMMING.

Why do we use the AES algorithm?

Originally developed in 1998 by two Belgian cryptographers, Vincent Rijmen and Joan Daemen, AES has been around for more than 20 years.

At first, it was referred to as Rijndael - a combination of the names of its developers.

Thanks to its impenetrability, AES encryption has already served as the encryption standard for 18 years. That's because, in 2002, the National Institute of Standards and Technology (NIST) replaced the outdated Data Encryption Standard (DES) with AES.

Where is the AES algorithm used?
AES encryption quickly took the world by storm, becoming the encryption standard for basically anything we see online. As a result, you will have trouble finding industries or services that don't use the AES algorithm.

Online banking credentials, passwords, and messages all need to be protected from people who can do harm. So, aside from "serving" the government (like the National Security Agency), the advanced encryption standard protects the sensitive data of a myriad of products.
Examples of AES usage: -
Wi-Fi. That's right - wireless networks also use AES encryption (usually, together with WPA2). This is not the only type of encryption Wi-Fi networks can use, however, most of the other encryption methods are far less safe.
Programming language libraries. The libraries of such coding languages like Java, Python, and C++ implement AES encryption.
Password manager:- . These are the programs that carry a lot of sensitive information. That's why password managers like LastPass and Dashlane don't skip the important step of AES implementation.
AES decryption: - With the help of inverse encryption, the AES ciphertext can be restored to its initial state.

As mentioned before, the advanced encryption standard implements the method of symmetric cryptography. In other words, it uses the same key for both data encryption and decryption.

In this way, it differs from the algorithms that use asymmetric encryption, when both public and private keys are required.

So, in our case, AES decryption begins with the inverse round key. Afterwards, the algorithm reverses every single action (shift rows, byte substitution, and, later on, column mixing), until it deciphers the original message.

Related-key attacks: -

Side-channel attacks

In case of improper implementation of a computer system, AES encryption is not completely immune to side-channel attacks.

This type of attack relies on data leakage, for example, electromagnetic leaks.

However, if AES is properly implemented, it can help detect the data leaks before anything bad happens.

Known-key distinguishing attacks

This type of attack requires the hacker to have at least one pair of encrypted and decrypted messages.

However, the test didn’t provide significant results, as it only proved to be four times faster than a brute-force attack (which would still take billions of years).

Chapter 4

Tweaked Block-Chain(TBC)

Blockchain is a system of recording information in a way that makes it difficult or impossible to change, hack, or cheat the system.
A blockchain is essentially a digital ledger of transactions that is duplicated and distributed across the entire network of computer systems on the blockchain. Each block in the chain contains a number of transactions, and every time a new transaction occurs on the blockchain, a record of that transaction is added to every participant's ledger. The decentralised database managed by multiple participants is known as Distributed Ledger Technology (DLT).

Blockchain is a type of DLT in which transactions are recorded with an immutable cryptographic signature called a hash.

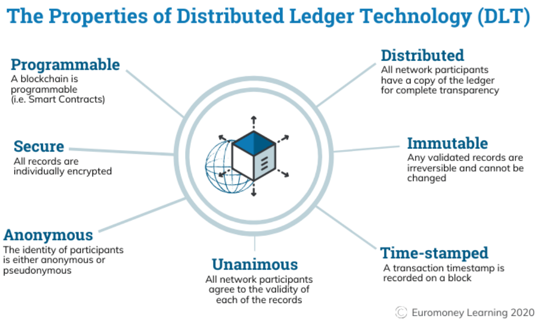

The Properties of Distributed Ledger Technology (DLT)
Programmable
A blockchain is programmable (i.e. Smart Contracts)
Distributed
All network participants have a copy of the ledger for complete transparency
Secure
All records are individually encrypted
Immutable
Any validated records are irreversible and cannot be changed
Anonymous
The identity of participants is either anonymous or pseudonymous
Unanimous
All network participants agree to the validity of each of the records
Time-stamped
A transaction timestamp is recorded on a block
© Euromoney Learning 2020

Figure 4.1: BLOCKCHAIN

This means if one block in one chain was changed, it would be immediately apparent it had been tampered with. If hackers wanted to corrupt a blockchain system, they would have to change every block in the chain, across all of the distributed versions of the chain.

Why is there so much hype around blockchain technology?

There have been many attempts to create digital money in the past, but they have always failed.

The prevailing issue is trust. If someone creates a new currency called the X dollar, how can we trust that they won't give themselves a million X dollars, or steal your X dollars for themselves?

Bitcoin was designed to solve this problem by using a specific type of database called a blockchain. Most normal databases, such as an SQL database, have someone in charge who can change the entries (e.g. giving themselves a million X dollars). Blockchain is different because nobody is in charge; it's run by the people who use it. What's more, bitcoins can't be faked, hacked or double spent – so people that own this money can trust that it has some value.

How Does Blockchain Work?

Blockchain consists of three important concepts: blocks, nodes and miners.

Blocks: -

Every chain consists of multiple blocks and each block has three basic elements: -

- The data in the block.
- A 32-bit whole number called a nonce. The nonce is randomly generated when a block is created, which then generates a block header hash.
- The hash is a 256-bit number wedded to the nonce. It must start with a huge number of zeroes (i.e., be extremely small).

When the first block of a chain is created, a nonce generates the cryptographic hash. The data in the block is considered signed and forever tied to the nonce and hash unless it is mined.

Miners: -

Miners create new blocks on the chain through a process called mining.

In a blockchain every block has its own unique nonce and hash, but also references the hash of the previous block in the chain, so mining a block isn't easy, especially on large chains.

Miners use special software to solve the incredibly complex math problem of finding a nonce that generates an accepted hash. Because the nonce is only 32 bits and the hash is 256, there are roughly four billion possible nonce-hash combinations that must be mined before the right one is found. When that happens miners are said to have found the "golden nonce" and their block is added to the chain.

Making a change to any block earlier in the chain requires re-mining not just the block with the change, but all of the blocks that come after. This is why it's extremely difficult to manipulate blockchain technology. Think of it as "safety in math" since finding golden nonces requires an enormous amount of time and computing power.

When a block is successfully mined, the change is accepted by all of the nodes on the network and the miner is rewarded financially.

Nodes: -

One of the most important concepts in blockchain technology is decentralization. No one computer or organization can own the chain. Instead, it is a distributed ledger via the nodes connected to the chain. Nodes can be any kind of electronic device that maintains copies of the blockchain and keeps the network functioning. **Blockchain Characteristics:-**

Immutability: A hallmark feature of blockchain is that it is incorruptible.

Data once saved in a block cannot be changed without breaking the chain.

Decentralisation: Blockchain does not require a central authority to maintain the ledger. All nodes in the network have a copy of the ledger that gets updated with every transaction.

Enhanced security: Decentralisation and consensus mechanisms built into blockchain make it highly secure, as every transaction has to be verified with 'proof of work'.

Distributed: The blockchain ledger is not a singular entity in the hands of a few; instead it is a shared entity that is accessible and maintained by everyone in the network.

Transparency: Every addition to the blockchain requires consensus from all nodes in the peer-to-peer network. This makes every transaction transparent and public.

Open Source Characteristics: -

Transparency: The founding principle of open source is to make the source code of the project public. This enhances the transparency of the project, as every detail that goes into its development can be freely accessed by anyone.

Enhanced security: Finding security loopholes in the software is way easier when the code is non-proprietary and is publicly available. This naturally enhances the security of the software or, at the very least, can help raise public awareness about the potential security-related pitfalls.

Collaboration: The primary purpose of open source software is to encourage other developers to collaborate and contribute to the development of the software. This makes open source development, by nature, a community experience.

Modifiability: Open source software can be modified and re-distributed to solve different problems. This is the case with Linux, as its modified ‘distributions’ are used in various different gadgets and computers.

Decentralisation: One can argue that open source software is not inherently decentralised as making changes to the main project still requires the approval of the project moderator(s) and the owner(s). But if perceived from a different angle, given that the source code can be modified and re-distributed under proper licensing, this gives the developer the autonomy to use the software as and how desired thus making the process decentralised.

Every node has its own copy of the blockchain and the network must algorithmically approve any newly mined block for the chain to be updated, trusted and verified. Since blockchains are transparent, every action in the ledger can be easily checked and viewed. Each participant is given a unique alphanumeric identification number that shows their transactions.

A Blockchain Transaction.

To understand the fundamentals of the blockchain, we need to understand how a transaction can be added to the existing blockchain. This can be done by going through a few important steps, as illustrated in the below figure.

1.Assume that user 1 wants to send X amount of digital bitcoins to user 2. User 1 then initiates the transaction.

2.A transaction is initiated by user 1’s node by first making it and then digitally signing it with its private key. A transaction in a blockchain can reflect a range of actions.

3.A peer-to-peer (P2P) network broadcasts the desired transaction to each individual computer (or node).

4.Individual nodes receive the request and attempt to validate the transaction using cryptographic techniques. The miner node is the final node in the validation chain. These miner nodes are compensated in

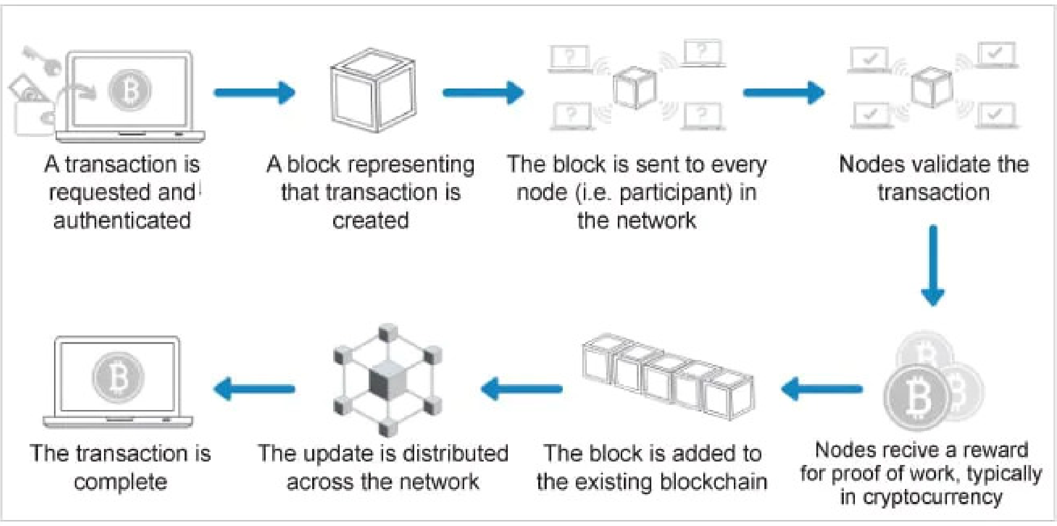

A transaction is requested and authenticated
A block representing that transaction is created
The block is sent to every node (i.e. participant) in the network
Nodes validate the transaction
Nodes recive a reward for proof of work, typically in cryptocurrency
The block is added to the existing blockchain
The update is distributed across the network
The transaction is complete

Figure 4.2: BLOCKCHAIN TRANSACTION.

Bitcoin.
5.The verified blocks are now added to the blockchain network. Hashing is used to connect these nodes.
6.The approved transactions are then recorded in a public ledger. Transactions are completed and the ledger is updated after the block is added to an existing chain.

Cryptography Techniques Used in Blockchain:-

Hashing, public-private key mechanisms, and digital signatures are the key cryptographic techniques used by the blockchain community. In this case, a hash feature is implemented to offer each user the opportunity to view the blockchain in its entirety. Blockchain characteristically uses the SHA-256 hashing algorithm. Public-private key pairs are used to gain access to the information and allow transactions. Digital signatures are also used for multi-signature contracts and digital wallets on the blockchain, as well as to ratify transactions by signing them securely (offline).

Working of A Blockchain Network

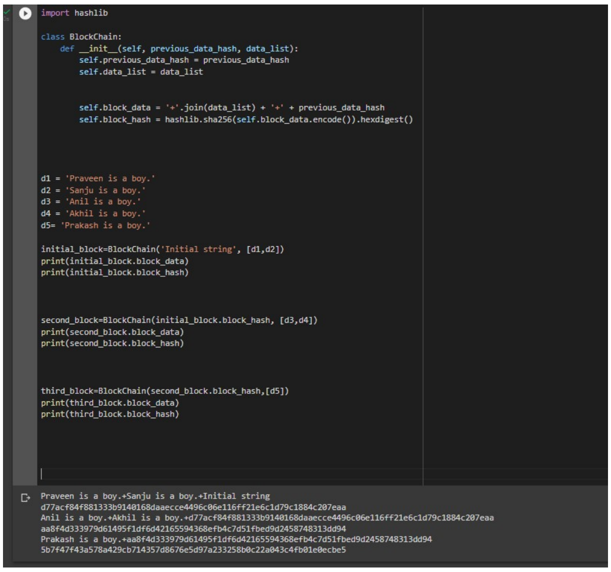
In a blockchain network, each node holds a complete copy of the distributed ledger. The four fields of a block in this network are the block number, the data field, the hash value, and the nonce. Miners adjust the value of nonce to make it a suitable number for hashing the value of a block. A nonce is a random whole number that is a 32-bit (4-byte) field that is changed by miners to make it an acceptable number for hashing the value of a block. In addition, the chain of blocks is generated by applying the hash of the previous block to the current block. The previous block's hash is stored in the field prior.

Four zeros are used to begin the hash value. This determines whether or not a block is valid. Each new hash value is created by combining the old hash value, the new transaction block, and a nonce. The hash cash difficulty factor is determined by the number of leading zeroes required in the hash output. The difficulty is tweaked so that block production stays consistent at about one block per ten minutes. The more difficult it is to alter the blockchain, the tougher it is to overwrite the blockchain and double-spend coins.

Combining public information with a system of checks-and-balances helps the blockchain maintain integrity and creates trust among users. Essentially, blockchains can be thought of as the scalability of trust via technology.

Here we will understand the concept of Tweak Block-Chain through Tweaked Block-Cipher.

First of all,obviously, we want any tweakable block ciphers we design to be as efficient as possible (just as with any scheme). Specifically, a tweakable block cipher should have the property that changing the tweak should be less costly than changing the key. Many block ciphers have the property that changing the encryption key is relatively expensive, since a "key setup" operation needs to be performed. In contrast, changing the tweak should be cheaper.



```python
import hashlib

class BlockChain:
    def __init__(self, previous_data_hash, data_list):
        self.previous_data_hash = previous_data_hash
        self.data_list = data_list


        self.block_data = '+'.join(data_list) + '+' + previous_data_hash
        self.block_hash = hashlib.sha256(self.block_data.encode()).hexdigest()




d1 = 'Praveen is a boy.'
d2 = 'Sanju is a boy.'
d3 = 'Anil is a boy.'
d4 = 'Akhil is a boy.'
d5= 'Prakash is a boy.'

initial_block=BlockChain('Initial string', [d1,d2])
print(initial_block.block_data)
print(initial_block.block_hash)



second_block=BlockChain(initial_block.block_hash, [d3,d4])
print(second_block.block_data)
print(second_block.block_hash)



third_block=BlockChain(second_block.block_hash,[d5])
print(third_block.block_data)
print(third_block.block_hash)
```

```
Praveen is a boy.+Sanju is a boy.+Initial string
d77acf84f881333b9140168daaecce4496c06e116ff21e6c1d79c1884c207eaa
Anil is a boy.+Akhil is a boy.+d77acf84f881333b9140168daaecce4496c06e116ff21e6c1d79c1884c207eaa
aa8f4d333979d61495f1df6d42165594368efb4c7d51fbed9d2458748313dd94
Prakash is a boy.+aa8f4d333979d61495f1df6d42165594368efb4c7d51fbed9d2458748313dd94
5b7f47f43a578a429cb714357d8676e5d97a233258b0c22a043c4fb01e0ecbe5
```

Figure 4.3: BLOCKCHAIN

A **tweakable block cipher** should also be secure, meaning that even if an adversary has control of the tweak input, we want the tweakable block cipher to remain secure. We'll define what this means more precisely later on. But intuitively, each fixed setting of the tweak gives rise to a different, apparently independent, family of standard block cipher encryption operators. We wish to carefully distinguish between the function of the key, which is to provide uncertainty to the adversary, and the role of the tweak, which is to provide variability.
The tweak is not intended to provide additional uncertainty to an adversary. Keeping the tweak secret need not provide any greater cryptographic strength.
Tweakable Modes of Operation: -
The new "tweak" input of a tweakable block ciphers enables a multitude of new modes of operation.
Blockchain is this new technology that many are calling the "New Internet".

Basically it's a distributed digital ledger that permanently records transactions and activity. It is the engine that powers cryptocurrency, but it is also used in many, many other verticals, like car purchases, fashion, healthcare etc.

One of the main hurdles that Blockchain faces is the fact that it is poorly understood.

But there are other improvements that can be made:-

Scams and Fraudulent projects:-

As crypto markets develop, there are a lot of scams popping up like fraudulent ICOs, techniques to manipulate token value, etc. These kinds of scams are common in developing markets and hopefully the industry will find ways to sort this out before we stifle creativity or create too much regulation

Not as decentralized as people think they are:-

The public blockchains are not as decentralized as people think they are. If you look at the Ethereum and Bitcoin networks, you can see that there are millions of users globally, but the code bases of those technologies are still controlled by a relatively small group of developers. Additionally, network operations — bitcoin mining — is no longer economic for small participants, and this is by design! So it has become centralized to handful of companies mostly in China. You have legions of people touting the value and power of decentralization, but you should question how decentralized these systems really are. And then consider how decentralized they need to be in order for them to work. That's a question more for product designers and managers when trying to figure out what to do with their Blockchain solution. Very often you hear that a solution needs to be as "permissionless" and decentralized as possible. But the standards for permissionless and decentralized systems are really more of an ideal than a reality.

Regulatory Ambiguity:-

The lack of promulgated guidance from US regulatory agencies with respect to how companies are utilizing this technology in their business ventures makes it difficult for companies and their legal advisors to properly assess all potential risks before, during, and after the ICO process.

Cryptocurrency exchange hacks:-

Cryptocurrency exchange hacks and lack of investor protection. At least 3 dozen heists of Cryptocurrency exchanges and many hacked ones are shut down. As the technology keeps advancing, there are those who work hard to sabotage its success, the technology no doubt has some high level of security but there's room for more improvement.

Tweak Block Chaining (TBC)->

Tweak block chaining (TBC) is similar to cipher block chaining (CBC). An initial tweak T0 plays the role of the initialization vector (IV) for CBC. Each successive message block Mi is encrypted under control of the encryption key K and a tweak Ti1, where Ti = Ci for i > 0.

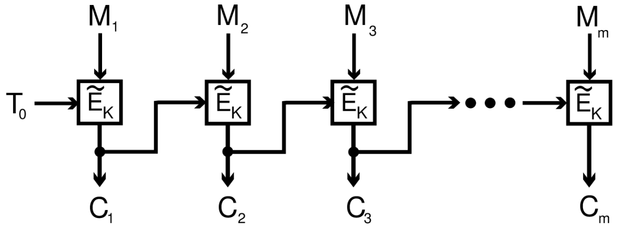
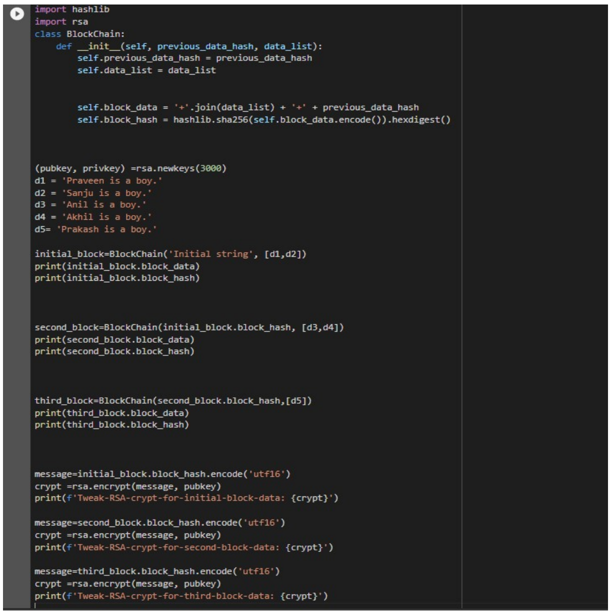


Figure 4.4: Tweak block chaining: a chaining mode for a tweakable block cipher. Each ciphertext becomes the tweak for the next encryption

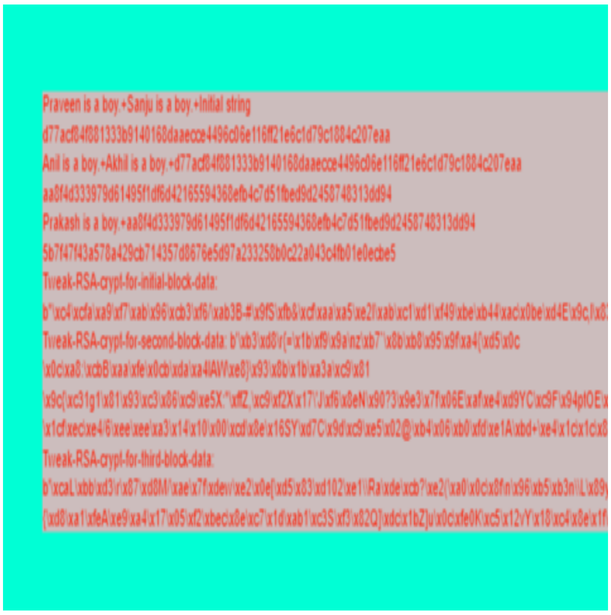
To handle messages whose length is greater than n but not a multiple of n, a variant of ciphertext-stealing [13] can be used; see Figure 3. One can also adapt the TBC construction to make a TBC-MAC in the same manner that one can use the CBC construction to make a CBC-MAC, though these constructions still need a security analysis.

Winding Up:- Blockchain technology is widely accepted by a variety of businesses. It has opened up a plethora of opportunities in the areas of healthcare, finance, accountancy and the online marketplace. The adoption of blockchain technology is expected to grow manifold in the coming years, offering quite a wide range of jobs for young aspirants.



```python
import hashlib
import rsa
class BlockChain:
    def __init__(self, previous_data_hash, data_list):
        self.previous_data_hash = previous_data_hash
        self.data_list = data_list


        self.block_data = '+'.join(data_list) + '+' + previous_data_hash
        self.block_hash = hashlib.sha256(self.block_data.encode()).hexdigest()



(pubkey, privkey) =rsa.newkeys(3000)
d1 = 'Praveen is a boy.'
d2 = 'Sanju is a boy.'
d3 = 'Anil is a boy.'
d4 = 'Akhil is a boy.'
d5= 'Prakash is a boy.'

initial_block=BlockChain('Initial string', [d1,d2])
print(initial_block.block_data)
print(initial_block.block_hash)



second_block=BlockChain(initial_block.block_hash, [d3,d4])
print(second_block.block_data)
print(second_block.block_hash)



third_block=BlockChain(second_block.block_hash,[d5])
print(third_block.block_data)
print(third_block.block_hash)



message=initial_block.block_hash.encode('utf16')
crypt =rsa.encrypt(message, pubkey)
print(f'Tweak-RSA-crypt-for-initial-block-data: {crypt}')

message=second_block.block_hash.encode('utf16')
crypt =rsa.encrypt(message, pubkey)
print(f'Tweak-RSA-crypt-for-second-block-data: {crypt}')

message=third_block.block_hash.encode('utf16')
crypt =rsa.encrypt(message, pubkey)
print(f'Tweak-RSA-crypt-for-third-block-data: {crypt}')
```

Figure 4.5: TWEAKED BLOCKCHAIN



```
Praveen is a boy.+Sanju is a boy.+Initial string
d77acf84f881333b9140168daaecce4496c06e116ff21e6c1d79c1884c207eaa
Anil is a boy.+Akhil is a boy.+d77acf84f881333b9140168daaecce4496c06e116ff21e6c1d79c1884c207eaa
aa8f4d333979d61495f1df6d42165594368efb4c7d51fbed9d2458748313dd94
Prakash is a boy.+aa8f4d333979d61495f1df6d42165594368efb4c7d51fbed9d2458748313dd94
5b7f47f43a578a429cb714357d8676e5d97a233258b0c22a043c4fb01e0ecbe5
Tweak-RSA-crypt-for-initial-block-data:
b"\xc4\xcfa\xa9\xf7\xab\x96\xcb3\xf6/\xab3B-#\x9fS\xfb&\xcf\xaa\xa5\xe2I\xab\xc1\xd1\xf49\xbe\xb44\xac\x0be\xd4E\x9c,I\x8
Tweak-RSA-crypt-for-second-block-data: b'\xb3\xd8\r{=\x1b\xf9\x9a\nz\xb7"\x8b\xb8\x95\x9f\xa4{\xd5\x0c
\x0c\xa8:\xcbB\xaa\xfe\x0cb\xda\xa4IAW\xe8}\x93\x8b\x1b\xa3a\xc9\x81
\x9c{\xc31g1\x81\x93\xc3\x86\xc9\xe5X:"\xffZ,\xc9\xf2X\x17\'J\xf6\x8eN\x90?3\x9e3\x7f\x06E\xaf\xe4\xd9YC\xc9F\x94ptOE\x
\x1cf\xec\xe4/6\xee\xee\xa3\x14\x10\x00\xcd\x8e\x16SY\xd7C\x9d\xc9\xe5\x02@\xb4\x06\xb0\xfd\xe1A\xbd+\xe4\x1c\x1c\x8
Tweak-RSA-crypt-for-third-block-data:
b'\xcaL\xbb\xd3\r\x87\xd8M/\xae\x7f\xdew\xe2\x0e[\xd5\x83\xd102\xe1\\Ra\xde\xcb?\xe2(\xa0\x0c\x8f\n\x96\xb5\xb3n\\L\x89y
{\xd8\xa1\xfeA\xe9\xa4\x17\x05\xf2\xbec\x8e\xc7\x1d\xab1\xc3S\xf3\x82Q}\xdc\x1bZ}u\x0c\xfe0K\xc5\x12vY\x18\xc4\x8e\x1f
```

Figure 4.6: OUTPUT OF TWEAKED BLOCKCHAIN

Chapter 5

Future works.

Based on the topic i have worked upon, will try to build an app for educational institute user-friendly and encrypted using algorithms and hash codes.There must be a body governing the app-community and it's guidelines. In other words, block-chain needs some chain of networks or blocks of data to maintain a chain around the blocks so that if any changes regarding the apps or any thing related to the app is to be changed then, the person seeking permission to change should have majority of votes for the proposal.

The app will contain all of the college students and teachers associated with the institutes, along with some of their academic data , the collected data will accordingly help us in sorting the students and respective teachers.

CASE-1: - some students want to change or organise an event then they can post out the ideas in there it will be voted accordingly blocks by blocks of associated teachers and officials, after which it will be implemented.

CASE-2: - Student willing to publish a paper or research work they can ask for ideas and help in it, it will be interactive and responsive if implemented properly.

CASE-3: - Alumni of the institute or the final year students opting for

placement course can also get to know about it.
They would need to undergo different training tests by office so that they will be well prepared for placement course!

Conclusion:- Therefore it’s my vision to implement this ideas through the project work. Here u would be guessing what is the need of encryption? But the thing is that we would never want our chats to be exposed,so, hereby comes the idea of encryption and security algorithms.

Chapter 6

Conclusion

In this paper, we have made a study on the different types of Encryption algorithms . We have presented different types of cryptosystems and its applications including communication,multimedia systems. This approach is relatively easy to implement and assure for the security enhancement.Here a new demanding concept is brought up into light i.e. **Blockchain** and its uses in Cyber-security (**Tweaked Block-Chain**).It will surely urge the readers with keen interest to learn more and more information about the block-chain developments and it's applied use in **Crypto-currency**.

Finally, reader can get to know the basic concepts about the theoretic and practical aspects of the encryption algorithms,encouraging them to dive deep into it and work out in securing different fields of **Data transmission**.

Chapter 7

Bibliography

1.Google

2.Design-for-Trust Techniques for Digital Microfluidic Biochip Layout with Error Control Mechanism,
Debasis Gountia, Student Member, IEEE and Sudip Roy, Member, IEEE,

3. Towards Security Aspects of Secret key Transmission Debasis Gountia1, Member, IEEE.

4.https://realpython.com/python-encodings-guide/#other-encodings-available-in-python

5.https://stuvel.eu/python-rsa-doc/usage.html

6.https://science.jrank.org/computer-science/Cryptography.html

7.https://www.mygreatlearning.com/blog/cryptography-tutorial/

8.https://www.comparitech.com/blog/information-security/rsa-encryption/

9.https://www.educative.io/answers/what-is-the-aes-algorithm

10.https://cybernews.com/resources/what-is-aes-encryption/