BLOCKCHAIN, CRYPTOGRAPHY & MATHEMATICAL FUNDAMENTAL FOR CRYPTOGRAPHY

A HANDBOOK WITH QUICK EXAMPLES

BY
ARYANADI IC

FIRST EDITION 2021

PREFACE

I've been trying to re-invent and transform myself since last May 2020 into somebody new and fresh with a new career in computer science field. So I'm making a move. I started it all from the scratch because filmmaking has been my skill and educational background. NO previous computer science nor mathematics background at all.

So I am exploring my new galaxy. I've earned certificates of Computer Science, Cryptography, Mathematics Logic, and Introduction to Artificial Intelligence with Python after finishing the online studies I enrolled in Harvard University Online, Stanford University Online, and Colorado University Online. I also have earned Introduction to Cybersecurity Tools & Cyber Attacks online study certificate authorized by IBM.

Why cryptography too? Recalling my childhood memory, my grandpa and I used to create unique symbols for all alphabets to write a letter for both of us alone. It makes me now realize that is a little defining moment why I am in love with cryptography.

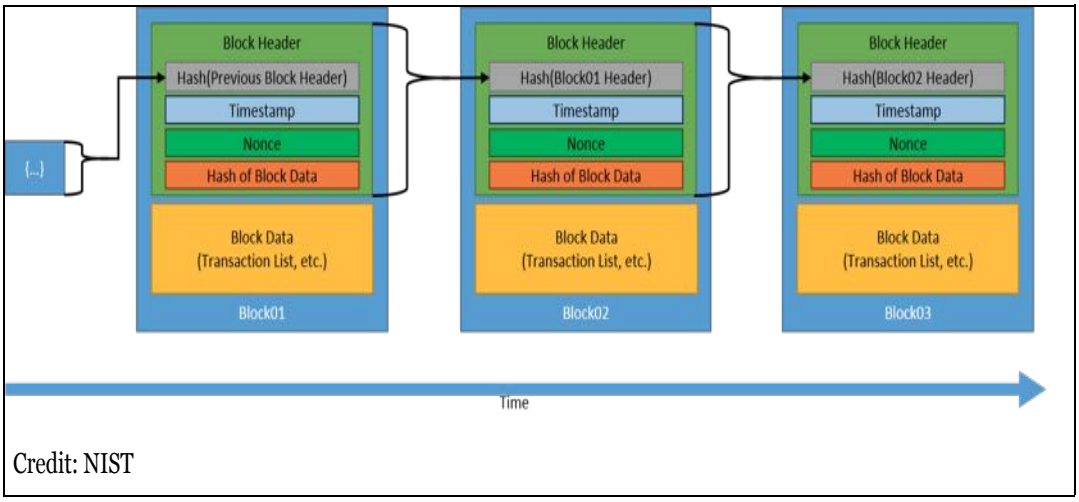
This book is a compilation of explanation and theory from many readings/sources and meant to be a fundamental introduction with most of quick examples I wrote. I also wrote quick examples by hand in the Cryptography chapter. The main purpose of this book is that I want this book to become a manual handbook I can always personally refer to at a new career in near future hopefully. And I'd like to humbly dedicate this book to my wife, my parents, my grandfather and grand mother who have been very supportive to me in billion ways. I do hope this new skill of mine and the book will be useful and meaningful in broader ways specially to my life and to also the country I was born in, United Republic of Indonesia (NKRI). I would like to dedicate this book to all good people around the globe as well. God bless you all.

Jakarta | March 2021 | Aryanadi IC


BHINNEKA TUNGGAL IKA

BLOCKCHAIN ARCHITECTURE

WHAT IS BLOCKCHAIN?



Blockchain technology is a new science within the field of computer network technology. It involves a solid computer science and cryptography knowledge. Blockchain technology needs an unbreakable CRYPTOGRAPHY to provide security for all users.

To me personally the function of blockchain technology is **to decentralize an event being performed by a participant in a transparent, safe and secure way for all participants in P2P network without a middleman in between.**

To explain what blockchain is in a simple way that I compiled from few sources to you, a blockchain is a chain of blocks that contain of information. Blocks that are linked together to form a chain.

A blockchain has properties that each block contains of **a data**, **hash** and **hash of previous block**. Once some data has been recorded inside the blockchain then it will become very difficult to change the data.

The **data** stored inside the block is different between the types of the blockchain. For example, a bitcoin blockchain stores the detail about an

event of transaction such as data of the sender, the receiver and the amount of a bitcoin. Bitcoin is one of the example that uses a public blockchain. But today we have a lot of public blockchains and a lot of companies are also getting together to build a private blockchain. Soon in near future blockchain technology could also be used for other things such as creating a medical record, governance votes, consensus protocol votes, e-notary and even collecting taxes.

The technical definition of blockchain I compiled from few sources are:

To repeat, a blockchain logically is made up of a chain of blocks. In bitcoin cases, a blockchain is a collaborative, tamper-resistant and distributed ledger that maintains transactional record that is completely open to everyone, who is the participant. The transactional records (data) are grouped into a block. A block is connected to the previous one by including a unique identifier that is based on the previous block's data.

Blocks are produced roughly every 10 minutes and are made up of transactions. This mechanism is called Proof of Work. Transactions are transfers of bitcoin from one account to another broadcasted on the network. Transactions within the blocks are ordered by the Miners (or we call it as the leader or the genesis blockchain or the block creator) according to the optional fee a participant includes as an incentive. The higher the fee, the more likely the transaction is included. Anyone in the network can run a mining node. Every miner has a copy of the same blockchain. The act of creating a block is called Mining. Blocks are organized chronologically by linking each block with the hash of the previous block. And again this technique makes the blockchain very secure.

According to Prof. Dan Boneh (a Cryptographer and lecture in Stanford University) is that Proof of Work is a mechanism from the first generation of blockchain and it was proven to be problematic because it was relatively too slow and it burned a lot of energy. We are moving

away from it and will be using new mechanism. There are Proof of Stake, Proof of Space and many more mechanism ideas coming which is much faster and they don't burn energy. So the future of blockchain is much more exciting.

In regards of security, blockchain uses a **hash**, meaning a CRYPTOGRAPHY to encode all of transactions so we can't really see exactly what happens but we know it happens. For an example of hash is: *71b04d9376a0385d6d9376a036a038571b04d9379869d0f35354a0*. You can compare a hash to a fingerprint. A hash identifies a block and all of its content and it is always unique just like a fingerprint.

Once a block has been created, its hash has been stored too. The hash changes wildly with any change of data. Another words, a hash is very useful for the participants when they need to detect a change of the block. If the hash changes then it is no longer the same block. And if the data is changed in one block, its unique identifier changes which can be seen in every subsequent block (providing tamper evidence). This domino effect allows all participants within the blockchain to know if a previous block's data has been tampered with. Since a blockchain network is difficult to alter or destroy, it provides a resilient method of collaborative record keeping.

The last property inside a block is the **hash of previous block**. This effectively creates a chain of blocks and this technique makes the blockchain so secure.

To quickly summarize, we can say blockchain technology is basically a new tool which is a computer network. In this case of bitcoin, blockchain technology allows all the participants in P2P network to trade one on one, and participants will have the same history of transactions. This technology updates the history by itself because the participants constantly verify and validate it. So all the participants own their transaction history.

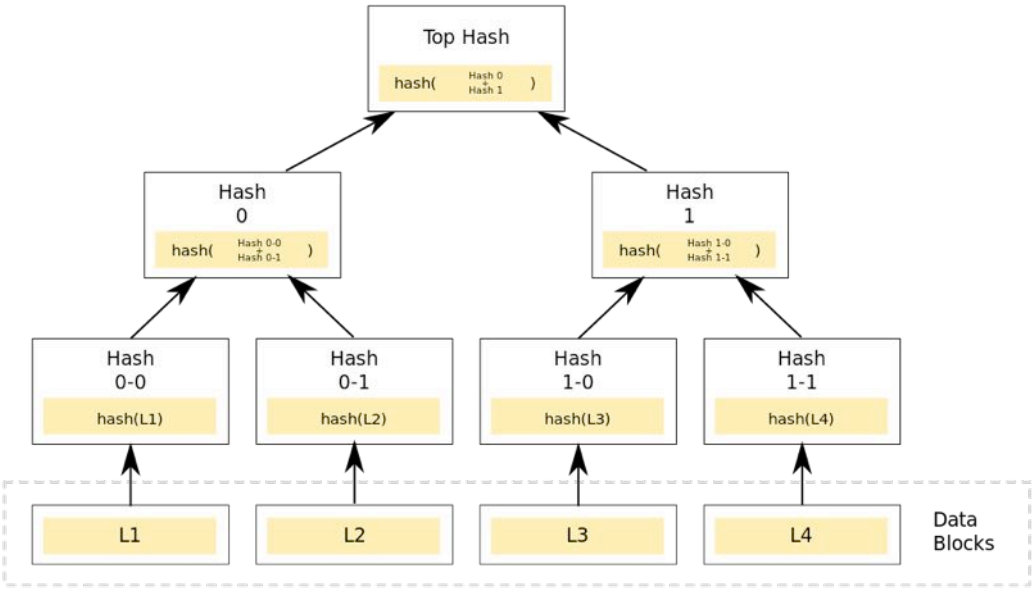
There is some kind of interface for the participants where the participants can have an app or they can use their computer to do the transaction. The participants's computer run the same software that guarantees their transactions as they happen. So no need of middleman anymore in between when doing the event called a transaction, in this case.

Blockchain represents a new paradigm for digital interactions and serves as the underlying technology for most crypto currencies. However, one of issues the crypto currencies bitcoin has been facing is a trust issues because one of the bitcoin problem has faced is bitcoin is getting stolen or lost. But many of them really come from the people trying to recentralize the bitcoin in different ways and making themselves pretty easy target to hack.

There are a lot of people today are working on how to create an identity structure that leverages blockchain and one of the tool for doing that is being able to CRYPTOGRAPHICALLY sign for a given attribute. So for example, a university could sign that someone is currently an enrolled student in the university. So could a government sign that someone is a truly citizen of the country.

Merkle Tree In cryptography and computer science, a hash tree or Merkle Tree is a tree in which every leaf node is labelled with the cryptographic hash of a data block, and every non-leaf node is labelled with the cryptographic hash of the labels of its child nodes. Hash trees allow efficient and secure verification of the contents of large data structures. Hash trees are a generalization of hash lists and hash chains.

The initial Bitcoin implementation of Merkle trees by Satoshi Nakamoto applies the compression step of the hash function to an excessive degree, which is mitigated by using Fast Merkle Trees.



(https://en.wikipedia.org/wiki/Merkle_tree)

Demonstrating that a leaf node is a part of a given binary hash tree requires computing a number of hashes proportional to the logarithm of the number of leaf nodes of the tree; this contrasts with hash lists, where the number is proportional to the number of leaf nodes itself. The concept of hash trees is named after Ralph Merkle, who patented it in 1979.

Hash trees can be used to verify any kind of data stored, handled and transferred in and between computers. They can help ensure that data blocks received from other peers in a peer-to-peer network are received undamaged and unaltered, and even to check that the other peers do not lie and send fake blocks. Hash trees are used in hash-based cryptography.

Blockchain Revolution to Trusts The internet that exists today (web 1.0 and web 2.0) allows global coordination through intermediaries that become trust layers for fellow foreigners to interact, such as from Facebook, Amazon, AirBNB, or Uber. Unfortunately, we become too

dependent or dependent on these platforms and they go from "attract" to "extract". Users (whether individual or business) can either incur high fees or face platform risk (the platform has the power to destroy other businesses running on it).

With the concept of Web 3.0, everyone including machines and businesses can do transaction, exchange values or information with strangers globally that they don't need to believe in, and without going through the middleman.

The biggest evolution occurring in Web 3.0 powered by blockchain is the minimization of trust required for global scale coordination. The concept of web 3.0 will fundamentally expand the scale and scope of interactions between humans and between machines, much larger than what we can currently imagine. These interactions range from seamless payments to a richer flow of information between the participants involved.

Web 3.0 will allow the participants to interact with machines anywhere in the world without having to go through middleman who usually charge large fees. This major change will allow for a new wave of business and new business models: from being a global corporate to a Decentralized Autonomous Organization, and a self-sovereign data marketplace. This is very important because:

* The community will become more efficient by disintermediating the industry, reducing the number of rent seeking middleman (the value captured is greater than the value given to the customer / user) and returning the extracted value directly to users and suppliers / contributors of the networks.
* Humans, enterprises / companies, and machines can share more data with higher privacy and security.
* Entrepreneurial activities and investing are more future proof, because they reduce dependence on certain platforms.

* We can own / hold our own data along with the digital footprint with the concept of tokenized digital assets and data that are scarce / scarce with the blockchain.
* If you think "self driving car" is a great concept and technology, in early January 2021, blockchain practitioners started talking about the term "self driving bank" or "autonomous bank" which is not just a concept but already exists and many of its users are application of blockchain in decentralized finance.

The Languages Behind Blockchain Blockchain technology gets best defined as a kind of distributed ledger technology that offers an optimum level of security for the stored data making it nearly implausible for anyone to cheat or hack the system. Being a form of distributed ledger technology, it has the following characteristics:

- Each participant of the distributed network has access to the ledger.
- Data once saved over the network can neither be modified nor reversed, which makes it immutable.
- Each piece of data in the network gets time stamped.
- The participants of the network are anonymous as far as the validity of records is concerned.
- Data gets encrypted.
- The technology is programmable.

There are some programming languages that are effective for blockchain:

Python Python has ruled the world of application development and blockchain is no exception to it. Python is listed among the official languages of Google and is also compatible with AI, Machine learning, Big Data, Internet of things, etc. Web3.py 5.17.0 documentation https://web3py.readthedocs.io/en/stable/ .

Fortunately, Python is an open-source programming language that makes it a preferred choice for reducing the overall cost of web

development. The list of excellent benefits of using Python for blockchain programming applications are as follows:

Python is highly stable, reliable, and comes with a gentle learning curve which makes it easier for a novice to master the language in much less time.

It offers the developers to go for pre-compilation of the code, which proves beneficial for the project. As it is a scripted language so compiling the code isn't a must for the developer's team, which saves time.

Python developers in India can develop a simple blockchain in less than 50 lines of code.

The versatility and speed performance of Python makes it a compelling option for blockchain development.

Solidity Solidity is an object-oriented, high-level language for implementing smart contracts. Smart contracts are programs which govern the behaviour of accounts within the Ethereum state. Solidity was influenced by Python, C++ and JavaScript and is designed to target the Ethereum Virtual Machine (EVM). Solidity is statically typed, supports inheritance, libraries and complex user-defined types among other features. With Solidity you can create contracts for uses such as voting, crowdfunding, blind auctions, and multi-signature wallets. Solidity 0.5.4 documentation https://docs.soliditylang.org/en/v0.5.4/ .

The other programming languanges are **Motoko**, **Move**, **Go**, **Java**, **C#**, **Ruby**, **Rholang**, and **Javascript**.

Blockchain is taking us to a new world to live. Blockchain technology is going to contribute how internet works very differently for us. And most importantly, **Blockchain technology saves a lot of money**, **doesn't need a vast amount of storage keeping**, and is changing the IT department to a whole different world. Blockchain technology is going

to free us from the dependency on banks and other government-controlled financial systems.

Blockchain happens to be an open-source technology that is shared between thousands of computers. In the bitcoin cases, these computers follow a set of rules to track money that's been sent from accounts tied to the blockchain software. All operations within a Blockchain network are carried out by the open-source community. And in the blockchain, I can send you money without using a middleman called a bank!

THE FATHER OF INDONESIAN CRYPTOGRAPHY
(BAPAK PERSANDIAN NEGARA KESATUAN REPUBLIK INDONESIA)

Mayjen TNI (Purn.) dr. Roebiono Kertopati (lahir di Ciamis, 11 April 1914 – meninggal 23 Juni 1984 pada umur 70 tahun) adalah Bapak Persandian Negara Republik Indonesia. Sebelumnya ia adalah seorang dokter di Kementerian Pertahanan RI bagian B (intelijen). Pada tanggal 4 April 1946, dr. Roebiono Kertopati dengan pangkat Letkol. mendapat perintah dari Amir Syarifuddin, Menteri Pertahanan RI saat itu, untuk mendirikan sebuah badan yang mengelola persandian nasional.

Mayjen TNI (Purn.) dr. Roebiono Kertopati tidak mengenyam pendidikan persandian secara formal, tetapi hanya berupa kursus singkat pengenalan sandi dari kementerian luar negeri Belanda pada tahun 1949. Namun melalui bacaan serta imajinasi, logika dan intuisi, diciptakanlah sistem-sistem sandi sendiri.

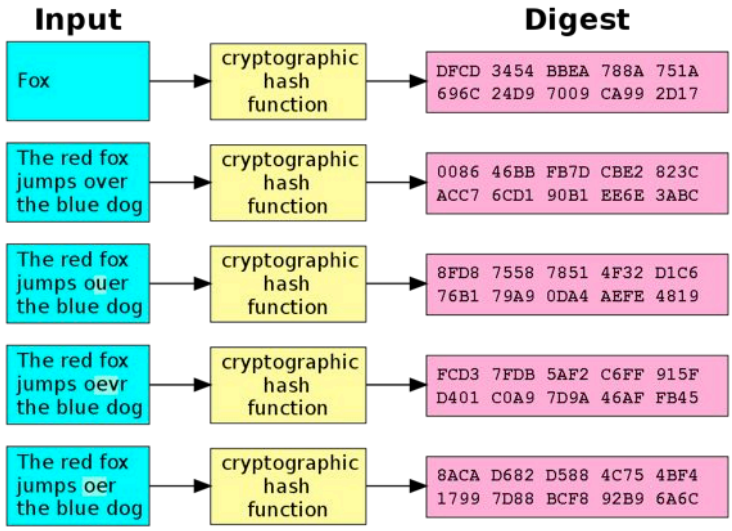
Major General (Ret.) Dr. Roebiono Kertopati (born in Ciamis, April 11, 1914 - died June 23, 1984 at 70 years old) is the Father of Indonesian Cryptography. He was a doctor at the Ministry of Defense of the Republic of Indonesia, part B (intelligence). On April 4, 1946 dr.

Roebiono Kertopati received orders from Amir Syarifuddin, the Minister of Defense of the Republic of Indonesia, to establish a national institution of cryptography.

Major General Dr. Roebiono Kertopati never received a formal cryptography education, but only a short course on the introduction level from the Dutch foreign ministry in 1949. However, by reading and using his mindful logic and intuition, he was able to invent his own cryptography for Indonesia country defense. (https://id.wikipedia.org/wiki/Roebiono_Kertopati).

CRYPTOGRAPHIC HASH FUNCTION

Cryptographic Hash Function



A cryptographic hash function (specifically SHA-1) at work. A small change in the input (in the word "over") drastically changes the output (digest). This is the so-called avalanche effect.

A cryptographic hash function (CHF) is a mathematical algorithm that maps data of arbitrary size (often called the "message") to a bit array of a fixed size (the "hash value", "hash", or "message digest"). It is a one-way function, that is, a function which is practically infeasible to invert. Ideally, the only way to find a message that produces a given hash is to attempt a brute-force search of possible inputs to see if they produce a match, or use a rainbow table of matched hashes. Cryptographic hash functions are a basic tool of modern cryptography.

The ideal cryptographic hash function has the following main properties:

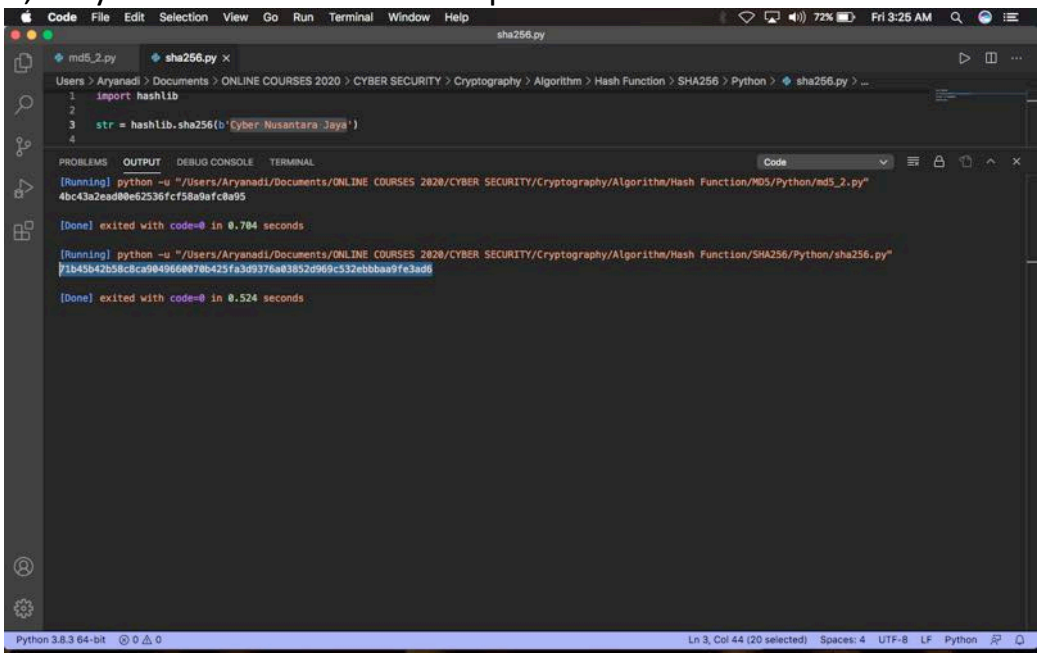
- it is deterministic, meaning that the same message always results in the same hash
- it is quick to compute the hash value for any given message
- it is infeasible to generate a message that yields a given hash value (i.e. to reverse the process that generated the given hash value)

- it is infeasible to find two different messages with the same hash value a small change to a message should change the hash value so extensively that the new hash value appears uncorrelated with the old hash value (avalanche effect)

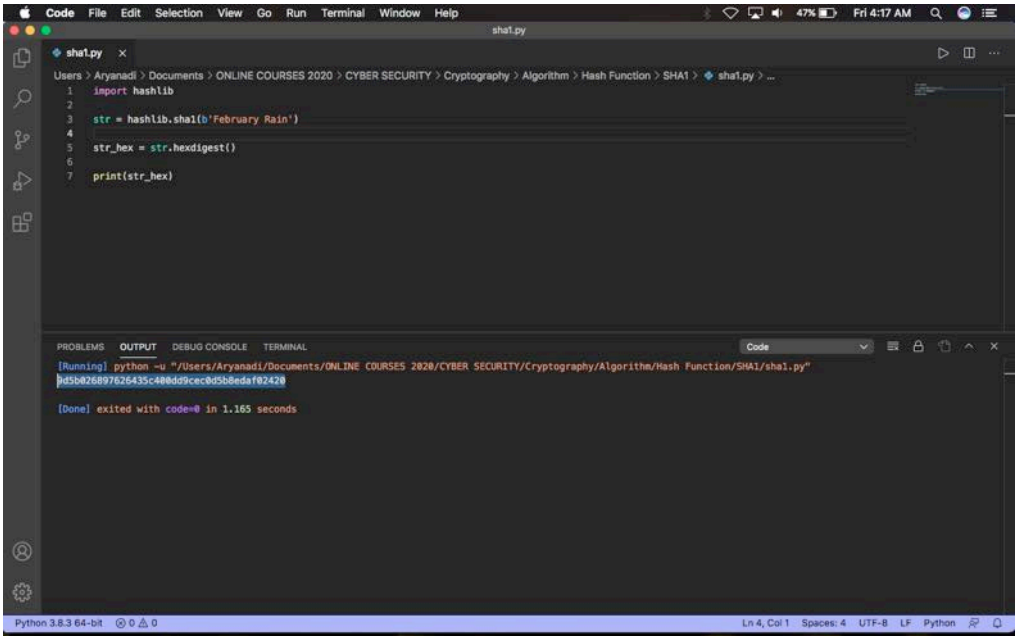
Cryptographic hash functions have many information-security applications, notably in digital signatures, message authentication codes (MACs), and other forms of authentication. They can also be used as ordinary hash functions, to index data in hash tables, for fingerprinting, to detect duplicate data or uniquely identify files, and as checksums to detect accidental data corruption. Indeed, in information-security contexts, cryptographic hash values are sometimes called (digital) fingerprints, checksums, or just hash values, even though all these terms stand for more general functions with rather different properties and purposes.

Cryptographic hash algorithms:
- MD5
- SHA-1 ; SHA = Secure Hash Algorithm
- SHA-2
- SHA-3
- RIPEMD-160
- Whirlpool
- BLAKE2
- BLAKE3

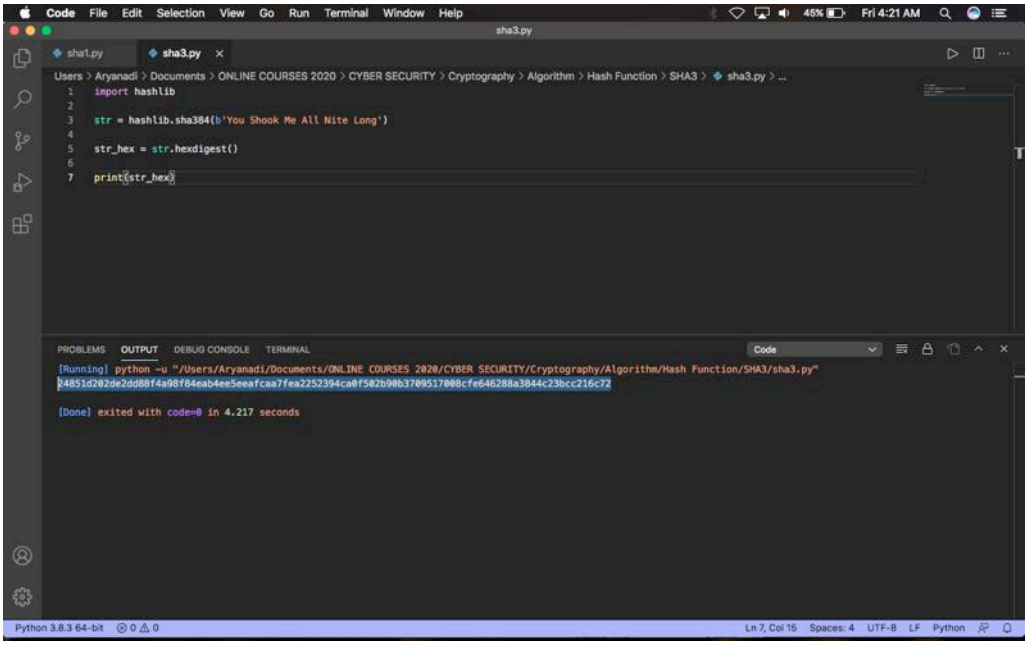
These are quick examples of how SHA256, SHA1 and SHA384 worked in Python with Visual Studio Code and in Online Tools on my MacBook Pro, they resulted in the same output:



SHA256 with Python in Visual Studio Code; text: Cyber Nusantara Jaya



SHA1 with Python in Visual Studio Code; text: February Rain

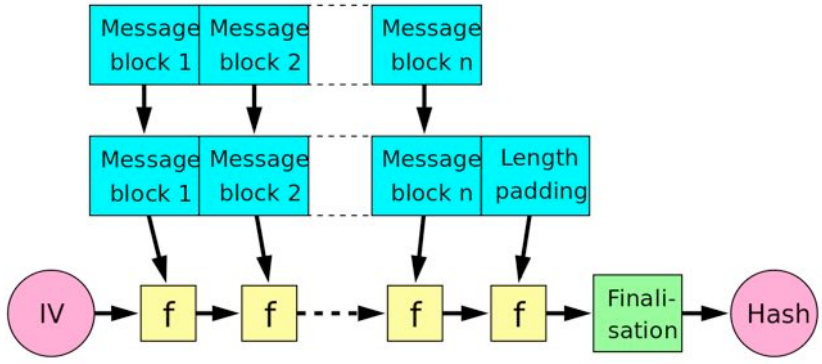


SHA384 with Python in Visual Studio Code; text: You Shook Me All Nite Long

Applications:

- Verifying the integrity of messages and files.
- Signature generation and verification.
- Password verification.
- Proof-of-work File or data identifier.

Hash function design:



Merkle-Damgård hash construction that is used inside almost all modern cryptographic hash functions. Illustration by David Göthberg, Sweden. Released by David as public domain. Remake and extension of Merkle-damgard.png made by Matt Crypto. Also added ideas from Merkle-Damgard.png by Mangojuice.

A hash function must be able to process an arbitrary-length message into a fixed-length output. This can be achieved by breaking the input up into a series of equal-sized blocks, and operating on them in sequence using a one-way compression function. The compression function can either be specially designed for hashing or be built from a block cipher.

A hash function built with the Merkle–Damgård construction is as resistant to collisions as is its compression function; any collision for the full hash function can be traced back to a collision in the compression function.

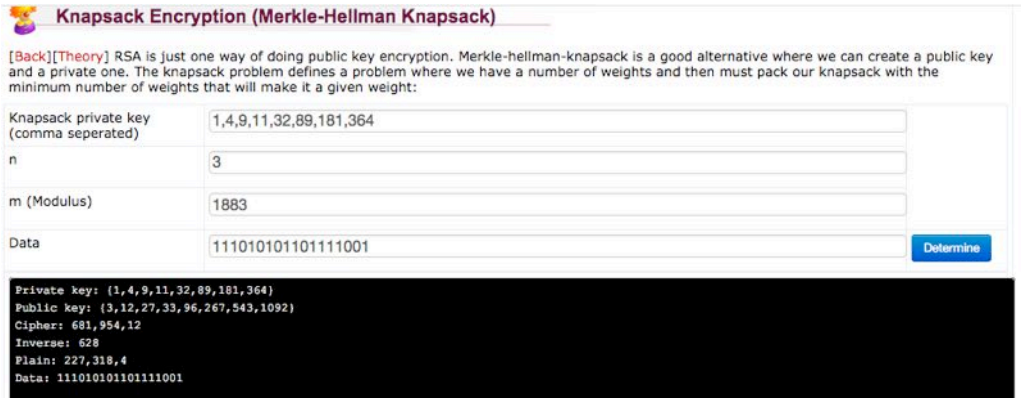
The last block processed should also be unambiguously length padded; this is crucial to the security of this construction. This construction is called the Merkle–Damgård construction. Most common classical hash functions, including SHA-1 and MD5, take this form.

Wide Pipe versus Narrow Pipe A straightforward application of the Merkle–Damgård construction, where the size of hash output is equal to the internal state size (between each compression step), results in a narrow-pipe hash design. This design causes many inherent flaws, including length-extension, multi collisions, long message attacks, generate-and-paste attacks, [citation needed] and also cannot be parallelized. As a result, modern hash functions are built on wide-pipe constructions that have a larger internal state size – which range from tweaks of the Merkle–Damgård construction to new constructions such as the sponge construction and HAIFA construction. None of the entrants in the NIST hash function competition use a classical Merkle–Damgård construction.

Meanwhile, truncating the output of a longer hash, such as used in SHA-512/256, also defeats many of these attacks (https://en.wikipedia.org/wiki/Cryptographic_hash_function#Attacks_on_cryptographic_hash_algorithms)

Merkle–Hellman Knapsack Merkle–Hellman is a public key cryptosystem, meaning that two keys are used, a public key for encryption and a private key for decryption. It is based on the subset sum problem (a special case of the knapsack problem). The problem is as follows: given a set of integers *A* and an integer *c*, find a subset of which sums to *c*. In general, this problem is known to be NP-complete. However, if *A* is superincreasing, meaning that each element of the set is greater than the sum of all the numbers in the set lesser than it, the problem is "easy" and solvable in polynomial time with a simple greedy algorithm.

Quick Example



Key generation: Choose a block size *n*. Integers up to *n* bits in length can be encrypted with this key. Create a key to encrypt 8-bit numbers by creating a random superincreasing sequence of 8 values:

$W = (1 + 4 + 9 + 11 + 32 + 89 + 181 + 364) = 691$

Now choose a random integer *q,* such that $q > W$:

$q = 1883$

Choose a random integer *r* such that gcd(*r*, *q*) = 1 that is r and q are coprime where *r* is any number/value between 1 to $W - 1$:

$r = 4 - 1 = 3$

So now we can see:

W = 1 + 4 + 9 + 11 + 32 + 89 + 181 + 364

q = 1883

r = 3

W = the superincreasing values, r = the private key, and q = the divisor.

Construct the public key B by multiplying each element in W by r modulo q, expressed as $B = (W \cdot r) \bmod q$, so that:

$(1 \cdot 3) \bmod 1883 = 3$

$(4 \cdot 3) \bmod 1883 = 12$

$(9 \cdot 3) \bmod 1883 = 27$

$(11 \cdot 3) \bmod 1883 = 33$

$(32 \cdot 3) \bmod 1883 = 96$

$(89 \cdot 3) \bmod 1883 = 267$

$(181 \cdot 3) \bmod 1883 = 543$

$(364 \cdot 3) \bmod 1883 = 1092$

Hence the public key B = (3 ,12 ,27, 33, 96, 267, 543, 1092).

Encryption: Let the 8-bit message be $m = 47 = 01000111_2$. We multiply each bit by the corresponding number in B and add the result. Use the ASCII Conversion table to convert each alphabet/character m in Hexadecimal into a binary number. Let's take ***c*** as the message m = 47 in Hexadecimal = 01000111 in Binary, thus

$0 \cdot 3 = 0$

$1 \cdot 12 = 12$

$0 \cdot 27 = 0$

$0 \cdot 33 = 0$

$0 \cdot 96 = 0$

$1 \cdot 267 = 267$

$1 \cdot 543 = 543$

$1 \cdot 1092 = 1092$ +

= **1914** is the ciphertext ***c***

Decryption: In Merkle–Hellman, decrypting a message requires solving an apparently "hard" knapsack problem. The private key contains a superincreasing list of numbers *W*, and the public key contains a non-superincreasing list of numbers *B*, which is actually a "disguised" version of *W*. The private key also contains some "trapdoor" information that can be used to transform a hard knapsack problem using *B* into an easy knapsack problem using *W*.

Unlike some other public key cryptosystems such as RSA, the two keys in Merkle-Hellman are not interchangeable; the private key cannot be used for encryption. Thus Merkle-Hellman is not directly usable for authentication by cryptographic signing, although Shamir published a variant that can be used for signing.

To decrypt the ciphertext *c*, we must find the subset of *B* which sums to *c*. We do this by transforming the problem into one of finding a subset of *W*. That problem can be solved in polynomial time since *W* is superincreasing.

1. Calculate the modular inverse of r modulo q using the Extended Euclidean algorithm. The inverse will exist since r is coprime to q

$r' = r^{-1} \mod q$

2. Calculate $c' = cr' \mod$

3. Solve the subset sum problem for c' using the superincreasing sequence W, by the simple greedy algorithm described below. Let X = ($x_1, x_2, \ldots, x_k$) be the resulting list of indexes of the elements of W which sum to c ’

$$\text{that is } m = \sum_{i=1}^{k} 2^{n - x1}$$

4. Construct the message *m* with a 1 in each x_i bit position and a 0 in all other bit positions:

$$m = \sum_{i=1}^{k} 2^{n - x1}$$

So to decrypt **1914**, first use the Extended Euclidean Algorithm to find the modular inverse of *r* mod *q*:

$r' = r^{-1} \bmod q = 3^{-1} \bmod 1883 = \mathbf{628}$

Step	**Q**	**t1**	**t2**	**t3**	**u1**	**u2**	**u3 (mod *q*)**	**v1**	**v2**	**v3 (r)**
0					1	0	1883	0	1	3
	Floor (u3/v3)	u1 – Q · v1	u2 – Q · v2	u3 – Q · v3						
1	627	1	-627	2	0	1	3	1	-627	2
2	1	-1	628	1	1	-627	2	-1	628	1
3	2	3	-1883	0	-1	628	1	-3	-1883	0
Mod Inverse										**628**

assumption for u1 = 1, u2 = 0 and v1 = 0, v2 = 1

Compute $c' = cr' \bmod q = 1914 \cdot 628 \bmod 1883 = 638$

Use the greedy algorithm to decompose 638 into a sum of values:

$c' = 638$

$W8 = 364 \leq 638$

$c' = 638 - 364 = 274$

$W3 = 9 \leq 274$

$c' = 274 - 9 = 265$

$W2 = 4 \leq 265$

$c' = 265 - 4 = 461$

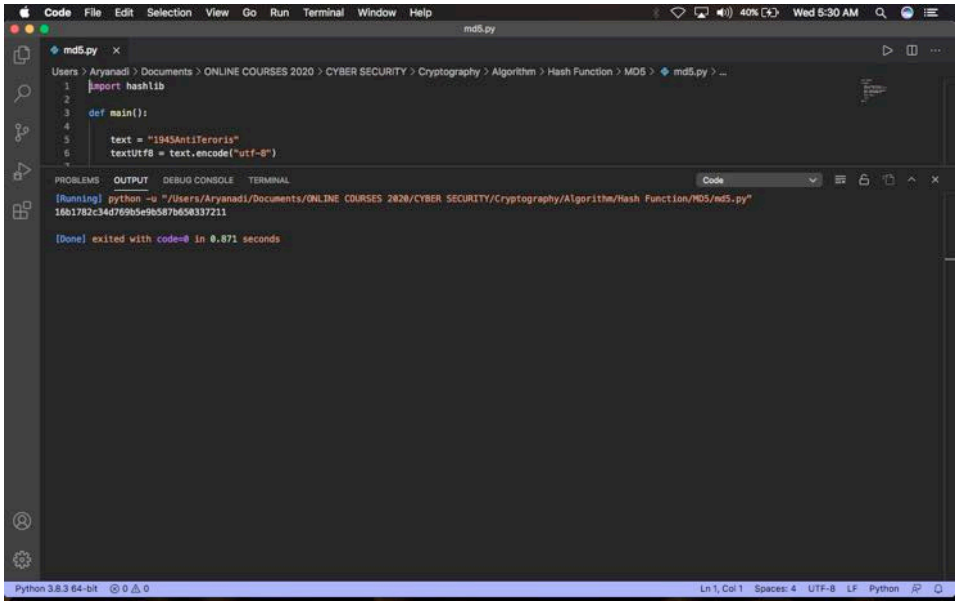
Thus 638 = 364 + 9 + 4 = *W*8 + *W*3 + *W*2, and the list of index is X = (8, 3, 2). The message can be computed as

3

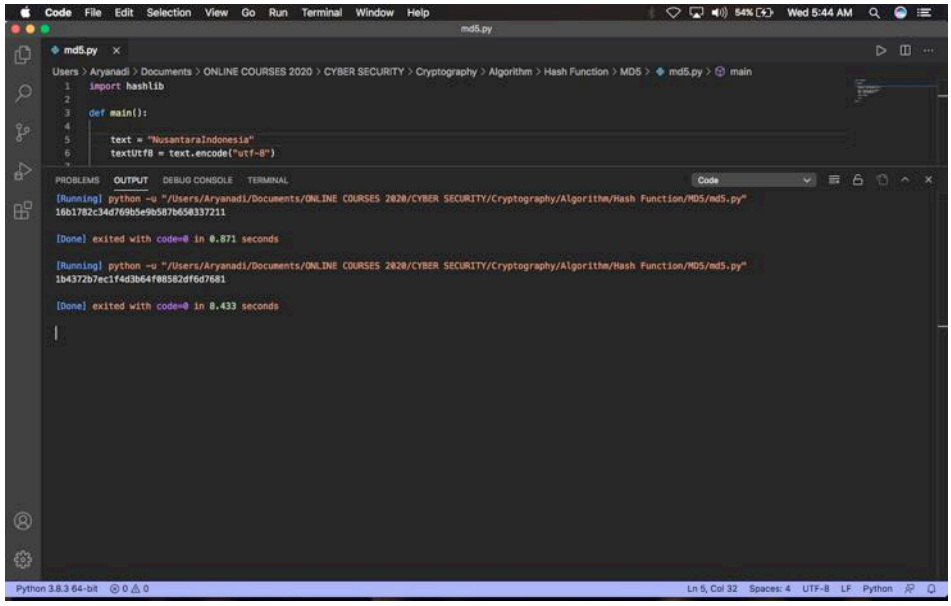
$$m = \sum_{i=1} 2^{n - x1}$$

The Merkle–Hellman Knapsack cryptosystem was one of the earliest public key cryptosystems. It was published by Ralph Merkle and Martin Hellman in 1978. polynomial time attack was published by Adi Shamir in 1984. As a result, the cryptosystem is now considered insecure. (Merkle–Hellman knapsack cryptosystem – Wikipedia).

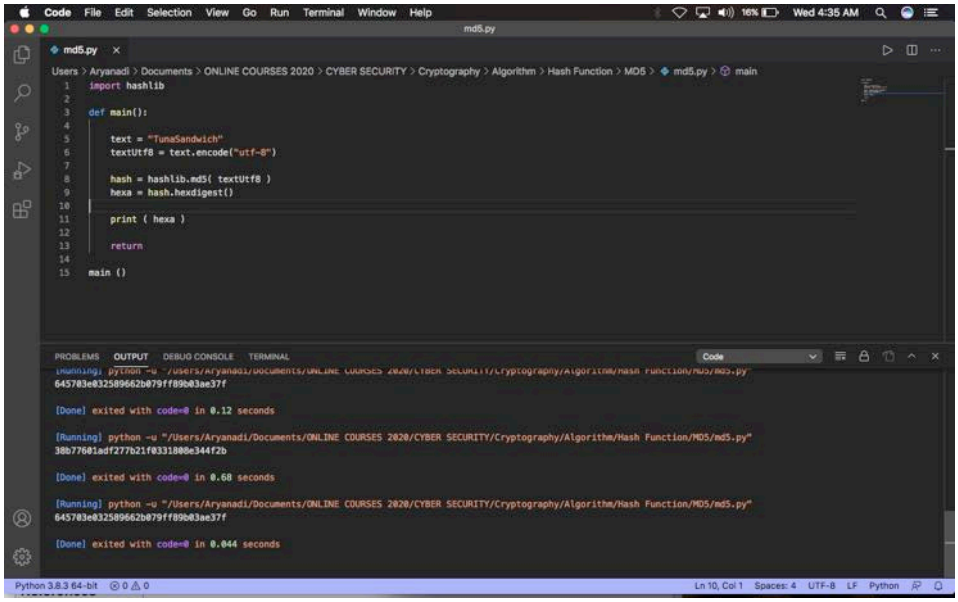
MD5 - Message-Digest Algorithm These are quick examples of how MD5 worked in Python with Visual Studio Code:



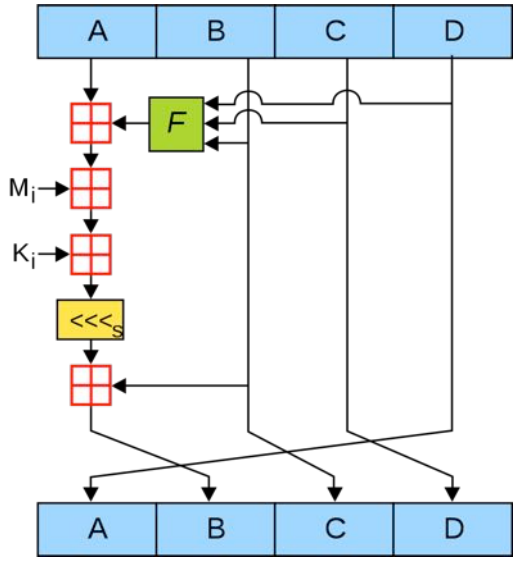
MD5 with Python in Visual Studio Code; text: 1945AntiTeroris



MD5 with Python in Visual Studio Code; text: NusantaraIndonesia



MD5 with Python in Visual Studio Code; text: TunaSandwich



One MD5 operation. MD5 consists of 64 of these operations, grouped in four rounds of 16 operations. F is a nonlinear function; one function is used in each round. Mi denotes a 32-bit block of the message input, and Ki denotes a 32-bit constant, different for each operation. <<<s denotes a left bit rotation by s places; s varies for each operation. denotes addition modulo 232.

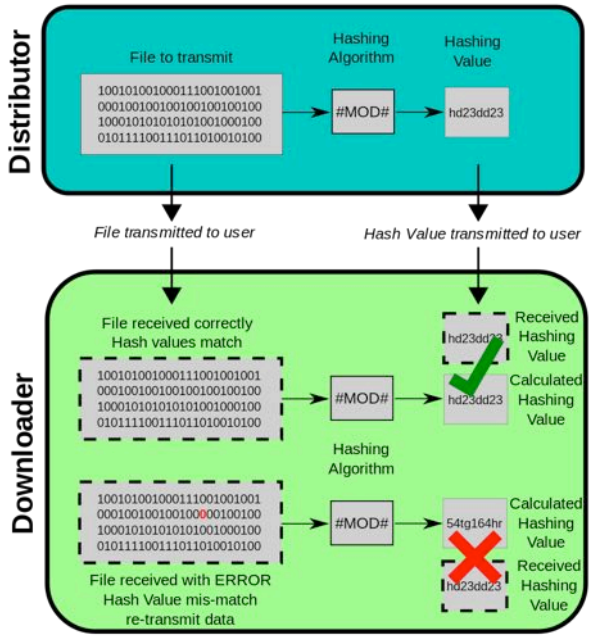
MD5 processes a variable-length message into a fixed-length output of 128 bits. The input message is broken up into chunks of 512-bit blocks (sixteen 32-bit words); the message is padded so that its length is divisible by 512. The padding works as follows: first a single bit, 1, is appended to the end of the message. This is followed by as many zeros as are required to bring the length of the message up to 64 bits fewer than a multiple of 512. The remaining bits are filled up with 64 bits representing the length of the original message, modulo 2 to the 64.

The main MD5 algorithm operates on a 128-bit state, divided into four 32-bit words, denoted A, B, C, and D. These are initialized to certain fixed constants. The main algorithm then uses each 512-bit message block in turn to modify the state. The processing of a message block consists of four similar stages, termed rounds; each round is composed of 16 similar operations based on a non-linear function F, modular addition, and left rotation. Figure 1 illustrates one operation within a

round. There are four possible functions; a different one is used in each round:

$$F(B, C, D) = (B \wedge C) \vee (\neg B \wedge D)$$
$$G(B, C, D) = (B \wedge D) \vee (C \wedge \neg D)$$
$$H(B, C, D) = B \oplus C \oplus D$$
$$I(B, C, D) = C \oplus (B \vee \neg D)$$

MD5 digests have been widely used in the software world to provide some assurance that a transferred file has arrived intact. For example, file servers often provide a pre-computed MD5 (known as md5sum) checksum for the files, so that a user can compare the checksum of the downloaded file to it. Most unix-based operating systems include MD5 sum utilities in their distribution packages; Windows users may use the included PowerShell function "Get-FileHash", install a Microsoft utility, or use third-party applications. Android ROMs also use this type of checksum.



As it is easy to generate MD5 collisions, it is possible for the person who created the file to create a second file with the same checksum, so this technique cannot protect against some forms of malicious tampering. In some cases, the checksum cannot be trusted (for example, if it was

obtained over the same channel as the downloaded file), in which case MD5 can only provide error-checking functionality: it will recognize a corrupt or incomplete download, which becomes more likely when downloading larger files.

Historically, MD5 has been used to store a one-way hash of a password, often with key stretching. NIST does not include MD5 in their list of recommended hashes for password storage.

MD5 is also used in the field of electronic discovery, in order to provide a unique identifier for each document that is exchanged during the legal discovery process. This method can be used to replace the Bates stamp numbering system that has been used for decades during the exchange of paper documents. As above, this usage should be discouraged due to the ease of collision attacks.

The MD5 message-digest algorithm is a widely used hash function producing a 128-bit hash value. Although MD5 was initially designed to be used as a cryptographic hash function, it has been found to suffer from extensive vulnerabilities. It can still be used as a checksum to verify data integrity, but only against unintentional corruption. It remains suitable for other non-cryptographic purposes, for example for determining the partition for a particular key in a partitioned database.

MD5 was designed by Ronald Rivest in 1991 to replace an earlier hash function MD4, and was specified in 1992 as RFC 1321. One basic requirement of any cryptographic hash function is that it should be computationally infeasible to find two distinct messages that hash to the same value. MD5 fails this requirement catastrophically; such collisions can be found in seconds on an ordinary home computer.

The weaknesses of MD5 have been exploited in the field, most infamously by the Flame malware in 2012. The CMU Software Engineering Institute considers MD5 essentially "cryptographically broken and unsuitable for further use". As of 2019, MD5 continues to

be widely used, in spite of its well-documented weaknesses and deprecation by security experts.

MD5 digests have been widely used in the software world to provide some assurance that a transferred file has arrived intact. For example, file servers often provide a pre-computed MD5 (known as md5sum) checksum for the files, so that a user can compare the checksum of the downloaded file to it. Most unix-based operating systems include MD5 sum utilities in their distribution packages; Windows users may use the included PowerShell function "Get-FileHash", install a Microsoft utility, or use third-party applications. Android ROMs also use this type of checksum (https://en.wikipedia.org/wiki/MD5#Applications).

ONE TIME PAD

According to Prof. Dan Boneh (Stanford University, U.S.A.)
One Time Pad has a PERFECT SECRECY because
element k in the set of K: E (K, M) = C) = 1.

Professor Boneh heads the applied cryptography group and co-direct the computer security lab. Professor Boneh's research focuses on applications of cryptography to computer security.

ONE TIME PAD

EXAMPLE A

Encryption $C_i = (P_i + K_i) \bmod 26$

Decryption $P_i = (C_i - K_i) \bmod 26$

C = ciphertext
P = plaintext
K = key
i = index

Plaintext = NusantarakuXJaya
Key = BlackXCoffeeXMix

Encryption →
(N+B) mod 26 = O
(u+l) mod 26 = f
(s+a) mod 26 = s
(a+c) mod 26 = c
(n+k) mod 26 = x
(t+X) mod 26 = q
(a+C) mod 26 = c
(r+o) mod 26 = f
(a+f) mod 26 = f
(k+f) mod 26 = p
(u+e) mod 26 = y
(x+e) mod 26 = b
(J+X) mod 26 = G
(a+M) mod 26 = m
(y+i) mod 26 = g
(a+x) mod 26 = x

$C_i = (P_i + K_i) \bmod 26$		
C	P	K
O : 14	N : 13	B : 1
f : 5	u : 20	L : 11
s : 18	s : 18	a : 0
c : 2	a : 0	c : 2
x : 23	n : 13	k : 10
q : 16	t : 19	X : 23
c : 2	a : 0	C : 2
f : 5	r : 17	o : 14
f : 5	a : 0	f : 5
p : 15	k : 10	f : 5
y : 24	u : 20	e : 4
b : 1	x : 23	e : 4
G : 6	J : 9	X : 23
m : 12	a : 0	M : 12
g : 6	y : 24	i : 8
x : 23	a : 0	x : 23

Ciphertext = OfscxqcffpybGmgx

Decryption ⟶

(O-B) mod 26 = N
(f-l) mod 26 = v
(s-a) mod 26 = s
(c-c) mod 26 = a
(x-k) mod 26 = n
(q-X) mod 26 = t
(c-c) mod 26 = a
(f-O) mod 26 = r
(f-f) mod 26 = a
(p-f) mod 26 = k
(y-e) mod 26 = u
(b-e) mod 26 = X
(G-x) mod 26 = J
(m-M) mod 26 = a
(g-i) mod 26 = y
(x-x) mod 26 = a

$P_i = (C_i - K_i) \bmod 26$		
P	C	K
N : 13	O : 14	B : 1
v : 20	f : 5	l : 11
s : 18	s : 18	a : 0
a : 0	c : 2	c : 2
n : 13	x : 23	k : 10
t : 19	q : 16	X : 23
a : 0	c : 2	C : 2
r : 17	f : 5	O : 14
a : 0	f : 5	f : 5
k : 10	p : 15	f : 5
u : 20	y : 24	e : 4
X : 23	b : 1	e : 4
J : 9	G : 6	X : 23
a : 0	m : 12	M : 12
y : 24	g : 6	i : 8
a : 0	x : 23	x : 23

Plaintext = Nusantaraku X jaya

OTP (2)

EXAMPLE B

Encryption $C_i = (P_i \oplus K_i)$
Decryption $P_i = (C_i \oplus K_i)$

Plaintext (P) = Oke
Key (K) = Yes

	Encryption		
	Plaintext (P)	Key (K)	Ciphertext (C)
ASCII	O k e	Y e s	M o w
Hexadecimal	4F 6B 65	59 65 73	4D 6F 77
Binary	4F = 01001111	59 = 01011001	4D = 01001101
	6B = 01101011	65 = 01100101	6F = 01101111
	65 = 01100101	73 = 01110011	77 = 01110111

	Decryption		
	Ciphertext (C)	Key (K)	Plaintext (P)
ASCII	M o w	Y e s	O k e
Hexadecimal	4D 6F 77	59 65 73	4F 6B 65
Binary	4D = 01001101	59 = 01011001	4F = 01001111
	6F = 01101111	65 = 01100101	6B = 01101011
	77 = 01110111	73 = 01110011	65 = 01100101

One Time Pad

One-time pads are "information-theoretically secure" in that the cipher text provides NO information about the original message to a cryptanalyst (except the maximum possible length of the message).

In this technique, a plaintext is paired with a random secret key (also referred to as a one-time pad). Then, each bit or character of the plaintext is encrypted by combining it with the corresponding bit or character from the pad using modular addition. The resulting cipher text will be impossible to decrypt or break if the following four conditions are met:

1. The key must be truly random.
2. The key must be at least as long as the plaintext.
3. The key must never be reused in whole or in part
4. The key must be kept completely secret.

It has also been proven that any cipher with the property of perfect secrecy must use keys with effectively the same requirements as OTP keys. Digital versions of one-time pad ciphers have been used by nations for critical diplomatic and military communication, but the problems of secure key distribution have made them impractical for most applications.

Given perfect secrecy, in contrast to conventional symmetric encryption, OTP is immune even to brute-force attacks. Trying all keys simply yields all plaintexts, all equally likely to be the actual plaintext. Even with known plaintext, like part of the message being known, brute-force attacks cannot be used, since an attacker is unable to gain any information about the parts of the key needed to decrypt the rest of the message. The parts that are known will reveal only the parts of the key corresponding to them, and they correspond on a strictly one-to-one basis; no part of the key is dependent on any other part.

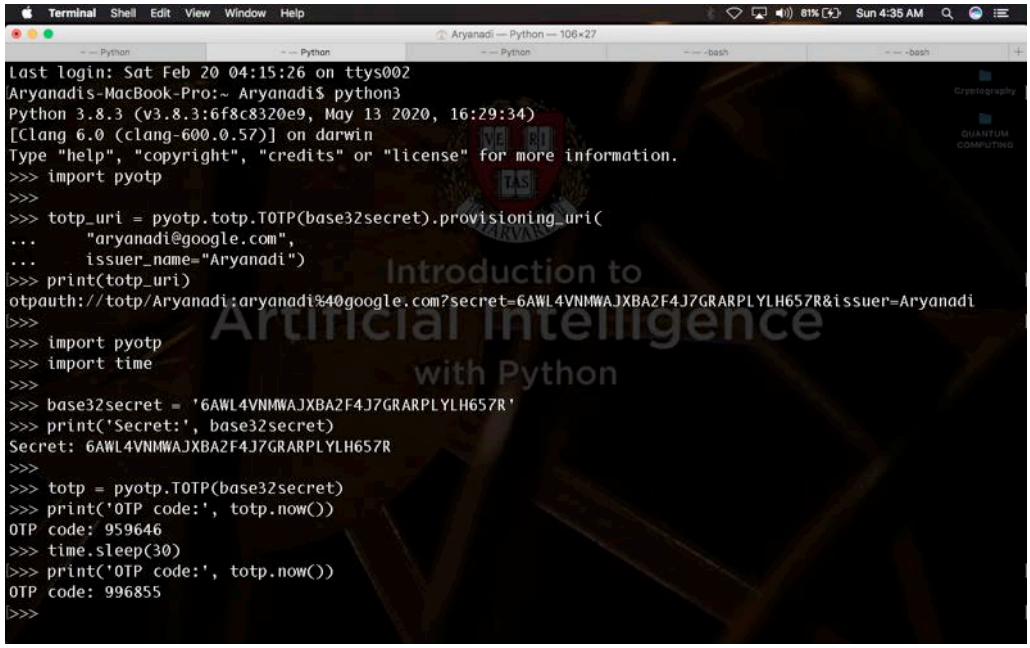
Properly used, one-time pads are secure in this sense even against adversaries with infinite computational power. If quantum computers are built with enough qubits, and overcoming some limitations to error-correction, some public key cryptography algorithms will become obsolete. One-time pads, however, will remain secure.

First described by Frank Miller in 1882, the one-time pad was re-invented in 1917. On July 22, 1919, U.S. Patent 1,310,719 was issued to Gilbert Vernam for the XOR operation used for the encryption of a one-time pad (https://en.wikipedia.org/wiki/One-time_pad).

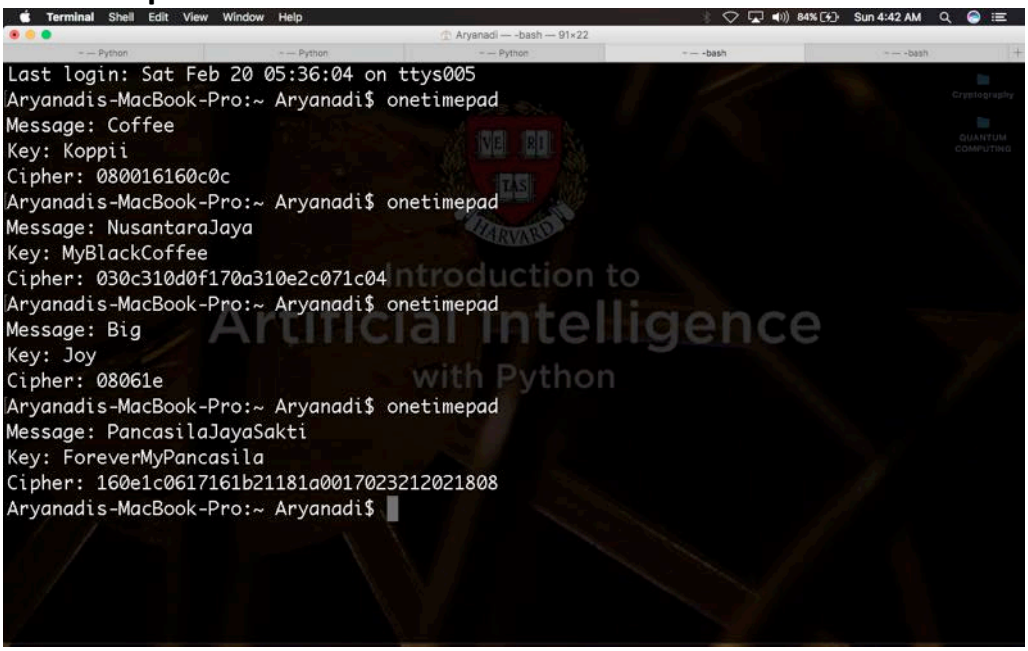
Quick Example 1

The Python library of our choice is PyOTP, which implement the RFC 4226 and RFC 6238 standards. If you want to use this library you should follow the requirements in those standards.

Let's create a base32 secret which has to be shared between the authentication server and the client. Once the client stores the secret in a secure way, in a time-interval of a 30 seconds (by default) a new code will be generated.



Quick Example 2



RSA
(RIVEST–SHAMIR–ADLEMAN)

RSA

EXAMPLE A

1. SETTING UP A KEY (TO GENERATE A PUBLIC KEY)

- $p = 7$, $q = 11$, ⟶ two distinc prime numbers
- $n = p \cdot q = 7 \cdot 11 = 77$ ⟶ Public Key
- $\Phi(n) = \text{lcm}(p-1 \cdot q-1)$

  $\Phi(n) = \text{lcm}(6 \cdot 10)$ ⟶ lcm = Least Common Multiple T

  $= 60$
- $e = 1 < e < \Phi(n)$ ⟶ choose an integer $1 < e < \Phi(n)$ co-prime to $\Phi(n)$

  $e = 1 < e < 60$

Divisor of 60 is 1, 2, 3, 4, 5, 6, 12, 15, 20, 30, & 60

$e = 7$, because GCD of 60 and 7 is 1 so 60 is co-prime to 7.

A	B	Q	R		
60	7	8	4		
7	4	1	3		
4	3	1	1		
3	1	1	0		
1	0				

GCD

$\text{GCD}(60,7) = (60,7)\ 7 \times 8 + 4$

$= (7,4)\ 4 \times 1 + 3$

$= (4,1)\ 3 \times 1 + 1$

$= (1)$

- HOW TO GENERATE A PRIVATE KEY

d = modular multicative inverse of $e \pmod{\varphi(n)}$

$7 \pmod{60}$

$$d = \frac{1 + (k \times (\varphi n)}{e} = \frac{1 + (1 \times 60)}{7} = \frac{61}{7} = 8,71 \text{xxxxx}$$

$$= \frac{1 + (2 \times 60)}{7} = \frac{121}{7} = 17,28 \text{xxxx}$$

$$= \frac{1 + (3 \times 60)}{7} = \frac{181}{7} = 25,85 \text{xxxx}$$

$$= \frac{1 + (4 \times 60)}{7} = \frac{241}{7} = 34,42 \text{xxxx}$$

$$= \frac{1 + (5 \times 60)}{7} = \frac{301}{7} = 43$$

If d is not an integer, compute it again by replacing k with positive number 2,3, 4,5,6,7,... until d results in an integer.

PRIVATE				PUBLIC	
p	q	$\varphi(n)$	d	e	n
7	11	60	43	7	77

2. ENCRYPTION

- Plaintext (m) = GNR
- Encrytion $c(m) = m^e \pmod n$
- Numerical representation in alphabet for the plaintext is

$G = 7$, $N = 14$, $R = 18$

- G encryption $7^7 \pmod{77} = 7^{\wedge}(1+2+2+2) \bmod 77$

$= 7^1 \pmod{77} = 7 \times 1 \bmod 77$

$= 7 \bmod 77 = 7$

$= 7^2 \pmod{77} = 7 \times 7 \bmod 77$

$= 49 \bmod 77 = 49$

$= 7^7 \pmod{77} = (7 \times 49 \times 49 \times 49) \bmod 77$

$= (7 \bmod 77) \times (49 \times 49 \bmod 77) \times (49 \bmod 77)$

$= 7 \times (2401 \bmod 77 = 14) \times 49$

$= (7 \times 14 \times 49) \bmod 77$

$= 4802 \bmod 77 = (28)$

$77 \times 62 = 4774 + 28 = 4802$

OR ALTERNATIVELY

$= 7^7 \pmod{77} = 823{,}543 \bmod 77 = 28$

$77 \times 10{,}695 = 823{,}515$

$= 823{,}543 - 823{,}515$

$= 28$

Ciphertext for G = 28

- N encryption $14^7 \pmod{77} = 14\hat{}(2+2+3) \bmod 77$

$= 14^2 \pmod{77} = (14 \times 14) \bmod 77 = 196 \bmod 77$
$= 42$

$= 14^3 \pmod{77} = (14 \times 14 \times 14) \bmod 77 = 2744 \bmod 77$
$= 49$

$= 14^5 \pmod{77} = (42 \times 49) \bmod 77$
$= 2058 \bmod 77 = 56$

$= 14^7 \pmod{77} = (56 \times 42) \bmod 77$
$= 2352 \bmod 77 = (42)$

Ciphertext for N = 42

- R encryption $18^7 \pmod{77} = 18\hat{}(2+2+3) \bmod 77$

$= 18^2 \pmod{77} = (18 \times 18) \bmod 77$
$= 324 \bmod 77 = 16$

$= 18^3 \pmod{77} = (18 \times 18 \times 18) \bmod 77$
$= 5,832 \bmod 77 = 57$

$= 18^5 \pmod{77} = (16 \times 57) \bmod 77$
$= 912 \bmod 77 = 65$

$= 18^7 \pmod{77} = (65 \times 16) \bmod 77$
$= 1040 \bmod 77 = (39)$

Ciphertext for R = 39

3. DECRYPTION

- Ciphertext 28, 42, 39 to decryt
- Decryption $m(c) = m^d \bmod n$
- 28 decryption $= 28^{43} (\text{mod } 77)$

$28^3 (\text{mod } 77) = 21952 \bmod 77 = 7 \longrightarrow 77 \times 285 = 21945 + 7$

$28^5 (\text{mod } 77) = 17210368 \bmod 77 = 21 \longrightarrow 77 \times 223511 = 17210347 + 21$

$28^{10} (\text{mod } 77) = (21 \times 21) \bmod 77$

$= 441 \bmod 77 = 56 \longrightarrow 77 \times 5 + 56$

$28^{20} (\text{mod } 77) = (56 \times 56) \bmod 77$

$= 3136 \bmod 77 = 56 \longrightarrow 77 \times 40 + 56$

$28^{43} (\text{mod } 77) = (56 \times 56 \times 7) \bmod 77$

$= 21952 \bmod 77 = (7)$

7 is the numerical representation in alphabet for plaintext (m) = G

- 42 decryption $= 42^{43} (\text{mod } 77)$

$42^3 (\text{mod } 77) = 74088 \bmod 77 = 14 \longrightarrow 77 \times 962 + 14$

$42^5 (\text{mod } 77) = 130691232 \bmod 77 = 56 \longrightarrow 77 \times 1697288 + 56$

$42^{10} (\text{mod } 77) = (56 \times 56) \bmod 77 = 56$

$= 3136 \bmod 77 = 56 \longrightarrow 77 \times 40 + 56$

$42^{20} (\text{mod } 77) = (56 \times 56) \bmod 77 = 56$

$42^{43} (\text{mod } 77) = (56 \times 56 \times 14) \bmod 77$

$= 43904 \bmod 77 = (14) \longrightarrow 77 \times 570 + 14$

14 is the numerical representation in alphabet for plaintext (m) = N

- 39 decryption $39^{43} \pmod{77}$

$39^3 \pmod{77} = 59319 \bmod 77 = 29 \longrightarrow 77 \times 770 + 29$

$39^5 \pmod{77} = 90224199 \bmod 77 = 65 \longrightarrow 77 \times 1171742 + 65$

$39^{10} \pmod{77} = (65 \times 65) \bmod 77$

$= 4225 \bmod 77 \quad = 67 \longrightarrow 77 \times 54 + 67$

$39^{20} \pmod{77} = (67 \times 67) \bmod 77$

$= 4489 \bmod 77 \quad = 23 \longrightarrow 77 \times 58 + 23$

$39^{43} \pmod{77} = (23 \times 23 \times 29) \bmod 77$

$= 15341 \bmod 77 = (18)$

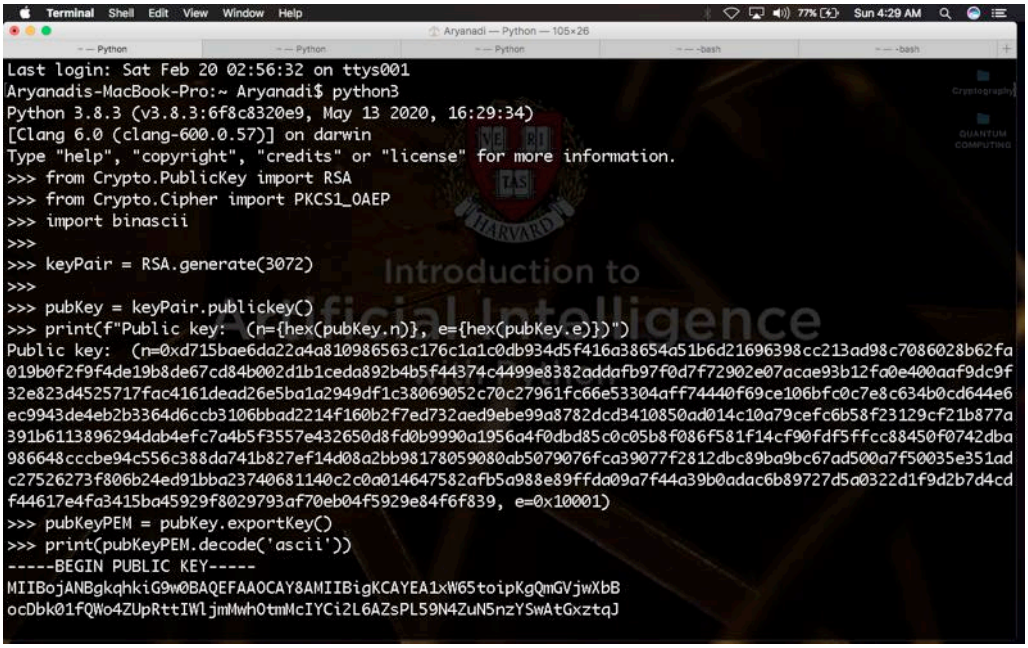
18 is the numerical representation in alphabet for plaintext (m) = R

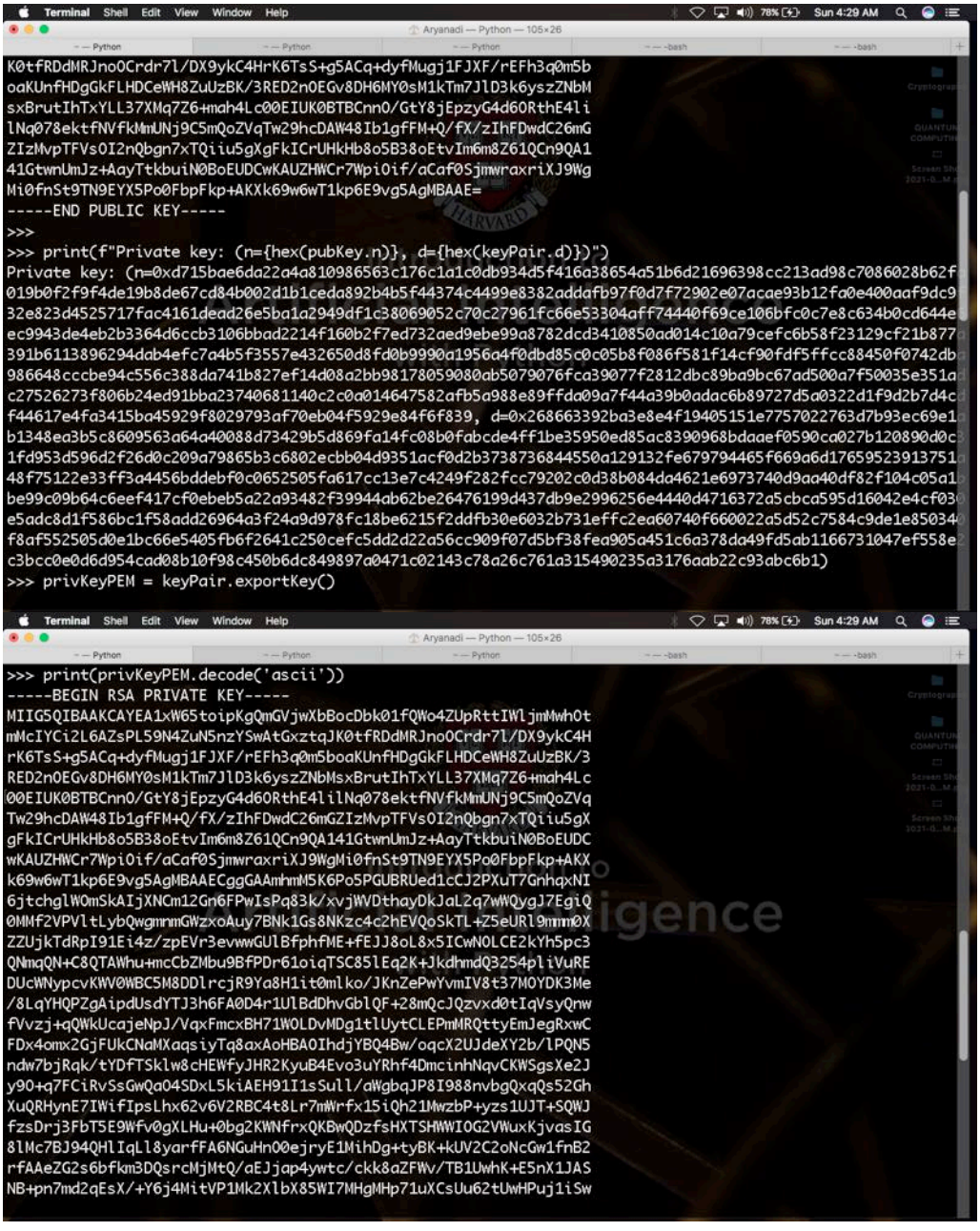
RSA (Rivest–Shamir–Adleman) Cryptosystem

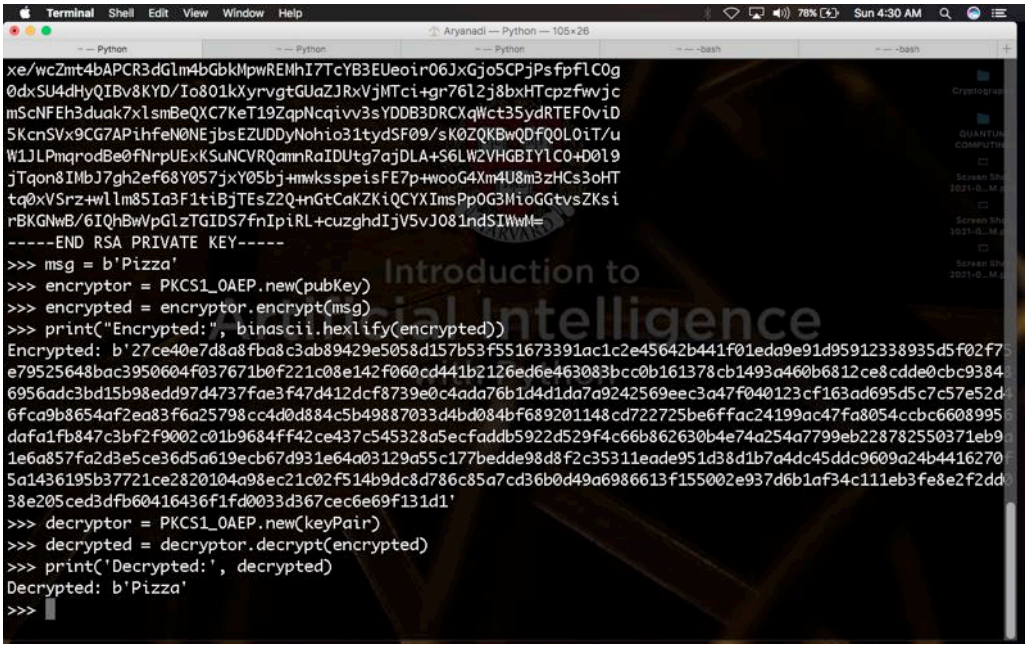
Ron Rivest, Adi Shamir, and Leonard Adleman at the Massachusetts Institute of Technology, made several attempts over the course of a year to create a one-way function that was hard to invert. They publicly described the algorithm in 1977. In such a cryptosystem, the encryption key is public and distinct from the decryption key which is kept secret (private). In RSA, this asymmetry is based on the practical difficulty of factoring the product of two large prime numbers, the "factoring problem".

Quick Example

Let's generate the RSA keys (1024-bit) and print them on the console (as hex numbers and in the PKCS#8 PEM ASN.1 format). Let me use short key length to keep the sample input short, but in a real world it is recommended to use 3072-bit or 4096-bit keys. Encrypt the message using RSA-OAEP encryption scheme (RSA with PKCS#1 OAEP padding) with the RSA public key. Finally, decrypt the message using using RSA-OAEP with the RSA private key:




Terminal Shell Edit View Window Help
Sun 4:29 AM
Aryanadi — Python — 105×26
K0tfRDdMRJno0Crdr7l/DX9ykC4HrK6TsS+g5ACq+dyfMugj1FJXF/rEFh3q0m5b
oaKUnfHDgGkFLHDCeWH8ZuUzBK/3RED2n0EGv8DH6MY0sM1kTm7JlD3k6yszZNbM
sxBrutIhTxYLL37XMq7Z6+mah4Lc00EIUK0BTBCnn0/GtY8jEpzyG4d60RthE4li
lNq078ektfNVfkMmUNj9C5mQoZVqTw29hcDAW48Ib1gfFM+Q/fX/zIhFDwdC26mG
ZIzMvpTFVsOI2nQbgn7xTQiiu5gXgFkICrUHkHb8o5B38oEtvIm6m8Z61QCn9QA1
41GtwnUmJz+AayTtkbuiN0BoEUDCwKAUZHWCr7Wpi0if/aCaf0SjmwraxriXJ9Wg
Mi0fnSt9TN9EYX5Po0FbpFkp+AKXk69w6wT1kp6E9vg5AgMBAAE=
-----END PUBLIC KEY-----
>>>
>>> print(f"Private key: (n={hex(pubKey.n)}, d={hex(keyPair.d)})")
Private key: (n=0xd715bae6da22a4a810986563c176c1a1c0db934d5f416a38654a51b6d21696398cc213ad98c7086028b62f
019b0f2f9f4de19b8de67cd84b002d1b1ceda892b4b5f44374c4499e8382addafb97f0d7f72902e07acae93b12fa0e400aaf9dc9
32e823d4525717fac4161dead26e5ba1a2949df1c38069052c70c27961fc66e53304aff74440f69ce106bfc0c7e8c634b0cd644e
ec9943de4eb2b3364d6ccb3106bbad2214f160b2f7ed732aed9ebe99a8782dcd3410850ad014c10a79cefc6b58f23129cf21b877
391b6113896294dab4efc7a4b5f3557e432650d8fd0b9990a1956a4f0dbd85c0c05b8f086f581f14cf90fdf5ffcc88450f0742db
986648cccbe94c556c388da741b827ef14d08a2bb98178059080ab5079076fca39077f2812dbc89ba9bc67ad500a7f50035e351a
c27526273f806b24ed91bba2374068114 0c2c0a01647582afb5a988e89ffda09a7f44a39b0adac6b89727d5a0322d1f9d2b7d4c
f44617e4fa3415ba45929f8029793af70eb04f5929e84f6f839, d=0x268663392ba3e8e4f19405151e7757022763d7b93ec69e1
b1348ea3b5c8609563a64a40088d73429b5d869fa14fc08b0fabcde4ff1be35950ed85ac8390968bdaaef0590ca027b120890d0c
1fd953d596d2f26d0c209a79865b3c6802ecbb04d9351acf0d2b373873684455 0a129132fe679794465f669a6d17659523913751
48f75122e33ff3a4456bddebf0c0652505fa617cc13e7c4249f282fcc79202c0d38b084da4621e6973740d9aa40df82f104c05a1
be99c09b64c6eef417cf0ebeb5a22a93482f39944ab62be26476199d437db9e2996256e4440d4716372a5cbca595d16042e4cf03
e5adc8d1f586bc1f58add26964a3f24a9d978fc18be6215f2ddfb30e6032b731effc2ea60740f660022a5d52c7584c9de1e85034
f8af552505d0e1bc66e5405fb6f2641c250cefc5dd2d22a56cc909f07d5bf38fea905a451c6a378da49fd5ab1166731047ef558e
c3bcc0e0d6d954cad08b10f98c450b6dc849897a0471c02143c78a26c761a315490235a3176aab22c93abc6b1)
>>> privKeyPEM = keyPair.exportKey()
Terminal Shell Edit View Window Help
Sun 4:29 AM
Aryanadi — Python — 105×26
>>> print(privKeyPEM.decode('ascii'))
-----BEGIN RSA PRIVATE KEY-----
MIIG5QIBAAKCAYEA1xW65toipKgQmGVjwXbBocDbk01fQWo4ZUpRttIWljmMwh0t
mMcIYCi2L6AZsPL59N4ZuN5nzYSwAtGxztqJK0tfRDdMRJno0Crdr7l/DX9ykC4H
rK6TsS+g5ACq+dyfMugj1FJXF/rEFh3q0m5boaKUnfHDgGkFLHDCeWH8ZuUzBK/3
RED2n0EGv8DH6MY0sM1kTm7JlD3k6yszZNbMsxBrutIhTxYLL37XMq7Z6+mah4Lc
00EIUK0BTBCnn0/GtY8jEpzyG4d60RthE4lilNq078ektfNVfkMmUNj9C5mQoZVq
Tw29hcDAW48Ib1gfFM+Q/fX/zIhFDwdC26mGZIzMvpTFVsOI2nQbgn7xTQiiu5gX
gFkICrUHkHb8o5B38oEtvIm6m8Z61QCn9QA141GtwnUmJz+AayTtkbuiN0BoEUDC
wKAUZHWCr7WpiOif/aCaf0SjmwraxriXJ9WgMi0fnSt9TN9EYX5Po0FbpFkp+AKX
k69w6wT1kp6E9vg5AgMBAAECggGAAmhmM5K6Po5PGUBRUed1cCJ2PXuT7GnhqxNI
6jtchglWOmSkAIjXNCm12Gn6FPwIsPq83k/xvjWVDthayDkJaL2q7wWQygJ7EgiQ
0MMf2VPVltLybQwgmnmGWzxoAuy7BNk1Gs8NKzc4c2hEVQoSkTL+Z5eURl9mmm0X
ZZUjkTdRpI91Ei4z/zpEVr3evwwGUlBfphfME+fEJJ8oL8x5ICwN0LCE2kYh5pc3
QNmqQN+C8QTAWhu+mcCbZMbu9BfPDr61oiqTSC85lEq2K+JkdhmdQ3254pliVuRE
DUcWNypcvKWV0WBC5M8DDlrcjR9Ya8H1it0mlko/JKnZePwYvmIV8t37MOYDK3Me
/8LqYHQPZgAipdUsdYTJ3h6FA0D4r1UlBdDhvGblQF+28mQcJQzvxd0tIqVsyQnw
fVvzj+qQWkUcajeNpJ/VqxFmcxBH71WOLDvMDg1tlUytCLEPmMRQttyEmJegRxwC
FDx4omx2GjFUkCNaMXaqsiyTq8axAoHBAOIhdjYBQ4Bw/oqcX2UJdeXY2b/lPQN5
ndw7bjRqk/tYDfTSklw8cHEWfyJHR2KyuB4Evo3uYRhf4DmcinhNqvCKWSgsXe2J
y90+q7FCiRvSsGwQaO4SDxL5kiAEH91I1sSull/aWgbqJP8I988nvbgQxqQs52Gh
XuQRHynE7IWifIpsLhx62v6V2RBC4t8Lr7mWrfx15iQh21MwzbP+yzs1UJT+SQWJ
fzsDrj3FbT5E9Wfv0gXLHu+0bg2KWNfrxQKBwQDzfsHXTSHWWIOG2VWuxKjvasIG
8lMc7BJ94QHlIqLl8yarfFA6NGuHn00ejryE1MihDg+tyBK+kUV2C2oNcGw1fnB2
rfAAeZG2s6bfkm3DQsrcMjMtQ/aEJjqp4ywtc/ckk8aZFWv/TB1UwhK+E5nX1JAS
NB+pn7md2qEsX/+Y6j4MitVP1Mk2XlbX85WI7MHgMHp71uXCsUu62tUwHPuj1iSw



The output will be different because it generates different random RSA key-pair at each execution. Even if I encrypt the same message several times with the same public key, I will get different output. This is because the OAEP padding algorithm injects some randomness with the padding. If I try to encrypt larger messages, I will get and exception, because the 1024-bit key limits the maximum message length.

RSA is a relatively slow algorithm, and because of this, it is less commonly used to directly encrypt user data. More often, RSA passes encrypted shared keys for symmetric key cryptography which in turn can perform bulk encryption-decryption operations at much higher speed.

The RSA algorithm involves four steps: key generation, key distribution, encryption and decryption.

RSA produces a *public key* and a *private key*. The public key can be known by everyone, and it is used for encrypting messages. The intention is that messages encrypted with the public key can only be decrypted in a reasonable amount of time by using the private key. The public key is represented by the integers *n* and *e*; and, the private key, by the integer *d* (although *n* is also used during the decryption process.

Thus, it might be considered to be a part of the private key, too). *m* represents the message (https://en.wikipedia.org/wiki/RSA_(cryptosystem)).

As of 2017, the largest RSA number that has been factored is RSA 768, a 768-bit number, which has 232 decimal digits. Factored back in 2009, it took a couple of years and involved hundreds of processors. It is estimated that it would take about 2000 years on a single core 2.2 Gigahertz processor. A couple of smaller RSA numbers still haven't been factored.

To give you some idea of how processing capabilities have progressed, the original RSA key length for RSA in the late 1970s was 512 bits. As is often the case in cryptography, the desired key length was longer than this but the choice of 512 gets reflected practical contemporary implementation constraints. The first 512-bit key factor, RSA 155, involved a dedicated research effort, a close to state-of-the-art Cray supercomputer, and 8000 MIPS years of processing effort. A decade later, the 512-bit RSA key used by Texas Instruments designed firmware for the TI-83+ graphing calculator was broken by a single person, Benjamin Moody, in 73 days using a single 1.9 Gigahertz dual core processor.

Currently, RSA encryption commonly uses 1024 bit public keys. But it is increasingly recommended to use 2048 bit keys and some applications use 4096 bit keys. These are numbers that have about 300, 600 and 1200 decimal digits, respectively. Since these keys are semi-prime numbers, which are numbers that are the products of two primes, and since the two prime numbers used are roughly the same size. This means that we are talking about prime numbers that have approximately 150, 300 and 600 digits, respectively. While the largest prime number we know to date has over 22 million digits, it is estimated that the limit below which all of the prime numbers have been found is somewhere between 10 to the 18th and 10 to the 19th, or basically an 18-digit number.

Some experts suspect that current 1024-bit keys can already be factored in meaningful timeframes by well-funded state actors, while others dispute this claim. But few doubt that it will be possible in the not too distant future.

VIGENÈRE CIPHER

VIGENÈRE CIPHER

Algebraic description

If the letters A-Z are taken to be numbers 0-25 ($A \hat{=} 0, B \hat{=} 1, C \hat{=} 2, D \hat{=} 3$, and so on...), and addition is performed modulo 26, Vigenère encryption E using the key K can be written as follow:

$$C_i = E_K(M_i) = (M_i + K_i) \bmod 26$$

and decryption D using the key K as follow:

$$M_i = D_K(C_i) = (C_i - K_i + 26) \bmod 26$$

Alphabet	A	B	C	D	E	F	G	H	I	J	K	L	M
Numerical Representation	0	1	2	3	4	5	6	7	8	9	10	11	12
Alphabet	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Numerical Representation	13	14	15	16	17	18	19	20	21	22	23	24	25

Example

Plaintext / Message (M): Anti Terrorist

Key (K) : Nkri

Plaintext	A	N	T	I	T	E	R	R	O	R	I	S	T
Num. Rep.	0	13	19	8	19	4	17	17	14	17	8	18	19
Key	N	K	R	I	N	K	R	I	N	K	R	I	N
Num. Rep.	13	10	17	8	13	10	17	8	13	10	17	8	13
Ciphertext	N	X	K	Q	G	O	I	Z	B	B	Z	A	G
Num Rep.	13	23	36	16	32	14	34	25	27	27	25	26	32

To recover the plaintext is to substract Ciphertext with Key.

Vigenère Cipher

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Vigenere-square Table

The Vigenère cipher (French pronunciation: [viʒnɛːʁ]) is a method of encrypting alphabetic text by using a series of interwoven Caesar ciphers, based on the letters of a keyword. The Vigenère cipher is classed as a type of polyalphabetic substitution.

First described by Giovan Battista Bellaso in 1553, the cipher is easy to understand and implement, but it resisted all attempts to break it until 1863, three centuries later. This earned it the description le chiffre indéchiffrable (French for 'the indecipherable cipher'). Many people have tried to implement encryption schemes that are essentially Vigenère ciphers. In 1863, Friedrich Kasiski was the first to publish a general method of deciphering Vigenère ciphers (https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher).

CAESAR CIPHER

CAESAR CIPHER

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Encryption of a letter x by a shift n described as:

Caesar's Encryption: $E_n(x) = (x + n) \bmod 26$

Caesar's Decryption: $D_n(x) = (x - n) \bmod 26$

The encryption represented using modular arithmetic: A→0, B→1...

The transformation can be represented by aligning 2 alphabets; the cipher alphabet is the plain alphabet ROTATED LEFT or RIGHT by some number of positions. For instance:

Plain : Tuna Sandwich

Cipher : Vwoc Ucpfykej

Above is a caesar cipher using a right rotation of 2 places.

Plain : Tuna Sandwich

Cipher : Rsly Qylbugaf

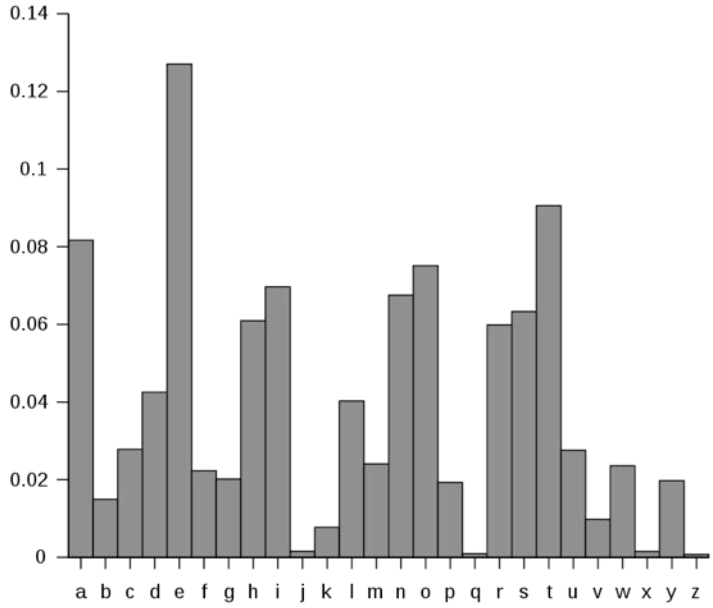
Above is a caesar cipher using a left rotation of 2 places.

When encrypting, a person looks up each letter of the message in the plain alphabet line and writes down the corresponding letter in the cipher alphabet line.

Decrypting is done in reverse with a left or right shift of n.

Caesar cipher is classed as a type of monoalphabet substitution.

Caesar Cipher



The distribution of letters in a typical sample of English language text has a distinctive and predictable shape. A Caesar shift "rotates" this distribution, and it is possible to determine the shift by examining the resultant frequency graph.

In cryptography, a Caesar cipher, also known as Caesar's cipher, the shift cipher, Caesar's code or Caesar shift, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a left shift of 3, D would be replaced by A, E would become B, and so on. The method is named after Julius Caesar, who used it in his private correspondence. The cipher is classed as a type of monoalphabetic substitution.

The encryption step performed by a Caesar cipher is often incorporated as part of more complex schemes, such as the Vigenère cipher, and still has modern application in the ROT13 system. As with all single-alphabet substitution ciphers, the Caesar cipher is easily broken and in modern practice offers essentially no communications security.

The Caesar cipher is named after Julius Caesar, who, according to Suetonius, used it with a shift of three (A becoming D when encrypting, and D becoming A when decrypting) to protect messages of military significance. While Caesar's was the first recorded use of this scheme, other substitution ciphers are known to have been used earlier.

Caesar His nephew, Augustus, also used the cipher, but with a right shift of one, and it did not wrap around to the beginning of the alphabet, "Whenever he wrote in cipher, he wrote B for A, C for B, and the rest of the letters on the same principle, using AA for Z." — Suetonius, Life of Augustus.

It is unknown how effective the Caesar cipher was at the time, but it is likely to have been reasonably secure, not least because most of Caesar's enemies would have been illiterate and others would have assumed that the messages were written in an unknown foreign language. There is no record at that time of any techniques for the solution of simple substitution ciphers. The earliest surviving records date to the 9th-century works of Al-Kindi in the Arab world with the discovery of frequency analysis (https://en.wikipedia.org/wiki/Caesar_cipher).

A E S
(ADVANCED ENCRYPTION STANDARD)

AES
128 BITS

Plaintext : Big = 42 69 67 (in hexadecimal)
Key : joy = 6a 6f 79 (in hexadecimal)

B	0	0	0
i	0	0	0
g	0	0	0
0	0	0	0

0 = zero = 00 = zero zero

Plaintext input bytes

42	0	0	0
69	0	0	0
67	0	0	0
0	0	0	0

0 = zero = 00 = zero zero

Plaintext state array

j	0	0	0
o	0	0	0
y	0	0	0
0	0	0	0

0 = zero = 00 = zero zero

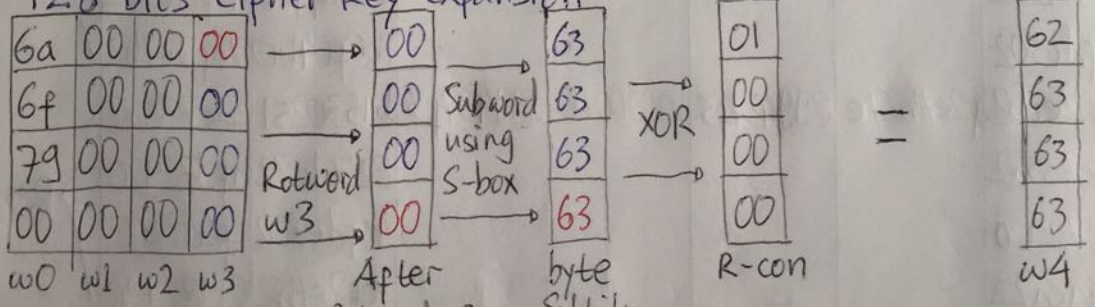
Key input bytes

6a	0	0	0
6f	0	0	0
79	0	0	0
0	0	0	0

0 = zero = 00 = zero zero

Key state array

128 bits cipher key expansion



aes 9

Word	Temp	After Rot Word	After Sub-word by S-Box	Rcon [i/Nk]	After XOR with Rcon	w[i-Nk]	w[i] = temp XOR w[i-Nk]	
0	6a6f7900							
1	00000000							
2	00000000							
3	00000000							
4	00000000	00000000	63636363	01000000	62636363	6a6f7900	080c1a63	
5	080c1a63					00000000	080c1a63	R1
6	080c1a63					00000000	080c1a63	
7	080c1a63					00000000	080c1a63	
8	080c1a63	0c1a6308	fea2fb30	02000000	fca2fb30	080c1a63	f4aee153	
9	f4aee153					080c1a63	fca2fb30	R2
10	fca2fb30					080c1a63	f4aee153	
11	f4aee153					080c1a63	fca2fb30	
12	fca2fb30	a2fb30fc	3a0f04b0	04000000	3e0f04b0	f4aee153	caa1e5e3	
13	caa1e5e3					fca2fb30	3603ed3	R3
14	3603ed3					f4aee153	c2adff80	
15	c2adff80					fca2fb30	3e0f04b0	
16	3e0f04b0	0f04b03e	76f2e7b2	08000000	7ef2e7b2	caa1e5e3	b4530251	
17	b4530251					3603ed3	82501c81	R4
18	82501c82					c2adff80	40fde302	
19	40fde302					3e0f04b0	7ef2e7b2	
20	7ef2e7b2	f2e7b27e	8994374f3	10000000	9994374f3	b4530251	2dc735a2	
21	2dc735a2					82501c82	af972920	R5
22	af972920					40fde302	ef6aca22	
23	ef6aca22					7ef2e7b2	91982d90	
24	91982d90	982d9091	46d86081	20000000	66d86081	2dc735a2	4b1f5523	
25	4b1f5523					af972920	e4887c03	R6
26	e4887c03					ef6aca22	0be2b621	
	0be2b631					91982d90	9a7a9bb1	

Word	Temp	After RotWord	After Subword by S-BOX	Rcon [i/Nk]	After XOR with Rcon	w[i-Nk]	w[i] = temp XOR w[i-Nk]	
24				20000000				
25								
26								
27								
28	9a7a9bb1	7a9bb19a	da14c8b8	40000000	9a14c8b8	4b1f5523	d10b9d9b	R7
29	d10b9d9b					e4887c03	3583e198	
30	3583e198					0be2b621	3e6157b9	
31	3e6157b9					9a7a9bb1	a41bcc08	
32	a41bcc08	1bcc08a4	af4b3049	80000000	2f4b3049	d10b9d9b	fe40add2	R8
33	fe40add2					3583e198	cbc34c4a	
34	cbc34c4a					3e6157b9	f5a21bf3	
35	f5a21bf3					a41bcc08	51b9d7fb	
36	51b9d7fb	b9d7fb51	560e0fd1	1b000000	4d0e0fd1	fe40add2	b34ea203	R9
37	b34ea203					cbc34c4a	788dee49	
38	788dee49					f5a21bf3	8d2ff5ba	
39	8d2ff5ba					51b9d7fb	dc962241	
40	dc962241	962241dc	90938386	36000000	a6938386	b34ea203	15dd2185	R10
41	15dd2185					788dee49	6d50cfcc	
42	6d50cfcc					8d2ff5ba	e07f3a76	
43	e07f3a76					dc962241	3ce91837	

Cipher

start of Round 0

Input Plaintext = 42 69 67 00
00 00 00 00 00 00 00 00
00 00 00 00

42	00	00	00
69	00	00	00
67	00	00	00
00	00	00	00

round key value 0

Cipher Key = 6a 6f 79 00
00 00 00 00 00 00 00 00
00 00 00 00

6a	00	00	00
6f	00	00	00
79	00	00	00
00	00	00	00

28	00	00	00
06	00	00	00
1e	00	00	00
00	00	00	00

1. Round 1

28	00	00	00
06	00	00	00
1e	00	00	00
00	00	00	00

start of round

34	63	63	63
6f	63	63	63
72	63	63	63
63	63	63	63

after sub bytes with S-box

34	63	63	63
63	63	63	6f
63	63	72	63
63	63	63	63

after shift rows

cd	63	72	77
34	63	50	7b
34	63	41	6f
9a	63	72	6f

after mix columns

⊕

08	08	08	08
0c	0c	0c	0c
1a	1a	1a	1a
63	63	63	63

round key value

$(02.34) \oplus (03.63) \oplus (01.63) \oplus (01.63)$

$= (x(00110100) \oplus (x+1)(01100011) \oplus (01100011) \oplus (01100011)$

$= (x^6+x^5+x^3) \oplus (x^7+x^6+x^2+x+x^6+x^5+x+1) \oplus (x^6+x^5+x+1) \oplus$
(x^6+x^5+x+1)

$= x^7+x^6+x^3+x^2+1$

$= 11001101 = cd$

$(02.63) \oplus (03.63) \oplus (01.63) \oplus (01.63)$

$= (x(01100011)) \oplus (x+1(01100011)) \oplus (01100011) \oplus (01100011)$

$= (x^7+x^6+x^2+x) \oplus (x^7+x^6+x^2+x+x^6+x^5+x+1) \oplus (x^6+x^5+x+1) \oplus$
(x^6+x^5+x+1)

$= x^6+x^5+x+1$

$= 01100011 = 63$

$(02.63) \oplus (03.63) \oplus (01.72) \oplus (01.63)$

$= (x(00110011)) \oplus (x+1(01100011)) \oplus (01110010) \oplus (01100011)$

$= (x^7+x^6+x^2+x) \oplus (x^7+x^6+x^2+x+x^6+x^5+x+1) \oplus (x^6+x^5+x^4+x)$
$\oplus (x^6+x^5+x+1)$

$= x^6+x^5+x^4+x$

$= 01110010 = 72$

$(02.63) \oplus (03.6f) \oplus (01.63) \oplus (01.63)$

$= (x(01100011)) \oplus (x+1(01101111)) \oplus (01100011) \oplus (01100011)$

$= (x^7+x^6+x^2+x) \oplus (x^7+x^6+x^4+x^3+x^2+x+x^6+x^5+x^3+x^2+x+1) \oplus$
$(x^6+x^5+x+1) \oplus (x^6+x^5+x+1)$

$= x^6+x^5+x^4+x^2+x+1$

$= 01110111 = 77$

des 5

(01.34) ⊕ (02.63) ⊕ (03.63) ⊕ (01.63)

= (00110100) ⊕ (x(01100011)) ⊕ (x+1(01100011)) ⊕ (01100011)

= (x5+x4+x2) ⊕ (x7+x6+x2+x) ⊕ (x7+x6+x2+x+x6+x5+x+1) ⊕ (x6+x5+x+1)

= x5 + x4 + x2 = 00110100 = 34

(01.63) ⊕ (02.63) ⊕ (03.63) ⊕ (01.63)

= (01100011) ⊕ (x(01100011)) ⊕ (x+1(01100011)) ⊕ (01100011))

= (x6+x5+x+1) ⊕ (x7+x6+x2+x) ⊕ (x7+x6+x2+x+x6+x5+x+1) ⊕ (x6+x5+x+1)

= x6 + x5 + x + 1

= 01100011 = 63

(01.63) ⊕ (02.63) ⊕ (03.72) ⊕ (01.63)

= (01100011) ⊕ (y(01100011)) ⊕ (x+1(01110010)) ⊕ (01100011)

= (x6+x5+x+1) ⊕ (x7+x6+x2+x) ⊕ (x7+x6+x5+x2+x6+x5+x4+x) ⊕ (x6+x5+x+1)

= x6 + x4

= 01010000 = 50

(01.63) ⊕ (02.6f) ⊕ (03.63) ⊕ (01.63)

= (01100011) ⊕ (x(01101111)) ⊕ (x+1(01100011)) ⊕ (01100011)

= (x6+x5+x+1) ⊕ (x7+x6+x4+x3+x2+x) ⊕ (x7+x6+x2+x+x6+x5+x+1) ⊕ (x6+x5+x+1)

= x6 + x5 + x4 + x3 + x + 1 = 01111011 = 7b

(01.34) ⊕ (01.63) ⊕ (02.63) ⊕ (03.63)

= (00110100) ⊕ (01100011) ⊕ (x(01100011)) ⊕ (x+1(01100011))

= (x5+x4+x2) ⊕ (x6+x5+x+1) ⊕ (x7+x6+x2+x) ⊕
(x7+x6+x2+x+x6+x5+x+1)

= x5 + x4 + x2

= 00110100 = 34

(01.63) ⊕ (01.63) ⊕ (02.63) ⊕ (03.63)

= (01100011) ⊕ (01100011) ⊕ (x(01100011)) ⊕ (x+1(01100011))

= (x6+x5+x+1) ⊕ (x6+x5+x+1) ⊕ (x7+x6+x2+x) ⊕
(x7+x6+x2+x6+x5+x+1)

= x6 + x5 + x + 1

= 01100011 = 63

(01.63) ⊕ (01.63) ⊕ (02.72) ⊕ (03.63)

= (01100011) ⊕ (01100011) ⊕ (x(01110010)) ⊕ (x+1(01100011))

= (x6+x5+x+1) ⊕ (x6+x5+x+1) ⊕ (x7+x6+x5+x2) ⊕
(x7+x6+x2+x+x6+x5+x+1)

= x6 + 1

= 01000001 = 41

(01.63) ⊕ (01.6f) ⊕ (02.63) ⊕ (03.63)

= (01100011) ⊕ (01101111) ⊕ (x(01100011)) ⊕ (x+1(01100011))

= (x6+x5+x+1) ⊕ (x6+x5+x3+x2+x+1) ⊕
(x7+x6+x2+x) ⊕ (x7+x6+x2+x+x6+x5+x+1)

= x6 + x5 + x3 + x2 + x + 1

= 01101111 = 6f

$(03.34) \oplus (01.63) \oplus (01.63) \oplus (02.63)$

$= (x+1(00110100)) \oplus (01100011) \oplus (01100011) \oplus (x(01100011))$

$= (x^6+x^5+x^3+x^5+x^4+x^2) \oplus (x^6+x^5+x+1) \oplus (x^6+x^5+x+1) \oplus$
$(x^7+x^6+x^2+x)$

$= x^7+x^4+x^3+x$

$= 10011010 = 9a$

$(03.63) \oplus (01.63) \oplus (01.63) \oplus (02.63)$

$= (x+1(01100011)) \oplus (01100011) \oplus (01100011) \oplus (x(01100011))$

$= (x^7+x^6+x^2+x+x^6+x^5+x+1) \oplus (x^6+x^5+x+1) \oplus (x^6+x^5+x+1) \oplus$
$(x^7+x^6+x^2+x)$

$= x^6+x^5+x+1$

$= 01100011 = 63$

$(03.63) \oplus (01.63) \oplus (01.72) \oplus (02.63)$

$= (x+1(01100011)) \oplus (01100011) \oplus (01110010) \oplus (x(01100011))$

$= (x^7+x^6+x^2+x+x^6+x^5+x+1) \oplus (x^6+x^5+x+1) \oplus$
$(x^6+x^5+x^4+x) \oplus (x^7+x^6+x^2+x)$

$= x^6+x^5+x^4+x$

$= 01110010 = 72$

$(03.63) \oplus (01.6f) \oplus (01.63) \oplus (02.63)$

$= (x+1(01100011)) \oplus (01101111) \oplus (01100011) \oplus (x(01100011))$

$= (x^7+x^6+x^2+x+x^6+x^5+x+1) \oplus (x^6+x^5+x^3+x^2+x+1) \oplus$
$(x^6+x^5+x+1) \oplus (x^7+x^6+x^2+x)$

$= x^6+x^5+x^3+x^2+x+1 = 01101111 = 6f$

2. ROUND 2

Start of Round

c5	6b	7a	7f
38	6f	5c	77
2e	79	5b	75
f9	00	11	0c

After SubBytes with S-Box

a6	7f	da	d2
07	a8	4a	f5
31	b6	39	9d
99	63	82	fe

After Shift Rows

a6	7f	da	d2
a8	4a	f5	07
39	9d	31	b6
fe	99	63	82

After Mix Columns

73	24	f9	82
58	ce	1b	9f
65	a4	e8	3f
87	7f	77	c3

⊕ XOR

Round Key Value

f4	fc	f4	fc
ae	a2	ae	a2
e1	fb	e1	fb
53	30	53	30

3. ROUND 3

Start of Round

87	d8	0d	7e
f6	6c	b5	3d
84	9f	09	c4
d4	4f	24	f3

#After SubBytes with S-Box

17	61	d7	f3
42	50	d5	27
5f	cf	01	1c
48	84	36	0d

#After Shift Rows

17	61	d7	f3
50	d5	27	42
01	1c	5f	cf
0d	48	84	36

#After Mix Columns

d2	f2	07	c7
b9	bc	fc	0b
52	54	d9	6e
72	fa	09	ef

⊕ XOR

Round Key Value

ca	36	c2	3e
a1	03	ad	0f
e5	1e	ff	04
e3	d3	80	60

aes 14

4. ROUND 4

Start of Round

18	c4	c5	fc
18	bf	51	04
b7	4a	26	6a
91	29	89	5f

After Sub Bytes with S-Box

ad	1c	a6	b0
ad	08	d1	f2
a9	d6	f7	02
81	a5	a7	cf

After Shift Rows

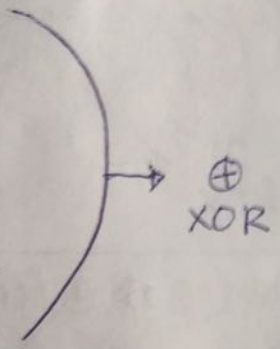
ad	1c	a6	b0
08	d1	f2	ad
f7	02	a9	d6
cf	81	a5	a7

After Mix Columns

61	d3	56	e6
70	22	1c	37
1a	51	e9	58
96	ee	fb	e5

Round Key Value

b4	82	40	7e
53	50	fd	f2
02	1c	e3	e7
51	82	02	b2



5. ROUND 5

Start of Round

d5	51	16	98
23	72	e1	c5
18	4d	0a	bf
c7	6c	f9	57

After Sub Bytes with S-Box

03	d1	47	46
26	40	f6	a6
ad	e3	67	c8
c6	50	99	5b

After Shift Rows

03	d1	47	46
40	f6	a6	26
67	c8	ad	e3
5b	c6	50	99

After Mix Columns

fa	64	82	9c
71	e4	ac	ad
60	68	50	0d
94	0f	62	26

Round Key Value

2d	af	ef	91
c7	97	6a	98
35	29	ca	2d
a2	20	22	90

⊕ XOR

6. ROUND 6

Start of Round

d7	cb	6d	0d
b6	73	c6	35
55	41	9a	20
36	2f	40	b6

After SubBytes with S-Box

0e	1f	3c	d7
4e	8f	b4	96
fc	83	b8	b7
05	15	09	4e

After Shift Rows

0e	1f	3c	d7
8f	b4	96	4e
b8	b7	fc	83
4e	05	15	09

After Mix Columns

60	4b	30	ed
96	ab	01	dc
38	d1	76	9f
b9	28	04	bd

Round Key Value

4b	e4	0b	9a
1f	88	e2	7a
55	7c	b6	9b
23	03	21	b1

⊕ XOR

7. ROUND 7

Start of Round

2b	af	3b	77
89	23	e3	a6
6d	ad	c0	04
9a	2b	25	0c

After Sub Bytes with S-Box

f1	79	e2	f5
a7	26	11	24
3c	95	ba	f2
b8	f1	3f	fe

After Shift Rows

f1	79	e2	f5
26	11	24	a7
ba	f2	3c	95
fe	b8	f1	3f

After Mix Columns

d7	8b	7e	a9
96	ee	1f	3b
a1	4a	b6	22
73	03	dc	48

Round Key Value

d1	35	3e	a4
0b	83	61	1b
9d	e1	57	cc
9b	98	69	08

⊕ XOR

8. ROUND 8

Start of Round

06	be	40	0d
9d	6d	7e	20
3c	a5	e1	ee
e8	9b	65	40

After Sub-Bytes with S-Box

6f	ae	09	d7
5e	3c	f3	b7
eb	06	f8	28
9b	14	4d	09

After Shift Rows

6f	ae	09	d7
3c	f3	b7	5e
f8	28	eb	06
09	9b	14	4d

After Mix Columns

6b	fa	2f	1c
0d	b0	4e	2c
a3	bb	4f	52
67	1f	6f	a0

⊕ XOR

Round Key Value

fe	cb	f5	51
40	c3	a2	b9
ad	4c	1b	d7
d2	4a	f3	fb

9. ROUND 9

Start of Round

95	31	da	4d
4d	73	ec	95
0e	f7	54	85
b5	55	9c	5b

After SubBytes with S-Box

2a	c7	57	e3
e3	8f	ce	2a
ab	68	20	97
d5	fc	de	39

After Shift Rows

2a	c7	57	e3
8f	ce	2a	e3
20	97	ab	68
39	d5	fc	de

After Mix Columns

c7	9e	87	55
76	37	19	58
ae	58	2f	a9
a3	ba	9b	12

Round Key Value

b8	78	8d	dc
4e	8d	2f	96
a2	ee	f5	22
03	49	ba	41

⊕ XOR

10. ROUND 10

Start of Round

74	e6	0a	89
38	ba	36	ce
0e	b6	da	8b
a0	f3	21	53

After SubBytes with S-Box

92	8e	67	a7
07	f4	05	8b
fe	4e	57	3d
e0	0d	fd	ed

After Shift Rows

92	8e	67	a7
f4	05	8b	07
57	3d	fe	4e
ed	e0	0d	fd

Round Key Value

15	6d	e0	3c
dd	50	7f	e9
21	cf	3a	18
85	cc	76	37

⊕ XOR

Output Ciphertext

87	e3	87	9b
29	55	f4	ee
76	f2	c4	56
68	2c	7b	ca

Inverse Cipher

#start of Round

Output / Ciphertext

87	e3	87	9b
29	55	f4	ee
76	f2	c4	56
68	2c	7b	ca

⊕

#round key value 10

15	6d	e0	3c
dd	50	7f	e9
21	cf	3a	18
85	cc	76	37

↓

92	8e	67	a7
f4	05	8b	07
57	3d	fe	4e
ed	e0	0d	fd

10. Round 10

92	8e	67	a7
f4	05	8b	07
57	3d	fe	4e
ed	e0	0d	fd

#start of round

92	8e	67	a7
07	f4	05	8b
fe	4e	57	3d
e0	0d	fd	ed

#after shift rows

74	e6	0a	89
38	ba	36	ce
0c	b6	da	8b
a0	f3	21	53

#after subbytes with inverse s-box

⊕

b3	78	8d	dc
4e	8d	2f	96
a2	ee	f5	22
03	49	ba	41

round key value 9

9. Round 9

Start of Round

c7	9e	87	55
76	37	19	58
ae	58	2f	a9
a3	ba	9b	12

After Inverse Mix Columns

2a	c7	57	e3
8f	ce	2a	e3
20	97	ab	68
39	d5	fc	de

After Shift Rows

2a	c7	57	e3
e3	8f	ce	2a
ab	68	20	97
d5	fc	de	39

After Sub Bytes with Inverse S-Box

95	31	da	4d
4d	73	ec	95
0e	57	54	85
b5	55	9c	5b

Round Key Value 8

fe	cb	f5	51
40	c3	a2	89
ad	4c	1b	d7
d2	4a	f3	fb

Inverse Mix Columns Round 9 - row 1

$= (0e \cdot c7) = ((x3+x2+x) \cdot (x7+x6+x2+x+1))$

$= x10 + x9 + x5 + x4 + x3 + x9 + x8 + x4 + x3 + x2 + x8 + x7 + x3 + x2 + x$

$= x10 + x7 + x5 + x3 + x = x2 \cdot x8 + x7 + x5 + x3 + x$

$= x2(x4+x3+x+1) + x7 + x5 + x3 + x = x6 + x5 + x3 + x2 + x7 + x5 + x3 + x$

$= x7 + x6 + x2 + x = 11000110 = c6$

$= (0b \cdot 76) = ((x3+x+1) \cdot (x6+x5+x4+x2+x))$

$= x9 + x8 + x7 + x5 + x4 + x7 + x6 + x5 + x3 + x2 + x6 + x5 + x4 + x2 + x$

$= x9 + x8 + x5 + x3 + x = x \cdot x8 + x8 + x5 + x3 + x$

$= x(x4+x3+x+1) + (x4+x3+x+1) + x5 + x3 + x$

$= x5 + x4 + x2 + x + x4 + x3 + x + 1 + x5 + x3 + x = x2 + x + 1$

$= 07$

$= (0d \cdot ae) = ((x3+x2+1) \cdot (x7+x5+x3+x2+x))$

$= x10 + x8 + x6 + x5 + x4 + x9 + x7 + x5 + x4 + x3 + x7 + x5 + x3 + x2 + x$

$= x10 + x9 + x8 + x6 + x5 + x2 + x = x2 \cdot x8 + x \cdot x8 + x8 + x6 + x5 + x2 + x$

$= x2(x4+x3+x+1) + x(x4+x3+x+1) + (x4+x3+x+1) + x6 + x5 + x2 + x$

$= x6 + x5 + x3 + x2 + x5 + x4 + x2 + x + x4 + x3 + x + 1 + x6 + x5 + x2 + x$

$= x5 + x2 + x + 1 = 00100111 = 27$

$= (09 \cdot a3) = ((x3+1) \cdot (x7+x5+x+1))$

$= x10 + x8 + x4 + x3 + x7 + x5 + x + 1$

$= x2 \cdot x8 + x8 + x4 + x3 + x7 + x5 + x + 1$

$= x2(x4+x3+x+1) + (x4+x3+x+1) + x4 + x3 + x7 + x5 + x + 1$

$= x6 + x5 + x3 + x2 + x4 + x3 + x + 1 + x4 + x3 + x7 + x5 + x + 1$

$= x7 + x6 + x3 + x2 = 11001100 = cc$

$c6 \oplus 07 \oplus 27 \oplus cc =$ **2a**

A E S
(Advanced Encryption Standard)

AES has 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys. High-level description of the algorithm:

1. KeyExpansion – round keys are derived from the cipher key using the AES key schedule. AES requires a separate 128-bit round key block for each round plus one more.
2. Initial round key addition:
   - *AddRoundKey – each byte of the state is combined with a byte of the round key using bitwise xor.*

3. 9, 11 or 13 rounds:
   1. *SubBytes – a non-linear substitution step where each byte is replaced with another according to a lookup table.*
   2. *ShiftRows – a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.*
   3. *MixColumns – a linear mixing operation which operates on the columns of the state, combining the four bytes in each column.*
   4. *AddRoundKey*

4. Final round (making 10, 12 or 14 rounds in total):
   1. *SubBytes*
   2. *ShiftRows*
   3. *AddRoundKey*

In June 2003, the U.S. Government announced that AES could be used to protect classified information: ***The design and strength of all key lengths of the AES algorithm (i.e., 128, 192 and 256) are sufficient to protect classified information up to the SECRET level. TOP SECRET information will require use of either the 192 or 256 key lengths.***

By 2006, the best known attacks were on 7 rounds for 128-bit keys, 8 rounds for 192-bit keys, and 9 rounds for 256-bit keys (https://en.wikipedia.org/wiki/Advanced_Encryption_Standard).

Cipher

Values of rc_i in hexadecimal

i	1	2	3	4	5	6	7	8	9	A
rc_i	01	02	04	08	10	20	40	80	1B	36

Round Constant for Key Expansion

Bit Pattern	Character
0000	0
0001	1
0010	2
0011	3

Bit Pattern	Character
0100	4
0101	5
0110	6
0111	7

Bit Pattern	Character
1000	8
1001	9
1010	a
1011	b

Bit Pattern	Character
1100	c
1101	d
1110	e
1111	f

Hexadecimal representation of bit patterns

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

AES S-box substitution values for the byte xy (in hexadecimal format). The column is determined by the least significant nibble, and the row by the most significant nibble. For example, the value 9a16 is converted into b816.

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}$$

Polynomial representation in Mix Columns:

- 01 = 00000001 = 1
- 02 = 00000010 = x
- 03 = 00000011 = x + 1
- when resulted to x8 = 10000000 then x8 + x4 + x3 + x + 1 applied

Inverse Cipher

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Inverse S-box: substitution values for the byte xy (in hexadecimal format). The inverse affine transformation also represents the sum of multiple rotations of the byte as a Vector, where addition is the XOR operation

$$\begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix}$$

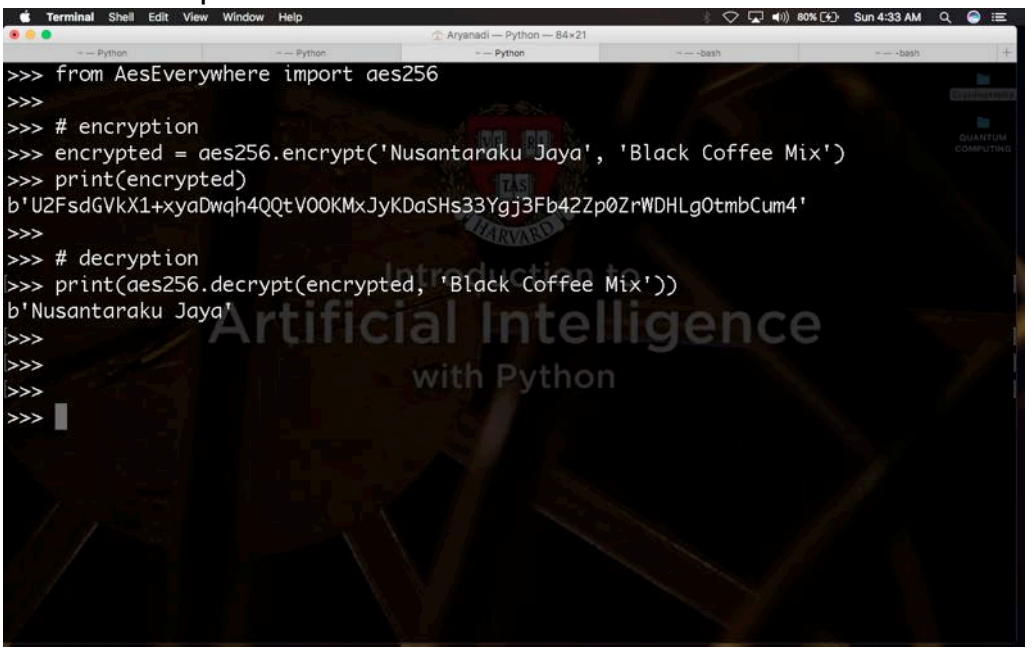
Inverse Mix Columns

Polynomial representation in Inverse Mix Columns:

- 0e = 00001110 = x3 + x2 + x
- 0d = 00001101 = x3 + x2 + 1
- 0b = 00001011 = x3 + x + 1
- 09 = 00001001 = x3 + 1
- x10 = x2 . x8 = x2(x4 + x3 + x + 1)
- x 9 = x . x8 = x(x4 + x3 + x + 1)
- x 8 = x4 + x3 + x + 1

Quick Example:

AES 256 bits implementation



MATHEMATICAL FUNDAMENTAL

Mathematical Fundamental

Modern cryptography is all about understanding integer mathematics.

- Integers $Z = \{..., -3, -2, -1, 0, 1, 2, 3, ...\}$
- Positive integers $Z^+ = \{1, 2, 3, ...\}$
- Negative integers $Z^- = \{..., -3, -2, -1\}$
- Non-negative integers $Z^{0+} = \{0,1,2,3,...\}$
- Non-positive integers $Z^{0-} = \{..., -3, -2, -1, 0\}$

- Discrete mathematics and abstract algebra
- Calculations must be exact (no round off error)
- Several “hard” problems to provide crypto keys
- Integers and their subsets
- Divisibility
- Prime and composite numbers
- Fundamental theorem of arithmetic
- Greatest common divisor and relatively prime

MODULAR ARITHMETIC is what we actually all the time. For instance, if it is currently 9:40 AM and I have to leave in 25 minutes, I don't say that I need to leave at 9:65 AM. Instead, I add 25 to 40, get 65, and for the minutes I keep the remainder of this number after dividing by 60. The reason that I can reduce 65 by 60 is because the world of minutes only has 60 elements. We call this a modulus 60 system (mod60).

So Modular Arithmetic is an arithmetic system in which all relations are reduced by a common modulus. Basic concept of Modular Arithmetic:

$$n = q \cdot d + r$$

n is dividend
q is quotient
d is divisor (“modulus”)
r is reminder (“residue”)

Quick Example

$n = q \cdot d + r$

$37 = 13 \cdot 2 + 11$

$37 = 13 \cdot 3 + (-2)$

$37 = 13 \cdot 4 + (-15)$

……infinite possibilities…..by convention: choose q such that $0 \leq r < d$

$80 = 12 \cdot 6 + 8$

$80 = 12 \cdot 7 + (-4)$

$80 = 12 \cdot 8 + (-16)$

……infinite possibilities…..by convention: choose q such that $0 \leq r < d$

The modular arithmetic operator is expressed as **mod** or **%**, where the **modulus** is denoted as **m**, and the **remainder** or **residue** is denoted as **r**.

integer **r = n mod m** where **n >= m**

if **n** < **m** then **r = n** , 3 mod 8 = 3

Commutative, associative and distributive function in **Modular Arithmetic Properties** expressed as follow:

1. **(a · (b + c)) mod n = ((a · b) mod n)) + ((a · c) mod n)) mod n**
   (37 · (99 + 2)) mod 6 = ((37 · 99) mod 6)) + ((37 · 2) mod 6)) mod 6
   (37 · (99 + 2)) mod 6 = 3665 mod 6 = 5
   (3663 mod 6) + (74 mod 6) mod 6 = (3 + 2) mod 6 = 5
   so 3665 mod 6 = (3 + 2) mod 6 = 5
2. **(a + b) mod n = ((a mod n) + (b mod n)) mod n**
   (37 + 99) mod 6 = ((37 mod 6) + (99 mod 6)) mod 6 = 4
3. **(a – b) mod n = ((a mod n) – (b mod n)) mod n**
   ((37 mod 6) - (99 mod 6)) mod 6 = -2 mod 6 = 4
   (37 – 99) mod 6 = -62 mod 6 = 4
   so (37 – 99) mod 6 = ((37 mod 6) - (99 mod 6)) mod 6 = 4
4. **(a · b) mod n = ((a mod n) · (b mod n)) mod n**
   ((37 mod 6) + (99 mod 6)) mod 6 = 3 mod 6 = 3
   (37 · 99) mod 6 = 3663 mod 6 = 3
   so (37 · 99) mod 6 = ((37 mod 6) · (99 mod 6)) mod 6 = 3

GREATEST COMMON DIVISOR (GCD) is largest **integer** that divides both *a* and *b*, and it is the largest factor that two positive integers have in common.

The integer *c* is a common divisor of *a* and *b* if:

- $a = c \cdot m$ and $b = c \cdot n$ for some *m* and *n* ; OR if $c \mid a$ and $c \mid b$
- gcd (*a, b*) is the largest common divisor of *a* and *b*
  - gcd (*a,b*) ≥ 1
  - By convention, *a* and *b* can't both be zero
- gcd (*a, b*) = 1 → *a* and *b* are "relatively prime"

Quick Example

Divisor of 40 = 1, 2, 4, **5**, 8, 10, 20, 40
Divisor of 15 = 1, 3, **5**, 15
GCD of 40 and 15 = **5**

Divisor of 86 = **1**, 2, 43, 86
Divisor of 23 = **1**, 23
GCD of 86 and 23 = **1**

EUCLIDEAN ALGORITHM is the oldest known algorithm but an efficient one used to find the GCD between two positive integers of *a* and *b*, where *a* ≥ *b.*

gcd (*a, b*) = gcd (*b, a* mod *b*) ; OR
gcd (*a, b*) = gcd (*b, a* % *b*)

Quick Example 1 - to find the GCD of *a* = 40, and *b* = 15 using Euclidean Algorithm:

gcd (40, 15) = gcd (15, 40 mod 15); remainder of 40 : 15 is 10
gcd (15, 10) = gcd (10, 15 mod 10); remainder of 15 : 10 is 5
gcd (10, 5) = gcd (5 , 10 mod 5); remainder of 10 : 5 is 0
gcd (5, 0) = **5**

a	***b***	***q***	***r***
40	15	2	10
15	10	1	5
10	**5**	2	0

Quick Example 2 - to find the GCD of *a* = 86, and *b* = 23 using Euclidean Algorithm:

gcd (86, 23) = gcd (23, 86 mod 23); remainder of 86 : 23 is 17
gcd (23, 17) = gcd (17, 23 mod 17); remainder of 23 : 17 is 6
gcd (17, 6) = gcd (6, 17 mod 6); remainder of 17 : 6 is 5
gcd (6, 5) = gcd (5, 6 mod 5); remainder of 6 : 5 is 1
gcd (5, 1) = gcd (1, 5 mod 1); remainder of 5 : 1 is 5
gcd (1, 5) = gcd (5, 1 mod 5); 1 mod 5 is 1

a	***b***	***q***	***r***
86	23	3	17
23	17	1	6
17	6	2	5
6	5	1	1
5	**1**	5	0

Quick Example 3 - to find the GCD of *a* = 2017, and *b* = 1024 using Euclidean Algorithm:

gcd (2017, 1024) = gcd (1024, 2017 mod 1024);
remainder of 2017 : 1024 is 993
gcd (1024, 993) = gcd (993, 1024 mod 993);
remainder of 1024 : 993 is 31
gcd (993, 31) = gcd (31, 993 mod 31); remainder of 993 : 31 is 1
gcd (31,1) = gcd (1, 31 mod 1); remainder of 31 : 1 is 0
gcd (1,0) = **1**

a	***b***	***q***	***r***
2017	1024	1	993
1024	993	1	31
993	31	32	1
31	**1**	31	**0**

EXTENDED EUCLIDEAN ALGORITHM is an efficient algorithm for finding the modular multiplicative inverse of one number and a modulus, if it exists.

Quick Example 1

a = 2017

b = 1024

assumption for s1 = 1, s2 = 0 and t1 = 0, t2 = 1

a	***b***	***q***	***r***	***s1***	***s2***	***s3***	***t1***	***t2***	***t3***
2017	1024	1	993	**1**	**0**	1	**0**	**1**	-1
1024	993	1	31	0	1	-1	1	-1	2
993	31	32	1	1	-1	33	-1	2	-65
31	**1**	31	0	-1	33	-1024	2	-65	2017

s3 = s1 – q · s2

= 0 – 1 · 1 = -1

1 – 32 · (-1) = 33

(-1) – 31 · 33 = -1024

t3 = t1 – q . t2

= 1 – 1 · (-1) = 2

(-2) – 32 · (3) = 98

3 – 31 · (-98) = -3041

When r = 0, then gcd = $|b|$

To prove gcd (2017, 1024) = $s2 \cdot a + t2 \cdot b$

= 33 · 2017 + (-65) · 1024 = **1**

alternative table layout

q	***r1***	***r2***	***r***	***s1***	***s2***	***s3***	***t1***	***t2***	***t3***
1	2017	1024	993	**1**	**0**	1	**0**	**1**	-1
1	1024	993	31	0	1	-1	1	-1	2
32	993	31	1	1	-1	33	-1	2	-65
31	31	**1**	**0**	-1	33	-1024	2	-65	2017

Quick Example 2

$a = 40$

$b = 15$

assumption for s1 = 1, s2 = 0 and t1 = 0, t2 = 1

a	***b***	***q***	***r***	***s1***	***s2***	***s3***	***t1***	***t2***	***t3***
40	15	2	10	**1**	**0**	1	**0**	**1**	-2
15	10	1	5	0	1	-1	1	-2	3
10	**5**	2	**0**	1	-1	3	-2	3	-8

s3 = s1 – q · s2

$= 1 - 2 \cdot 0$ $\quad 0 - 1 \cdot 1$ $\quad 1 - 2 \cdot (-1)$

$= 1$ $\quad -1$ $\quad 3$

t3 = t1 – q . t2

$= 0 - 2 \cdot 1$ $\quad 1 - 1 \cdot (-2)$ $\quad (-2) - 2 \cdot 3$

$= -2$ $\quad 3$ $\quad -8$

When $r = 0$, then gcd = $|b|$

To prove gcd (40, 15) = $s2 \cdot a + t2 \cdot b$

$= (-1) \cdot 40 + 3 \cdot 15 =$ **5**

alternative table layout

q	***r1***	***r2***	***r***	***s1***	***s2***	***s3***	***t1***	***t2***	***t3***
2	40	15	10	**1**	**0**	1	**0**	**1**	-2
1	15	10	5	0	1	-1	1	-2	3
2	10	**5**	**0**	-1	3	3	-2	3	-8

MULTIPLICATIVE INVERSE

mod 6	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

$3 \cdot 1 \bmod 6 = 3$
$3 \cdot 2 \bmod 6 = 0$
$3 \cdot 3 \bmod 6 = 3$
$3 \cdot 4 \bmod 6 = 0$
$3 \cdot 5 \bmod 6 = 3$

$4 \cdot 2 \bmod 6 = 2$
$4 \cdot 3 \bmod 6 = 0$
$4 \cdot 4 \bmod 6 = 4$
$4 \cdot 5 \bmod 6 = 2$

$5 \cdot 2 \bmod 6 = 4$
$5 \cdot 3 \bmod 6 = 3$
$5 \cdot 4 \bmod 6 = 2$
$5 \cdot 5 \bmod 6 = 1$

In mod-6 multiplication table above shows us every nonzero residue class has an inverse. Please note that zero does not have a multiplicative inverse.

The multiplicative invers of a mod N, if it exists, is the value b, written as a^{-1}, (or any member of b's residue class) such that $a \cdot b \equiv 1 \pmod N$.

Quick Example 1

q	***r1***	***r2***	***r***	***t1***	***t2***	***t3***
3	86	23	17	**0**	**1**	-3
1	23	17	6	1	-3	4
2	17	6	5	-3	4	-11
1	6	5	1	4	-11	15
5	5	**1**	**0**	**-11**	15	-86

t3 = t1 – t2 · q

Multiplicative Inverse for 23 in Z 86 is ***t1*** = **-11** ---> negative value.

If ***t1*** results to a negative value, then calculate ***r1 + t1*** to get the positive value from multiplicative inverse**.** So the positive value of multiplicative inverse for 23 in Z 86 is 86 + (-11) = **75**.

15 mod 86 = 15

To prove:
23 · 15 mod 86 ≡ **1**
345 mod 86 ≡ **1**
86 · 4 = 344
*345 – 344 = **1***

Quick Example 2

q	***r1***	***r2***	***r***	***t1***	***t2***	***t3***
45	780	17	15	**0**	**1**	-45
1	17	15	2	1	-45	46
7	15	2	1	-45	46	-367
2	2	**1**	**0**	**46**	-367	780

t3 = t1 – t2 · q

Multiplicative Inverse for 17 in Z 780 is ***t1*** = **46** ---> positive value

-367 mod 780 = 413 ------ > simply 780 – 367 = 413

To prove:
17 · 413 mod 780 ≡ **1**
7021 mod 780 ≡ **1**
780 · 9 = 7020
*7021 – 7020 = **1***

Quick Example 3

q	*r1*	*r2*	*r*	*t1*	*t2*	*t3*
1	2017	1024	993	**0**	**1**	-1
1	1024	993	31	1	-1	2
32	993	31	1	-1	2	-65
31	31	**1**	**0**	2	-65	2017

t3 = t1 – t2 . q

Multiplicative Inverse for 1024 in Z 2017 is ***t1*** = **2** ---> positive value

-65 mod 2017 = 1952 ------ > simply 2017 – 65 = 1952

To prove:

1024 · 1952 mod 2017 ≡ **1**

1998848 mod 2017 ≡ **1**

2017 · 991 = 1998847

*1998848 – 1998847 = **1***

Quick Example 4

q	*r1*	*r2*	*r*	*t1*	*t2*	*t3*
8	60	7	4	**0**	**1**	-8
1	7	4	3	1	-8	9
1	4	3	1	-8	9	-17
3	3	**1**	**0**	9	-17	60

t3 = t1 – t2 · q

Multiplicative Inverse for 7 in Z 60 is ***t1*** = **-17** ---> negative value

So the positive value of multiplicative inverse for 7 in Z 60 is ***r1 + t1*** = 60 + (-17) = **43**.

-17 mod 60 = 43 ------ > simply 60 – 17 = 43

To prove: 7 · 43 mod 60 ≡ **1**

301 mod 60 ≡ **1**

60 · 5 = 300

*301 – 300 = **1***

MODULAR CONGRUENCE is two integers, *a* and *b*, are congruent modulo *N* if their difference is divisible by *N* ---- > ***a* ≡ *b* (mod *N*) iff *n*|(*a* – *b*)**

Quick Example 1

a = 48, *b* = 30, *N* = 9 48 ≡ 30 (mod 9) *since 9 divides 48 – 30 = 18;* *18 : 9 = 2* *remainder 0*	*a* = 53, *b* = 38, *N* = 5 53 ≡ 38 (mod 5) *since 5 divides 53 – 38 = 15;* *15 : 5 = 3* *remainder 0*
a = 20, *b* = 14, *N* = 6 20 ≡ 14 (mod 6) *since 6 divides 20 – 14 = 6;* *6 : 6 = 1* *remainder 0*	**Division is NOT defined in modular arithmetic.** **Don't get tempted to do it!** **15/3 ≡ 3/3** **WRONG! NOT TRUE!**
a = 15, *b* = 3, *N* = 6 15 ≡ 3 (mod 6) *since 6 divides 15 – 3 = 12;* *12 : 6 = 2* *remainder 0*	*a* = 30, *b* = 12, *N* = 9 30 ≡ 12 (mod 9) *since 9 divides 30 – 12 = 18;* *18 : 9 = 2* *remainder 0*
a = 555552, *b* = 7882, *N* = 10 555552 ≡ 7882 (mod 10) *since 10 divides 555552 – 7882 = 547670 : 10 = 54767* *remainder 0*	*a* = -29, *b* = 5, *N* =17 -29 ≡ 5 (mod 17) *since 17 divides (-29) - 5 = -34;* *(-34) : 17 = 2* *remainder 0*

Quick Example 2

Suppose that a and b are integers, $a \equiv 4 \pmod{13}$, and $b \equiv 9 \pmod{13}$, so the integers c with $0 \leq c \leq 12$ such that

$c \equiv a \cdot b \pmod{13}$

$c \equiv 4 \cdot 9 \pmod{13}$

$c \equiv 36 \pmod{13}$

$c = 10$

$10 \equiv 36 \pmod{13}$; *since* $36 = 13 \cdot 2 + 10$;

Also 13 divides 10 – 36 = - 26 ; (-26) : 13 = -2 , remainder 0

$c \equiv 5 \pmod{17}$ $c = -29$ $-29 \equiv 5 \pmod{17}$; *since* $17 \cdot (-2) + 5 = -29$; *Also 17 divides (−29) – 5* *(-34) : 17 = -2 , remainder 0*	$c \equiv 5 \pmod{17}$ $c = 56$ $56 \equiv 5 \pmod{17}$; *since* $17 \cdot 3 + 5 = 56$; *Also 17 divides 56 – 5* *51 : 17 = 3 , remainder 0*

EULER'S TOTIENT

Properties:

1. $\phi(p) = p - 1$; for a p is a prime number
2. $\phi(m \cdot n) = \phi(m) \cdot \phi(n)$; if m is not equal to n (if $m \neq n$)
3. $(p^e) = p^e - p^{e-1}$; for a p is not a prime number
4. $(1) = 0$

Quick Example 1 for $\phi(p) = p - 1$

$\phi(17) =$???

17 is a prime number so we can apply $\phi(p) = p - 1$ whenever we meet a prime number.

$\phi(17) = 17 - 1 = 16$

$\phi(23) =$???

23 is a prime number so we can apply $\phi(p) = p - 1$ whenever we meet a prime number.

$\phi(23) = 23 - 1 = 22$

Quick Example 2 for $\phi(m \cdot n) = \phi(m) \cdot \phi(n)$

$\phi(314191) =$???

314191 is not a prime number so we can split it into prime numbers of 379 and 829 then apply $\phi(m \cdot n) = \phi(m) \cdot \phi(n)$

$\phi(379 \times 829) = \phi(379) \times \phi(829)$

$= (379 - 1) \times (829 - 1)$

$= 378 \times 828 = 312984$

$\phi(10) =$???

10 is not a prime number so we can split it into prime numbers of 5 and 2 then apply $\phi(m \cdot n) = \phi(m) \cdot \phi(n)$

$\phi(5 \times 2) = \phi(5) \times \phi(2)$

$= (5 - 1) \times (2 - 1)$

$= 4 \times 1 = 4$

$\phi(15) =$???

10 is not a prime number so we can split it into prime numbers of 5 and 2 then apply $\phi(m \cdot n) = \phi(m) \cdot \phi(n)$

$\phi(5 \times 3) = \phi(5) \times \phi(3)$

$= (5 - 1) \times (3 - 1)$

$= 4 \times 2 = 8$

Quick Example 3 for $(p^e) = p^e - p^{e-1}$

$\phi(81) =$???

81 is not a prime number so we can apply $(p^e) = p^e - p^{e-1}$

$\phi(3^4) = 3^4 - 3^4 - 1$

$= 3^4 - 3^4 - 1$

$= 81 - 27 = 54$

$\phi(64) =$???

$(2^6) = 2^6 - 2^6 - 1$

$= 64 - 32 = 32$

ϕ(49) = ???

$(7^2) = 7^2 - 7^{2-1}$

$= 49 - 7 = 42$

Quick Example 4 for (1) = 0

$\phi(1^{10})$ = ???

$\phi(1^{10}) = 1^{10} - 1^{10-1}$

$= 1^{10} - 1^9$

$= 1 - 1 = 0$

Quick Example 5

ϕ(8) = ???

8 is not a prime number and we cannot split it into prime numbers so we must break it down to yield it into prime numbers.

2	8
2	4
2	2
	1

$\phi(8) = \phi(2^3)$

we have a prime number to the power of 3 so we must apply

$(p^e) = p^e - p^{e-1}$

$= 2^3 - 2^{3-1}$

$= 8 - (2^2)$

$= 8 - 4 = 4$

ϕ(9) = ???

9 is not a prime number and we cannot split it into prime numbers so we must break it down to yield it into prime numbers.

3	9
3	3
	1

$\phi(9) = \phi(3^2)$

we have a prime number to the power of 2 so we must apply

$(p^e) = p^e - p^{e-1}$

$= 3^2 - 3^{2-1}$

$= 9 - (3)$

$= 9 - 3 = 6$

ϕ(180) = ???

180 is not a prime number and we cannot split it into prime numbers so we must break it down to yield it into prime numbers.

2	180
2	90
5	45
3	9
3	3
	1

$\phi(180) = \phi(2^2 \times 5^1 \times 3^2)$

$= \phi(2^2) \times \phi(5) \times \phi(9)$

we have a prime number to the power of 2 so we must apply

$(p^e) = p^e - p^{e-1}$

$= (2^2 - 2^{2-1}) \times (5 - 1) \times (9)$

9 is not a prime number and we have computed it and got the result which is 6.

$= (4 - 2) \times (4) \times (6)$

$= (2) \times (4) \times (6) = 48$

Euler's Totient Theorem: (when gcd (x, N) = 1) $x^{\varphi(N)} \equiv 1 \pmod{N}$
x and N must be relatively prime

Euler's Totient Theorem only applies to bases relatively prime to modulus. Euler's totient theorem claims that X raised to the totient of N is congruent to one moduli N for any X that is relatively prime to N and where, as we've already stated, the totient of N is the size of the set, S, consisting of all positive integers that are both less than and relatively prime to N. Don't gloss over the requirement that the base and the modulus must be relatively prime. This is an easy and commonly forgotten caveat. You can't just blindly reduce the exponent by the totient of the modulus and be done with it.

A proof in 8 simple steps:

1. $S = \{k \mid 0 < k < N \text{ and } \gcd(k, N) = 1\}$
2. $m = |S|$
3. Choose $x \mid \gcd(x, N) = 1$
4. $R = \{j \mid j = x \cdot k \bmod (N) \text{ for each } k \in S\}$
5. Establish four key properties of sets S and R
6. Show that S = R
7. Define P_S and P_R, the products over each set
8. Use $P_S = P_R$ to establish the theorem's claim

• Generic algebraic proof and numeric example in parallel

Step 1: *S = {k | 0 < k < N and gcd (k, N) = 1}*

$5^{\text{tot}(12)} \bmod 21 = 1$

- N = 21
- S = {1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20}
- |S| = 12

Choose N = 21 is our modulus. What is the totient of 21? We can just list the positive integers that are less than 21, keeping only those that are not divisible by either 3 or 7 because 21 is the result of 7 · 3. We find that there are 12 of them. So, the totient of 21 is 12.

Let's use 5 because X that is relatively prime to 21. The theorem

therefore claims that 5 to the twelfth power is congruent to 1 in a mod 21 world. Is it? Let's practice our square and multiply technique:

- x = 5
  - $5^1 \equiv 5 \pmod{21}$
  - $5^2 \equiv 25 \equiv 4 \pmod{21}$
  - $5^4 \equiv 16 \pmod{21}$
  - $5^8 \equiv 256 \equiv 4 \pmod{21}$
- $5^{12} \equiv 5^8 \cdot 5^4 \equiv 4 \cdot 16 \equiv 63 \equiv 1 \pmod{21}$

Our exponent of 12 has a binary representation of 1,1, 0,0. So, we need to multiply 5 to the 8^{th} and 5 to the 4^{th}. We can square and reduce our base of 5 until we get to 5 to the 8^{th}, then we just use the ones that we need in. As expected, the result is congruent to 1.

But what about that caveat that the base has to be relatively prime to the modulus? This is not just some form of fine print in the proof. The claim simply does not hold if this constraint is not respected. Let's repeat our previous example, except use X = 6, which is not relatively prime to 21.

$$6^{tot(12)} \bmod 21 \neq 1$$

- N = 21
- S = {1, 2, 4, 5, 8, 10, 11 ,13, 16, 17, 19, 20}
- |S|=12
- x = 6
  - $6^1 \equiv 6 \pmod{21}$
  - $6^2 \equiv 36 \equiv 15 \pmod{21}$
  - $6^4 \equiv 225 \equiv 15 \pmod{21}$
  - $6^8 \equiv 225 \equiv 15 \pmod{21}$
- $6^{12} \equiv 6^8 \cdot 6^4 \equiv 15 \cdot 15 \equiv 225 \equiv 15 \pmod{21}$

Thus, 6 to the 12 is not congruent to 1 in a mod 21 world. This is a critically important restriction and ignoring it can and has resulted in crypto packages that just don't work.

Step 2: *m = |S|*

- N = 21
- S = {1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20}
- m = |S| = 12

• By definition, m = ϕ ***N*** = totient (N) = tot (N)

In this step, let's use N = 21 which results in this set for S. We merely assign the number of elements in set S, known as its cardinality, to the parameter M. For our example, M=12.

Step 3: *Choose x | gcd (x, N) = 1*

- N = 21
- S = {1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20}
- m = |S| = 12

Now pick any integer X with the only restriction that it be relatively prime to N. Notice that this restriction, while allowing X to be negative, precludes it from being 0 since N divides 0 according to our definition of divisibility. And thus, 0 is not relatively prime to any other integer. Also, note that since we are working in a Mod N world, we can reduce our chosen integer Mod N without loss of generality which coincidentally will leave us with a member of the set S. For our example, let's stick with what we know and choose X=5.

Step 4: R = *{j | j = x · k mod (N) for each k ∈ S}*

- N = 21
  - S = {1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20}
  - m = |S| = 12
- x = 5
  - R = {5, 10, 20, 4, 19, 8, 13, 2, 17, 1, 11, 16}

In this step, we create a new set that we arbitrarily call R, that consists of our chosen value for x multiplied in turn by each member of the set S, and reduced modulo N. Taking our example set, we get this for R after the reduction, keeping them in the order they were generated just for clarity.

Step 5: *Establish four key properties of sets S and R*

1. All elements of S are distinct, **meaning that there are no repeats**
   - Self-evident by nature of construction. This is self-evident because of the way S was constructed, since only distinct positive integers less than the modulus and relatively prime to it were included.

2. $|S| = |R| = m$, **both S and R contain the same number of elements**
   - Self-evident by nature of construction. This is also self-evident since each element in S was used to create exactly one element in R. Now it's possible that, after the modular reduction, that we have some repeats. For now, we just care that both sets have the same number of elements, namely m.

3. $R \subseteq S$ **every element in R is also an element in S.**
   All element in R are non-negative integers less than N.
   Any element relatively prime to N must also be in S.
   - Assume $j \in R \mid \gcd(j, N) = d$ where $d > 1$
   - $j = k \cdot x \bmod N$ where $k \in S$
   - $j = k \cdot x + q \cdot N \ (0 \leq j < N)$
   - $c_1 \cdot d = k \cdot x + q \cdot c_2 \cdot d$
   - $c_1 = k \cdot x / d + q \cdot c_2$
   - $\gcd(j, N) = 1$ (contradiction!)
   - If $j \in R$ then $j \in S$, thus $R \subseteq S$

We constructed R by performing some math and reducing the result modulo N. This means that all elements in R are non-negative integers strictly less than N. And since all such integers that are relatively prime

to N are contained within S, that means that if all elements of R are relatively prime to N, then they must also be in S. So we will assume that there is some element R that is not relatively prime to N. Meaning that there is some element, we'll call it j, that has a divisor in common with N that is greater than one. We'll call the greatest such divisor, D.

The element j was generated using the relation, $j = k \cdot x \bmod N$, using one of the elements K from set S. By our defining relation for modular arithmetic, we know that $j = k \cdot x + q \cdot N$, where j is greater than or equal to zero, but less than N. If j is not relatively prime to N, that means that it can be written as the product of d and some other integer. We'll call it c1. Likewise, our modulus can be written as the product of d and some other integer, c2. Dividing both sides by c, since we're not working in a modular world, division is allowed, we have the result that an integer is equal to the sum of two terms, the second of which is clearly an integer. This means that the first term, which involves division, must also be an integer. However, k and x were specifically chosen such they were relatively prime to N, meaning that they must also be relatively prime to any factors of N, including d. The result is that k x divided by d cannot be an integer, unless the d is exactly equal to one. This is our contradiction, because this requires that j and N be relatively prime, which we assumed was not the case. Hence, j must be a positive integer, less than and relatively prime to N. Meaning that it is a member of the set S, by definition. And thus, R is a subset of S.

- Proof by contradiction. If we can establish this, then that establishes that R is a subset of S. We prove this by contradiction by assuming that some element of R exists that is not also an element of S.

4. All elements of S are distinct
    - Proof by contradiction. no repeats even after the modular reductions are performed. This proof is also done by contradiction by assuming that there do exist two elements in R that are not distinct.

- Assume that $j_1, j_2 \in R$ exist such that $j_1 = j_2$
- $j_1 \equiv k_1 \cdot x \pmod N$ and $j_2 \equiv k_2 \cdot x \pmod N$
- $j_1 = j_2 \Rightarrow j_1 \equiv j_2 \pmod N$
- $k_1 \cdot x \equiv k_2 \cdot x \pmod N$
- $x^{-1} \pmod N$ exists
- $k_1 \equiv k_2 \pmod N$
- All $k \mid 0 \leq k < N$ are in distinct residue classes
- $k_1 = k_2$ (contradiction!)
- $j_1 \not\equiv j_2 \pmod N \Rightarrow$ All $j \in R$ are distinct

Let's assume that two different elements of R are the same. Since all elements in R were generated from distinct elements in S, if R contains two identical elements, they had to be generated using two different elements of S. Let's assume that elements were k1 and k2. The corresponding elements in R, are j1 and j2, and are related to k1 and k2 as shown. Clearly, if j1 and j2 are equal to each other, then they are also congruent mod N, which requires that $k1 \cdot x$ and, $k2 \cdot x$, also be congruent mod N. Since x and N are relatively prime, x has a multiplicative inverse mod N. This allows us to cancel x from both sides, yielding this result. Since k1 and k2 are both positive integers less than N, and all such integers are in distinct residue classes, the only way for k1 and k2 to be congruent is for them to be equal, which contradicts our assumption that they are distinct. Therefore, j1 and j2 cannot be congruent, and must also be distinct elements of R.

Step 6: *Show that S = R*

1. All elements of S are distinct **meaning that there are no repeats**
2. $|S|=|R| = m$ **both S and R contain the same number of elements**
3. $R \subseteq S$ **every element in R is also an element in S.**
   All element in R are non-negative integers less than N.
   Any element relatively prime to N must also be in S.
4. All elements of S are distinct

- S $\subseteq$ R by pigeonhole principle
- R $\subseteq$ S and S $\subseteq$ R $\Rightarrow$ R = S

If the sets S an R each contain the same number of distinct elements, and R is a subset of S, then the pigeonhole principle requires that every element in S is equivalent to one of the elements in R, making S a subset of R. If two sets are each subsets of each other, then the two sets must be equivalent, i.e., they contain the same elements varying only in the ordering of them. The two sets in our example clearly exhibit this property as verified by inspection.

Step 7: *Define P_S and P_R, the products over each set*

We simply form the product of all of the elements within each set. P_sub_S is the product of all the K values in S, while P_sub_R is the product of all the j values in R.

$$P_S = \prod_{i=1}^{m} k_i$$

$$P_R = \prod_{i=1}^{m} j_i$$

- **Example: $P_S = P_R = 47{,}297{,}536{,}000$**

For our two example sets, those products are a large, and largely meaningless number. The fact that the two products must be equal is, however, self-evident.

Step 8*: Use $P_S = P_R$ to establish the theorem's claim*

Since the two sets are equivalent, the products of all their elements must be equal regardless of ordering since multiplication is commutative. Writing the elements of R, in terms of the elements of S that produce them, we have this relation:

$$P_S = P_R$$

$$\prod_{i=1}^{m} k_i = \prod_{i=1}^{m} j_i \equiv \prod_{i=1}^{m} k_i \, x \, (mod \, N)$$

$$\prod_{i=1}^{m} k_i \equiv x^m \prod_{i=1}^{m} k_i \; (mod \, N)$$

$$1 \equiv x^m \equiv x^{\varphi(N)} (mod \, N)$$

We can pull out the m factors of x leaving us with this. Since each k is relatively prime to N, it has a multiplicative inverse. Multiplying both sides by each of these inverses, we get our final result that one is congruent to x, raised to the m mod N. Recall that M is, by definition, the totient of N.

The exponent lives in a mod-totient world – but only when the base is r.p. to the modulus. Since the two sets are equivalent, the products of all their elements must be equal regardless of ordering since multiplication is commutative. Writing the elements of R, in terms of the elements of S that produce them, we have this relation:

$$x^y = x^{qm + r} = x^{qm + y} \bmod m$$

$$x^y = x^{qm} x^y \bmod m$$

$$x^y = x^{m\,q} x^y \bmod m$$

$$x^y \equiv x^{m\,q} x^y \bmod \varphi(N) \pmod N$$

$$x^y \equiv 1^q \, x^y \bmod \varphi(N) \pmod N \text{ IF } \gcd(x, N) = 1$$

$$x^y \equiv x^y \bmod \varphi(N) \pmod N \text{ IF } \gcd(x, N) = 1$$

We can pull out the m factors of x leaving us with this. Since each k is relatively prime to N, it has a multiplicative inverse. Multiplying both sides by each of these inverses, we get our final result that one is congruent to x, raised to the m mod N. Recall that M is, by definition, the totient of N.

In general $x^y \not\equiv x^{y \bmod \varphi N} \bmod N$.

Congruence only holds when x, N are relatively prime!

All that is left is to formally show that the exponent in a modular expression lives in a modulo world given by the totient of the modulus. This is quite straightforward. Choose an arbitrary exponent, y. By our defining relationship for modular arithmetic, this can be written in terms of our totient m, as some quotient Q times m plus R, where R is, y mod N. By the additive property of exponents, we can write this as the product of two factors. And by the multiplicative property of exponents, we can write the first factor as an integer power of x to the m. When we take this into a mod N world, we can replace m with the totient of the modulus. If x is relatively prime to N, then x to the m is congruent to one. And since any integer powered one is one, we are left with our final result. Remember that, in general, this relationship is not true. It is not true that the exponent in modular expressions can always be reduced modulo to the totient of the expression modulus, despite our recent result. That result is only valid when x is also relatively prime to N. This point simply cannot be stressed enough, primarily because it is such a tempting trap to fall into.

Euler's Totient Function is the number of positive integers less than and relatively prime to N, written in expression: $\varphi(N) = S$ where S = {k | 0 < k < N and gcd(N, k) = 1}. Fundamental theorem of arithmetic:

$$N = \prod_{i=1}^{P} p_i^{k_i}$$

The Euler's totient function expression:

$$\varphi(N) = N \prod_{i=1}^{P} \frac{p_i - 1}{p_i}$$

Totient of a prime, p, is simply (p-1):

$$\varphi(p) = p\prod_{i=1}^{1}\frac{p_i - 1}{p_i} = p - 1$$

Quick Example

- ϕ(89) = 88
- ϕ(97) = 96

Totient of a product of primes is the product of one less than each prime

$$\varphi\left(\prod_{i=1}^{P} p_i\right) = \prod_{i=1}^{P}(p_i - 1)$$

Quick Example

- N = 89 · 97 = 8633
  - ϕ(8633) = 88 · 96 = 8448

- N = product of first ten primes ($p_i < 30$)
  - ϕ (6,469,693.230) = 1,106,472,960

The product of the first ten prime numbers, which is all prime numbers less than 30, is 6 billion, and a bunch of change, and its totient is little over 1 billion. As you can see, the numbers can get very large, very quickly. In fact the growth of the product of the first k primes grows at a rate on the order of, but not quite as fast as, the factorial function, which makes sense if you think about it.

Totient of a prime power is a simple fraction of the value of the number

$$\varphi\left(N = p^k\right) = N\left(\frac{p-1}{p}\right) = p^k\left(\frac{p-1}{p}\right) = p^{(k-1)}(p-1)$$

What about a value of N that is an integer power of a prime number? This means that we have a single prime factor, and so our totient

function reduces to one of several useful forms. The one that is probably the most useful for us is the first one, since it will be the one that is the easiest to prove when we get there in a bit. Consider the totient of an integer power of 2, it turns out that this is exactly half of the integer.

Quick Example

- $\phi(2^k) = 2^{(k-1)}$
    - $\phi(1024 = 2^{10}) = 512$

But don't fall into the trap of thinking that the totient of p to the k is always p raised to the k – 1, this is true only for p equals 2, and that's just because 2 minus 1 is 1.

Quick Example

Let's consider 7 raised to the k, here we see that we pick up an extra factor of 6, making the totient of 343 equal to 294. Meaning that more than 85% of the numbers less than 343 are relatively prime to it.

- $\phi(7^k) = 6 \cdot 7^{(k-1)}$
    - $\phi(343 = 7^k) = 6 \cdot 7^2 = 294$

Totient of an arbitrary number is the product of the totient of the prime powers

We can take our formula for the totient function, and combine it with the fundamental theorem of arithmetic. To get the result that the totient of an arbitrary number N can be expressed as the product of totients of the individual prime powers that are the factors of N. This is a consequence of the multiplicative property of the totient function, which says that the totient of the product of two positive integers is the product of the totient of the two integers. But it only applies when the two integers are relatively prime. However, we know it applies here, because if a and b are both prime powers of different prime numbers, then by definition they are relatively prime.

$$\varphi(N) = N\prod_{i=1}^{P}\frac{p_i - 1}{p_i} = \prod_{i=1}^{P} p^k \frac{p_i - 1}{p_i} = \prod_{i=1}^{P}\varphi(p^k)$$

Multiplicative property of ϕ(N)

- ϕ(a · b) = ϕ(a) · ϕ(b) IF gcd(a, b) = 1

Quick Example

ϕ(1100 = $2^2 \cdot 5^2 \cdot 11^1$)

- ϕ(1100) = 1100 · (1/2) (4/5) (10/11) = 400
- ϕ($2^2 \cdot 5^2 \cdot 11^1$) = ϕ(2^2) ϕ(5^2) ϕ(11) = 2 · 20 · 10 = 400

Using the first form of the totient function we get a result of 400. Not surprisingly, this matches the result we get if we use the multiplicative property.

Now, which form is easier to use? Well that depends on the situation, and having both forms available gives us greater flexibility in doing our work.

Proof of Totient formula requires proof of just two special cases

$$\varphi(N) = N\prod_{i=1}^{P}\frac{p_i - 1}{p_i} = \prod_{i=1}^{P} p^k \frac{p_i - 1}{p_i} = \prod_{i=1}^{P}\varphi(p^k)$$

To prove that the totient function formula is correct:

- Prove ϕ(p^k) = (N / p) (p - 1)
- Prove ϕ(a · b) = ϕ(a) · ϕ(b) IF gcd(a, b) = 1

To find totient of N=p^k, subtract number of multiples of p from number of positive integers less than N

- Number of candidates is N – 1 (i.e., 1 ≤ x ≤ N)
- Count how many values of x for which gcd(x, N) > 1
- Gcd(x, N) > 1 only if x = m · p for(1 ≤ m < N / p)

To prove our formula for the totient of a prime, p, raised to an integer

power k. We only need to subtract the number of integers less than N that are not relatively prime to N, from the number of integers that are candidates. Since any strictly positive integer x that is less than N is a candidate for being relatively prime to N, the number of candidates is N-1.

Next, we need to identify the number of these integers that are not relatively prime, meaning those values of x for which the gcd (x, N) is greater than 1. Since N only has a single prime factor, if x is not relatively prime to N, it must also have the same prime factor, meaning that x is a multiple of p. The values that m can take on, and keep x within the allowed range, is anything from 1,up to 1 less than the ratio of N/p, since p times N/p is N, which is not a candidate.

Next, we just take the difference of these two quantities, and by writing N in terms of p and k, we get the result that we expect.

$$\boldsymbol{\varphi\left(N = p^k\right) = (N-1)\ -\left(\frac{N}{p} - 1\right)}$$

$$\boldsymbol{\varphi\left(N = p^k\right) = N\left(\frac{1}{p} - 1\right) = p^k\,\frac{p_i - 1}{p_i}}$$

To find totient of N=PQ, identify groups of integers that are relatively prime to N

Next we need to show that the totient function is multiplicative when finding the totient of the product of two relatively prime numbers, P and Q. Keep in mind that P and Q do not need to be prime themselves, only relatively prime to each other.

- Create table of all x|0 ≤ x < N
- x = qP + p with 0 ≤ q < Q ; 0 ≤ p < P

	0	1	2	...	q	...	(Q-1)
0	0	P	2P		qP		(Q-1)P
1	1	P+1	2P+1		qP+1		(Q-1)P+1
2	2	P+2	2P+2		qP+2		(Q-1)P+2
...							
p	3	P+3	2P+3		qP+p		(Q-1)P+p
...							
P-1	P-1	2P-1	3P-1		(q+1)P-1		QP-1

There are a number of ways to do this proof, and perhaps the most straightforward is using the Chinese Remainder Theorem. But let's take a somewhat less elegant approach.

We know that the integers we need to consider are 1 through N-1, giving us a total of N- 1integers that are candidates. If it makes things more convenient for us, we can include either 0 or N as a potential integer candidate, and since neither is relatively prime to N, nothing changes. So to make the bookkeeping easier, we will include 0 by considering all of the nonnegative integers less than N. By using two in dices, lower case p and q, that can take on values from 0 up to, but not including, uppercase P and Q , respectively. We can generate all oft he allowed values of x as a linear combination of our two indices. If we tabulate this in a two-dimensional table, where each row represents a particular value of p, and each column represents a particular value of q, we get this.

Eliminate rows containing no values relatively prime to N, then eliminate elements in the remaining rows

- Choose p that is NOT relatively prime to P
  - gcd(p, P) = c ; p = cp' ; P = cP'
  - x = qP + p = qcP' + cp ; = c(qP' + p')
  - Every integer on row p shares a factor with P, and hence N
  - Number of rows left is equal to $\phi(P)$

If we choose a value of little p that is not relatively primed to big P, then little p and big P can both be written in terms of a common factor,

which we'll call c. Using our equation for x, in terms of little p and little q, we can factor out c, which means that every integer on that row is divisible by c, which is a factor of big P and hence a factor of N. So none of the integers on that row count toward the totient of the N. Therefore the only rows that we need to consider are the rows for which little p is relatively prime to big P. The number of such rows is, by definition, the totient of big P.

- On each row, we add P to prior entry
    - P & Q are relatively prime, Q columns
    - Each residue mod – Q appears exactly once in each row
    - Number of elements relatively prime to N is $\phi(Q)$
- $\phi(N) = \phi(P) \cdot \phi(Q)$
- Can general to any number of relatively prime factors.

Now let's consider an arbitrary row, and see how many of the integers on that row are relatively prime to big Q. As we go across each row, we add big P to the previous value. Since we know that big P and big Q are relatively prime, and since we have big Q columns, each integer in the row is in a different residue class, and each residue class is present exactly once. At this point you might be asking, how do we know this? How do we know at don't have repeats of some residue classes, and others that aren't represented at all? The key is to assume that there does exist two integers on the same row, that are in the same residue class. And use the fact that the difference between any two integers has to be a multiple of big P, combined with the fact that big P and big Q relatively prime, to show that this results in a contradiction. Once we know that every integer on a row must be in a different residue class mod Q, we can use the pigeonhole principle to establish that every residue class is represented exactly once.

Once we are at that point, we can ask how many of those elements are relatively prime to big Q , with the answer being that it is the totient of big Q. Note that this is therefore the same number for every row. Bringing this together, we have the totient of P rows that contains any

elements relatively prime to N, and each of those rows contains the totient of Q such elements. Thus the total number of elements is the torsion of P times the totient of Q , thereby proving our multiplicative property of the totient function, when the two factors are relatively prime. We can easily extend this multiplicative property to an integer that is the product of m relatively prime factors, by induction. Since any partitioning of the factors into two groups results in two overall factors of N, that are relatively prime. This completes our proof of the formula for Euler's totient function.

DISCRETE LOGARITHMS In a world where all of our numbers must be integers, logarithms are referred to as **discrete logarithms** because they can only take on distinct values.
The basic definition of the logarithm is extremely simple. If we can express a value as b raised to some exponent, then the base b logarithm of that value is simply the exponent $\mathbf{log_b\ (b^y) \equiv y}$

Similarly, the exponential function of a number to the base b is nothing more than b raised to that number $\mathbf{exp_b\ (y) \equiv b^y}$

Hence, the logarithm and exponential functions are reciprocal functions, one undoes the other. In other words, if we take a logarithm of a number, we undo an exponentiation. In the system of real numbers, we can choose any strictly positive value other than one as our base $\mathbf{exp_b\ (log_b(x)) = log_b\ (exp_b(x)) = x}$

Let's assume that our base is strictly greater than one. We have two very widely used bases. The natural laws use Euler's number, e, as the base. The value of e is an irrational number approximately equal to 2.718.

natural logs (**b = e = 2.718**…) : $\mathbf{ln(}x\mathbf{) = log_e}\ x$

A logarithm is a function that does all this work for you. We define one type of logarithm (called "log base 2" and denoted $\log_2$) to be the solution to the problem. Log base 2 is defined so that $\mathbf{\log_2 \mathit{c} = \mathit{k}}$ is the solution to the problem $\mathbf{2^{\mathit{k}} = \mathit{c}}$.

Quick Example

$\log_2 8 = 3$ is the solution to the problem $2^3 = 8$

$\log_2 4 = 2$ is the solution to the problem $2^2 = 4$

$\log_2 16 = 4$ is the solution to the problem $2^4 = 16$

$\log_2 1 = 0$ is the solution to the problem $2^0 = 1$

$\log_{10} 1000 = 3$ is the solution to the problem $10^3 = 1000$

$\log_{10} 100 = 2$ is the solution to the problem $10^2 = 100$

Not all positive integers have a discrete logarithm. What about the discrete logarithm of $\mathbf{\log_{10} 512}$? It doesn't exist. It would have to lie somewhere between 2 and 3 and we are restricted to integers. However, the discrete logarithm $\mathbf{\log_8 512 = 3}$ so whether a number has a discrete logarithm depends not only on the number but on the base.

In $\mathbf{Z^+}$ (the set of positive integers)

- if discrete log exists it is unique
- all other properties still hold
- Easy to find if log exists and what it is

This means that we do not have a one-to-one mapping between a set of positive integers and their discrete logarithms. However, while a particular integer may not have a logarithm, if it does, that logarithm is unique. In other words, since 10 to the three is 1000, we know that there is no other integer k such that 10 to the k is a thousand. While many overwhelming majority of numbers fail to have logarithms in an integer world, all of the other properties still hold. In particular, the monotonic nature of the relationship between a number and its logarithm. This means that if the log does exist, it is computationally

easy to find, and if it doesn't exist, it is computationally easy to determine that fact.

The notation commonly seen on scientific calculators for the natural logarithm function is **ln(x)**, and then **e** to the **x** for the reciprocal exponential function:

In other word $k = \ln(c)$ (1)

is the solution to the problem $e^k = c$ (2)

for any number *c*.

Since using base *e* is so natural to mathematicians, they will sometimes just use the notation log *x* instead of **ln** ***x***. However, others might use the notation **log** ***x*** for a logarithm base 10, i.e., as a shorthand notation for $\log_{10} x$. Because of this ambiguity, if someone uses log *x* without stating the base of the logarithm, you might not know what base they are implying. In that case, it's good to ask.

Since taking a logarithm is the opposite of exponentiation (more precisely, the logarithmic function $\log_b x$ is the inverse function of the exponential function b^x), we can derive the basic rules for logarithms from the basic rules for exponents.

For simplicity, we'll write the rules in terms of the natural logarithm $\ln(x)$. The rules apply for any logarithm $\log_b x$, except that you have to replace any occurrence of *e* with the new base *b*.

The natural log was defined by equations (1) and (2). If we plug the value of *k* from equation (1) into equation (2), we determine that a relationship between the natural log and the exponential function is

$e^{\ln c} = c$ (3)

Or, if we plug in the value of *c* from (2) into equation (1), we'll obtain another relationship $\ln(e^k) = k$ (4)

These equations simply state that e^x and ln x are inverse functions. We'll use equations (3) and (4) to derive the following rules for the logarithm.

Logarithmic Laws	**Formula**
Product $\ln(xy) =$	$\ln(x) + \ln(y)$
Quotient $\ln(x/y) =$	$\ln(x) - \ln(y)$
Log of power $\ln(x^y) =$	$y \ln(x)$
Log of e $\ln(e) =$	1
Log of one $\ln(1) =$	0
Log reciprocal $\ln(1 / x) =$	$- \ln(x)$

Logarithmic laws

Products:	$\log_b mn$	=	$\log_b m + \log_b n$
Ratios:	$\log_b \frac{m}{n}$	=	$\log_b m - \log_b n$
Powers:	$\log_b n^p$	=	$p \log_b n$
Roots:	$\log_b \sqrt[q]{n}$	=	$\frac{1}{q} \log_b n$
Change of bases:	$\log_b n$	=	$\log_a n \log_b a$

We also have the common logs, which is a base of 10. Most calculators use the word log for this function. But as you've likely already encountered, mathematicians prefer to use this to mean the natural log and then use the subscript 10 for the common logs.

common logs (b = 10) : log (x) ; 10^x

In computer science, it is often convenient to use a base of two and this is commonly indicated using **lg** for the name of the logarithm function to the base 2. We won't assume any particular but we will assume that the same base is used both for the log and the exponential functions if I don't specify it otherwise.

Some of The Basic Properties of Logarithms

Let's restrict ourselves to positive integers **b > 1** when we move to a modular world

1. The logarithm of one is always zero regardless of the base:
   log(1) = 0

2. The logarithms of zero or any negative number are undefined:
   log(x ≤ 0) = undefined;

   As we approach zero from the positive side (**$x \to 0^+$**), the logarithm x approaches negative infinity (**lim x = – ∞**). The logarithm x approaches negative infinity. However, while the value we are taking a logarithm of must be positive, the logarithm itself can be any real number, positive, negative, or zero. The logs of values between zero and one are negative, the log of one is zero, and the logs of values greater than one are positive.

3. The logarithm is a continuous, monotonically increasing function:
   if x > y then **log(x) > log(y)**

   Meaning that given two values of x and y, if x is greater than y, then the log of x is greater than the log of y.

   This property alone makes finding the logarithms of an arbitrary number straight foward because we can use any of several iterative techniques such as Newton-Raphson or binary search, to efficiently find the answer to an arbitrary level of accuracy. As a side note, if the base is smaller than one, then the logarithm function is monotonically decreasing.

Furthermore, there is a one-to-one mapping between the positive real numbers and their logarithms in a particular base. Each number has exactly one logarithm and vice versa:

log(x) and **exp (x)** are unique

Next, we have the log of a product is the sum of the log of the factors:

$$\log(x \cdot y) = \log(x) + \log(y)$$

The other is that the logarithm of a number raised to a power is the product of the power and the logarithm of the number. So logarithms transform multiplication into addition and exponentiation into multiplication:

$$\log (x^y) = y \cdot \log(x)$$

One of the properties that makes logarithms very powerful is that they are relatively easy to calculate:

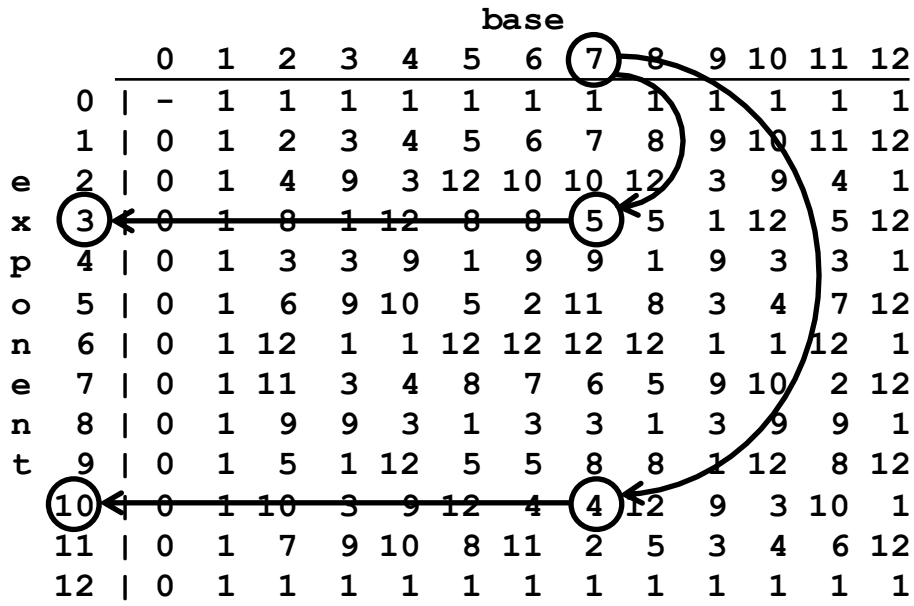
$$\log_a(x) = \log_b(x) / \log_b(a)$$

One thing that makes this true is that if we know how to find the logarithm of a number in 1 base, we can find the logarithm of that number in any other base since they differ only by a multiplicative constant.

Exponentiation Table in Mod 13 World

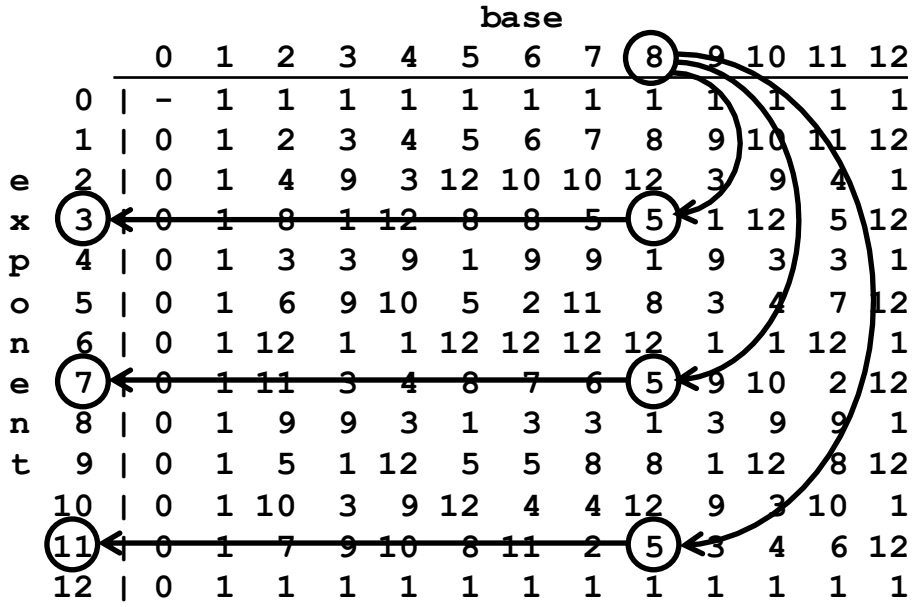
An exponentiation table can be used to find discrete logs, to find discrete logs $\log_7()$, to find discrete logs $\log_7(4)$, and to find discrete logs $\log_7(4) \equiv 10 \pmod{13}$.

Discrete logs are not monotonic $\log_7(4) \equiv 10 \pmod{13}$, and are not monotonic $\log_7(4) \equiv 10 \pmod{13} > \log_7(5) \equiv 3 \pmod{13}$.



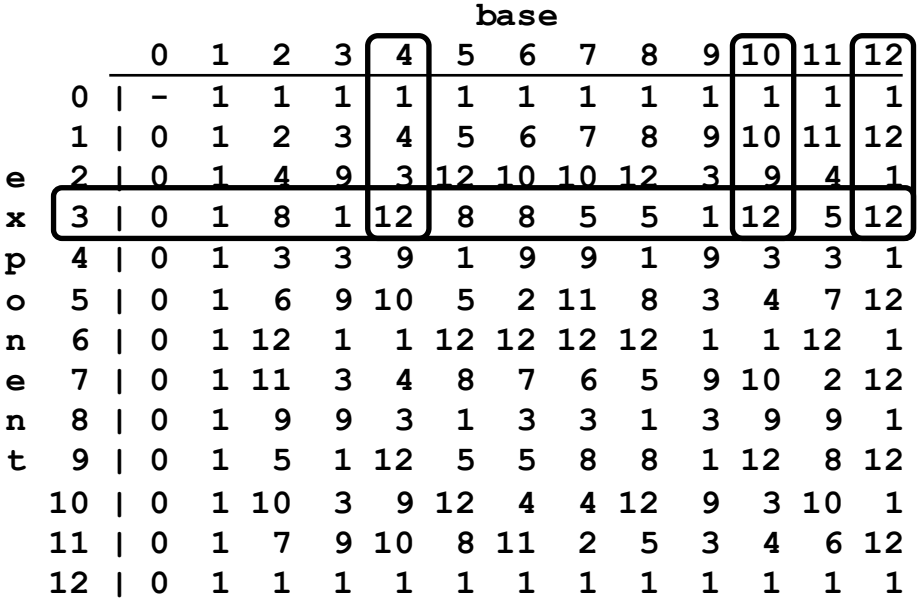
The base 7 logarithm of 4 is 10. But what is the base 7 logarithm of 5? We know that it might not exist, we might also expect that since base 7 logarithm of 5 is 10, for it to be larger than 10 if it does exist. However, not only does it exist, it turns out to be only 3. Thus our expectation of a monotonic relation simply doesn't hold. But this shouldn't trouble us because the whole notion of ordering is problematic in a modular world due to the very nature of the residue classes. Nonetheless, this lack of monotonicity is perhaps the single most important trait that makes finding discrete logarithms non-trivial.

Discrete logs are not unique. Discrete logs are not unique $\log_8(5) \equiv 3 \pmod{13}$, are not unique $\log_8(5) \equiv \{3, 5\} \pmod{13}$, and are not unique $\log_8(5) \equiv \{3, 5, 11\} \pmod{13}$.



Our logarithms may not be unique. Consider the base 8 logarithm of 5. Our table quickly shows that it is 3. But if we look further, we see that it is also 7 and also 11.

b$^{\mathbf{k}}$ not unique for a given value of k.



In addition, we can see many instances in which b to the **k** yields the same value for the same exponent using different values for the base.

While we expect this in the special case where the exponent is 0 (zero), consider the exponent of 3. We see that 4 to the 3rd, 10 to the 3rd, and 12 to the 3rd are all congruent to 12 in a mod 13 world.

$$\log_4(12) \equiv 3 \pmod{13}$$

$$\log_{10}(12) \equiv 3 \pmod{13}$$

$$\log_{12}(12) \equiv 3 \pmod{13}$$

A primitive root is a value **g** such that every integer that is relatively prime to the modulus can be written as a power of **g**. Primitive roots generate all relatively prime values. If **g** is primitive root & **gcd(a, N) = 1, $a \equiv g^k \pmod{N}$** for some **k**. This is also known as a generator of the group in number theory. Another way of phrasing this, is that **g** is the primitive root mod **N**. If every residue class that is relatively prime to **N** has a discrete logarithm base **g**.

Looking at our mod 13 table, we see that 2, 6, 7, and 11 are primitive roots. Although this is just one example, we can still glimpse that there isn't any obvious pattern here.

Before actually looking at this table, it might seem reasonable to expect that since our modulus is a prime number, that all of the integers greater than one might be primitive roots. Clearly, this isn't the case. But barring that, it would seem likely that any prime number would be a generator, yet neither 3 nor 5 are? But going the other way, it might also seem likely that if a particular number is not a generator then neither would any number that is a multiple of it. Yet six is a generator even though three is not.

		base											
		0	1	2	3	4	5	6	7	8	9	10	11	12
	0	-	1	1	1	1	1	1	1	1	1	1	1	1
	1	0	1	2	3	4	5	6	7	8	9	10	11	12
e	2	0	1	4	9	3	12	10	10	12	3	9	4	1
x	3	0	1	8	1	12	8	8	5	5	1	12	5	12
p	4	0	1	3	3	9	1	9	9	1	9	3	3	1
o	5	0	1	6	9	10	5	2	11	8	3	4	7	12
n	6	0	1	12	1	1	12	12	12	12	1	1	12	1
e	7	0	1	11	3	4	8	7	6	5	9	10	2	12
n	8	0	1	9	9	3	1	3	3	1	3	9	9	1
t	9	0	1	5	1	12	5	5	8	8	1	12	8	12
	10	0	1	10	3	9	12	4	4	12	9	3	10	1
	11	0	1	7	9	10	8	11	2	5	3	4	6	12
	12	0	1	1	1	1	1	1	1	1	1	1	1	1

Cyclic Groups and Generators

Some groups have an interesting property: all the elements in the group can be obtained by repeatedly applying the group operation to a particular group element. If a group has such a property, it is called a cyclic group and the particular group element is called a generator. A trivial example is the group Zn, the additive group of integers modulo n. In Zn, 1 is always a generator:

$2 \equiv 2 \bmod 5$

$2+2 \equiv 4 \bmod 5$

$2+2+2 \equiv 6 \equiv 1 \bmod$

5 $2+2+2+2 \equiv 8 \equiv 3 \bmod 5$

$2+2+2+2+2 \equiv 10 \equiv 0 \bmod 5$

If a group is cyclic, then there may exist multiple generators. For example, we know Z5 is a cyclic group. The element 2 is a generator for sure. And if we take a look at 1, we can find:

$1 \equiv 1 \bmod n$

$1 + 1 \equiv 2 \bmod n$

$1 + 1 + 1 \equiv 3 \bmod n$

...

$1 + 1 + 1 + \ldots + 1 \equiv n \equiv 0 \bmod n$

So all the group elements {0, 1, 2, 3, 4} in Z_5 can also be generated by 2. That is to say, 2 is also a generator for the group Z_5.

Not every element in a group is a generator. For example, the identity element in a group will never be a generator. No matter how many times you apply the group operator to the identity element, the only element you can yield is the identity element itself. For example, in Z_n, 0 is the identity element and $0 + 0 + ... + 0 \equiv 0 \bmod n$ in all cases.

Not every group is cyclic. For example, Z_n^*, the multiplicative group modulo n, is cyclic if and only if n is 1 or 2 or 4 or p^k or $2 * p^k$ for an odd prime number p and $k \geq 1$. So Z_5^* must be a cyclic group because 5 is a prime number. Actually all the elements in Z_5^*, {1, 2, 3, 4} can be generated by 2:

$2^1 \equiv 2 \bmod 5$

$2^2 \equiv 4 \bmod 5$

$2^3 \equiv 8 \equiv 3 \bmod 5$

$2^4 \equiv 16 \equiv 1 \bmod 5$

And Z_{12}^* is not a cyclic group. The elements in Z_{12}^* are: {1, 5, 7, 11}. Obviously the identity element 1 cannot be a generator. Let's check the other three elements:

$51 \equiv 5 \bmod 12$	$71 \equiv 7 \bmod 12$	$111 \equiv 11 \bmod 12$
$52 \equiv 25 \equiv 1 \bmod 12$	$72 \equiv 49 \equiv 1 \bmod 12$	$112 \equiv 121 \equiv 1 \bmod 12$

None of the elements can generate the whole group. Therefore, none of them is a generator. So Z_{12}^* is indeed not cyclic. If Z_n^* is cyclic and g is a generator of Z_n^*, then g is also called a primitive root modulo n.

Discrete Logarithm Problem

Over the decades, countless very bright people have attempted to find an efficient solution and many have made chipping away at the problem, the focus of their life's work. Their collective failure to do so leads a lot of people to suspect that no such algorithm exists, enough so that governments and militaries are willing to rely on that being the case. But every year or so, we hear about another problem being solved that was suspected of being unsolvable, precisely because countless people had worked on it without success. Sometimes for centuries.

Discrete logarithms are logarithms defined with regard to multiplicative cyclic groups. If G is a multiplicative cyclic group and g is a generator of G, then from the definition of cyclic groups, we know every element h in G can be written as g^x for some x. The discrete logarithm to the base g of h in the group G is defined to be x. For example, if the group is Z_5^*, and the generator is 2, then the discrete logarithm of 1 is 4 because $2^4 \equiv 1 \bmod 5$.

The discrete logarithm problem is defined as: given a group G, a generator g of the group and an element h of G, to find the discrete logarithm to the base g of h in the group G. Discrete logarithm problem is not always hard.

The hardness of finding discrete logarithms depends on the groups. For example, a popular choice of groups for discrete logarithm based crypto-systems is Z_p^* where p is a prime number. However, if $p-1$ is a product of small primes, then the Pohlig–Hellman algorithm can solve the discrete logarithm problem in this group very efficiently. That's why we always want p to be a safe prime when using Z_p^* as the basis of discrete logarithm based crypto-systems.

A safe prime is a prime number which equals $2q + 1$ where q is a large prime number. This guarantees that $p - 1 = 2q$ has a large prime factor so that the Pohlig–Hellman algorithm cannot solve the discrete logarithm problem easily. Even p is a safe prime, there is a sub-

exponential algorithm which is called the index calculus. That means *p* must be very large (usually at least 1024-bit) to make the crypto-systems safe.

Some modern crypto relies on no efficient solution to Discrete Log Problem even though such a solution could be found tomorrow. Finding discrete logarithms in a modular world is "hard":

- $\log_7 (8) \equiv 9 \pmod{13}$
- $\log_7 (8) \not\equiv \log_7 (4) + \log_7 (2) \equiv 10 + 11 \equiv 8 \pmod{13}$
- They do not obey normal rules even when they exist
- Logarithms live in a mod - totient world
    - $b^x \equiv b^y \pmod{N}$ if $x \equiv y \pmod{\phi(N)}$
    - $7^9 \equiv 7^{10} \cdot 7^{11} \equiv 7^{(10 + 11) \bmod 12} \equiv 7^9 \pmod{13}$

No reliable pattern exists to help find discrete logs. Discrete Log Problem is no known efficient way to find them ; "efficient" = poly - time in number of bits of modulus.

CHINESE REMAINDER THEOREM

Quick Example 1

Let $x \equiv 2 \pmod{3}$

$x \equiv 3 \pmod{4}$

$x \equiv 4 \pmod{5}$

Find **X**

To find x, firstly we need to find the LCM (lowest common multiple) of m which is $3 \cdot 4 \cdot 5 = 60$. Then 1 has to be the gcd (4, 5) = 1, the gcd (3, 5) = 1, and the gcd (3, 4) = 1. So now let's find m_1, m_2 and m_3 which is:

$m_1 = 4 \cdot 5 = 20 \pmod{3}$

$m_2 = 3 \cdot 5 = 15 \pmod{4}$ and $m_3 = 3 \cdot 4 = 12 \pmod{5}$

Now we want x to be equivalent to x = 2 (mod 3), so
$x = 20 + 15 + 12 \pmod 3$
$x = 20 + 0 + 0 \pmod 3$
$x = 20 \pmod 3$
$x = 2 \pmod 3$

Now we want x to be equivalent to x = 4 (mod 5), so
$x = 20 + 15 + 12 \pmod 5$
$x = 0 + 0 + 12$
$x = 12 \pmod 5$
$x = 2 \pmod 5$, but we want to be 4
So we can quickly find that $2 \cdot 7 = 14 = 4 \pmod 5$, so that
$x = 2 \pmod 5$
$2 \cdot 7 = 14 = 4 \pmod 5$

Now we want x to be equivalent to x = 3 (mod 4), so
$x = 20 + 15 + 12 \cdot 7 \pmod 4$
$x = 20 + 15 + 84 \pmod 4$
$x = 15 = 3 \pmod 4$
$x = 3 \pmod 4$

And we are now done!

If we now add all those 20 + 15 + 84 up we will get x = 119. And we will surely find that
$119 = 39 \cdot 3 + 2$
$119 = 29 \cdot 4 + 3$
$119 = 23 \cdot 5 + 4$

But since we are dealing with mod 3, 4, and 5 which is $3 \cdot 4 \cdot 5 = 60$, and if we add 119 + 60 = 179. So the final answer is

$x = 119 \pmod{60}$
$\mathbf{x = 59} \pmod{60}$

Quick Example 2

Let $x \equiv 2 \pmod 3$

$x \equiv 2 \pmod 4$

$x \equiv 1 \pmod 5$

Find **X**

To find x, firstly we need to find the LCM (lowest common multiple) of 3, 4, and 5 which is $3 \cdot 4 \cdot 5 = 60$. Then 1 has to be the gcd (4, 5) = 1, the gcd (3, 5) = 1, and the gcd (3, 4) = 1. So now let's find m_1, m_2 and m_3 which is:

$m_1 = 4 \cdot 5 = 20 \pmod 3$

$m_2 = 3 \cdot 5 = 15 \pmod 4$ and $m_3 = 3 \cdot 4 = 12 \pmod 5$

Now we want x to be equivalent to $x = 2 \pmod 3$, so

$x = 20 + 15 + 12 \pmod 3$

$x = 20 + 0 + 0 \pmod 3$

$x = 20 \pmod 3$

$x = 2 \pmod 3$

Now we want x to be equivalent to $x = 1 \pmod 5$, so

$x = 20 + 15 + 12 \pmod 5$

$x = 0 + 0 + 12$

$x = 12 \pmod 5$

$x = 2 \pmod 5$, but we want to be 1

So we can quickly find that $2 \cdot 3 = 6 = 1 \pmod 5$, so that

$x = 2 \pmod 5$

$2 \cdot 3 = 6 = 1 \pmod 5$

Now we want x to be equivalent to $x = 2 \pmod 4$, so

$x = 20 + 15 + 12 \cdot 3 \pmod 4$

$x = 20 + 15 + 36 \pmod 4$

$x = 15 = 3 \pmod 4$

To get $x = 2 \pmod 4$ we have to firstly consider $3 \pmod 4$ - which we have already calculated it, and we'll go from $1 \pmod 4$ to $2 \pmod 4$. So now:

To get 1 (mod 4) from x = 15 = 3 (mod 4) is we can quickly find that
$3 \cdot 3 = 9 = 1 \pmod 4$

Now to get 2 (mod 4) we can calculate
$x = 15 \cdot 3 \cdot 2 \pmod 4$
$x = 90 \pmod 4$
$x = 22 \pmod 4$
$x = 2 \pmod 4$

And we are now done!

If we now add all those $20 + 15 \cdot 3 \cdot 2 + 36$ up we will get x = 146. And we will surely find that
$146 = 48 \cdot 3 + 2$
$146 = 36 \cdot 4 + 2$
$146 = 29 \cdot 5 + 1$
But since we are dealing with mod 3, 4, and 5 which is $3 \cdot 4 \cdot 5 = 60$, and if we add 146 + 60 = 206. So the final answer is

$x = 146 \pmod{60}$
$\mathbf{x = 26} \pmod{60}$

Quick Example 3
Let $x \equiv 1 \pmod 3$
$x \equiv 2 \pmod 4$
$x \equiv 4 \pmod 5$
Find **X**
To find x, firstly we need to find the LCM (lowest common multiple) of m which is $3 \cdot 4 \cdot 5 = 60$. Then 1 has to be the gcd (4, 5) = 1, the gcd (3, 5) = 1, and the gcd (3, 4) = 1. So now let's find m_1, m_2 and m_3 which is:
$m_1 = 4 \cdot 5 = 20\ y_1$
$m_2 = 3 \cdot 5 = 15\ y_2$ and $m_3 = 3 \cdot 4 = 12\ y_3$

Now we want x to find y_1, so
$20\ y_1 \equiv 1 \pmod 3$
$2\ y_1 \equiv 1 \pmod 3$
$y_1 \equiv 2$

Now we want x to find y_2, so
$15\ y_2 \equiv 1 \pmod 4$
$3\ y_2 \equiv 1 \pmod 4$
$y_2 \equiv 3$

Now we want x to find y_3, so
$12\ y_3 \equiv 1 \pmod 5$
$2\ y_3 \equiv 1 \pmod 5$
$y_3 \equiv 3$

To yield final answer for **x**, we calculate:

$1 \cdot 20 \cdot 2 = 40$
$2 \cdot 15 \cdot 3 = 90$
$\underline{4 \cdot 12 \cdot 3 = 144\ +}$
$x \equiv 274 \pmod{60}$
$x \equiv 34 \pmod{60}$
$x = 34$

And now we are done!

FERMAT'S PRIMALITY AND MILLER-RABIN TEST Fermat's Primality tests seek to determine if a number is prime without factoring it. Factoring large numbers is hard and we can keep choosing larger numbers. Primality tests do not attempt to factor the number. Fermat's Primality Test is a probabilistic, based on Fermat's Little Theorem:

$$a^p \equiv a \pmod p$$

If it fails, number is NOT prime. If it passes, number MAY be prime.

Fermat's Primality Test assumes the number is prime and checks for behavior inconsistent with a prime. If a and p are relatively prime, a^{-1} exists

$$a^{p-1} \equiv 1 \pmod{p}$$

The algorithm:

- Pick a random value for p of the right size
- Pick a value of a such that $1 < a < p - 1$
- Check if $a^{p-1} \equiv 1 \pmod{p}$. If NO: a is a fermat witness to compositeness of p. If YES: there are two possibilities
  - p is prime
  - p is a pseudoprime to base a; a is a Fermat liar

Is 15 prime? Let's find out by randomly picking a huge candidate prime, say p = 15 and let's pick a = 4

- Is $4^{14} \equiv 1 \pmod{15}$? YES
- 15 might be prime

Now Randomly pick a = 2

- Is $2^{14} \equiv 1 \pmod{15}$? NO (it's congruent to 4)
- 15 is NOT prime (2 is a Fermat witness for 15)
- 15 is a pseudo prime to the base 4 (4 is a Fermat liar for 15)

Fermat's Primality Test is a simple and efficient, but it has a significant flaw in that, there exists some numbers, such as the Carmichael numbers, for which it is incapable of identifying the fact that they are not prime.

Fermat's Little Theorem lets us cheat $a^{n/1} \equiv 1 \bmod n$; If n is prime.

Start with carefully chosen value of a^d such that squaring a^d an integer number of times yields $a^{(n-1)}$. If this is congruent to 1, this value of a

passes Fermat. Last value squared is potentially nontrivial root of 1, if congruent to 1, proceed back down list.

If we reach a^d without a nontrivial root of 1, test passes, assume n is prime or repeat test with different a.

Relatively few trials need for good security. 1024-bit RSA primes: 5 trials yield 112-bits of security. 133-bit RSA primes: 39 trials yield 112-bit security.

There are many more sophisticated tests at the expense of speed, that are able to identify prime numbers with a higher degree of certainty. And while some of them are deterministic, meaning that they will tell us definitively if the number is prime or not, most of them are still probabilistic in nature. The deterministic tests are just too slow for most purposes, so we won't consider them further. Perhaps the most common next step up in probabilistic tests is Miller-Rabin. This test takes an observation regarding the square roots of one in a modular field and then uses for Fermat's little theorem to exploit that observation.

Miller-Rabin test is actually quite tame. Let’s pick a random value for n then prefilter with Fermat’s Primality Test if desired.

Compute (n-1) and determine the value of s. Number of times division by 2 needed to get an odd d.

Calculate $x \equiv a^d$, If congruent to -1, +1, end test with PASS (possible prime).

Square x a total of s times and, at each step. If $x \equiv -1 \pmod{n}$, end test with PASS (possible prime). End test with COMPOSITE (is not prime).

Quick Examples
n = 561 (smallest Charmichael number)
s = 4, d = 35
a = 7
$7^{35} \equiv 241 \pmod{561}$
Four squares: 298, 166, 67, 1
561 is NOT a prime number.

n = 563
s = 1, d = 281
a = 7
$7^{281} \equiv -1$
563 MAY be a prime number.

SIEVE OF ERATOSTHENES A prime number is a natural number that has exactly two distinct natural number divisors: the number 1 and itself. In mathematics, the sieve of Eratosthenes is an ancient algorithm for finding all prime numbers up to any given limit. It does so by iteratively marking as composite (i.e., not prime) the multiples of each prime, starting with the first prime number, 2. The multiples of a given prime are generated as a sequence of numbers starting from that prime, with constant difference between them that is equal to that prime. This is the sieve's key distinction from using trial division to sequentially test each candidate number for divisibility by each prime. Once all the multiples of each discovered prime have been marked as composites, the remaining unmarked numbers are primes.

To find all the prime numbers less than or equal to 30, proceed as follows.

First, generate a list of integers from 2 to 30 is: 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

The first number in the list is 2; cross out every 2nd number in the list after 2 by counting up from 2 in increments of 2 (these will be all the multiples of 2 in the list): 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ 9 ~~10~~ 11 ~~12~~ 13 ~~14~~ 15 ~~16~~ 17 ~~18~~ 19 ~~20~~ 21 ~~22~~ 23 ~~24~~ 25 ~~26~~ 27 ~~28~~ 29 ~~30~~

The next number in the list after 2 is 3; cross out every 3rd number in the list after 3 by counting up from 3 in increments of 3 (these will be all the multiples of 3 in the list): 2 3 ~~4~~ 5 ~~6~~ 7 ~~8 9 10~~ 11 ~~12~~ 13 ~~14 15 16~~ 17 ~~18~~ 19 ~~20 21 22~~ 23 ~~24~~ 25 ~~26 27 28~~ 29 ~~30~~

The next number not yet crossed out in the list after 3 is 5; cross out every 5th number in the list after 5 by counting up from 5 in in increments of 5 (i.e. all the multiples of 5): 2 3 ~~4~~ 5 ~~6~~ 7 ~~8 9 10~~ 11 ~~12~~ 13 ~~14 15 16~~ 17 ~~18~~ 19 ~~20 21 22~~ 23 ~~24 25 26 27 28~~ 29 ~~30~~

The next number not yet crossed out in the list after 5 is 7; the next step would be to cross out every 7th number in the list after 7, but they are all already crossed out at this point, as these numbers (14, 21, 28) are also multiples of smaller primes because 7 × 7 is greater than 30. The numbers not crossed out at this point in the list are all the prime numbers below 30:

2 3 5 7 11 13 17 19 23 29

(https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes)

COMPOSITE NUMBER LIST To make life a bit easier and to save times in finding a prime number, use this following composite number list to determine a prime number. So another word, if a number/value is not in this list then it is possibly a prime number.

4, 6, 8, 9, 10, 12, 14, 15, 16, 18, 20, 21, 22, 24, 25, 26, 27, 28, 30, 32, 33, 34, 35, 36, 38, 39, 40, 42, 44, 45, 46, 48, 49, 50, 51, 52, 54, 55, 56, 57, 58, 60, 62, 63, 64, 65, 66, 68, 69, 70, 72, 74, 75, 76, 77, 78, 80, 81, 82, 84, 85, 86, 87, 88, 90, 91, 92, 93, 94, 95, 96, 98, 99, 100, 102, 104, 105, 106, 108, 110, 111, 112, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 128, 129, 130, 132, 133, 134, 135, 136, 138, 140, 141, 142, 143, 144, 145, 146, 147, 148, 150, 152, 153, 154, 155, 156, 158, 159, 160, 161, 162, 164, 165, 166, 168, 169, 170, 171, 172, 174, 175, 176, 177, 178, 180, 182, 183, 184, 185, 186, 187, 188, 189, 190, 192, 194, 195, 196, 198, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 224, 225, 226, 228, 230, 231, 232, 234, 235, 236, 237, 238, 240, 242, 243, 244, 245, 246, 247, 248, 249, 250, 252, 253, 254, 255, 256, 258, 259, 260, 261, 262, 264, 265, 266, 267, 268, 270, 272, 273, 274, 275, 276, 278, 279, 280, 282, 284, 285, 286, 287, 288, 289, 290, 291, 292, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 308, 309, 310, 312, 314, 315, 316, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 332, 333, 334, 335, 336, 338, 339, 340, 341, 342, 343, 344, 345, 346, 348, 350, 351, 352, 354, 355, 356, 357, 358, 360, 361, 362, 363, 364, 365, 366, 368, 369, 370, 371, 372, 374, 375, 376, 377, 378, 380, 381, 382, 384, 385, 386, 387, 388, 390, 391, 392, 393, 394, 395, 396, 398, 399, 400, 402, 403, 404, 405, 406, 407, 408, 410, 411, 412, 413, 414, 415, 416, 417, 418, 420, 422, 423, 424, 425, 426, 427, 428, 429, 430, 432, 434, 435, 436, 437, 438, 440, 441, 442, 444, 445, 446, 447, 448, 450, 451, 452, 453, 454, 455, 456, 458, 459, 460, 462, 464, 465, 466, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 480, 481, 482, 483, 484, 485, 486, 488, 489, 490, 492, 493, 494, 495, 496, 497, 498, 500, 501, 502, 504, 505, 506, 507, 508, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 522, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 542,

543, 544, 545, 546, 548, 549, 550, 551, 552, 553, 554, 555, 556, 558, 559, 560, 561, 562, 564, 565, 566, 567, 568, 570, 572, 573, 574, 575, 576, 578, 579, 580, 581, 582, 583, 584, 585, 586, 588, 589, 590, 591, 592, 594, 595, 596, 597, 598, 600, 602, 603, 604, 605, 606, 608, 609, 610, 611, 612, 614, 615, 616, 618, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 632, 633, 634, 635, 636, 637, 638, 639, 640, 642, 644, 645, 646, 648, 649, 650, 651, 652, 654, 655, 656, 657, 658, 660, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 674, 675, 676, 678, 679, 680, 681, 682, 684, 685, 686, 687, 688, 689, 690, 692, 693, 694, 695, 696, 697, 698, 699, 700, 702, 703, 704, 705, 706, 707, 708, 710, 711, 712, 713, 714, 715, 716, 717, 718, 720, 721, 722, 723, 724, 725, 726, 728, 729, 730, 731, 732, 734, 735, 736, 737, 738, 740, 741, 742, 744, 745, 746, 747, 748, 749, 750, 752, 753, 754, 755, 756, 758, 759, 760, 762, 763, 764, 765, 766, 767, 768, 770, 771, 772, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 788, 789, 790, 791, 792, 793, 794, 795, 796, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 810, 812, 813, 814, 815, 816, 817, 818, 819, 820, 822, 824, 825, 826, 828, 830, 831, 832, 833, 834, 835, 836, 837, 838, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 854, 855, 856, 858, 860, 861, 862, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 878, 879, 880, 882, 884, 885, 886, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 908, 909, 910, 912, 913, 914, 915, 916, 917, 918, 920, 921, 922, 923, 924, 925, 926, 927, 928, 930, 931, 932, 933, 934, 935, 936, 938, 939, 940, 942, 943, 944, 945, 946, 948, 949, 950, 951, 952, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 968, 969, 970, 972, 973, 974, 975, 976, 978, 979, 980, 981, 982, 984, 985, 986, 987, 988, 989, 990, 992, 993, 994, 995, 996, 998, 999, 1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1010, 1011, 1012, 1014, 1015, 1016, 1017, 1018, 1020, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1032, 1034, 1035, 1036, 1037, 1038, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1050, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1062, 1064, 1065, 1066, 1067, 1068, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1088, 1089, 1090, 1092, 1094,

1095, 1096, 1098, 1099, 1100, 1101, 1102, 1104, 1105, 1106, 1107, 1108, 1110, 1111, 1112, 1113, 1114, 1115, 1116, 1118, 1119, 1120, 1121, 1122, 1124, 1125, 1126, 1127, 1128, 1130, 1131, 1132, 1133, 1134, 1135, 1136, 1137, 1138, 1139, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1150, 1152, 1154, 1155, 1156, 1157, 1158, 1159, 1160, 1161, 1162, 1164, 1165, 1166, 1167, 1168, 1169, 1170, 1172, 1173, 1174, 1175, 1176, 1177, 1178, 1179, 1180, 1182, 1183, 1184, 1185, 1186, 1188, 1189, 1190, 1191, 1192, 1194, 1195, 1196, 1197, 1198, 1199, 1200, 1202, 1203, 1204, 1205, 1206, 1207, 1208, 1209, 1210, 1211, 1212, 1214, 1215, 1216, 1218, 1219, 1220, 1221, 1222, 1224, 1225, 1226, 1227, 1228, 1230, 1232, 1233, 1234, 1235, 1236, 1238, 1239, 1240, 1241, 1242, 1243, 1244, 1245, 1246, 1247, 1248, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1260, 1261, 1262, 1263, 1264, 1265, 1266, 1267, 1268, 1269, 1270, 1271, 1272, 1273, 1274, 1275, 1276, 1278, 1280, 1281, 1282, 1284, 1285, 1286, 1287, 1288, 1290, 1292, 1293, 1294, 1295, 1296, 1298, 1299, 1300, 1302, 1304, 1305, 1306, 1308, 1309, 1310, 1311, 1312, 1313, 1314, 1315, 1316, 1317, 1318, 1320, 1322, 1323, 1324, 1325, 1326, 1328, 1329, 1330, 1331, 1332, 1333, 1334, 1335, 1336, 1337, 1338, 1339, 1340, 1341, 1342, 1343, 1344, 1345, 1346, 1347, 1348, 1349, 1350, 1351, 1352, 1353, 1354, 1355, 1356, 1357, 1358, 1359, 1360, 1362, 1363, 1364, 1365, 1366, 1368, 1369, 1370, 1371, 1372, 1374, 1375, 1376, 1377, 1378, 1379, 1380, 1382, 1383, 1384, 1385, 1386, 1387, 1388, 1389, 1390, 1391, 1392, 1393, 1394, 1395, 1396, 1397, 1398, 1400, 1401, 1402, 1403, 1404, 1405, 1406, 1407, 1408, 1410, 1411, 1412, 1413, 1414, 1415, 1416, 1417, 1418, 1419, 1420, 1421, 1422, 1424, 1425, 1426, 1428, 1430, 1431, 1432, 1434, 1435, 1436, 1437, 1438, 1440, 1441, 1442, 1443, 1444, 1445, 1446, 1448, 1449, 1450, 1452, 1454, 1455, 1456, 1457, 1458, 1460, 1461, 1462, 1463, 1464, 1465, 1466, 1467, 1468, 1469, 1470, 1472, 1473, 1474, 1475, 1476, 1477, 1478, 1479, 1480, 1482, 1484, 1485, 1486, 1488, 1490, 1491, 1492, 1494, 1495, 1496, 1497, 1498, 1500, 1501, 1502, 1503, 1504, 1505, 1506, 1507, 1508, 1509, 1510, 1512, 1513, 1514, 1515, 1516, 1517, 1518, 1519, 1520, 1521, 1522, 1524, 1525, 1526, 1527, 1528, 1529, 1530, 1532, 1533, 1534, 1535, 1536, 1537, 1538,

1539, 1540, 1541, 1542, 1544, 1545, 1546, 1547, 1548, 1550, 1551, 1552, 1554, 1555, 1556, 1557, 1558, 1560, 1561, 1562, 1563, 1564, 1565, 1566, 1568, 1569, 1570, 1572, 1573, 1574, 1575, 1576, 1577, 1578, 1580, 1581, 1582, 1584, 1585, 1586, 1587, 1588, 1589, 1590, 1591, 1592, 1593, 1594, 1595, 1596, 1598, 1599, 1600, 1602, 1603, 1604, 1605, 1606, 1608, 1610, 1611, 1612, 1614, 1615, 1616, 1617, 1618, 1620, 1622, 1623, 1624, 1625, 1626, 1628, 1629, 1630, 1631, 1632, 1633, 1634, 1635, 1636, 1638, 1639, 1640, 1641, 1642, 1643, 1644, 1645, 1646, 1647, 1648, 1649, 1650, 1651, 1652, 1653, 1654, 1655, 1656, 1658, 1659, 1660, 1661, 1662, 1664, 1665, 1666, 1668, 1670, 1671, 1672, 1673, 1674, 1675, 1676, 1677, 1678, 1679, 1680, 1681, 1682, 1683, 1684, 1685, 1686, 1687, 1688, 1689, 1690, 1691, 1692, 1694, 1695, 1696, 1698, 1700, 1701, 1702, 1703, 1704, 1705, 1706, 1707, 1708, 1710, 1711, 1712, 1713, 1714, 1715, 1716, 1717, 1718, 1719, 1720, 1722, 1724, 1725, 1726, 1727, 1728, 1729, 1730, 1731, 1732, 1734, 1735, 1736, 1737, 1738, 1739, 1740, 1742, 1743, 1744, 1745, 1746, 1748, 1749, 1750, 1751, 1752, 1754, 1755, 1756, 1757, 1758, 1760, 1761, 1762, 1763, 1764, 1765, 1766, 1767, 1768, 1769, 1770, 1771, 1772, 1773, 1774, 1775, 1776, 1778, 1779, 1780, 1781, 1782, 1784, 1785, 1786, 1788, 1790, 1791, 1792, 1793, 1794, 1795, 1796, 1797, 1798, 1799, 1800, 1802, 1803, 1804, 1805, 1806, 1807, 1808, 1809, 1810, 1812, 1813, 1814, 1815, 1816, 1817, 1818, 1819, 1820, 1821, 1822, 1824, 1825, 1826, 1827, 1828, 1829, 1830, 1832, 1833, 1834, 1835, 1836, 1837, 1838, 1839, 1840, 1841, 1842, 1843, 1844, 1845, 1846, 1848, 1849, 1850, 1851, 1852, 1853, 1854, 1855, 1856, 1857, 1858, 1859, 1860, 1862, 1863, 1864, 1865, 1866, 1868, 1869, 1870, 1872, 1874, 1875, 1876, 1878, 1880, 1881, 1882, 1883, 1884, 1885, 1886, 1887, 1888, 1890, 1891, 1892, 1893, 1894, 1895, 1896, 1897, 1898, 1899, 1900, 1902, 1903, 1904, 1905, 1906, 1908, 1909, 1910, 1911, 1912, 1914, 1915, 1916, 1917, 1918, 1919, 1920, 1921, 1922, 1923, 1924, 1925, 1926, 1927, 1928, 1929, 1930, 1932, 1934, 1935, 1936, 1937, 1938, 1939, 1940, 1941, 1942, 1943, 1944, 1945, 1946, 1947, 1948, 1950, 1952, 1953, 1954, 1955, 1956, 1957, 1958, 1959, 1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971, 1972, 1974, 1975, 1976, 1977, 1978, 1980,

1981, 1982, 1983, 1984, 1985, 1986, 1988, 1989, 1990, 1991, 1992, 1994, 1995, 1996, 1998, 2000, 2001, 2002, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2012, 2013, 2014, 2015, 2016, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2028, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2064, 2065, 2066, 2067, 2068, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2082, 2084, 2085, 2086, 2088, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2112, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125, 2126, 2127, 2128, 2130, 2132, 2133, 2134, 2135, 2136, 2138, 2139, 2140, 2142, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2162, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 2177, 2178, 2180, 2181, 2182, 2183, 2184, 2185, 2186, 2187, 2188, 2189, 2190, 2191, 2192, 2193, 2194, 2195, 2196, 2197, 2198, 2199, 2200, 2201, 2202, 2204, 2205, 2206, 2208, 2209, 2210, 2211, 2212, 2214, 2215, 2216, 2217, 2218, 2219, 2220, 2222, 2223, 2224, 2225, 2226, 2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2235, 2236, 2238, 2240, 2241, 2242, 2244, 2245, 2246, 2247, 2248, 2249, 2250, 2252, 2253, 2254, 2255, 2256, 2257, 2258, 2259, 2260, 2261, 2262, 2263, 2264, 2265, 2266, 2268, 2270, 2271, 2272, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2282, 2283, 2284, 2285, 2286, 2288, 2289, 2290, 2291, 2292, 2294, 2295, 2296, 2298, 2299, 2300, 2301, 2302, 2303, 2304, 2305, 2306, 2307, 2308, 2310, 2312, 2313, 2314, 2315, 2316, 2317, 2318, 2319, 2320, 2321, 2322, 2323, 2324, 2325, 2326, 2327, 2328, 2329, 2330, 2331, 2332, 2334, 2335, 2336, 2337, 2338, 2340, 2342, 2343, 2344, 2345, 2346, 2348, 2349, 2350, 2352, 2353, 2354, 2355, 2356, 2358, 2359, 2360, 2361, 2362, 2363, 2364, 2365, 2366, 2367, 2368, 2369, 2370, 2372, 2373, 2374, 2375, 2376, 2378, 2379, 2380, 2382, 2384, 2385, 2386, 2387, 2388, 2390, 2391, 2392, 2394, 2395, 2396, 2397, 2398, 2400, 2401, 2402, 2403, 2404, 2405, 2406, 2407, 2408, 2409, 2410, 2412, 2413, 2414, 2415, 2416, 2418, 2419, 2420, 2421, 2422, 2424, 2425, 2426,

2427, 2428, 2429, 2430, 2431, 2432, 2433, 2434, 2435, 2436, 2438, 2439, 2440, 2442, 2443, 2444, 2445, 2446, 2448, 2449, 2450, 2451, 2452, 2453, 2454, 2455, 2456, 2457, 2458, 2460, 2461, 2462, 2463, 2464, 2465, 2466, 2468, 2469, 2470, 2471, 2472, 2474, 2475, 2476, 2478, 2479, 2480, 2481, 2482, 2483, 2484, 2485, 2486, 2487, 2488, 2489, 2490, 2491, 2492, 2493, 2494, 2495, 2496, 2497, 2498, 2499, 2500, 2501, 2502, 2504, 2505, 2506, 2507, 2508, 2509, 2510, 2511, 2512, 2513, 2514, 2515, 2516, 2517, 2518, 2519, 2520, 2522, 2523, 2524, 2525, 2526, 2527, 2528, 2529, 2530, 2532, 2533, 2534, 2535, 2536, 2537, 2538, 2540, 2541, 2542, 2544, 2545, 2546, 2547, 2548, 2550, 2552, 2553, 2554, 2555, 2556, 2558, 2559, 2560, 2561, 2562, 2563, 2564, 2565, 2566, 2567, 2568, 2569, 2570, 2571, 2572, 2573, 2574, 2575, 2576, 2577, 2578, 2580, 2581, 2582, 2583, 2584, 2585, 2586, 2587, 2588, 2589, 2590, 2592, 2594, 2595, 2596, 2597, 2598, 2599, 2600, 2601, 2602, 2603, 2604, 2605, 2606, 2607, 2608, 2610, 2611, 2612, 2613, 2614, 2615, 2616, 2618, 2619, 2620, 2622, 2623, 2624, 2625, 2626, 2627, 2628, 2629, 2630, 2631, 2632, 2634, 2635, 2636, 2637, 2638, 2639, 2640, 2641, 2642, 2643, 2644, 2645, 2646, 2648, 2649, 2650, 2651, 2652, 2653, 2654, 2655, 2656, 2658, 2660, 2661, 2662, 2664, 2665, 2666, 2667, 2668, 2669, 2670, 2672, 2673, 2674, 2675, 2676, 2678, 2679, 2680, 2681, 2682, 2684, 2685, 2686, 2688, 2690, 2691, 2692, 2694, 2695, 2696, 2697, 2698, 2700, 2701, 2702, 2703, 2704, 2705, 2706, 2708, 2709, 2710, 2712, 2714, 2715, 2716, 2717, 2718, 2720, 2721, 2722, 2723, 2724, 2725, 2726, 2727, 2728, 2730, 2732, 2733, 2734, 2735, 2736, 2737, 2738, 2739, 2740, 2742, 2743, 2744, 2745, 2746, 2747, 2748, 2750, 2751, 2752, 2754, 2755, 2756, 2757, 2758, 2759, 2760, 2761, 2762, 2763, 2764, 2765, 2766, 2768, 2769, 2770, 2771, 2772, 2773, 2774, 2775, 2776, 2778, 2779, 2780, 2781, 2782, 2783, 2784, 2785, 2786, 2787, 2788, 2790, 2792, 2793, 2794, 2795, 2796, 2798, 2799, 2800, 2802, 2804, 2805, 2806, 2807, 2808, 2809, 2810, 2811, 2812, 2813, 2814, 2815, 2816, 2817, 2818, 2820, 2821, 2822, 2823, 2824, 2825, 2826, 2827, 2828, 2829, 2830, 2831, 2832, 2834, 2835, 2836, 2838, 2839, 2840, 2841, 2842, 2844, 2845, 2846, 2847, 2848, 2849, 2850, 2852, 2853, 2854, 2855, 2856, 2858, 2859, 2860, 2862, 2863, 2864, 2865, 2866, 2867,

2868, 2869, 2870, 2871, 2872, 2873, 2874, 2875, 2876, 2877, 2878, 2880, 2881, 2882, 2883, 2884, 2885, 2886, 2888, 2889, 2890, 2891, 2892, 2893, 2894, 2895, 2896, 2898, 2899, 2900, 2901, 2902, 2904, 2905, 2906, 2907, 2908, 2910, 2911, 2912, 2913, 2914, 2915, 2916, 2918, 2919, 2920, 2921, 2922, 2923, 2924, 2925, 2926, 2928, 2929, 2930, 2931, 2932, 2933, 2934, 2935, 2936, 2937, 2938, 2940, 2941, 2942, 2943, 2944, 2945, 2946, 2947, 2948, 2949, 2950, 2951, 2952, 2954, 2955, 2956, 2958, 2959, 2960, 2961, 2962, 2964, 2965, 2966, 2967, 2968, 2970, 2972, 2973, 2974, 2975, 2976, 2977, 2978, 2979, 2980, 2981, 2982, 2983, 2984, 2985, 2986, 2987, 2988, 2989, 2990, 2991, 2992, 2993, 2994, 2995, 2996, 2997, 2998, 3000, 3002, 3003, 3004, 3005, 3006, 3007, 3008, 3009, 3010, 3012, 3013, 3014, 3015, 3016, 3017, 3018, 3020, 3021, 3022, 3024, 3025, 3026, 3027, 3028, 3029, 3030, 3031, 3032, 3033, 3034, 3035, 3036, 3038, 3039, 3040, 3042, 3043, 3044, 3045, 3046, 3047, 3048, 3050, 3051, 3052, 3053, 3054, 3055, 3056, 3057, 3058, 3059, 3060, 3062, 3063, 3064, 3065, 3066, 3068, 3069, 3070, 3071, 3072, 3073, 3074, 3075, 3076, 3077, 3078, 3080, 3081, 3082, 3084, 3085, 3086, 3087, 3088, 3090, 3091, 3092, 3093, 3094, 3095, 3096, 3097, 3098, 3099, 3100, 3101, 3102, 3103, 3104, 3105, 3106, 3107, 3108, 3110, 3111, 3112, 3113, 3114, 3115, 3116, 3117, 3118, 3120, 3122, 3123, 3124, 3125, 3126, 3127, 3128, 3129, 3130, 3131, 3132, 3133, 3134, 3135, 3136, 3138, 3139, 3140, 3141, 3142, 3143, 3144, 3145, 3146, 3147, 3148, 3149, 3150, 3151, 3152, 3153, 3154, 3155, 3156, 3157, 3158, 3159, 3160, 3161, 3162, 3164, 3165, 3166, 3168, 3170, 3171, 3172, 3173, 3174, 3175, 3176, 3177, 3178, 3179, 3180, 3182, 3183, 3184, 3185, 3186, 3188, 3189, 3190, 3192, 3193, 3194, 3195, 3196, 3197, 3198, 3199, 3200, 3201, 3202, 3204, 3205, 3206, 3207, 3208, 3210, 3211, 3212, 3213, 3214, 3215, 3216, 3218, 3219, 3220, 3222, 3223, 3224, 3225, 3226, 3227, 3228, 3230, 3231, 3232, 3233, 3234, 3235, 3236, 3237, 3238, 3239, 3240, 3241, 3242, 3243, 3244, 3245, 3246, 3247, 3248, 3249, 3250, 3252, 3254, 3255, 3256, 3258, 3260, 3261, 3262, 3263, 3264, 3265, 3266, 3267, 3268, 3269, 3270, 3272, 3273, 3274, 3275, 3276, 3277, 3278, 3279, 3280, 3281, 3282, 3283, 3284, 3285, 3286, 3287, 3288, 3289, 3290, 3291, 3292, 3293, 3294, 3295, 3296, 3297, 3298,

3300, 3302, 3303, 3304, 3305, 3306, 3308, 3309, 3310, 3311, 3312, 3314, 3315, 3316, 3317, 3318, 3320, 3321, 3322, 3324, 3325, 3326, 3327, 3328, 3330, 3332, 3333, 3334, 3335, 3336, 3337, 3338, 3339, 3340, 3341, 3342, 3344, 3345, 3346, 3348, 3349, 3350, 3351, 3352, 3353, 3354, 3355, 3356, 3357, 3358, 3360, 3362, 3363, 3364, 3365, 3366, 3367, 3368, 3369, 3370, 3372, 3374, 3375, 3376, 3377, 3378, 3379, 3380, 3381, 3382, 3383, 3384, 3385, 3386, 3387, 3388, 3390, 3392, 3393, 3394, 3395, 3396, 3397, 3398, 3399, 3400, 3401, 3402, 3403, 3404, 3405, 3406, 3408, 3409, 3410, 3411, 3412, 3414, 3415, 3416, 3417, 3418, 3419, 3420, 3421, 3422, 3423, 3424, 3425, 3426, 3427, 3428, 3429, 3430, 3431, 3432, 3434, 3435, 3436, 3437, 3438, 3439, 3440, 3441, 3442, 3443, 3444, 3445, 3446, 3447, 3448, 3450, 3451, 3452, 3453, 3454, 3455, 3456, 3458, 3459, 3460, 3462, 3464, 3465, 3466, 3468, 3470, 3471, 3472, 3473, 3474, 3475, 3476, 3477, 3478, 3479, 3480, 3481, 3482, 3483, 3484, 3485, 3486, 3487, 3488, 3489, 3490, 3492, 3493, 3494, 3495, 3496, 3497, 3498, 3500, 3501, 3502, 3503, 3504, 3505, 3506, 3507, 3508, 3509, 3510, 3512, 3513, 3514, 3515, 3516, 3518, 3519, 3520, 3521, 3522, 3523, 3524, 3525, 3526, 3528, 3530, 3531, 3532, 3534, 3535, 3536, 3537, 3538, 3540, 3542, 3543, 3544, 3545, 3546, 3548, 3549, 3550, 3551, 3552, 3553, 3554, 3555, 3556, 3558, 3560, 3561, 3562, 3563, 3564, 3565, 3566, 3567, 3568, 3569, 3570, 3572, 3573, 3574, 3575, 3576, 3577, 3578, 3579, 3580, 3582, 3584, 3585, 3586, 3587, 3588, 3589, 3590, 3591, 3592, 3594, 3595, 3596, 3597, 3598, 3599, 3600, 3601, 3602, 3603, 3604, 3605, 3606, 3608, 3609, 3610, 3611, 3612, 3614, 3615, 3616, 3618, 3619, 3620, 3621, 3622, 3624, 3625, 3626, 3627, 3628, 3629, 3630, 3632, 3633, 3634, 3635, 3636, 3638, 3639, 3640, 3641, 3642, 3644, 3645, 3646, 3647, 3648, 3649, 3650, 3651, 3652, 3653, 3654, 3655, 3656, 3657, 3658, 3660, 3661, 3662, 3663, 3664, 3665, 3666, 3667, 3668, 3669, 3670, 3672, 3674, 3675, 3676, 3678, 3679, 3680, 3681, 3682, 3683, 3684, 3685, 3686, 3687, 3688, 3689, 3690, 3692, 3693, 3694, 3695, 3696, 3698, 3699, 3700, 3702, 3703, 3704, 3705, 3706, 3707, 3708, 3710, 3711, 3712, 3713, 3714, 3715, 3716, 3717, 3718, 3720, 3721, 3722, 3723, 3724, 3725, 3726, 3728, 3729, 3730, 3731, 3732, 3734, 3735, 3736, 3737, 3738, 3740, 3741, 3742, 3743,

3744, 3745, 3746, 3747, 3748, 3749, 3750, 3751, 3752, 3753, 3754, 3755, 3756, 3757, 3758, 3759, 3760, 3762, 3763, 3764, 3765, 3766, 3768, 3770, 3771, 3772, 3773, 3774, 3775, 3776, 3777, 3778, 3780, 3781, 3782, 3783, 3784, 3785, 3786, 3787, 3788, 3789, 3790, 3791, 3792, 3794, 3795, 3796, 3798, 3799, 3800, 3801, 3802, 3804, 3805, 3806, 3807, 3808, 3809, 3810, 3811, 3812, 3813, 3814, 3815, 3816, 3817, 3818, 3819, 3820, 3822, 3824, 3825, 3826, 3827, 3828, 3829, 3830, 3831, 3832, 3834, 3835, 3836, 3837, 3838, 3839, 3840, 3841, 3842, 3843, 3844, 3845, 3846, 3848, 3849, 3850, 3852, 3854, 3855, 3856, 3857, 3858, 3859, 3860, 3861, 3862, 3864, 3865, 3866, 3867, 3868, 3869, 3870, 3871, 3872, 3873, 3874, 3875, 3876, 3878, 3879, 3880, 3882, 3883, 3884, 3885, 3886, 3887, 3888, 3890, 3891, 3892, 3893, 3894, 3895, 3896, 3897, 3898, 3899, 3900, 3901, 3902, 3903, 3904, 3905, 3906, 3908, 3909, 3910, 3912, 3913, 3914, 3915, 3916, 3918, 3920, 3921, 3922, 3924, 3925, 3926, 3927, 3928, 3930, 3932, 3933, 3934, 3935, 3936, 3937, 3938, 3939, 3940, 3941, 3942, 3944, 3945, 3946, 3948, 3949, 3950, 3951, 3952, 3953, 3954, 3955, 3956, 3957, 3958, 3959, 3960, 3961, 3962, 3963, 3964, 3965, 3966, 3968, 3969, 3970, 3971, 3972, 3973, 3974, 3975, 3976, 3977, 3978, 3979, 3980, 3981, 3982, 3983, 3984, 3985, 3986, 3987, 3988, 3990, 3991, 3992, 3993, 3994, 3995, 3996, 3997, 3998, 3999, 4000, 4002, 4004, 4005, 4006, 4008, 4009, 4010, 4011, 4012, 4014, 4015, 4016, 4017, 4018, 4020, 4022, 4023, 4024, 4025, 4026, 4028, 4029, 4030, 4031, 4032, 4033, 4034, 4035, 4036, 4037, 4038, 4039, 4040, 4041, 4042, 4043, 4044, 4045, 4046, 4047, 4048, 4050, 4052, 4053, 4054, 4055, 4056, 4058, 4059, 4060, 4061, 4062, 4063, 4064, 4065, 4066, 4067, 4068, 4069, 4070, 4071, 4072, 4074, 4075, 4076, 4077, 4078, 4080, 4081, 4082, 4083, 4084, 4085, 4086, 4087, 4088, 4089, 4090, 4092, 4094, 4095, 4096, 4097, 4098, 4100, 4101, 4102, 4103, 4104, 4105, 4106, 4107, 4108, 4109, 4110, 4112, 4113, 4114, 4115, 4116, 4117, 4118, 4119, 4120, 4121, 4122, 4123, 4124, 4125, 4126, 4128, 4130, 4131, 4132, 4134, 4135, 4136, 4137, 4138, 4140, 4141, 4142, 4143, 4144, 4145, 4146, 4147, 4148, 4149, 4150, 4151, 4152, 4154, 4155, 4156, 4158, 4160, 4161, 4162, 4163, 4164, 4165, 4166, 4167, 4168, 4169, 4170, 4171, 4172, 4173, 4174, 4175, 4176, 4178, 4179, 4180,

4181, 4182, 4183, 4184, 4185, 4186, 4187, 4188, 4189, 4190, 4191, 4192, 4193, 4194, 4195, 4196, 4197, 4198, 4199, 4200, 4202, 4203, 4204, 4205, 4206, 4207, 4208, 4209, 4210, 4212, 4213, 4214, 4215, 4216, 4218, 4220, 4221, 4222, 4223, 4224, 4225, 4226, 4227, 4228, 4230, 4232, 4233, 4234, 4235, 4236, 4237, 4238, 4239, 4240, 4242, 4244, 4245, 4246, 4247, 4248, 4249, 4250, 4251, 4252, 4254, 4255, 4256, 4257, 4258, 4260, 4262, 4263, 4264, 4265, 4266, 4267, 4268, 4269, 4270, 4272, 4274, 4275, 4276, 4277, 4278, 4279, 4280, 4281, 4282, 4284, 4285, 4286, 4287, 4288, 4290, 4291, 4292, 4293, 4294, 4295, 4296, 4298, 4299, 4300, 4301, 4302, 4303, 4304, 4305, 4306, 4307, 4308, 4309, 4310, 4311, 4312, 4313, 4314, 4315, 4316, 4317, 4318, 4319, 4320, 4321, 4322, 4323, 4324, 4325, 4326, 4328, 4329, 4330, 4331, 4332, 4333, 4334, 4335, 4336, 4338, 4340, 4341, 4342, 4343, 4344, 4345, 4346, 4347, 4348, 4350, 4351, 4352, 4353, 4354, 4355, 4356, 4358, 4359, 4360, 4361, 4362, 4364, 4365, 4366, 4367, 4368, 4369, 4370, 4371, 4372, 4374, 4375, 4376, 4377, 4378, 4379, 4380, 4381, 4382, 4383, 4384, 4385, 4386, 4387, 4388, 4389, 4390, 4392, 4393, 4394, 4395, 4396, 4398, 4399, 4400, 4401, 4402, 4403, 4404, 4405, 4406, 4407, 4408, 4410, 4411, 4412, 4413, 4414, 4415, 4416, 4417, 4418, 4419, 4420, 4422, 4424, 4425, 4426, 4427, 4428, 4429, 4430, 4431, 4432, 4433, 4434, 4435, 4436, 4437, 4438, 4439, 4440, 4442, 4443, 4444, 4445, 4446, 4448, 4449, 4450, 4452, 4453, 4454, 4455, 4456, 4458, 4459, 4460, 4461, 4462, 4464, 4465, 4466, 4467, 4468, 4469, 4470, 4471, 4472, 4473, 4474, 4475, 4476, 4477, 4478, 4479, 4480, 4482, 4484, 4485, 4486, 4487, 4488, 4489, 4490, 4491, 4492, 4494, 4495, 4496, 4497, 4498, 4499, 4500, 4501, 4502, 4503, 4504, 4505, 4506, 4508, 4509, 4510, 4511, 4512, 4514, 4515, 4516, 4518, 4520, 4521, 4522, 4524, 4525, 4526, 4527, 4528, 4529, 4530, 4531, 4532, 4533, 4534, 4535, 4536, 4537, 4538, 4539, 4540, 4541, 4542, 4543, 4544, 4545, 4546, 4548, 4550, 4551, 4552, 4553, 4554, 4555, 4556, 4557, 4558, 4559, 4560, 4562, 4563, 4564, 4565, 4566, 4568, 4569, 4570, 4571, 4572, 4573, 4574, 4575, 4576, 4577, 4578, 4579, 4580, 4581, 4582, 4584, 4585, 4586, 4587, 4588, 4589, 4590, 4592, 4593, 4594, 4595, 4596, 4598, 4599, 4600, 4601, 4602, 4604, 4605, 4606, 4607, 4608, 4609, 4610, 4611, 4612, 4613, 4614,

4615, 4616, 4617, 4618, 4619, 4620, 4622, 4623, 4624, 4625, 4626, 4627, 4628, 4629, 4630, 4631, 4632, 4633, 4634, 4635, 4636, 4638, 4640, 4641, 4642, 4644, 4645, 4646, 4647, 4648, 4650, 4652, 4653, 4654, 4655, 4656, 4658, 4659, 4660, 4661, 4662, 4664, 4665, 4666, 4667, 4668, 4669, 4670, 4671, 4672, 4674, 4675, 4676, 4677, 4678, 4680, 4681, 4682, 4683, 4684, 4685, 4686, 4687, 4688, 4689, 4690, 4692, 4693, 4694, 4695, 4696, 4697, 4698, 4699, 4700, 4701, 4702, 4704, 4705, 4706, 4707, 4708, 4709, 4710, 4711, 4712, 4713, 4714, 4715, 4716, 4717, 4718, 4719, 4720, 4722, 4724, 4725, 4726, 4727, 4728, 4730, 4731, 4732, 4734, 4735, 4736, 4737, 4738, 4739, 4740, 4741, 4742, 4743, 4744, 4745, 4746, 4747, 4748, 4749, 4750, 4752, 4753, 4754, 4755, 4756, 4757, 4758, 4760, 4761, 4762, 4763, 4764, 4765, 4766, 4767, 4768, 4769, 4770, 4771, 4772, 4773, 4774, 4775, 4776, 4777, 4778, 4779, 4780, 4781, 4782, 4784, 4785, 4786, 4788, 4790, 4791, 4792, 4794, 4795, 4796, 4797, 4798, 4800, 4802, 4803, 4804, 4805, 4806, 4807, 4808, 4809, 4810, 4811, 4812, 4814, 4815, 4816, 4818, 4819, 4820, 4821, 4822, 4823, 4824, 4825, 4826, 4827, 4828, 4829, 4830, 4832, 4833, 4834, 4835, 4836, 4837, 4838, 4839, 4840, 4841, 4842, 4843, 4844, 4845, 4846, 4847, 4848, 4849, 4850, 4851, 4852, 4853, 4854, 4855, 4856, 4857, 4858, 4859, 4860, 4862, 4863, 4864, 4865, 4866, 4867, 4868, 4869, 4870, 4872, 4873, 4874, 4875, 4876, 4878, 4879, 4880, 4881, 4882, 4883, 4884, 4885, 4886, 4887, 4888, 4890, 4891, 4892, 4893, 4894, 4895, 4896, 4897, 4898, 4899, 4900, 4901, 4902, 4904, 4905, 4906, 4907, 4908, 4910, 4911, 4912, 4913, 4914, 4915, 4916, 4917, 4918, 4920, 4921, 4922, 4923, 4924, 4925, 4926, 4927, 4928, 4929, 4930, 4932, 4934, 4935, 4936, 4938, 4939, 4940, 4941, 4942, 4944, 4945, 4946, 4947, 4948, 4949, 4950, 4952, 4953, 4954, 4955, 4956, 4958, 4959, 4960, 4961, 4962, 4963, 4964, 4965, 4966, 4968, 4970, 4971, 4972, 4974, 4975, 4976, 4977, 4978, 4979, 4980, 4981, 4982, 4983, 4984, 4985, 4986, 4988, 4989, 4990, 4991, 4992, 4994, 4995, 4996, 4997, 4998, 5000, 5001, 5002, 5004, 5005, 5006, 5007, 5008, 5010, 5012, 5013, 5014, 5015, 5016, 5017, 5018, 5019, 5020, 5022, 5024, 5025, 5026, 5027, 5028, 5029, 5030, 5031, 5032, 5033, 5034, 5035, 5036, 5037, 5038, 5040, 5041, 5042, 5043, 5044, 5045, 5046, 5047, 5048, 5049, 5050, 5052,

5053, 5054, 5055, 5056, 5057, 5058, 5060, 5061, 5062, 5063, 5064, 5065, 5066, 5067, 5068, 5069, 5070, 5071, 5072, 5073, 5074, 5075, 5076, 5078, 5079, 5080, 5082, 5083, 5084, 5085, 5086, 5088, 5089, 5090, 5091, 5092, 5093, 5094, 5095, 5096, 5097, 5098, 5100, 5102, 5103, 5104, 5105, 5106, 5108, 5109, 5110, 5111, 5112, 5114, 5115, 5116, 5117, 5118, 5120, 5121, 5122, 5123, 5124, 5125, 5126, 5127, 5128, 5129, 5130, 5131, 5132, 5133, 5134, 5135, 5136, 5137, 5138, 5139, 5140, 5141, 5142, 5143, 5144, 5145, 5146, 5148, 5149, 5150, 5151, 5152, 5154, 5155, 5156, 5157, 5158, 5159, 5160, 5161, 5162, 5163, 5164, 5165, 5166, 5168, 5169, 5170, 5172, 5173, 5174, 5175, 5176, 5177, 5178, 5180, 5181, 5182, 5183, 5184, 5185, 5186, 5187, 5188, 5190, 5191, 5192, 5193, 5194, 5195, 5196, 5198, 5199, 5200, 5201, 5202, 5203, 5204, 5205, 5206, 5207, 5208, 5210, 5211, 5212, 5213, 5214, 5215, 5216, 5217, 5218, 5219, 5220, 5221, 5222, 5223, 5224, 5225, 5226, 5228, 5229, 5230, 5232, 5234, 5235, 5236, 5238, 5239, 5240, 5241, 5242, 5243, 5244, 5245, 5246, 5247, 5248, 5249, 5250, 5251, 5252, 5253, 5254, 5255, 5256, 5257, 5258, 5259, 5260, 5262, 5263, 5264, 5265, 5266, 5267, 5268, 5269, 5270, 5271, 5272, 5274, 5275, 5276, 5277, 5278, 5280, 5282, 5283, 5284, 5285, 5286, 5287, 5288, 5289, 5290, 5291, 5292, 5293, 5294, 5295, 5296, 5298, 5299, 5300, 5301, 5302, 5304, 5305, 5306, 5307, 5308, 5310, 5311, 5312, 5313, 5314, 5315, 5316, 5317, 5318, 5319, 5320, 5321, 5322, 5324, 5325, 5326, 5327, 5328, 5329, 5330, 5331, 5332, 5334, 5335, 5336, 5337, 5338, 5339, 5340, 5341, 5342, 5343, 5344, 5345, 5346, 5348, 5349, 5350, 5352, 5353, 5354, 5355, 5356, 5357, 5358, 5359, 5360, 5361, 5362, 5363, 5364, 5365, 5366, 5367, 5368, 5369, 5370, 5371, 5372, 5373, 5374, 5375, 5376, 5377, 5378, 5379, 5380, 5382, 5383, 5384, 5385, 5386, 5388, 5389, 5390, 5391, 5392, 5394, 5395, 5396, 5397, 5398, 5400, 5401, 5402, 5403, 5404, 5405, 5406, 5408, 5409, 5410, 5411, 5412, 5414, 5415, 5416, 5418, 5420, 5421, 5422, 5423, 5424, 5425, 5426, 5427, 5428, 5429, 5430, 5432, 5433, 5434, 5435, 5436, 5438, 5439, 5440, 5442, 5444, 5445, 5446, 5447, 5448, 5450, 5451, 5452, 5453, 5454, 5455, 5456, 5457, 5458, 5459, 5460, 5461, 5462, 5463, 5464, 5465, 5466, 5467, 5468, 5469, 5470, 5472, 5473, 5474, 5475, 5476, 5478, 5480, 5481, 5482, 5484, 5485, 5486,

5487, 5488, 5489, 5490, 5491, 5492, 5493, 5494, 5495, 5496, 5497, 5498, 5499, 5500, 5502, 5504, 5505, 5506, 5508, 5509, 5510, 5511, 5512, 5513, 5514, 5515, 5516, 5517, 5518, 5520, 5522, 5523, 5524, 5525, 5526, 5528, 5529, 5530, 5532, 5533, 5534, 5535, 5536, 5537, 5538, 5539, 5540, 5541, 5542, 5543, 5544, 5545, 5546, 5547, 5548, 5549, 5550, 5551, 5552, 5553, 5554, 5555, 5556, 5558, 5559, 5560, 5561, 5562, 5564, 5565, 5566, 5567, 5568, 5570, 5571, 5572, 5574, 5575, 5576, 5577, 5578, 5579, 5580, 5582, 5583, 5584, 5585, 5586, 5587, 5588, 5589, 5590, 5592, 5593, 5594, 5595, 5596, 5597, 5598, 5599, 5600, 5601, 5602, 5603, 5604, 5605, 5606, 5607, 5608, 5609, 5610, 5611, 5612, 5613, 5614, 5615, 5616, 5617, 5618, 5619, 5620, 5621, 5622, 5624, 5625, 5626, 5627, 5628, 5629, 5630, 5631, 5632, 5633, 5634, 5635, 5636, 5637, 5638, 5640, 5642, 5643, 5644, 5645, 5646, 5648, 5649, 5650, 5652, 5654, 5655, 5656, 5658, 5660, 5661, 5662, 5663, 5664, 5665, 5666, 5667, 5668, 5670, 5671, 5672, 5673, 5674, 5675, 5676, 5677, 5678, 5679, 5680, 5681, 5682, 5684, 5685, 5686, 5687, 5688, 5690, 5691, 5692, 5694, 5695, 5696, 5697, 5698, 5699, 5700, 5702, 5703, 5704, 5705, 5706, 5707, 5708, 5709, 5710, 5712, 5713, 5714, 5715, 5716, 5718, 5719, 5720, 5721, 5722, 5723, 5724, 5725, 5726, 5727, 5728, 5729, 5730, 5731, 5732, 5733, 5734, 5735, 5736, 5738, 5739, 5740, 5742, 5744, 5745, 5746, 5747, 5748, 5750, 5751, 5752, 5753, 5754, 5755, 5756, 5757, 5758, 5759, 5760, 5761, 5762, 5763, 5764, 5765, 5766, 5767, 5768, 5769, 5770, 5771, 5772, 5773, 5774, 5775, 5776, 5777, 5778, 5780, 5781, 5782, 5784, 5785, 5786, 5787, 5788, 5789, 5790, 5792, 5793, 5794, 5795, 5796, 5797, 5798, 5799, 5800, 5802, 5803, 5804, 5805, 5806, 5808, 5809, 5810, 5811, 5812, 5814, 5815, 5816, 5817, 5818, 5819, 5820, 5822, 5823, 5824, 5825, 5826, 5828, 5829, 5830, 5831, 5832, 5833, 5834, 5835, 5836, 5837, 5838, 5840, 5841, 5842, 5844, 5845, 5846, 5847, 5848, 5850, 5852, 5853, 5854, 5855, 5856, 5858, 5859, 5860, 5862, 5863, 5864, 5865, 5866, 5868, 5870, 5871, 5872, 5873, 5874, 5875, 5876, 5877, 5878, 5880, 5882, 5883, 5884, 5885, 5886, 5887, 5888, 5889, 5890, 5891, 5892, 5893, 5894, 5895, 5896, 5898, 5899, 5900, 5901, 5902, 5904, 5905, 5906, 5907, 5908, 5909, 5910, 5911, 5912, 5913, 5914, 5915, 5916, 5917, 5918, 5919, 5920, 5921, 5922, 5924,

5925, 5926, 5928, 5929, 5930, 5931, 5932, 5933, 5934, 5935, 5936, 5937, 5938, 5940, 5941, 5942, 5943, 5944, 5945, 5946, 5947, 5948, 5949, 5950, 5951, 5952, 5954, 5955, 5956, 5957, 5958, 5959, 5960, 5961, 5962, 5963, 5964, 5965, 5966, 5967, 5968, 5969, 5970, 5971, 5972, 5973, 5974, 5975, 5976, 5977, 5978, 5979, 5980, 5982, 5983, 5984, 5985, 5986, 5988, 5989, 5990, 5991, 5992, 5993, 5994, 5995, 5996, 5997, 5998, 5999, 6000, 6001, 6002, 6003, 6004, 6005, 6006, 6008, 6009, 6010, 6012, 6013, 6014, 6015, 6016, 6017, 6018, 6019, 6020, 6021, 6022, 6023, 6024, 6025, 6026, 6027, 6028, 6030, 6031, 6032, 6033, 6034, 6035, 6036, 6038, 6039, 6040, 6041, 6042, 6044, 6045, 6046, 6048, 6049, 6050, 6051, 6052, 6054, 6055, 6056, 6057, 6058, 6059, 6060, 6061, 6062, 6063, 6064, 6065, 6066, 6068, 6069, 6070, 6071, 6072, 6074, 6075, 6076, 6077, 6078, 6080, 6081, 6082, 6083, 6084, 6085, 6086, 6087, 6088, 6090, 6092, 6093, 6094, 6095, 6096, 6097, 6098, 6099, 6100, 6102, 6103, 6104, 6105, 6106, 6107, 6108, 6109, 6110, 6111, 6112, 6114, 6115, 6116, 6117, 6118, 6119, 6120, 6122, 6123, 6124, 6125, 6126, 6127, 6128, 6129, 6130, 6132, 6134, 6135, 6136, 6137, 6138, 6139, 6140, 6141, 6142, 6144, 6145, 6146, 6147, 6148, 6149, 6150, 6152, 6153, 6154, 6155, 6156, 6157, 6158, 6159, 6160, 6161, 6162, 6164, 6165, 6166, 6167, 6168, 6169, 6170, 6171, 6172, 6174, 6175, 6176, 6177, 6178, 6179, 6180, 6181, 6182, 6183, 6184, 6185, 6186, 6187, 6188, 6189, 6190, 6191, 6192, 6193, 6194, 6195, 6196, 6198, 6200, 6201, 6202, 6204, 6205, 6206, 6207, 6208, 6209, 6210, 6212, 6213, 6214, 6215, 6216, 6218, 6219, 6220, 6222, 6223, 6224, 6225, 6226, 6227, 6228, 6230, 6231, 6232, 6233, 6234, 6235, 6236, 6237, 6238, 6239, 6240, 6241, 6242, 6243, 6244, 6245, 6246, 6248, 6249, 6250, 6251, 6252, 6253, 6254, 6255, 6256, 6258, 6259, 6260, 6261, 6262, 6264, 6265, 6266, 6267, 6268, 6270, 6272, 6273, 6274, 6275, 6276, 6278, 6279, 6280, 6281, 6282, 6283, 6284, 6285, 6286, 6288, 6289, 6290, 6291, 6292, 6293, 6294, 6295, 6296, 6297, 6298, 6300, 6302, 6303, 6304, 6305, 6306, 6307, 6308, 6309, 6310, 6312, 6313, 6314, 6315, 6316, 6318, 6319, 6320, 6321, 6322, 6324, 6325, 6326, 6327, 6328, 6330, 6331, 6332, 6333, 6334, 6335, 6336, 6338, 6339, 6340, 6341, 6342, 6344, 6345, 6346, 6347, 6348, 6349, 6350, 6351, 6352, 6354, 6355, 6356, 6357, 6358,

6360, 6362, 6363, 6364, 6365, 6366, 6368, 6369, 6370, 6371, 6372, 6374, 6375, 6376, 6377, 6378, 6380, 6381, 6382, 6383, 6384, 6385, 6386, 6387, 6388, 6390, 6391, 6392, 6393, 6394, 6395, 6396, 6398, 6399, 6400, 6401, 6402, 6403, 6404, 6405, 6406, 6407, 6408, 6409, 6410, 6411, 6412, 6413, 6414, 6415, 6416, 6417, 6418, 6419, 6420, 6422, 6423, 6424, 6425, 6426, 6428, 6429, 6430, 6431, 6432, 6433, 6434, 6435, 6436, 6437, 6438, 6439, 6440, 6441, 6442, 6443, 6444, 6445, 6446, 6447, 6448, 6450, 6452, 6453, 6454, 6455, 6456, 6457, 6458, 6459, 6460, 6461, 6462, 6463, 6464, 6465, 6466, 6467, 6468, 6470, 6471, 6472, 6474, 6475, 6476, 6477, 6478, 6479, 6480, 6482, 6483, 6484, 6485, 6486, 6487, 6488, 6489, 6490, 6492, 6493, 6494, 6495, 6496, 6497, 6498, 6499, 6500, 6501, 6502, 6503, 6504, 6505, 6506, 6507, 6508, 6509, 6510, 6511, 6512, 6513, 6514, 6515, 6516, 6517, 6518, 6519, 6520, 6522, 6523, 6524, 6525, 6526, 6527, 6528, 6530, 6531, 6532, 6533, 6534, 6535, 6536, 6537, 6538, 6539, 6540, 6541, 6542, 6543, 6544, 6545, 6546, 6548, 6549, 6550, 6552, 6554, 6555, 6556, 6557, 6558, 6559, 6560, 6561, 6562, 6564, 6565, 6566, 6567, 6568, 6570, 6572, 6573, 6574, 6575, 6576, 6578, 6579, 6580, 6582, 6583, 6584, 6585, 6586, 6587, 6588, 6589, 6590, 6591, 6592, 6593, 6594, 6595, 6596, 6597, 6598, 6600, 6601, 6602, 6603, 6604, 6605, 6606, 6608, 6609, 6610, 6611, 6612, 6613, 6614, 6615, 6616, 6617, 6618, 6620, 6621, 6622, 6623, 6624, 6625, 6626, 6627, 6628, 6629, 6630, 6631, 6632, 6633, 6634, 6635, 6636, 6638, 6639, 6640, 6641, 6642, 6643, 6644, 6645, 6646, 6647, 6648, 6649, 6650, 6651, 6652, 6654, 6655, 6656, 6657, 6658, 6660, 6662, 6663, 6664, 6665, 6666, 6667, 6668, 6669, 6670, 6671, 6672, 6674, 6675, 6676, 6677, 6678, 6680, 6681, 6682, 6683, 6684, 6685, 6686, 6687, 6688, 6690, 6692, 6693, 6694, 6695, 6696, 6697, 6698, 6699, 6700, 6702, 6704, 6705, 6706, 6707, 6708, 6710, 6711, 6712, 6713, 6714, 6715, 6716, 6717, 6718, 6720, 6721, 6722, 6723, 6724, 6725, 6726, 6727, 6728, 6729, 6730, 6731, 6732, 6734, 6735, 6736, 6738, 6739, 6740, 6741, 6742, 6743, 6744, 6745, 6746, 6747, 6748, 6749, 6750, 6751, 6752, 6753, 6754, 6755, 6756, 6757, 6758, 6759, 6760, 6762, 6764, 6765, 6766, 6767, 6768, 6769, 6770, 6771, 6772, 6773, 6774, 6775, 6776, 6777, 6778, 6780, 6782, 6783, 6784, 6785, 6786, 6787, 6788, 6789,

6790, 6792, 6794, 6795, 6796, 6797, 6798, 6799, 6800, 6801, 6802, 6804, 6805, 6806, 6807, 6808, 6809, 6810, 6811, 6812, 6813, 6814, 6815, 6816, 6817, 6818, 6819, 6820, 6821, 6822, 6824, 6825, 6826, 6828, 6830, 6831, 6832, 6834, 6835, 6836, 6837, 6838, 6839, 6840, 6842, 6843, 6844, 6845, 6846, 6847, 6848, 6849, 6850, 6851, 6852, 6853, 6854, 6855, 6856, 6858, 6859, 6860, 6861, 6862, 6864, 6865, 6866, 6867, 6868, 6870, 6872, 6873, 6874, 6875, 6876, 6877, 6878, 6879, 6880, 6881, 6882, 6884, 6885, 6886, 6887, 6888, 6889, 6890, 6891, 6892, 6893, 6894, 6895, 6896, 6897, 6898, 6900, 6901, 6902, 6903, 6904, 6905, 6906, 6908, 6909, 6910, 6912, 6913, 6914, 6915, 6916, 6918, 6919, 6920, 6921, 6922, 6923, 6924, 6925, 6926, 6927, 6928, 6929, 6930, 6931, 6932, 6933, 6934, 6935, 6936, 6937, 6938, 6939, 6940, 6941, 6942, 6943, 6944, 6945, 6946, 6948, 6950, 6951, 6952, 6953, 6954, 6955, 6956, 6957, 6958, 6960, 6962, 6963, 6964, 6965, 6966, 6968, 6969, 6970, 6972, 6973, 6974, 6975, 6976, 6978, 6979, 6980, 6981, 6982, 6984, 6985, 6986, 6987, 6988, 6989, 6990, 6992, 6993, 6994, 6995, 6996, 6998, 6999, 7000, 7002, 7003, 7004, 7005, 7006, 7007, 7008, 7009, 7010, 7011, 7012, 7014, 7015, 7016, 7017, 7018, 7020, 7021, 7022, 7023, 7024, 7025, 7026, 7028, 7029, 7030, 7031, 7032, 7033, 7034, 7035, 7036, 7037, 7038, 7040, 7041, 7042, 7044, 7045, 7046, 7047, 7048, 7049, 7050, 7051, 7052, 7053, 7054, 7055, 7056, 7058, 7059, 7060, 7061, 7062, 7063, 7064, 7065, 7066, 7067, 7068, 7070, 7071, 7072, 7073, 7074, 7075, 7076, 7077, 7078, 7080, 7081, 7082, 7083, 7084, 7085, 7086, 7087, 7088, 7089, 7090, 7091, 7092, 7093, 7094, 7095, 7096, 7097, 7098, 7099, 7100, 7101, 7102, 7104, 7105, 7106, 7107, 7108, 7110, 7111, 7112, 7113, 7114, 7115, 7116, 7117, 7118, 7119, 7120, 7122, 7123, 7124, 7125, 7126, 7128, 7130, 7131, 7132, 7133, 7134, 7135, 7136, 7137, 7138, 7139, 7140, 7141, 7142, 7143, 7144, 7145, 7146, 7147, 7148, 7149, 7150, 7152, 7153, 7154, 7155, 7156, 7157, 7158, 7160, 7161, 7162, 7163, 7164, 7165, 7166, 7167, 7168, 7169, 7170, 7171, 7172, 7173, 7174, 7175, 7176, 7178, 7179, 7180, 7181, 7182, 7183, 7184, 7185, 7186, 7188, 7189, 7190, 7191, 7192, 7194, 7195, 7196, 7197, 7198, 7199, 7200, 7201, 7202, 7203, 7204, 7205, 7206, 7208, 7209, 7210, 7212, 7214, 7215, 7216, 7217, 7218, 7220, 7221, 7222, 7223, 7224,

7225, 7226, 7227, 7228, 7230, 7231, 7232, 7233, 7234, 7235, 7236, 7238, 7239, 7240, 7241, 7242, 7244, 7245, 7246, 7248, 7249, 7250, 7251, 7252, 7254, 7255, 7256, 7257, 7258, 7259, 7260, 7261, 7262, 7263, 7264, 7265, 7266, 7267, 7268, 7269, 7270, 7271, 7272, 7273, 7274, 7275, 7276, 7277, 7278, 7279, 7280, 7281, 7282, 7284, 7285, 7286, 7287, 7288, 7289, 7290, 7291, 7292, 7293, 7294, 7295, 7296, 7298, 7299, 7300, 7301, 7302, 7303, 7304, 7305, 7306, 7308, 7310, 7311, 7312, 7313, 7314, 7315, 7316, 7317, 7318, 7319, 7320, 7322, 7323, 7324, 7325, 7326, 7327, 7328, 7329, 7330, 7332, 7334, 7335, 7336, 7337, 7338, 7339, 7340, 7341, 7342, 7343, 7344, 7345, 7346, 7347, 7348, 7350, 7352, 7353, 7354, 7355, 7356, 7357, 7358, 7359, 7360, 7361, 7362, 7363, 7364, 7365, 7366, 7367, 7368, 7370, 7371, 7372, 7373, 7374, 7375, 7376, 7377, 7378, 7379, 7380, 7381, 7382, 7383, 7384, 7385, 7386, 7387, 7388, 7389, 7390, 7391, 7392, 7394, 7395, 7396, 7397, 7398, 7399, 7400, 7401, 7402, 7403, 7404, 7405, 7406, 7407, 7408, 7409, 7410, 7412, 7413, 7414, 7415, 7416, 7418, 7419, 7420, 7421, 7422, 7423, 7424, 7425, 7426, 7427, 7428, 7429, 7430, 7431, 7432, 7434, 7435, 7436, 7437, 7438, 7439, 7440, 7441, 7442, 7443, 7444, 7445, 7446, 7447, 7448, 7449, 7450, 7452, 7453, 7454, 7455, 7456, 7458, 7460, 7461, 7462, 7463, 7464, 7465, 7466, 7467, 7468, 7469, 7470, 7471, 7472, 7473, 7474, 7475, 7476, 7478, 7479, 7480, 7482, 7483, 7484, 7485, 7486, 7488, 7490, 7491, 7492, 7493, 7494, 7495, 7496, 7497, 7498, 7500, 7501, 7502, 7503, 7504, 7505, 7506, 7508, 7509, 7510, 7511, 7512, 7513, 7514, 7515, 7516, 7518, 7519, 7520, 7521, 7522, 7524, 7525, 7526, 7527, 7528, 7530, 7531, 7532, 7533, 7534, 7535, 7536, 7538, 7539, 7540, 7542, 7543, 7544, 7545, 7546, 7548, 7550, 7551, 7552, 7553, 7554, 7555, 7556, 7557, 7558, 7560, 7562, 7563, 7564, 7565, 7566, 7567, 7568, 7569, 7570, 7571, 7572, 7574, 7575, 7576, 7578, 7579, 7580, 7581, 7582, 7584, 7585, 7586, 7587, 7588, 7590, 7592, 7593, 7594, 7595, 7596, 7597, 7598, 7599, 7600, 7601, 7602, 7604, 7605, 7606, 7608, 7609, 7610, 7611, 7612, 7613, 7614, 7615, 7616, 7617, 7618, 7619, 7620, 7622, 7623, 7624, 7625, 7626, 7627, 7628, 7629, 7630, 7631, 7632, 7633, 7634, 7635, 7636, 7637, 7638, 7640, 7641, 7642, 7644, 7645, 7646, 7647, 7648, 7650, 7651, 7652, 7653, 7654, 7655, 7656, 7657,

7658, 7659, 7660, 7661, 7662, 7663, 7664, 7665, 7666, 7667, 7668, 7670, 7671, 7672, 7674, 7675, 7676, 7677, 7678, 7679, 7680, 7682, 7683, 7684, 7685, 7686, 7688, 7689, 7690, 7692, 7693, 7694, 7695, 7696, 7697, 7698, 7700, 7701, 7702, 7704, 7705, 7706, 7707, 7708, 7709, 7710, 7711, 7712, 7713, 7714, 7715, 7716, 7718, 7719, 7720, 7721, 7722, 7724, 7725, 7726, 7728, 7729, 7730, 7731, 7732, 7733, 7734, 7735, 7736, 7737, 7738, 7739, 7740, 7742, 7743, 7744, 7745, 7746, 7747, 7748, 7749, 7750, 7751, 7752, 7754, 7755, 7756, 7758, 7760, 7761, 7762, 7763, 7764, 7765, 7766, 7767, 7768, 7769, 7770, 7771, 7772, 7773, 7774, 7775, 7776, 7777, 7778, 7779, 7780, 7781, 7782, 7783, 7784, 7785, 7786, 7787, 7788, 7790, 7791, 7792, 7794, 7795, 7796, 7797, 7798, 7799, 7800, 7801, 7802, 7803, 7804, 7805, 7806, 7807, 7808, 7809, 7810, 7811, 7812, 7813, 7814, 7815, 7816, 7818, 7819, 7820, 7821, 7822, 7824, 7825, 7826, 7827, 7828, 7830, 7831, 7832, 7833, 7834, 7835, 7836, 7837, 7838, 7839, 7840, 7842, 7843, 7844, 7845, 7846, 7847, 7848, 7849, 7850, 7851, 7852, 7854, 7855, 7856, 7857, 7858, 7859, 7860, 7861, 7862, 7863, 7864, 7865, 7866, 7868, 7869, 7870, 7871, 7872, 7874, 7875, 7876, 7878, 7880, 7881, 7882, 7884, 7885, 7886, 7887, 7888, 7889, 7890, 7891, 7892, 7893, 7894, 7895, 7896, 7897, 7898, 7899, 7900, 7902, 7903, 7904, 7905, 7906, 7908, 7909, 7910, 7911, 7912, 7913, 7914, 7915, 7916, 7917, 7918, 7920, 7921, 7922, 7923, 7924, 7925, 7926, 7928, 7929, 7930, 7931, 7932, 7934, 7935, 7936, 7938, 7939, 7940, 7941, 7942, 7943, 7944, 7945, 7946, 7947, 7948, 7950, 7952, 7953, 7954, 7955, 7956, 7957, 7958, 7959, 7960, 7961, 7962, 7964, 7965, 7966, 7967, 7968, 7969, 7970, 7971, 7972, 7973, 7974, 7975, 7976, 7977, 7978, 7979, 7980, 7981, 7982, 7983, 7984, 7985, 7986, 7987, 7988, 7989, 7990, 7991, 7992, 7994, 7995, 7996, 7997, 7998, 7999, 8000, 8001, 8002, 8003, 8004, 8005, 8006, 8007, 8008, 8010, 8012, 8013, 8014, 8015, 8016, 8018, 8019, 8020, 8021, 8022, 8023, 8024, 8025, 8026, 8027, 8028, 8029, 8030, 8031, 8032, 8033, 8034, 8035, 8036, 8037, 8038, 8040, 8041, 8042, 8043, 8044, 8045, 8046, 8047, 8048, 8049, 8050, 8051, 8052, 8054, 8055, 8056, 8057, 8058, 8060, 8061, 8062, 8063, 8064, 8065, 8066, 8067, 8068, 8070, 8071, 8072, 8073, 8074, 8075, 8076, 8077, 8078, 8079, 8080, 8082, 8083, 8084, 8085, 8086,

8088, 8090, 8091, 8092, 8094, 8095, 8096, 8097, 8098, 8099, 8100, 8102, 8103, 8104, 8105, 8106, 8107, 8108, 8109, 8110, 8112, 8113, 8114, 8115, 8116, 8118, 8119, 8120, 8121, 8122, 8124, 8125, 8126, 8127, 8128, 8129, 8130, 8131, 8132, 8133, 8134, 8135, 8136, 8137, 8138, 8139, 8140, 8141, 8142, 8143, 8144, 8145, 8146, 8148, 8149, 8150, 8151, 8152, 8153, 8154, 8155, 8156, 8157, 8158, 8159, 8160, 8162, 8163, 8164, 8165, 8166, 8168, 8169, 8170, 8172, 8173, 8174, 8175, 8176, 8177, 8178, 8180, 8181, 8182, 8183, 8184, 8185, 8186, 8187, 8188, 8189, 8190, 8192, 8193, 8194, 8195, 8196, 8197, 8198, 8199, 8200, 8201, 8202, 8203, 8204, 8205, 8206, 8207, 8208, 8210, 8211, 8212, 8213, 8214, 8215, 8216, 8217, 8218, 8220, 8222, 8223, 8224, 8225, 8226, 8227, 8228, 8229, 8230, 8232, 8234, 8235, 8236, 8238, 8239, 8240, 8241, 8242, 8244, 8245, 8246, 8247, 8248, 8249, 8250, 8251, 8252, 8253, 8254, 8255, 8256, 8257, 8258, 8259, 8260, 8261, 8262, 8264, 8265, 8266, 8267, 8268, 8270, 8271, 8272, 8274, 8275, 8276, 8277, 8278, 8279, 8280, 8281, 8282, 8283, 8284, 8285, 8286, 8288, 8289, 8290, 8292, 8294, 8295, 8296, 8298, 8299, 8300, 8301, 8302, 8303, 8304, 8305, 8306, 8307, 8308, 8309, 8310, 8312, 8313, 8314, 8315, 8316, 8318, 8319, 8320, 8321, 8322, 8323, 8324, 8325, 8326, 8327, 8328, 8330, 8331, 8332, 8333, 8334, 8335, 8336, 8337, 8338, 8339, 8340, 8341, 8342, 8343, 8344, 8345, 8346, 8347, 8348, 8349, 8350, 8351, 8352, 8354, 8355, 8356, 8357, 8358, 8359, 8360, 8361, 8362, 8364, 8365, 8366, 8367, 8368, 8370, 8371, 8372, 8373, 8374, 8375, 8376, 8378, 8379, 8380, 8381, 8382, 8383, 8384, 8385, 8386, 8388, 8390, 8391, 8392, 8393, 8394, 8395, 8396, 8397, 8398, 8399, 8400, 8401, 8402, 8403, 8404, 8405, 8406, 8407, 8408, 8409, 8410, 8411, 8412, 8413, 8414, 8415, 8416, 8417, 8418, 8420, 8421, 8422, 8424, 8425, 8426, 8427, 8428, 8430, 8432, 8433, 8434, 8435, 8436, 8437, 8438, 8439, 8440, 8441, 8442, 8444, 8445, 8446, 8448, 8449, 8450, 8451, 8452, 8453, 8454, 8455, 8456, 8457, 8458, 8459, 8460, 8462, 8463, 8464, 8465, 8466, 8468, 8469, 8470, 8471, 8472, 8473, 8474, 8475, 8476, 8477, 8478, 8479, 8480, 8481, 8482, 8483, 8484, 8485, 8486, 8487, 8488, 8489, 8490, 8491, 8492, 8493, 8494, 8495, 8496, 8497, 8498, 8499, 8500, 8502, 8503, 8504, 8505, 8506, 8507, 8508, 8509, 8510, 8511, 8512, 8514, 8515, 8516, 8517,

8518, 8519, 8520, 8522, 8523, 8524, 8525, 8526, 8528, 8529, 8530, 8531, 8532, 8533, 8534, 8535, 8536, 8538, 8540, 8541, 8542, 8544, 8545, 8546, 8547, 8548, 8549, 8550, 8551, 8552, 8553, 8554, 8555, 8556, 8557, 8558, 8559, 8560, 8561, 8562, 8564, 8565, 8566, 8567, 8568, 8569, 8570, 8571, 8572, 8574, 8575, 8576, 8577, 8578, 8579, 8580, 8582, 8583, 8584, 8585, 8586, 8587, 8588, 8589, 8590, 8591, 8592, 8593, 8594, 8595, 8596, 8598, 8600, 8601, 8602, 8603, 8604, 8605, 8606, 8607, 8608, 8610, 8611, 8612, 8613, 8614, 8615, 8616, 8617, 8618, 8619, 8620, 8621, 8622, 8624, 8625, 8626, 8628, 8630, 8631, 8632, 8633, 8634, 8635, 8636, 8637, 8638, 8639, 8640, 8642, 8643, 8644, 8645, 8646, 8648, 8649, 8650, 8651, 8652, 8653, 8654, 8655, 8656, 8657, 8658, 8659, 8660, 8661, 8662, 8664, 8665, 8666, 8667, 8668, 8670, 8671, 8672, 8673, 8674, 8675, 8676, 8678, 8679, 8680, 8682, 8683, 8684, 8685, 8686, 8687, 8688, 8690, 8691, 8692, 8694, 8695, 8696, 8697, 8698, 8700, 8701, 8702, 8703, 8704, 8705, 8706, 8708, 8709, 8710, 8711, 8712, 8714, 8715, 8716, 8717, 8718, 8720, 8721, 8722, 8723, 8724, 8725, 8726, 8727, 8728, 8729, 8730, 8732, 8733, 8734, 8735, 8736, 8738, 8739, 8740, 8742, 8743, 8744, 8745, 8746, 8748, 8749, 8750, 8751, 8752, 8754, 8755, 8756, 8757, 8758, 8759, 8760, 8762, 8763, 8764, 8765, 8766, 8767, 8768, 8769, 8770, 8771, 8772, 8773, 8774, 8775, 8776, 8777, 8778, 8780, 8781, 8782, 8784, 8785, 8786, 8787, 8788, 8789, 8790, 8791, 8792, 8793, 8794, 8795, 8796, 8797, 8798, 8799, 8800, 8801, 8802, 8804, 8805, 8806, 8808, 8809, 8810, 8811, 8812, 8813, 8814, 8815, 8816, 8817, 8818, 8820, 8822, 8823, 8824, 8825, 8826, 8827, 8828, 8829, 8830, 8832, 8833, 8834, 8835, 8836, 8838, 8840, 8841, 8842, 8843, 8844, 8845, 8846, 8847, 8848, 8850, 8851, 8852, 8853, 8854, 8855, 8856, 8857, 8858, 8859, 8860, 8862, 8864, 8865, 8866, 8868, 8869, 8870, 8871, 8872, 8873, 8874, 8875, 8876, 8877, 8878, 8879, 8880, 8881, 8882, 8883, 8884, 8885, 8886, 8888, 8889, 8890, 8891, 8892, 8894, 8895, 8896, 8897, 8898, 8899, 8900, 8901, 8902, 8903, 8904, 8905, 8906, 8907, 8908, 8909, 8910, 8911, 8912, 8913, 8914, 8915, 8916, 8917, 8918, 8919, 8920, 8921, 8922, 8924, 8925, 8926, 8927, 8928, 8930, 8931, 8932, 8934, 8935, 8936, 8937, 8938, 8939, 8940, 8942, 8943, 8944, 8945, 8946, 8947, 8948, 8949, 8950, 8952, 8953, 8954,

8955, 8956, 8957, 8958, 8959, 8960, 8961, 8962, 8964, 8965, 8966, 8967, 8968, 8970, 8972, 8973, 8974, 8975, 8976, 8977, 8978, 8979, 8980, 8981, 8982, 8983, 8984, 8985, 8986, 8987, 8988, 8989, 8990, 8991, 8992, 8993, 8994, 8995, 8996, 8997, 8998, 9000, 9002, 9003, 9004, 9005, 9006, 9008, 9009, 9010, 9012, 9014, 9015, 9016, 9017, 9018, 9019, 9020, 9021, 9022, 9023, 9024, 9025, 9026, 9027, 9028, 9030, 9031, 9032, 9033, 9034, 9035, 9036, 9037, 9038, 9039, 9040, 9042, 9044, 9045, 9046, 9047, 9048, 9050, 9051, 9052, 9053, 9054, 9055, 9056, 9057, 9058, 9060, 9061, 9062, 9063, 9064, 9065, 9066, 9068, 9069, 9070, 9071, 9072, 9073, 9074, 9075, 9076, 9077, 9078, 9079, 9080, 9081, 9082, 9083, 9084, 9085, 9086, 9087, 9088, 9089, 9090, 9092, 9093, 9094, 9095, 9096, 9097, 9098, 9099, 9100, 9101, 9102, 9104, 9105, 9106, 9107, 9108, 9110, 9111, 9112, 9113, 9114, 9115, 9116, 9117, 9118, 9119, 9120, 9121, 9122, 9123, 9124, 9125, 9126, 9128, 9129, 9130, 9131, 9132, 9134, 9135, 9136, 9138, 9139, 9140, 9141, 9142, 9143, 9144, 9145, 9146, 9147, 9148, 9149, 9150, 9152, 9153, 9154, 9155, 9156, 9158, 9159, 9160, 9162, 9163, 9164, 9165, 9166, 9167, 9168, 9169, 9170, 9171, 9172, 9174, 9175, 9176, 9177, 9178, 9179, 9180, 9182, 9183, 9184, 9185, 9186, 9188, 9189, 9190, 9191, 9192, 9193, 9194, 9195, 9196, 9197, 9198, 9200, 9201, 9202, 9204, 9205, 9206, 9207, 9208, 9210, 9211, 9212, 9213, 9214, 9215, 9216, 9217, 9218, 9219, 9220, 9222, 9223, 9224, 9225, 9226, 9228, 9229, 9230, 9231, 9232, 9233, 9234, 9235, 9236, 9237, 9238, 9240, 9242, 9243, 9244, 9245, 9246, 9247, 9248, 9249, 9250, 9251, 9252, 9253, 9254, 9255, 9256, 9258, 9259, 9260, 9261, 9262, 9263, 9264, 9265, 9266, 9267, 9268, 9269, 9270, 9271, 9272, 9273, 9274, 9275, 9276, 9278, 9279, 9280, 9282, 9284, 9285, 9286, 9287, 9288, 9289, 9290, 9291, 9292, 9294, 9295, 9296, 9297, 9298, 9299, 9300, 9301, 9302, 9303, 9304, 9305, 9306, 9307, 9308, 9309, 9310, 9312, 9313, 9314, 9315, 9316, 9317, 9318, 9320, 9321, 9322, 9324, 9325, 9326, 9327, 9328, 9329, 9330, 9331, 9332, 9333, 9334, 9335, 9336, 9338, 9339, 9340, 9342, 9344, 9345, 9346, 9347, 9348, 9350, 9351, 9352, 9353, 9354, 9355, 9356, 9357, 9358, 9359, 9360, 9361, 9362, 9363, 9364, 9365, 9366, 9367, 9368, 9369, 9370, 9372, 9373, 9374, 9375, 9376, 9378, 9379, 9380, 9381, 9382, 9383, 9384, 9385, 9386,

9387, 9388, 9389, 9390, 9392, 9393, 9394, 9395, 9396, 9398, 9399, 9400, 9401, 9402, 9404, 9405, 9406, 9407, 9408, 9409, 9410, 9411, 9412, 9414, 9415, 9416, 9417, 9418, 9420, 9422, 9423, 9424, 9425, 9426, 9427, 9428, 9429, 9430, 9432, 9434, 9435, 9436, 9438, 9440, 9441, 9442, 9443, 9444, 9445, 9446, 9447, 9448, 9449, 9450, 9451, 9452, 9453, 9454, 9455, 9456, 9457, 9458, 9459, 9460, 9462, 9464, 9465, 9466, 9468, 9469, 9470, 9471, 9472, 9474, 9475, 9476, 9477, 9478, 9480, 9481, 9482, 9483, 9484, 9485, 9486, 9487, 9488, 9489, 9490, 9492, 9493, 9494, 9495, 9496, 9498, 9499, 9500, 9501, 9502, 9503, 9504, 9505, 9506, 9507, 9508, 9509, 9510, 9512, 9513, 9514, 9515, 9516, 9517, 9518, 9519, 9520, 9522, 9523, 9524, 9525, 9526, 9527, 9528, 9529, 9530, 9531, 9532, 9534, 9535, 9536, 9537, 9538, 9540, 9541, 9542, 9543, 9544, 9545, 9546, 9548, 9549, 9550, 9552, 9553, 9554, 9555, 9556, 9557, 9558, 9559, 9560, 9561, 9562, 9563, 9564, 9565, 9566, 9567, 9568, 9569, 9570, 9571, 9572, 9573, 9574, 9575, 9576, 9577, 9578, 9579, 9580, 9581, 9582, 9583, 9584, 9585, 9586, 9588, 9589, 9590, 9591, 9592, 9593, 9594, 9595, 9596, 9597, 9598, 9599, 9600, 9602, 9603, 9604, 9605, 9606, 9607, 9608, 9609, 9610, 9611, 9612, 9614, 9615, 9616, 9617, 9618, 9620, 9621, 9622, 9624, 9625, 9626, 9627, 9628, 9630, 9632, 9633, 9634, 9635, 9636, 9637, 9638, 9639, 9640, 9641, 9642, 9644, 9645, 9646, 9647, 9648, 9650, 9651, 9652, 9653, 9654, 9655, 9656, 9657, 9658, 9659, 9660, 9662, 9663, 9664, 9665, 9666, 9667, 9668, 9669, 9670, 9671, 9672, 9673, 9674, 9675, 9676, 9678, 9680, 9681, 9682, 9683, 9684, 9685, 9686, 9687, 9688, 9690, 9691, 9692, 9693, 9694, 9695, 9696, 9698, 9699, 9700, 9701, 9702, 9703, 9704, 9705, 9706, 9707, 9708, 9709, 9710, 9711, 9712, 9713, 9714, 9715, 9716, 9717, 9718, 9720, 9722, 9723, 9724, 9725, 9726, 9727, 9728, 9729, 9730, 9731, 9732, 9734, 9735, 9736, 9737, 9738, 9740, 9741, 9742, 9744, 9745, 9746, 9747, 9748, 9750, 9751, 9752, 9753, 9754, 9755, 9756, 9757, 9758, 9759, 9760, 9761, 9762, 9763, 9764, 9765, 9766, 9768, 9770, 9771, 9772, 9773, 9774, 9775, 9776, 9777, 9778, 9779, 9780, 9782, 9783, 9784, 9785, 9786, 9788, 9789, 9790, 9792, 9793, 9794, 9795, 9796, 9797, 9798, 9799, 9800, 9801, 9802, 9804, 9805, 9806, 9807, 9808, 9809, 9810, 9812, 9813, 9814, 9815, 9816, 9818, 9819, 9820, 9821, 9822,

9823, 9824, 9825, 9826, 9827, 9828, 9830, 9831, 9832, 9834, 9835, 9836, 9837, 9838, 9840, 9841, 9842, 9843, 9844, 9845, 9846, 9847, 9848, 9849, 9850, 9852, 9853, 9854, 9855, 9856, 9858, 9860, 9861, 9862, 9863, 9864, 9865, 9866, 9867, 9868, 9869, 9870, 9872, 9873, 9874, 9875, 9876, 9877, 9878, 9879, 9880, 9881, 9882, 9884, 9885, 9886, 9888, 9889, 9890, 9891, 9892, 9893, 9894, 9895, 9896, 9897, 9898, 9899, 9900, 9902, 9903, 9904, 9905, 9906, 9908, 9909, 9910, 9911, 9912, 9913, 9914, 9915, 9916, 9917, 9918, 9919, 9920, 9921, 9922, 9924, 9925, 9926, 9927, 9928, 9930, 9932, 9933, 9934, 9935, 9936, 9937, 9938, 9939, 9940, 9942, 9943, 9944, 9945, 9946, 9947, 9948, 9950, 9951, 9952, 9953, 9954, 9955, 9956, 9957, 9958, 9959, 9960, 9961, 9962, 9963, 9964, 9965, 9966, 9968, 9969, 9970, 9971, 9972, 9974, 9975, 9976, 9977, 9978, 9979, 9980, 9981, 9982, 9983, 9984, 9985, 9986, 9987, 9988, 9989, 9990, 9991, 9992, 9993, 9994, 9995, 9996, 9997, 9998, 9999, 10000.

(https://www.wikiprimes.com/composite-number-list/)

BINARY & DECIMAL SYSTEM

Why and how 9752 in binary system is

128th place	***64th place***	***32nd place***	***16th place***	***8th place***	***4th place***	***2nd place***	***1st place***
0	**0**	**0**	**0**	**9**	**7**	**5**	**2**
10^7	10^6	10^5	10^4	$9 \cdot 1000 = 9000$ 10^3	$7 \cdot 100 = 700$ 10^2	$5 \cdot 10 = 50$ 10^1	$2 \cdot 1 = 2$ 10^0
So the result is 9000 + 700 + 50 + 2 = 9 7 5 2							

Why and how 78 in binary system is

128th place	***64th place***	***32nd place***	***16th place***	***8th place***	***4th place***	***2nd place***	***1st place***
0	**0**	**0**	**0**	**0**	**0**	**7**	**8**
10^7	10^6	10^5	10^4	10^3	10^2	$7 \cdot 10 = 70$ 10^1	$8 \cdot 1 = 8$ 10^0
So the result is 70 + 8 = 7 8							

Quick Example 1 Converting binary number to decimal number

What is the decimal number for 0 0 0 0 1 1 1 0 ?

128^{th} place	***64^{th} place***	***32^{nd} place***	***16^{th} place***	***8^{th} place***	***4^{th} place***	***2^{nd} place***	***1^{st} place***
0	**0**	**0**	**0**	**1**	**1**	**1**	**0**
$128 \cdot 0 = 0$ 2^7	$64 \cdot 0 = 0$ 2^6	$32 \cdot 0 = 0$ 2^5	$16 \cdot 0 = 0$ 2^4	$8 \cdot 1 = 8$ 2^3	$4 \cdot 1 = 4$ 2^2	$2 \cdot 1 = 2$ 2^1	$1 \cdot 0 = 0$ 2^0
So the result is 0 + 0 + 0 + 0 + 8 + 4 + 2 + 0 = 14							

Quick Example 2

What is the decimal number for 1 0 0 0 1 1 0 1 ?

128^{th} place	***64^{th} place***	***32^{nd} place***	***16^{th} place***	***8^{th} place***	***4^{th} place***	***2^{nd} place***	***1^{st} place***
1	**0**	**0**	**0**	**1**	**1**	**0**	**1**
$1 \cdot 128 = 128$ 2^7	$0 \cdot 64 = 0$ 2^6	$0 \cdot 32 = 0$ 2^5	$0 \cdot 16 = 0$ 2^4	$1 \cdot 8 = 8$ 2^3	$1 \cdot 4 = 4$ 2^2	$2 \cdot 0 = 0$ 2^1	$1 \cdot 1 = 1$ 2^0
So the result is 128 + 0 + 0 + 0 + 8 + 4 + 0 + 1 = 1 4 1							

Quick Example 3 To convert decimal to binary

What is binary number for the decimals 7?

128th place	***64th place***	***32nd place***	***16th place***	***8th place***	***4th place***	***2nd place***	***1st place***
0	0	0	0	**0**	**1**	**1**	**1**
128 > 7	64 > 7	32 > 7	16 > 7	8 > 7	4 < 7, 7 – 4 = 3	2 < 3, 3 – 2 = 1	1 = 1

If the binary value place is greater than decimal value, we must put 0. Because 128 > 7 so we put 0 on the 128^{th} place, 64 > 7 so we put 0 on the 64^{th} place, 32 > 7 so we put 0 on the 32^{nd} place, 16 > 7, 8 > 7 so we put 0 on the 8^{th} place, but because 4 < 7 so we put 1 on the 4^{th} place. Then 7 – 4 = 3 so now 2 < 3 so we put 1 on the 2^{nd} place. Then 3 - 2 = 1 and 1 is equal to 1 on the 1^{st} place so now we put 1 on the 1^{st} place.

Quick Example 4

What is binary number for the decimals 8?

128th place	***64th place***	***32nd place***	***16th place***	***8th place***	***4th place***	***2nd place***	***1st place***
0	0	0	0	**1**	**0**	**0**	**0**
128 > 8	64 > 8	32 > 8	16 > 8	8 = 8, 8 – 8 = 0	4 > 0	2 > 0	1 > 0

If the binary value place is greater than decimal value, we must put 0. Because 128 > 8, 64 > 8, 32 > 8, 16 > 8, but because 8 = 8 so we put 1 on the 8^{th} place. Now 8 – 8 = 0 so 4 > 0 so we put 0 on the 4^{th} place, 2 > 0 so we put 0 on the 2^{nd} place, and lastly 1 > 0 so we put 0 on the 1^{st} place.

Quick Example 5

What is binary number for the decimals 27?

128th place	***64th place***	***32nd place***	***16th place***	***8th place***	***4th place***	***2nd place***	***1st place***
0	0	0	**1**	**1**	**0**	**1**	**1**
128 > 27	64 > 27	32 > 27	16 < 27, 27 – 16 = 11	8 < 11, 11 – 8 = 3	4 > 3	2 < 3	1 < 3

If the binary value place is greater than decimal value, we must put 0. 128 > 27, 64 > 27, 32 > 27, but because 16 < 27 so we put 1 on the 16^{th} place. Then for the 8^{th} place is 27 – 16 = 11 so now 8 < 11 so we put 1 on the 8^{th} place. Then for the 4^{th} place is 11 – 8 = 3 so we put 0 on the 4^{th} place because 4 > 3, and 2 < 3, and lastly 1 < 3 so we put 1 on the 1^{st} place.

ASCII, decimal, hexadecimal, octal, and binary conversion table
(https://www.ibm.com/support/knowledgecenter/ssw_aix_72/network/conversion_table.html)

Helpful information for converting ASCII, decimal, hexadecimal, octal, and binary values can be referenced in this table.

ASCII	Decimal	Hexadecimal	Octal	Binary
null	0	0	0	0
start of header	1	1	1	1
start of text	2	2	2	10
end of text	3	3	3	11
end of transmission	4	4	4	100
enquire	5	5	5	101
acknowledge	6	6	6	110
bell	7	7	7	111
backspace	8	8	10	1000
horizontal tab	9	9	11	1001
linefeed	10	A	12	1010
vertical tab	11	B	13	1011
form feed	12	C	14	1100
carriage return	13	D	15	1101
shift out	14	E	16	1110
shift in	15	F	17	1111
data link escape	16	10	20	10000
device control 1/Xon	17	11	21	10001
device control 2	18	12	22	10010
device control 3/Xoff	19	13	23	10011
device control 4	20	14	24	10100
negative acknowledge	21	15	25	10101

ASCII	Decimal	Hexadecimal	Octal	Binary
synchronous idle	22	16	26	10110
end of transmission block	23	17	27	10111
cancel	24	18	30	11000
end of medium	25	19	31	11001
end of file/ substitute	26	1A	32	11010
escape	27	1B	33	11011
file separator	28	1C	34	11100
group separator	29	1D	35	11101
record separator	30	1E	36	11110
unit separator	31	1F	37	11111
space	32	20	40	100000
!	33	21	41	100001
"	34	22	42	100010
#	35	23	43	100011
$	36	24	44	100100
%	37	25	45	100101
&	38	26	46	100110
'	39	27	47	100111
(	40	28	50	101000
)	41	29	51	101001
*	42	2A	52	101010
+	43	2B	53	101011
,	44	2C	54	101100
-	45	2D	55	101101
.	46	2E	56	101110
/	47	2F	57	101111

ASCII	Decimal	Hexadecimal	Octal	Binary
0	48	30	60	110000
1	49	31	61	110001
2	50	32	62	110010
3	51	33	63	110011
4	52	34	64	110100
5	53	35	65	110101
6	54	36	66	110110
7	55	37	67	110111
8	56	38	70	111000
9	57	39	71	111001
:	58	3A	72	111010
;	59	3B	73	111011
<	60	3C	74	111100
=	61	3D	75	111101
>	62	3E	76	111110
?	63	3F	77	111111
@	64	40	100	1000000
A	65	41	101	1000001
B	66	42	102	1000010
C	67	43	103	1000011
D	68	44	104	1000100
E	69	45	105	1000101
F	70	46	106	1000110
G	71	47	107	1000111
H	72	48	110	1001000
I	73	49	111	1001001
J	74	4A	112	1001010

ASCII	Decimal	Hexadecimal	Octal	Binary
K	75	4B	113	1001011
L	76	4C	114	1001100
M	77	4D	115	1001101
N	78	4E	116	1001110
O	79	4F	117	1001111
P	80	50	120	1010000
Q	81	51	121	1010001
R	82	52	122	1010010
S	83	53	123	1010011
T	84	54	124	1010100
U	85	55	125	1010101
V	86	56	126	1010110
W	87	57	127	1010111
X	88	58	130	1011000
Y	89	59	131	1011001
Z	90	5A	132	1011010
[	91	5B	133	1011011
\	92	5C	134	1011100
]	93	5D	135	1011101
^	94	5E	136	1011110
_	95	5F	137	1011111
`	96	60	140	1100000
a	97	61	141	1100001
b	98	62	142	1100010
c	99	63	143	1100011
d	100	64	144	1100100
e	101	65	145	1100101

ASCII	Decimal	Hexadecimal	Octal	Binary
f	102	66	146	1100110
g	103	67	147	1100111
h	104	68	150	1101000
i	105	69	151	1101001
j	106	6A	152	1101010
k	107	6B	153	1101011
l	108	6C	154	1101100
m	109	6D	155	1101101
n	110	6E	156	1101110
o	111	6F	157	1101111
p	112	70	160	1110000
q	113	71	161	1110001
r	114	72	162	1110010
s	115	73	163	1110011
t	116	74	164	1110100
u	117	75	165	1110101
v	118	76	166	1110110
w	119	77	167	1110111
x	120	78	170	1111000
y	121	79	171	1111001
z	122	7A	172	1111010
{	123	7B	173	1111011
|	124	7C	174	1111100
}	125	7D	175	1111101
~	126	7E	176	1111110
DEL	127	7F	177	1111111
	128	80	200	10000000

ASCII	Decimal	Hexadecimal	Octal	Binary
	129	81	201	10000001
	130	82	202	10000010
	131	83	203	10000011
	132	84	204	10000100
	133	85	205	10000101
	134	86	206	10000110
	135	87	207	10000111
	136	88	210	10001000
	137	89	211	10001001
	138	8A	212	10001010
	139	8B	213	10001011
	140	8C	214	10001100
	141	8D	215	10001101
	142	8E	216	10001110
	143	8F	217	10001111
	144	90	220	10010000
	145	91	221	10010001
	146	92	222	10010010
	147	93	223	10010011
	148	94	224	10010100
	149	95	225	10010101
	150	96	226	10010110
	151	97	227	10010111
	152	98	230	10011000
	153	99	231	10011001
	154	9A	232	10011010
	155	9B	233	10011011

ASCII	Decimal	Hexadecimal	Octal	Binary
	156	9C	234	10011100
	157	9D	235	10011101
	158	9E	236	10011110
	159	9F	237	10011111
	160	A0	240	10100000
	161	A1	241	10100001
	162	A2	242	10100010
	163	A3	243	10100011
	164	A4	244	10100100
	165	A5	245	10100101
	166	A6	246	10100110
	167	A7	247	10100111
	168	A8	250	10101000
	169	A9	251	10101001
	170	AA	252	10101010
	171	AB	253	10101011
	172	AC	254	10101100
	173	AD	255	10101101
	174	AE	256	10101110
	175	AF	257	10101111
	176	B0	260	10110000
	177	B1	261	10110001
	178	B2	262	10110010
	179	B3	263	10110011
	180	B4	264	10110100
	181	B5	265	10110101
	182	B6	266	10110110

ASCII	Decimal	Hexadecimal	Octal	Binary
	183	B7	267	10110111
	184	B8	270	10111000
	185	B9	271	10111001
	186	BA	272	10111010
	187	BB	273	10111011
	188	BC	274	10111100
	189	BD	275	10111101
	190	BE	276	10111110
	191	BF	277	10111111
	192	C0	300	11000000
	193	C1	301	11000001
	194	C2	302	11000010
	195	C3	303	11000011
	196	C4	304	11000100
	197	C5	305	11000101
	198	C6	306	11000110
	199	C7	307	11000111
	200	C8	310	11001000
	201	C9	311	11001001
	202	CA	312	11001010
	203	CB	313	11001011
	204	CC	314	11001100
	205	CD	315	11001101
	206	CE	316	11001110
	207	CF	317	11001111
	208	D0	320	11010000
	209	D1	321	11010001

ASCII	Decimal	Hexadecimal	Octal	Binary
	210	D2	322	11010010
	211	D3	323	11010011
	212	D4	324	11010100
	213	D5	325	11010101
	214	D6	326	11010110
	215	D7	327	11010111
	216	D8	330	11011000
	217	D9	331	11011001
	218	DA	332	11011010
	219	DB	333	11011011
	220	DC	334	11011100
	221	DD	335	11011101
	222	DE	336	11011110
	223	DF	337	11011111
	224	E0	340	11100000
	225	E1	341	11100001
	226	E2	342	11100010
	227	E3	343	11100011
	228	E4	344	11100100
	229	E5	345	11100101
	230	E6	346	11100110
	231	E7	347	11100111
	232	E8	350	11101000
	233	E9	351	11101001
	234	EA	352	11101010
	235	EB	353	11101011
	236	EC	354	11101100

ASCII	Decimal	Hexadecimal	Octal	Binary
	237	ED	355	11101101
	238	EE	356	11101110
	239	EF	357	11101111
	240	F0	360	11110000
	241	F1	361	11110001
	242	F2	362	11110010
	243	F3	363	11110011
	244	F4	364	11110100
	245	F5	365	11110101
	246	F6	366	11110110
	247	F7	367	11110111
	248	F8	370	11111000
	249	F9	371	11111001
	250	FA	372	11111010
	251	FB	373	11111011
	252	FC	374	11111100
	253	FD	375	11111101
	254	FE	376	11111110
	255	FF	377	11111111

Top 5 Cybersecurity Facts, Figures, Predictions, And Statistics for 2020 To 2021

(https://cybersecurityventures.com/top-5-cybersecurity-facts-figures-predictions-and-statistics-for-2019-to-2021/)

***What you need to know about the trillion dollar cyber economy over the next 2 years – Steve Morgan, Editor-in-Chief* - Sausalito, Calif. – Mar. 29, 2020**

Cybercrime Magazine extrapolates the top 5 market data points from our research in order to summarize the cybersecurity industry through 2021.

Although the numbers listed below have been featured and quoted (with attribution to us as the source) by hundreds of major media outlets, vendors, academia, governments, associations, event producers, and industry experts — the material is all original research which first appeared in reports published by Cybersecurity Ventures.

1. Cybercrime damage costs are predicted to hit $6 trillion annually by 2021.

Cyber crime damages will cost the world $6 trillion annually by 2021, up from $3 trillion in 2015.

This represents the greatest transfer of economic wealth in history, risks the incentives for innovation and investment, and will be more profitable than the global trade of all major illegal drugs combined.

Cybercrime costs include damage and destruction of data, stolen money, lost productivity, theft of intellectual property, theft of personal and financial data, embezzlement, fraud, post-attack disruption to the normal course of business, forensic investigation, restoration and deletion of hacked data and systems, and reputational harm.

2. Cybersecurity spending will exceed $1 trillion from 2017 to 2021.

The cybersecurity market is continuing its stratospheric growth and hurtling towards the trillion dollar mark that we originally predicted in 2017.

In 2004, the global cybersecurity market was worth $3.5 billion — and in 2017 it was expected to be worth more than $120 billion. The cybersecurity market grew by roughly 35X over 13 years entering our most recent prediction cycle.

Global spending on cybersecurity products and services are predicted to exceed $1 trillion (cumulatively) over five years, from 2017 to 2021.

3. The world will have 3.5 million unfilled cybersecurity jobs by the end of 2021.

Every IT position is also a cybersecurity position now. Every IT worker, every technology worker, needs to be involved with protecting and defending apps, data, devices, infrastructure and people.

There will be 3.5 million unfilled cybersecurity jobs by 2021 — enough to fill 50 NFL stadiums — according to Cybersecurity Ventures. This is up from Cisco's previous estimation of 1 million cybersecurity openings in 2014.

The cybersecurity unemployment rate is at zero percent in 2019, where it's been since 2011.

4. Ransomware damage costs are predicted to grow more than 57X from 2015 to 2021.

Global ransomware damage costs are predicted to reach $20 billion by 2021, up from $325 million in 2015.

Ransomware attacks on healthcare organizations — often called the No. 1 cyber-attacked industry — will quadruple by 2020.

Cybersecurity Ventures expects that a business will fall victim to a ransomware attack every 11 seconds by 2021, up from every 14 seconds in 2019. This makes ransomware the fastest growing type of cybercrime.

5. 70 percent of cryptocurrency transactions will be for illegal activity by 2021

Cryptocrime is an emerging segment of the cybercrime ecosystem, and it's booming.

Around $76 billion of illegal activity per year involves bitcoin, which is close to the scale of the U.S. and European markets for illegal drugs, according to a study published by the University of Sydney in Australia, ranked as one of the top 100 universities globally.

Cybersecurity Ventures predicts that by 2021 more than 70 percent of all cryptocurrency transactions annually will be for illegal activity, up from current estimates ranging anywhere from 20 percent (of the 5 major cryptocurrencies) to nearly 50 percent (of bitcoin).

Stay tuned for a year-end update with more cybersecurity market research from the editors at Cybersecurity Ventures.

– *Steve Morgan is founder and Editor-in-Chief at Cybersecurity Ventures.*

Alarming Cybersecurity Stats:
What You Need to Know for 2021

(https://www.forbes.com/sites/chuckbrooks/2021/03/02/alarming-cybersecurity-stats-------what-you-need-to-know-for-2021/?sh=6698ebb58d3d)

The year 2020 broke all records when it came to data lost in breaches and sheer numbers of cyber-attacks on companies, government, and individuals. In addition, the sophistication of threats increased from the application of emerging technologies such as machine learning, artificial intelligence, and 5G, and especially from greater tactical cooperation among hacker groups and state actors. The recent Solar Winds attack, among others, highlighted both the threat and sophistication of those realities.

The following informational links are compiled from recent statistics pulled from a variety of articles and blogs. As we head deeper into 2021, it is worth exploring these statistics and their potential cybersecurity implications in our changing digital landscape.

To make the information more useable, I have broken down the cybersecurity statistics in several categories, including Top Resources for Cybersecurity Stats, The State of Cybersecurity Readiness, Types of Cyber-threats, The Economics of Cybersecurity, and Data at Risk.

There are many other categories of cybersecurity that do need a deeper dive, including perspectives on The Cloud, Internet of Things, Open Source, Deep Fakes, the lack of qualified Cyber workers, and stats on many other types of cyber-attacks. The resources below help cover those various categories.

The State of Cybersecurity Readiness:

Despite all the warnings and high-profile breaches, that state of readiness for most when it comes to cybersecurity is dismal. The need for better cyber-hygiene is evident from using stronger passwords,

patching software, employing multi-factor authentication and many other important security steps. The reality is reflected in the stats below.

78% Lack Confidence in Their Company's Cybersecurity Posture 78% Lack Confidence in Their Company's Cybersecurity Posture, Prompting 91% to Increase 2021 Budgets (yahoo.com)

On average, only 5% of companies' folders are properly protected. 2019 Global Data Risk Report | Varonis

Cyber Attacks More Likely to Bring Down F-35 Jets Than Missiles "In our ever-increasing digitalized world of cybersecurity, threats keep growing.Take the F-35 fighter jet, for instance. It's been called the "flying computer" thanks to its myriad new contraptions that include AI-like sensor fusion, 360-degree camera views, improved data links, a database of threat information at-the-ready, and a highly advanced computerized logistics systems." Cyber Attacks More Likely to Bring Down an F-35 Than Missiles | IE (interestingengineering.com)

Nearly 80% of senior IT and IT security leaders believe their organizations lack sufficient protection against cyberattacks despite increased IT security investments made in 2020 to deal with distributed IT and work-from-home challenges, according to a new IDG Research Services survey commissioned by Insight Enterprises "Just 57% conducted a data security risk assessment in 2020." 78% Lack Confidence in Their Company's Cybersecurity Posture, Prompting 91% to Increase 2021 Budgets (yahoo.com)

Data breaches have lasting financial effects on hospitals, report suggests "More than 90 percent of all healthcare organizations reported at least one security breach in the last three years. Data breaches have lasting financial effects on hospitals, report suggests (beckershospitalreview.com)

Identity theft spikes amid pandemic "The US Federal Trade Commission received 1.4 million reports of identity theft last year, double the number from 2019" Identity theft spikes amid pandemic | WeLiveSecurity

The Economics of Cybersecurity "Cost of breaches have been consistently rising in the last few years. The new vulnerabilities that emerged from shifting to a remote workforce greatly expanded the cyber-attack surface and added many vulnerabilities for hackers to exploit from home offices. Also, automated attacks by hackers and the ability to convert cryptocurrencies via ransomware has added to the cost of cybercrime."

Cybercrime To Cost The World $10.5 Trillion Annually By 2025 Cybercrime To Cost The World $10.5 Trillion Annually By 2025 (cybersecurityventures.com)

Evil Internet Minute 2019 "Every minute, $2,900,000 is lost to cybercrime and top companies pay $25 per minute due to cyber security breaches" The Evil Internet Minute 2019 | RiskIQ

The average cost of a data breach is $3.86 million as of 2020 Data Breach Costs: Calculating the Losses for Security and IT Pros (dice.com)

Cybersecurity Market Forecasted To Be Worth $403B by 2027 "Over a 5-year period, the cybersecurity market is forecasted to experience a compound annual growth rate (CAGR) of 12.5%. Cybersecurity Market Forecasted To Be Worth $403B by 2027 - CE Pro

Types of Cyber-Threats:

Phishing still ranks as a "go to" by most hackers because it is easy to do and it often works. The malware just keeps on coming...

Malware increased by 358% in 2020 "A research study conducted by Deep Instinct reports on the hundreds of millions of attempted

cyberattacks that occurred every day throughout 2020 showing malware increased by 358% overall and ransomware increased by 435% as compared with 2019."Malware increased by 358% in 2020 - Help Net Security

Check Point Software´s Security Report Reveals Extent of Global Cyber Pandemic, and Shows How Organizations Can Develop Immunity in 2021 "The world faces over 100,000 malicious websites and 10,000 malicious files daily. 87% of organizations have experienced an attempted exploit of an already-known, existing vulnerability**"** Check Point Software´s Security Report Reveals Extent of Global Cyber Pandemic, and Shows How Organizations Can Develop Immunity in 2021 Nasdaq:CHKP (globenewswire.com)

Phishing attacks account for more than 80% of reported security incidents. Top cybersecurity facts, figures and statistics | CSO Online

Google has registered 2,145,013 phishing sites as of Jan 17, 2021. "This is up from 1,690,000 on Jan 19, 2020 (up 27% over 12 months)". Phishing Statistics (Updated 2021) | 50+ Important Phishing Stats | Tessian

Ransomware Victim Every 10 Seconds in 2020 One Ransomware Victim Every 10 Seconds in 2020 - Infosecurity Magazine (infosecurity-magazine.com)

Terrifying Statistics: 1 in 5 Americans Victim of Ransomware "According to data gathered by Anomali and The Harris Poll, ransomware attacks 1 in 5 Americans. The survey was based on responses from more than 2,000 American citizens**."** Terrifying Statistics: 1 in 5 Americans Victim of Ransomware (sensorstechforum.com)

Attackers disrupting COVID-19 efforts and critical supply chains "Cyberattacks evolved in 2020 as threat actors sought to profit from the unprecedented socioeconomic, business and political challenges

brought on by the COVID-19 pandemic, IBM Security reveals." Attackers disrupting COVID-19 efforts and critical supply chains - Help Net Security **Cybercriminals are quick to find ways to get around strengthened security "next gen" supply chain attacks grew 420%** in just 12 months" State of the Software Supply Chain 2020 Report | Download (sonatype.com)

Ransomware, Phishing Will Remain Primary Risks in 2021 "Attackers have doubled down on ransomware and phishing — with some tweaks — while deepfakes and disinformation will become more major threats in the future, according to a trio of threat reports." Ransomware, Phishing Will Remain Primary Risks in 2021 (darkreading.com)

Netscout Threat Intelligence saw 4.83 million DDoS attacks in 1H 2020*. "*This is roughly 26,000 attacks a day or 18 attacks per minute." NETSCOUT Threat Intelligence Report Findings from 1H 2020

Dragos: ICS security threats grew threefold in 2020 "A new report highlights the challenges facing ICS vendors today, including practices that are geared toward traditional IT and not designed for ICS security." Dragos: ICS security threats grew threefold in 2020 (techtarget.com)

12-top-cybersecurity-threats-against-organisations-2019-statistics-e1556643214683.jpg (980×585) (comparitech.com)

The Data at Risk:

Cybercrime To Cost The World $10.5 Trillion Annually By 2025 "The world will store 200 zettabytes of data by 2025, according to Cybersecurity Ventures. This includes data stored on private and public IT infrastructures, on utility infrastructures, on private and public cloud data centers, on personal computing devices — PCs, laptops, tablets, and smartphones — and on IoT (Internet-of-Things) devices." Cybercrime To Cost The World $10.5 Trillion Annually By 2025 (cybersecurityventures.com)

The number of Internet connected devices is expected to increase from 31 billion in 2020 to 35 billion in 2021 and 75 billion in 2025. Security Today's The IoT Rundown for 2020

Cybersecurity statistics do have a heuristic value in that they can point to gaps, growing threats, and alert to trends. The challenge is adapting the data into a functional and agile risk management strategy to be able to better protect ourselves. The alarming cybersecurity statistics for 2021 are a call to take the risk management mission more seriously.

Readings/Sources

NIST (https://www.nist.gov/, https://www.nist.gov/cryptography, https://www.nist.gov/blockchain)

Prof. Dan Boneh's Presentation in 2020
about Blockchain & Primitive Cryptography

The Best Blockchain Programming Languages to Learn in 2021
(https://python.plainenglish.io/the-best-blockchain-programming-languages-to-learn-in-2021-c3c66307512c)

Network Security Essentials, Application and Standards
by William Stallings

Cryptography and Network Security Principles and Practice,
Global Edition by William Stallings

Mathematical Foundations for Cryptography Lectures
by William Bahn, B.Sc. from University of Colorado, USA

Crypto Book (https://cryptobook.nakov.com)

Math in Network Security (https://www.doc.ic.ac.uk/)

23 blockchain languages driving the future of programming
(https://techbeacon.com/app-dev-testing/23-blockchain-languages-driving-future-programming)

Simply Explained: Why is Proof of Work Required in Bitcoin?
(https://medium.com/coinmonks/simply-explained-why-is-proof-of-work-required-in-bitcoin-611b143fc3e0)

Ahli Kriptografi CIA Pun Tak Bisa Memecahkan Kode Sandi Indonesia (https://www.kompasiana.com/cahsaren/55007c46a3331123705111d7/ahli-kriptografi-cia-pun-tak-bisa-memecahkan-kode-sandi-indonesia)

Kryptos, Kode Tak Terpecahkan di Markas CIA (https://www.cnnindonesia.com/internasional/20200726144652-134-529069/kryptos-kode-tak-terpecahkan-di-markas-cia)

Mengerikan, 7 Badan Intelijen Ini Konon Memiliki Mata-Mata Terbaik di Dunia (https://www.liputan6.com/global/read/3873990/mengerikan-7-badan-intelijen-ini-konon-memiliki-mata-mata-terbaik-di-dunia)

10 Naskah dan Kode Paling Misterius di Dunia (https://sains.kompas.com/read/2013/07/29/2324124/10.Naskah.dan.Kode.Paling.Misterius.di.Dunia?page=all)

Indonesia Disebut dalam Skandal Alat Komunikasi Rahasia yang Libatkan CIA (https://www.suara.com/tekno/2020/02/13/073500/indonesia-disebut-dalam-skandal-alat-komunikasi-rahasia-yang-libatkan-cia?page=all)

Soal Alat Komunikasi Rahasia yang Disadap CIA, Ini Kata Kemenlu (https://www.suara.com/tekno/2020/02/13/200347/soal-alat-komunikasi-rahasia-yang-disadap-cia-ini-kata-kemenlu?page=all)

BSSN Akui Masih Gunakan Alat Komunikasi Crypto AG yang Disadap CIA (https://www.suara.com/tekno/2020/02/13/151205/bssn-akui-masih-gunakan-alat-komunikasi-crypto-ag-yang-disadap-cia)

Sepenting Apa Belajar soal Blockchain? Cek Nih! (https://www.cnbcindonesia.com/tech/20210119182452-37-217253/sepenting-apa-belajar-soal-blockchain-cek-nih)

Education

Computer Science & Introduction to Artificial Intelligence with Python | Harvard University Online Study, USA

Cryptography 1 | Stanford University Online Study, USA

Introduction to Mathematical Thinking | Stanford University Online Study, USA

Cryptography | Colorado University – Colorado Springs Online Study, USA

Introduction to Cybersecurity Tools & Cyber Attacks | IBM

Filmmaking & Film Directing | New York Film Academy, Manhattan, USA

Cinematography Course, Indonesia

Photography Canon Course, Indonesia

Foreign Language Academy, Indonesia

Seventy High School, Indonesia

Book's Cover

digital-world-map-<a href='https- www.freepik.com photos background'>Background photo created by natanaelginting - www.freepik.com< a>

BLANK PAGE

BLANK PAGE

BLANK PAGE

BLANK PAGE

BLANK PAGE