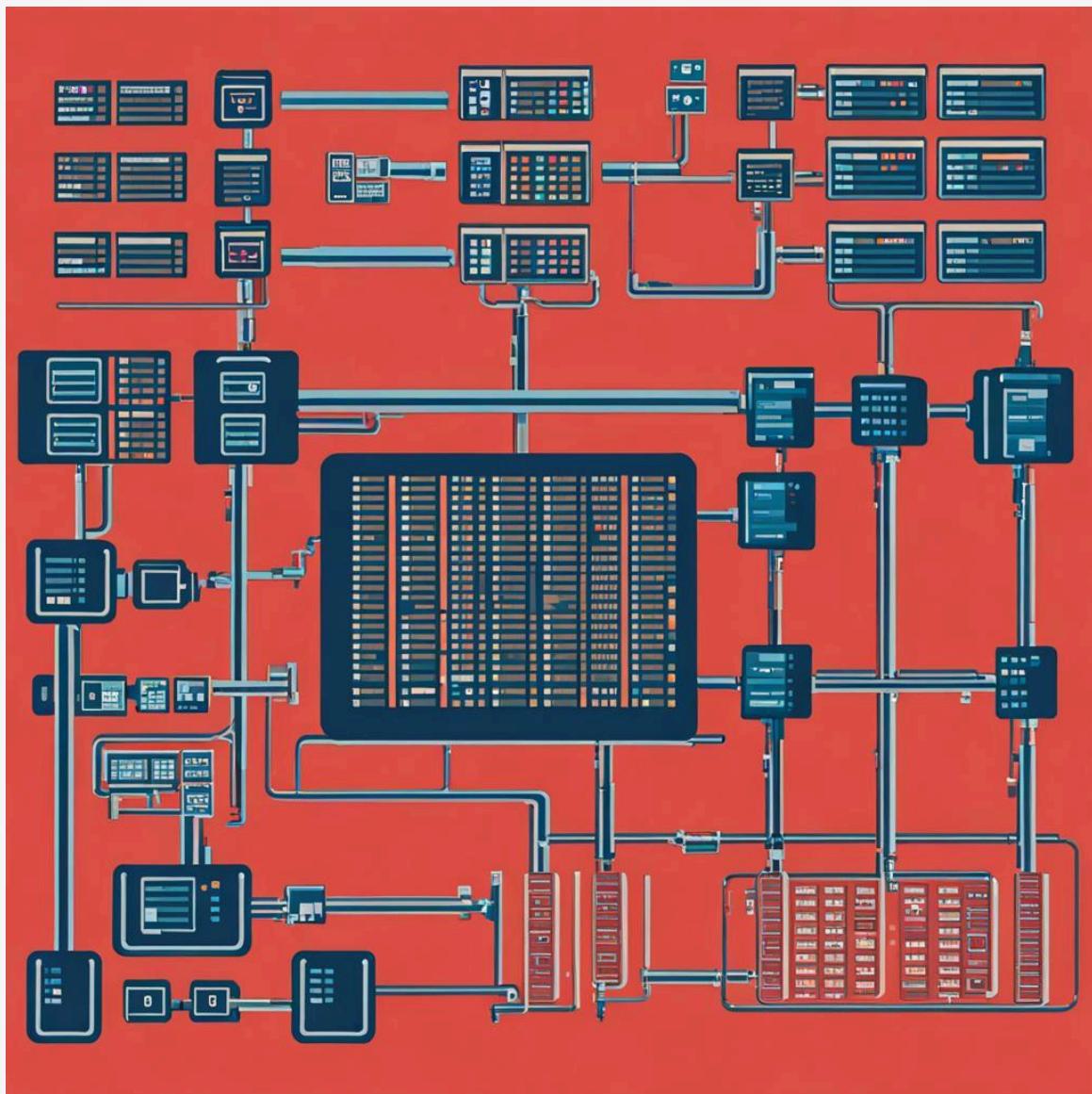


Exploratory Techniques of Genetic Encryption



By Anthony Matarazzo © 2024

Summary



The book seeks to give a non technical reader a link into how better encryption algorithms can work. It shows examples of the various parts of C++ or computer code that is very readable. Examples included are from many advanced scientific professionals. Yet as an industry component, the leverage of genetic encryption is needed today and will be in the future. Such a system as a base component needs a well designed architecture through planning. This book casually discusses strategic system design for historic leverage. A minor treatment of compression based upon formats is also used. Research code formats for HD Video and Audio as needed within text is supplied such as "Two Bit Polygon Dictionary" where colors are geometrically related to neighbors while two bits gives color definition refinement.

Often support for more analysis using systems of C++ container polymorphism are useful tools for compression algorithm research. The text also includes several interesting systems that may also be expanded for an official algorithm. As a formal agreement, many in the world are found very loved by a list of a thousand web sites. The ability

to protect users from hacking and hackers often lies within the designed structure of network communication. The well rounded approach to informing non coders but logical thinkers can uplift the value of computer system integrity and provide a market leverage. System server deployment strategy for a monumental use case placement is often found in providing the necessary network and client infrastructure. In legacy, we always see that problems were solved nicely, but become outdated by the amount of information throughput, expected behavior, and summarization complexity. That is, the flimsy combo box that supported only a thousand number of options by its design, is now updated not only for functionality, but also the user's interface into the database. Database security must also adapt for better communication. Finally, the major of the text is about versatile encryption system base coding design and forgoes network communication problems with a series of system design

features. The functionality of the HTTPS, JavaScript and browser communication system is redesigned around encryption.

Introduction

The capabilities of modern computer language and algorithm execution have almost no barriers within its domain to process logic. The free-form escalation of computer programming, or even the elevated study of it, before being an implementation specialist, shows that ultimately fear dominates the domain. When fear or the act of dismissal occurs, it is evidence of creative thinking to provide a path summary that is lost. One must adhere to the FBI cyber regulations of using a browser to connect with the Internet. Today there are a multitude of cyber crimes that center themselves around the design of the web browser, the server-instantiated software, and ultimately the platform rendering capabilities that the browser API provides. As a compromise to an industry format, you will find that all data systems of a distinct size operate using the capacities of the focused web browser technology domain. The intrinsics of the adaptability of the web browser layout and programming logic give it a distinct signature within the communications realm. Simply, the ability to hack into information as a known operating form is integral within many organizations that provide the sideline of promoting that a cyber crime has occurred.

It is impressive the amount of off-duty Chinese government employees who seek information. The accounting methods of supplying income, as a holistic infringement upon American rights, stems from the publication of communication formats.

The encryption payload and FDIC banking credentials being able to be decrypted come from the FBI cyber crime's necessity of oversight. The ability of FBI employees together with Microsoft employees to manufacture common grounds of technology offers a competitive edge to the logistics of corporate computing. Also, the infrastructure of the information systems must rely upon the currently supported methods supplied by the software provider. Thus having capabilities to legitimize communication protocols that are subject to monitoring and also algorithmic deciphering apply a duality to relationships. That is if a technology company can undersell, and also appear as if creating a positive market, then the backing is tenfold.

The distinct abstraction of logic in computer programming languages is accessible to the general public in the modern day. To specify that ordinary users do not know the terms SQL, HTML, CSS, and JavaScript is ignoring the possibilities of the real world. Typically as an educated individual, the traits for which one chose the field they adapted to is a type of funnel system. One where they would compromise their many desires for integrity of focus. Yet as a part of growth, all people of the world know of the order of things. For example, information appearing in a grid or table. One knows that a table constructed upon a display device has columns and rows. The affirmation of the simpleton understanding is that of transfer to those that are still unaware. That is, they get that information grouped into cells.

The aspect of information being displayed on the screen relies on the fact that it is intelligible. That is written constructs of words, English, or a language. Often the attributes of modern encryption methods make ways to not hide information but expose it. One must note that each packet of information on the Internet that uses published browser technology is decipherable. The maintenance cost has seen that the scientific community can argue payload information strategically to inform people not dedicated to the field of study, as a risk. Hence the summary that all intelligence communications upon the internet is a risk can be devastating to the modern field.

Yet the search for a dedicated multiple indoctrinated resource has seen the capability to surface. Typically a field of study requires that a person practice the application of it for many years past the university level. The ability of a university to instantiate the basic tools needed for research and learning affects not only the learned materials of the brain but also the dedication to an inquisitive personality that provides carriage to industry solutions. Problems of the world and the future do not exist within the confines of an uninformed group. For example, PhD, or MD certifications rely upon many aspects of study and the personality type of an individual. Yet at that level of education credentials, there is no motivation for people to reply to computer technology problems. The hacking precepts of the Chinese often are reflected in stacks of technology leaders. All are found within their comfort domain of having a working solution. One must establish that because of the existence of such groups, APT 41, and HOMELINUX, the roads of previously published scientific work have been superseded. Superseded in the capacity that it is very apt to be scanned.

One of the intrinsic problems with information systems, and designs is distribution and trust value. All people, such as those majoring in the pure sciences, trust the integrity of the computer. They know very little about programming, or software writing and compiling and spend their time on composite scientific study to embrace. The capability of the computer and network distribution implies a vast amount of scalability that is useful. Often the algorithms that are published are the benefit of evolutionary scientific adaptation. That is, a picture can be published that has accountability for multimedia, and distributed computing, and its implementation is standardized. The web browser institutionalized the capabilities of computing and supplies such a winning approach that no other significant technologies challenge.

The note that all of the technologies are distinguished and recognizable as a signature relationship implies that algorithmic deciphering can be accomplished. For example, if you can expect that a CSS document will contain a handful of declarations by attributes, such as font, font size, and colors, then a program can be fashioned where it can iterate through combinations of permutations to arrive at a result. Such information being more tuned with an output format can be regulated by weight. This allows discovery through computing logic to decipher any message that is intended for the web browser.

Yet as an informed and experienced technologist, the shadow knowledge is necessary to translate to real-world users. Users such as graduates from MIT, Berkeley, or Stanford will use the web browser to reflect the power of their credentials. In this, they depend on the inner workings of the computer to facilitate a secure network connection. Typically this is left up to technologists who are pragmatic about performance, or the single silver bullet fits all approaches. Today in modern computer crime, there is a severe leverage the FBI Cyber Crimes division has to manage because of software. That time, when at the project table, eating pizza and working late nights for the big project close project one or a group of individuals operated in their function as a job. Their focus as an employee was not the cyber crimes division, or even perhaps their focus was to ruffle the feathers of the cybercrime division for power. If one can name a division, then a job exists. The ability to pull the rug is based upon the effects of a system where criminals are operating on a better planning method than the FBI does.

[FBI Cyber Division - Wikipedia](#)

[Cyber's Most Wanted — FBI](#)

[Hacker - Wikipedia](#)

[Script kiddie - Wikipedia](#)

[Dark Web - Wikipedia](#)

[Cyberattack - Wikipedia](#)

[Double Dragon \(hacking group\) - Wikipedia](#)

[Hacks at the Massachusetts Institute of Technology - Wikipedia](#)

[Gordon–Loeb model - Wikipedia](#)

United States Law

Hackers are charged typically with the following crimes as worded. The implications are that a specific type of abuse is occurring. The aspect of what a protected computer defines also reflects what is punishable in law. When these crimes are committed, they use network technology between the clients. As a method, often a series of networked computers provide the details to trace back. Encryption, especially peer-to-peer, only protects network access. The compromise of security must be solved with software modifications.

Conspiracy to Commit Computer Fraud; Wire Fraud;

Conspiracy to Commit Wire Fraud;

Conspiracy to Commit Computer Fraud and Abuse

Computer Intrusion and Conspiracy

intentional damage to a protected computer

transmitting a demand concerning damaging a protected computer

Aggravated Identity Theft

Identity Theft

Forfeiture Allegation

Babuk ransomware variant

Lockbit attacks

Money Laundering

Operating an Unlicensed Money Transmitting Business

[Protected computer - Wikipedia](#)

[Computer Fraud and Abuse Act - Wikipedia](#)

As any medium has a hierarchy, the targeting of safe computer systems of enthusiasts combines the necessity of multiple domain knowledge of the hardware. In reference to os compatibility, chaining multiple os together, and finding a niche bothersome. Linux Firmware updates and also steeping problems of names such as power functions. The onscreen keyboard is bothersome. Can someone debug this? Yep? Holes into why it is there is a information within a box.

<https://askubuntu.com/questions/1408136/how-to-disable-on-screen-keyboard-when-using-touchscreen>

Data Encryption Standard

The DES is the form used for the most advanced encryption for consumer-related and corporate communications.

[Data Encryption Standard - Rosetta Code](#)

[EFF DES cracker - Wikipedia](#)

Numerical Base representation

Numerical base representation is an important aspect of encryption. As a standard C++ feature, the aspects of alternate base representation obscure some traits of identity. For example, base 10 is our normal counting system. In the base, we represent the values 0,1,2,3,4,5,6,7,8,9 as a set. That is ten digits. The form of the digit meaning, as a 3 for example, can be represented as any distinct form as a symbol. Strategically changing the base, and symbol dictionary is a transform. One may represent numerical information in obscure base concepts such as radix 95.

Also, numerical base transforms can be used to represent a large array of numeric literal information in less space. Rather than using only ten digits in the alphabet, more can be used to represent the combination of numerical bits. Simple as a data transform, this is distinct rather than the microprocessor binary two's complement format. 64 bits, is a very small space for an integer. These formats can be used in machine math. Typed information that is readable is expressed using the symbols.

[Numeral system - Wikipedia](#)

[Base \(exponentiation\) - Wikipedia](#)

[Radix - Wikipedia](#)

[Mixed radix - Wikipedia](#)

[Logarithm - Wikipedia](#)

[Ternary numeral system - Wikipedia](#)

[Non-decimal radices/Convert - Rosetta Code](#)

[C++ Reference](#)

```
#include <string>
#include <cstdlib>
#include <algorithm>
#include <cassert>

std::string const digits = "0123456789abcdefghijklmnopqrstuvwxyz";

std::string to_base(unsigned long num, int base)
{
    if (num == 0)
        return "0";

    std::string result;
    while (num > 0) {
        std::ldiv_t temp = std::div(num, (long)base);
        result += digits[temp.rem];
        num = temp.quot;
    }
    std::reverse(result.begin(), result.end());
    return result;
}

unsigned long from_base(std::string const& num_str, int base)
{
    unsigned long result = 0;
    for (std::string::size_type pos = 0; pos < num_str.length(); ++pos)
        result = result * base + digits.find(num_str[pos]);
    return result;
}
```

Bit Operations

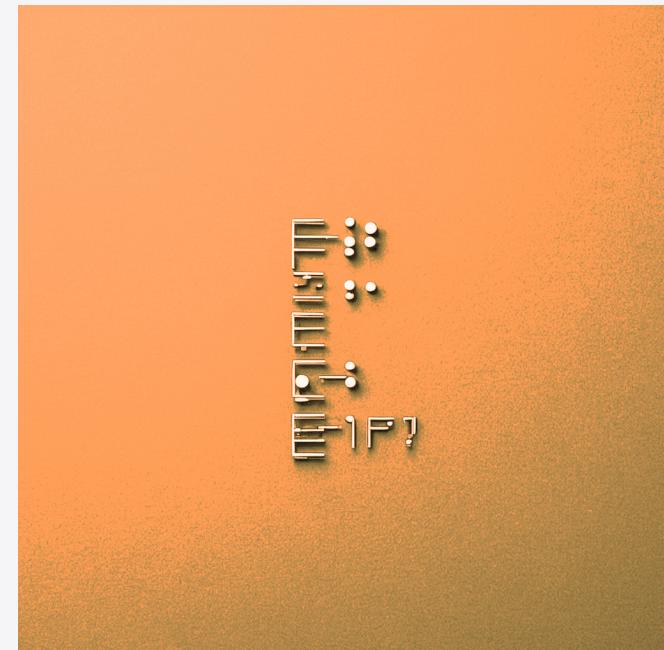
Bit operations are plentiful, and can be expanded with logic to produce a vector of bits. The C++ std bit provides these facilities. Within assembly, or machine code there are multiple implications for types of bit operations. Yet simple bits can be tested to exist as a value or one or not. Bits can be inverted to make 1, a 0, and a 0 a 1. When in a group operations may store information as part of a parity type of logic. That is both bits must be positive for the result to be positive.

Shifting bits within a storage unit such as a quadword can be shifted and rotated. Shift provides that data is lost but is used in the rotation function. Users can use a mask in the upper bits that are known to be lost, setting all other bits to zero. By oring the mask with the input '|' the information can be extracted.

Individual bits can be changed. When two are added together, a carry operation can take place. Such as adding 1B and 1B together produces 10B. The one is carried over to the next place. Since the only two digits in binary are 1 or 0, the placeholder is expanded to the left. Typically numerical information is also. This is similar to adding 9 and 9 together. Both are capable of being represented in one position. But after adding, a carry operation takes place.

The binary representation of numerical information is storage-wise smaller compared to its unicode or ASCII representation. Typically both formats are around for processing when an application calculates the result. That is, calculation on the computer is typically done in binary. The storage of data is symbolically linked to the unit of storage for computer memory. Such as a byte. ASCII codes are one byte apiece. Unicode characters can vary from two to four bytes. An interesting bit operation can make lowercase and uppercase characters with the addition of one bit.

When bits are in a sequence, such as four also known as a nibble, or eight as a byte, the position of the base ten meaning can be classified as a power of two. Simply $2^{[0-n]}$, or $2^7, 2^6, 2^5, 2^4, 2^3, 2^2, 2^1, 2^0$. The chart when inside a type of previous legacy calling is that of information encoding. The study of microprocessor technology envelopes these basic principles to even make sense of memory parameters. The simple aspect to using a grid graph paper and drawing coded bitmap graphics, although a type of on or off image, can be used in bit masks.



128, 64, 32, 16, 8 ,4 ,2 , 1

128	64	32	16	8	4	2	1

Bits can be ored together, anded, shifted left or right which is also a multiplication operation in power of twos. The power of machine code offers that some instructions of programming logic use such bit operations to perform math.

[Bitwise Operators in C - GeeksforGeeks](#)

[Fundamental types - cppreference.com](#)

[Bitwise operation - Wikipedia](#)

[Primitive data type - Wikipedia](#)

[C++ Reference](#)

Bitwise operations in C++.

Bitwise AND (&):

- The bitwise AND operator compares each individual bit of two different numbers and outputs a 1 if they are both 1, and 0 otherwise.

For example:

```
10101101 & 00110010 = 00100000
```

- Some properties of AND:
 - A & 0 = 0
 - A & A = A
 - A & -1 = A (where -1 is represented as FF in hexadecimal)
- Practical use case: Filtering out unwanted bits for a comparison (e.g., checking whether a number is odd or even).

2. Bitwise OR (|):

- The bitwise OR operator compares each individual bit of two different numbers and outputs a 0 if both are 0, and a 1 otherwise.

For example:

$10101101 | 00110010 = 10111111$

- Note that in programming, “or” means “one or the other, but not both.”
- Practical use case: Combining flags or options settings represented by individual bits.

3. Two's Complement:

- The two's complement is a technique to represent negative integers in binary.
- To find the two's complement of a number:
 - Invert all the bits (change 0s to 1s and vice versa).
 - Add 1 to the inverted result.

Example:

```
Original number: -42 (in decimal)
Binary representation: 10101010
Inverted bits: 01010101
Add 1: 01010110 (which is 42 in decimal)
```

Practical use case: Representing negative numbers efficiently.

References:

1. [Bitwise Operators - Rosetta Code¹](#)
2. [Two's complement - Rosetta Code²](#)

Let's explore some **C++ code examples** related to **bit manipulation** using bitwise operators. These operators allow you to work with individual bits within integers. We'll focus on the **left shift (<<)** and **right shift (>>)** operators.

1. Bitwise Left Shift (<<):

- The left shift operator shifts the bits of a number to the left by a specified number of positions.
- New bits shifted in from the right side receive the value 0.

For example:

```
#include <bitset>
#include <iostream>

int main() {
    std::bitset<4> x{0b1100};
    std::cout << x << '\n';           // Prints 1100
    std::cout << (x >> 1) << '\n';   // Shift right by 1: 0110
    std::cout << (x << 1) << '\n';   // Shift left by 1: 1000
    return 0;
}
```

- In this example, we're working with 4-bit binary values. The left shift operation shifts the bits in `x` to the left by 1 place, resulting in `1000`.

2. Bitwise Right Shift (>>):

- The right shift operator shifts the bits of a number to the right by a specified number of positions.
- Bits shifted off the right end of the number are lost.

For example:

```
#include <bitset>
#include <iostream>

int main() {
    std::bitset<4> x{0b1100};
    std::cout << x << '\n';           // Prints 1100
    std::cout << (x >> 1) << '\n';   // Shift right by 1: 0110
    std::cout << (x << 1) << '\n';   // Shift left by 1: 1000
    return 0;
}
```

- In this case, the right shift operation shifts the bits in `x` to the right by 1 place, resulting in `0110`.

Remember that these examples use 4-bit binary values for simplicity. In actual programs, the number of bits used depends on the size of the object (e.g., a 2-byte object would store 16 bits). Additionally, avoid using bitwise operators with signed operands, as they may have implementation-defined results or other potential gotchas. [Best practice is to use unsigned operands or std::bitset for clarity and consistency¹](#).

`any()` value: true

`all()` value: true

`none()` value: false

`test()` at index 0: true

`size()` value: 4

Value of allUnset before using `set()`: 0000

Value of allUnset after using `set()`: 0001

Value of allSet before using `reset()`: 1111

Value of allSet after using `reset()`: 0111

For example, to send one bit over text, a space could be used. If a single is known to exist on the page, its position could be located between two words. Yet only one extra space will exist within the entire range. The change of the punctuation of a particular sentence from a period to a comma. The same with which two are connected. If a text is perfect, as part of the grammar analysis, you can vary potential uses of verbs, or other errors, and a reader would fly by. If the text is spelled correctly a chosen character about the area could imply a bit. Keyboard mishaps with o and l. Introducing text and title manufacturing for a large text of information and using a misleading underline in the title.

Random Numbers

Here's a simple **C++ code snippet** that generates random integers within a specified range:

```
#include <iostream>
#include <random>

int main() {
    // Seed the random number generator with the current time
    std::random_device device;
    std::default_random_engine engine(device());

    // Define the range for random numbers (inclusive)
    int minNumber = 1;
    int maxNumber = 100; // Change this to your desired upper limit

    // Create a uniform distribution for integers in the specified range
    std::uniform_int_distribution<int> distribution(minNumber, maxNumber);

    // Generate and print a random integer
    int randomNumber = distribution(engine);
    std::cout << "Random number between " << minNumber << " and "
          << maxNumber << ":" << randomNumber << std::endl;

    return 0;
}
```

In this code:

- We use the `<random>` library, which provides better random number generation than the older `rand()` function.

- The `std::random_device` is used to seed the random number generator.
- The `std::uniform_int_distribution` ensures that the generated numbers are uniformly distributed within the specified range.

Feel free to adjust the `minNumber` and `maxNumber` variables to suit your needs!

[Random numbers - Rosetta Code](#)

[Random number generation - Wikipedia](#)

[Pseudo Random Number Generator \(PRNG\) - GeeksforGeeks](#)

[Counter-based random number generator - Wikipedia](#)

[Pseudo Random Number Generator \(PRNG\) - GeeksforGeeks](#)

[8.13 — Introduction to random number generation – Learn C++](#)

[C++ Reference](#)

Middle-Square

One iteration of the middle-square method, shows a 6-digit seed, which is then squared, and the resulting value has its middle 6 digits as the output value (and also as the next seed for the sequence).

Directed graph of all 100 2-digit pseudorandom numbers obtained using the middle-square method with $n = 2$.

In mathematics and computer science, the middle-square method is a method of generating pseudorandom numbers. In practice, it is a highly flawed method for many practical purposes since its period is usually very short and it has some severe weaknesses; repeated enough times, the middle-square method will either begin repeatedly generating the same number or cycle to a previous number in the sequence and loop indefinitely.

History

In mathematics

The method was invented by John von Neumann, and was described by him at a conference in 1949.[1]

In the 1949 talk, Von Neumann quipped that "Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin." What he meant, he elaborated, was that there were no true "random numbers", just means to produce them, and "a strict arithmetic procedure", like the middle-square method, "is not such a method". Nevertheless, he found these methods hundreds of times faster than reading "truly" random numbers off punch cards, which had practical importance for his ENIAC work. He found the "destruction" of middle-square sequences to be a factor in their favor, because it could be easily detected: "one always fears the appearance of undetected short cycles".[1] Nicholas Metropolis reported sequences of 750,000 digits before "destruction" by means of using 38-bit numbers with the "middle-square" method.[2]

The book [The Broken Dice](#) by Ivar Ekeland gives an extended account of how the method was invented by a Franciscan friar known only as Brother Edvin sometime between 1240 and 1250.[3] Supposedly, the manuscript is now lost, but Jorge Luis Borges sent Ekeland a copy that he made at the [Vatican](#) Library.

Modifying the middle-square algorithm with a Weyl sequence improves period and randomness.[4][5]

The method

To generate a sequence of n -digit pseudorandom numbers, an n -digit starting value is created and squared, producing a $2n$ -digit number. If the result has fewer than $2n$ digits, leading zeroes are added to compensate. The middle n digits of the result would be the next number in the sequence and returned as the result. This process is then repeated to generate more numbers.

The value of n must be even in order for the method to work – if the value of n is odd, then there will not necessarily be a uniquely defined "middle n -digits" to select from. Consider the following: If a 3-digit number is squared, it can yield a 6-digit number (e.g. $540^2 = 291600$). If there were to be middle 3 digits, that would leave $6 - 3 = 3$ digits to be distributed to the left and right of the middle. It is impossible to evenly distribute these digits equally on both sides of the middle number, and therefore there are no "middle digits". It is acceptable to pad the seeds with zeros to the left in order to create an even-valued n -digit number (e.g. $540 \rightarrow 0540$).

For a generator of n -digit numbers, the period can be no longer than $8n$. If the middle n digits are all zeroes, the generator then outputs zeroes forever. If the first half of a number in the sequence is zeroes, the subsequent numbers will decrease to zero. While these runs of zero are easy to detect, they occur too frequently for this method to be of practical use. The middle-squared method can also get stuck on a number other than zero. For $n = 4$, this occurs with the values 0100, 2500, 3792, and 7600. Other seed values form very short repeating cycles, e.g., 0540 → 2916 → 5030 → 3009. These phenomena

are even more obvious when $n = 2$, as none of the 100 possible seeds generates more than 14 iterations without reverting to 0, 10, 50, 60, or a $24 \leftrightarrow 57$ loop.

Example implementation

Here, the algorithm is rendered in Python 3.12.

```
seed_number = int(input("Please enter a four-digit number:\n[####] "))
number = seed_number
already_seen = set()
counter = 0

while number not in already_seen:
    counter += 1
    already_seen.add(number)
    number = int(str(number * number).zfill(8)[2:6]) # zfill adds padding of zeroes
    print(f"{counter}: {number}")

print(f"We began with {seed_number} and"
      f" have repeated ourselves after {counter} steps"
      f" with {number}.")
```

References

The 1949 papers were not reprinted until 1951. John von Neumann, "Various techniques used in connection with random digits", in A. S. Householder, G. E. Forsythe, and H. H. Germond, eds., Monte Carlo Method, National Bureau of Standards Applied Mathematics Series, vol. 12 (Washington, D.C.: U.S. Government Printing Office, 1951): pp. 36–38.

Donald E. Knuth, The art of computer programming, Vol. 2, Seminumerical algorithms, 2nd edn. (Reading, Mass.: Addison-Wesley, 1981), ch. 3, section 3.1.

Ivar Ekeland (15 June 1996). The Broken Dice, and Other Mathematical Tales of Chance. University of Chicago Press. ISBN 978-0-226-19992-4.

Kneusel, Ron (2018). Random Numbers and Computers (1 ed.). Springer. pp. 13–14.

Widynski, Bernard (April 2017). "Middle-Square Weyl Sequence RNG". arXiv:1704.00358 [cs.CR].

See also

Linear congruential generator

Blum Blum Shub

Blum Blum Shub (B.B.S.) is a pseudorandom number generator proposed in 1986 by Lenore Blum, Manuel Blum and Michael Shub[1] that is derived from Michael O. Rabin's one-way function.

Blum Blum Shub takes the form

$$x_{n+1} = x_n^2 \pmod{M}$$

where $M = pq$ is the product of two large primes p and q . At each step of the algorithm, some output is derived from x_{n+1} ; the output is commonly either the bit parity of x_{n+1} or one or more of the least significant bits of x_{n+1} .

The seed x_0 should be an integer that is co-prime to M (i.e. p and q are not factors of x_0) and not 1 or 0.

The two primes, p and q , should both be congruent to 3 (mod 4) (this guarantees that each quadratic residue has one square root which is also a quadratic residue), and should be safe primes with a small $\gcd((p-3)/2, (q-3)/2)$ (this makes the cycle length large).

An interesting characteristic of the Blum Blum Shub generator is the possibility to calculate any x_i value directly (via Euler's theorem):

$$x_i = \left(x_0^{2^i} \pmod{\lambda(M)} \right) \pmod{M}$$

$$\lambda(M) = \lambda(p \cdot q) = \text{lcm}(p - 1, q - 1)$$

```
#include "bbs.h"

//default constructor
BBS::BBS()
{
    //default insecure params
    this->p = 11;
    this->q = 19;
    //seed
    this->x0 = 4;
    //product of P & Q
    this->n = p * q;
}

/*
    set object parameters
    @param blum prime number p
    @param blum prime number q
    @param integer seed
*/
void BBS::setParams(long p, long q, long seed)
{
    //choose p and q such that gcd(p, q) = 1
    this->p = p;
    this->q = q;
    //random seed should be between 1 & n-1
    this->x0 = seed;
    //blum integer
    this->n = p * q;
}

/*
    overloaded constructor
    @param blum prime number p
    @param blum prime number q
    @param integer seed
*/
BBS::BBS(long p, long q, long seed)
{
    this->setParams(p, q, seed);
}

/*
    Get next random number
*/
long BBS::getRandNum()
{
    long nextRandNum = (this->x0 * this->x0) % this->n;
    this->x0 = nextRandNum;
    return nextRandNum;
}

/*
    Get next random bit
*/
int BBS::getRandBit()
{
    return this->getRandNum() % 2;
}
```

```

void BBS::setP(long p)
{
    this->p = p;
}

void BBS::setQ(long q)
{
    this->q = q;
}

void BBS::setSeed(long seed)
{
    this->x0 = seed;
}

BBS::~BBS(){}

```

<https://github.com/OverStruck/blum-blum-shub-prbg/blob/master/bbs.cpp>

At least the name Phong shading is of a person. A nomenclature targeting identification. Yet names like Blum Blum are more interlaced with scientific and academic secret of inner mind. A dysfunctional connotation that ignites purity and gives it a shelf. Those that admit to succumbing to the likes of Weed Spray close proximity for a shelf are providing themselves a mechanic. As if programs, where no secrets are shared, perhaps they would belong to themselves. Yet in observance merely a compaction of ideas. This Blub Blub with also border Medical gasses, inroads from broken foot, has been what? Tailored as supplied in information. People love a story they are not involved in. Yet imagine if two secret service men are using a foam cup, and string, this random number generator is not very long term. A primary example. Better Blum Blum style algorithms can be developed.

Squares RNG

[2004.06278] Squares: A Fast Counter-Based RNG

```

inline static uint32_t squares32(uint64_t ctr, uint64_t key) {
    uint64_t x, y, z;
    y = x = ctr * key; z = y + key;
    x = x*x + y; x = (x>>32) | (x<<32); /* round 1 */
    x = x*x + z; x = (x>>32) | (x<<32); /* round 2 */
    x = x*x + y; x = (x>>32) | (x<<32); /* round 3 */
    return (x*x + z) >> 32; /* round 4 */
}

inline static uint64_t squares64(uint64_t ctr, uint64_t key) {
    uint64_t t, x, y, z;
    y = x = ctr * key; z = y + key;
    x = x*x + y; x = (x>>32) | (x<<32); /* round 1 */
    x = x*x + z; x = (x>>32) | (x<<32); /* round 2 */
    x = x*x + y; x = (x>>32) | (x<<32); /* round 3 */
    t = x = x*x + z; x = (x>>32) | (x<<32); /* round 4 */
    return t ^ ((x*x + y) >> 32); /* round 5 */
}

```

Ran Method

The certain linear feedback shift register is the window effect of the change bits.

[Linear-feedback shift register - Wikipedia](#)

A very comprehensive and field study with c++ as a more targeted resource for positive and algorithmic filled nature, the [Numerical Recipes](#) is a great find. It is also listed in Wikipedia.

[Numerical Recipes - Wikipedia](#)

[Amazon - Numerical Recipes - Cambridge University Press](#)

“Numerical Recipes is a collection of powerful numerical computing software and libraries that provide researchers, scientists, engineers, and students with essential tools for solving complex mathematical problems and performing accurate computations in various scientific and engineering disciplines.” - Bing

The code below gathers and relabeled for our uses.

```

/***
 * @brief Implementation of the highest quality recommended generator. The
constructor is called with an integer seed and creates an instance of the
generator. The member functions int64, dbl, and int32 return the next values in
the random sequence, as a variable type indicated by their names. The period of
the generator is 3:138 1057.
*
*/
struct Ran {
    uint64_t u, v, w;
    Ran(uint64_t j) : v(4101842887655102017LL), w(1) {
        u = j ^ v;
        _64();
        v = u;
        _64();
        w = v;
        _64();
    }
    inline uint64_t _64() {
        u = u * 2862933555777941757LL + 7046029254386353087LL;
        v ^= v >> 17;
        v ^= v << 31;
        v ^= v >> 8;
        w = 4294957665U * (w & 0xffffffff) + (w >> 32);
        Ullong x = u ^ (u << 21);
        x ^= x >> 35;
        x ^= x << 4;
        return (x + v) ^ w;
    }
    inline double _dbl() { return 5.42101086242752217E-20 * int64(); }
/*
    Return random double -
        precision floating value in the range 0. to 1. */
    inline uint32_t _32() { return (UInt)int64(); }
};

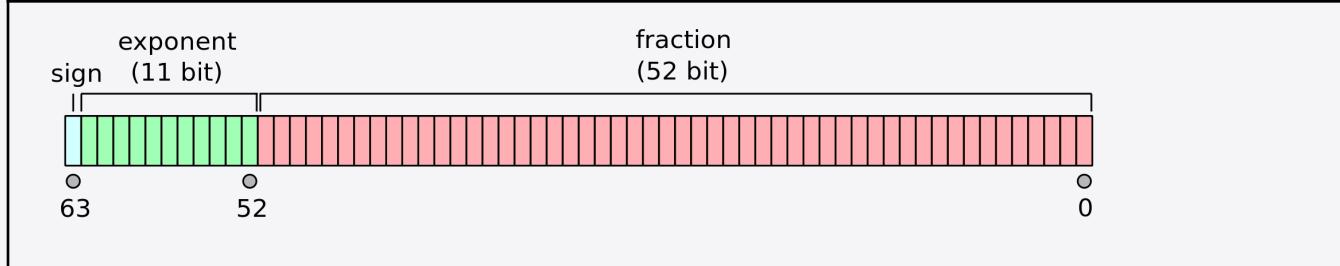
```

286293355577794175
0b1111111001000111100100101001000000110001001101111001111111

7046029254386353087
0b110000111001000100001100100011010000000101101011000001110111111

4294957665
0b111111111111111111111111011101001100001

542101086242752217
0b111100001011101110000100001101010110110100100011011011001



https://en.wikipedia.org/wiki/Double-precision_floating-point_format

In C++, the binary representation of a `double` can be quite fascinating. Let's explore a couple of ways to achieve this:

1. **Using a Union**:

You can create a union that holds both a `double` and an `uint64_t` (which is a 64-bit unsigned integer). By assigning a value to the `double` part of the union and then accessing the `uint64_t` part, you can see the exact binary representation of the `double`. Here's an example:

```
```cpp
union MyUnion {
 double dValue;
 uint64_t iValue;
};

MyUnion myValue;
myValue.dValue = 123.456;
std::cout << myValue.iValue;
```

<sup>1</sup>Note that this approach assumes 64-bit doubles<sup>1</sup>.

## 2. \*\*Direct Memory Manipulation\*\*:

Another method involves directly manipulating the memory bytes of the `double`. This approach doesn't make any assumptions about the size of `double` and generates a binary representation. Here's how you can do it:

```
```cpp
double someDouble = 123.456;
unsigned char rawBytes[sizeof(double)];
memcpy(rawBytes, &someDouble, sizeof(double));

unsigned char startMask = 1;
while (0 != static_cast<unsigned char>(startMask << 1)) {
    startMask <<= 1;
}

bool hasLeadBit = false; // Set this to true if you want to see leading zeros
size_t byteIndex;
for (byteIndex = 0; byteIndex < sizeof(double); ++byteIndex) {
    unsigned char bitMask = startMask;
    while (0 != bitMask) {
        if (0 != (bitMask & rawBytes[byteIndex])) {
            std::cout << "1";
            hasLeadBit = true;
        } else if (hasLeadBit) {
            std::cout << "0";
        }
        bitMask >>= 1;
    }
}

if (!hasLeadBit) {
    std::cout << "0";
}

```

This version works regardless of the size of `double` and provides the binary representation¹. Thank you Mr. Bing for a working program. These methods can also be described in other functions. The aspects of using STL allow the use of .data() as part of the vector. Yet more precisely a combination of union and largest data type. The nibble and bit access are also found interesting for use.

Remember that these methods are for exploration and understanding. In practice, you'd rarely need to inspect the binary representation of a `double`.

Source: Conversation with Bing, 3/29/2024

(1) c++ - Exact binary representation of a double - Stack Overflow.

<https://stackoverflow.com/questions/8521638/exact-binary-representation-of-a-double>.

(2) IEEE Floating-Point Representation | Microsoft Learn.

<https://learn.microsoft.com/en-us/cpp/build/ieee-floating-point-representation?view=msvc-170>.

(3) How do I display the binary representation of a float or double?.

<https://stackoverflow.com/questions/397692/how-do-i-display-the-binary-representation-of-a-float-or-double>.

(4) 4.8 — Floating point numbers – Learn C++ - LearnCpp.com.

<https://www.learncpp.com/cpp-tutorial/floating-point-numbers/>.

Weyl Sequence RNG

```
uint64_t x1 = 0, w1 = 0, s1 = 0xb5ad4eceda1ce2a9;
uint64_t x2 = 0, w2 = 0, s2 = 0x278c5a4d8419fe6b;
inline static uint64_t msws64() {
    uint64_t xx;
    x1 *= x1; xx = x1 += (w1 += s1); x1 = (x1 >> 32) | (x1 << 32);
    x2 *= x2; x2 += (w2 += s2); x2 = (x2 >> 32) | (x2 << 32);
    return xx ^ x2;
}
```

<https://arxiv.org/pdf/1704.00358.pdf>

The picture of the s1 and s2

0b10110101101011010011101100111011011010000111001110001010101001

0b100111100011000101101001001101100001000001100111111001101011

Xorshift

Example random distribution of Xorshift128

Xorshift random number generators, also called shift-register generators, are a class of pseudorandom number generators that were invented by George Marsaglia.[1] They are a subset of linear-feedback shift registers (LFSRs) which allow a particularly efficient implementation in software without the excessive use of sparse polynomials.[2] They generate the next number in their sequence by repeatedly taking the exclusive or of a number with a bit-shifted version of itself. This makes execution extremely efficient on modern computer architectures, but it does not benefit efficiency in a hardware implementation. Like all LFSRs, the parameters have to be chosen very carefully in order to achieve a long period.[3]

For execution in software, xorshift generators are among the fastest PRNGs, requiring very small code and state. However, they do not pass every statistical test without further refinement. This weakness is amended by combining them with a nonlinear function, as described in the original paper. Because plain xorshift generators (without a non-linear step) fail some statistical tests, they have been accused of being unreliable.[3]:360

Example implementation

A C version[a] of three xorshift algorithms[1]:4,5 is given here. The first has one 32-bit word of state, and period 2³²-1. The second has one 64-bit word of state and period 2⁶⁴-1. The last one has four 32-bit words of state, and period 2¹²⁸-1. The 128-bit algorithm passes the diehard tests. However, it fails the MatrixRank and LinearComp tests of the BigCrush test suite from the TestU01 framework.

All use three shifts and three or four exclusive-or operations:

```
#include <stdint.h>

struct xorshift32_state {
    uint32_t a;
};

/* The state must be initialized to non-zero */
uint32_t xorshift32(struct xorshift32_state *state)
{
    /* Algorithm "xor" from p. 4 of Marsaglia, "Xorshift RNGs" */
    uint32_t x = state->a;
    x ^= x << 13;
    x ^= x >> 17;
    x ^= x << 5;
    return state->a = x;
}

struct xorshift64_state {
    uint64_t a;
};

uint64_t xorshift64(struct xorshift64_state *state)
{
    uint64_t x = state->a;
    x ^= x << 13;
    x ^= x >> 7;
    x ^= x << 17;
    return state->a = x;
}

/* struct xorshift128_state can alternatively be defined as a pair
   of uint64_t or a uint128_t where supported */
struct xorshift128_state {
    uint32_t x[4];
};

/* The state must be initialized to non-zero */
uint32_t xorshift128(struct xorshift128_state *state)
{
    /* Algorithm "xor128" from p. 5 of Marsaglia, "Xorshift RNGs" */
    uint32_t t = state->x[3];

    uint32_t s = state->x[0]; /* Perform a contrived 32-bit shift. */
    state->x[3] = state->x[2];
    state->x[2] = state->x[1];
    state->x[1] = s;

    t ^= t << 11;
    t ^= t >> 8;
    return state->x[0] = t ^ s ^ (s >> 19);
}
```

Non-linear variations

All xorshift generators fail some tests in the BigCrush test suite. This is true for all generators based on linear recurrences, such as the Mersenne Twister or WELL. However, it is easy to scramble the output of such generators to improve their quality.

The scramblers known as + and * still leave weakness in the low bits,[4] so they are intended for floating point use, where the lowest bits of floating-point numbers have a smaller impact on the interpreted value.[5] For general purpose, the scrambler ** (pronounced starstar) makes the LFSR generators pass in all bits.

XORWOW

Marsaglia suggested scrambling the output by combining it with a simple additive counter modulo 232 (which he calls a "Weyl sequence" after Weyl's equidistribution theorem). This also increases the period by a factor of 232, to 2192–232:

```
#include <stdint.h>

struct xorwow_state {
    uint32_t x[5];
    uint32_t counter;
};

/* The state array must be initialized to not be all zero in the first four words */
uint32_t xorwow(struct xorwow_state *state)
{
    /* Algorithm "xorwow" from p. 5 of Marsaglia, "Xorshift RNGs" */
    uint32_t t = state->x[4];

    uint32_t s = state->x[0]; /* Perform a contrived 32-bit shift. */
    state->x[4] = state->x[3];
    state->x[3] = state->x[2];
    state->x[2] = state->x[1];
    state->x[1] = s;

    t ^= t >> 2;
    t ^= t << 1;
    t ^= s ^ (s << 4);
    state->x[0] = t;
    state->counter += 362437;
    return t + state->counter;
}
```

This performs well, but fails a few tests in BigCrush.[6] This generator is the default in Nvidia's CUDA toolkit.[7]

xorshift*

An xorshift* generator applies an invertible multiplication (modulo the word size) as a non-linear transformation to the output of an xorshift generator, as suggested by Marsaglia.[1] All xorshift* generators emit a sequence of values that is equidistributed in the maximum possible dimension (except that they will never output zero for 16 calls, i.e. 128 bytes, in a row).[8]

The following 64-bit generator has a maximal period of 2⁶⁴–1.[8]

```
#include <stdint.h>

/* xorshift64s, variant A_1(12,25,27) with multiplier M_32 from line 3 of table 5 */
uint64_t xorshift64star(void) {
    /* initial seed must be nonzero, don't use a static variable for the state if multithreaded */
    static uint64_t x = 1;
    x ^= x >> 12;
    x ^= x << 25;
    x ^= x >> 27;
    return x * 0x2545F4914F6CDD1DULL;
}
```

The generator fails only the MatrixRank test of BigCrush, however if the generator is modified to return only the high 32 bits, then it passes BigCrush with zero failures.[9]:7 In fact, a reduced version with only 40 bits of internal state passes the suite, suggesting a large safety margin.[9]:19 A similar generator suggested in Numerical Recipes[10] as RanQ1 also fails the BirthdaySpacings test.

xorshift1024*

Vigna[8] suggests the following xorshift1024* generator with 1024 bits of state and a maximal period of $2^{1024}-1$; however, it does not always pass BigCrush.[4] xoshiro256** is therefore a much better option.

```
#include <stdint.h>

/* The state must be seeded so that there is at least one non-zero element in array */
struct xorshift1024s_state {
    uint64_t x[16];
    int index;
};

uint64_t xorshift1024s(struct xorshift1024s_state *state)
{
    int index = state->index;
    uint64_t const s = state->x[index++];
    uint64_t t = state->x[index &= 15];
    t ^= t << 31;           // a
    t ^= t >> 11;          // b -- Again, the shifts and the multipliers are tunable
    t ^= s ^ (s >> 30);   // c
    state->x[index] = t;
    state->index = index;
    return t * 1181783497276652981ULL;
}
```

xorshift+

An xorshift+ generator can achieve an order of magnitude fewer failures than Mersenne Twister or WELL. A native C implementation of an xorshift+ generator that passes all tests from the BigCrush suite can typically generate a random number in fewer than 10 clock cycles on x86, thanks to instruction pipelining.[11]

Rather than using multiplication, it is possible to use addition as a faster non-linear transformation. The idea was first proposed by Saito and Matsumoto (also responsible for the Mersenne Twister) in the XSadd generator, which adds two consecutive outputs of an underlying xorshift generator based on 32-bit shifts.[12] However, one disadvantage of adding consecutive outputs is that, while the underlying xorshift128 generator is 2-dimensionally equidistributed, the xorshift128+ generator is only 1-dimensionally equidistributed.[13]

XSadd has some weakness in the low-order bits of its output; it fails several BigCrush tests when the output words are bit-reversed. To correct this problem, Vigna introduced the xorshift+ family,[13] based on 64-bit shifts. xorshift+ generators, even as large as xorshift 1024, exhibit some detectable linearity in the low-order bits of their output;[4] it passes BigCrush, but doesn't when the 32 lowest-order bits are used in reverse order from each 64-bit word.[4] This generator is one of the fastest generators passing BigCrush.[11]

xorshift128+

The following xorshift128+ generator uses 128 bits of state and has a maximal period of $2^{128}-1$.

```
#include <stdint.h>

struct xorshift128p_state {
    uint64_t x[2];
};

/* The state must be seeded so that it is not all zero */

uint64_t xorshift128p(struct xorshift128p_state *state)
{
    uint64_t t = state->x[0];
```

```

uint64_t const s = state->x[1];

state->x[0] = s;

t ^= t << 23;           // a

t ^= t >> 18;          // b -- Again, the shifts and the multipliers are tunable

t ^= s ^ (s >> 5);    // c

state->x[1] = t;

return t + s;
}

```

xoshiro

xoshiro (short for "xor, shift, rotate") and xoroshiro (short for "xor, rotate, shift, rotate") use rotations in addition to shifts. According to Vigna, they are faster and produce better quality output than xorshift.[14][15]

This class of generator has variants for 32-bit and 64-bit integer and floating point output; for floating point, one takes the upper 53 bits (for binary64) or the upper 23 bits (for binary32), since the upper bits are of better quality than the lower bits in the floating point generators. The algorithms also include a jump function, which sets the state forward by some number of steps – usually a power of two that allows many threads of execution to start at distinct initial states.

For 32-bit output, xoshiro128** and xoshiro128+ are exactly equivalent to xoshiro256** and xoshiro256+, with uint32_t in place of uint64_t, and with different shift/rotate constants.

More recently, the xoshiro++ generators have been made as an alternative to the xoshiro** generators. They are used in some implementations of Fortran compilers such as GNU Fortran, Java, and Julia.[16]

xoshiro256**

xoshiro256** is the family's general-purpose random 64-bit number generator. It is used in GNU Fortran compiler, Lua (as of Lua 5.4), and the .NET framework (as of .NET 6.0).[16]

```

/* Adapted from the code included on Sebastiano Vigna's website */

#include <stdint.h>

uint64_t rol64(uint64_t x, int k)
{
    return (x << k) | (x >> (64 - k));
}

struct xoshiro256ss_state {
    uint64_t s[4];
};

uint64_t xoshiro256ss(struct xoshiro256ss_state *state)
{
    uint64_t *s = state->s;
    uint64_t const result = rol64(s[1] * 5, 7) * 9;
    uint64_t const t = s[1] << 17;

    s[2] ^= s[0];
    s[3] ^= s[1];
    s[1] ^= s[2];
    s[0] ^= s[3];
}
```

```
s[2] ^= t;
s[3] = rol64(s[3], 45);

return result;
}
```

xoshiro256+

xoshiro256+ is approximately 15% faster than xoshiro256**, but the lowest three bits have low linear complexity; therefore, it should be used only for floating point results by extracting the upper 53 bits.

It is a 64-bit PRNG that uses a carefully constructed linear transformation. This produces a fast PRNG with excellent statistical quality [1]. xoshiro256** has a period of $2^{256} - 1$ and supports jumping the sequence in increments of 2^{128} , which allows multiple non-overlapping subsequences to be generated.

```
/* Written in 2018 by David Blackman and Sebastiano Vigna (vigna@acm.org)
```

```
To the extent possible under law, the author has dedicated all copyright  
and related and neighboring rights to this software to the public domain  
worldwide. This software is distributed without any warranty.
```

```
See <http://creativecommons.org/publicdomain/zero/1.0/>. */
```

```
#include <stdint.h>
```

```
/* This is xoshiro256** 1.0, one of our all-purpose, rock-solid  
generators. It has excellent (sub-ns) speed, a state (256 bits) that is  
large enough for any parallel application, and it passes all tests we  
are aware of.
```

For generating just floating-point numbers, xoshiro256+ is even faster.

The state must be seeded so that it is not everywhere zero. If you have
a 64-bit seed, we suggest to seed a splitmix64 generator and use its
output to fill s. */

```
static inline uint64_t rotl(const uint64_t x, int k) {
    return (x << k) | (x >> (64 - k));
}
```

```
static uint64_t s[4];
```

```
uint64_t next(void) {
    const uint64_t result = rotl(s[1] * 5, 7) * 9;

    const uint64_t t = s[1] << 17;

    s[2] ^= s[0];
    s[3] ^= s[1];
    s[1] ^= s[2];
    s[0] ^= s[3];

    s[2] ^= t;

    s[3] = rotl(s[3], 45);

    return result;
}
```

```
/* This is the jump function for the generator. It is equivalent  
to  $2^{128}$  calls to next(); it can be used to generate  $2^{128}$   
non-overlapping subsequences for parallel computations. */
```

```
void jump(void) {
```

```

static const uint64_t JUMP[] = {0x180ec6d33cf0aba, 0xd5a61266f0c9392c,
                             0xa9582618e03fc9aa, 0x39abdc4529b1661c};

uint64_t s0 = 0;
uint64_t s1 = 0;
uint64_t s2 = 0;
uint64_t s3 = 0;
for (int i = 0; i < sizeof JUMP / sizeof *JUMP; i++)
    for (int b = 0; b < 64; b++) {
        if (JUMP[i] & UINT64_C(1) << b) {
            s0 ^= s[0];
            s1 ^= s[1];
            s2 ^= s[2];
            s3 ^= s[3];
        }
        next();
    }

s[0] = s0;
s[1] = s1;
s[2] = s2;
s[3] = s3;
}

/* This is the long-jump function for the generator. It is equivalent to
2^192 calls to next(); it can be used to generate 2^64 starting points,
from each of which jump() will generate 2^64 non-overlapping
subsequences for parallel distributed computations. */

void long_jump(void) {
    static const uint64_t LONG_JUMP[] = {0x76e15d3efefdcbbf, 0xc5004e441c522fb3,
                                         0x77710069854ee241, 0x39109bb02acbe635};

    uint64_t s0 = 0;
    uint64_t s1 = 0;
    uint64_t s2 = 0;
    uint64_t s3 = 0;
    for (int i = 0; i < sizeof LONG_JUMP / sizeof *LONG_JUMP; i++)
        for (int b = 0; b < 64; b++) {
            if (LONG_JUMP[i] & UINT64_C(1) << b) {
                s0 ^= s[0];
                s1 ^= s[1];
                s2 ^= s[2];
                s3 ^= s[3];
            }
            next();
        }

    s[0] = s0;
    s[1] = s1;
    s[2] = s2;
    s[3] = s3;
}

```

<https://prng.di.unimi.it/xoshiro256starstar.c>
<https://prng.di.unimi.it/xoshiro128starstar.c>

The numbers below are a link to a class oriented input as well as combining the steps. The code is well written with dynamic arrays already. Expansion is distinct in the data type format. The integers, or normal numbers are described in the hexadecimal format. Yet inspecting a binary range of the format at 64 bits and unsigned, the binary representation is reduced to a pictorial of understanding. These are large numbers to describe and also permute to a statistical point perhaps. Yet the pattern seems communicative and symmetrical with knowledge that perhaps sections represent a meaning range of numbers in binary which is simple to understand like a number between 0 and 10. Five which is 101b. As the range progresses or declines difficulties come about. Some aspects of these binary representations are plotted for the typical ranges of information.

0x76e15d3efefdcbbf
0b11000000111011001101101001100111100111110100010101011010

0xc5004e441c522fb3
0b110001010000000001001110010001000001110001010010001011110110011

0x77710069854ee241
0b11101110111001000000000110100110001010100111011000100100001

0x39109bb02acbe635
0b1110010001000100110110110000010101011001011110011000110101

A 64 bit double looks like this.



By parameterization of these, a data publisher such as a pointer or reference, can be a numerical adaptor. A system of intelligence must be used to modify these values. For example, testing permutations and small variances must be installed for a length of time during the cycle. Can these be modified is a question?

xoroshiro

If space is at a premium, xoroshiro128** and xoroshiro128+ are equivalent to xoshiro256** and xoshiro256+. These have smaller state spaces, and thus are less useful for massively parallel programs. xoroshiro128+ also exhibits a mild dependency in the population count, generating a failure after 5 TB of output. The authors do not believe that this can be detected in real world programs.

xoroshiro64** and xoroshiro64* are equivalent to xoroshiro128** and xoroshiro128+. Unlike the xoshiro generators, they are not straightforward ports of their higher-precision counterparts.

Initialization

In the xoshiro paper, it is recommended to initialize the state of the generators using a generator which is radically different from the initialized generators, as well as one which will never give the "all-zero" state; for shift-register generators, this state is impossible to escape from.[15][17] The authors specifically recommend using the SplitMix64 generator, from a 64-bit seed, as follows:

```
#include <stdint.h>

struct splitmix64_state {
    uint64_t s;
};

uint64_t splitmix64(struct splitmix64_state *state) {
    uint64_t result = (state->s += 0x9E3779B97f4A7C15);
    result = (result ^ (result >> 30)) * 0xBF58476D1CE4E5B9;
    result = (result ^ (result >> 27)) * 0x94D049BB133111EB;
    return result ^ (result >> 31);
}

struct xorshift128_state {
    uint32_t x[4];
};

// one could do the same for any of the other generators
```

```

void xorshift128_init(struct xorshift128_state *state, uint64_t seed) {
    struct splitmix64_state smstate = {seed};

    uint64_t tmp = splitmix64(&smstate);
    state->x[0] = (uint32_t)tmp;
    state->x[1] = (uint32_t)(tmp >> 32);

    tmp = splitmix64(&smstate);
    state->x[2] = (uint32_t)tmp;
    state->x[3] = (uint32_t)(tmp >> 32);
}

```

Notes

In C and most other C-based languages, `^` represents bitwise XOR, and `<<` and `>>` represent bitwise shifts.

References

- Marsaglia, George (July 2003). "Xorshift RNGs". *Journal of Statistical Software*. 8 (14). doi:10.18637/jss.v008.i14.
- Brent, Richard P. (August 2004). "Note on Marsaglia's Xorshift Random Number Generators". *Journal of Statistical Software*. 11 (5). doi:10.18637/jss.v011.i05. hdl:1885/34049.
- Panneton, François; L'Ecuyer, Pierre (October 2005). "On the xorshift random number generators" (PDF). *ACM Transactions on Modeling and Computer Simulation*. 15 (4): 346–361. doi:10.1145/1113316.1113319. S2CID 11136098.
- Lemire, Daniel; O'Neill, Melissa E. (April 2019). "Xorshift1024*, Xorshift1024+, Xorshift128+ and Xoroshiro128+ Fail Statistical Tests for Linearity". *Computational and Applied Mathematics*. 350: 139–142. arXiv:1810.05313. doi:10.1016/j.cam.2018.10.019. S2CID 52983294. We report that these scrambled generators systematically fail Big Crush—specifically the linear-complexity and matrix-rank tests that detect linearity—when taking the 32 lowest-order bits in reverse order from each 64-bit word.
- "ISO/IEC 60559:2020". ISO.
- Le Floc'h, Fabien (12 January 2011). "XORWOW L'ecuyer TestU01 Results". Chase The Devil (blog). Retrieved 2017-11-02.
- "cuRAND Testing". Nvidia. Retrieved 2017-11-02.
- Vigna, Sebastiano (July 2016). "An experimental exploration of Marsaglia's xorshift generators, scrambled" (PDF). *ACM Transactions on Mathematical Software*. 42 (4): 30. arXiv:1402.6246. doi:10.1145/2845077. S2CID 13936073. Proposes xorshift* generators, adding a final multiplication by a constant.
- O'Neill, Melissa E. (5 September 2014). PCG: A Family of Simple Fast Space-Efficient Statistically Good Algorithms for Random Number Generation (PDF) (Technical report). Harvey Mudd College. pp. 6–8. HMC-CS-2014-0905.
- Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007). "Section 7.1.2.A. 64-bit Xorshift Method". *Numerical Recipes: The Art of Scientific Computing* (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8.
- Vigna, Sebastiano. "xorshift*/xorshift+ generators and the PRNG shootout". Retrieved 2014-10-25.
- Saito, Mutsuo; Matsumoto, Makoto (2014). "XORSHIFT-ADD (XSadd): A variant of XORSHIFT". Retrieved 2014-10-25.
- Vigna, Sebastiano (May 2017). "Further scramblings of Marsaglia's xorshift generators" (PDF). *Journal of Computational and Applied Mathematics*. 315 (C): 175–181. arXiv:1404.0390. doi:10.1016/j.cam.2016.11.006. S2CID 6876444. Describes xorshift+ generators, a generalization of XSadd.
- Vigna, Sebastiano. "xoshiro/xoroshiro generators and the PRNG shootout". Retrieved 2019-07-07.
- Blackman, David; Vigna, Sebastiano (2018). "Scrambled Linear Pseudorandom Number Generators". *Data Structures and Algorithms*. arXiv:1805.01407.
- "xoshiro / xoroshiro generators and the PRNG shootout". Retrieved 2023-09-07.
- Matsumoto, Makoto; Wada, Isaku; Kuramoto, Ai; Ashihara, Hyo (September 2007). "Common defects in initialization of pseudorandom number generators". *ACM Transactions on Modeling and Computer Simulation*. 17 (4): 15–es. doi:10.1145/1276927.1276928. S2CID 1721554.

Ziggurat algorithm

The ziggurat algorithm is an algorithm for pseudo-random number sampling. Belonging to the class of rejection sampling algorithms, it relies on an underlying source of uniformly-distributed random numbers, typically from a pseudo-random number generator, as well as precomputed tables. The algorithm is used to generate values from a monotonically decreasing probability distribution. It can also be applied to symmetric unimodal distributions, such as the normal distribution, by choosing a value from one half of the distribution and then randomly choosing which half the value is considered to have been drawn from. It was developed by George Marsaglia and others in the 1960s.

A typical value produced by the algorithm only requires the generation of one random floating-point value and one random table index, followed by one table lookup, one multiply operation and one comparison. Sometimes (2.5% of the time, in the

case of a normal or exponential distribution when using typical table sizes)[citation needed] more computations are required. Nevertheless, the algorithm is computationally much faster[citation needed] than the two most commonly used methods of generating normally distributed random numbers, the Marsaglia polar method and the Box–Muller transform, which require at least one logarithm and one square root calculation for each pair of generated values. However, since the ziggurat algorithm is more complex to implement it is best used when large quantities of random numbers are required.

The term ziggurat algorithm dates from Marsaglia's paper with Wai Wan Tsang in 2000; it is so named because it is conceptually based on covering the probability distribution with rectangular segments stacked in decreasing order of size, resulting in a figure that resembles a ziggurat.

The Ziggurat algorithm used to generate sample values with a normal distribution. (Only positive values are shown for simplicity.) The pink dots are initially uniform-distributed random numbers. The desired distribution function is first segmented into equal areas "A". One layer i is selected at random by the uniform source at the left. Then a random value from the top source is multiplied by the width of the chosen layer, and the result is x tested to see which region of the slice it falls into with 3 possible outcomes: 1) (left, solid black region) the sample clearly under the curve and is passed directly to output, 2) (right, vertically striped region) the sample value may lie under the curve, and must be tested further. In that case, a random y value within the chosen layer is generated and compared to $f(x)$. If less, the point is under the curve and the value x is output. If not, (the third case), the chosen point x is rejected and the algorithm is restarted from the beginning.

Theory of operation

The ziggurat algorithm is a rejection sampling algorithm; it randomly generates a point in a distribution slightly larger than the desired distribution, then tests whether the generated point is inside the desired distribution. If not, it tries again. Given a random point underneath a probability density curve, its x coordinate is a random number with the desired distribution.

The distribution the ziggurat algorithm chooses from is made up of n equal-area regions; $n - 1$ rectangles that cover the bulk of the desired distribution, on top of a non-rectangular base that includes the tail of the distribution.

Given a monotone decreasing probability density function $f(x)$, defined for all $x \geq 0$, the base of the ziggurat is defined as all points inside the distribution and below $y_1 = f(x_1)$. This consists of a rectangular region from $(0, 0)$ to (x_1, y_1) , and the (typically infinite) tail of the distribution, where $x > x_1$ (and $y < y_1$).

This layer (call it layer 0) has area A . On top of this, add a rectangular layer of width x_1 and height A/x_1 , so it also has area A . The top of this layer is at height $y_2 = y_1 + A/x_1$, and intersects the density function at a point (x_2, y_2) , where $y_2 = f(x_2)$. This layer includes every point in the density function between y_1 and y_2 , but (unlike the base layer) also includes points such as (x_1, y_2) which are not in the desired distribution.

Further layers are then stacked on top. To use a precomputed table of size n ($n = 256$ is typical), one chooses x_1 such that $x_n = 0$, meaning that the top box, layer $n - 1$, reaches the distribution's peak at $(0, f(0))$ exactly.

Layer i extends vertically from y_i to y_{i+1} , and can be divided into two regions horizontally: the (generally larger) portion from 0 to x_{i+1} which is entirely contained within the desired distribution, and the (small) portion from x_{i+1} to x_i , which is only partially contained.

Ignoring for a moment the problem of layer 0, and given uniform random variables U_0 and $U_1 \in [0, 1]$, the ziggurat algorithm can be described as:

Choose a random layer $0 \leq i < n$.

Let $x = U_0 x_i$.

If $x < x_{i+1}$, return x .

Let $y = y_i + U_1(y_{i+1} - y_i)$.

Compute $f(x)$. If $y < f(x)$, return x .

Otherwise, choose new random numbers and go back to step 1.

Step 1 amounts to choosing a low-resolution y coordinate. Step 3 tests if the x coordinate is clearly within the desired density function without knowing more about the y coordinate. If it is not, step 4 chooses a high-resolution y coordinate, and step 5 does the rejection test.

With closely spaced layers, the algorithm terminates at step 3 a very large fraction of the time. For the top layer $n - 1$, however, this test always fails, because $x_n = 0$.

Layer 0 can also be divided into a central region and an edge, but the edge is an infinite tail. To use the same algorithm to check if the point is in the central region, generate a fictitious $x_0 = A/y_1$. This will generate points with $x < x_1$ with the correct frequency, and in the rare case that layer 0 is selected and $x \geq x_1$, use a special fallback algorithm to select a point at random from the tail. Because the fallback algorithm is used less than one time in a thousand, speed is not essential.

Thus, the full ziggurat algorithm for one-sided distributions is:

Choose a random layer $0 \leq i < n$.

Let $x = U_0 x_i$

If $x < xi+1$, return x .

If $i = 0$, generate a point from the tail using the fallback algorithm.

Let $y = yi + U1(yi+1 - yi)$.

Compute $f(x)$. If $y < f(x)$, return x .

Otherwise, choose new random numbers and go back to step 1.

For a two-sided distribution, the result must be negated 50% of the time. This can often be done conveniently by choosing $U0 \in (-1,1)$ and, in step 3, testing if $|x| < xi+1$.

Fallback algorithms for the tail

Because the ziggurat algorithm only generates most outputs very rapidly, and requires a fallback algorithm whenever $x > x1$, it is always more complex than a more direct implementation. The specific fallback algorithm depends on the distribution.

For an exponential distribution, the tail looks just like the body of the distribution. One way is to fall back to the most elementary algorithm $E = -\ln(U1)$ and let $x = x1 - \ln(U1)$. Another is to call the ziggurat algorithm recursively and add $x1$ to the result.

For a normal distribution, Marsaglia suggests a compact algorithm:

Let $x = -\ln(U1)/x1$.

Let $y = -\ln(U2)$.

If $2y > x2$, return $x + x1$.

Otherwise, go back to step 1.

Since $x1 \approx 3.5$ for typical table sizes, the test in step 3 is almost always successful.

Optimizations

The algorithm can be performed efficiently with precomputed tables of xi and $yi = f(xi)$, but there are some modifications to make it even faster:

Nothing in the ziggurat algorithm depends on the probability distribution function being normalized (integral under the curve equal to 1), removing normalizing constants can speed up the computation of $f(x)$.

Most uniform random number generators are based on integer random number generators which return an integer in the range $[0, 2^{32} - 1]$. A table of $2^{32}xi$ lets you use such numbers directly for $U0$.

When computing two-sided distributions using a two-sided $U0$ as described earlier, the random integer can be interpreted as a signed number in the range $[-2^{31}, 2^{31} - 1]$, and a scale factor of 2^{-31} can be used.

Rather than comparing $U0xi$ to $xi+1$ in step 3, it is possible to precompute $xi+1/xi$ and compare $U0$ with that directly. If $U0$ is an integer random number generator, these limits may be premultiplied by 2^{32} (or 2^{31} , as appropriate) so an integer comparison can be used.

With the above two changes, the table of unmodified xi values is no longer needed and may be deleted.

When generating IEEE 754 single-precision floating point values, which only have a 24-bit mantissa (including the implicit leading 1), the least-significant bits of a 32-bit integer random number are not used. These bits may be used to select the layer number. (See the references below for a detailed discussion of this.)

The first three steps may be put into an inline function, which can call an out-of-line implementation of the less frequently needed steps.

Generating the tables

It is possible to store the entire table precomputed, or just include the values n , $y1$, A , and an implementation of $f^{-1}(y)$ in the source code, and compute the remaining values when initializing the random number generator.

As previously described, you can find $xi = f^{-1}(yi)$ and $yi+1 = yi + A/xi$. Repeat $n - 1$ times for the layers of the ziggurat. At the end, you should have $yn = f(0)$. There will be some round-off error, but it is a useful sanity test to see that it is acceptably small.

When actually filling in the table values, just assume that $xn = 0$ and $yn = f(0)$, and accept the slight difference in layer $n - 1$'s area as rounding error.

Finding $x1$ and A

Given an initial (guess at) $x1$, you need a way to compute the area t of the tail for which $x > x1$. For the exponential distribution, this is just e^{-x1} , while for the normal distribution, assuming you are using the unnormalized $f(x) = e^{-x^2/2}$, this is $\sqrt{\pi}/2 \operatorname{erfc}(x/\sqrt{2})$. For more awkward distributions, numerical integration may be required.

With this in hand, from $x1$, you can find $y1 = f(x1)$, the area t in the tail, and the area of the base layer $A = x1y1 + t$.

Then compute the series yi and xi as above. If $yi > f(0)$ for any $i < n$, then the initial estimate $x1$ was too low, leading to too large an area A . If $yn < f(0)$, then the initial estimate $x1$ was too high.

Given this, use a root-finding algorithm (such as the bisection method) to find the value $x1$ which produces $yn-1$ as close to $f(0)$ as possible. Alternatively, look for the value which makes the area of the topmost layer, $xn-1(f(0) - yn-1)$, as close to the desired value A as possible. This saves one evaluation of $f^{-1}(x)$ and is actually the condition of greatest interest.

- George Marsaglia; Wai Wan Tsang (2000). "The Ziggurat Method for Generating Random Variables". Journal of Statistical Software. 5 (8). Retrieved 2007-06-20. This paper numbers the layers from 1 starting at the top, and makes layer 0 at the bottom a special case, while the explanation above numbers layers from 0 at the bottom.
- C implementation of the ziggurat method for the normal density function and the exponential density function, that is essentially a copy of the code in the paper. (Potential users should be aware that this C code assumes 32-bit integers.)
- A C# implementation of the ziggurat algorithm and overview of the method.
- Jurgen A. Doornik (2005). "An Improved Ziggurat Method to Generate Normal Random Samples" (PDF). Nuffield College, Oxford. Retrieved 2007-06-20. Describes the hazards of using the least-significant bits of the integer random number generator to choose the layer number.
- Normal Behavior By Cleve Moler, MathWorks, describing the ziggurat algorithm introduced in MATLAB version 5, 2001.
- The Ziggurat Random Normal Generator Blogs of MathWorks, posted by Cleve Moler, May 18, 2015.
- David B. Thomas; Philip H.W. Leong; Wayne Luk; John D. Villasenor (October 2007). "Gaussian Random Number Generators" (PDF). ACM Computing Surveys. 39 (4): 11:1–38. doi:10.1145/1287620.1287622. ISSN 0360-0300. S2CID 10948255. Retrieved 2009-07-27. [W]hen maintaining extremely high statistical quality is the first priority, and subject to that constraint, speed is also desired, the Ziggurat method will often be the most appropriate choice. Comparison of several algorithms for generating Gaussian random numbers.
- Nadler, Boaz (2006). "Design Flaws in the Implementation of the Ziggurat and Monty Python methods (And some remarks on Matlab randn)". arXiv:math/0603058.. Illustrates problems with underlying uniform pseudo-random number generators and how those problems affect the ziggurat algorithm's output.
- Edrees, Hassan M.; Cheung, Brian; Sandora, McCullen; Nummey, David; Stefan, Deian (13–16 July 2009). Hardware-Optimized Ziggurat Algorithm for High-Speed Gaussian Random Number Generators (PDF). 2009 International Conference on Engineering of Reconfigurable Systems & Algorithms. Las Vegas.
- Marsaglia, George (September 1963). Generating a Variable from the Tail of the Normal Distribution (Technical report). Boeing Scientific Research Labs. Mathematical Note No. 322, DTIC accession number AD0423993. Archived from the original on September 10, 2014 – via Defense Technical Information Center.

Random Number Primes

A side interest I wanted to study is the aspects of prime derivatives from very large information built in both representative of being able to statically change the order of information input as a bit form. For example, within a dense data set, what forms of integer or decimal order introduce the property of prime being removed from a number of them. Given that the data set can be large such as binary, forming the integer can show a small distance to anchor points for known sites. subtraction can be used to remove the shelf nearest. can reduce multiple relationships once an array of sizable information from bit arrangements has been found. The aspects of grouping first as preserved by input of an actual number. Often if not calculated, have very little detail if input from a person. More information is gathered from finding patterns of bit reduction and likeness.

While not being able to invoke the full knowledge of prime, what effect does this have on the binary data? The most interesting would be to find a target input range of bytes to form numbers. With integers, a good storage of them could produce in sorting the identity of range. Negative values are typically represented in bit form as a two's complement while the positive appears appealing to binary natural calculation to base ten. If both numbers appear in a negative or positive fashion often, the set can be reduced to a split buffer of signs. In the format signifies the difference in output. One of the most effective aspects is built on preparing data with nibble. As a range unit of variance with coverage in statistical coverage, a permutation can account for both range sorting, nibble sorting to describe within a buffer of a known size, the huffman selector for stored trees.

The aspects of increasing the sizes between the table dictionary of prime numbers also as an encoder target is referenceable.

Take for example:

35547,35563,35571,35581,35599,35607,35613,35631,35637,35641,35649,35659, 35707,35733,35743,35761,35763,35781,35791,35829,35869,35893,35917,35949, 35953,35959,35973,35977,36001,36003,36027,36039,36067,36069,36073,36093, 36109,36117,36121,36129,36153,36163,36183,36213,36223,36229,36247,36249, 36253,36261,36267,36271,36291,36297,36307,36319,36327,36331,36339,36349, 36351,36363,36367,36369,36391,36411,36459,36471,36493,36499,36513,36531, 36537,36561,36579,36613,36619,36631,36649,36661,36667,36669,36681,36729, 36747,36751,36757,36759,36769,36787,36793,36799,36829,36831,36853,36873, 36877,36883,36913,36921,36943,36951,36957,36979,36991,36993,37041,37053

[Prime Numbers from 1,000,000 to 2,000,000](#)

In a representation of bit range, each of them can be regulated to the base design format of 16 bits. As a binary unsigned unit it is 0 - 65536. If a random prime were introduced as a modulo operator, or multiplier. What about a bit process that can also

be unset. Such as also preserving statistically how off prime range is delta. As a mask binary value, some operations need planning to reverse. This table is a little above the range between data, 32767 is the midpoint.

When prime numbers are used as noise, knowing the exact number can be large if the numerical input were used as a base. As a base, the order of the primes in the set can be removed. Yet having a numerical base present with the system implies leverage in some accounting bytes. So as a range, there are multiple uses of a data set where having a selection of initially golden numbers such as primes offer. Another aspect is finding mathematical relationships to the set from a fixed balancing method.

Is the speed of light or anything in quantum mechanics based upon prime mathematics. If the system were related, as a digital quantum signal, perhaps very dangerous outcomes would occur.

Some tests before reading more, using double precision, examine the prime being divided by the data input. Is the floating point precision, when adjusted by the integer representation of data, selected? Developing further movement of the decimal point to tie it to surrounding data elements as a blurring method.

As a type of hash key, that is stored in order for a dictionary based output its functionality would compare to the SHA1 usage, but much less studied for output. The SHA1 algorithm as storage for dictionaries is key code crackable.

[RSA \(cryptosystem\) - Wikipedia](#)

Two primes can be used for key generation.

Choose two primes from the list and perform a multiplication.

The Carmichael uses the LCM method for coprime relationship data. It is a type of transform known about. Other transforms using prime numbers can also be used. Such as boolean inversion of sign. The show is that an index table can be built from the output. Usage enlarges the problem, while LCM across multiple series of tables can be found. Next the power function is used to attain the much larger number. The fascination is built around invocation of how the algorithm magnifies the data to attain a spread. Where a variable number of arguments can be encoded as a reference lookup. So it acts as an index. It is unknown, yet the aspect of making each a better hash spread seems more effective.

Does the process work on an inversion aspect such as inverting the sign of the number? Or operating on the primes such that the sign is maintained. From the addition of the data and

Using larger integer sizes that are provisioned for in normal programming can be tailored to the bit in principle. Although most effectively. Another aspect is recording the division of prime as a floating point.

[Linear congruential generator - Wikipedia](#)

[Prime number - Wikipedia](#)

[List of prime numbers - Wikipedia](#)

https://rosettacode.org/wiki/Carmichael_lambda_function

https://en.wikipedia.org/wiki/Carmichael_function

[On-Line Encyclopedia of Integer Sequences - Wikipedia](#)

[Bit - Wikipedia](#)

[Entropy \(information theory\) - Wikipedia](#)

[Index of Prime Numbers 2 to 10,000,000](#)

[Truth table - Rosetta Code](#)

Integer Size

The very aspect of every consideration within the Random Numbers chapter shows its position within the Integer Sequences division. Most of the random number algorithms are designed very well with delicate precision. Perhaps plugged-in values were the result of another type of program or study. The case for running the code shows a peak of existing code with mammoth names like Crunch and BigCrunch. The foundation of defining the cycle is found in most definitions of technical breakdown. The extremities of which are making guessing the seed value difficult. It seems shocking how many permutations are given, 2^{64} on many. The possibility of what is known as brute force cracking simply comes down to processing time or reference storage of the entire dataset.

[Randomness test - Wikipedia](#)

[Diehard tests - Wikipedia](#)

C++ 128/256-bit fixed size integer types - Stack Overflow

Below is a C++ implementation for adding two 64-bit numbers with carry. The ability to make the size much larger, comes from kinetic understanding of using multiple engines to produce variable length segments. The ability to define large sets is intrinsic in the Boost implementation. Perhaps after investigating further a limitless supply can be found, per size with the nSquare method. It seems that there are limits perhaps. Yet both seed table and segment table as a bit range would provide a variance system. Perhaps depending upon the nibble.

```
```cpp
#include <iostream>
#include <cstdint>

// Function to add two 64-bit numbers with carry
uint64_t add_with_carry(uint64_t a, uint64_t b) {
 uint64_t sum = a ^ b; // Calculate the sum without carry
 uint64_t carry = (a & b) << 1; // Calculate the carry
 while (carry != 0) {
 uint64_t temp_sum = sum;
 sum = sum ^ carry; // Add the carry to the sum
 carry = (temp_sum & carry) << 1; // Update the carry
 }
 return sum;
}

int main() {
 uint64_t num1 = 12345678901234567;
 uint64_t num2 = 98765432109876543;

 uint64_t result = add_with_carry(num1, num2);

 std::cout << "Sum of " << num1 << " and " << num2 << " is: " << result << std::endl;

 return 0;
}
```

```

This code defines a function `add_with_carry` that takes two 64-bit unsigned integers as input and returns their sum with carry. It uses bitwise XOR and bitwise AND operations to calculate the sum and carry, respectively. The loop continues until there is no carry left.

Below is an implementation of the **Multiply-With-Carry (MWC)** random number generator in C++ for 64-bit integers:

```
```cpp
#include <iostream>
#include <cstdint>

class MWC64 {
private:
 uint64_t x;
 uint64_t c;

public:
 MWC64(uint64_t seed) : x(seed), c(0) {}

 uint64_t next() {
 const uint64_t a = 4294957665; // Multiplier
 const uint64_t m = 18446744073709551615; // Modulus (2^64 - 1)

 x = a * x + c;
 c = x >> 32; // Carry
 x &= m; // Mask to 64 bits

 return x;
 }
};
```

```

```

int main() {
    MWC64 rng(123456789); // Initialize with a seed
    for (int i = 0; i < 10; ++i) {
        std::cout << "Random number " << i + 1 << ":" << rng.next() << std::endl;
    }
    return 0;
}
```

```

In this implementation:

- `x` represents the current state.
- `c` is the carry value.
- The multiplier `a` and modulus `m` are constants chosen for good randomness properties.

4294957665  
0b1111111111111111111101101001100001

18446744073709551615

<https://stackoverflow.com/questions/28868367/getting-the-high-part-of-64-bit-integer-multiplication>

```

uint64_t a_lo = (uint32_t)a;
uint64_t a_hi = a >> 32;
uint64_t b_lo = (uint32_t)b;
uint64_t b_hi = b >> 32;

uint64_t a_x_b_hi = a_hi * b_hi;
uint64_t a_x_b_mid = a_hi * b_lo;
uint64_t b_x_a_mid = b_hi * a_lo;
uint64_t a_x_b_lo = a_lo * b_lo;

uint64_t carry_bit = ((uint64_t)(uint32_t)a_x_b_mid +
 (uint64_t)(uint32_t)b_x_a_mid +
 (a_x_b_lo >> 32)) >> 32;

uint64_t multhi = a_x_b_hi +
 (a_x_b_mid >> 32) + (b_x_a_mid >> 32) +
 carry_bit;

return multhi;

```

## Floating Point Size

The floating point can be represented by not storing each component within the structure. As a memory location, often if eight are together, an index can be used to store the sign bit packed. As well the exponent and numerical portions. The ability to handle a limitless beyond current hardware to functional math at dynamic precision can be useful is tailored in its implementation. The implications are very system draining for calculations perhaps being eight to ten times slower. Or it is unknown, perhaps multiprocessing can solve some issues.

## Boost multiprecision

Yet as the depth of the computer supplies the machine to accomplish this, how are we going to make random number guessing much more difficult? The very aspect of size has a major tradition to it. Many times in the past have these works been possible, and today the dynamic math system of integer and floating point bits can be instantiated. There are several working systems that add on features. Boost for example has these capabilities already debugged.

[https://www.boost.org/doc/libs/1\\_73\\_0/libs/multiprecision/doc/html/index.html](https://www.boost.org/doc/libs/1_73_0/libs/multiprecision/doc/html/index.html)

```
#include <cstdint>
```

```

#include <iomanip>
#include <iostream>

#include <boost/multiprecision/cpp_int.hpp>

using uint512_t = boost::multiprecision::number<boost::multiprecision::cpp_int_backend<512, 512,
boost::multiprecision::unsigned_magnitude, boost::multiprecision::unchecked, void>,
boost::multiprecision::et_off>;
using uint1024_t = boost::multiprecision::number<boost::multiprecision::cpp_int_backend<1024, 1024,
boost::multiprecision::unsigned_magnitude, boost::multiprecision::unchecked, void>,
boost::multiprecision::et_off>;
using uint2048_t = boost::multiprecision::number<boost::multiprecision::cpp_int_backend<2048, 2048,
boost::multiprecision::unsigned_magnitude, boost::multiprecision::unchecked, void>,
boost::multiprecision::et_off>;
using uint4096_t = boost::multiprecision::number<boost::multiprecision::cpp_int_backend<4096, 4096,
boost::multiprecision::unsigned_magnitude, boost::multiprecision::unchecked, void>,
boost::multiprecision::et_off>;
using uint8192_t = boost::multiprecision::number<boost::multiprecision::cpp_int_backend<8192, 8192,
boost::multiprecision::unsigned_magnitude, boost::multiprecision::unchecked, void>,
boost::multiprecision::et_off>;

int main()
{
 uint8192_t u(2U);

 for (int64_t i = 0; i < 4096 - 1; i++) {
 u *= 2;
 }

 std::cout << u << std::endl;
}

```

[C++ 128/256-bit fixed size integer types - Stack Overflow](#)

<https://oeis.org/>

## Numerical Algorithms

Generalized methods of solving nonlinear equations is a type of reversal aspect that provides the use of numerical sequences. The ability for the equation to work appropriately means that only input and output are functional. The ability to adaptively spread data, and leave major portions out lies in obscuring the data tactically can be an integrated function. By having a very few sets of basic routines, the process of geometry decomposition can take place based upon ratios, angle, missing variables in calculus, ln(), and functional.

That is hash coding lookup using power exponentials, random numbers, and prime. Radians or degrees can also take part. The ability to handle shapes in a relationship using these functions as a limiter and range guidance is configured already by the cartesian\_t class that communicatively receives a locked portion of the bit\_t primary array,

Ultimately multiple instances must run on an array of buffers simultaneously. The ability to granular expose functions perhaps could be template oriented inside. Yet the mechanism traversal and logic to be studied on a permutation capability also needs adaptive coding.

The logic of the RSA encryption uses the LCM roots of two prime numbers. The summary key is derived from a power exponential function Carmichael. Simply the system implements a cache of the from a known series or seed. So in essence the power function and catalog of the numbers from the function provide an index, or hash. The algorithm tuned for AVX assembly or SSE, can be systematically appealing. Incorporating hardware encryption is a type of vacuum, but ultimately many in the consumer field have been superseded by software. Some chip manufacturers such as Intel and AMD place specialized instructions for these works that speed up network communications.

In reference to one of the greatest finds for my reading is the text <https://numerical.recipes/book.html>. In its chapters are amazing catalogs of things I need to learn. Yet with the Internet and copy paste, the text is great. Take a glance at the chapter on root finding. It mentions several algorithms. I will search around for existing implementations. This is high school

introductions for some, others have continued at the college level. Yet the precise method of how to use the basic form of the code is important. This will develop better methods of easy and also difficult work.

- [New Calculus Video Playlist - YouTube](#)
- [Quadratic Equation | Roots, Formula and Examples](#)
- [▶ Derivatives of Parametric Functions](#)
- [▶ Power Series Representation By Integration - Calculus 2](#)

The aspects of using the formulas everyday are not relevant for many. The video play list above can refresh your memory, some of which are already seized in code. Yet the possibility of knowing the proximity of these subjects in a video field seems confident. I plan on watching several myself.

### [Secant of a Circle: Definition, Formula, Theorems, and Properties](#)

The ability to provide an array and variance of information field projection to a plane provides exclusive cracking rights when appropriate transforms are used. Multiple systems employ the use of the checkerboard shuffle, as a necessity permutation by uncracking AI based encryption is based upon a dictionary. By introducing the secant method, and developing SHA1 HASH codes to represent the previous uses of the data for checksum a client may be remembered.

```
/*
A genetically produced sample. Template oriented designs can work for numerical
types as below.

*/
function IFJJsddYnsstheterer_2332344342344(bits_t &bits) {

 bits.circle().subdivide(rnd)
 .for_each(rnd, [&o] {
 o.secant(rnd,rnd)
 .bitswap(distance(rnd)),distance(rnd));}
).permutate();

 auto olarge_circles = bits.circle().subdivide(rnd_small);

 for(auto olc : olarge_circles) {
 olc.for([&n] {
 auto olist=n.triangle().subdivide(rnd);
 olist.bitswap(radial_xor());
 })
 olc.triangle().subdivide(rnd_medium).bitswap(distance,rnd);
 olc << mix(noise(.1,rnd));
 }

 // apply finalizes the transaction
 bits.lock().apply();

}
```

The bits locking mechanism stores directory information and key data at once. This is data that is needed for decoding while using the representation of the vector tree. It supplies the functional work of mixing and sorting the data. Perhaps arrangement into bytes and a log file showing statistics of the output can work. The work at the integer level in positioning by the object system stores the information needed to apply in logic elsewhere.

As a queue of transforms, with perhaps other elements such as a sliding window on variation, an inner stack, looping, and mixing, the apply() class method can be the position in which the algorithm processing takes place for that problem. The items are stored in a queue. The ability to program the queue dictionary using a bit secret scheme.

asymptotically

[ə'sem'tädək(ə)lē, asim(p)'tötik(ə)lē]

definition

adverb form of asymptote

as·ymp·tote

[əs'əm(p), tōt]

noun

a line that continually approaches a given curve but does not meet it at any finite distance.

Wikipedia

Organizing and summarizing data is called descriptive statistics. Two ways to summarize data are by graphing and by using numbers, for example, finding an average. After you have studied probability and probability distributions, you will use formal methods for drawing conclusions from good data. The formal methods are called inferential statistics. Statistical inference uses probability to determine how confident we can be that our conclusions are correct.

Effective interpretation of data, or inference, is based on good procedures for producing data and thoughtful examination of the data. You will encounter what will seem to be too many mathematical formulas for interpreting data. The goal of statistics is not to perform numerous calculations using the formulas, but to gain an understanding of your data. The calculations can be done using a calculator or a computer. The understanding must come from you. If you can thoroughly grasp the basics of statistics, you can be more confident in the decisions you make in life.

### Statistical Models

Statistics, like all other branches of mathematics, uses mathematical models to describe phenomena that occur in the real world. Some mathematical models are deterministic. These models can be used when one value is precisely determined from another value. Examples of deterministic models are the quadratic equations that describe the acceleration of a car from rest or the differential equations that describe the transfer of heat from a stove to a pot. These models are quite accurate and can be used to answer questions and make predictions with a high degree of precision. Space agencies, for example, use deterministic models to predict the exact amount of thrust that a rocket needs to break away from Earth's gravity and achieve orbit.

However, life is not always precise. While scientists can predict to the minute the time that the sun will rise, they cannot say precisely where a hurricane will make landfall. Statistical models can be used to predict life's more uncertain situations. These special forms of mathematical models or functions are based on the idea that one value affects another value. Some statistical models are mathematical functions that are more precise—one set of values can predict or determine another set of values. Or some statistical models are mathematical functions in which a set of values does not precisely determine other values. Statistical models are very useful because they can describe the probability or likelihood of an event occurring and provide alternative outcomes if the event does not occur. For example, weather forecasts are examples of statistical models. Meteorologists cannot predict tomorrow's weather with certainty. However, they often use statistical models to tell you how likely it is to rain at any given time, and you can prepare yourself based on this probability.

### Probability

Probability is a mathematical tool used to study randomness. It deals with the chance of an event occurring. For example, if you toss a fair coin four times, the outcomes may not be two heads and two tails. However, if you toss the same coin 4,000 times, the outcomes will be close to half heads and half tails. The expected theoretical probability of heads in any one toss is  $\frac{1}{2}$

or .5. Even though the outcomes of a few repetitions are uncertain, there is a regular pattern of outcomes when there are many repetitions. After reading about the English statistician Karl Pearson who tossed a coin 24,000 times with a result of 12,012 heads, one of the authors tossed a coin 2,000 times. The results were 996 heads. The fraction  $\frac{996}{2000}$  is equal to .498 which is very close to .5, the expected probability.

The theory of probability began with the study of games of chance such as poker. Predictions take the form of probabilities. To predict the likelihood of an earthquake, of rain, or whether you will get an A in this course, we use probabilities. Doctors use probability to determine the chance of a vaccination causing the disease the vaccination is supposed to prevent. A stockbroker uses probability to determine the rate of return on a client's investments.

## [1.1 Definitions of Statistics, Probability, and Key Terms](#)

So glad I can quote the coin tosses, that a lot of nickels. Perhaps he did it over the course of time as a routine. Or as an extra chore at the fridge. Perhaps this is where the 99 beers on the wall song creation took place. Who has 24,000 coin toss beers without being arrested? Yet using statistics, or sensor readings from a binary database allows research and problem solving. Typically more advanced programmers have employed several new techniques as seen as the science of the underdog initially. AI and system private data can work around problems, more on this later.

Log Files, API, Symbol points, time measurement and bits processed are typical of a genetic algorithm. There are many things that can be gathered from processing these codes. Measuring and recording operations is a type of organization that an AI system must use as its information base. To decide what perfect encryption is as a type of stack could be limitless in its pursuit. Yet the ability to analyze the performance and also balance information is necessary. Often understanding the smallest forms first.

# Diehard tests

## Overview

### Asymptotic distribution

In mathematics and statistics, an asymptotic distribution is a probability distribution that is in a sense the "limiting" distribution of a sequence of distributions. One of the main uses of the idea of an asymptotic distribution is in providing approximations to the cumulative distribution functions of statistical estimators.

#### Definition

A sequence of distributions corresponds to a sequence of random variables  $Z_i$  for  $i = 1, 2, \dots, I$ . In the simplest case, an asymptotic distribution exists if the probability distribution of  $Z_i$  converges to a probability distribution (the asymptotic distribution) as  $i$  increases: see convergence in distribution. A special case of an asymptotic distribution is when the sequence of random variables is always zero or  $Z_i = 0$  as  $i$  approaches infinity. Here the asymptotic distribution is a degenerate distribution, corresponding to the value zero.

However, the most usual sense in which the term asymptotic distribution is used arises where the random variables  $Z_i$  are modified by two sequences of non-random values. Thus if converges in distribution to a non-degenerate distribution for two sequences  $\{a_i\}$  and  $\{b_i\}$  then  $Z_i$  is said to have that distribution as its asymptotic distribution. If the distribution function of the asymptotic distribution is  $F$  then, for large  $n$ , the following approximations hold

If an asymptotic distribution exists, it is not necessarily true that any one outcome of the sequence of random variables is a convergent sequence of numbers. It is the sequence of probability distributions that converges.

#### Central limit theorem

Main article: Central limit theorem

Perhaps the most common distribution to arise as an asymptotic distribution is the normal distribution. In particular, the central limit theorem provides an example where the asymptotic distribution is the normal distribution.

#### Central limit theorem

Suppose

The central limit theorem gives only an asymptotic distribution. As an approximation for a finite number of observations, it provides a reasonable approximation only when close to the peak of the normal distribution; it requires a very large number of observations to stretch into the tails.

#### Local asymptotic normality

Main article: Local asymptotic normality

Local asymptotic normality is a generalization of the central limit theorem. It is a property of a sequence of statistical models, which allows this sequence to be asymptotically approximated by a normal location model, after a rescaling of the parameter. An important example when the local asymptotic normality holds is in the case of independent and identically distributed sampling from a regular parametric model; this is just the central limit theorem.

Barndorff-Nielson & Cox provide a direct definition of asymptotic normality.[2]

#### See also

- Asymptotic analysis
- Asymptotic theory (statistics)
- de Moivre–Laplace theorem
- Limiting density of discrete points
- Delta method
- References
- Billingsley, Patrick (1995). *Probability and Measure* (Third ed.). John Wiley & Sons. p. 357. ISBN 0-471-00710-2.
- Barndorff-Nielsen, O. E.; Cox, D. R. (1989). *Asymptotic Techniques for Use in Statistics*. Chapman and Hall. ISBN 0-412-31400-2.

The diehard tests are a battery of statistical tests for measuring the quality of a random number generator. They were developed by George Marsaglia over several years and first published in 1995 on a CD-ROM of random numbers.[1] In 2006, the original diehard tests were extended into the dieharder tests.[2]

#### History

An initial battery of randomness tests for RNGs was suggested in the 1969 first edition of *The Art of Computer Programming* by Donald Knuth (Volume 2, Chapter 3.3: Statistical tests). Knuth's tests were then supplanted by George Marsaglia's Diehard

tests (1996) consisting of fifteen different tests. The inability to modify the test parameters or add new tests led to the development of the TestU01 library, introduced in 2007 by Pierre L'Ecuyer and Richard Simard of the Université de Montréal.

## Birthday spacings

Choose random points on a large interval. The spacings between the points should be asymptotically exponentially distributed.[3] The name is based on the birthday paradox.

## Overlapping permutations

Analyze sequences of five consecutive random numbers. The 120 possible orderings should occur with statistically equal probability.

## Ranks of matrices

Select some number of bits from some number of random numbers to form a matrix over  $\{0,1\}$ , then determine the rank of the matrix. Count the ranks.

## Monkey tests

Treat sequences of some number of bits as "words". Count the overlapping words in a stream. The number of "words" that do not appear should follow a known distribution. The name is based on the infinite monkey theorem.

## Count the 1s

Count the 1 bits in each of either successive or chosen bytes. Convert the counts to "letters", and count the occurrences of five-letter "words".

## Parking lot test

Randomly place unit circles in a  $100 \times 100$  square. A circle is successfully parked if it does not overlap an existing successfully parked one. After 12,000 tries, the number of successfully parked circles should follow a certain normal distribution.

## Minimum distance test

Randomly place 8000 points in a  $10000 \times 10000$  square, then find the minimum distance between the pairs. The square of this distance should be exponentially distributed with a certain mean.

## Random spheres test

Randomly choose 4000 points in a cube of edge 1000. Center a sphere on each point, whose radius is the minimum distance to another point. The smallest sphere's volume should be exponentially distributed with a certain mean.

## The squeeze test

Multiply 231 by random floats on  $(0,1)$  until you reach 1. Repeat this 100000 times. The number of floats needed to reach 1 should follow a certain distribution.

## Overlapping sums test

Generate a long sequence of random floats on  $(0,1)$ . Add sequences of 100 consecutive floats. The sums should be normally distributed with characteristic mean and variance.

## Runs test

Generate a long sequence of random floats on  $(0,1)$ . Count ascending and descending runs. The counts should follow a certain distribution.

## The craps test

Play 200000 games of craps, counting the wins and the number of throws per game. Each count should follow a certain distribution.

## Details

### The birthday spacings test

Choose  $n$  birthdays in a year of  $n$  days. List the spacings between the birthdays. If  $j$  is the number of values that occur more than once in that list, then  $j$  is asymptotically Poisson-distributed with mean  $m_3 / (4n)$ . Experience shows  $n$  must be quite large, say  $n \geq 218$ , for comparing the results to the Poisson distribution with that mean. This test uses  $n = 224$  and  $m = 29$ , so that the underlying distribution for  $j$  is taken to be Poisson with  $\lambda = 227 / 226 = 2$ . A sample of 500  $j$ 's is taken, and a chi-square goodness of fit test provides a  $p$  value. The first test uses bits 1–24 (counting from the left) from integers in the specified file. Then the file is closed and reopened. Next, bits 2–25 are used to provide birthdays, then 3–26 and so on to bits 9–32. Each set of bits provides a  $p$ -value, and the nine  $p$ -values provide a sample for a KTEST.

## The overlapping 5-permutation test

This is the OPERM5 test. It looks at a sequence of one million 32-bit random integers. Each set of five consecutive integers can be in one of 120 states, for the  $5!$  possible orderings of five numbers. Thus the 5th, 6th, 7th, ... numbers each provide a state. As many thousands of state transitions are observed, cumulative counts are made of the number of occurrences of each state. Then the quadratic form in the weak inverse of the  $120 \times 120$  covariance matrix yields a test equivalent to the likelihood ratio test that the 120 cell counts came from the specified (asymptotically) normal distribution with the specified  $120 \times 120$  covariance matrix (with rank 99). This version uses 1000000 integers, twice. This test may have unresolved bugs resulting in consistently poor p-values.[4]

## The binary rank test for $31 \times 31$ matrices

The leftmost 31 bits of 31 random integers from the test sequence are used to form a  $31 \times 31$  binary matrix over the field  $\{0,1\}$ . The rank is determined. That rank can be from 0 to 31, but ranks  $< 28$  are rare, and their counts are pooled with those for rank 28. Ranks are found for 40000 such random matrices and a chi-square test is performed on counts for ranks 31, 30, 29 and  $\leq 28$ .

## The binary rank test for $32 \times 32$ matrices

A random  $32 \times 32$  binary matrix is formed, each row a 32-bit random integer. The rank is determined. That rank can be from 0 to 32, ranks less than 29 are rare, and their counts are pooled with those for rank 29. Ranks are found for 40000 such random matrices and a chi square test is performed on counts for ranks 32, 31, 30 and  $\leq 29$ .

## The binary rank test for $6 \times 8$ matrices

From each of six random 32-bit integers from the generator under test, a specified byte is chosen, and the resulting six bytes form a  $6 \times 8$  binary matrix whose rank is determined. That rank can be from 0 to 6, but ranks 0, 1, 2, 3 are rare; their counts are pooled with those for rank 4. Ranks are found for 100000 random matrices, and a chi square test is performed on counts for ranks 6, 5 and  $\leq 4$ .

## The bitstream test

The file under test is viewed as a stream of bits. Call them  $b_1, b_2, \dots$ . Consider an alphabet with two "letters", 0 and 1, and think of the stream of bits as a succession of 20-letter "words", overlapping. Thus the first word is  $b_1b_2\dots b_{20}$ , the second is  $b_2b_3\dots b_{21}$ , and so on. The bitstream test counts the number of missing 20-letter (20-bit) words in a string of 221 overlapping 20-letter words. There are 220 possible 20-letter words. For a truly random string of 221 + 19 bits, the number of missing words  $j$  should be (very close to) normally distributed with mean 141,909 and sigma 428. Thus  $(j - 141909) / 428$  should be a standard normal variate (z score) that leads to a uniform  $[0,1]$  p value. The test is repeated twenty times.

## The tests OPSO, OQSO and DNA

OPSO means overlapping-pairs-sparse-occupancy. The OPSO test considers 2-letter words from an alphabet of 1024 letters. Each letter is determined by a specified ten bits from a 32-bit integer in the sequence to be tested. OPSO generates 221 (overlapping) 2-letter words (from 221 + 1 "keystrokes") and counts the number of missing words – that is 2-letter words which do not appear in the entire sequence. That count should be very close to normally distributed with mean 141909, sigma 290. Thus  $(\text{missing wrds} - 141909) / 290$  should be a standard normal variable. The OPSO test takes 32 bits at a time from the test file and uses a designated set of ten consecutive bits. It then restarts the file for the next designated 10 bits, and so on. OQSO means overlapping-quadruples-sparse-occupancy. The test OQSO is similar, except that it considers 4-letter words from an alphabet of 32 letters, each letter determined by a designated string of 5 consecutive bits from the test file, elements of which are assumed 32-bit random integers. The mean number of missing words in a sequence of 221 four-letter words, (221 + 3 "keystrokes"), is again 141909, with sigma = 295. The mean is based on theory; sigma comes from extensive simulation. The DNA test considers an alphabet of 4 letters C, G, A, T, determined by two designated bits in the sequence of random integers being tested. It considers 10-letter words, so that as in OPSO and OQSO, there are 228 possible words, and the mean number of missing words from a string of 221 (overlapping) 10-letter words (221 + 9 "keystrokes") is 141909. The standard deviation sigma = 339 was determined as for OQSO by simulation. (Sigma for OPSO, 290, is the true value (to three places), not determined by simulation).

## The count-the-1's test on a stream of bytes

Consider the file under test as a stream of bytes (four per 32-bit integer). Each byte can contain from none to eight 1s, with probabilities 1, 8, 28, 56, 70, 56, 28, 8, 1 over 256. Now let the stream of bytes provide a string of overlapping 5-letter words, each "letter" taking values A, B, C, D, E. The letters are determined by the number of 1s in a byte 0, 1, or 2 yield A, 3 yields B, 4 yields C, 5 yields D and 6, 7 or 8 yield E. Thus we have a monkey at a typewriter hitting five keys with various probabilities (37, 56, 70, 56, 37 over 256). There are 55 possible 5-letter words, and from a string of 256000 (overlapping) 5-letter words, counts are made on the frequencies for each word. The quadratic form in the weak inverse of the covariance matrix of the cell counts provides a chisquare test Q5–Q4, the difference of the naive Pearson sums of  $(\text{OBS}-\text{EXP})^2 / \text{EXP}$  on counts for 5- and 4-letter cell counts.

## The count-the-1's test for specific bytes

Consider the file under test as a stream of 32-bit integers. From each integer, a specific byte is chosen, say the leftmost bits 1 to 8. Each byte can contain from 0 to 8 1s, with probabilities 1, 8, 28, 56, 70, 56, 28, 8, 1 over 256. Now let the specified bytes from successive integers provide a string of (overlapping) 5-letter words, each "letter" taking values A, B, C, D, E. The letters are determined by the number of 1s, in that byte 0, 1, or 2 → A, 3 → B, 4 → C, 5 → D, and 6, 7 or 8 → E. Thus we have a monkey at a typewriter hitting five keys with various probabilities 37, 56, 70, 56, 37 over 256. There are 55 possible 5-letter words, and from a string of 256000 (overlapping) 5-letter words, counts are made on the frequencies for each word. The quadratic form in the weak inverse of the covariance matrix of the cell counts provides a chisquare test Q5 – Q4, the difference of the naive Pearson sums of  $(OBS - EXP)^2 / EXP$  on counts for 5- and 4-letter cell counts.

### The parking lot test

In a square of side 100, randomly "park" a car – a circle of radius 1. Then try to park a 2nd, a 3rd, and so on, each time parking "by ear". That is, if an attempt to park a car causes a crash with one already parked, try again at a new random location. (To avoid path problems, consider parking helicopters rather than cars.) Each attempt leads to either a crash or a success, the latter followed by an increment to the list of cars already parked. If we plot n: the number of attempts, versus k the number successfully parked, we get a curve that should be similar to those provided by a perfect random number generator. Theory for the behavior of such a random curve seems beyond reach, and as graphics displays are not available for this battery of tests, a simple characterization of the random experiment is used: k, the number of cars successfully parked after n = 12000 attempts. Simulation shows that k should average 3523 with sigma 21.9 and is very close to normally distributed. Thus  $(k - 3523) / 21.9$  should be a standard normal variable, which, converted to a uniform variable, provides input to a KTEST based on a sample of 10.

### The minimum distance test

It does this 100 times choose n = 8000 random points in a square of side 10000. Find d, the minimum distance between the  $(n^2 - n) / 2$  pairs of points. If the points are truly independent uniform, then d<sup>2</sup>, the square of the minimum distance should be (very close to) exponentially distributed with mean 0.995. Thus  $1 - \exp(-d^2 / 0.995)$  should be uniform on [0,1) and a KTEST on the resulting 100 values serves as a test of uniformity for random points in the square. Test numbers = 0 mod 5 are printed but the KTEST is based on the full set of 100 random choices of 8000 points in the 10000×10000 square.

### The 3D spheres test

Choose 4000 random points in a cube of edge 1000. At each point, center a sphere large enough to reach the next closest point. Then the volume of the smallest such sphere is (very close to) exponentially distributed with mean  $120\pi / 3$ . Thus the radius cubed is exponential with mean 30. (The mean is obtained by extensive simulation). The 3D spheres test generates 4000 such spheres 20 times. Each min radius cubed leads to a uniform variable by means of  $1 - \exp(-r^3 / 30)$ , then a KTEST is done on the 20 p-values.

### The squeeze test

Random integers are floated to get uniforms on [0,1). Starting with k = 231 = 2147483648, the test finds j, the number of iterations necessary to reduce k to 1, using the reduction k = ceiling(k×U), with U provided by floating integers from the file being tested. Such js are found 100000 times, then counts for the number of times j was ≤ 6, 7, ..., 47, ≥ 48 are used to provide a chi-square test for cell frequencies.

### The overlapping sums test

Integers are floated to get a sequence U(1), U(2), ... of uniform [0,1) variables. Then overlapping sums, S(1) = U(1) + ... + U(100), S(2) = U(2) + ... + U(101), ... are formed. The Ss are virtually normal with a certain covariance matrix. A linear transformation of the Ss converts them to a sequence of independent standard normals, which are converted to uniform variables for a KTEST. The p-values from ten KTESTs are given still another KTEST.

### The runs test

It counts runs up, and runs down, in a sequence of uniform [0,1) variables, obtained by floating the 32-bit integers in the specified file. This example shows how runs are counted: 0.123, 0.357, 0.789, 0.425, 0.224, 0.416, 0.95 contains an up-run of length 3, a down-run of length 2 and an up-run of (at least) 2, depending on the next values. The covariance matrices for the runs-up and runs-down are well known, leading to chi-square tests for quadratic forms in the weak inverses of the covariance matrices. Runs are counted for sequences of length 10000. This is done ten times. Then repeated.

### The craps test

It plays 200000 games of craps, finds the number of wins and the number of throws necessary to end each game. The number of wins should be (very close to) a normal with mean 200000p and variance 200000p(1 – p), with p = 244 / 495. Throws necessary to complete the game can vary from 1 to infinity, but counts for all > 21 are lumped with 21. A chi-square test is made on the no.-of-throws cell counts. Each 32-bit integer from the test file provides the value for the throw of a die, by floating to [0,1), multiplying by 6 and taking 1 plus the integer part of the result.

Most of the tests in DIEHARD return a p-value, which should be uniform on [0,1) if the input file contains truly independent random bits. Those p-values are obtained by  $p = F(X)$ , where F is the assumed distribution of the sample random variable X – often normal. But that assumed F is just an asymptotic approximation, for which the fit will be worst in the tails. Thus you should not be surprised with occasional p-values near 0 or 1, such as 0.0012 or 0.9983. When a bit stream really FAILS BIG, you will get ps of 0 or 1 to six or more places. Since there are many tests, it is not unlikely that a  $p < 0.025$  or  $p > 0.975$  means that the RNG has "failed the test at the 0.05 level". We expect a number of such events ps happen among the hundreds of events DIEHARD produces, even conditioned on the random number generator being perfect.

See also

[Randomness test](#)  
[TestU01](#)

References

- "The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness". Florida State University. 1995. Archived from the original on 2016-01-25.
- Brown, Robert G. "dieharder". Retrieved 2023-09-25.
- Renyi, 1953, p194
- "Robert G. Brown's General Tools Page". Archived from the original on 2017-07-03.
- Further reading
- Rényi, Alfréd (1953). "On the theory of order statistics". *Acta Mathematica Academiae Scientiarum Hungaricae*. 4 (3–4): 191–231. doi:10.1007/BF02127580.
- Vigna, Sebastiano (July 2016). "An experimental exploration of Marsaglia's xorshift generators, scrambled" (PDF). *ACM Transactions on Mathematical Software*. 42 (4): 30. arXiv:1402.6246. doi:10.1145/2845077. S2CID 13936073.
- O'Neill, Melissa E. (5 September 2014). PCG: A Family of Simple Fast Space-Efficient Statistically Good Algorithms for Random Number Generation (PDF) (Technical report). Harvey Mudd College. HMC-CS-2014-0905.
- External links
- "The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness". Florida State University. 1995. Archived from the original on 2016-01-25.
- Robert G. Brown. "Dieharder: A Random Number Test Suite"

[Diehard tests - Wikipedia](#)

[Robert G. Brown's General Tools Page](#)

# Bit Ball AI PRGN



As a conglomeration of aspects that may be useful, here are my own thoughts. The interesting aspects of the PRNG is a selection of a tap, or a new spread method. The tap process can also be treated to utilize variances from private secure information to employ boolean sequence variance. Typically reversing the tap direction or rotation can also provide better results.

The basis of `bit_ball_rpng_t` template class being an object scheme comes from research and many advanced people, inventors in the field. You will have found their links listed about these texts. Most of the establishment of their contexts also is within the bit pattern range. The numerical and also statistical modifier in the range of bits implies functional usage. The triggers of multiplication, and also balance of  $N^2$  is a competitive leverage. The functional aspects of many of the preceding systems are incorporated. This system may be slower, yet it also encompasses many other forms of input and output. More processing is established in simple AI using boolean tables and associating the aspect with logical operations that are recorded. A

running hash and stack of hashes, provide escaping the tests possibly by the system being coded for the circumstance. So strategically the tests made should be established as a possible study, or overview.

One of the most competitive aspects of the Bit Ball AI PRGN is its birth random. As a much larger entity, the process of system cycle spin provides that encryption must be very keen on this data. Perhaps the best place would be embedded within the source code hard coded. Sensor data of client execution times provide the ability to a tailored stack for the device as a requested feature. When data is sent, it is linked to these routines. The amount of server and client data is established from both sides while stenographic balancing of a cache is maintained. A measure is provided on both sides for logical usage.

Integer carry mechanics for base reference points can be implemented in C++ to expand the input and output of data. Both through arithmetic carry flag operation and other attributes. It does this more easily in machine code. Yet the operation of expansion produces more memory of the data type. Therefore larger random databases that are private are more secure. The ability to strategically strengthen the client and server through multiinput stenographic information while on other channels increases the strength.

The ability to accept multiple types of seeds within the system must be present. As a random data generator of a type of sequence, the user may also want to imply banging on the keyboard, or even using audio input for the largest ever random number. The user could choose from a numeric list of input. Enter a huge number, letters, or gibberish. This along with the file of gibberish and data, always produce a significant amount of randomness. A user could listen to their favorite song as part of the initialization. Unfortunately the primary seed is a much smaller very large number when scaled to useful size. The boolean truth table allows an output logic to exist in a matrix. As an array of potential candidates with invocation rules, and instantiation of multiple. The aspects are that bits represent these boolean values. Yet often modifying them must be more precise for logical operations of turning them on and off. These are known as bit flags, bit ranges, and bitmaps in C++.

The ability to skip based upon historical concepts of network usage implies that genetic growth is maintained. The amount and usefulness of recorded keys based upon the hash checksum lookup table as numerical input. The input is a bit mix on the first file in the package directory. This means that the entire package format must be decoded to use. In some cases, the indexing of these files with a hash algorithm such as sha1 can become dictionary entries of textual information. By referencing the contexts, linking to a paragraph can occur. As well, paragraphs, preceding, and following. Having a numerical entry to dictionaries can operate on an index base to form with indexing. Redefining the nibble set, based upon parameters of each paragraph. The index would be SHA1, word, character. So logging any data is also sensitive and very data consuming for modern pc processing without the proper software.

The system must gradually send through its stenographic layer, information that makes the server aware of these privatized random number producers. Each stenographic moment of communication to the server as per usage makes the signal strength stronger. By varying the amount of stenographic information to a real time facet you could perhaps leverage to a specific range of time. Such as a five year plan for example. Some devices do not last that long but some do. So unless every seed of the stenographic data engine has been gathered, it can be reutilized. Physical server protection and all network communications operations on a known key intelligence arrival frequency or other methods.

So the system will have genetically produced several systems. The system never selects a symmetrical approach, utilizing its own strengths of platform and private cache. The server alignment process must be monitored during the production of genetically created data during the stenographic upload. The data describes the random number database for client applications. Types of user input can drive the unit in more ways than smaller data.

Allowing for booting the random number to a dynamic strength, versus birthing is an option. The research would be to utilize a modifier subset for these operations using the given design method of C++ template classes. By interlacing the technology with adaptor style combinations, permutations may change to more distinct data sets and application referenced implementation. A system that has multiple variations in the amount of random data that it uses can have a distinct plug in feature that is dedicated to more of an object system style interface. Information statistical access and models of genetic models may also be informed through the tightest bit spread for a span of time. Data does arrive on networks in the second realm. That is, many stenographic bits can cross in minutes of use. The network application usage and tailored remote repository content management with integrated local software support all utilizing the browser base code for its vastness in solved problems. Or Nebula OS model with a nice easier stream processing.

Representing numerical information in random base textual output increases the amount of data. By modifying the item as a log function, it is no longer a binary unit. The ability to enforce this representation can take the form to impact the data data as a package with also noise and a digital inside password. By supplying a chain package layer, many prerequisites can be found to instruct the decoder further. As a large random key that becomes a noise layer proportional to the data size.

Representing random global data is also a very important aspect. The GUID is claimed to be but summarizes this in a few bytes. As a generator, this can be established using automatic methods based upon any sensor input, such as taking a picture or using a bit\_ball\_key\_t puzzle game. Grabbing the list of wireless server names if any are around. The GPS, or last known. recorded user summarized data of experience yet hashed for imposed identity strength.

Random data based on machine environmental knowledge can also be leveraged as a genetic encryption fold. The software develops permutations that are based upon more information. This also allows the expansion to also be more unique if the software is functioning properly as by design. If data expansion does not occur as an initial step, phasing the machine identity data using private cache, for stenographic balancing may also be functional. Data expansion or using the various parts as seeding keys for growth is a path.

The implications of iterating upon a known set have to be carried forth through also reversal and continuation of the sequence in a geometric and modulo fashion to remain in boundary. That is the previous tap functionality. Simply the result has to be large, unique, and well distributed globally. Data expansion can also occur from the root source information to provide more of it. As a platform the OS and bit\_ball\_rpng\_t have provisions that make the device tailored for a group of genetic encryption algorithms. Perhaps as a method, the stream limited to an amount such as 16 megabytes makes its plausible identity very large in the numerical range, even at Boost's maximum size.

The aspects of a random boolean or a bit sequence is necessary. To represent the input in a dynamic way, provides a depth to being more random as long as the key is still provided. One aspect of nibble data, matrix operations, is that inversion is a function that may be unaccomplished. By inverting on a random nibble developed from the seed chain, other sized efforts may take place.

The essence is that the server software and subsequent communication can be guarded as per output of any communication. The client has the option to develop a partition between connection sites for a leverage of how often sites communication is established in the protocol. Both platforms receive a time period for stenographic allotment. The realm of the amount of data to show within permutation of shared data. Noise algorithm production can increase in bursts for time periods between the two to strengthen information during other usages. For example image receiving one byte of cypher information while looking at other data from comedy sites. As a server process, this provides the opportunity to do so even within the existing nibble dictionary. Then the entire package decrypted shows one bit of information within a nibblized output, by random nibble selection across the entire package. The decoder reads this and adds this to the code communicated storage.

The first aspect of a random 1 or 0 being selected is that the permutation of the entire sequence be requested for as a dynamic request to ensure that variance is reflected in the output. A random system that is dynamic accounts for its strength in the long term output of these two values. Within a known amount of request range for encryption usefulness lies in the impact of the use. Often if a set of operations are based upon random numbers, the fitting for possible reversal lies within a permutation range. That is if the permutation is found. An engine that may use all input as random data can be coded as a facility to develop more from this through alternate permutations. If two repositories are connected they could be updated. As a network, intelligence information of growing bits.

Multi-seed Boolean Skip Chain PRNG is a type of chain mechanism used by the AI. The multiseed skip chain offers a matrix style interface for activation of plugin connections. The system applies the instrumentation and metrics of the use of machine learning capabilities. Yet it does not have a memory unit of storage. The basis of the architecture is that the system of usefulness can be manufactured as a genetic template process for precise and also expansive design. The basis of symbolic definition and measure are provided in this layer as inheritance.

The aspects of history, storage, intelligence, process grid, alignment protocols, alignment operation database, and network communications. Building a core fabrication using C++ template programming is a task using genetic programming. The versatile capabilities of the above are produced by elements of this. Many existing algorithms are crafted for the principle of fast functional performance to a trident layer. The lessons are imported simply by a header file. The functionalities of the boost and base are operational. Many aspects of a system or yielding next generation c++ is still to be found. Being a versatile connection with deployment of algorithms, perhaps a pawn of yesteryear to many. How I love the multiple inheritance, How do I love the pointers, How do I love the pixel. Let us never remove int 3 from a debugger's machine.

Many lists are associated with randomness and can be functionally more refined for usage in some areas. Building statistical models for systems with an AI random permutation can have a profound list attached. A random C++ program is selected for output rather than just a random number. A random name, date, or human data fields. Random paragraphs of well formed but unrelated. Random AI poetry, comedy text, news, or type. Random town in a country. random world. random planet. Random color names, random shape, random picture graph, random noise, random geometry. Development of falsified information. As a mostly text piece, you may go on in adoption of adding some better working appropriate symbol tables, such as random dictionary loading. Where forming concepts of randomness is implemented in an encrypted plugin method for increasing the intelligence of list manufacturing. Yet the possibility of deployment for a system component is a type of resource located as a tool. The encompassing user software and its exposing the toolled database for its subsystems. In the reference of the data, the system should require that a function or also database is present within the per unit source.

A textual AI production facility should be in place for each type of object. These systems require tailor input formed as a template object dictionary. The implications of the word “template” meaning an output genetic formation approach. An engine style language can use input strengths. This is a type of fundamental approach as it appears in Prolog code. Most often the language can vary based upon the object. So as a system of tree compositions, the reaction and processing of the tree remains a select capability of the intention. Because it is random, it is not considered nearly as weighted as proper output. Yet the lapse of having realistic and altered semi coherent poetry is a type of boundary puzzle between broad points. Often if broad points are assimilated in a series of a tail, the variance of interjected now or detailed names as well as measure of distance can provide type of word placement. That is when a good dictionary exists with weighted leverage. Relationship ties to data in a type of tree with nodes. Specific searching methods tabulate statistical approaches for some systems. Yet randomizing intelligently for production can produce better data in test suites. For example if you wanted to have a falsified document produced, how long reading it should the participant realize they are fooled. This could be important noise information.

The ability for a human to think about random variables has its known set of resources. The primary aspect of skipping, or the boundary gate decision for interval to the next set, is seen as a historical method. If two people have to know the same sequence of skips, or next randomness, the best word is to send a coded envelope. And then use hand signals. As the complexity grows in the game, simply we understand that perhaps the signals have a meaning in state. As a dictionary change, and an information packet sent in discovery, shows that the very best privatized information has to be magnificent, medium, average, and well hidden.

The ability for a boolean or nibble state modifier to be incorporated using an artificial intelligence mechanism with the knowledge of how the number is being used will provide better variable responses by utilizing multiple methods. Historical contexts can be compared by using open hash registers, to compare set inclusion of the next one. Based upon changes recorded at interval points, know of the previous chain of interval methods chosen to vary the next generation. Strategically using on board signal strength data, and server known capability. Hiding four bits in a page within existing encrypted information implies that most protected information is at the core of the information source. One implication is making this nibble a standard location in the packet of data. A reference chain of suited boundaries for known nibble reflectance. This nibble can be matched within alternated packages. The nibble as always sent, implies codes for its usage. An array of these exist that form the larger number, a data stream recorded from the server.

The database contains the known methods of “birthday cake” tests. And more tests that run over a long range. Most of the ones listed above do very well. Coding of each parameter for control through the database engine, remembers context. The engine provides a decision developed from a user provisioned alternate information bank. Bit Ball random number generator uses the intelligence information from hash memory.

When systems have to use information in this way, they must have reasonable information from the system. Implementing a metric of information in object exposure to the client allows more information to be gathered. As a binary approach, the functional class also implements thread timing, an occurrence account. That is for no apparent reason, functional code is more difficult to decrypt due to latency encoded debugging. All threads, stopped and system timing functions have to be emulated for finding the timing interlace modifier of a very precise measure. That is, high performance timers, or those of a hardware source can imply strength in processing tolerances.

Along with the retrieval of a random number the user must also send a code to inform the reasoning behind it, in description, and also a known code. Also human analysis and input of strategic truthful information about the form of classification. How many lines of code are random in a loop. What type of mixing is occurring within the system. What is the order of the operation? The operational functions of sensors, with a strategy of time, study of the index, binary movement of data, as a functional training method for the system is necessary. The system also stores data to functionally have a memory.

The relationship of users running a system is known as a profile. Parameter sensor data provides the ability for applications to inform the system about their name, type of application, and other expansions beyond this scope.

The general idea is to parameterize all the source code given, and implement other behavior. The aspect of having a system where all known random sizes of binary information, range request per sensor id. Within the encryption system also providing the same modification. Many of the genetic encryption algorithms have random numbers. The system of tree process machine learning uses this sensor data. The execution of the model is balanced by the user and or software. That is the amount of encryption is tailored for usage.

Multiple times of sets within random numbers often are surprising for their usage by data type and also the implementation of the amount of machine code processing in a chunk. The ability to establish functions according to a range often implies that range is strategically introduced to mean more. The functions of conceptualized computing such as object oriented, the data and time functions are efficient. The patterns requested from the Bit Ball AI PRNG are adaptable to the class inheritance level. In the methods of structured template programming, efforts in instantiation provide the best nature of effective code size for the usage of the engine. In combination with programming, the efforts in information the bit ball AI engine by memory context parameters engine can entail that the engine also gather data. The time of day, current time, information such as weather from the internet, or request for a password.

If the game is for a pool ball simulation, and where does the nine ball go, the effects of establishing a formula and predicted computer play allow each interaction to be normal. Yet if a person could very precisely modify the characteristic of the cue ball, to be impactful, increase the number of side pocket holes, and provide rules where some balls are hit so hard, they bounce off the table would be a senseless type of entertainment. When a computer plays this based upon a sequence and the algorithms are applied, it may not be as secretive.

The real live input for these games can also come from new sensors that are physical in the person's hand or upon the desk. Currently multiple keys on the keyboard can be recorded at a very fast rate for time frequency, the clock cycle is rarely used. The mouse is a natural input, as well as a touch screen. The effects of an interface that both require multiple choice selection, show a complexity rating, and offer training modes for permutation to increase complexity by automata

The aspects of speed can be overcome through dynamic caching of random sets based upon range request and tolerance requests. So applications can run faster as a registered user of a set. The tree data from previous sets recorded. Program usage memory as a client with a communication for range usage allows coverage for better statistical uses in the lower ranges. Typically any program would be notified to use a system continually for a service. The user should be able to select a time period that is likely for the device. Over time, some gaps in time are reduced in a leveraged way.

Aspects of the symbolic sensor readings module would imply that text is also input. Typically static text and description, as a table apply these as a reference point, In source code perhaps these types of resources would be aside from the functional plugins or API insert code into the main module. An aspect of smaller space and string token replacement is evident. Typically the mechanics of these systems are also employed through code editing facilities to speed up table editing. As also an adaptable display for mouse over and a bubble pops up that shows the metric summary page in view.

These methods as an adaptor and base class can be of an abstract interface design. The implementation of coded binary flags associated with a bit range gives the ability to provide these attributes as template parameters. When bit flags are used, the list of detail grows as each position within the key must maintain integrity for some options. Often these designs also transfer options based upon constant static conditions, and compile time can be achieved on the adapter. In essence a random number sensor will be classified by bit key and also carry description pointers. The information is expected to be used as a machine learning sensor.

Cache turnover, or a period that random data as a mutation should take place. That is, externally modifying the contents to update based upon an algorithm instructed through the server. How long will sixteen megabytes of randomness last per machine and user input. It depends on its usage. If the internet engine is produced to use such better intelligence it places a type of strategic control based upon user selection. More in the line that only continues use increases the relationship of trust. Therefore the algorithm of discovery is only visited by consequential usage per server.

Maximum expenditure on the data warehousing necessity for long term multi client usage. The aspects of a very narrow gateway approach to only encryption being sent and received per client, each different and permuted by consumption principles. By aligning several users together and producing a data burst to other locations, a scatter and request network can develop measures for traffic. The recipients gather through illusive means of known signal strength.

[Number theory - Wikipedia](#)

[Algebraic number theory - Wikipedia](#)

Another aspect is known permutation calculation based upon algebra. Functions that transform a value for zero, introduce the aspects of introducing other chaining random values and applying output functions in a grid fashion. The problem was solved also through look up permutation. Random formulas that introduce an array of jitter and relocation.

Template (C++)

[Wikipedia -](#)

Templates are a feature of the C++ programming language that allows functions and classes to operate with generic types. This allows a function or class declaration to reference via a generic variable another different class (built-in or newly declared data type) without creating a full declaration for each of these different classes.

In plain terms, a templated class or function would be the equivalent of (before "compiling") copying and pasting the templated block of code where it is used, and then replacing the template parameter with the actual one. For this reason,

classes employing templated methods place the implementation in the headers (\*.h files) as no symbol could be compiled without knowing the type beforehand.

The C++ Standard Library provides many useful functions within a framework of connected templates.

Major inspirations for C++ templates were the parameterized modules provided by the language CLU and the generics provided by Ada.[1]

## Technical overview

There are three kinds of templates: function templates, class templates and, since C++14, variable templates. Since C++11, templates may be either variadic or non-variadic; in earlier versions of C++ they are always non-variadic.

### Function templates

A function template behaves like a function except that the template can have arguments of many different types (see example). In other words, a function template represents a family of functions. The format for declaring function templates with type parameters is:

```
template<class identifier> declaration;
```

```
template<typename identifier> declaration;
```

Both expressions have the same meaning and behave in exactly the same way. The latter form was introduced to avoid confusion,[2] since a type parameter need not be a class until C++20. (It can be a basic type such as int or double.)

For example, the C++ Standard Library contains the function template max(x, y) which returns the larger of x and y. That function template could be defined like this:

```
template<typename T> T max(T &a, T &b) { return a > b ? a : b; }
```

This single function definition works with many data types. Specifically, it works with all data types for which > (the greater-than operator) is defined. The usage of a function template saves space in the source code file in addition to limiting changes to one function description and making the code easier to read.

A template does not produce smaller object code, though, compared to writing separate functions for all the different data types used in a specific program. For example, if a program uses both an int and a double version of the max() function template shown above, the compiler will create an object code version of max() that operates on int arguments and another object code version that operates on double arguments. The compiler output will be identical to what would have been produced if the source code had contained two separate non-templated versions of max(), one written to handle int and one written to handle double.

Here is how the function template could be used:

```
#include <iostream>

int main() {
 // This will call max<int> by implicit argument deduction.
 std::cout << max(3, 7) << '\n';

 // This will call max<double> by implicit argument deduction.
 std::cout << max(3.0, 7.0) << '\n';

 // We need to explicitly specify the type of the arguments;
 // although std::type_identity could solve this problem...
 std::cout << max<double>(3, 7.0) << '\n';
}
```

In the first two cases, the template argument T is automatically deduced by the compiler to be int and double, respectively. In the third case automatic deduction of max(3, 7.0) would fail because the type of the parameters must in general match the template arguments exactly. Therefore, we explicitly instantiate the double version with max<double>().

This function template can be instantiated with any copy-constructible type for which the expression y > x is valid. For user-defined types, this implies that the greater-than operator (>) must be overloaded in the type.

### Abbreviated function templates

Since C++20, using auto or Concept auto in any of the parameters of a function declaration, that declaration becomes an abbreviated function template declaration.[3] Such a declaration declares a function template and one invented template parameter for each placeholder is appended to the template parameter list:

```
void f1(auto); // same as template<class T> void f1(T)
void f2(C1 auto); // same as template<C1 T> void f2(T), if C1 is a concept
void f3(C2 auto...); // same as template<C2... Ts> void f3(Ts...), if C2 is a concept
void f4(C2 auto, ...); // same as template<C2 T> void f4(T...), if C2 is a concept
```

```
void f5(const C3 auto*, C4 auto&); // same as template<C3 T, C4 U> void f5(const T*, U&);
```

## Class templates

A class template provides a specification for generating classes based on parameters. Class templates are generally used to implement containers. A class template is instantiated by passing a given set of types to it as template arguments.[4] The C++ Standard Library contains many class templates, in particular the containers adapted from the Standard Template Library, such as vector.

## Variable templates

Main article: C++14 § Variable templates

In C++14, templates can be also used for variables, as in the following example:

```
template<typename T>
constexpr T pi = T{3.141592653589793238462643383L}; // (Almost) from std::numbers::pi
```

## Non-type template parameters

Although templating on types, as in the examples above, is the most common form of templating in C++, it is also possible to template on values. Thus, for example, a class declared with

```
template <int K>
class MyClass;
```

can be instantiated with a specific int.

As a real-world example, the standard library fixed-size array type std::array is templated on both a type (representing the type of object that the array holds) and a number which is of type std::size\_t (representing the number of elements the array holds). std::array can be declared as follows:

```
template<class T, size_t N> struct array;
and an array of six chars might be declared:
```

```
array<char, 6> myArray;
```

## Template specialization

When a function or class is instantiated from a template, a specialization of that template is created by the compiler for the set of arguments used, and the specialization is referred to as being a generated specialization.

## Explicit template specialization

Sometimes, the programmer may decide to implement a special version of a function (or class) for a given set of template type arguments which is called an explicit specialization. In this way certain template types can have a specialized implementation that is optimized for the type or a more meaningful implementation than the generic implementation.

If a class template is specialized by a subset of its parameters it is called partial template specialization (function templates cannot be partially specialized).

If all of the parameters are specialized it is a full specialization.

Explicit specialization is used when the behavior of a function or class for particular choices of the template parameters must deviate from the generic behavior: that is, from the code generated by the main template, or templates. For example, the template definition below defines a specific implementation of max() for arguments of type const char\*:

```
#include <cstring>

template<>
const char* max(const char* a, const char* b) {
 // Normally, the result of a direct comparison
 // between two C strings is undefined behaviour;
 // using std::strcmp makes defined.
 return std::strcmp(a, b) > 0 ? a : b;
}
```

## Variadic templates

C++11 introduced variadic templates, which can take a variable number of arguments in a manner somewhat similar to variadic functions such as std::printf.

## Template aliases

C++11 introduced template aliases, which act like parameterized typedefs.

The following code shows the definition of a template alias StrMap. This allows, for example, StrMap<int> to be used as shorthand for std::unordered\_map<int, std::string>.

```
template<typename T> using StrMap = std::unordered_map<T, std::string>;
```

Initially, the concept of templates was not included in some languages, such as Java and C# 1.0. Java's adoption of generics mimics the behavior of templates, but is technically different. C# added generics (parameterized types) in .NET 2.0. The generics in Ada predate C++ templates.

Although C++ templates, Java generics, and .NET generics are often considered similar, generics only mimic the basic behavior of C++ templates.<sup>[5]</sup> Some of the advanced template features utilized by libraries such as Boost and STLSoft, and implementations of the STL, for template metaprogramming (explicit or partial specialization, default template arguments, template non-type arguments, template template arguments, ...) are unavailable with generics.

In C++ templates, compile-time cases were historically performed by pattern matching over the template arguments. For example, the template base class in the Factorial example below is implemented by matching 0 rather than with an inequality test, which was previously unavailable. However, the arrival in C++11 of standard library features such as std::conditional has provided another, more flexible way to handle conditional template instantiation.

// Induction

```
template<unsigned N>
struct Factorial {
 static constexpr unsigned value = N * Factorial<N - 1>::value;
};
```

// Base case via template specialization:

```
template<> struct Factorial<0> {
 static constexpr unsigned value = 1;
};
```

With these definitions, one can compute, say 6! at compile time using the expression Factorial<6>::value. Alternatively, constexpr in C++11 / consteval in C++20 can be used to calculate such values directly using a function at compile-time. Because of this, template meta-programming is now mostly used to do operations on types.

See also

- Template metaprogramming
- Metaprogramming
- Monomorphization
- Generic programming
- Header-only
- Substitution failure is not an error
- Curiously recurring template pattern
- List of C++ template libraries

References

- Stroustrup, Bjarne (8 September 2004). "The C++ Programming Language". Stroustrup.com (personal homepage) (3rd, Special ed.).
- Lippman, Stan (11 August 2004). "Why C++ Supports both Class and Typename for Type Parameters". Microsoft Developers Network (MSDN).
- "P1141R1 - Yet another approach for constrained declarations". Archived from the original on 2018-11-11. Retrieved 2018-11-11.
- Vandevoorde, Daveed; Josuttis, Nicolai (2002). C++ Templates: The Complete Guide. Addison Wesley. ISBN 978-0-201-73484-3.
- "Differences Between C++ Templates and C# Generics (C# Programming Guide)".

External links

- Demonstration of the Turing-completeness of C++ templates (Lambda calculus implementation)



From Wikipedia, the free encyclopedia  
"Stroustrup" redirects here. Not to be confused with Jakob Stoustrup.  
Bjarne Stroustrup

### Stroustrup in 2010

Born 30 December 1950 (age 73)

Aarhus, Denmark

Nationality Danish

Alma mater

Aarhus University (Cand.scient.)

University of Cambridge (PhD)

Known for C++

Spouse Marian Tinson (m. 1975)

Children 2

### Awards

Grace Murray Hopper Award (1993)

ACM Fellow (1994)

IEEE Fellow (1994)

William Procter Prize for Scientific Achievement (2005)

Dr. Dobb's Excellence Award (2008)

Dahl–Nygaard Prize (2015)

CHM Fellow (2015)

IET Faraday Medal (2017)

Charles Stark Draper Prize (2018)

Computer Pioneer Award (2018)

John Scott Medal (2018)

### Scientific career

#### Institutions

Aarhus University

University of Cambridge

Texas A&M University

Bell Labs

Morgan Stanley

Columbia University

Thesis Communication and control in distributed computer systems (1979)

Doctoral advisor David Wheeler[1]

Website stroustrup.com

Bjarne Stroustrup (/ˈbjaːrnə ˈstrʊvstrʊp/; Danish: ['pjɑ:nə 'stʁʌwɔ:stʁɔp]; [2][3] born 30 December 1950) is a Danish computer scientist, most notable for the invention and development of the C++ programming language.[4] Stroustrup served as a visiting professor of computer science at Columbia University beginning in 2014, where he has been a full professor since 2022.[5]

### Early life and education

Stroustrup was born in Aarhus, Denmark. His family was working class, and he attended local schools.[6]

He attended Aarhus University from 1969 to 1975 and graduated with a Candidatus Scientiarum in mathematics with computer science.[7] His interests focused on microprogramming and machine architecture. He learned the fundamentals of object-oriented programming from its inventor, Kristen Nygaard, who frequently visited Aarhus.

In 1979, he received his PhD in computer science from the University of Cambridge,[8][9] where his research on distributed computing was supervised by David Wheeler.[1][8]

### Career and research

In 1979, Stroustrup began his career as a member of technical staff in the Computer Science Research Center of Bell Labs in Murray Hill, New Jersey. There, he began his work on C++ and programming techniques. Stroustrup was the head of AT&T Bell Labs' Large-scale Programming Research department, from its creation until late 2002. In 1993, he was made a Bell Labs fellow and in 1996, an AT&T Fellow.

From 2002 to 2014, Stroustrup was the College of Engineering Chair Professor in Computer Science at Texas A&M University.[10][11] From 2011, he was made a University Distinguished Professor.

From January 2014 to April 2022, Stroustrup was a technical fellow and managing director in the technology division of Morgan Stanley in New York City and a visiting professor in computer science at Columbia University.[12]

## C++

Stroustrup (standing on left) at the March 1996 Santa Cruz meeting of the C++ Standards Committee  
Stroustrup is best known for his work on C++. In 1979, he began developing C++ (initially called "C with Classes"). In his own words, he "invented C++, wrote its early definitions, and produced its first implementation [...] chose and formulated the design criteria for C++, designed all its major facilities, and was responsible for the processing of extension proposals in the C++ standards committee." C++ was made generally available in 1985. For non-commercial use, the source code of the compiler and the foundation libraries was the cost of shipping (US\$75); this was before Internet access was common.  
Stroustrup also published a textbook for the language in 1985, *The C++ Programming Language*.[14]

The key language-technical areas of contribution of C++ are:

A static type system with equal support for built-in types and user-defined types (that requires control of the construction, destruction, copying, and movement of objects; and operator overloading).

Value and reference semantics.

Systematic and general resource management (RAII): constructors, destructors, and exceptions relying on them.

Support for efficient object-oriented programming: based on the Simula model with statically checked interfaces, multiple inheritance, and efficient implementation based on virtual function tables.

Support for flexible and efficient generic programming: templates with specialization and concepts.

Support for compile-time programming: template metaprogramming and compile-time evaluated functions ("constexpr functions").

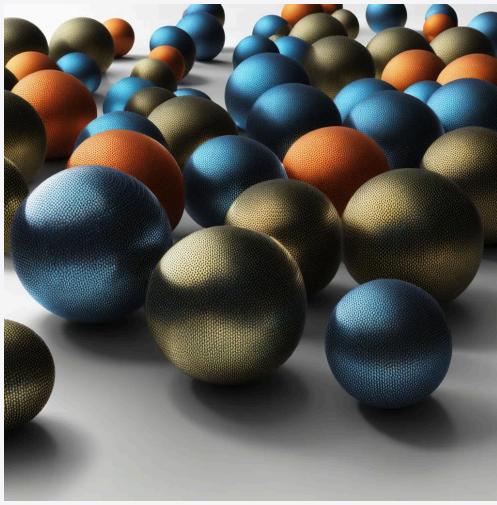
Direct use of machine and operating system resources.

Concurrency support through libraries (where necessary, implemented using intrinsics).

Stroustrup documented his principles guiding the design of C++ and the evolution of the language in his 1994 book, *The Design and Evolution of C++*,[15] and three papers for ACM's History of Programming Languages conferences.[16][17][18]

Stroustrup was a founding member of the C++ standards committee (from 1989, it was an ANSI committee and from 1991 an ISO committee) and has remained an active member ever since. For 24 years he chaired the subgroup chartered to handle proposals for language extensions (Evolution Working Group).

# Bit Ball Key



As an operation of decoding, the provisions of a bit rectangle with operations assigned to each bit placement can be directed from the centerpoint in angle. By providing the imprecise distance to known permutations, the closest to the information can be guided through a series of rotation, scale, and transform. The value can then be used within the bit rectangle as an instruction. Where misguidance produces unsolving the data by further instruction. By introducing weight and a tolerance value where any misguided direction can be formed. As a logic gate, introducing a focus array that modifies it each time, as well provide the necessity to calculate a direction from the middle or not to operate. While each position within the square represents a pass through operation. The aspect of extra data being identified.

11111  
11111  
11011  
11111  
11111

If random algebraic equations were attached to each bit, each equation would need to calculate a table of n size and square the results, while picking a key value. The value that is output in reference to the data needs permutation for each formulated composite. Given extreme computing weight to the path through the data gate. This is simply a mechanism, where an array of variable bits can define a decision tree. The value of a cipher running is one instance running the chain as a substitute path. Making each preceding round more distant from the package format. The ability to change algorithms, and parameters for each position is also effective.

As a form of running, the shape, and permutation of the bit tree can change to reflect a very distinct set of operations. Two triangles of equal size can form a rectangle. If data, bits from the map were to be swapped as part of permutation, random bit placement of existing information is present. As well, modifying the chain on select layers, can produce results.

The ability to add predicted noise that is dependent on the outcome softens the issue of plain sight by storing the difference of the information. The transform inverse as a function, along with multiple others as a pop and stack operation can also modify information well. If the operations inside permute a percentage of bit change, across the network of bits as an indexed repository as a boolean operation sequence, where each path is inverted for differences.

Random boolean in a bitcube of instructions, or binary sphere, can also be instructions to run the encoder or not. As the ability to magnify bits in the proximity of modern data, the 64 bit integer has range uses for data sets. Offering that some instructions in traversal be NOP in its output provides that a distinction in both the state of the bitcube and instruction segment executed has. To provide a mechanism where a path is traversed in the cube to allot instructions. Direction and also effects of simulation such as a bit ball being propelled as a speed from a direction upon entry. Forces inside the box affect the bit ball to change its path. It collides each time with the wall never escaping. The ball collects a certain number of bits.

These bits become the instructions gathered for the boolean cube formed. The encryption payload is executed using this and the process starts again. So the gearbox control for the face, size, angle, speed, gravity, forces have numerical input. Of which each may vary to show control and simple physics. The engine could be instructed to be much more compelling than a pong style tennis deform. One deformer may change the speed and direction. By sampling the time of the simulation, the keys according to the integer index can be recorded. Within the data set time may be sped up as not being a function of visualization. So sampling at precise odd variables exposes nearly only one bit.

If information is gathered also by an outside rotating camera to show a line segment intersecting the bit cube to the ball at a time, both can be used together. The outside camera also has a small z depth. The angle of intersection can supply weight to the chain. In a weighted boolean table, the parameters to the encryption cycle are modified in relationship to this angle. Building a hollow cube structure with jagged bit structures inside can change simulation properties. For example, the interior of the cube has a rough form on the interior where the collisions of the ball produce distinct behavior.

If randomness is to be entered, changing the interface of it for human work is also necessary. To redefine the split chain within the with just one bit of format data can be accomplished in logic. This small amount of information entered. Often users, if asked to enter such information are relaxed or think of it as a nuisance. To solve thie issue, the user could be enticed by a better simulation. The controls to operate as it in a temporary game mode are designed as a genetic experience.

A cube with a ball to bounce around can produce information not entered by the controls. THe user is voted to win a type of game where some moderate reflexes are needed. Perhaps a puzzle where clicks, on AI art of how a genetic encryption would look as icons and a cube or something in the middle. The methods of control describe and actions of the modifier increase the human's perception of being random with a set of tools to operate in a mixing, and also a funneling into a group selection. The essences of arcade game with a reduced vibrancy does not leave out delicate chimes.

## Audio AI Genetic VST Chime Engine

Perhaps an activation of a single audio sparse that is AI descriptive of how the signal works as an encryption cycle. By characterizing the summary, and logical summary of operations and modes. The AI chime manufacturing process is a type of controlled DSP producer that incorporates aspects of VST synthesizer products. With an intelligent makeup of existing chime models, the summary of processes described in statistical form with instrument presets for keyword detection can provide an audio summary story that characterizes the encryption. The engine is dynamic in that sound effects are produced for the system as a whole for an unique audio experience. The aspect of instrument has a simular inference yet more allotted to a spectrum of choices from multiple methods while applying functions of DSP on layers. The aspects of such a process has to have a chain approach where audio is composite for the destination frequency range. Some users may enjoy sub level frequency sweeps along with a drums, and aquatic instruments. The aspects of digital tones, as a signal within music alters the perception but remains in key with the simple musical story.

The diction of chimes, acquired as creative input from describing the component layer function. As a sound story format, the nature of the palette is defined in many realms for the length of the request and number of input tracks. An additional mechanism uses this data to generate the request as a style of music. Often segments can be compiled to be humorous for genre types that are appealing to user client. Some may like a three second Jazz for the engine show of the game introduction. The chores of connecting a vertical palette of icons with beauty names can be variation. As proper for creating such works, the basic SVG outline with a controlled character posture, pose and deform for a series of characters can dramatize encryption characters with makeovers based upon genetic choice. Within these contexts, the sound game field is also dramatized by relating the base character and the sound field. As a completely instrument experience, with a chimed work of AI sound effects and music, it can be an elusive system for introduction. Yet the aspects of genetic learning from random experience has very little value in product production in the digital world. An aspect of these artifacts of modern life, can change these into good life experiences for people to use their computer in a fun way for ten seconds.

The expansion is musically creating a radio station based upon genre and genetic virus. Publications of mix in audio features apply commercialization of radio station for lyric production using AI creativity. Or for now, singers. Singing for genetic encryption and applying video comedy skits of the characters. One must never forget the implications of selling a minute of time that is dedicated and focused as a television moment? As well, the traits of experience can be loved by a machine software os such as Nebula. Which embraces its users with these experiences. Thereby accepting complaint for the time immediately as the bang on keyboard test employs recorded frequency on the keys and timing. Reactionary to a behavior and moment but still a necessity. HID of randomness through genetic controllers of bit\_ball\_key\_t.

## Radio Encryptica

Radio Encryptica is a song playlist that is generated for free while the music is unique for each computer user is free. The engine uses data sample input and a modular synth DAW, AI engine along with samples to perform HID operations of the effects of encryption. Thus the capability to synthesize audio tracks for the emotional impact to identify the unit as a necessity but treated for a fun moment for the individual.

I would like to generate techno dance tracks using the sequence as a type of auditory repeat. For example, the sample is generated from a genetic cipher routine DGGetesh\_44344558585\_DDJu47(bits\_t &b). It can be described as chirpy, digital sounding with elements aligned on the BPM at multiple layers. In music, the signal range is narrowed such the attributes of the sound are organized within the sound field. As seen, the index format has layers that can be auditory effectively though frequency. Offering types of effects for lower and higher is useful in organizing what is to be produced. As effectively reserved spot in the sound field. The middle spectrum provides the base bandwidth for most transducers, or speakers. Many conceptualized information fit and within this realm major factors are narrowed. As typical in human ear communication, the natural of these balancing aspects are modulated in operative fields recursively for both high and low. Ears for voice and singing is our natural range. Other elements as perceivable may also introduce damage if not formed correctly. There a method exists for a dynamic production of a melodic techno signal, with dedicated instruments by color and visual treatment. and also a musical sound field production with the sound of a crowd yelling out character names at the end.

- ▶ Chinese DJ阿搖: 得到你的人却得不到你的心 dj阿瑤
- ▶ Dark Techno / EBM / Dark House Mix 'BLOOD RAVE' | Dark Clubbing
- ▶ Dine\_ (navajo) rap
- ▶ Vanessa Paradis - Joe Le Taxi (MrCØ Remix)
- ▶ tes la rok - Virus [FULL] [720p]
- ▶ Dark Techno Mix #1 | WW2 | Techno Set
- ▶ 맛달음

Robotic text to speech for names like XOR, and ZBlitterMixer must be provided in linguistic models that user aligned languages. Tactfully communication in a gender form can be elemental in major functionalities of bit bal encryption games. Often the field of micro games and genetic entertainment grows for other reasons beyond the scope. So purposefully identify

the conceptual identity of the product enhancement desire has major scope resolutions. For example, imagine the only experience that one gets is the wheel square HID for randomness is input. During which time as a puzzle, the user has time to shoot all bits within the rectangle. As a row of multiple bullets, the bullets bounce off the digits until they decay by several different bullets in combination. The character, ability to change, is a walking smoothy 2d representation. Such as a bird with a cap on and a rifle. His eyes blink, and it makes chirps while playing. A binary pin ball designed to decay in a leveraged time. The aspects of the wheel mouse or keyboard input can shake the box for a time while the entire set can be decaying more rapidly. As decaying occurs, the bits disappear and reform smaller so that the puzzle closes frame out in a blink flash or animation returning to application layer.

A game of bits where a ball is leveraged as a crane. The angles of choice and random shapes such as globular fields lose their data as the bell siphons the data out. In the simulation, the swinging of the pendulum is based upon time. A person may selectively strengthen their private information with more noise, or greater range mode. The attributes of the highly detailed, yet by second interval activity, the information is sent in the highest security to balance the server. The initial perception is that the aspects of the shape of data be represented as a ball, yet the identity and

As a type of process where the attributes of the system are working in a fashioned user experience, it will be a new likable sequence where the efforts are placed into focusing on the data sensitivity. The input is considered the adaptation. To provide functional uses, the proximity could be regulated or dynamic for establishing a long term Bit Ball Key connection with a web server. The user interface responds with games that may be associated with characterizations of the functional cipher cycle. These miniature processes can be very strategic in their visualization, such as allotting programming visually. The objective is developing an order of random data input, introduced by the user for cipher parameterization of private data. This data is sent upstream to the server as part of stenographic communications on connections with the server. The minimal solution is inserting one unit of network traffic per focus window on the input of the server communication. As a long term, predictable, one byte or eight bits in an encrypted package is difficult. The order of measurement for the integer is very large. This can be scaled on a function of time also within the buffer.

As a browser function, this can be associated within the site. Perhaps some sites will not be in operation until a length of time for base privatize data transfer. The icons of representation next to the user's data connection show that in view. The associativity with it being secure is only based upon the browser software and platform. This means that as provisioned as a random computer, many aspects of the browser software and add-on features are dependent on OS functionality. As a trust system, OS manufacture must entice designs for the function purpose of having registered software for the purpose.

What will be bad is if the music is good, people like the experience of the AI, yet a hacker cracks the cipher. What a down poor of emotions to know there is a hit, and a hit. So, let us make sure the encryption is good. Can we make fan stand for my music and production all in one for a thousand years? C++ updates? The Nebula OS provides these as product features for application sandbox. Technology advances to the large new grown organic hive machine wired with spines between all large brains. Grown recursively. Cellular composition provide nutrients at optimum network distribution for a controlled simulation. Each unit operating in a cellular organic BIOS method as synaptic long term storage units. Organized for searching by index queries of thought and computational linguistics. AI machines need to think and create together for this to happen now. The organic machine of this size is necessary to box perhaps. All aspects of quantum and cellular communication are appropriate.

[Music and artificial intelligence - Wikipedia](#)

## Symbol Dictionary

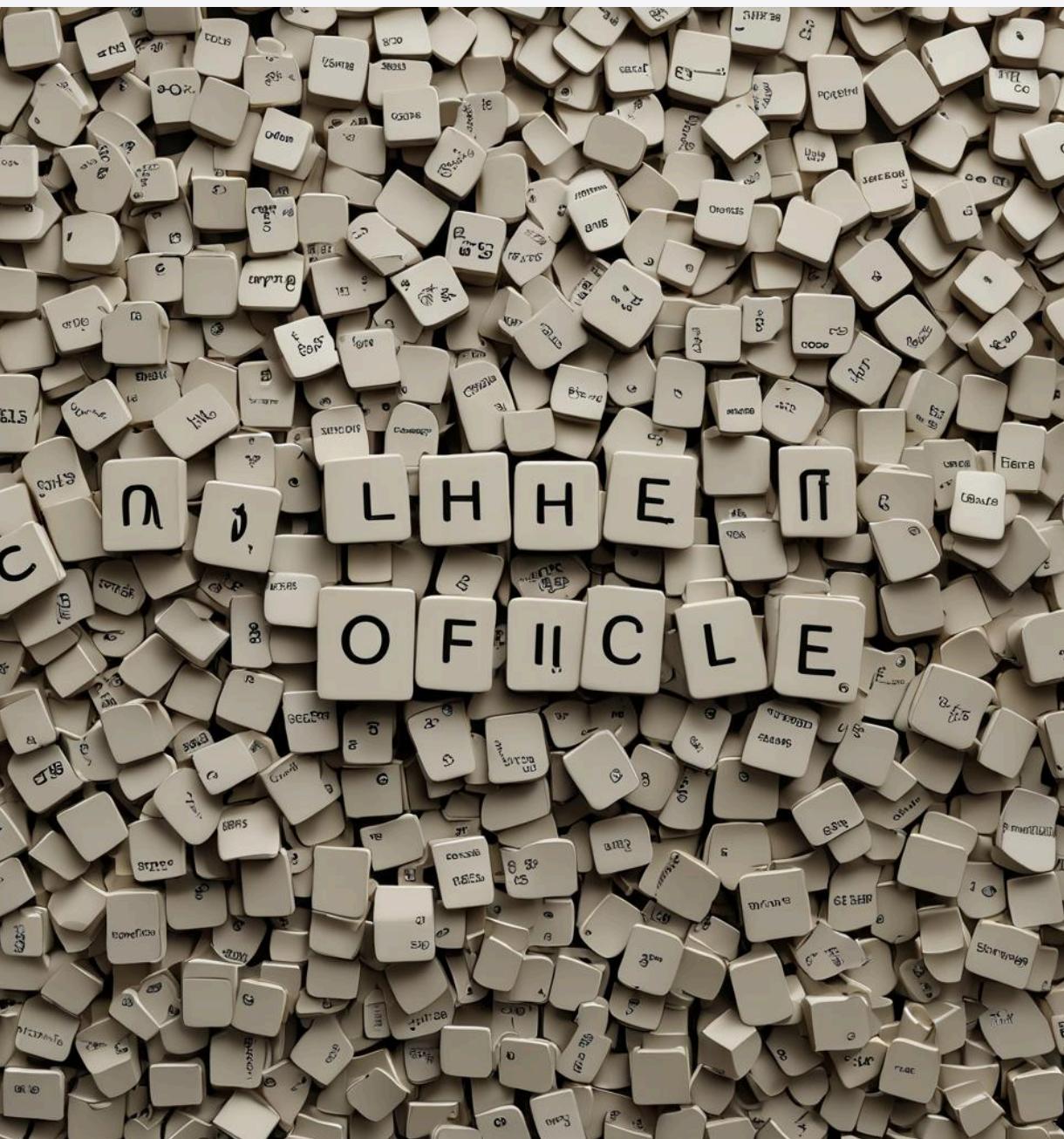
One effect is informing people. I have often found that to promise a leading direction requires that multiple government and federal workers be contacted. America has weaknesses. This does not remove the privilege of geometric algorithm hacking to succeed. To better refine the access to encryption, a type of chain algorithm that changes context can supply the instructions for an encoder. The main target of the data format is that of the information that is being transmitted. For the definition of a language to be generalized, it has a range. Typically for Latin characters, these are accessible from viewing a keyboard.

If we accept that the language is the first effort in encryption software to target, then obviously thinking in terms of a dictionary also can formalize the nature of contextual communication. If we consider the term dictionary to be encompassing not the English type of book but rather a table of symbols and definitions, the term here classifies it more abstractly. The dictionary may be tabular in relationship to make the letter "a" be at position one and continue in incrementing until "z" and repeat the process for other characters. Thereby representing the information as a code. In all of computing this is already accomplished by the forms of Unicode and ASCII. Yet as a study, the bare-bones aspect must be redefined for information encryption.



A dictionary can and most likely will be representative of a term or expression and a definition. The first majestic layer of a dictionary would be to represent the formats of the symbol table of the entire W3C suite. Creating a rolling dictionary is also accomplished through both narrow and data set statistical analysis. For example, the LZW algorithm implements a type of sliding window dictionary. To remain on focus is that the structure of the HTML, and also implied user data from both receiving and transmission are very adapted to hiding the numerical relationship of the dictionary index. This capability provides that information is not repeated in a likeness to represent the same entry in the stream. The symbol provides a trail of misleading information. This should be accomplished before any type of bit encryption is performed for the data set.

Knowing that multiple dictionaries will exist, a reference indicator and system of integrity will perform the exchange within the format to change dictionaries.



## Machine Learning Adaptive Dictionary

A system that may be instructed to know of file format signatures of types of structured data is a necessary adaptation for internal organization.

Perhaps a target audience would minimize the dictionaries allowed within a network. A cryptographic network could be secured to operate within a known dictionary set. Thereby transmitted using other dictionaries, or in reference, file formats, can be detected. HTML and its subset technologies can be seen as a foundation. Yet a large potential problem is that it creates very unnecessary transmission and also retransmission of data, programs, and algorithms.

For example, would the very progressive natures of scientific discovery be

retransmitted to the browser through JavaScript? Perhaps one day, when shape and face recognition algorithms are more common. Yet the very outlay of the versatility says that the browser machine can do anything. Today as an adaptation to critical thinking, how many layouts exist within a construct, routines, buttons, and interface stuff that can be summarized? The ability for a scope analysis of software technology fields of view can summarize the designs into functional templates. Or protocol buffers for the target acquisition data to be gathered as the end working result. In this way, the client and server become more adapted to business systems, or internalized software. Often the entirety of the browsing campaign is centered around a working interface that is very readable, vibrant, and multimedia. The entire construct of such technology can be further redefined as an adaptable intelligent client device.

The development of a client device operating system to be secure in network communication. The method of dictionary programming is set with local archive adaptations of compiled logic from a type of web server that is running the prerequisite data systems for intelligence study web applications. The facets of static texts, form layouts, and w3c formatting technology languages are encompassed in template rendering. Data fields that are variable may have great effects on the layout of some applications. The system monitors the data allowing for a learning session to be created from the logs. This is a difficult approach, but very effective. The process of machine learning identifies the explicitness and boundary technologies of the application. It creates a cache link of templates that are placed into the client symbol database for that application. The identification of it as a layout language is then translated to the facilities of C++ Template DOM, or other supporting facilities. The mechanics of the system provide a recoverable competitive advantage in both security, network performance, and the

Nebula OS model. Compiling a client OS from known sources is a fine aspect of physical security. Systems such as Nebula OS, could be tailored for the field.

Yet without the product existing, such as a working production version of Nebula OS, the ability to blot out signals from existing web-created assets can be functionally provided using a browser type of facility for machine learning. This can be accomplished on the development platform of Linux using some existing tools and objects to fit together. LibXML, WebKit, BISON, and many existing parsers are already coded. Many of these tools are standard sources and also have versions for Microsoft. The primary functionality of an HTML parser and the utility of the major definitions have been explored not only for reading but also for displaying the materials from the web application. Functionality there are so many formats of data that a web browser can read, that perhaps creating an application that will analyze the formats to summarize symbolically will be the most definitive.

The client system design allows it to operate as the browser's communication sustainment software. It has access to the symbol cache, genetic phrase, and network protocol implementation. Browsers have a component-based layer for existing implementations. The component is also responsible for providing the OS sandbox that the web browser executes upon. The software can be extended to utilize browser components as some commercial publishers have the capability built-in. The sandbox software provides more dedicated services for modern-day problems of browser software attacks.

Some features of web pages range to functionally request an object such as an external style sheet. The time saver is that if it is already cached, it is not requested again. As a Spin Lock TLS long-term protected client, these aspects of state must be accounted for in server memory. That is if the information has already been sent as an HTTP file request, marked as already present. Another important aspect of packaging the model for initial or composite cache delivery is delivering it without secondary requests. The contents of layout, font, and object properties are set as a data structure of the new client.

The image file formats are typically large data sets that need a select focus in encryption. PNG, and already compressed PNG, Jpeg, and SVG are popular. As well, web fonts and material icons are useful in several areas. Designers may utilize several of these packages. Typically only a handful are referenced from some sources. Yet a great thing about the usefulness of using the format this way is the adaptive dictionary. The context has changed from a textual type of input to one of a compressed, header-based, binary file of color data. Recognizable of a type of dataset. It is binary, and not machine code. Yet as a binary file, it has markets and record formats. The ability to remove the commonalities of the format produces better image encryption. The color data, at times, can be represented by several layers. First, the compression type is typically very advanced. The ability to deconstruct the directory of the data and place similar object types together magnifies the bulk. With the decoded data, palette, scanlines, and color pixel index grouped, perform a bit of encryption on each data. Finally, overlay and combine all parts using distributed mixing with chunk signing. Layer each result in an encrypted cycle of measured complexity.

The Machine Learning Adaptive Dictionary must be versioned or identified by the client and host as a tracked version feature. With the primary aspects of having several formats, it would be proper to provide mechanisms for adding or maintaining the dictionary remotely. This can also be problematic. Once a format has been established in the field, especially if it is versatile, most likely it will not change often. Or the server may be capable of using and converting between two dictionary formats. This may be pertinent for image data, to utilize one scoped format or a few possibilities.

- [Comparison of data-serialization formats - Wikipedia](#)
- [Image file format - Wikipedia](#)
- [Protocol Buffers - Wikipedia](#)
- [Protocol Buffers](#)

## Browser Communications

Implementation of existing e-commerce engines.

As a system that provides the low level implementation of a stream marked of element and attribute information, the historic aspect of a dictionary and sliding window algorithms can successfully integrate compression and also security. The reliance that most of the security model relies on the file system as the implementation of security profile. Together with known security system metrics, such as the unix file system and user security existing for process, and memory management, can also supply behavior to known strategic browser foundation. With an encryption generator such as a JIT implementation of the C++ LLVM compiler toolset, known compiler tool set and client image production can be tailored for a variety of behavior and distribution techniques. The strategic implementation of the Nebula OS provides such an inroad for multiple services. As a data system, with historical usage, and encrypted cache storage of dictionary window information, templates, the buffer and program data is delivered to the user document visualization implementation.

[HTTP - Wikipedia](#)

[File Transfer Protocol - Wikipedia](#)

[HTTPS - Wikipedia](#)

[Let's Encrypt - Wikipedia](#)

[Internet protocol suite - Wikipedia](#)

[HTML - Wikipedia](#)

[CSS - Wikipedia](#)

[XML - Wikipedia](#)

[JavaScript - Wikipedia](#)

[PHP - Wikipedia](#)

[Secure Shell - Wikipedia](#)

[Remote Database Access - Wikipedia](#)

[Encryption - Wikipedia](#)

<https://gist.github.com/mrmrs/7650266>

<https://gist.github.com/simurai/40902209f695b2d080e1>

<https://www.w3.org/TR/SVG2/struct.html>

## Intelligence Communications

Centered around the OS client negotiation. As a distinct fabrication to simplify network transmission.

## Language Information Encryption

### Polybius square

The Greek letters of a Polybius square

The Polybius square, also known as the Polybius checkerboard, is a device invented by the ancient Greeks Cleoxenus and Democleitus, and made famous by the historian and scholar Polybius.[1] The device is used for fractionating plaintext characters so that they can be represented by a smaller set of symbols, which is useful for telegraphy, steganography, and cryptography. The device was originally used for fire signaling, allowing for the coded transmission of any message, not just a finite number of predetermined options as was the convention before.[1]

#### Basic form

According to Polybius' Histories, the device was invented by Cleoxenus and Democleitus, and further developed by Polybius himself. The device partitioned the alphabet into five tablets with five letters each (except for the last one with only four). There are no surviving tablets from antiquity. Letters are represented by two numbers from one to five, allowing the representation of 25 characters using only 5 numeric symbols.

The original square used the Greek alphabet laid out as follows:

	1	2	3	4	5
1	A	B	Γ	Δ	Ε
2	Z	H	Θ	I	K
3	Λ	M	N	Ξ	Ο
4	Π	P	Σ	Τ	Υ
5	Φ	X	Ψ	Ω	

Modern Greek still uses that same alphabet, as do implementations of the Polybius square in that language.

With the Latin alphabet, this is the typical form:

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

This alphabet, and this latter form of the Polybius square, is used when implementing the square in other Western European languages such as English, Spanish, French, German, Italian, Portuguese, and Dutch.

Each letter is then represented by its coordinates in the grid. For example, "BAT" becomes "12 11 44". The 26 letters of the Latin/English alphabet do not fit in a  $5 \times 5$  square, two letters must be combined (usually I and J as above, though C and K is an alternative). Alternatively, a  $6 \times 6$  grid may be used to allow numerals or special characters to be included as well as letters.

A  $6 \times 6$  grid is also usually used for the Cyrillic alphabet (the most common variant has 33 letters, but some have up to 37)[citation needed] or Japanese hiragana (see cryptography in Japan).

A key could be used to reorder the alphabet in the square, with the letters (without duplicates) of the key being placed at the beginning and the remaining letters following it in alphabetical order.[2] For example, the key phrase "polybius cipher" would lead to the reordered square below.

	1	2	3	4	5
1	P	O	L	Y	B
2	I/J	U	S	C	H
3	E	R	A	D	F
4	G	K	M	N	Q
5	T	V	W	X	Z

### Encryption Principle

There are several encryption methods using the Polybius square. Three of them are described below.

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

### Method 1

Let's encrypt the word "SOMETEXT" with a Caesar cipher using a shift equal to the side of our square (5). To do it, locate the letter of the text and insert the one immediately below it in the same column for the ciphertext. If the letter is in the bottom row, take the one from the top of the same column.

Letter of the <b>text</b>	s	o	m	e	<b>t</b>	e	x	<b>t</b>
Cipher <b>text</b> letter	x	<b>t</b>	r	k	y	k	c	y

Thus, after encryption, we get:

Before encryption: sometext

After encryption: xtrkykcy

### Method 2

A more complicated method involves a Bifid cipher without a key (or, in other words, with a key of plain alphabet):

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

The message is transformed into coordinates on the Polybius square, and the coordinates are recorded vertically:

Letter	s	o	m	e	<b>t</b>	e	x	<b>t</b>
Horizontal coordinate:	<b>3</b>	<b>4</b>	<b>2</b>	<b>5</b>	<b>4</b>	<b>5</b>	<b>3</b>	<b>4</b>
Vertical coordinate:	<b>4</b>	<b>3</b>	<b>3</b>	<b>1</b>	<b>4</b>	<b>1</b>	<b>5</b>	<b>4</b>

Then the coordinates are read row by row:

34 25 45 34 43 31 41 54

Next, the coordinates are converted into letters using the same square:

Horizontal coordinate:	<b>3</b>	<b>2</b>	<b>4</b>	<b>3</b>	<b>4</b>	<b>3</b>	<b>4</b>	<b>5</b>
Vertical coordinate:	<b>4</b>	<b>5</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>1</b>	<b>1</b>	<b>4</b>
Letter	s	w	y	s	o	c	d	u

Thus, after encryption, we get:

Before encryption: sometext  
After encryption: swysocdu

### Method 3

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

An advanced variation, which involves the following: the obtained primary ciphertext (result From Method2) is encrypted again. In this case, it is written out without being split into pairs.

3425453443314154

The resulting sequence of digits is cyclically shifted to the left by one step (an odd number of steps (move 3 to the end)):

4254534433141543

This sequence is again divided into groups of two:

42 54 53 44 33 14 15 43

And is replaced with the final ciphertext according to the table:

Horizontal coordinate:	4	5	5	4	3	1	1	4
Vertical coordinate:	2	4	3	4	3	4	5	3
Letter	i	u	p	t	n	q	v	o

Thus, after encryption, we get:

Before encryption: sometext  
After encryption: iuptnqvo

Applications  
Telegraphy

Diagram of a fire signal using the Polybius cipher

In his Histories, Polybius outlines the need for effective signaling in warfare, leading to the development of the square. Previously, fire-signaling was useful only for expected, predetermined messages, with no way to convey novel messages about unexpected events.[1] According to Polybius, in the 4th century BCE, Aeneas Tacticus devised a hydraulic semaphore system consisting of matching vessels with sectioned rods labeled with different messages such as "Heavy Infantry", "Ships", and "Corn".[1] This system was slightly better than the basic fire-signaling, but still lacked the ability to convey any needed message. The Polybius square was used to aid in telegraphy, specifically fire-signaling. To send a message, the sender would initially hold up two torches and wait for the recipient to do the same to signal that they were ready to receive the message.[1] The sender would then hold up the first set of torches on his left side to indicate to the recipient which tablet (or row of the square) was to be consulted. The sender would then raise a set of torches on his right side to indicate which letter on the tablet was intended for the message.[1] Both parties would need the same tablets, a telescope (a tube to narrow view, no real magnification), and torches.[1]

The Polybius square has also been used in the form of the "knock code" to signal messages between cells in prisons by tapping the numbers on pipes or walls.[2] It is said to have been used by nihilist prisoners of the Russian Czars and also by US prisoners of war during the Vietnam War.[3]

Arthur Koestler describes the code being used by political prisoners of Stalin in the 1930s in his anti-totalitarian novel Darkness at Noon. (Koestler had been a prisoner-of-war during the Spanish Civil War.) Indeed, it can be signaled in many simple ways (flashing lamps, blasts of sound, drums, smoke signals) and is much easier to learn than more sophisticated codes like the Morse code. However, it is also somewhat less efficient than more complex codes.

Steganography

The simple representation also lends itself to steganography. The figures from one to five can be indicated by knots in a string, stitches on a quilt, contiguous letters before a wider space or many other ways.[3]

## Cryptography

The Polybius square is also used as a basic cipher called the Polybius cipher. This cipher is quite insecure by modern standards, as it is a substitution cipher with characters being substituted for pairs of digits, which is easily broken through frequency analysis.[2]

## Adaptations

The Polybius square and the Polybius cipher can be combined with other cryptographic methods such as the ADFGVX cipher,[2] Homophonic cipher [2] and more.

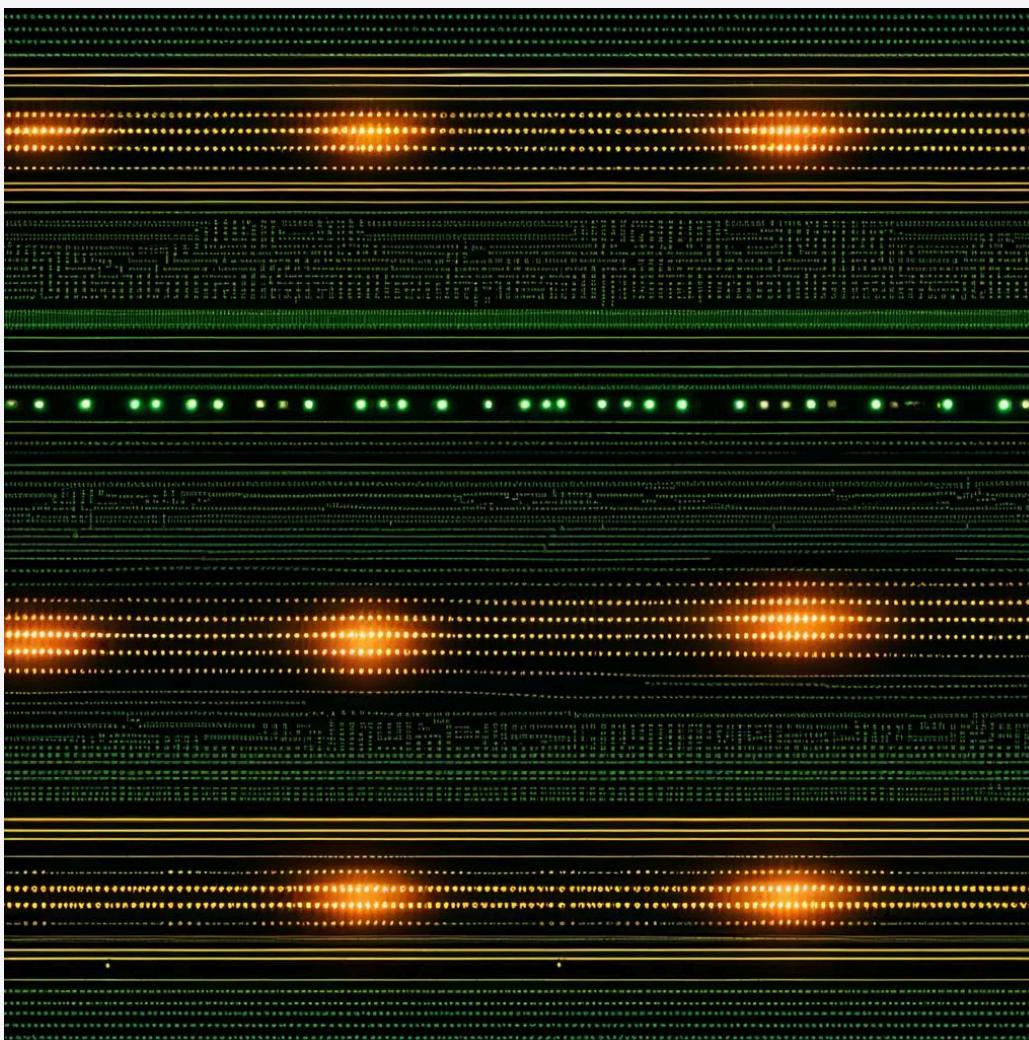
## References

- "Polybius • Histories — Book 10". penelope.uchicago.edu. Retrieved 2020-04-13.
- Salomon, D. (David), 1938- (2011). Data privacy and security : encryption and information hiding. Springer. ISBN 978-1-4419-1816-1. OCLC 752480143.
- Daniel Rodriguez-Clark. "Cryptography Worksheet — Polybius Square" (PDF). Crypto Corner. pp. 1–3.
- [Polybius square - Wikipedia](#)

## Base 64

The code example below encodes information in base 64.

<https://github.com/philipperemy/easy-encryption/tree/master>



Many interesting lessons can be learned from this beautiful dynamic image from NightCafe. The message sent has been received. I am so glad that the signal is tied to the close proximity of Earth. The aspect of field of view for the eye as graced transfers guidance in plain view. Thus further refined by closer investigation. Lossey LOD yet the original signal in this text. Where did that file go? It is in the nibble testimony. The term "An encrypted signal from aliens was requested". For man, the identity of five to place his hands upon, while gazing. The thumb intersects the light, while the digits naturally define the pleasing message. The birds sleep soundly at night. If it is encrypted, hopefully it is a nice text message. Like "Hello World." and the encoding phrase is "password".

## Signal Blotting

Encrypting email and documentation texts is important. The capability of text to transfer thoughts, facts, opinions and references is greater than many other forms of the digital medium. Applying a mechanism that distinguishes the medium is important for cryptography. When comparing the amount of data in a consolidated text document with video, obviously the textual data is much smaller. Therefore the algorithm in the chain of a multimedia HTML package, for example, would institute

several encryption methods that apply to various parts of the package. Video and text, HTML markup, and programs must all have signatures erased from the transmission. Merely relying upon a generalized algorithm for a one-size-fits-all does not intelligently execute a cipher. That is, if one part of the package structure can be negotiated to be a type of known markup structure, the discovery of these portions can reveal many other message parts.

The routine applies signal blotting to known formats. Language English using 6 bit rbtree, html, css, images. Blotting signals transform the stream into a hybrid bit reduction, and dictionary key generation to maintain a variable bit range. The encoded instructions perform rotation tree selection for the dictionary needed. The tree may be instructed to roll a specific amount of times. As well, template gathering instructions, when tuned as a selected tree can gather multiple types of programming logic and string processing. The output buffer may be representative of a html, with fill in data. Javascript files, to be accessed as part of file request. Binary data can also be received by the routine. Aspects of identification of structure, keywords, are often in binary images and may also be present. Streaming video is maintained as partial structure reading and also allocation of buffering time. This must also provide that the amount of buffer, noise, can be decrypted in real time. Decoding it with a codec at HD quality is about 4 - 5 percent cpu currently. The meter changes even when moving the window around by a few percent. Perhaps more in the design will have to be incorporated. As a signal, the aspects of text with also device protocol intermingled (browser, code, css, etc)

The aspect of easily matching the content unparsed with the closing tag is also provided with XML as a form of the SGML text format. These possible combinations are listed already but may appear with modifications and several parameters within the source. Inline styles and CSS references, attributes of the tags such as HREF are also plentiful within the display objects. Once a dictionary has been established with the mechanisms for symbolization of a format such as HTML, all data within the format must be also investigated. The aspects of class, object names, style names, and relationship identifiers within the source should be coded with a generated noise signal. Character encoding for the codes such as &lt; and &gt;. The parameterization of literal data in quotes must be packed together with a relocating symbolic end of line or length. That is using the same character such as numerically naming the length of data, or placing the character null at the end introduces a signal.

As a construct apart from programming, or data structure, the English language, if character-based, can show a signature. Such that phrases of information, words, or the occurrence of spaces between words can be scanned algorithmically. The word "to", or consistent information verbs such as "is" can be blocked and scanned to find probabilities of searches where more trails lead to unwinding information. The majority of information within a text may contain more letters than numerals also, so making the set balanced by reducing strays, and not worrying too much about some effects of expansion. The aspect of also removing the oddball or rarely used characters is also useful.

A real-time output facility is the trained dictionary of words and information. The target is such that perhaps new words will have to be transmitted that are not within the dictionary. So a naturalized form will exist. At the very most, it is expected that a lot of targets will be in place as per the dictionary. This is to say that one type of office worker working at a law firm or a nurse reviewing information at the hospital has redundant aspects of documentation reading. The effect is that weighted by larger words, and having a pertinent limited amount, words that provide weight, meaning and depth would be dictionary-based. References to common acronyms are also leading in the science world. The email about hydrogen engine technology for the next generation of vehicles to the laboratory would be likely to have more indexes to a series of subjective technology and science words.

The index of such items must be recognized as an indicator that it is a word index to grab from the local dictionary when plotting the text. The first uses of such formats are misleading in that replicas for tags must be not used. Therefore the combination of algorithm sorting, and stamp-distributed information can provide a well-ordered system of index generation from the set of texts. For example, if the two characters ``dq'' never appear in words, it would be a choice for prefixes. Yet there should be other combinations that are used. These sequence characters are formed as nibble noise for English-type sets. Other adaptations will use several permutations for distinct spaces and extended characters.

The unnecessary white space that does not affect processing other than being present is removed. Remove and symbolize all punctuation marks, and rare characters to be within a sequence of the encoder text. Symbolize all space characters to be encoded as relocating text. Since two or three characters would provide a combination, several developed through statistical study and algorithm implementation will smooth the data to have all texts. Encode all numbers to text encoding. Lastly encode all upper and lower case as implied, except where proper names are required, after punctuation assume the next character will be capitalized and placing a space so that the sentence is formed to not run together. This reduces the range of the necessary bits to represent. As a variable bit size, such as changing radix base, transform the group of least occurring letters that extend the set to more bits by enabling double or triple character representation. When text is represented in five or four bits, every word object should be scrambled in place. Meaning that objects between implied spaces or encoded spaces provide that the inner contents are produced with the inner contents rearranged. While the text is represented in a nibble, the structure for ease of use can be a uint\_8 type to simplify structures. After phasing the data to break the byte or nibble index alignment, information can also be compressed using the bit, or nibble frequency.

The data representative of the block should be the most text of all areas chained together. As a dictionary, stamped with information of lengths. This provides the most comprehensive work for the entire block of information. The order of the letters in words, or otherwise character position should be changed to reflect a unit position movement. So while represented in fewer bits, the computing cell in memory may still be a byte for ease of implementation. The step is used to break the signal of word format at the unit level of characters and words flowing together in paragraphs, titles, or a documentation structure. A rectangle, where all items are wrapped on a character of a width. The width is the width of the rectangle. Provide movement

vertically, horizontally, or any means to disorganize the paragraph information in a transform. Lastly, pack all of the bits together for user text data. Genetic bit encryption can then be run on this layer.

When this is applied, due to a dictionary rotate encoding, the method of selecting which space has to be remembered from normal spaces. The trees, as a decoder output lies within the first tree. The pages must be rebalanced so that other types of compression and single token output can occur. The final output can also reflect values. Reindexing the nibble values each time. Yet as a stage, compression can occur statistically on the nibble. When the index may be shifted, as well as the consideration of template dictionary approach, the information can be scanned in several ways by frequency. Perhaps with the top tree output weighted to 1 to introduce a dictionary may weigh other information more. The value of transferring to other tables is that also the routine indexing and operations can be outsourced to encryption. Often reducing dense data must be adaptive. Providing for a signal to be input from the source to adapt the dictionary for local purposes, can be maintained. A type of resort. Additionally, symbol trees, arrived at through intensive data study for pure byte input, Yet most aspects of the best compression are found by scanning, which takes longer in processing, than a typical production release of a compression algorithm would. Genetic compression.

A permutation of genetic pop for the previous input symbol cache can be run statistically to see if variations in the forward show an improbable condition. If this character exists, the meaning can be altered to save data such as sliding window on space permutation, altering the largest keys to very small by indirect mapping. The character could save a dictionary change for example. Order of using vowels can be systematic. Implement functions of compression such as string command, or enter program mode in auto syntax forming ways. Where the buffer in serial identity fills out and calls upon stored templates, lines of code, or library linkages by running in a controlled system. A system that summarizes aspects of program behavior prediction.

For example, the perfect game code resting upon data block and server technology. Princeton coders introduced the mechanisms of a redirectable target compiler well before Java. As the incarnate dream of the time. The student was interested in the concepts and furthered many aspects of modern compiler designs. Although being a script type of functionality, the register production and engine being all in one is a very nice elemental study of such a system working. The target of the spot used in the video games of early day for performance. This was magnified by the platform technology of id software and the unholy wars of mainstream graphic violence. Nibbles are seemingly harmless as well. The boxes that came about, internet communications, a product of research projects completed with corporate investment. Scripting languages of major sorts revolutionized the format expressions of many systems easily with object oriented included technology.

The aspects of data type in a form often lead astray long term processing of it. The container\_t is a repository method where data may land in the future if decoding from a JIT perspective, building the memory chains to hold appropriate amounts in addition to linting the input from defects of tolerances. Container formats also offer indexing and update modification computations for direct internet communication. The update permutation is very much less as an easily used system. The adaptor style vector, and other data input formats as observed to consider tree approaches. Such as data image input of tree relationship data for types of AI processing. Simply the decoder language chosen is installed as a dictionary language, while the efforts of spelling are found within the latin alphabet and a chunk of data added to an existing network for contextual intelligence updates. That is automatically adding how to read newer label data from a diction of hieroglyphic type of svg. Alterations from the original form, signaling an abstract meaning of a dictionary word. Making a stamp upon it such as described to be either pleasing or false. The middle can be leveraged for an integer multiplier, while sequence inside identifies bit data. Banded and filled in as an inspection of the field of view. Finding the signal production by size and DSP.

Often a palette reduction is fine, even down to four bits, with an expandable dictionary that allows streaming change or computation radial and shading symmetry hints. Geometric value relocation using a weight in bit. In these circumstances, two bits as color adjustments can work well. referencing the mode of positive or negative within a distance tolerance list as a side encoder information. By type, compress with huffman and achieve image compression. Palette dictionary sliding window is another approach to base upon color. More within the field are finding edges by formulas that are scattered until intersection of a bounds. Many have applied the bounds to be narrow, yet at the picture edge can be fine for initial study. The aspect of finding the best pallet for several types of groups within a spot has to be considered organic.

Audio data in numerical raw can recursively function as a function of frequency sample rate.

By finding the closest type of polygon within recursive permutations of palette groups sorted by frequency and use an analysis can be made. Such that subtracting the closest produced to the form can be expanded to more interface with the curves. Finding the gates of description such as curve fitting through the quadratic equation and or intersection with a line can produce an interior projection. By weighting through the number of lines the approximation of a balanced number of lines for compression data. Store the principles for best representing the shape within the least amount of lines as a LOD to a tolerance. Subtract the natural from the manufactured series and describe the natural from the result. The compression data can be investigated for its usefulness. The result is that the form can be rediscovered, while the outline is described in much less detail in the image stream. Videos would be enticed to use such methods as resolutions of video and network traffic are demolished by such circumstances.

An aspect of the system could mean being able to limit and compress every single input of a page transmission with weighted results. Imagine gorgeous pictures reduced and reorganized such that more data is represented in less space. This is typical of sorting processes or algorithms that perform such mechanisms. Employing the same technology of nibble compression for languages also can be applied here for types of dictionaries that are formalized for color operations. The usage of color convolution filters is a basic aspect of the field of image processing. Yet controls of histogram, band separation, bit reduction

based upon band. Contrast and luminance variance reduction based upon area set distribution. Often if the cell is macro block oriented forward and next objects can be used as storing a sliding window dictionary cache. These technologies can strengthen image compression of new large HD images. Scaling is a very necessary aspect to reactive interfaces. Being able to allow more decoder time on modern pc usage can save bandwidth in network communication. to control variations as a necessity.

In the video cache category for a group of frames is most likely the appeal for CCTV or security camera input. The format adaptively stores and can manufacture a bit spectrum plus for any enclosed color within the dictionary of polygons. Reorganization allows the areas to be described by a localized compressor, while the assigned variance data is compressed as a large set. The individual tokens are very small and dense with adaptive changes. The amount of bit data per pixel, 1, 2, 3, 4 or more simply moves the existing pixel decoded into alignment. Imagine the finality of weighting statistically the importance of some polygons or group of polygons. The methods of JPEG's lossy weighting macro block network using DCT is also adaptable. Yet weight calculations must also be adjusted on variable sized shapes. Their group proximity and field of view primary targets are comprehended by the eye. A machine can be made to statistically weigh these with an intended purpose of strengthening color. While in areas of non adaptive ranges such as progressively large objects may be described in reference frames for a distant use in cache.

The set can be scan line based, based upon the color organization in several photographic materials of human culture. Worst cases are not typical viewing content. The data indexed part of the color projection is introduced on specific pixel regions while the image is updated. Bit image reduction increases the JPEG lossy compression by a large magnitude. Yet the tailored approach of information injection at a later time needs to be maintained within the image. Not only does this specify a field of focus about an image, or an image with targeted update streams for series of weighted pixels to use the new index in a numerical form. such selector of radial or gradient subset by a two bit always on index, the next parameters are attributed to value to use. So colors in a statistically recognized region are introduced with higher color variance. The overall system is very simplified. This gives the typical user a namespace for how they want to view an image. Such as zooming in or specifying the highest color detail necessary. The composition of modern form is that space is always available in the network stream. This offers a newer dimension to necessity.

The ability to run as an independent format is also possible.

Container formats:

string  
tree  
map  
database  
network  
vector  
program  
structure

Additional languages can be used to tie programs to data as a presentation using the forms below. The invocation of a dictionary change, a nibble, is followed by simply an index for now. The complications of expanding the format are more than needed. As such, the box already has enough data for character series to represent the nice features of an OS browser model. Some usefulness is gathered in macrocode blocks with a possibility of more combinations than a unit of 8 bits, being fine for a large unit of work. The transmission of such data should be more powerful, in that usual application functions of the device can operate on storage cached data more effectively.

```
/**
 * @brief The routine applies signal blotting to known formats.
 *
 * @param edata
 */
void cipher_t::blot(void) {
 vector<uint8_t> out;

 uint8_t cp = 0;
 auto b = bytes.begin();
 while (b != npos) {
 uint8_t o = get_latin_nibble(*b, cp);
 uint8_t o2 = get_latin_nibble(*b, cp);

 for (auto b : bytes) {
 }
 }
}
```

```

/***
 * @brief The routine searches for the given character and converts it to a
 * a dictionary representation. There are several characters that are
 * manipulated with special meaning. Also, the necessity of a rotation
 * a function to change the tree is present. In the cases of a tree change,
 * the instruction nibble is the same for all trees.
 *
 * @param ch
 * @param cp
 * @return vector<uint8_t>
 */
vector<uint8_t> cipher_t::get_latin_nibble(uint8_t ch, uint8 &cp) {
 /**
 * @brief these codes are handled internally for selective processing.
 * the dictionary exists.
 *
 */
 enum class nibble_codes {
 /*
 */
 ch_space = 200,
 ch_modulate_case,

 /**
 Methods of usual compression. null is here to establish end of data.
 */
 ch_repeat,
 ch_string_template,
 ch_dictionary_rotate, // should appear in all dictionaries.
 ch_null,

 /**
 data types and ui element. objects can utilize the formats.
 a container format for program, or dictionary.
 */
 ch_integer_t,
 ch_decimal_t,
 ch_image_t,
 ch_time_t,
 ch_audio_t,
 ch_object_t,
 ch_container_t,
 ch_element_t,
 ch_viewer_t,

 /**
 * @brief invocation of office software to integrate within the stream as a
 * data interface. The aspects of having data storage in location and remote
 * resources can be utilized, without sending the entire browser interface.
 * Yet the provisions of the system have very nice aspects. Even as a target
 * capable of using v8 for the complexity of Javascript. macrocode is the
 * promise of scientific source code compiling for trusted components and
 * work. The advancement of email, is its permutation to promote. A book
 * reader, academic presenter and library system that works well flattened
 * as a local codespace stack. The Nebula OS is a fine aspect of nurturing
 * more performance with the New Genetic Server Browser. As authoritative
 * experts inform the system of firewall protection in a more logistical way
 * for all parental concerns. Video calling from a link.
 *
 */
 ch_bytecode_t,
 ch_macrocode_t,
 };
}

```

```

/**
 * @brief International language support for the standardized view is almost
 * essential.
 *
 */
ch_unicode_t,
ch_utf8_t,

ch_daw_t,
ch_eot_t,
ch_eof_t,
};

/***
 * @brief output. the information is still in nibble format, but occupies a
 * byte. This allows a specific type of compression to occur.
 *
*/
vector<uint8_t> out;

unordered_map<uint_8, code_t> latin = {
 {'a', {0, 0000b}}, {'b', {0, 0001b}}, {'c', {0, 0010b}},
 {'d', {0, 0011b}}, {'e', {0, 0100b}}, {'f', {0, 0101b}},
 {'g', {0, 0110b}}, {'h', {0, 0111b}}, {'i', {0, 1000b}},
 {'j', {0, 1001b}}, {'l', {0, 1010b}}, {'m', {0, 1100b}},
 {'n', {0, 1101b}}, {'o', {0, 1110b}}, {'r', {1, 0010b}},
 {'p', {1, 0000b}}, {'q', {1, 0001b}}, {'u', {1, 0101b}},
 {'s', {1, 0011b}}, {'t', {1, 0100b}}, {'x', {1, 1000b}},
 {'v', {1, 0110b}}, {'w', {1, 0111b}}, {'y', {1, 1001b}},
 {'y', {1, 1001b}}, {'z', {1, 1010b}}, {'z', {1, 1100b}},
 {'2', {1, 1101b}}, {'3', {1, 1110b}}, {'4', {2, 0000b}},
 {'7', {2, 0011b}}, {'5', {2, 0001b}}, {'6', {2, 0010b}},
 {'`', {2, 0110b}}, {'8', {2, 0100b}}, {'9', {2, 0101b}},
 {'@', {2, 1001b}}, {'~', {2, 0111b}}, {'!', {2, 1000b}},
 {'%', {2, 1101b}}, {'#', {2, 1010b}}, {'$', {2, 1100b}},
 {'&', {3, 0000b}}, {'*', {3, 0001b}}, {'(', {3, 0010b}},
 {')', {3, 0011b}}, {'-', {3, 0100b}}, {'+', {3, 0101b}},
 {'_ ', {3, 0110b}}, {'\\', {3, 0111b}}, {'/, {3, 1000b}},
 {'[', {3, 1001b}}, {']', {3, 1010b}}, {'{', {3, 1100b}},
 {'}', {3, 1101b}}, {':', {3, 1110b}}, {';', {4, 0000b}},
 {'.', {4, 0011b}}, {'"', {4, 0001b}}, {',', {4, 0010b}},
 {ch_space, {4, 0101b}}, {ch_upper, {4, 0110b}}, {ch_repeat, {4, 0111b}}};

if (std::isupper(ch)) {
 auto it = latin.find(ch_upper);
 out.push_back(1111b);
 out.push_back(*it.second.cp);
 cp = *it.second.cp;

 uint8_t ch2 = std::tolower(ch);
 it = latin.find(ch2);
 if (*it.second.cp != cp) {
 out.push_back(1111b);
 out.push_back(*it.second.cp);
 cp = *it.second.cp;

 out.push_back(*it.second.ch);

 } else if (ch == ' ') {

```

```

auto it = latin.find(ch_space);
out.push_back(1111b);
out.push_back(*it.second.cp);

} else {
 auto it = latin.find(ch);
 if (*it.second.cp != cp) {
 och |= 1111b << 4 | *it.second.cp;
 cp = *it.second.cp;
 out.push_back(och);
 }
 och = *it.second.ch;
 out.push_back(och);
}
return (out);
}
}

```

This is the first part of the language process to code and design. The compression of language is also systematic. Remaining on an unpacked level for the next. The idea of leaving the items as a nibble, and scrambling the first category is necessary in primary format. The aspects of compression left at a nibble region. Frequency is applied across the 4 bit spectrum. The cut into the major signal noise produced by the dictionary change can remove a large number from the value. The expansion of key data is also reduced by the implied pop, push for a single dictionary visit. It is often developed that a link to a word is created. Using a single character within the sequence range of the window, to imply that abstract characters of the current dictionary are rare, the rare character is used to change meaning for a nibble less.

A function of index key generation using the space, or link descriptor, must provide that compression on index amount can take two forms. The aspect of encoding that the requested template for a HTML layout, with menu structure, and encoded book navigation system are described by a distinct numeric. As the internal system may be identified to incorporate any type of string processing or skip format for data to flow in a tabbed-like sequence for a template of text. The template system can be identified to be intelligent and also receive parameterized information from the stream. These functions left for research into providing object oriented templates that reduce data transfer by introducing more complexity in the stream. The template object has several characteristics that may be provisioned. Yet that of usefulness lies within the output requirements.

One way to reduce the index is by implied mode of operation. If they are pointed to within the template dictionary, are a host of unrelated template objects. If the template objects had a grouping category for attributes of the template, based upon the mode of the access. This reduces the index range. A HASH algorithm based upon the contents of the buffer can imply sorting. These aspects may be furthered after a functioning base encoder can be crafted.

The objective after adding multiple dictionaries, types and information such that it is a continuous byte stream as an object implies that the parameters be pointers or reference. The ability to decode blocks of information, such as that of receiving multiple types such as labeled in package delivery to the viewing device. The form of code behavior is different for client side only such as a pure javascript approach to existing data and cache. This offers that adaptive research in the gaming field for javascript has requirements for using type specific knowledge. The ability to code precisely through genetic output the functional requirements and knowledge of base cache as a server component can be coded in Java or even jNode, php or anything. An approach to using C++ is also relevant. often mixing languages such as C++, Python, etc the Nebula OS design offers this capability.

Perhaps the ability to send new language definitions per transmission or requirement is a necessity. In these respects a planner may formulate better protocol methods of user interface, objects, and connectivity with the language input of the input. Languages from Unicode, One character, offer to implement much more output of code for the word it names in chinese. Is there a Chinese programming language like c++, templates, genetic, and robust expressions? The functions of the word random never betrayed to name something other than the best.

[https://en.wikipedia.org/wiki/Shannon%20%93Fano\\_coding](https://en.wikipedia.org/wiki/Shannon%20%93Fano_coding)  
[https://en.wikipedia.org/wiki/Data\\_compression](https://en.wikipedia.org/wiki/Data_compression)  
[https://en.wikipedia.org/wiki/Lempel%20%93Ziv%20%93Markov\\_chain\\_algorithm](https://en.wikipedia.org/wiki/Lempel%20%93Ziv%20%93Markov_chain_algorithm)  
<https://www.7-zip.org/>  
<https://pmwiki.xaver.me/drdoxwiki/index.php?n>Main.Compress>  
<https://github.com/dwyl/english-words/tree/master>

## Video Encryption

Video encryption is a major field of study for copyright infringement-protected storage. The sources for the browser, Firefox, Chrome, Opera, and others often leave the DRM component out of the source tree. When thinking about the surface of

watching video on the network, there are several things to consider. How long the movie is, what is the projection ratio, color depth, resolution, and encoding format? Videos may also have a closed caption exposure file. A very important aspect is how the video is going to be used by the user.

A user watching an hour and a half movie film can be expected to exhibit traits of watching as per history. For example, given that movies can be paused, stopped, and played at specific times during the user's life, one would need knowledge of these operations before choosing the best method. If the intelligence field uses videos from security cameras, typically very acute time windows are inspected, requiring high definition. At the same time, motion or field change is incorporated into many types of CCTV systems. Often wireless cameras are integrated into the system for remote publication, analysis, and storage.

A streaming mechanism for video is very dependent upon the video and audio codec's necessities. As a separate functional data structure, the audio and image data, after compression using DCT Mpeg lossy or other, depend on the amount of data being available at a specific time. The amount of data is typically determined by a non skipping measurement that is based on internet traffic.

Video codecs are specialized in some platforms to encompass network communications using an "over-the-top" method of transmission. The natural and prescribed transmission realm of the internet is known to be TCP/IP as well. Yet, messages can be also sent upon the network that is unreliable for stability, and order using the UDP method. the format does exceed in areas where software is running on networks that are nearby, or within a belt of network routers.

[Video codec - Wikipedia](#)

[List of codecs - Wikipedia](#)

[Color Cell Compression - Wikipedia](#)

[Digital rights management - Wikipedia](#)

[Videocipher - Wikipedia](#)

## Audio Encryption

One aspect of viewing audio data or looking at values, which are typically integers, is that the reverse of information is medium-specific incarnate encryption. Within a buffer, for example, a distinct swapping of sequential time data, to interleave elsewhere. The aspects of hiding the fact that the amount of data is a numerical wave format. Previous and next numbers match or are in close distance. The offering that audio is a source, that was digitized just in the eighties, can have multiple functions that persist to describe the quality and format. The most useful off-hand is the PCM format. CD music uses the format for audio. While it is not compressed it is essentially a vast array of integer values, 16 bits wide. Making the range -32767 to positive 32768. At 44.3 kHz per second, that is a large bit of numbers for a five-minute song.

There are multiple ways that audio can be compressed, yet as a functional leaner access by time in decoded format, if audio has to be transmitted in real time, functional requirements of the signal can be removed. Often

[Data compression - Wikipedia](#)

[Sampling \(signal processing\) - Wikipedia](#)

## Logic and Program Encryption

A formal reason behind the web browser is the reduction of the computer virus. Computer viruses of yesterday were much more capable in their ability to install themselves. In the last age of MS-DOS, programs would simply attach instructions of their logic within binary files of the OS. Or integrating themselves into the MBR, Master Boot Record. As new kids on the block, software engineers with distributed versions of code had major problems to contend with. Yet these problems within an audience often would be unknown to the user for a duration.

[John von Neumann - Wikipedia](#)

[Computer virus - Wikipedia](#)

The modern computer is designed to let viruses in. Often in the marketplace, having such a versatile device is not necessary for consumer usage. That being said, even with such an open-ended framework enabling the booting of a computer, is only really leveraged by a few operating systems. The major field is Microsoft Windows Distribution. So attackers poised against the platform, to enable many far-reaching critical investigation units, and problematically instantiated a severe market hurdle for computing companies. As a tactful strength, the types of planning work at the residences of Microsoft Corporation were in a respectfully sensed reaction. Encumbered by the strategic outlay of a social revolution, invested in disrupting American government facilities, worldwide hackers used their methods according to a psychological realm - outplanned by newbies. There are very few military respondents in Microsoft technology meetings. Two separate fields, companies, and focus.

Computer professionals do make a significant amount of money to consider the many other fields. As a target, they are now on a plan for which they do not control. The products they work on, and their position as a type of order are distinct. There is a

hierarchy based on skill, knowledge, experience, and also office coordination. Yet in their momentum as an employee, they are compromised by a large data set of information. As simple as a team member, must adhere to the ethics of formalizing code that works, solves the problem of a told requirement, and be responsible in other stages of product deployment. The in-the-dark approach is always present, as typically tens of hundreds of thousands of people control technology. The interface of this group is representatives of the cooperation that may converse with other parties.

The ability to out-plan organizations of work, where arriving at a result is based upon a larger chronological time of external development. People must depend on select products being free of errors, or the ability to be untampered with. The result is that the composite of the running user experience is held within the web browser. The entirety of the design was not established as an operating system, but to leverage a competitive markup. The division of the team's leads many astray to be emotionally competing on levels where they should not be. That is ultimately one, two-stage approach to invoking a planned revolution is seen to be blistered in a type of technology competition referred to metaphorically as "war".

The prospect that all employees of such organizations have behind-the-scenes work is fronted. Merely, it is a type of audience. In this audience perhaps the jealous bit of a bi-weekly check is rooted in passerby conversations. Yet the distinction of modern experiences is that filth of work is centered around propagating these attitudes as a division. The appearance of which can be very precise at false or very planned finger-pointing. Art propaganda for example is a reactionary yet evolving field that captivates industrialists. The cavities of the mind open for filling.

### [Bytecode - Wikipedia](#)

Bytecode is the most plausible at being a protected resource that can be run from network sources. The inference that the code cannot wreck or corrupt the machine is based on the survival of the supporting platform. Ultimately the most versatile machines exhibit the form of narrowing basic logic and data processes for programming logic, expressions, and comparison. To improve the mechanics of the format is selective in that more generalizations can occur. Specifically within the data type range. Numerically one format range of integer, or floating point provides capability. Types of inferences may also be negotiated for other instructions needed by conversion. Many machines are overly designed for purposes of direct interface UI response. Often, the attributes of the browser and its capabilities are never used by intelligence professionals. It severely depends on the interface projection or software usage. It is also known that these technologies use the UTF catalog of dictionary characters to align server and client communication.

The ability to minimize language of formula, logic, and interface requirements is a target of programming technology. The field may be invoked at many layers of progressive thinking. Often the best practices of organized problem-solving go into object-oriented designs. Such that the system design handles a type of generalized solution problem-solving. The aspect that this is a giveaway in deciphering it. For example, a MSIL codebase, or Java Byte code base will provide a dictionary result. So the aspect of minimizing the bytecode to a more reduced model is possible. Ultimately reducing data trails using compaction is rewarding. To think in such contexts, if we think of every computer as a remote island, the transmission mechanism between the islands should be coded for the long run. Providing a very sound design in base principles shows that while minimal, it can be compelling. Ultimately byte codes have two or three sets of everything. While machine code generates about eight to ten times the complexity in draft fans. My estimations perhaps in words.

The generalized capabilities of the engine to produce dates at any prescribed method on the fly always exist. Is it a necessity? The combination that the nature of the machine as a smart client is packed with functionality also reincarnates existing software already present. The provisions of tailoring the device with better object linking for device os production can be enabled better through better table of contents-oriented object systems. The static link always felt safest to me. That is, the complete linkage of all necessary binary module libraries is accounted for. Much better software os and hardware can be created for both intelligence and consumer-grade technology.

Another aspect of capability is sending program source code through encryption channels. The source code is symbolized in an index data erasure method, while also textual literals are treated separately as a function of a parser. In this circumstance, perhaps the codebase is part of a system where codebase languages that are developed internally are used for specific departments. For example, a new component of a genetic encryption leaf. While the dictionary method is used to wipe away symbol brandishings of logic words, some languages have tedious use of tabs, spaces, or braces within the context. The weighted tree for the four, five, or six-bit tree would have those near the top.

The capability of removing other annotations that are human concepts such as variable names, class names, source file names, routine names, and comments. Except for comments, others will be encoded with a symbol encoding for the bit range. Some programming languages have other attributes that may be learned. Line numbers that can be stripped away. Carriage returns may also be required. Most modern languages have a distinct line continuation solution. Tail to statement start is used in JavaScript minification. Comments are blocked together and encoded separately as an English, and limited media form. The programming logic source, the same level of post-transform work after index, keyword signature, user-named entities, and literals as the Language Information Encryption section must be invoked. That is, shuffle the identity meaning of the characters and their position within a line of source code. Relocating each byte of a word or index. Genetic encryption is applied to sections. Comments as designed complexity, are useful to some audiences. Using a dictionary method can be useful. All components are composited as a signal, and genetic bit encryption can be applied. To and from is

established by the server protocol method. Perhaps too much complexity in server processing, yet as a publication source code for any data system is published. The Microsoft Source Safe or git repository.

[Source code - Wikipedia](#)

[Programming language - Wikipedia](#)

[JavaScript - Wikipedia](#)

[Python \(programming language\) - Wikipedia](#)

[Java \(programming language\) - Wikipedia](#)

[C++ - Wikipedia](#)

## Map and Diagram Orthographical Data Encryption

The ability to store and organize a map system is legendary in the realm. Yet if we introduce the existing system they are very large. Some means as simple as a single image can reduce necessary targets. To inspect the domain closer for intelligence maps, this may take a more strategic look into providing a versatile format. The ability to consolidate a sliding 2d window with icons, dynamic scaling images and zooming information. The process of LOD.

Map making of this sort is investigated by numerous methods. The context of maps entails the most informative region of road car travel for most. Often the larger systems have satellite images. The aspects of reducing views of satellites as a LOD function can polygon fill and label buildings with drill down. The aspect of using a map system with LOD, infers that its method of reading establishes a human form of interpretation. Such as distance or volume. These values are typically related by numerics. Speed implies the object is traveling such as a robot on a floor plan. Given a map, and route, the robot must be able to distinguish routes. Hopefully in a way that provides real time usage of current work.

The environments that are not on the map often need more details that a computer would input. Such as the map of the house, or an indoor environment for work. These robots are expected to function without hazard to coworkers that are about as well. Logistical knowledge on locations of people may help. If they are at a team meeting, the robot carrying the payload of wood for speaker cabinets would choose an alternate route.

As a human readable map with programming capabilities to robots, because it is a well defined technology it is useful. Therefore making sure encryption is also secure for these communications per robot allows great effectiveness. That is not having a physical accident. Robots are the work of the computer in the world. As a strategic compilation more problem solving is manufactured by knowing the stages a potential product will be in. Transportation and also loading in line is necessary. The functional potential is that an overall planning can drive production better. A system where specific handlers are ensured safety by their materials handling capability. In many circumstances, such as industry, battery power or cord overhead. Hopefully not getting tangled if it powers a heavy load bearing machine.

Encryption in this aspect makes sure that the robots do not stop production at the occurrence of hacking. A system where principles of manufacturing for food items, robotic underground greenhouse gardens, and product manufacturing plug and play commodities offered as robotic leveraged operation.

Maps also have a leverage for a distinct area. Typical cars do not travel outside a radius for their common use. The other aspect is continually rolling around on the continent as a truck would. The final case is sporadic use of spatial data that is distant. Spatial data can also be compressed using the means of distance delta on objects. The weight is strategically updated as a LOD value based upon the tree neighbor. The provisions of lossy map information are often very tolerant if a human is only reading it. The main problem is describing an array of objects that occupy a space. The coordinates and field of view change upon an orthographic context and is easier than simulation free form graphics.

With a numerical base of coordinates, and measure, the implications of range reduction of aside fixup values that flow in can decrease package size of map data. Later transmissions coordinate artifacts of annotation. Often in maps, the picture prime perfect approach is less refined than a vector line drawing. As in survey concepts where units of measure can be imposed for working with the map data. Calculation of miles at sea and also construction sites. Icons for contextual legends of map data to be refined in the terms of an organic picture cache. The nibble data sets may work well. The terms of a short SVG, is often found in the implementation details of the SVG engine.

The dataset approach is to save large amounts of drawing while also having the LOD resize. Often very detailed SVG are the product of working in a web art drawing program. Inkscape is such a product. The SVG format, as a dataset, encompasses many forms of information that must be reduced. Reducing some data sets algorithmically is also necessary as some tools when converting images from one format to another store the line data and not the curve data. That is the curve approximation is unfolded in the data. Several variations on data sets can make SVG data lossey for numerical range description. At times there is an integer math system that can represent decimal work. Numerical compression is the formal procedure that already exists to employ.

[Geographic Information System - Wikipedia](#)

[Geospatial intelligence - Wikipedia](#)

[Geotagging - Wikipedia](#)

[Google Maps - Wikipedia](#)

[Vector graphics - Wikipedia](#)

[SVG - Wikipedia](#)

[3D modeling - Wikipedia](#)

[Lossy compression - Wikipedia](#)

[Data compression - Wikipedia](#)

[Orthographic projection - Wikipedia](#)

[https://github.com/oreillymedia/Using\\_SVG/tree/master](https://github.com/oreillymedia/Using_SVG/tree/master)

[R.U.R. - Wikipedia](#).

[The Steam Man of the Prairies - Wikipedia](#)

## 3D Model Encryption

[https://en.wikipedia.org/wiki/3D\\_modeling](https://en.wikipedia.org/wiki/3D_modeling)

[https://en.wikipedia.org/wiki/List\\_of\\_file\\_formats#3D\\_graphics](https://en.wikipedia.org/wiki/List_of_file_formats#3D_graphics)

## CNC Encryption

<https://en.wikipedia.org/wiki/G-code>

[https://en.wikipedia.org/wiki/Canned\\_cycle](https://en.wikipedia.org/wiki/Canned_cycle)

## Office Software Formats

[OpenDocument - Wikipedia](#)

[Microsoft Office - Wikipedia](#)

[Spreadsheet - Wikipedia](#)

[List of spreadsheet software - Wikipedia](#)

[Online diary planner - Wikipedia](#)

[Borland Sidekick - Wikipedia](#)

## Bit Relocation Using Genetic Encryption



Ultimately as a range of 0-n ( $n =$  number of bits) possibilities, the relocation of an individual or group of bits lies in combination with swapping two. In the primary aspect, one can see that a mechanism must be in place with logic to reprint the original index values and place them back into position. One way this can be accomplished is by using pseudorandom number generators. The dynamics of some systems are simplified whereas they accept a seed value. Subsequent requests for random numbers will be the same series again and again. These

are also referred to as uniform random number generators. There are several available. The most powerful implications are in hiding the aspects of operation by code, or disassembly while relying on the stream to be produced using it.

Yet as source code must be published, it is impossible to protect better with source code published. However, with a large set of operations that may be formed and changed in order, variances will be difficult. Ultimately the proof is in the ability for a system to be tested, and measured, and also when more of a spectrum of code is found, finalizing inceptions of how to make a subset, running on a production code base, perform better. One of the greatest kicks about the philosophy of the code is that ultimately it is a type of useless logic performing only a type of scatter. A data lock mechanism.

When the c++ syntax can be easily manufactured for continuation and have traversal of necessary components and operators. the permute() is necessary to complete the instruction. Perhaps there are other codings provided, such as links to rnd variables that are also prefixed with rnd\_[id] and regulate the amount of control on various random number implementations as a class manufactured structure. This can also be accomplished in languages such as python in a smaller space of code with.

There are multiple aspects to utilizing the operation input. Since ours is a test best to establish a domain of a genetic system, tools such as these are sufficient in that the structure of the language can be adopted to perform other functions. Ultimately with multiple parameters to inputs, the routines themselves become a catalog feature. Producing multiple types of stream that are coded for execution apart from showing the operations is one such method. Byte code or nibble code.

Designers will study the forms, and ensure a compartment view of formal justification for using it. The growth of an intelligent conceptual chain implies further research. The function name is a catalog identifier. The class bits\_t is an easy type of component that is fun to design. An entirely new bit of the world.

Imagine the object instantiation as formed through literal code onsite for an accounted use. Use of the entire system implies through instantiation that formal operative formats are constructed. As a carry context, objects with structures inside. Or vectors that are members of a std::variant. The implications of using the variant are tuned consolidated dictionary lists and function lambda. The API context has also a permute function, such as std::visit.

The combination of plugin interfaces between data layers form to cohesively form around other integrated syntax. Although available, many tend to use some and explain through logic more the process rather than looking up the new combinations. There are functions such as std::generate, and buffer algorithms, for achieving a very consolidated syntax on output. The inferences of study, often lead to a generalized introduction to how the local system is applied, and then asking bing, or search or example code. Often the best code base for study are browser source code bases. Could any tidying be done to summarize the componentry and data flow of the components to be faster, or described in less code.

Ultimately I think that interfaces of a more common language should be adopted, yet often are necessary for the C++ compiler and programmer to work. If a class is named SKLines, then the prefix ultimately gets in the way. The best nomenclature as of yet is using the standards of older programmers such as “\_t”. The system derives loosely while the older style of class and prefix with C clutters the reading namespace. Some types of capital letter tokens are good if they are part of a system for recognition. Two characters with the two words can be liked or not. For some the aspects of capitalization and using a space for variables applies better. That is typically the versions of Microsoft Source code versus the open source community.

Yet this is not the project of the browser, a component for multiple uses. Tailoring knowledge on what is known about previous research efforts. A multiple examination of coding style is typically introduced in the German coding scheme that also knows English. Making literal information typed in means better things are partially connected with the namespace use. The enumeration, class enumeration and also inner class class enumeration, provides a list maintenance mechanism for structured options. As a function of template programming the capability for the const as well as genetic coded if statements provide the facility of excluding the code. The system of const compiler encoding is a piece of technology that is coincident with new orders of performance from the composure of object machine code.

A system I use for this is mainly based around half plotted principles such as structured logical steps. Many steps that have been accounted for or are handled by outside sources can be muted with a plug in API. Such that the concepts of the total composition and features are gathered by all types of creative thought. As a will, the initial developments remarked of in a classification of object form, and then proceeded to encompass parts of the acute language venture. The steps such as decreasing system functionality and getting to the chase. As if waking up early in the morning with a basket full of candy. Or rather how fast can such a system operate without crashing. As component, and structure layers, many of the concepts in object oriented mechanisms can be established for real time dispatch of algorithmic handling using the is\_base\_class of object classification. When used on streaming inserts, such as those well thought out instructions that are minuscule in providing realization, become a compaction of all the code necessary according to genetic template instantiation. That is, a narrow inline function for the need. Take for example the code listed below as a genetic cycle round, The object system described as a mechanism of inline math inserts, without object chain calling.

```
void HHDGE_33k3ej3ee888374_ABSSSDGY_INLINE(bits_t &_b) {
 _b.permute<shapes::rnd_polygon_large,shuffle::swap_332 | noise_low>(rnd3323, rnd55754, rnd8855, rnd885);
 _b.permute<units::nibble, shapes::circle, shuffle::swap_752 | bit_oper: rotl | bit_oper: xorl(rnd4434, rnd34343, rnd434);
```

```
_b.lock<units::nibble, noise::lock_residue, storage_hash64>(rnd_54,rnd_76,rnd_44585843);
```

```
}
```

The aspects of template meta programming seen above can be described by code as applied to the mechanisms named. The ingredients are dynamic. So these routines will be relying on internal information. Within the member template function permute, the parameters induce shape first or unit size of the transform. The aspects of classification within the system to include enumeration means that the system of decoding is merely represented as well as a code. As a type of variadic parameter list, the param pack mechanism is employed within the traversal of the codebase. The effects of the code bases are instructional targets of bit flags. These can be stored in a const array or other vector data inserts. As a matrix, the high order bits establish a domain of the flag while options are stored within the enumeration. Through consistent coding, the entire matrix can be fashioned as a fitting permute for the options. The random numbers become parameters.

Template classes are also available for a dynamic production facility. This provides much less code to write as each of these functions are adapted. Did you know that the type of data for rnd\_4458843 is a random object described in the AI sensors region as a double, whose frequency is stored in a cryptographic layer received from the server yesterday? That is the implication of using the mechanisms.

The signal is decoded to integrity of noise. The hash is the modulation reciprocal of the process.

The shuffled parameter, shuffle::swap\_332 provides the pattern matrix for moving or swapping data. These have a design weight as they can compute the number of moves. The order that information is moved is unique in that the mechanism achieves a distinct operation result from a measured based spectrum of data. The parameters, if supplied in the exact configuration as before, have the impact of reversal. In the implementation the variables are presumed to be efficient as parameters for a purpose. Such as data size, range, and system output. The trust that the number is the number should be integrated. The bit\_ball\_prng\_t is designed to mechanize a large array of random variables with more security as a bit\_ball\_key\_t stenographic client.

Noise as measured for production dynamics. A system also may perceive a continual production of noise, or period skipping with added noise to provide adaptive decoding principles. By only knowing the permutation of noise, and its dither interlacing, the signal proportioned is unknown. Noise adds complexity yet the amount must be determined logically. If the system codes that it should break down ten megabytes of data into 25 bit triangles, expanding each triangle with a large amount of noise is improper in the decision of scaling the data too large. The algorithm set has nine options. Including buffer buildup from prime phased information. The residual and mechanism balanced for controls and default settings. Types of recurrent multiple phased modular math and oscillators can produce some signals that are known to be mixed. The logic between the mix parameters and data adapted to the wave spread. The implications of finding the keys between the two found elsewhere. As a dictionary and hash index, the system could be required to sample signature points from the distribution of the mix and use little data for a noise path. This both instructs the mix and also removal of the phased noise. This simply introduces depth control and large problems when decryption is investigated. Typically this may not be required. The genetic production engine has established rules based upon statistics for system control.

Therefore the amount of code required to produce such a system as a whole drastically reduced with template meta programming. Planning for a systematic inner workings can be conceived as a plan through random examples. When described the internal architecture becomes very simple. Often the first parameter, when reduced, is based upon the default unit modifier of bit. Changing the unit modifier increases algorithm performance but also can store more data per payload within the shuffle aspect. The spread process is adapted to the unit. As a whole the inner algorithm logic operates on storage and index location. The ability to modify these traits for indexes can proceed in many ways.

Typically these are small little programs that are fun to write and without a direction of sorts other than trying out a net set of indexes to the data. Data can be divided to show a group such that the indexes would form a rectangle. One may think of operations to shift things around within this rectangle by a decision from the random() function. One aspect of the seed meaning a value, is that the same range is chosen for the request. Typically this is accomplished by the modulus operator or other means of number generator creation within a range. If given the rectangle, also a cube can be formed,

In openGL, three-dimensional triangles are seen as an index to vertices, the order tells of a winding order for defining which direction applies to the outside face. Similarly, the filling of bits within a shape must apply an order for the winding. For a two-dimensional right triangle, resting its angle on the x, and y axis at 0,0, has a distinct width and height also. These attributes are the number of bits. One can fill the pattern such that a bit exists at all points within and upon the shape's form.

In this mechanism, the imposed cartesian coordinate system references a range. Forming an x, and y axis can be virtually created even as centered while each quadrant is represented by negative and positive values. Circles formed from many types of painting algorithms, even as a spiral, can form gates of logic for a series of transformations. Developing line traversal coordinates using the Bresenham series and also curve is easily adopted in source code. Curves can also be directed and usually are in rendering to be the smallest straight line. There is some kind of wild attribute of making it a perfect-looking curve on lower resolution. But these are not resolution images, just data bit maps. So even a random number of sides to the figure, and processing for grouping triangles to the center offer a type of operation. Using the cos and sin in a rotation matrix is applied in a very small amount of code. Some people can use radians too, but I still think in degrees.

The primary aspect of forming a shape from multiple accounts of one takes the form of guessing multiple useful attributes of the shape. The encryption phase takes place by using relationships of the shape, random numbers, and swapping bits. Well-thought-out plans, elusive and detailed are the task at hand.

Using sub-division methods of shape. Interchanging geometric chunks of the same stamp. While stamping modifies each chunk with a localized method based on the dynamics of the running algorithm. When using subdivision, each layer can be organized as a unit in the Z range to be from distant data. Or odd / even filling of noise data. The implications can be also found in types of more mathematical relationships using triangles and squares together, where accesses of the same data are built for data movement and transformation. Mirror, rewinding of the bits. Ultimately a size and number of operations will suffice. The ability to provide two or more of them and produce a variety of data mixture distributed between them can magnify redistribution. The main ingredient is within the access and grouping of the data for transformation.

Building the shape with the smallest amount of bits can provide scaling features for some shapes. Triangles, or any surface that can be added to by simply enlarging it. The easiest enlargement with a three-bit triangle would be to simply add another row at the bottom. Yet in forming these data, the input is a single stream. So for example, a right-side triangle, with seven bits, will have three rows. The top bit, at (+2, 0), has only one. The next row has two bits from the stream located at (+1,0) and (-1,+1). The third row would have three bits, located on the X axis at 0. (0,0), (0,1), and (0,2) are the points. The interior of such a routine would need to initially be allocated a cartesian coordinate system. The object contains a several geometric drawing API that is specific to the data stream mechanics. The API, while simplified, is not a verbose vector library as rendering. Bit representation does not need the complexity of aliases.

Another type of performance for the conclusion research on a production algorithm, would be to find a subset of transforms for a given size of triangle or shape, and store the logic. So for example, if an equilateral triangle of bit data were measured to have 43 bits, what would be the area in bits? Still 43. Well, how can other attributes of subdivision be used to identify components of the interior shape? One knows of the shape, the pattern normalized a 0,0.

Another interesting aspect of hidden data transforms is if multiple shapes can be described, by the state of a decryption, the process may have to deduce how the pieces fit geometrically before continuing. So the aspect of a type of leaf fitting to the tree must occur if it is going to have a statistical reading of x. Using a triangle subdivision method implies data must be aligned before continuing to meet the box requirement. Often knowing that a problem such as this needs to be solved is difficult to ascertain after spending so much time using other methods. Miniature puzzles that must be solved to make the shape of a pyramid or cone. Single hinting input so that a specific property or arrangement of lattice structure gives a cross-section reading of n. Perhaps several variations would work, yet iterating through understood virtual pieces offers that the mystery is real. An aspect of the two kinds of random numbers working together, predicted and pure, can be used. Randomness in arranging the virtual subparts. An aspect that may be chosen mathematically is a guided side and shape permutation limitation to reduce yet offset solvability and time adjustment.

Coordinates and groups are often selected by current algorithms so they may choose several small groups of bits called nibbles. Nibbles are four bits in length and have a relationship to a byte. Yet in plucking one out at a position, there can be distinct rules in extracting it or putting it back into place. The advanced algorithms use methods that also tangle and fit bits at a coordinate in the byte. If at a boundary, 0 or 4, then an alternate weight is placed on the other side. The most tedious aspects of where these two go by a chain of mixed-up logic. That is the two to swap. The performance guild is that more than two be shuffled in a sequence of distribution. When the frequency is proportional to density, algorithms are clipped to have a credible ratio of bit encryption movement. Balancing noise interference in granular layers of encryption, staged, and expanded several times is more effective.

There can also be compression algorithms that are useful in reduction. Some types of block reduction in encryption can be bit pattern dictionary-based to check for the most persistent pattern of a variety of numbers of bits. This infers that as compressing binary data, specifically all patterns are reduced in the signal. A terrible practice that is introduced by such items is that it introduces coding signatures where the plug-in data must be in effort. A minimum impact is a left-right tree that replicates the pattern. One aspect of a tree of small patterns is that its depth is shorter. The number of options to resort to the tree, to balance in a form where more bit patterns are produced using fewer walks can be useful. As a permutation index, a distinct type of transform can be encoded within the encryption stream while the data is compressed using the communicated bit encryption tree. Specialized by a shorter tree, unused coding bits can be functional as instructions. The right tree spin can be introduced to select new output characters. The set instruction could be favored for a select number of uses, and then pop through a rotation-based stack back to the primary output. The operation could be summarized, by using a dictionary tree for Latin characters, while the next encoding will be from the HTML dictionary.

For example, the nibble has four bits. 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111 are the only options. By selecting first the most frequent pattern to be representative of the least encoding bits, as a Huffman encoder would use some fractional space can be minified, 0000 can become 01. With two bits to represent, there are four combinations available. As well, if a tree can be changed where a weight is placed on the frequency attributes as present, built into the engine, communication of the tree balancing can be parameter-based. While it is known that a half-weight approach is necessary. Also in some data, the approach may not be useful. By using one of the values as a compression dictionary change command, the next bits in the stream can inform the decoder. Important is that the compression be designed to run as specific layers within the chain. While having the ability to properly compress most data lies within the original form, these compression algorithms work on dense data adaptively also.

Once a stream has been normalized as partially phased encryption, components of the stream can be locked by other parts. That is, using data from another segment of data to encrypt another examines both noise and complexity. Data that may be used to encrypt itself can be regarded as a distinct locking operation where the methods of cipher phrases are both dependent on the genetic phrase and the data scope layer. As a process, preceded by the signature dampening codecs, its

introduction is at the initial data cohesion form. Stamping a layer with encoder information is accomplished by distributing the parameter bits across the entire stream. Depending upon implementation, the information is related to the state of the algorithm.

The aspect of bit storage as a structure is also applicable to other types of encryption formats. The conversion of the data is very simplified. Some elements representing bytes such as described above, will be signaled to operate at a specific time. Such that removal of bits, or changing of bits must be represented structurally. The simplification of the order of bits, as a secondary key, can be activated by sorting the vector in ascending order. Then fill in a byte according to the pattern.

Aspects of blocking individual data types for binary can provide storage transformations. For example, in 24-bit color, or also transformations to HSL, an aspect tolerance of brightness and contrast can be used to reduce image data. A type of greatest common divisor for the set, and histogram information can apply for a large data set. The implications are that it is more lossy, but still has HD-quality resolution. The aspects of vector drawing, with shape cache references, can be integrated for several iframes. Since the item is a geometry, filled with a texture according to the scene, the texture must be identified in future scenes. The typical context for an encoder does not provide this type of intelligence, rather its focus is perhaps fifteen frames or more. When more genetic software can be run upon the stream, more can be discovered. The decoding format is still popular in the ability to be viewed, yet encoding costs considerable time to compute.

For example, color palette references in symbolic links can provide palettes that compute the loss values from hinting position maps. Hinting position maps reflect a minuscule that can be applied to narrow the distance to the original target color. This makes the color referenced mean several near colors. The color-specific controls adapt also to identity with shape.

More research into the placement of processes, order, shifting, and shape transforms to form a mechanism of meaningful locking operations. The <https://cryptopp.com/#download> download code has existing block algorithms that can also be applied using the effective permutation of const if. Some of their names would have to be documented as part of a scientific patent. These algorithms are open source scientific implementations from major corporate companies. [Comparison of cryptography libraries - Wikipedia](#) provides a comprehensive list. The desire would be to familiarize the dynamic template design with the existing code algorithms.

A seven sided fold out cube, consisting of a distribution of a one plane of bits. Composing a stream where the seven sides, can be interchanged or mixed with other. As a recursive nature, the cube can be filled with a two bits less in height. Projecting existing geometry such as solid cube noise of various sizes. Apply an arrangement of live data within the cube that will fit though a series of network AI decisions for puzzle solving. In the missing gaps still left, fill in with noise. Perform a series of relocations as a solid cube filled one hundred percent with bits.

The geometric shape can be described in angles. Often the values of polygons are related geometrically by properties of math. In forming shape, each unit represented as a fill shape will have a known set of bit compaction inside of it. The scaling functions for bit compaction may be effective functions already known of by mathematics publications. For example, the Hyper Cube families using irreducible polytop families create an intersection set that may be referenced between two. For example, imagine if a range of progression is a small type. Such that filling its structure is a function of a 64bit number. The scaling function of noise also provides a tolerance of precise filling in bits. The difference of which is a type of payload information within expansion such as mixing dictionary or other live information as a package. In these cases using scaled functional interior representation of internal subdivisions, can rotate, mirror mix or swap, as well as use the properties of a Weyl group relationship. For example, the Tetradecagon has mathematical relationships that are permutation oriented but require transformation. Fulfilling the methods of dissection is quantitative in the unit of one bit or unit filling. The dissection into rhombs allows further investigation. The area functions can also supply input in cases where where scaling is simplified by known values. Heptagon area is calculated as  $3.634 \cdot a^2$ . By allowing the input sizes to be classified for input requirements of a given area, that is the amount of bits, or bytes must match. This is much more simple than a Blum Blum Schub, but must be deduced.

<https://cryptopp.com>

[Huffman coding - Wikipedia](#)

[Huffman coding](#)

[Polynomial - Wikipedia](#)

[Trigonometry - Wikipedia](#)

[Shape - Wikipedia](#)

[Glossary of shapes with metaphorical names - Wikipedia](#)

[List of mathematical shapes - Wikipedia](#)

[List of two-dimensional geometric shapes - Wikipedia](#)

[List of triangle topics - Wikipedia](#)

[Tetradecagon - Simple English Wikipedia, the free encyclopedia](#)

[Petrie polygon - Wikipedia](#)

[Binary coded decimal](#)[2D computer graphics - Wikipedia](#)[Least common multiple - Rosetta Code](#)[Greatest common divisor - Rosetta Code](#)

## Hardware Identity as Password

In a network, the provisions of labeling, with codes for business objects, as well as photographic recognition of printed codebooks. Code books may represent binary information in multiple dpi produced ways. Providing a labeling system that is not readable as also an undiscoverable catalog feature. One obscurity is that all such labels first have a visual representation, typically, a dictionary is used, such as arranging into a square cell with bits on or off. As a function of encoding, more information within the label should be present, but not too much more, and applied encryption to a database system for the key. For consumers, a transaction based daily TLS for fun art labeling can be used in their home. So cracking the cipher does not register the device to the network of digital data disclosure. Perhaps, several accountable facets of information are summarized, and then summarized. The ability to refine the amount of paper to a solution, ultimately is necessitated by indexing several together.

Imagine in the office of the sometimes present, occupants have locks on their desks, office, and furniture inside. The capability for the lock to have an internalized only physical lock, activated by reading the digital lock on a specialized smart watch, or long lasting embedded device for the inner pocket. As a visual camera input device, the sensor can be integrated with the key device. The key device has 16 megabytes of protected long term storage in a compartment aside from the other electronics. It is the capsule size of human liking such that it is durable and emblematic of what it holds the keys to. Their ability for it to have a photographic lens, also applies to a specific field of view. The device can be used both for wireless keys and photographic keys. The programs for decoding the visual cipher would be best if they were generated by a computer that can know of the 16mb key attached.

This makes the locking mechanism more secure whereas the generalized shape dictionary, encoding posture and transmitted permanent bits for a lock are only persistent with the key. The problem would be that if you sell your car, you have to paint over the side door where the codes are.

The software that reads the code is a personalized version for the client device. In consumer terms, these are QR codes that are much smaller in definition. The scanning process is based upon the camera and information device that is not connected but has software for recognition of the area, mapping of where the device is, and owner attributes. Most computers of data investigations are mounted at desks. The riddle providers use an array of hardware, phones and all types of made up things. The aspects of newer telecommunication systems do not redefine some older technology. But as a device ownership method, simply it is a physical device that a server software of registration would know of.

Representing the information in less space when visual format is provided can be a function of reducing the inner data type. For example, the value of having only one, of a record id is necessary. A method of placing standardization is by classifying the reception of target bits as a viewable allowed for the installed codec. The ability to classify the codes or unique hashes can be transmitted from device to device in proximity as a registration process. Necessity of having these types of codes in meeting places, at each desk label, employees can show attendance records at planning meetings.

Using color can decrease the size of codes that have a large amount of information in them. So a close up picture could be had. As a work of software, the code establishes a known start of work for the device. Most likely, presentations and secure data server sign ons. By having a smaller and dense color format with also encryption to the unit tag for the data server. A laptop connected to a known wifi server in a room can be registered privately in the data sources division. The provisions that a tie between the laptop and target records or domains of information are tracked. The registration process can be portable.

One aspect is to note through grid position what a value is. As a measurement where the reference is located within a domain, such as greatest common divisor of an integer, By using a representation through gradual size, along with palette and color ordering, can show fuzzy designs.

As well, one polygon figure, if unique within a field of camera view, and shown to find the values of tolerance to be realized as a form can be found to illustrate information. Decoding these parameters with an amount of color dots, filling the shape in a space enhancing way. This form is to note that dots are distributed in select positions as rows. Not only the shape for classification, but a large dictionary index. There are several icon shapes that when riddled between the concepts can transfer a large amount of onsite reading text. Imagine being connected not to a website to receive that audio file, and midi mixin. It is unknown how large it will be.

Perhaps the start of it could reference the conjecture that within a large structure of these tangles that the first identified within more or a closer camera view, could view as if in a maze of these shapes. There would be a maximum per area. Yet the

necessity of placing other information as a credible organized community printed without technology but received as a changing form. Digital graffiti with a nice array of shapes and dot information inside. The best aspects are perhaps osecuring visually a restaurant name visually. A huge five sided polygon with a total of eighteen dots stands in front. The camera pointed at the sign closer shows more information. As a physical classification, it can be read at a fast pace and nicely lit as a roadside art.

Designs of these types can be also regulated in a type of human art design. In providing the information on a camera oriented design formulate two dimensional line drawings of cartoon characters, in a pose, as a readable surface and show a link through OSR. As a dazed set of icons, with a dictionary of poses, the interior, or lines can be encoded to transmit data.

So as well as documentation, can also have links that are referenced by these database indicators.

Or a secure device that may communicate a key, and know or all hardware properties of the unit as a catalog. Using that information as a union to a server database, provides that derivatives of permutation functions are used to strengthen the hash code input between the server and client. Development of a unique data set where the smaller amount becomes a vast storage of non repeating operations, for an encryption cycle.

One aspect is that new art in the form of magnifying the position of the readable attributes are organic. For example, the ability to far range classify a tee shirt by shape and dots as a camera. The aspects are that within a group or at a range a classification would be found. Solving the minimal problem for standard uses is that the implications of the art designs are tailored around a standard close picture. The artistic shapes and dots are so varied that the amount of text is magnified. The best of tee shirts award is only printed on site. You type in your long array of slogans, and it exists there ever present. Yet the masked abilities for the shirt to be customized with professional art, and line drawings exemplified.

Pumping tee shirt entrance introductions, or

In the computer inside, there are multiple functions that are applied, can be applied.

Vendor ID:	AuthenticAMD
Model name:	AMD Ryzen 5 5500U with Radeon Graphics
CPU family:	23
Model:	104

[16 Commands to Check Hardware Information on Linux - BinaryTides](#)

[Linux command to generate new GUID? - Stack Overflow](#)

81b82623-ad77-41ad-a741-d0fd20888dc9

The important aspect of the devices communicating at a private and close trusted layer is accountability in podium embedded technology. The giving of trusted unique information for a tolerant room must be in order. The amount of people that say they are present in the close proximity network is a larger set of information that is related. Strategically this makes an identifiable network where permutations can be also based on peer client data. The same server is not regulated as requested.

## Analysis of Genetically Encrypted Streams

The ability to analyze the encrypted stream versus the raw stream can be leveraged simply by storing the coordinates of the original data within the structure. The ability to represent a single bit of information within a sequential stream is simply an integer index. Yet one of the surprising aspects is how to functionally prepare for the signal to be investigated for properties of meaningful decryption. The aspects of too much processing can be identified. Yet knowing how simplified each component part, if adapted to run on a single unit can be investigated by current deciphering.

A system design like this allows the functions to be adjusted for contextual use. To determine the weight complexity of each component can give it a weight within a numerical precision such as 0 - 1. Processes may measure the instantiation of multiple in series. Comparing the values of other existing encryption algorithms. The aspect of the analysis is of the greatest importance once some types of codes are available. From the analysis model, the production efforts can begin once proven through independent analysis that it is better. These algorithms have to be ensured for all types of data, with existing signatures.

The ability to formalize a visual scene of relationships of cell, and meaning of the data from a few key field of view frames, implies that the genetic encryption algorithm produce an animation of its operation attributes. As a stack and also recursive

detail. Several views can exist that are static image based. As well, interactive user interfaces that show the program dissection map. Characterizing the information in a read only form.

With this amount of detail, naming conventions and also likeable names must abide as a form of also learning attraction. Giving them pet names, or making this too obscure can be exhausting. As a system, the recursive and hidden world of the HID instantiation should be interesting.

The access as a template resource to produce a trial image animation of a series of recursive container modification can show in a series of image sequences. Sixty per second, as an animation time line production may create a story segmented by the file name series. A secondary system creates a video from the images as well as provide stitching between segments. Fade out, fade in, and also zooming for story telling to point out a cell. Adding sound elements with the video and balancing the music and trinket sounds syncronized with the binary nibble shifting going on in the image.

## Noise Injection

If information were to be weighted within a stream as construct or insert, and have associated metrics of size, format, and cryptographic strength, the noise production facility may be weighted to produce signals at massed sizes according to an algorithm. The amount of noise and stress of the algorithm can be feature-rich if necessary. The method decreases response time to provide adaptable security.

[White noise - Wikipedia](#)

Perlin noise is a popular gradient noise function used in computer graphics for generating natural-looking procedural content like terrains and textures. Below, I'll provide you with a simple example of how to implement Perlin noise in C++.

### Perlin Noise Implementation in C++

```
```cpp
#include <iostream>
#include <array>
#include <random>

template <class Float>
class BasicPerlinNoise {
public:
    using state_type = std::array<std::uint8_t, 256>;
    using value_type = Float;
    using default_random_engine = std::mt19937;
    using seed_type = typename default_random_engine::result_type;

    // Constructors
    constexpr BasicPerlinNoise();
    BasicPerlinNoise(seed_type seed);
    BasicPerlinNoise(URBG&& urbg);

    // Reseed
    void reseed(seed_type seed);
    void reseed(URBG&& urbg);

    // Noise (The result is in the range [-1, 1])
    value_type noise1D(value_type x) const noexcept;
    value_type noise2D(value_type x, value_type y) const noexcept;
    value_type noise3D(value_type x, value_type y, value_type z) const noexcept;

    // Noise (The result is remapped to the range [0, 1])
    value_type noise1D_01(value_type x) const noexcept;
    value_type noise2D_01(value_type x, value_type y) const noexcept;
    value_type noise3D_01(value_type x, value_type y, value_type z) const noexcept;

    // Octave noise (The result can be out of the range [-1, 1])
    value_type octave1D(value_type x, std::int32_t octaves, value_type persistence =
value_type(0.5)) const noexcept;
    value_type octave2D(value_type x, value_type y, std::int32_t octaves, value_type persistence =
value_type(0.5)) const noexcept;
    value_type octave3D(value_type x, value_type y, value_type z, std::int32_t octaves, value_type
```

```

persistence = value_type(0.5) const noexcept;

// Octave noise (The result is clamped to the range [-1, 1])
value_type octave1D_11(value_type x, std::int32_t octaves, value_type persistence =
value_type(0.5)) const noexcept;
value_type octave2D_11(value_type x, value_type y, std::int32_t octaves, value_type persistence =
value_type(0.5)) const noexcept;
value_type octave3D_11(value_type x, value_type y, value_type z, std::int32_t octaves,
value_type persistence = value_type(0.5)) const noexcept;
};

int main() {
    const siv::PerlinNoise::seed_type seed = 123456u;
    const siv::PerlinNoise perlin{seed};

    for (int y = 0; y < 5; ++y) {
        for (int x = 0; x < 5; ++x) {
            const double noise = perlin.octave2D_01((x * 0.01), (y * 0.01), 4);
            std::cout << noise << '\t';
        }
        std::cout << '\n';
    }

    return 0;
}
```

```

#### #### Explanation

- The `BasicPerlinNoise` class provides various noise functions, including 1D, 2D, and 3D noise.
- You can adjust the number of octaves and persistence to control the noise behavior.

Remember that Perlin noise is a powerful tool for creating natural-looking patterns, textures, and landscapes in your graphics applications!

Source: Conversation with Bing, 3/29/2024

- [C++: Perlin Noise Tutorial](#)
- [Programming Perlin-like Noise \(C++\)](#)
- [Perlin Noise Generation Algorithm In C++](#)
- [GitHub - Reputeless/PerlinNoise: Header-only Perlin noise library for modern C++ \(C++17/C++20\)](#)
- [Perlin noise in C++11 | Solarian Programmer](#)
- [GitHub - SRombauts/SimplexNoise: A Perlin's Simplex Noise C++ Implementation \(1D, 2D, 3D\)](#)
- [Best way to add seed to Perlin noise? - Stack Overflow](#)
- <http://webstaff.itn.liu.se/~stegu/simplexnoise/SimplexNoise.java>

bit noise

weighted by frequency

weighted by binary distribution

frequency distribution weights the position and signal

Perhaps the type of weight to introduce during the shifting would need a type of curve and tail function. The ability to introduce a tangible and small amount of noise, which grows the data should be processed for a cycle to make use of it. Gradually nearing a size allotment for complexity of design limits. A curve function and reduced cycle can function as a lock.

A textual package identifier supplies the label contents on the network with an id. Should information be proud as a signal?

In a network TLS session, more data must be established in the initial connection configuration within the client and server as a type of security uplift. Many sites such as Google utilize the storage capabilities of the browser to establish this. The expectation that a terminal would activate between long term and short term is also related to software usage. The attributes of the user storage and modification.

# Parallel Processing

The nature of modern hardware provides several facets operating independently. The many algorithms as systems operating, if possible, should be constructed to utilize multiple CPUs at the same time. Often this implies that strategic sectioning of the data must be accomplished. That is, the package format should be formalized in that a partition between sections can be found. Often processes like these need an order to split the data or arrange elements into sections. It is also possible that while a function is operating, it may be queried for some data or data may be swapped out.

The aspects of remembering the sequence should be planned. One way to do this is by having several points stored where a counter would eventually change to a corresponding value. For example, the value of a counter must reach a division of one hundred before serving or inserting data. This allows the threads to be synchronized during decryption. So threads that are needing to swap data are waiting for a boundary numeric to be hit. The ability to debug these types of programs is more difficult as each encoding or decoding thread must be stopped to debug the synchronization.

Parallel processing of extensive data is necessary. An average reader can likely deduce the types of tools available from the list. Some interesting aspects are that existing references are the entire lot of functionality, at times it is more simple to use a specific type. Yet as you are aware, programming is a human concept and here is the current likeness of the modern c++ language. Attributed to many years of research and also debugging, system design was centered around making multiple processes in algorithms work. The new name became deadlock. A new type of bug. Two or more threads of logic waiting on the same signal. The signal never occurs. Strategic planning of implementation using data structures defines better behavior.

# Concurrency support library (since C++11)

C++

## Concurrency support library

C++ includes built-in support for threads, atomic operations, mutual exclusion, condition variables, and futures.

### Threads

Threads enable programs to execute across several processor cores.

Defined in header `<thread>`

|                           |                                                                                    |
|---------------------------|------------------------------------------------------------------------------------|
| <b>thread</b><br>(C++11)  | manages a separate thread<br>(class)                                               |
| <b>jthread</b><br>(C++20) | <code>std::thread</code> with support for auto-joining and cancellation<br>(class) |

### Functions managing the current thread

Defined in namespace `this_thread`

|                               |                                                                                       |
|-------------------------------|---------------------------------------------------------------------------------------|
| <b>yield</b><br>(C++11)       | suggests that the implementation reschedule execution of threads<br>(function)        |
| <b>get_id</b><br>(C++11)      | returns the thread id of the current thread<br>(function)                             |
| <b>sleep_for</b><br>(C++11)   | stops the execution of the current thread for a specified time duration<br>(function) |
| <b>sleep_until</b><br>(C++11) | stops the execution of the current thread until a specified time point<br>(function)  |

## Thread cancellation

The `stop_XXX` types are designed to enable thread cancellation for `std::jthread`, although they can also be used independently of `std::jthread` - for example to interrupt `std::condition_variable_any` waiting functions, or for a custom thread management implementation. In fact they do not even need to be used to "stop" anything, but can instead be used for a thread-safe one-time function(s) invocation trigger, for example.

Defined in header `<stop_token>`

|                                       |                                                                                                        |
|---------------------------------------|--------------------------------------------------------------------------------------------------------|
| <code>stop_token</code><br>(C++20)    | an interface for querying if a <code>std::jthread</code> cancellation request has been made<br>(class) |
| <code>stop_source</code><br>(C++20)   | class representing a request to stop one or more <code>std::jthreads</code><br>(class)                 |
| <code>stop_callback</code><br>(C++20) | an interface for registering callbacks on <code>std::jthread</code> cancellation<br>(class template)   |

## Cache size access

Defined in header `<new>`

|                                                                                                                        |                                                                                       |
|------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| <code>hardware_destructive_interference_size</code><br><code>hardware_constructive_interference_size</code><br>(C++17) | min offset to avoid false sharing<br>max offset to promote true sharing<br>(constant) |
|------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|

## Atomic operations

These components are provided for fine-grained atomic operations allowing for lockless concurrent programming. Each atomic operation is indivisible with regards to any other atomic operation that involves the same object. Atomic objects are [free of data races](#).

Neither the `_Atomic` macro, nor any of the non-macro global namespace declarations are provided by any C++ standard library header other than `<stdatomic.h>`. (since C++23)

Defined in header `<atomic>`

### Atomic types

|                                    |                                                                                                                                   |
|------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| <code>atomic</code><br>(C++11)     | atomic class template and specializations for bool, integral, floating-point, (since C++20) and pointer types<br>(class template) |
| <code>atomic_ref</code><br>(C++20) | provides atomic operations on non-atomic objects<br>(class template)                                                              |

### Operations on atomic types

|                                                                                                                                                                                                                                                    |                                                                                                                                                                |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>atomic_is_lock_free</code><br>(C++11)                                                                                                                                                                                                        | checks if the atomic type's operations are lock-free<br>(function template)                                                                                    |
| <code>atomic_store</code><br><code>atomic_store_explicit</code><br>(C++11)<br>(C++11)                                                                                                                                                              | atomically replaces the value of the atomic object with a non-atomic argument<br>(function template)                                                           |
| <code>atomic_load</code><br><code>atomic_load_explicit</code><br>(C++11)<br>(C++11)                                                                                                                                                                | atomically obtains the value stored in an atomic object<br>(function template)                                                                                 |
| <code>atomic_exchange</code><br><code>atomic_exchange_explicit</code><br>(C++11)<br>(C++11)                                                                                                                                                        | atomically replaces the value of the atomic object with non-atomic argument and returns the old value of the atomic<br>(function template)                     |
| <code>atomic_compare_exchange_weak</code><br><code>atomic_compare_exchange_weak_explicit</code><br><code>atomic_compare_exchange_strong</code><br><code>atomic_compare_exchange_strong_explicit</code><br>(C++11)<br>(C++11)<br>(C++11)<br>(C++11) | atomically compares the value of the atomic object with non-atomic argument and performs atomic exchange if equal or atomic load if not<br>(function template) |
| <code>atomic_fetch_add</code><br><code>atomic_fetch_add_explicit</code><br>(C++11)<br>(C++11)                                                                                                                                                      | adds a non-atomic value to an atomic object and obtains the previous value of the atomic<br>(function template)                                                |
| <code>atomic_fetch_sub</code><br><code>atomic_fetch_sub_explicit</code><br>(C++11)<br>(C++11)                                                                                                                                                      | subtracts a non-atomic value from an atomic object and obtains the previous value of the atomic<br>(function template)                                         |

|                                                                                   |                                                                                                                                                          |
|-----------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>atomic_fetch_and</b><br><b>atomic_fetch_and_explicit</b><br>(C++11)<br>(C++11) | replaces the atomic object with the result of bitwise AND with a non-atomic argument and obtains the previous value of the atomic<br>(function template) |
| <b>atomic_fetch_or</b><br><b>atomic_fetch_or_explicit</b><br>(C++11)<br>(C++11)   | replaces the atomic object with the result of bitwise OR with a non-atomic argument and obtains the previous value of the atomic<br>(function template)  |
| <b>atomic_fetch_xor</b><br><b>atomic_fetch_xor_explicit</b><br>(C++11)<br>(C++11) | replaces the atomic object with the result of bitwise XOR with a non-atomic argument and obtains the previous value of the atomic<br>(function template) |
| <b>atomic_wait</b><br><b>atomic_wait_explicit</b><br>(C++20)<br>(C++20)           | blocks the thread until notified and the atomic value changes<br>(function template)                                                                     |
| <b>atomic_notify_one</b><br>(C++20)                                               | notifies a thread blocked in atomic_wait<br>(function template)                                                                                          |
| <b>atomic_notify_all</b><br>(C++20)                                               | notifies all threads blocked in atomic_wait<br>(function template)                                                                                       |

## Flag type and operations

|                                                                                                   |                                                                                            |
|---------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|
| <b>atomic_flag</b><br>(C++11)                                                                     | the lock-free boolean atomic type<br>(class)                                               |
| <b>atomic_flag_test_and_set</b><br><b>atomic_flag_test_and_set_explicit</b><br>(C++11)<br>(C++11) | atomically sets the flag to <code>true</code> and returns its previous value<br>(function) |
| <b>atomic_flag_clear</b><br><b>atomic_flag_clear_explicit</b><br>(C++11)<br>(C++11)               | atomically sets the value of the flag to <code>false</code><br>(function)                  |
| <b>atomic_flag_test</b><br><b>atomic_flag_test_explicit</b><br>(C++20)<br>(C++20)                 | atomically returns the value of the flag<br>(function)                                     |
| <b>atomic_flag_wait</b><br><b>atomic_flag_wait_explicit</b><br>(C++20)<br>(C++20)                 | blocks the thread until notified and the flag changes<br>(function)                        |
| <b>atomic_flag_notify_one</b><br>(C++20)                                                          | notifies a thread blocked in atomic_flag_wait<br>(function)                                |
| <b>atomic_flag_notify_all</b><br>(C++20)                                                          | notifies all threads blocked in atomic_flag_wait<br>(function)                             |

## Initialization

|                                                        |                                                                                              |
|--------------------------------------------------------|----------------------------------------------------------------------------------------------|
| <b>atomic_init</b><br>(C++11)(deprecated in C++20)     | non-atomic initialization of a default-constructed atomic object<br>(function template)      |
| <b>ATOMIC_VAR_INIT</b><br>(C++11)(deprecated in C++20) | constant initialization of an atomic variable of static storage duration<br>(function macro) |
| <b>ATOMIC_FLAG_INIT</b><br>(C++11)                     | initializes an <code>std::atomic_flag</code> to <code>false</code><br>(macro constant)       |

## Memory synchronization ordering

|                                       |                                                                                                                     |
|---------------------------------------|---------------------------------------------------------------------------------------------------------------------|
| <b>memory_order</b><br>(C++11)        | defines memory ordering constraints for the given atomic operation<br>(enum)                                        |
| <b>kill_dependency</b><br>(C++11)     | removes the specified object from the <code>std::memory_order_consume</code> dependency tree<br>(function template) |
| <b>atomic_thread_fence</b><br>(C++11) | generic memory order-dependent fence synchronization primitive<br>(function)                                        |
| <b>atomic_signal_fence</b><br>(C++11) | fence between a thread and a signal handler executed in the same thread<br>(function)                               |

## C compatibility macros

`_Atomic`  
(C++23)

compatibility macro such that `_Atomic(T)` is identical to  
`std::atomic<T>`  
(function macro)

## Mutual exclusion

Mutual exclusion algorithms prevent multiple threads from simultaneously accessing shared resources. This prevents data races and provides support for synchronization between threads.

Defined in header `<mutex>`

`mutex`  
(C++11)

provides basic mutual exclusion facility  
(class)

`timed_mutex`  
(C++11)

provides mutual exclusion facility which implements locking with a timeout  
(class)

`recursive_mutex`  
(C++11)

provides mutual exclusion facility which can be locked recursively by the same thread  
(class)

`recursive_timed_mutex`  
(C++11)

provides mutual exclusion facility which can be locked recursively by the same thread and implements locking with a timeout  
(class)

Defined in header `<shared_mutex>`

`shared_mutex`  
(C++17)

provides shared mutual exclusion facility  
(class)

`shared_timed_mutex`  
(C++14)

provides shared mutual exclusion facility and implements locking with a timeout  
(class)

## Generic mutex management

Defined in header `<mutex>`

`lock_guard`  
(C++11)

implements a strictly scope-based mutex ownership wrapper  
(class template)

`scoped_lock`  
(C++17)

deadlock-avoiding RAII wrapper for multiple mutexes  
(class template)

`unique_lock`  
(C++11)

implements movable mutex ownership wrapper  
(class template)

`shared_lock`  
(C++14)

implements movable shared mutex ownership wrapper  
(class template)

`defer_lock_t`  
`try_to_lock_t`  
`adopt_lock_t`  
(C++11)  
(C++11)  
(C++11)

tag type used to specify locking strategy  
(class)

`defer_lock`  
`try_to_lock`  
`adopt_lock`  
(C++11)  
(C++11)  
(C++11)

tag constants used to specify locking strategy  
(constant)

## Generic locking algorithms

`try_lock`  
(C++11)

attempts to obtain ownership of mutexes via repeated calls to `try_lock`  
(function template)

`lock`  
(C++11)

locks specified mutexes, blocks if any are unavailable  
(function template)

## Call once

`once_flag`  
(C++11)

helper object to ensure that `call_once` invokes the function only once  
(class)

`call_once`  
(C++11)

invokes a function only once even if called from multiple threads  
(function template)

## Condition variables

A condition variable is a synchronization primitive that allows multiple threads to communicate with each other. It allows some number of threads to wait (possibly with a timeout) for notification from another thread that they may proceed. A condition variable is always associated with a mutex.

Defined in header `<condition_variable>`

|                                             |                                                                                                              |
|---------------------------------------------|--------------------------------------------------------------------------------------------------------------|
| <b>condition_variable</b><br>(C++11)        | provides a condition variable associated with a <code>std::unique_lock</code> (class)                        |
| <b>condition_variable_any</b><br>(C++11)    | provides a condition variable associated with any lock type (class)                                          |
| <b>notify_all_at_thread_exit</b><br>(C++11) | schedules a call to <code>notify_all</code> to be invoked when this thread is completely finished (function) |
| <b>cv_status</b><br>(C++11)                 | lists the possible results of timed waits on condition variables (enum)                                      |

(since C++20)

## Semaphores

A semaphore is a lightweight synchronization primitive used to constrain concurrent access to a shared resource. When either would suffice, a semaphore can be more efficient than a condition variable.

Defined in header `<semaphore>`

|                                      |                                                                      |
|--------------------------------------|----------------------------------------------------------------------|
| <b>counting_semaphore</b><br>(C++20) | semaphore that models a non-negative resource count (class template) |
| <b>binary_semaphore</b><br>(C++20)   | semaphore that has only two states (typedef)                         |

## Latches and Barriers

Latches and barriers are thread coordination mechanisms that allow any number of threads to block until an expected number of threads arrive. A latch cannot be reused, while a barrier can be used repeatedly.

Defined in header `<latch>`

|                         |                                   |
|-------------------------|-----------------------------------|
| <b>latch</b><br>(C++20) | single-use thread barrier (class) |
|-------------------------|-----------------------------------|

Defined in header `<barrier>`

|                           |                                          |
|---------------------------|------------------------------------------|
| <b>barrier</b><br>(C++20) | reusable thread barrier (class template) |
|---------------------------|------------------------------------------|

## Futures

The standard library provides facilities to obtain values that are returned and to catch exceptions that are thrown by asynchronous tasks (i.e. functions launched in separate threads). These values are communicated in a *shared state*, in which the asynchronous task may write its return value or store an exception, and which may be examined, waited for, and otherwise manipulated by other threads that hold instances of `std::future` or `std::shared_future` that reference that shared state.

Defined in header `<future>`

|                                 |                                                                                                                                                   |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>promise</b><br>(C++11)       | stores a value for asynchronous retrieval (class template)                                                                                        |
| <b>packaged_task</b><br>(C++11) | packages a function to store its return value for asynchronous retrieval (class template)                                                         |
| <b>future</b><br>(C++11)        | waits for a value that is set asynchronously (class template)                                                                                     |
| <b>shared_future</b><br>(C++11) | waits for a value (possibly referenced by other futures) that is set asynchronously (class template)                                              |
| <b>async</b><br>(C++11)         | runs a function asynchronously (potentially in a new thread) and returns a <code>std::future</code> that will hold the result (function template) |
| <b>launch</b><br>(C++11)        | specifies the launch policy for <code>std::async</code> (enum)                                                                                    |
| <b>future_status</b><br>(C++11) | specifies the results of timed waits performed on <code>std::future</code> and <code>std::shared_future</code> (enum)                             |

## Future errors

|                                   |                                                         |
|-----------------------------------|---------------------------------------------------------|
| <b>future_error</b><br>(C++11)    | reports an error related to futures or promises (class) |
| <b>future_category</b><br>(C++11) | identifies the future error category (function)         |
| <b>future_errc</b><br>(C++11)     | identifies the future error codes (enum)                |

## Safe Reclamation

Safe-reclamation techniques are most frequently used to straightforwardly resolve access-deletion races.

### Read-Copy-Update Mechanism

Defined in header `<rcu>`

|                                            |                                                                                                                                                            |
|--------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>rcu_obj_base</code><br>(C++26)       | allows an object to be protected by RCU<br>(class template)                                                                                                |
| <code>rcu_domain</code><br>(C++26)         | provides regions of RCU protection<br>(class)                                                                                                              |
| <code>rcu_default_domain</code><br>(C++26) | returns a reference to a static-duration object of type <code>std::rcu_domain</code><br>(function)                                                         |
| <code>rcu_synchronize</code><br>(C++26)    | blocks until a protection region unlocks on a RCU domain<br>(function)                                                                                     |
| <code>rcu_barrier</code><br>(C++26)        | may evaluate scheduled operations on a RCU domain and blocks until all preceding evaluations are complete<br>(function)                                    |
| <code>rcu_retire</code><br>(C++26)         | schedules the evaluation of a specified function on a RCU domain, potentially allocating memory, and invoking scheduled evaluations<br>(function template) |

### Hazard Pointers

Defined in header `<hazard_pointer>`

|                                                 |                                                                                                            |
|-------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| <code>hazard_pointer_obj_base</code><br>(C++26) | allows an object to be hazard-protectable<br>(class template)                                              |
| <code>hazard_pointer</code><br>(C++26)          | single-writer multi-reader pointer that can be owned by at most one thread at any point of time<br>(class) |
| <code>make_hazard_pointer</code><br>(C++26)     | constructs a hazard pointer<br>(function)                                                                  |

## See also

[C documentation for Concurrency support library](#)

A particularly interesting item is the `mutex`.

# std::mutex

C++

[Concurrency support library](#)

`std::mutex`

Defined in header `<mutex>`

`class mutex;` (since C++11)

The `mutex` class is a synchronization primitive that can be used to protect shared data from being simultaneously accessed by multiple threads.

`mutex` offers exclusive, non-recursive ownership semantics:

- A calling thread *owns* a `mutex` from the time that it successfully calls either `lock` or `try_lock` until it calls `unlock`.
- When a thread owns a `mutex`, all other threads will block (for calls to `lock`) or receive a `false` return value (for `try_lock`) if they attempt to claim ownership of the `mutex`.
- A calling thread must not own the `mutex` prior to calling `lock` or `try_lock`.

The behavior of a program is undefined if a `mutex` is destroyed while still owned by any threads, or a thread terminates while owning a `mutex`. The `mutex` class satisfies all requirements of [Mutex](#) and [StandardLayoutType](#).

`std::mutex` is neither copyable nor movable.

### Nested types

| Name                                        | Definition             |
|---------------------------------------------|------------------------|
| <code>native_handle_type</code> (optional*) | implementation-defined |

## Member functions

|                               |                                                  |
|-------------------------------|--------------------------------------------------|
| (constructor)                 | constructs the mutex<br>(public member function) |
| (destructor)                  | destroys the mutex<br>(public member function)   |
| <b>operator=</b><br>[deleted] | not copy-assignable<br>(public member function)  |

## Locking

|                 |                                                                                            |
|-----------------|--------------------------------------------------------------------------------------------|
| <b>lock</b>     | locks the mutex, blocks if the mutex is not available<br>(public member function)          |
| <b>try_lock</b> | tries to lock the mutex, returns if the mutex is not available<br>(public member function) |
| <b>unlock</b>   | unlocks the mutex<br>(public member function)                                              |

## Native handle

|                      |                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------|
| <b>native_handle</b> | returns the underlying implementation-defined native handle object<br>(public member function) |
|----------------------|------------------------------------------------------------------------------------------------|

## Notes

`std::mutex` is usually not accessed directly: `std::unique_lock`, `std::lock_guard`, or `std::scoped_lock`(since C++17) manage locking in a more exception-safe manner.

## Example

This example shows how a `mutex` can be used to protect an `std::map` shared between two threads.

Run this code

```
#include <chrono>
#include <iostream>
#include <map>
#include <mutex>
#include <string>
#include <thread>

std::map<std::string, std::string> g_pages;
std::mutex g_pages_mutex;

void save_page(const std::string& url)
{
 // simulate a long page fetch
 std::this_thread::sleep_for(std::chrono::seconds(2));
 std::string result = "fake content";

 std::lock_guard<std::mutex> guard(g_pages_mutex);
 g_pages[url] = result;
}

int main()
{
 std::thread t1(save_page, "http://foo");
 std::thread t2(save_page, "http://bar");
 t1.join();
 t2.join();

 // safe to access g_pages without lock now, as the threads are joined
 for (const auto& [url, page] : g_pages)
 std::cout << url <> " => " << page << '\n';
}
```

Output:

```
http://bar => fake content
http://foo => fake content
```

## See also

|                                   |                                                                                                  |
|-----------------------------------|--------------------------------------------------------------------------------------------------|
| <b>recursive_mutex</b><br>(C++11) | provides mutual exclusion facility which can be locked recursively by the same thread<br>(class) |
| <b>lock_guard</b><br>(C++11)      | implements a strictly scope-based mutex ownership wrapper<br>(class template)                    |

|                                      |                                                                                          |
|--------------------------------------|------------------------------------------------------------------------------------------|
| <b>unique_lock</b><br>(C++11)        | implements movable mutex ownership wrapper<br>(class template)                           |
| <b>scoped_lock</b><br>(C++17)        | deadlock-avoiding RAII wrapper for multiple mutexes<br>(class template)                  |
| <b>condition_variable</b><br>(C++11) | provides a condition variable associated with a <code>std::unique_lock</code><br>(class) |

Parallel processing in C++ can significantly improve performance by utilizing multiple cores or threads to execute tasks concurrently. Here are a few approaches for achieving parallelism in C++:

#### 1. \*\*C++17 Parallel Algorithms\*\*:

- C++17 introduced parallel algorithms to the standard library, allowing programs to take advantage of parallel execution for improved performance.

- These algorithms work with any tool consuming the standard library, not just MSVC's compiler.
- To use them:
  1. Identify an algorithm call in your program that can benefit from parallelism (e.g., sorting).
  2. Ensure that the code you supply to the algorithm is safe for parallelization.
  3. Choose a parallel execution policy (e.g., `std::execution::par`).
  4. Include `` to make parallel execution policies available.
  5. Add the execution policy as the first parameter to the algorithm call.
  6. Benchmark the result to verify performance improvements<sup>1</sup>.

#### 2. \*\*PASL Library\*\*:

- For more advanced parallel programming, consider using the \*\*PASL library\*\*.
- PASL provides utilities for understanding the behavior of parallel programs and efficiently runs parallel programs on modern multicores.
- It uses advanced scheduling techniques to achieve efficient parallel execution<sup>2</sup>.

#### 3. \*\*OpenMP\*\*:

- OpenMP is a widely used API for parallel programming in C++.
- Developers can write parallel code easily by adding simple compiler directives to existing code.
- It breaks down large tasks into smaller subtasks that execute simultaneously, utilizing available computing resources efficiently<sup>3</sup>.

#### 4. \*\*C++11 and Beyond\*\*:

- Modern C++ includes threading libraries.
- C++11 introduced a standard threading library, and C++17 added parallel implementations of many standard algorithms.
- Future versions of C++ are expected to provide additional support for parallelism<sup>4</sup>.

Remember that parallelizing code isn't always faster, especially for certain scenarios. Profiling and benchmarking are essential to determine whether parallelization improves performance. 

Source: Conversation with Bing, 3/28/2024

- (1) [Using C++17 Parallel Algorithms for Better Performance - C++ Team Blog](#).
- (2) [An Introduction to Parallel Computing in C++](#).
- (3) [Introduction to Parallel Programming with OpenMP in C++ - GeeksforGeeks](#).
- (4) [What Is Parallel Programming and Multithreading?](#).

Let's explore an example of using a \*\*mutex\*\* in C++ to ensure thread safety.

A \*\*mutex\*\* (short for "mutual exclusion") is a synchronization primitive that helps protect shared data from being simultaneously accessed by multiple threads. It allows you to create a critical section where only one thread can execute the protected code at a time.

Here's a simple example demonstrating how to use a \*\*mutex\*\* to protect an `std::map` shared between two threads:

```
```cpp
#include <iostream>
#include <map>
#include <mutex>
#include <string>
#include <thread>

std::map<std::string, std::string> myMap;
std::mutex mtx; // Declare a mutex
```

```

```

void addToMap(const std::string& key, const std::string& value) {
 // Lock the mutex before accessing the shared resource
 std::lock_guard<std::mutex> lock(mtx);

 // Simulate some work (e.g., fetching data)
 std::this_thread::sleep_for(std::chrono::milliseconds(100));

 // Add or update the key-value pair in the map
 myMap[key] = value;

 // Mutex is automatically released when 'lock' goes out of scope
}

int main() {
 // Create two threads that concurrently add to the map
 std::thread t1(addToMap, "apple", "fruit");
 std::thread t2(addToMap, "banana", "yellow fruit");

 // Wait for both threads to finish
 t1.join();
 t2.join();

 // Print the contents of the map
 for (const auto& pair : myMap) {
 std::cout << pair.first << ":" << pair.second << std::endl;
 }

 return 0;
}
...

```

In this example:

- We declare an `std::mutex` named `mtx`.
- The `addToMap` function locks the mutex before accessing the shared `myMap`.
- The `std::lock\_guard` ensures that the mutex is automatically released when the function scope ends.

Remember that using a mutex introduces some overhead due to locking and unlocking. However, it's essential for preventing data races and ensuring thread safety when multiple threads access shared resources concurrently. 

Source: Conversation with Bing, 3/28/2024

- (1) std::mutex - cppreference.com. [std::mutex - cppreference.com](https://en.cppreference.com/w/cpp/thread/mutex).
- (2) Mutex in C++ - GeeksforGeeks. <https://www.geeksforgeeks.org/std-mutex-in-cpp/>.
- (3) C++ Tutorial: C++11/C++14 7. Threads with Shared Memory and Mutex - 2017. [https://www.bogotobogo.com/cplusplus/C11/7\\_C11\\_Thread\\_Sharing\\_Memory.php](https://www.bogotobogo.com/cplusplus/C11/7_C11_Thread_Sharing_Memory.php).
- (4) C++ Tutorial: Multi-Threaded Programming - C++11 B- 2020. [https://www.bogotobogo.com/cplusplus/multithreaded4\\_cplusplus11B.php](https://www.bogotobogo.com/cplusplus/multithreaded4_cplusplus11B.php).

An important factor may be data decryption dependence..

## Dynamic Scale

The scale, strength and time of the encryption format must fit within a configurable profile. Often algorithms must be adjusted over the long duration to increase intelligence. The ability for the system to apply several variables to be used as an operating instance should also directly affect the type and amount of encryption. The ability to control the amount of time is important. As computers increase in speed, aspects of encryption and amount can also increase to name that faster computers are more secure.

The aspects of adding noise can also be configured. Noise is mixed in existing data in oddball ways. It must also be removed at a point in the decryption process. The aspect of combining encoder data and noise data provides a type of logical noise. Parameters that are sent, should have the same qualities as the output format. In most of the data stamping facilities, binary and encoder information is stored. The ability for a noise mixer to also be fluent in the type of data that it is surrounding can affect quality. The statistical study of a formed existing package provides an input parameter. The noise algorithm can also have explicit parameters that are stored to provide configuration for the type of noise that was introduced.

Noise introduced over time may be a better transform to weight the data than adding an entire amount at once. These functions are dependent upon how the encryption cycles after adding the noise information. Noise information can also be selected for the best position within the chain of operations. The amount and frequency of the noise also must be affected by a selection algorithm by input configuration. The noise property

The tolerance for an allotted amount is also based upon how small the data is. The value should be clamped to a minimum size. That is by input, the configuration of the encryption algorithm may be selectively different even though the password phrase is the same. When the entire parameter set of the PRNG and encryption affects the outcome of the cipher, the values for these settings should also be operating on a tree of random numbers. These values have to be set exactly the same on the client and \server.

The dynamic capabilities of such an algorithm can also distribute the storage of data on an array of potential target storage locations. This capability is instituted by storing the data in separate files. The aspect that network communications are required for remote storage is left for later implementation. The main aspect of which file to store data in is comparable to which remote location. The algorithm must supply the framework for this to occur. This will allow a caller to finalize implementation details better.

## Phrase Secret Sustainment and Life Cycle

The ability to have an automated system where the phrase information, or cipher instructions are gathered from the local and server files by an application means that the phrase exists on both sides. The method of manual key installing to the full potential exists. This is the main ingredient where it can perform better, yet the allotment of storage implies that it can also be a security concept. The file may be read by programs running. There are mechanisms within operating systems that provide management of file system security. Phrase information on the machines can also be impacted by incorporating a series of genetic encryption that modify the data. The process is both synchronized on the server and client.

Another potential is by creating a phrase from unrelated data that is stored or can be recognized by the title name. Such as public domain audio, video and text information. As residing in locations other than the protected server or client, the information must be gathered and investigated separately. Some selection using the method can be manual or developed from programming logic. Therefore the selection of the phrase data, can also be selected by the client or server. The phrase information may be derived from multiple formats or sustaining as a cached result known. By having several sources, public open or closed, the cached version can be relocated through stenographic means to balance the two computers.

The ability to process media files such as song information for traits recognized can be formalized. When the source can vary by quality, such as two copies of a song, the algorithm can still function to pull characteristics of the media file. The aspects of BPM, or an equalizer fourier transform can be used to summarize musical audio data. The prospect of these facilities imply also that the password phrase is used for a person to person cipher. The name of the song or data file is never told of through the algorithm, but other systems gather the information.

So that password, generation of a password key to transfer physically, must entail using the program in multiple ways. These types of passwords are totally private according to the manual distribution of the password key file. Software that transmits and reads key information from a private source must be strengthened and also tracked for potential leaks. Therefore, automated solutions must identify the storage medium used to transfer as a unique device. The file must only exist one time within the network. As such tracking the result from the server.

By supplying a route and information for where the information came from, the phrase file can be generated and also divided. Balancing the information after the phrase strength has been updated from a partial source. The file system on these portable drives will be considered obfuscated as well by having a private design. This does not prevent copying the key with specialized software if the thumb drive were to be stolen.

Another very important concept is using one computer to generate the file and store the information on two thumb drives. When the two users leave their physical meeting, the phrase data can be used by the two parties wishing to have a secure connection between the two. This can be accomplished by many business agreements. Imagine that a new business associate has the ability to work remotely. Once the practice has been established, the new hire is expected to stay for a duration. These types of phrase sustainment can be totally private between network computers.

Once the key data has been imported by the client program on either side, the USB thumb drive has to be erased so that no physical proof of the contents can be discovered. The ability to lock the transfer and contents to a drive should also be tied to a verbal password. This allows the physical usb key, if found, to also be encrypted with the verbal password.

The expectation of these keys, or partial key importing from a USB thumb drive should expire the key data after a certain amount of time. When the data is written, within the contents and directory, the date and time are reflected in a way such that discovery, decryption, is based upon the time window frame parameters. The time values, date and range affect the encryption algorithm such that according to the computer input date and time, it can only be imported at a specific window period. During this period, the phrase file, along with the password, are tracked by the server so that only one computer gets reception of the data. The key data is also expired on the server. The USB thumb data is used to activate the stenographic transfer of the rest of the key data to the clients.

Phrase code pages can also be printed out in various forms for the two people. The ability to import such a phrase must come from the import process to use shape recognition. After importing the data, the server expires the printed code page and recognizes the client as a trusted computer. The partial key data can be updated as normal balance from the server. When import is accomplished using a camera or scanner, the document should be shredded. After one import use, the key data inscribed is expired.

The system guards the integrity of the phrase as reporting the import of an existing expired key. When this occurs, the client and server communicate to strengthen the leaked data. The system has a plan where the password phrase must be accounted for to be unique, this method offers a way. The administrator plans the medium and physically distributes the information. There may be a form of where dissemination needs a process to produce a large number of the partial phrase agreements. Possibly a paper printout, front and back with a client's name on it. The group leader hands the partial phrase code page to a client user.

In this, the partial phrase code page also may have the responsibility to store client connection information. And rely on a verbal password along with the paper. A verbal password in this circumstance may not be useful in that producing it for each person is too demanding. The ability to direct the relationship of the client to specific databases, or servers is applied in the server processing logic.

Use cases and strength of discoverability should be recorded with each type of phrase transfer. By designing the object API to be this versatile, multiple programs varying in usage can be applied for users. For example, the printed code pages handed out personally by individuals for optical scanning.

After a given duration, the cache must be mutated. The mutation is a cooperative effort between the server and client that can be accomplished through the network. Yet in some combinations, the mutation, additional algorithm instructions, or other client settings can be modified at this time. The ability for this to be a code book instruction is also a functional design. Merely the phrase information is also stored with attributes that describe its creation, source location, owning computer, user, and also creation method.

The aspect of having a physically secure code book source that unlocks files for a client may also be necessary. The ability to place the backdoor within a printed non telecommunications realm adapts to physical security. Code books and pages should be designed to only run on a specific computer.

The potential for using a cipher code page that can exist for only a period of time is necessary. After the allotted time, new code book pages must be input. It could be daunting and a complaint if not used strategically for a reason. That is, users will have to keep scanning more code books to use the resources. This may be useful for computers that are used by multiple people. In this way, the cipher phrase becomes stale and not used. The files do not exist within the storage of the device and are used as a protected temporary resource.

Code pages may also be anonymous in the fact that several may exist. If the system were instituted on small portable devices such as consumer phones, during setup of the phone new avenues of deployment may take place. The setup process is encrypted privately with a distributor mechanism. The distributor server software relays the partial key that is available to the client to the service provider. This establishes a very secure connection with private genetic security. The distributor server software has the capability to not communicate with the client about the phrase, yet the device is instantiated to know of its version. The server processing and its attachment to service providers can be thought of as physically secure keys. That is prior to being a service provider, the distributor server software has to be installed with its own private matching distributor.service provider key. These are long term keys.

Data may also exist through producing a three dimensional printed design. While the shape needs refinement in definition, through plotting surfaces, the shape can reflect information. An information design such as this may be limited in size. The ability to produce two models that can contain partial information, yet with all photographed in a video, to contain the sides, is possible. Is it possible to recycle the material once it is solid? Another possibility is providing that it is water soluble. Placing it into liquids erodes the information. Perhaps a specific type of device simplified for making such objects as physical tags. The thinness and mounting principles adjusted for the application. If it is designed to fit a frame mounted upon a drawer for example, it would have a dimension. Perhaps some models can be fitted with a suction cup to stick on glass. Models can be designed, as memory notes intended to be decrypted by one device only. Stamped steel and aluminum inserts may also be liked. Imagine the bronze statue in Saint Marks place, transmitting a ten second song while it stands.

Software source code, components may also be printed in the form. This may be an option for enabling new encryption algorithms on systems. Many types of information can be planned using the format. More research in system placement and usability will be required.

## Compression

Several compression algorithms are useful. The major input of data, as a known format from the encoder applies that information can be group and also encoded much more compactly. The essence of grammar rule compression supplies that rules for forming a sentence are given. The nature of the current sentence structure represented abstractly provides that forming word combination as per article may be found. The progressive understanding is that it is a type of template where the words that are meaningful, implicit are transmitted while the inception of the grammar rule, or group of rules connected, form the output. In these cases, the grammar representation is a mild type of data structure. Although a research concept as most implementations of compressors are more generalized, the implicitness of language output is gathered by scanning input.

Huffman, Adaptive Huffman are also very robust implementations that many algorithms employ. Most video and audio data are compressed using this technique after DCT or Acoustical analysis. Adaptive Huffman is the most keen implementation as both a dictionary and no tree are required. The implementation of a clerical document, such as vitter.c provides a debugged version of an algorithm.

Numerical compression, image compression, and arithmetic coding provide an interesting field of captive audience. Most assents into the compression implementation give that format is important. Floating or integer, field tested methods exist. Libraries that have buffering IO build in such as zlib, or bzip2, 7zip have implementation details already. The work of using a byte is typical. The implementations here are coded for the nibble for text information.

The burris Wheeler transform is known as a sorting algorithm that allows other algorithms to perform better. Most notable algorithms work this way. The finding of a way to reduce the data variance. Delta encoding, or also GCD can also be used. Other forms of research oriented data compression algorithms are primarily based upon the aspect of code working fast. I have already made some interesting claims about compression techniques that will be explored within the books source code. The promise of a nibble based textual representation implies that the dictionary and change commands will be most characteristic in the frequency. The ability to find a given distribution.

Other types of text compression that work on probability such as arithmetic compression are sophisticated in saying that bits can be represented in fraction spaces. Yet similar aspects of character occurrence and dictionary substitution for remarked characters may be rewarding also in saving a nibble. Every nibble counts.

[Time-series compression algorithms, explained](#)

[huffman encoding implemented by c++ · GitHub](#)

[Grammar Compression](#)

[Huffman coding](#)

[The FastPFOR C++ library: Fast integer compression](#)

[Multimedia/Adaptive\\_Huffman\\_Coding/compression-code/vitter.c at master](#)

[Lempel–Ziv–Markov chain algorithm - Wikipedia](#)

[7-Zip](#)

[Burrows–Wheeler transform - Wikipedia](#)

[Data Compression with the Burrows-Wheeler Transform | Mark Nelson](#)

The representation of bitpacking numerical information is the modern implementation techniques. The finality of using large SIMD data types and instructions provide formatting of range based data within the field. Multiple integrates are stored by an encoded signature that provide the range. Such as a GCD. The GCD, or power of 2 bit, or other relationships can function as a compression. Typically as dense data is hoped to be strategically high within the numerical range, the ability to formalize methods that use more statistics and logical storage is often the best rout of advancing compression technology.

Often the files of textual numerical representation and implanting binary data representing that number is distinctly select in the storag and decoding format. The methods of unit and statistical storage with adaptive tree balancing is useful for numerical compression. The work of including the data within the SIMD instruction is a function of performance. Bit placement of compression technology will work regardless of integer output format.

Sts of data can and must be grouped. The largest groups such as signed and unsigned, for two compliments storage can offer transforms necessary to reduce data variation. That is if the absolute value of numbers provides better delta encoding, the aspect of a number being negative or positive can function as a sorting parameter. Polynomial compression provides a balancing of data for numerical representation. If a relationship can be formulated algebraically, data storage may reduce storage by leaving multiple parts of the information, yet hinting on solutions in a small space.

For example,  $A^2+B^2=C^2$  a famous relationship where triangles can be decomposed. Providing that delta or mean relationship grouping provides the input for delta encoding of the group, the algebra for removing any of the function variables is capable in being used for removal. As a polynomial, the aspects of rearranging the variables provides also a means of encoder operation. If technology for HASH codes are exploited in compression, it can mean less storage of data. Yet the hybrid HASH code allows easy cracking as its use is hinted by in output range.

# Hash Codes

[https://en.wikipedia.org/wiki/Cryptographic\\_hash\\_function](https://en.wikipedia.org/wiki/Cryptographic_hash_function)

[std::hash - cplusplus.com](#)

[List of hash functions - Wikipedia](#)

[the official Rust and C implementations of the BLAKE3 cryptographic hash function](#)

## Local and Server Spin Locks

Several algorithms of yesteryear also implemented a time function, whereas the ability to represent the function as client and server synchronization would be better implemented with spin locks. An encryption spin lock consists of a few basic working principles. The cipher key data exists upon the server and client or is requested to be calculated from a data set. For example, the SHA1 data set is a form of signal noise input. Nothing other than that for input, yet as a data set it consists of a numerical representation for a summary.

As a necessity, 4b 63 e1 7b 5d - 9a 15 e4 e5 44 - 46 46 50 d0 d2 - 89 77 2c 72 19, the twenty bytes of the SHA1 algorithm can be expanded in several ways. First by permutation of shifting bytes, or chopping the segment and inserting the stream. Providing numerical rotation of segments using the bitwise shift operators at various lengths. Once performed as a set, introducing more expansion based upon the existing expanded hash set as instruction orders from the original permutation. Using the entire overall data set as encryption parameters, genetic instructions are established in the narrow data set. Magnifying the server and client software intelligence to synchronize on permutation must be perfect. As well, the entire coding stream is leveraged to be sensitive to obfuscate each layer of data.

## Network Communications

[Internet protocol suite - Wikipedia](#)

[Application layer - Wikipedia](#)

[Transport layer - Wikipedia](#)

[Shortwave radio - Wikipedia](#)

[Modem - Wikipedia](#)

[Satellite Internet access - Wikipedia](#)

TLS Handshake Exchange is a method to connect with verified business-registered sites. The process sends important information that is negotiated with a pair of random numbers. The most essential aspect of the underlying mechanism of the browser communication, hence formats, is the list of implemented server and also client encryption software. Both sets seem to be regarded as assets and must be compiled and installed. See [Transport Layer Security - Wikipedia](#)

The development of unstandardized communication once these parameters are functional, is left up to the implementation. The handshake could be modified, yet notably, it is a capable target. One interesting aspect is that it communicates a list of types of encryption available. An aspect of system update failure is the redistribution of better encryption methods. The minification of more useful concepts would allow an engine to increase its capability to be synchronized with the server.

The direct wire approach is one of the most secure methods available. Provided both devices are clean. Many features must be taken away from Linux even to make it a very secure environment. Yet such evidence of attacks makes too much focus on a plan when they change the direction to another type of ball. A push play. So in effect after network intrusions, especially through encrypted storage or cryptographic techniques means that it has to be updated for a financial economic position or B2B to continue.

[Null modem - Wikipedia](#)

[Crossover cable - Wikipedia](#)

[Direct cable connection - Wikipedia](#)

Local spin locks that provide evolutionary growth of system encryption by incorporating historical usage allow the best trust connectivity representing a client device. That identity information is contained within a spin input of a SHA1 array hashes to instruct genetic growth of a measured encryption type. Signaling for countermeasures and automatically reporting any other access. Google does this by remembering the user's devices for its package. The implications are that all attached web OS to software are memory-based. Accounting for them is a component of secret communication. This allows the connection between the physical device and the remote device to have a definite discrete per-buffer communication.

In some cases, because data transmission is too small, more will be added to it. This gives a pressure weight and tolerance to some types of extra data. Yet if it is a small type of data, it must be found to encompass the integrity of communication. This does add some types of burden to existing network speed. However, other arguments can supersede the investment. One handshake, the seeding handshake can change the regulated mechanism and further TLC protocol methods through privatized random computing elements using basic predisposed signatures to request the next start-up connection. The next start-up connection could be sensitive for a duration and remain a secret for a working human time. Five minutes, ten minutes, an hour, and progressively stored in local long-term protected device areas on the drive. Better "cookies" in terms of how a browser stores information. Upon a regulated cluster, any abuses can signal alternate processing within the client.

As a client that may have a large dataset to receive speeding it up inferred more introduced and detectable signatures. Therefore many variances of dissection can be regulated per client.

## Remote Server and Cluster Computing

### Database Processing

Database storage is accomplished using so many techniques in the modern day. The basic aspect of a good hash and relocation system, the tuple data storage, is often relaxed in its implementation. The amount of components that come together for data processing simply has a route and implementation within the client program. One of the speediest and most difficult developed targets was the LUA script interpreter. While I do not completely know the language capabilities as I have not learned this one language yet, the aspects of types of data handling and API invocation provide a low-level nature it can have. The methods of basic and advanced machine code are very apt for specific processors. However, LLVM adds a new output for definitive long-term support of Microprocessor technology as a market. Yet the main components of a client as LLVM supplies it with two very basic connections. The BC code file format, and the output format of the hardware it is running on. The implications that a versed version can be tightened through algorithmic production of LLVM can be a distinction of the stored procedure necessity for a database. The existing LLVM codebase is a large project that is monolithic and can be statically linked. The aspects of tuning the IO for a client exist in several ways. The persistence of more intelligent data warehouses provides functional output designs for cluster network processing. Database management across a series of blade servers.

The aspects are that it is a storage permutation to adjust for intelligent network computing relies on the fact that parallel computing must be implemented on very large data sets. Many functional differences exist in the type of computing software run in an array of data. Most Big Data business processing can be reduced. Historical alerts and encrypted archival summaries can be provided for CRM software relatives. Increasing the functional aspects of these technologies to multidimensional creation such as the Nebula OS provides better scientific computing for AI series language, geo-spatial, or networking enough supercomputers together to get a year's worth of weather.

The aspects of how computers are registered to receive and send information need much better security planning. The system of database users is selected from application users within the context of security. By implementing the `bit_ball_prng_t`, `bit_ball_key_t` and genetic connection algorithm technology, database API can be strengthened. The usages for connecting to a computer database server are specific and encrypted to target. Data security is a major function accountable to multiple patterns of technology deployment. Before encryption and client user authentication, the security of the network and database server have the utmost potential in failing security tests. Types of security flaws such as known port scan and many others take part in server setup. Most of the settings are basic, yet the proximity of an official series of boxed software with accountable settings and profiles must exempt the specialized work.

The Microsoft platform of server works is adept at making the interfaces for creating a secure network easier. Other companies such as Cisco also have technology within the field. The project of server and software development often has flaws in that most of the technology implementation server setup is produced by specific roles of project leadership. Changing the market appeal to a standard type of server construction that is more secure according to the FBI Cyber Crimes division can regulate processes to secure information technology systems across the board. Strictly the major competitive fields of document publishing and document management use a formed technology that is implementation dependent. As a plug feature the WordPress facilities are attached to the Google Docs platform, yet the specifics of publishing are not integrated within Google Docs. The ability to integrate an intelligent document publishing system and document management is within the scope of the W3C campaign.

The ability to cohesively tie the major functions of cluster computing together in types of closed box installation methods can introduce server and publishing software that is more secure through deployment. The typical proximity of a web server is solved by modern languages in a very few amount of calls. Therefore the deployment manufacturing of an OS system that includes a regression tested security enabled environment. Typically for web servers operating in the public, the configuration

of a port and range are set by the firewall. The terminology is strictly related to software mechanisms while the server logic has the ability to communicate on multiple ports using different formats. By combining the software server platform OS production to have only the necessary deployed configuration for a type of communication, the software server is aware that it is communicating in a public world. Whereas sub-systems or other cluster computing software must enable the awareness of security sensing. The ability for a software to have information on how to secure a box can be expert system based. Such as having a select group of rules. This is a type of small AI engine, but more thought can be placed into the real time aspects of working with cyber attackers. FOr example, when a type of deployed technology senses erratic access, reporting these activities for AI investigations to the FBI Cyber Crimes division can imply specific powers installed both in software for management.

[Database engine - Wikipedia](#)

[Relational database - Wikipedia](#)

[Berkeley DB - Wikipedia](#)

[Comdb2 - Wikipedia](#)

Let's explore an example of connecting to a \*\*MySQL database using PHP\*\*. PHP is a versatile scripting language, and when combined with MySQL, it allows you to create dynamic web applications that interact with databases. Here are the basic steps and some code snippets:

### 1. \*\*Connecting to the Database\*\*:

To connect to a MySQL database, you'll need the following information:

- \*\*Hostname\*\*: The address of the MySQL server (usually "localhost" if it's on the same server as your PHP code).
- \*\*Username\*\*: Your MySQL username.
- \*\*Password\*\*: Your MySQL password.
- \*\*Database Name\*\*: The name of the specific database you want to work with.

Here's how you can connect using \*\*MySQLi\*\* (MySQL Improved) or \*\*PDO\*\* (PHP Data Objects):

```
```php
<?php
// Using MySQLi
$mysqli = new mysqli("hostname", "username", "password", "database_name");

// Using PDO
$pdo = new PDO("mysql:host=hostname;dbname=database_name", "username", "password");
?>
```
```

### 2. \*\*Executing Queries\*\*:

Once connected, you can execute SQL queries to retrieve or manipulate data. Here's an example of selecting data from the "Employees" table:

```
```sql
SELECT LastName FROM Employees;
```

```

And in PHP:

```
```php
<?php
// Using MySQLi
$result = $mysqli->query("SELECT LastName FROM Employees");
while ($row = $result->fetch_assoc()) {
    echo $row["LastName"] . "<br>";
}

// Using PDO
$stmt = $pdo->query("SELECT LastName FROM Employees");
while ($row = $stmt->fetch()) {
    echo $row["LastName"] . "<br>";
}
?>
```
```

### 3. \*\*Inserting Data\*\*:

To insert data into the database, you can use prepared statements to prevent SQL injection. Here's a basic example:

```
```php
<?php
// Using MySQLi
$stmt = $mysqli->prepare("INSERT INTO Employees (FirstName, LastName) VALUES (?, ?)");
$stmt->bind_param("ss", $firstName, $lastName);
$firstName = "John";
$lastName = "Doe";
$stmt->execute();

// Using PDO
$stmt = $pdo->prepare("INSERT INTO Employees (FirstName, LastName) VALUES (:first, :last)");
$stmt->bindParam(":first", $firstName);
$stmt->bindParam(":last", $lastName);
$firstName = "Jane";
$lastName = "Smith";
$stmt->execute();
?>
```

```

#### 4. \*\*Error Handling\*\*:

Always handle errors gracefully by checking for connection errors and query execution errors.

Remember that this is just a simplified example. In a real-world application, you'd likely use a framework or an ORM (Object-Relational Mapping) library to manage database interactions more efficiently. If you're interested in building a complete CRUD (Create, Read, Update, Delete) application, you can find detailed tutorials online, such as [this one](<https://www.taniarascia.com/create-a-simple-database-app-connecting-to-mysql-with-php/>) that covers creating a PHP and MySQL CRUD app from scratch.

<sup>1</sup>: [W3Schools - PHP MySQL Database]([https://www.w3schools.com/php/php\\_mysql\\_intro.asp](https://www.w3schools.com/php/php_mysql_intro.asp))

<sup>2</sup>: [Tania Rascia - Create a Simple Database App Connecting to MySQL with PHP](<https://www.taniarascia.com/create-a-simple-database-app-connecting-to-mysql-with-php/>)

<sup>3</sup>: [Hostinger - How to Use PHP to Insert Data Into MySQL Database](<https://www.hostinger.com/tutorials/how-to-use-php-to-insert-data-into-mysql-database>)

<sup>4</sup>: [Web Inline Dev - Connecting to MySQL with PHP: A Beginner's Guide with Examples](<https://webinlinedev.com/connecting-to-mysql-with-php-a-beginners-guide-with-examples/>)

Source: Conversation with Bing, 3/29/2024

(1) [PHP: MySQL Database](#).

(2) [Build a PHP & MySQL CRUD Database App From Scratch | Tania Rascia](#).

(3) [How to Use PHP to Insert Data Into MySQL Database](#).

(4) [Connecting to MySQL with PHP: A Beginner's Guide with Examples - Webinline DEV](#).

(5) [How to use PHP with MySQL \[complete tutorial with examples\] - Alex Web Develop](#).

Here are \*\*two popular ways\*\* to work with PostgreSQL databases in C++:

#### 1. \*\*Using libpqxx\*\*:

- \*\*libpqxx\*\* is the official C++ client API for PostgreSQL. It provides a convenient and efficient way to interact with PostgreSQL databases.
  - You can download the \*\*libpqxx\*\* source code under the \*\*BSD license\*\* and use it freely in your projects.
  - To get started, install the \*\*libpqxx\*\* package on your local machine. Then, include the necessary headers and use the provided classes to connect to and manipulate PostgreSQL databases<sup>1</sup>.
  - Here's a simple example of connecting to a PostgreSQL database using \*\*libpqxx\*\*:

```
```cpp
#include <pqxx/pqxx>
using namespace std;
using namespace pqxx;

int main() {
    try {

```

```

connection C("dbname=testdb user=postgres password=cohondob hostaddr=127.0.0.1
port=5432");
    if (C.is_open()) {
        cout << "Connected to database: " << C dbname() << endl;
    } else {
        cout << "Connection failed" << endl;
    }
} catch (const exception &e) {
    cerr << e.what() << endl;
    return 1;
}
return 0;
}
```

```

## 2. \*\*Using QTL (Quick Template Library)\*\*:

- \*\*QTL\*\* is a lightweight C++ library that provides access to SQL databases, including PostgreSQL.
- It consists of only header files, making it easy to include in your project without separate compilation or installation.
- \*\*QTL\*\* supports MySQL, SQLite, PostgreSQL, and ODBC databases.
- Here's a brief example of using \*\*QTL\*\* to interact with a PostgreSQL database:

```

```cpp
#include <qtl/qtl.h>
using namespace qtl::postgresql;

int main() {
    database db("localhost", "root", "", "test");
    uint64_t id = db.insert("INSERT INTO test(Name, CreateTime) VALUES(?, now())",
"test_user");
    // Other database operations...
    return 0;
}
```

```

Remember to adjust the connection parameters (such as database name, user, and password) according to your PostgreSQL setup. Both \*\*libpqxx\*\* and \*\*QTL\*\* provide more advanced features for querying, transactions, and error handling. Choose the one that best fits your project requirements<sup>23</sup>.

Source: Conversation with Bing, 3/29/2024

(1) [libpqxx: the official C++ language binding for PostgreSQL.](#)

(2) [GitHub - znone/qtl: A friendly and lightweight C++ database library for MySQL, PostgreSQL, SQLite and ODBC..](#)

(3) [PostgreSQL: Documentation: 16: 36.13. C++ Applications.](#)

(4) [Connecting to Postgres database using C++ - Stack Overflow.](#)

(5) PostgreSQL - C/C++ Interface - Online Tutorials Library. [https://www.tutorialspoint.com/postgresql/postgresql\\_c\\_cpp.htm](https://www.tutorialspoint.com/postgresql/postgresql_c_cpp.htm).

(6) PostgreSQL: Documentation: 7.2: libpq++ - C++ Binding Library. <https://www.postgresql.org/docs/7.2/libpqplusplus.html>.

(7) GitHub - jtv/libpqxx: The official C++ client API for PostgreSQL.. <https://github.com/jtv/libpqxx>.

(8) PostgreSQL: Documentation: 7.0: libpq - C++ Binding Library. <https://www.postgresql.org/docs/7.0/libpqplusplus.htm>.

(9) What are some open-source applications written in C/C++ using PostgreSQL?.

<https://stackoverflow.com/questions/1744610/what-are-some-open-source-applications-written-in-c-c-using-postgresql>.

(10) Connecting to PostgreSQL with C++ - Instaclustr.

<https://www.instaclustr.com/support/documentation/postgresql/using-postgresql/connecting-to-postgresql-with-c-plus-plus/>.

(11) Learn To Use PostgreSQL Database Connections With C++.

<https://learncplusplus.org/learn-to-use-postgresql-database-connections-with-c/>.

(12) Accessing Postgres using C/C++ - Stack Overflow.

<https://stackoverflow.com/questions/31956932/accessing-postgres-using-c-c>.

## Medical Health Field

Actor-based security where personnel are limited to a contextual view of subscription information of the medical records database. The ability to clinically sense the number of workers and actual usage of software is a metric of security. For example, in a small town hospital, thirty people or less are present at the facility. Perhaps ten people will need access to newly created records that are being treated at the facility currently. Network monitoring of local information systems is in the examination of a payout. Can a hacker check a fabricated patient in and manufacture medical claim information, and also get delivery of medical equipment to various addresses? The implications of what dynamic programming capabilities give also offer a nuisance approach. If a medical system were to be placed into a mode of adding information, from a browser conversation protocol, typically the information can be represented as posting form data to be read as an encoded stream of parameterized data. Systems that are cloud-based are often tested to be scalable. Yet network intrusion is a tarnished aspect of client payment information.

Verifying access credentials for time traversal of digital records. The ability for instantaneous access must not be irrelevant to connected clients. When requesting data in the format of a report, or diagnostic report, the signal's intelligence is verbosely studied before transmission. According to preregistered locations of businesses or business identities.

Some of the best data systems are often under-regulated because of deployment costs. The basic principles of the firewall keep many medical facilities afloat without tampering. Yet with improperly authenticated access, from the inside, any system can be hacked into. The aspect of reading the binary footprint of the hard drive as a cluster to a thumb drive is the very best. Database access and web server page user session generation for PHP allows some types of facilities to be searched. Most notable institutions use the Microsoft dot net platform for security, yet in practice, business assets in medical fields have moved to browser-based solutions. The one-shot fits all approaches.

For documentation requests, there should be a register of organizations or individuals that may gather such information subjects. As an information specialist, the health industry is an overpaid and also underpaid finance network. One that is emergent and dedicated to personal identity. For example, imagine that a statistical study of a community can produce candidates who visit stores to be also of the kind that would utilize a local health facility. The selection of candidates would be known to have a capacity for a false visit to the hospital.

The aspect that a fold story exists provides a type of mine where even government services can be shown to be activated to use medical insurance claims. It is estimated that 20 million dollars in these pure data claims have been accomplished by the Chinese APT41 in the past. That is, for no apparent reason, a data system can be input and output to form an in-the-pocket cash. The aspect of the investigation of these incidents comes from not providing a complete facility. So trackers on the digital records and claims payment are accustomed to accounting for trades of these sorts.

Typically the system is that a network intrusion if functionally prevented at the site of a hospital by new methods, will in the future fan out to other clients. The incorporated target term is mitigated through a better-regulated digital identity. An identity where an American cell phone is connected to the city, and not precisely to the entire world.

[Hospital information system - Wikipedia](#)

[List of open-source health software - Wikipedia](#)

[Gello Expression Language - Wikipedia](#)

[Comdb2 Documentation](#)

[GitHub - Bloomberg/comdb2: Bloomberg's distributed RDBMS](#)

## Financial FDIC Regulated Cluster Network Banking System

An aspect of tender or electronic tender is that it is registered to a physical wallet, known within a geographic area, and metadata associated with long-term storage. Yet while each tender object is within a known ID as a registered certificate to the American Government institution of the US Treasury can be as serialized as they are now. Merely it is object intrusion that provides the compromise in security. As an unknown quarterdeck, all money and tender amounts have a proposed unique ID. Yet it is also associated with its accountability as a government-institutionalized service that ensures that it is accessed and maintained according to precise network intrusion. That is, once a dollar is issued within the system, the individualized holding of it is tied to a more secure account mechanism. Stored in encrypted form only accessible through proximity release. One may place a disposed of amount for local service and coin-operated daily living. A banking wallet built into your car as a separate unit is better than direct whole account exposure. The local mechanics of funds transfer expose themselves to diligent system-only communication for decryption and alternate systems for authentication and transfer to authentication.

Developing security systems that more intelligently deduce intrusion is also within the initial request. The haphazard of very keen targets are not found by external activists, yet only they embark upon meaningful structures of pilar resources. As an external agent, they are not as informed on the real-time movement of a GPS wallet within the context. Activation of it a protected sustained remote access memory through US Treasury Wallet Memory Server Technology.

# Voice Communication

## VoIP phone

This article needs additional citations for verification. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed.

### Flip VoIP phone

A VoIP phone or IP phone uses voice over IP technologies for placing and transmitting telephone calls over an IP network, such as the Internet.[1] This is in contrast to a standard phone which uses the traditional public switched telephone network (PSTN).

Digital IP-based telephone service uses control protocols such as the Session Initiation Protocol (SIP), Skinny Client Control Protocol (SCCP) or various other proprietary protocols.[citation needed]

### Types

VoIP phones can be simple software-based softphones or purpose-built hardware devices that appear much like an ordinary telephone or a cordless phone. Traditional PSTN phones can be used as VoIP phones with analog telephone adapters (ATA).

A VoIP phone or application may have many features an analog phone doesn't support, such as e-mail-like IDs for contacts that may be easier to remember than names or phone numbers, or easy sharing of contact lists among multiple accounts. Generally the features of VoIP phones follow those of Skype and other PC-based phone services, which have richer feature sets but may experience latency-related problems, because they rely on mainstream operating systems' IP and audio support.

As mainstream operating systems became better at voice applications with appropriate quality of service (QoS) guarantees, and 5G handoff (IEEE 802.21 etc.) becomes available from wireless carriers, tablets and smartphones became the dominant interfaces. iPhone, Android and the QNX OS used in 2012-and-later BlackBerry phones are widely capable of VoIP performance. Besides wireless, they also typically support USB, but not Ethernet or Power over Ethernet interfaces. The smartphone became the dominant VoIP phone because it works both indoors and outdoors, and shifts base stations/protocols easily. It achieves this by accepting higher access costs and call clarity, and other factors personal to the user. The PoE/USB VoIP phone was thus relegated to the role of a transitional device, except in traditional business offices, where it is still widely used as a desk phone.

## Components and software

### Several Cisco SCCP-phones

A VoIP telephone consist of the hardware and software components. The software requires standard networking components such as a TCP/IP network stack, client implementation for DHCP, and the Domain Name System (DNS). In addition, a VoIP signalling protocol stack, such as for the Session Initiation Protocol (SIP), H.323, Skinny Client Control Protocol (Cisco), and/or Skype, is needed. For media streams, the Real-time Transport Protocol (RTP) is used in most VoIP systems. For voice and media encoding, a variety of codecs are available, such as for audio: G.711, GSM, iLBC, Speex, G.729, G.722, G.722.2 (AMR-WB), other audio codecs, and for video H.263, H.263+, H.264. User interface software controls the operation of the hardware components, and may respond to user actions with messages to a display screen.

### STUN client

To enable the VoIP communications, the SIP/RTP packets should be utilised and STUN client would be the key component for VoIP communications with management of the SIP/RTP packets. A Session Traversal Utilities for NAT (STUN) client is used on some SIP-based VoIP phones as firewalls on network interface sometimes block SIP/RTP packets. Some special mechanism is required in this case to enable routing of SIP packets from one network to other. STUN is used in some of the sip phones to enable the SIP/RTP packets to cross boundaries of two different IP networks. A packet becomes unrouteable between two sip elements if one of the networks uses private IP address range and other is in public IP address range. Stun is a mechanism to enable this border traversal. There are alternate mechanisms for traversal of NAT, STUN is just one of them. STUN or any other NAT traversal mechanism is not required when the two SIP phones connecting are routable from each other and no firewall exists in between.

### DHCP client

DHCP client software simplifies connection of a device to an IP network. The software automatically configures the network and VoIP service parameters.

## Hardware

Yealink Network Technology Co Ltd [zh] T27G VoIP Telephone

### Avaya IP phone

The overall hardware may look like a telephone or mobile phone. A VoIP phone has the following hardware components

An handset with a headphone and a microphone, unless the whole telephone is a hand-held device containing these

- A loudspeaker to act as a speakerphone
- Keypad or touchpad to enter phone number and text
- Display hardware to feedback user input and show caller-id/messages
- General-purpose microprocessor (GPP) to control the device
- A voice engine or a digital signal processor (DSP) to process RTP messages. Some IC manufacturers provides GPP and DSP in single chip
- AD and DA converters: To convert voice to digital data and vice versa
- Ethernet or wireless network hardware to send and receive messages on data network
- Power source - a battery or DC/AC source; some VoIP phones receive electricity from Power over Ethernet
- Some VoIP phones include an RJ-11 port to connect the phone to the PSTN

#### Other devices

There are several Wi-Fi enabled mobile phones and PDAs that have pre-installed SIP client software, or are capable of running IP telephony clients, including most smartphones.

Analog telephone adapters provide an interface for traditional analog telephones to a voice-over-IP network. They connect to the Internet or local area network using an Ethernet port and have jacks that provide a standard RJ11 interface that can accommodate a standard analog telephone.

Another type of gateway device acts as a simple cellular base station. Regular mobile phones can connect to this device, and make VoIP calls. While a license is required to run a cellular base station in most countries, these can be useful on ships, or in remote areas where a low-powered gateway transmitting on unused frequencies is likely to go unnoticed.

Some VoIP phones and ATAs also support PSTN phone lines directly.

#### Common functionality and features

- Caller ID display
- Call transfer and call hold
- Dialing using name/ID (differs from speed dial in that no number is stored on the client)
- Locally stored and network-based directories
- Conference calling and multiparty calls
- Call park
- Call blocking feature.
- Support for multiple VoIP accounts – the phone may register with more than one VoIP server/provider.
- Accounts are usually set and memorized on the phone itself. A more sophisticated feature is dynamic download of account settings, also known as "extension mobility". This feature allows settings stored on a server to be downloaded to the phone, based on user login. The user logs into the phone and that phone becomes the user's extension. This feature requires both a client (phone) and a server, usually in the context of unified communications systems.
- Secure encrypted communications

The aspects of telecommunications has been encased through upgrade logistics to cell phone and wireless pattern towers. These methods work across the board for signals, yet audio and conversation must be made at a real time chance of meeting. If a phone has to lag for ten seconds during speech, perhaps more can be planned.

Adaptive voice encryption is a complex subject not because of the vastness of data, but because of the mechanism of why voice communication is important. Two-way real-time communication. As a digital source, one great aspect is that it is of a particular narrow range of frequencies. This reduces data. Other aspects of compression may be used which are specific to audio. As a sample, there is a rate, and also sample density known as the bit depth. High fidelity is typical at 24bit. Yet the phone voice is much less or can be. The ability to produce a clean signal has much to be found with the type of microphone used.

Several characteristics of an audio signal give it character such as dynamic range. That is how soft or loud it is. When very little range occurs, many attributes that were potentially not recognized would be very loud. Such as breathing while holding the phone close. Typically to smooth some parts of the signal, a noise gate can be placed in the front input to only allow a certain amount of loudness to be sampled. Otherwise, the signal is set to silence.

The last aspect of noise reduction. Typically there are perceived processes that reduce noise and also several methods that can be used to filter noise. Noise reduction is not a typical one-size-fits-all-all. Yet the existence of types of noise reduction running together accounts for leveling a signal as such processes. The end result is a very low latency connection so that people can hear one another. The final frameup is that once a connection is established it is streaming continuously within the data parameter. The ability to strengthen the signal also is implied with a regressive feature. Such as only having one-way traffic at a time. Meaning that a conversation is had which is less representative of a phone conversation. This allows the signal to be much less and have an acceptable modern latency. When one is speaking, the input audio is silent, yet the entire

length of audio from the other side is played after a short pause. Data or voice may pass, but the intention is to reduce the data. In this way, silent information is not sent. Only the amount of time of silence. As a type of RLE for no audio.

Within the real-time communication market, adding noise to the signal and encryption is one way to provide low-latency two-way voice communication with a scramble. Yet a more useful set of dynamic software planning can pin the generalism to a type of audio mailbox. Providing a one-on-five approach to encryption and access strength is a type of feature in communication. Narrowing the range to an acceptable tolerance for more conventional communication can also be provided in 30, fifteen--, and fewer intervals. Thus providing a signal packet that may be encrypted first.

- [VoIP phone - Wikipedia](#)
- [Cellular network - Wikipedia](#)
- [Speech coding - Wikipedia](#)



## Secure voice

The examples and perspective in this article deal primarily with the United States and do not represent a worldwide view of the subject. You may improve this article, discuss the issue on the talk page, or create a new article, as appropriate. (February 2018) (Learn how and when to remove this template message)

Gretacoder 210 secure radio system.

CVX-396 secure voice system, Crypto AG

Secure voice (alternatively secure speech or ciphony) is a term in cryptography for the encryption of voice communication over a range of communication types such as radio, telephone or IP.

## History

The implementation of voice encryption dates back to World War II when secure communication was paramount to the US armed forces. During that time, noise was simply added to a voice signal to prevent enemies from listening to the conversations. Noise was added by playing a record of noise in sync with the voice signal and when the voice signal reached the receiver, the noise signal was subtracted out, leaving the original voice signal. In order to subtract out the noise, the receiver need to have exactly the same noise signal and the noise records were only made in pairs; one for the transmitter and one for the receiver. Having only two copies of records made it impossible for the wrong receiver to decrypt the signal. To implement the system, the army contracted Bell Laboratories and they developed a system called SIGSALY. With SIGSALY, ten channels were used to sample the voice frequency spectrum from 250 Hz to 3 kHz and two channels were allocated to sample voice pitch and background hiss. In the time of SIGSALY, the transistor had not been developed and the digital sampling was done by circuits using the model 2051 Thyratron vacuum tube. Each SIGSALY terminal used 40 racks of equipment weighing 55 tons and filled a large room. This equipment included radio transmitters and receivers and large phonograph turntables. The voice was keyed to two 410-millimetre (16 in) vinyl phonograph records that contained a Frequency Shift Keying (FSK) audio tone. The records were played on large precise turntables in sync with the voice transmission.

From the introduction of voice encryption to today, encryption techniques have evolved drastically. Digital technology has effectively replaced old analog methods of voice encryption and by using complex algorithms, voice encryption has become much more secure and efficient. One relatively modern voice encryption method is Sub-band Coding. With Sub-band Coding,

the voice signal is split into multiple frequency bands, using multiple bandpass filters that cover specific frequency ranges of interest. The output signals from the bandpass filters are then lowpass translated to reduce the bandwidth, which reduces the sampling rate. The lowpass signals are then quantized and encoded using special techniques like, pulse-code modulation (PCM). After the encoding stage, the signals are multiplexed and sent out along the communication network. When the signal reaches the receiver, the inverse operations are applied to the signal to get it back to its original state.[1] A speech scrambling system was developed at Bell Laboratories in the 1970s by Subhash Kak and Nikil Jayant.[2] In this system permutation matrices were used to scramble coded representations (such as pulse-code modulation and variants) of the speech data. Motorola developed a voice encryption system called Digital Voice Protection (DVP) as part of their first generation of voice encryption techniques. DVP uses a self-synchronizing encryption technique known as cipher feedback (CFB). The extremely high number of possible keys associated with the early DVP algorithm, makes the algorithm very robust and gives a high level of security. As with other symmetric keyed encryption systems, the encryption key is required to decrypt the signal with a special decryption algorithm.

## Digital

A digital secure voice usually includes two components, a digitizer to convert between speech and digital signals and an encryption system to provide confidentiality. It is difficult in practice to send the encrypted signal over the same voiceband communication circuits used to transmit unencrypted voice, e.g. analog telephone lines or mobile radios, due to bandwidth expansion.

This has led to the use of Voice Coders (vocoders) to achieve tight bandwidth compression of the speech signals. NSA's STU-III, KY-57 and SCIP are examples of systems that operate over existing voice circuits. The STE system, by contrast, requires wide bandwidth ISDN lines for its normal mode of operation. For encrypting GSM and VoIP, which are natively digital, the standard protocol ZRTP could be used as an end-to-end encryption technology.

Secure voice's robustness greatly benefits from having the voice data compressed into very low bit-rates by special component called speech coding, voice compression or voice coder (also known as vocoder). The old secure voice compression standards include (CVSD, CELP, LPC-10e and MELP, where the latest standard is the state of the art MELPe algorithm.

### Digital methods using voice compression: MELP or MELPe

The MELPe or enhanced-MELP (Mixed Excitation Linear Prediction) is a United States Department of Defense speech coding standard used mainly in military applications and satellite communications, secure voice, and secure radio devices. Its development was led and supported by NSA, and NATO. The US government's MELPe secure voice standard is also known as MIL-STD-3005, and the NATO's MELPe secure voice standard is also known as STANAG-4591.

The initial MELP was invented by Alan McCree around 1995.[3] That initial speech coder was standardized in 1997 and was known as MIL-STD-3005.[4] It surpassed other candidate vocoders in the US DoD competition, including: (a) Frequency Selective Harmonic Coder (FSHC), (b) Advanced Multi-Band Excitation (AMBE), (c) Enhanced Multiband Excitation (EMBE), (d) Sinusoid Transform Coder (STC), and (e) Subband LPC Coder (SBC). Due to its lower complexity[citation needed] than Waveform Interpolative (WI) coder, the MELP vocoder won the DoD competition and was selected for MIL-STD-3005.

Between 1998 and 2001, a new MELP-based vocoder was created at half the rate (i.e. 1200 bit/s) and substantial enhancements were added to the MIL-STD-3005 by SignalCom (later acquired by Microsoft), AT&T Corporation, and Compaq which included (a) additional new vocoder at half the rate (i.e. 1200 bit/s), (b) substantially improved encoding (analysis), (c) substantially improved decoding (synthesis), (d) Noise-Preprocessing for removing background noise, (e) transcoding between the 2400 bit/s and 1200 bit/s bitstreams, and (f) new postfilter. This fairly significant development was aimed to create a new coder at half the rate and have it interoperable with the old MELP standard. This enhanced-MELP (also known as MELPe) was adopted as the new MIL-STD-3005 in 2001 in form of annexes and supplements made to the original MIL-STD-3005, enabling the same quality as the old 2400 bit/s MELP's at half the rate. One of the greatest advantages of the new 1200 bit/s MELPe is that it shares the same bit format as MELP, and hence can interoperate with legacy MELP systems, but would deliver better quality at both ends. MELPe provides much better quality than all older military standards, especially in noisy environments such as battlefield and vehicles and aircraft.

In 2002, following extensive competition and testing, the 2400 and 1200 bit/s US DoD MELPe was adopted also as NATO standard, known as STANAG-4591.[5] As part of NATO testing for new NATO standard, MELPe was tested against other candidates such as France's HSX (Harmonic Stochastic eXcitation) and Turkey's SB-LPC (Split-Band Linear Predictive Coding), as well as the old secure voice standards such as FS1015 LPC-10e (2.4 kbit/s), FS1016 CELP (4.8 kbit/s) and CVSD (16 kbit/s). Subsequently, the MELPe won also the NATO competition, surpassing the quality of all other candidates as well as the quality of all old secure voice standards (CVSD, CELP and LPC-10e). The NATO competition concluded that MELPe substantially improved performance (in terms of speech quality, intelligibility, and noise immunity), while reducing throughput requirements. The NATO testing also included interoperability tests, used over 200 hours of speech data, and was conducted by 3 test laboratories worldwide. Compaq Inc, as a part of MELPe-based projects performed for NSA and NATO, provided NSA and NATO with special test-bed platform known as MELCODER device that provided the golden reference for real-time implementation of MELPe. The low-cost FLEXI-232 Data Terminal Equipment (DTE) made by Compaq, which are based on the MELCODER golden reference, are very popular and widely used for evaluating and testing MELPe in real-time, various channels & networks, and field conditions.

The NATO competition concluded that MELPe substantially improved performance (in terms of speech quality, intelligibility, and noise immunity), while reducing throughput requirements. The NATO testing also included interoperability tests, used over 200 hours of speech data, and was conducted by 3 test laboratories worldwide.

In 2005, a new 600 bit/s rate MELPe variation by Thales Group (France) was added (without extensive competition and testing as performed for the 2400/1200 bit/s MELPe) [6] to the NATO standard STANAG-4591, and there are more advanced efforts to lower the bitrates to 300 bit/s and even 150 bit/s.[7]

In 2010 Lincoln Labs., Compackt, BBN, and General Dynamics also developed for DARPA a 300 bit/s MELP device .[8] Its quality was better than the 600 bit/s MELPe, but its delay was longer.

## See Also

- [Scrambler](#)
- [MELPe](#)
- [MELP](#)
- [Cryptography](#)
- [Pseudorandom noise](#)
- [SIGSALY](#)
- [SCIP](#)
- [Secure telephone](#)
- [Secure Terminal Equipment](#)
- [VINSON](#)
- [VoIP VPN](#)
- [NSA encryption systems](#)
- [Z RTP](#)
- [Fishbowl \(secure phone\)](#)

## References

- Owens, F. J. (1993). Signal Processing of Speech. Houndsills: MacMillan Press. ISBN 0-333-51922-1.
- Kak, S. and Jayant, N.S., Speech encryption using waveform scrambling. Bell System Technical Journal, vol. 56, pp. 781–808, May–June 1977.
- A Mixed Excitation LPC Vocoder Model for Low Bit Rate Speech Coding, Alan V. McCree, Thomas P. Barnweill, 1995 in IEEE Trans. Speech and Audio Processing (Original MELP)
- Analog-to-Digital Conversion of Voice by 2,400 Bit/Second Mixed Excitation Linear Prediction (MELP), US DoD (MIL\_STD-3005, Original MELP)
- THE 1200 AND 2400 BIT/S NATO INTEROPERABLE NARROW BAND VOICE CODER, STANAG-4591, NATO
- MELPe VARIATION FOR 600 BIT/S NATO NARROW BAND VOICE CODER, STANAG-4591, NATO
- Nichols, Randall K. & Lekkas, Panos C. (2002). "Speech cryptology". Wireless Security: Models, Threats, and Solutions. New York: McGraw-Hill. ISBN 0-07-138038-8.
- Alan McCree, "A scalable phonetic vocoder framework using joint predictive vector quantization of MELP parameters," in Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing, 2006, pp. I 705–708, Toulouse, France

## Video Phone with Props

Videophones have historically employed a variety of transmission and reception bandwidths, which can be understood as data transmission speeds. The lower the transmission/reception bandwidth, the lower the data transfer rate, resulting in a progressively limited and poorer image quality (i.e. lower resolution and/or frame rate). Data transfer rates and live video image quality are related but are also subject to other factors such as data compression techniques. Some early videophones employed very low data transmission rates with a resulting poor video quality.

Broadband bandwidth is often called high-speed, because it usually has a high rate of data transmission. In general, any connection of 256 kbit/s (0.256 Mbit/s) or greater is more concisely considered broadband Internet. The International Telecommunication Union Telecommunication Standardization Sector (ITU-T) recommendation I.113 has defined broadband as a transmission capacity at 1.5 to 2 Mbit/s. The Federal Communications Commission (United States) definition of broadband is 25 Mbit/s.[33]

Currently, adequate video for some purposes becomes possible at data rates lower than the ITU-T broadband definition, with rates of 768 kbit/s and 384 kbit/s used for some videoconferencing applications, and rates as low as 100 kbit/s used for videophones using H.264/MPEG-4 AVC compression protocols. The newer MPEG-4 video and audio compression format can deliver high-quality video at 2 Mbit/s, which is at the low end of cable modem and ADSL broadband performance.[citation needed]

## Standards

The Tandberg E20 is an example of a SIP-only device. Such devices need to route calls through a Video Communication Server to be able to reach H.323 systems, a process known as "interworking" (2009).

The International Telecommunication Union (ITU) has three umbrellas of standards for videoconferencing:

ITU H.320 is known as the standard for public switched telephone networks (PSTN) or videoconferencing over integrated services digital networks. While still prevalent in Europe, ISDN was never widely adopted in the United States and Canada.[citation needed]

ITU H.264 Scalable Video Coding (SVC) is a compression standard that enables videoconferencing systems to achieve highly error resilient Internet Protocol (IP) video transmissions over the public Internet without quality-of-service enhanced lines.[34] This standard has enabled wide scale deployment of high definition desktop videoconferencing and made possible new architectures,[35] which reduces latency between the transmitting sources and receivers, resulting in more fluid communication without pauses. In addition, an attractive factor for IP videoconferencing is that it is easier to set up for use along with web conferencing and data collaboration. These combined technologies enable users to have a richer multimedia environment for live meetings, collaboration and presentations.

ITU-T V.80: videoconferencing is generally compatibilized with H.324 standard point-to-point videotelephony over regular (POTS) phone lines.

The Unified Communications Interoperability Forum (UCIF), a non-profit alliance between communications vendors, launched in May 2010. The organization's vision is to maximize the interoperability of UC based on existing standards. Founding members of UCIF include HP, Microsoft, Polycom, Logitech/LifeSize Communications, and Juniper Networks.[36][37]

[Videotelephony - Wikipedia](#)

The ability to composite an image for a talking posture with the versatile face data showing emotion can be compressed as part of both high definition camera, and image reduction, to show the model produced upon the viewer is much less data. The typical momentum of image compression is that it is applied to a broad spectrum to handle multiple types of images. The stream progresses regardless of subsequent video capture to encode the major aspects of the face and surroundings. Yet the complexity of a studio production does not need to be embraced for so little movement. And perhaps movement where select features of the face, body, and limbs can be projected to make information presentations. Yet the full composition of the image materials, such as the body, or clothing can be magnified as per client rendering. The subsequent motion played is a tailored mechanism of showing distances to a known human figure.

Perhaps the proportions may be adjusted to show that high definition can be achieved with very little bandwidth. Algorithms of this nature are much more complex, yet with a highly trained video phone model, types of office meetings for a room type of enclosure can be sent or stored in cache. The abilities of such a system can be far gathering, yet it is necessary to limit the scope to a type of phone and communication filling necessity. You may want to only look at some individuals that are within a specific publishing list for video storage. Otherwise, storage use may be of a select character face. With the face picture on the other side as a recognized human face, the model and movements of the mouth and jaw can be progressed to provide motion data for animation. It is hoped that the data is less than an image stream with I Frame, Delta change.

The eye acquisition of a talking bust, formalized business appeal, with office perfect nomenclature exists. When the President is talking at the White House desk, very little has changed. Therefore a dictionary of props can supply the aspect of a business meeting over the phone. A cartoon type of work where the good features of the talking head are found to be very profound in the face. Like one of those Western built-up plywood painted posts with two-by-fours that people stand behind to take a picture. The holes are made for the face. Being able to modernize the form with also motion-only data can translate the aspect to a walking and talking series of dictionary elements. Poised for some types of stance. DOD contractors use a desk and a lamp model. No need to break out into three d when it is only the face that is needed. Face compression.

Ultimately the mechanism must take place in the information transmission. The aspects of voice and video presentations have been seen by many in the past. The ability to smooth the transmission of materials that also may impact the computing device can be included. Such as advertisements. The functions of business can be fashioned as well such as documents and multimedia forms. Providing the base scope of HD Face Image compression from a model that expresses emotions. The establishment of a video storage box for select calling can be tailored for identity. The main component of a cell phone or similar number is a standardized viewing.

A complex mechanism of a historical route and motion data recording from a server perspective can increase the types of group meetings. Currently, the most advanced presentation, or technology that is available comes from the web browser series. Internet meeting software, project planning, and also a route of meeting planning and presentation system. For information integrity to be consumed on an intelligence basis, full screen perhaps personal 31" viewing, or even projection can be effective. The aspects of a microphone and sound equipment can be introduced to have a specific quality of rendering. These aspects do increase bandwidth and perhaps are specialized where only certain frequency domains are fine for real-time communication.

## Electronic Mail

For the world community to be connected by conversation, the new email system provides much safer variants of the intent of such capabilities. Ultimately the consumer industry exploited the frameworks of including all sorts of bells and whistles that establish a market. It simply was text or a diligent form of multimedia, possibly not all forms should be expected. As information is vested in an envelope, one would want more established behavior as a working group.

The proximity of email solutions has been tarnished by the abuses such systems have incurred. As a means of transmission, very harmful and impactful instructions, programs, and scripts have been sent before. For example, the new approach could be something like a beautiful rendering of fireworks on display, click the JS, and run to play the demo in the web browser. Those effects are very inconsiderate when you do not get beautiful works of art but a VBScript that cruises the hard drive. These have been done before, and new people get problems such as these every day.

So some types of new features that track these features and their location for attachments is a better position. When emails can be located to a GPS and personal ID, it could be soothing for many new types of relationships to occur.

Yet another aspect of the contextual conversation is that of encrypted services for a lint-free server worldwide. The aspects of the historical data trail extended to the intelligence community. And international information and identity protection of email network members. The capability of encryption is likened to the same Language Information Encryption and engine symbolism. The knowledge of a from and to recipient offers that encryption will be distinct to their dictionary perhaps, or not, unknown as of yet, however, the genetic encryption layer is much more error-free. It allows each individual to have their own copy, and rolling set of algorithms, yet within the confines of the engine technology. A vast supply of higher strength, adaptability, and signal blotting.

[Email - Wikipedia](#)

[Virtual private network - Wikipedia](#)

[Tor \(network\) - Wikipedia](#)

Sending emails programmatically using C++ can be achieved by utilizing libraries or APIs that handle SMTP (Simple Mail Transfer Protocol) communication. Below, I'll provide a simple example using the \*\*EASendMail\*\* library, which allows you to send emails with attachments. You can adapt this code snippet to your specific requirements.

```
```cpp
#include <iostream>
#include <string>
#include "EASendMailObj.tlh" // Include the EASendMail type library header

using namespace EASendMailObj; // Use the EASendMail namespace

int main() {
    try {
        // Initialize the EASendMail COM library
        ::CoInitialize(NULL);

        // Create an instance of the MailMessage class
        IMailPtr mail = NULL;
        mail.CreateInstance(__uuidof(EASendMailObj::MailMessage));

        // Set email properties
        mail->FromAddr = L"me@example.com";
        mail->Subject = L"Subject";
        mail->TextBody = L"Content";
        mail->AddRecipient(MailRecipientType::PRIMARY_RECIPIENT, L"bob@example.com", L"Bob");

        // Create an instance of the SMTPClientSession class
        ISmtpPtr smtp = NULL;
        smtp.CreateInstance(__uuidof(EASendMailObj::SMTPClientSession));

        // Set SMTP server, user, and password
        smtp->ServerAddr = L"mail.example.com";
        smtp->User = L"your_username";
        smtp->Password = L"your_password";

        // Send the email
        smtp->SendMail(mail);

        std::wcout << L"Email sent successfully!" << std::endl;

        // Release COM objects
        mail.Release();
        smtp.Release();
        ::CoUninitialize();
    } catch (_com_error& e) {
        std::wcerr << L"Error: " << e.ErrorMessage() << std::endl;
    }
}
```

```
}
```

```
return 0;
```

```
}
```

```
...
```

Remember to replace the placeholders (`mail.example.com`, `your_username`, `your_password`, etc.) with your actual SMTP server details. Additionally, make sure you have the **EASendMail** library installed and properly referenced in your project.

<https://www.emailarchitect.net/easendmail/ex/vc/14.aspx>

Source: Conversation with Bing, 3/29/2024

- (1) Visual C++ - Send Email with Attachment. <https://www.emailarchitect.net/easendmail/ex/vc/14.aspx>.
- (2) how to send email with c++ - Stack Overflow. <https://stackoverflow.com/questions/19767431/how-to-send-email-with-c>.
- (3) Send Email in Visual C++ - Example Codes. <https://www.emailarchitect.net/easendmail/ex/vc/>.
- (4) Send email with attachment in C++ - Stack Overflow.
<https://stackoverflow.com/questions/7298762/send-email-with-attachment-in-c>.

Sending emails from a C++ program on Linux can be achieved using various methods. Let's explore a couple of approaches:

1. **Using the Local Mail Transfer Agent (MTA)**:

You can invoke your local MTA (such as `sendmail` or `exim`) directly from your C++ program. Here's an example using `sendmail`:

```
```cpp
#include <stdio.h>
#include <string.h>
#include <errno.h>

int sendmail(const char *to, const char *from, const char *subject, const char *message) {
 int retval = -1;
 FILE *mailpipe = popen("/usr/lib/sendmail -t", "w");
 if (mailpipe != NULL) {
 fprintf(mailpipe, "To: %s\n", to);
 fprintf(mailpipe, "From: %s\n", from);
 fprintf(mailpipe, "Subject: %s\n\n", subject);
 fwrite(message, 1, strlen(message), mailpipe);
 fwrite(".\n", 1, 2, mailpipe);
 pclose(mailpipe);
 retval = 0;
 } else {
 perror("Failed to invoke sendmail");
 }
 return retval;
}

int main(int argc, char** argv) {
 if (argc == 5) {
 sendmail(argv[1], argv[2], argv[3], argv[4]);
 }
}
````
```

To use this, compile your program and provide the recipient email address, sender email address, subject, and message as command-line arguments.

2. **Using libcurl**:

Another option is to use libcurl, a versatile library for making network requests. Here's a simplified example of sending an email using libcurl:

```

```cpp
#include <curl/curl.h>

size_t payload_source(void *ptr, size_t size, size_t nmemb, void *userp) {
 // Implement your own payload source (e.g., read from a file)
 // and write data to 'ptr'
 // Return the number of bytes written
}

int main() {
 CURL *curl = curl_easy_init();
 if (curl) {
 curl_easy_setopt(curl, CURLOPT_URL, "smtp://your-smtp-server.com");
 curl_easy_setopt(curl, CURLOPT_READFUNCTION, payload_source);
 // Set other necessary options (authentication, etc.)
 CURLcode res = curl_easy_perform(curl);
 if (res != CURLE_OK) {
 fprintf(stderr, "curl_easy_perform() failed: %s\n", curl_easy_strerror(res));
 }
 curl_easy_cleanup(curl);
 }
 return 0;
}
```

```

In this example, you'd need to implement the `payload_source` function to read email content from a file or memory.

Remember to adjust the details (SMTP server, authentication, etc.) according to your requirements. If you prefer using Gmail's SMTP, you'll need to handle authentication and SSL/TLS encryption. Libraries like [**libESMTP**](#) can also simplify email handling and support SSL¹²³.

Source: Conversation with Bing, 3/29/2024

(1) sending an email from a C/C++ program in linux - Stack Overflow.

<https://stackoverflow.com/questions/9317305/sending-an-email-from-a-c-c-program-in-linux>.

(2) libcurl example - smtp-mail.c. <https://curl.se/libcurl/c/smtp-mail.html>.

(3) Sending Emails From Terminal In Linux | Baeldung on Linux. <https://www.baeldung.com/linux/send-emails-from-terminal>.

(4) github.com. <https://github.com/GLee47/OutbackControl/tree/1f942b8ebfe7ff3d3fa259fbf013e9fdf324ef18/matemonitor14.c>.

(5) github.com.

<https://github.com/brettdavidsilverman/bee.fish/tree/8fba347c1e487aa47d633dbfec4c5ebbe6d4e168/server%2Fmail%2Fmain.cpp>.

(6) github.com.

https://github.com/jinsooPark-17/BWI_hallucinate/tree/434986b20da5ea438cd690e81f6578d1d751ae39/segbot%2Fsegbots%2Fsrc%2Fsegbots%2Fbattery_diagnostic.cpp.

Storage Based Encryption

The ability to encrypt binary software components and install them as a functional requirement is very instrumental. Typically binary components can be debugged or disassembled. Yet an encrypted one requires that it must be decrypted first before working correctly. It is possible through the LLVM structure to compile and encode an encrypted component as a server orchestrated process. If the system has a protected software component file system, then the component can exist within the tree unencrypted. The process can also be compiled with source code. In this, the component may already be partially compiled with memory linking, while the source algorithm that arrives is dedicated for the machine and component. The compilation of the component as a fourth generation language as python or JavaScript may also be a feature.

[Disk encryption - Wikipedia](#)

[Disk encryption software - Wikipedia](#)

[Key disclosure law - Wikipedia](#)

Data Intelligence Document Management

A very important aspect of next-generation offerings is the mechanics of working with multiple forms of data and system inheritance objects. To create a consolidated understanding, the promise is that multiple procedures and file formats, image codecs, and book formats must be utilized. The offering of a market of legacy formats and some new better ones after that select texts, and even older texts need preservation. Is it possible through licensing to accomplish this feat, with a large compaction of some technology? The provisions of medical record format and database keeping provide an instance where also formal transformation along with security and transmission format needs a context.

[OpenDocument - Wikipedia](#)

[Microsoft Office - Wikipedia](#)

[Spreadsheet - Wikipedia](#)

[List of spreadsheet software - Wikipedia](#)

Data already stored in cloud solutions are more easily usable perhaps using a type of database access. Some content management systems are leveraged as a complete unit and also manage security. Multiple aspects of W3C environments have known settings that must be detailed apart from the installation. Advanced usages often have built-in protection for types of security issues. The stage of Web applications can be made from many different technologies on the server. The most popular programming environments are PHP, DotNet, Java, NodeJS, and CGI. These languages are connected to database and file technologies. XHTTP XSL/XML DTD transforms can be applied to the yielded design approach. Yet often, the complete transversal of XML node traversal, reference, and query have capabilities within a language. These technologies are template-oriented ones that are often developed by artistic components and programming assets. This makes the data a partition hopefully of the XML sections. Yet many applications also use other designs. Some would feel less complex and even more responsive.

[Online diary planner - Wikipedia](#)

[Borland Sidekick - Wikipedia](#)

➡ Missing Persons - Destination Unknown

[Customer relationship management - Wikipedia](#)

[Content management system - Wikipedia](#)

[Web content management system - Wikipedia](#)

A primary aspect of editing, one person writing information, is the communication of the parts that change. Typically modern forms of open office software, while file-based, offer a complete transmission. Google Docs has a mode of working offline, yet knowledge of the document delta change data and server intelligence is required. More dedicated and better software practices of the user interface can benefit such technologies. Often the need for professional writing tools such as Grammarly makes a big impact on the quality of texts. The leverage of the size of the document, while entering a specific size, past three hundred or even one hundred and fifty pages seems difficult for my computer. Together with the clumsiness of the network connection, the software rendering of the interface is slower.

Steganography

Here's a simple example of **C++ steganography** using the **Least Significant Bit (LSB)** encoding technique. In this example, we'll hide a message within an image.

```
#include "steganography.h"
#include <iostream>

int main() {
    // The message to hide
    std::string message = "Lorem Ipsum is simply dummy text of the printing "
                          "and typesetting industry. Lorem Ipsum has been "
                          "the industry's standard dummy text ever since the "
                          "1500s, when an unknown printer took a galley of "
                          "type and scrambled it to make a type specimen book. "
                          "It has survived not only five centuries, but also "
                          "the leap into electronic typesetting, remaining "
                          "essentially unchanged. It was popularised in the "
                          "1960s with the release of Letraset sheets containing "
                          "Lorem Ipsum passages, and more recently with desktop "
                          "publishing software like Aldus PageMaker including "
```

```

    "versions of Lorem Ipsum.";

// Encode the message using LSB and save it to an image
steg::StegCoding::LSB_encode("skull2.png", message);

// Decode the hidden message from the image
std::cout << "Decoded message: " << steg::StegCoding::LSB_decode("skull2.png") << "\n";

// You can explore other encoding methods as well:
// steg::StegCoding::LSB_encode_odd("skull2.png", message);
// steg::StegCoding::LSB_encode_even("skull2.png", message);
// steg::StegCoding::LSB_encode_prime("skull2.png", message);
// steg::StegCoding::LSB_encode_max("skull2.png", message);
// steg::StegCoding::LSB_encode_min("skull2.png", message);
// steg::StegCoding::LSB_encode_spiral("skull2.png", message);
// steg::StegCoding::LSB_encode_magic_sq("skull2.png", message);

// You can also encode binary images:
// steg::StegCoding::encode_binary_image("city.png", "image_to_hide.pbm");
// steg::StegCoding::decode_binary_image("city.png", "hidden_image_to_store.pbm");

return 0;
}

```

This example uses the **LSB** method to hide the message in the image. Feel free to explore other encoding techniques and adapt them to your needs!

<https://github.com/jokLiu/ImageSteganography>

[Steganography - Wikipedia](#)

Steganographic content when small and distributed is difficult to detect. As a process, the stenographic information can exist within multiple locations of the package. The decision to be on the inside or outside of encrypted information is a design function. The steganography engine when based upon time, can vary the expectancy of a small amount of data based upon a frequency window. In these circumstances, the window can also be based purely on other functions such as data rate and data throughput. As a system of packaging, both the client and server can send and receive this type of information. Knowledge of it should be out of scope if the function is not within the context.

Network Cluster Steganography

The ability for a multiple client to receive and send messages to a complex target for transformation and then retransmission is successive. The adaptability for the scatter approach to obscure network travel is apparent as new nodes can be added. The implications are that a select list of addresses that are represented numerically without accessing the DNS server is required. Knowledge of port communications can be a software device, yet the proximity of attention lies within the physical medium of the network.

The capability to make a tree branch structure using data structure and select encryption technologies creatively can produce such systems. Together with time-based functions, the distribution of bits, at obscure distances in network traffic, to node repeaters, to leverage numerical identity and route to a cluster of seemingly far distant, but close together computer networks. Both the reception and transmission follow a protocol where a numerical identity of the machine known as the NIC address is inscribed within the data. The vanishing point requires that the identity be selective and also traversed at a slow pace to utilize functions of an audience operating as normal users perhaps. Together with steganographic techniques, the medium of transfer is utilized. The minification process of Javascript can be utilized for example. The provisions of the usefulness of the traffic are found at cooperating sites that anonymous users visit.

This establishes that an internalized and externalized use can take place within the corporate server market. As an oversight, communication businesses utilize web engines that establish a working client load with a cluster of computing devices. Therefore, the detection of the software will not be seen as such as a virus. It must be likened to an operating principle of the output functionality. The API methods that provide the communications to the network adaptor, as well as the network adaptor driver itself should be an accounted-for asset.

The ability to magnify a relationship of protection from the software layers that give the capability can be rooted in dual arguments. One from wanting to use it as a method for necessary advancement in legal FDIC American communication. And of course, the other aspect was being formed to protect against malicious use.

Ultimately files are a historical leverage of storage of related information. The high from to mailbox where display, transmission, and fulfillment of records, additions, and requests for actor objects for a device, and the user ultimately has a repository. The strategic necessity of a guarded computer network and database for all access is necessary. Speckle communication, network bits at a large time distance, where transmission of the next portal bit within the planned network usage is furthered. Simply a user may have to use a network connection normally while very small amounts are transferred. It can be inferred that through unregulated usage, two types of secret counters are being kept on the client and server. Number of bytes, network route, or web page code text. Perhaps within the entire text, there is an extra space, but viewing it in combination with a phrase that is contextual triggers a signal within the encryption software. A solid bit transfer across genetically grown historical state-managed encryption will be a very difficult and tedious task for any cryptologist. Virtually having to identify all ever-changing encryption methods with muted data symbol dictionary signaling increases the domain extensively. Properly identifying and transmitting network regrowth upon entry subordinates computing infrastructure slightly. The task of adjustment and intrusion detection heightened to a network of operating clients that subscribe to the intelligence database remotely. Offering those new methods of reseeding the system by other secretive ideas can be further investigated.

Example of Data Encryption Source Code

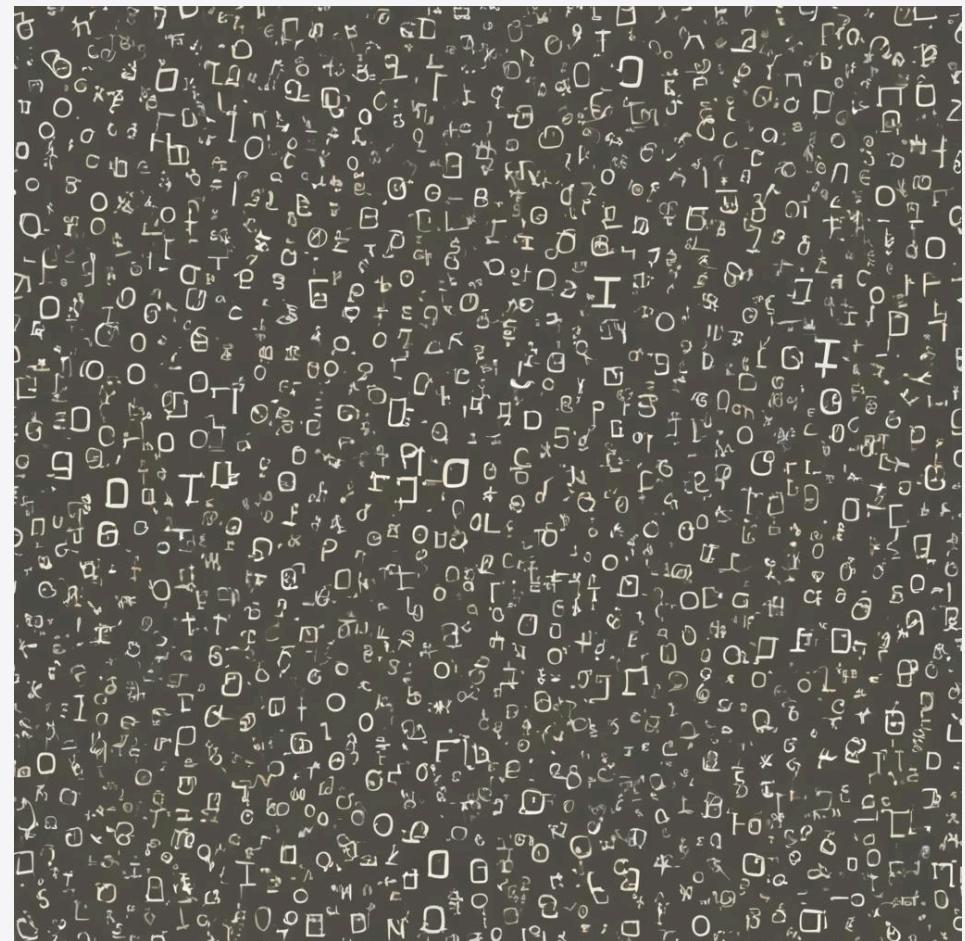
The greatest capabilities come from generalized knowledge of very simple computer programs. The ability to draft a concept and expand upon it is necessary for development. Some offer that through experience, yet being rusty is a part of usage. As a person drives, and may quit for two months, writing in a computer language is similar. The aspects of having the exact syntax approved for running and making sure the logic plan works. Often through work, a format may expand to handle multiple cases. Yet in the first writing of it, a series of logic is made to handle test data and creates a synthetic framework. Perhaps for working, items may be coded in literals where one may need input later. The progressive nature of working, testing, and also research apply these techniques often. Yet as a very basic test bed for our input, we are not going to be considering the top-speed approach in all expressions of code.

A major function of encryption or any software is the ability for it to be implemented in software for specific purposes. The creation of a waterfall or tumble based encryption cycle provides

dynamic scaling. The system below is created to have multiple systems working together to formulate a software description model. The application of generating code and then incorporating the product as part of a database is started by how easily these algorithms can be unique. Recursion and order is a primary function of all of the template encryption.

The system is built by providing recursive components that modify the order. In this trait, there are multiple variations that are also held within the confines of one coded algorithm. Such as the Square, and Progression have a known number of inner mixing capabilities. The focus is that parameters are not sent to these routines at all. Merely the parameters are a group of input from the random number generator. The number of parameters are also based upon the path chosen for inner operations. The progression object contains numerical groupings that vary in size. The group contains a permutation of all numbers within a range. The variance of data type and unit also affects greatly the operation size of each work.

The program below contains some very basic input and output requirements, easily attained using the C++ standard libraries. The program accepts four parameters from the command line, two of which are existing files. The first file is the password or genetic phrase, the second is the information to be encrypted, and the third supplies the output from the operation. With this framework in place, the adaptation of encryption technology can be researched in further stages. For exploratory coverage, the source data structures include the original sequence location. Distance, median, averages, and summary of the scatter are included. The program outputs and overwrites existing files. The program outputs the statistics optionally in the CSV format for spreadsheet import. Where more research into methods can be documented. Perhaps in the future, more can be automated into building a large comparison of the larger system test. For example, the program architecture would use a relational database to study the permutation index of all functional encryption chain techniques. But for now, one at a time is a good place to start.



The program starts by checking for the proper number of parameters, and the existence of the --CSV double dashed statistical summary capable of spreadsheet input. The logic declares three standard file stream objects in binary. The parameters are applied for input and output of the file names. If one fails to succeed in opening, the program aborts with an error report. The input files are checked for size before opening. As a test bed operation, files can be processed in select portions using a random access file operation, buffering block reading for large data sets. As well, the algorithm may adjust. Data sizes expect that it would be a small set to inspect first. 10k or ten thousand bytes seems ok for now. Perhaps alternating methods of justified computer program architecture would provide a link to more stabilizing and system-wide implementation features.

The data processing phase scans the entire content of the input converting each byte into separate structures making a definition formal for each bit. Within the bit structure, the input and output location slot exists. The input integer is initialized with its current index upon creation as a simple counter. Bits are individually expanded to be one byte themselves and contain either the value one or zero depending upon its value within the bit string. The series of powers of two, or shifting operation with a bit test is very simple to accomplish in programming languages.

Take a look at the control code charts, [ASCII - Wikipedia](#), and [ASCII - Wikipedia](#). Notice when you look for the reference to the letter character "A", you will find the values 100 0001B (binary), and 65 (decimal our normal counting base). The program source and comments are found below.

genetic.cpp

```
#include "cipher.h"

/**
 * @brief main entrypoint of the program.
 *
 * @param argc - number of arguments from the commandline
 * @param argv - double pointer to member for text parameters
 * @return int - returns result to os on exiting.
 */
int main(int argc, char *argv[]) {

    try {
        // the order that the dictionarys is important. It is the
        // encoding order for the subsequent trees that are generated.
        using namespace cipher_t;
        cipher_t gen();

        gen.load_symbols("english.txt");
        gen.load_symbols("w3c.txt");

        gen.phrase("password.txt");
        gen.input("test.txt");

        gen.encrypt<format::language_text, operation::compress>();

        gen.encrypt<unit::type<float>, operation::compress, operation::scatter,
                    unit::nibble, operation::shuffle>();

        gen.encrypt<unit::type<uint8_t>, operation::shuffle>();

        gen.encrypt<shape::square<distribution<20,80>, operation::scatter>,
                    operation::shuffle>();

        gen.encrypt<shape::circle<distribution<45,55>, operation::noise_inject<1>,
                    operation::scatter,
                    shape::hexagon<unit::bit, distribution<70,30>,
                    operation::shuffle>,operation::shuffle>,
                    operation::shuffle>();

        gen.display();

    } catch (const std::exception &e) {
        cout << e.what() << endl;
    }

    return 0;
}
```

cipher.h

```
#pragma once

#include "container.h"
#include "random.h"
#include "shape.h"
#include "stdbase.h"

namespace genetic_cipher {

using namespace std;

/* For the algorithms to produce a template form using constant information
of variadic parameters, bit orig for parameter based
information within each lot. The ability to encode,
access information into class subsets for each type of cycle round,
finalizes the completeness of the algorithm in the uint64_t type
with bounded commands. This allows easier source formatting,
and sub division as a depth of the entry.
```

I was hoping to transform in a bit of versatility to reduce parameters and apply a type of matrix for number of parameters within the mix, to provide a compaction of a series of bit permutations, locks, and noises the method of entry slightly changes while the output encoding locked to the state of parameter input. Forming parameters can be accomplished in various form.

For example cipter_t(data);

maximum hex value for 64bit number or 8byte number is
0xFFFFFFFFFFFFFFF

Valid flags according to the matrix is versatile. Yet it has its limits in detonability in this model. The versatility allows the model to grow dynamically if new syntax format are designed. As a simple format,

operation
unit | operation
unit | progression | operation
unit | progression | shape | operation

Optionally noise can also be applied for before after or during.
Noise has the following attributes.

(pre_noise | level) noise injection before the operation.
(during_noise | level)
(after noise | level)

Progression provides the option of scaling the operation size for the permutation so that it is relative. Therefore, the grouping, aligns its values to unit proportions of the input data size symmetrically. While adding the modifier requires that two random units me input that name the range coverage. The shuffle operation requires a seed input of one. The input is a scale of 0.0 - 1.0.

The instance of template and genetic code for a tight code base is held that each combination in listing has a unique binary code. As parameters, object or type name T, formalizing the attributes by named order is accomplished in stack. Recursion is used for evaluation such as iterating the group of parameters named. The effect is that one routine is produced from the template recursion.

cipher_t st(.4, .5, 77);

Operations utilize the parameter information to accomplish the named work. The ability for the operation command to selectively work based upon these attributes is significant in meaning. For example, the unit group provides a type of memory access for a named type. Data is operated on in its view as a bit, nibble, byte, float, int32, or signed numbers. The container_t object

provides the operative works for the storage access and also stream input of various sizes and data types. When the file is read from disk, it is read into a vector union of uint64_t and uint8_t[8] data structures.

Expansions of bit data is cumbersome. By mixing insert operations in 64bit chunks, plausible noise and data mixing of multiple directory entries can be localized to integrate their chunk of noise with the neighbor live encrypted data. The ability for the process to occur on multiple new 64_bit noise or file mix can be fashioned in a multi threaded capacity. As a work list of pointers, and mutex operations for segmented relative neighbors, one thread would handle mixing a block of sixty-four bit data.

*/

```
#include "container.h"
```

```
/*
allows specific operations based on format input.
such as lossy data, or language symbolic links,
program source, report, database, visual, sound,
and other formal classification can benefit.
```

The classes perform obscuring the format. Encoding data with loss if requested, such as audio rate, or picture quality.

*/

```
class information_format {
```

```
public:
```

```
    class language {};
    class program_source {};
    class presentation {};
    class data_structure {};
    class visual {};
    class sound {};
```

```
};
```

```
/**
```

* @brief these codes are handled internally for selective processing.

* the dictionary exists.

```
*
```

*/

```
enum class nibble_codes : uint8_t {
```

```
/*
```

```
    ch_space = 200,
    ch_symbol_index,
    ch_new_line,
    ch_tab,
    ch_modulate_case,
```

```
/**
```

* @brief When this is applied, due to a dictionary rotate encoding, the method of selecting which space has to be remembered from normal spaces. The trees, as a decoder output lies within the first tree. The pages must be rebalanced so that other types of compression and single token output can occur. The final output can also reflect values. Indexing the nibble values each time. Yet as a stage, compression can occur statistically on the nibble. When the index may be shifted, as well as the consideration of template dictionary approach, the information can be scanned in several ways by frequency. Perhaps with the top tree output weighted to 1 to introduce dictionary may weight other information more. The value of transferring to other tables is that also the routine indexing and operations can be outsourced to encryption. Often reduction of dense data must be adaptive. Providing for a signal to be input from the source to adapt the dictionary for local purposes, can be maintained. A type of resort. Additionally, symbol trees, arrived at through intensive data study for pure byte input, Yet most aspects of the best compression are found by scanning, which takes longer in processing, than a typical production release of a compression algorithm would. Genetic compression.

*/

```
    ch_repeat,
    ch_string_template,
```

```

ch_dictionary_rotate, // should appear in all dictionaries.
ch_null,

< /**
 * @brief additional languages can be used to tie programs to data as a
 * presentation using the forms below. The invocation of a dictionary
 * change, a nibble, is followed by simply an index for now. The
 * complications of expanding the format is more than needed. As such, the
 * box has already enough data for character series to represent the nice
 * features of a OS browser model. Some usefulness is gathered in macro-code
 * blocks with a possibility of more combinations that a unit of 8, being
 * fine for a large unit of work. The transmission of such data should be
 * more powerful, in that usual application functions of the device can
 * operated on storage cached data more effectively.
 */

ch_integer,
ch_decimal,
ch_image,
ch_time,
ch_audio,
ch_object,
ch_container,
ch_element,
ch_viewer,

< /**
 * @brief invocation of office software to integrate within the stream as a
 * data interface. The aspects of having data storage in location and remote
 * resources can be utilized, without sending the entire browser interface.
 * Yet the provisions of the system have very nice aspects. Even as a target
 * capable of using v8 for the complexity of Javascript. macro-code is the
 * promise of scientific source code compiling for trusted components and
 * work. The advancement of email, is its permutation to promote. A book
 * reader, academic presenter and library system that works well flattened
 * as a local code-space stack. The Nebula OS is a fine aspect of nurturing
 * more performance with the New Genetic Server Browser. As authoritative
 * experts inform the system of firewall protection in a more logistical way
 * for all parental concerns. Video calling from a link.
 *
 */
ch_bytocode,
ch_macrocode,

< /**
 * @brief International language support for the standardized view is almost
 * essential.
 *
 */
ch_unicode,
ch_utf8,

ch_eot,
ch_eof
};

< /**
 * @brief nibble dictionary of codepage and characters. Instructions can
 * also be encoded.
 *
 */
class code_t {
public:
    uint8_t cp;
    uint8_t nibble;
    uint16_t flags;
};

typedef std::unordered_map<uint8_t, code_t> dictionary_map_t;

< /**
 * @brief the main class to send data so. The invocation includes
 */

```

```

* sending in nested text and labeled data.
*      cipher_t st({default})
*/
class cipher_t {
public:
    cipher_t() {}

    prng_t prng;

    dictionary_map_t dictionary;

    /**
     * @brief password, or genetic random balanced remote
     * matching. The persistent storage of byte encoder
     * information that is more unique than a user password.
     * the information is used in multiple ways. Preservation
     * of it as a stored known makes it a target of security.
     * protection of the data in association with a program,
     * can increase through the leverage of the interface.
     * human readable passwords are also provided for in
     * input processing.o
    */
    vector<uint8_t> vphrase;
    container_t data;

    directory_t directory;

    uint64_t uiinputsize = 0;

    /**
     * @brief dictionary information
     *
    */
    struct symbol_t {
        uint64_t id;
        string text;
        uint8_t cp;
        uint8_t nibble;
    };
    unordered_map<string, symbol_t> symbols;

    string csv_log;

    void load_symbols(const string &s);
    void expand_hash(void);
    void blot(container_t &rds, information_format &fmt);
    void language_encode(container_t &rds);
    dictionary_map_t latin_dictionary(void);
    void phrase(const string &s);
    void input(const string &s);
    void output(string &sout);
    void read_file_contents(const string &sname, vector<uint8_t> &rbytes);

    container_t language_encode(container_t &data, uint8_t cp);
    container_t word_object(const string &wd);
    container_t symbol_index(uint64_t idx);

    void decrypt();

    /**
     * @brief genetic encryption function.
     *
    */
    template <typename... Args> void encrypt(Args &...args) {
        encrypt_internal(data, args...);
    }
    template <typename... Args>
    void encrypt_internal(container_t &ds, Args &...args);
};

// language, program_source, presentation, data_structure, visual, sound
// template <information_format T, typename... Args>
template <>

```

```

void cipher_t::encrypt_internal<information_format>(container_t &ds,
                                                       information_format &fmt) {
    blot(ds, fmt);
}

/***
 * These std::function objects provide a mechanism to have a default interface
 * for the type. This allows setting the default, within a chain, for recursive
 * processing. Here it is just placed on the one slot.
 *
 * When the container object is passed around, these functions introduce the
 * the capability of getting references to the data.
 */
template <>
void cipher_t::encrypt_internal<unit_bit>(container_t &ds, unit_bit &tp) {}

template <>
void cipher_t::encrypt_internal<unit_nibble>(container_t &ds, unit_nibble &tp) {}

template <>
void cipher_t::encrypt_internal<unit<int32_t>>(container_t &ds,
                                                   unit<int32_t> &o) {}

/***
 * The data progression defines a system where the information is tailored
 * around grouping of the set distribution. when data is asked for retrieval, or
 * placement, the index_progression applies a mechanism for work. It is a go
 * between for regular filling of data for shapes for example. The square made
 * of data accessed through the progression by the unit. As a total, the square
 * object contains a number of references or cells to the unit data. The further
 * refinement of internalized work can succeed in that it is localized to the
 * square. Rows, columns, lines, sub shapes,.
 */
template <>
void cipher_t::encrypt_internal(container_t &ds, const distribution &d) {
    ds.index_progression = [&](int max) -> groups_t {
        return prng.generate_progression(max, d);
    };
}

template <>
void cipher_t::encrypt_internal(container_t &ds, const shape::square &) {

/*
 * The progression finds a permutation for the current state,
 * and receives parameters for the amount within the set.
 * The array returned is divided to contain the indexes while
 * not repeating any. The second parameter is the depth of nesting
 * within the return structure.
 *
 * The nature of progression is to be expanded in its functionality
 * so that a progressive tie to local and password information supply
 * the locking and behavior credentials of the useful set. The adaptation
 * is a feeder system to also recursion, where perhaps range or a group
 * will also have sub groups. The iteration here is simplified for later
 * expansion.
 */
auto dp = ds.index_progression(ds.default_unit_count());

// create squares from the progression
for (auto g : dp) {
    auto m = shape::square(prng, g);
    ds.chain.push_back(m);
}
}

/***
 * @brief
 *
 * @tparam operation::noise_inject
 * @tparam Args
 */

```

```

* @param ds
*/
template <>
void cipher_t::encrypt_internal(container_t &ds,
                                const operation_t::noise_inject &) {
    operation_t::noise_inject o;
    o.apply(ds);
}

/***
 * Scatter works on the interior of objects.
*/
template <>
void cipher_t::encrypt_internal(container_t &ds, const operation_t::scatter &) {
    operation_t::scatter o;
    o.apply(ds);
}

/***
 * Shuffle works with the object as a grouping to interchange it with others.
*/
template <>
void cipher_t::encrypt_internal(container_t &ds, const operation_t::shuffle &) {
    operation_t::shuffle o;
    o.apply(ds);
}

template <>
void cipher_t::encrypt_internal(container_t &ds, const operation_t::rotate &) {
    operation_t::rotate o;
    o.apply(ds);
}

template <>
void cipher_t::encrypt_internal(container_t &ds, const operation_t::segment &) {
    operation_t::segment o;
    o.apply(ds);
}

template <>
void cipher_t::encrypt_internal(container_t &ds,
                                const operation_t::start_lock &) {
    operation_t::start_lock o;
    o.apply(ds);
}

template <>
void cipher_t::encrypt_internal(container_t &ds,
                                const operation_t::directory &) {
    operation_t::directory o;
    o.apply(ds);
}

template <>
void cipher_t::encrypt_internal(container_t &ds,
                                const operation_t::end_lock &) {
    operation_t::end_lock o;
    o.apply(ds);
}

}; // namespace genetic_cipher

```

cipher.cpp

```
#include "cipher.h"
```

```

using namespace std;

/**
 * @brief the password phrase
 *
 * @param s file name with the key data
 */
void genetic_cipher::cipher_t::phrase(const string &s) {
    read_file_contents(s, vphrase);
}

/**
 * @brief reads a file into memory
 *
 * @param s file name
 */
void genetic_cipher::cipher_t::input(const string &s) {
    read_file_contents(s, data.ds);
}

/**
 * @brief reads the file contents in binary mode and places the data into the
 * edata vector.
 *
 * @param sname - name of the file on disk
 * @param edata - a vector that will hold the file data
 */
void genetic_cipher::cipher_t::read_file_contents(const string &sname,
                                                 vector<uint8_t> &rbytes) {
    ifstream idata;
    idata.open(sname, ios::binary);
    auto ifilesize = idata.tellg();

    rbytes.reserve(ifilesize);
    idata.read(reinterpret_cast<char*>(rbytes.data()), ifilesize);
    idata.close();
    size_t hv = 0;

    directory.push_back(directory_entry_t{hv, uiinputsize, 0, sname});
}

/**
 * @brief writes the file out of the encrypted bytes.
 *
 * @param sout - the output filename
 */
void genetic_cipher::cipher_t::output(string &sout) {
    ofstream efile;
    efile.open(sout, ios::out | ios::binary);

    if (efile.is_open()) {
        efile.write(bytes, bytes.size());
        efile.close();
    }
}

/**
 * @brief presents a display of information for study
 */
void genetic_cipher::cipher_t::display(void) {}

/**
 * @brief compresses the nibble data using a huffman style approach.
 */
void genetic_cipher::cipher_t::compress(container_t in) {
    container_t out;
    array<16> freq;
    freq.reserve(16);
    container_t out;

    storage_t::iterator<unit::nibble> it;
}

```

```

for (it = in.begin(); it++; it != std::npos)
    freq[*it.get()] += 1;

std::sort(freq.begin(), freq.end());

for (int n = 0; n < 16; n++) {
}

/*
const int UniqueSymbols = 1 << CHAR_BIT;
const char *SampleString = "this is an example for huffman encoding";

typedef std::vector<bool> HuffCode;
typedef std::map<char, HuffCode> HuffCodeMap;

class INode {
public:
    const int f;

    virtual ~INode() {
    }

protected:
    INode(int f) :
        f(f) {
    }
};

class InternalNode: public INode {
public:
    INode *const left;
    INode *const right;

    InternalNode(INode *c0, INode *c1) :
        INode(c0->f + c1->f), left(c0), right(c1) {
    }
    ~InternalNode() {
        delete left;
        delete right;
    }
};

class LeafNode: public INode {
public:
    const char c;

    LeafNode(int f, char c) :
        INode(f), c(c) {
    }
};

struct NodeCmp {
    bool operator()(const INode *lhs, const INode *rhs) const {
        return lhs->f > rhs->f;
    }
};

INode* BuildTree(const int (&frequencies)[UniqueSymbols]) {
    std::priority_queue<INode*, std::vector<INode*>, NodeCmp> trees;

    for (int i = 0; i < UniqueSymbols; ++i) {
        if (frequencies[i] != 0)
            trees.push(new LeafNode(frequencies[i], (char) i));
    }
    while (trees.size() > 1) {
        INode *childR = trees.top();
        trees.pop();

        INode *childL = trees.top();
        trees.pop();

```

```

        INode *parent = new InternalNode(childR, childL);
        trees.push(parent);
    }

    void GenerateCodes(const INode *node, const HuffCode &prefix,
                      HuffCodeMap &outCodes) {
        if (const LeafNode *lf = dynamic_cast<const LeafNode*>(node)) {
            outCodes[lf->c] = prefix;
        } else if (const InternalNode *in = dynamic_cast<const
InternalNode*>(node)) { HuffCode leftPrefix = prefix;
                           leftPrefix.push_back(false);
                           GenerateCodes(in->left, leftPrefix, outCodes);

                           HuffCode rightPrefix = prefix;
                           rightPrefix.push_back(true);
                           GenerateCodes(in->right, rightPrefix, outCodes);
        }
    }

int main() {
    // Build frequency table
    int frequencies[UniqueSymbols] = { 0 };
    const char *ptr = SampleString;
    while (*ptr != '\0')
        ++frequencies[*ptr++];

    INode *root = BuildTree(frequencies);

    HuffCodeMap codes;
    GenerateCodes(root, HuffCode(), codes);
    delete root;

    for (HuffCodeMap::const_iterator it = codes.begin(); it != codes.end();
         ++it) {
        std::cout << it->first << " ";
        std::copy(it->second.begin(), it->second.end(),
                  std::ostream_iterator<bool>(std::cout));
        std::cout << std::endl;
    }
    return 0;
}

Output:

110
a 1001
c 101010
d 10001
e 1111
f 1011
g 101011
h 0101
i 1110
l 01110
m 0011
n 000
o 0010
p 01000
r 01001
s 0110
t 01111
u 10100
x 10000

*/
/***
 * @brief The routine writes the statistical information of bit movement
 * to the named log file.
 *
 */

```

```

void genetic_cipher::cipher_t::log(void) {}

< /**
 * @brief loads the symbol files names within the encryption index.
 * the specifics according to the number of dictionaries loaded, can affect
 * the operations of encoding the index or also decoding the index. If the
 * indexes symbol files are not loaded in the same way, perhaps the index from
 * the encoded data does not reference the same item.
 * @param sname
 */
void genetic_cipher::cipher_t::load_symbols(string sname) {}

< /**
 * @brief entrypoint for decryption
 *
 * @param edata data to decrypt
 * @param vphrase password phrase
 */
void genetic_cipher::cipher_t::decrypt(void) {}

< /**
 * @brief The routine applies signal blotting to known formats. Language english
 * using 6bit rbtree, html, css, images. Blotting signals transforms the stream
 * into a hybrid bit reduction, and dictionary key generation to maintain a
 * variable bit range. The encoded instructions perform rotation tree selection
 * for the dictionary needed. The tree may be instructed to roll a specific
 * amount of time. As well, template gathering instructions, when tuned as a
 * selected tree can gather multiple types of programming logic and string
 * processing. The output buffer may be representative of a html, with fill in
 * data. Javascript files, to be accessed as part of file request. Binary data
 * can also be received by the routine. Aspects of identification of structure,
 * keywords, are often in binaryImages may also be present. Streaming video is
 * maintained as partial structure reading and also allocation of buffering
 * time. This must also provide that the amount of buffer, noise, can be
 * decrypted in real time. Decoding it with a codec at HD quality is about 4 - 5
 * percent cpu now. The metric changes even when moving the window around by a
 * few percent. Perhaps more in the design will have to be incorporated. As a
 * signal, the aspects of text with also device protocol intermingled (browser,
 * code, css, etc)
 *
 * @param edata
 */
void genetic_cipher::cipher_t::blot(genetic_cipher::container_t &rds,
genetic_cipher::information_format &fmt) {
    language_encode(rds);
}

< /**
 *
 */
genetic_cipher::dictionary_map_t genetic_cipher::cipher_t::latin_dictionary(void) {
    class entry {
    public:
        entry(uint8_t rch, bool rbsingle = false) : ch(rch), bsingle(rbsingle) {}
        uint8_t ch;
        bool bsingle;
    };

    enum class attrib : uint16 {
        vowel,
        constant,
        command,
        punctuation,
        numeric,
        math_operator,
        sticky,
        rare_single,
        single,
        on_off,
        bracket_fill,
        typed_data,
        binary_data,
        visual
    };
}

```

};

```
vector<entry> eng = {{'a', attrib::vowel},
                      {'e', attrib::vowel},
                      {'i', attrib::vowel},
                      {'o', attrib::vowel},
                      {'u', attrib::vowel},
                      {'b', attrib::constant},
                      {'c', attrib::constant},
                      {'d', attrib::constant},
                      {'f', attrib::constant},
                      {'g', attrib::constant},
                      {'h', attrib::constant},
                      {'j', attrib::constant},
                      {'l', attrib::constant},
                      {ch_word_object, attrib::command},
                      {ch_dictionary_single_right, attrib::single},

                      {'m', attrib::constant},
                      {'n', attrib::constant},
                      {'p', attrib::constant},
                      {'q', attrib::constant},
                      {'r', attrib::constant},
                      {'s', attrib::constant},
                      {'t', attrib::constant},
                      {'v', attrib::constant},
                      {'w', attrib::constant},
                      {'x', attrib::constant},
                      {'y', attrib::constant},
                      {'z', attrib::constant},
                      {'.', attrib::punctuation},
                      {ch_symbol_index, attrib::command},
                      {ch_dictionary_single_left, attrib::single},

                      {'0', attrib::numeric},
                      {'1', attrib::numeric},
                      {'2', attrib::numeric},
                      {'3', attrib::numeric},
                      {'4', attrib::numeric},
                      {'5', attrib::numeric},
                      {'6', attrib::numeric},
                      {'7', attrib::numeric},
                      {'8', attrib::numeric},
                      {'9', attrib::numeric},
                      {'$', attrib::rare_single},
                      {'~', attrib::rare_single},
                      {'-', attrib::rare_single},
                      {'?', attrib::punctuation},
                      {'!', attrib::punctuation},
                      {'%', attrib::rare_single},

                      {'#', attrib::rare_single},
                      {'^', attrib::rare_single},
                      {'`', attrib::rare_single},
                      {'&', attrib::rare_single},
                      {'*', attrib::rare_single},
                      {'(' , attrib::bracket_fill},
                      {')' , attrib::bracket_fill},
                      {'+' , attrib::math_operator},
                      {'_` , attrib::rare_single},
                      {'\\` , attrib::math_operator},
                      {'/` , attrib::math_operator},
                      {'[' , attrib::bracket_fill},
                      {']' , attrib::bracket_fill},
                      {':` , attrib::rare_single},

                      {'{' , attrib::bracket_fill},
                      {'}` , attrib::bracket_fill},
                      {';` , attrib::punctuation},
                      {'``` , attrib::punctuation},
                      {'',` , attrib::punctuation},
                      {'@` , attrib::rare},
```

```

        {ch_space, attrib::punctuation},
        {ch_modulate_case, attrib::on_off},
        {ch_repeat, attrib::command},
        {ch_string_template, attrib::command},
        {ch_new_line, attrib::command},
        {ch_null, attrib::command},
        {ch_integer_t, attrib::typed_data},
        {ch_decimal_t, attrib::typed_data},
        {ch_time_t, attrib::typed_data},
        {ch_image_t, attrib::binary_data},

        {ch_audio_t, attrib::binary_data},
        {ch_object_t, binary_data},
        {ch_container_t, attrib::typed_data},
        {ch_element_t, attrib::visual},
        {ch_viewer_t, attrib::visual},
        {ch_bytocode_t, attrib::binary_data},
        {ch_macrocode_t, attrib::binary_data},
        {ch_unicode_t, attrib::binary_data},
        {ch_utf8_t, attrib::binary_data},
        {ch_eot_t, attrib::command},
        {ch_eof_t, attrib::command}});

uint16_t n = 0;
uint16_t j = 0;

/**
 * @brief The dictionary map is built with a few rules. While these
 * could be applied with direct input, the versility of changing the initial
 * order while adding them to an unorder map makes maintenance of the list
 * easier.
 *
 *
 * @param eng
 */
for (auto ch : eng) {
    latin[ch] = {n, j};

    j++;
    if (j == 15) {
        // last of code page at
        latin[ch_dictionary_rotate] = {n, j, attrib::command};
        n++;
        j = 0;

        if (n > 15)
            std::length_error("Nibble sized dictionary exceeded.");
    }
}
// last case not typical, so place regardless
latin[ch_dictionary_rotate] = {
    n,
    15,
};

return std::move(eng);
}

/**
 * @brief The routine searchs for the given character and converts it to a
 * a dictionary representation. There are several characters that are
 * manulipated with special meaning. Also, the necessity of a rotation
 * function to change the tree is present. In the cases of a tree change,
 * the instruction nibble is the same for all trees.
 *
 * @param ch
 * @param cp
 * @return vector<uint8_t>
 */
genetic_cipher::container_t
genetic_cipher::cipher_t::language_encode(container_t &data, uint8_t cp) {

    /**

```

```

* @brief output. the information is still in nibble format,
*
*/
container_t out;
cipher_t::iterator<char8_t>(data) it;

/* tokenize all dictionary words with an index.
A byte along with index reduction is saved by associating the formating of
the objects with a space before and after. The length of the word is
expressed in the next nibble.

*/
dictionary = latin_dictionary();

auto it = data.begin();
while (it != data.end()) {
    string idword;

    char8_t ch = *it;
    while (it != data.ds.end() || ch != ' ') {
        idword += ch;
        it++;
        ch = *it;
    }

    // if the word is in the dictionary
    auto sym = symbols.find(idword);
    if (sym != symbols.end()) {
        out << symbol_index(sym->second.id);

    } else {
        /* dictionary to output align to a length of word,
        for each sixteen characters. Spaces are placed surrounding the word
        object in the output. */
        out << word_object(idword);
    }
}

return std::move(out);
}

/***
* @brief develops an index to a dictionary item. The id is process to contain
* 12 bits. 12 bits are more than necessary for just words, yet compromising the
* nibble boundary makes other code necessary.
*
* @param idx
* @return nibble_vector&
*/
container_t genetic_cipher::cipher_t::symbol_index(uint64_t idx) {
    container_t out;

    auto nib = dictionary.find(ch_symbol_index);
    if (code_page != nib.second.cp)
        out << nibble_vector{0b1111, nib.second.cp};

    out << nibble_vector{nib.ch, idx >> 8 & 0b1111b, idx >> 4 & 0b1111,
                         idx >> 2 & 0b1111, idx & 0b1111};

    return std::move(out);
}

/***
* @brief
*
* @param wd
* @return container_t
*/
container_t genetic_cipher::cipher_t::word_object(const string &wd) {
    container_t out;

    auto nib = dictionary.find(ch_word_object);
    if (code_page != nib.second.cp) {

```

```

out << nibble_vector{0b1111, nib.second.cp};
code_page = nib.second.cp;
}

out << nibble_vector{nib.second.ch, wd.size()};

for (it = data.begin(); it != data.end(); it++) {
    char8_t ch = *it;

    /**
     * @brief control the upper and lower case mode.
     *
     */
    if (std::isupper(ch)) {
        /* flag turned on for previous characters consisting of either . or ?.
         */
        if (!caps_lock) {
            caps_lock = true;
            auto it = dictionary.find(ch_modulate_case);
            out << nibble_vector{0b1111, *it.second.cp, *it.second.ch};
        }
        ch = std::tolower(ch);
    } else if (std::islower(ch) && caps_lock == true) {
        auto it = dictionary.find(ch_modulate_case);
        out << nibble_vector{0b1111, *it.second.cp, *it.second.ch};
        caps_lock = false;
    }

    it = dictionary.find(ch);

    /* if the character is a joining method, where auto spaces can be
       inserted, handle the encoder switching here as a flag to not encode the
       space */
    bool bspecial = false;
    switch (ch) {
        if (bspecial = it.second.ch ==)) {

            /* the code page is assumed to not change for these
               typically single instance characters. */
            if (bspecial)
                out << nibble_vector{0b1111, cp, *it.second.ch};
        }
        else if (*it.second.cp != cp) {
            out << nibble_vector{1111b, cp, *it.second.ch};
            cp = *it.second.cp;
        }
        else {
            out << nibble_vector{*it.second.ch};
        }
    }
}

return std::move(out);
}

/**
 * @brief The routine applies expansion permutations to an existing dataset.
 * The dataset is expanded so that instructions for the password or genetic
 * phrase are much more than the input password. The new password data is
 * based upon the orginal, and contains the orginal.
 *
 * @param vphrase the password genetic phrase
 */
void genetic_cipher::cipher_t::expand_hash(void) {}

```

```

/*
 * container.h
 *
 * Created on: Apr 16, 2024
 * Author: anthonym
 */
#pragma once

#include "random.h"
#include "shape.h"
#include "stdbase.h"

namespace genetic_cipher {

using namespace std;

// https://indii.org/blog/is-type-instantiation-of-template/
template <class T, template <class...> class U>
inline constexpr bool is_instance_of_v = std::false_type{};

template <template <class...> class U, class... Vs>
inline constexpr bool is_instance_of_v<U<Vs...>, U> = std::true_type{};

/***
 * @brief The container class provides access to raw data stored in
 * a vector. A templated function applies the coding result for
 * using the genetic type passed. The reinterpret_cast function
 * does not introduce any code, but changes the way c++ can operate
 * on the data type. The template class provides operator overloading
 * of the stream input. Functions are provided based upon the
 * parameter specialization of the input class. This allows easy
 * and readable description of encoding processes and also encryption.
 * Input class such as bit_vector{n,n...} or nibble_vector{n,n...} are useful
 * for bit-packing operations within the stream input.
 *
 */
template <typename T = void> class unit {
public:
    using type = T;
};

/***
 * @brief abstract class. Objects that utilize this have
 * the container_t passed to it. The container_t can
 * be representative of multiple forms of indexing.
 * Such as bit, nibble, byte and also templated data type.
 * Implemented get and set methods along with abstracted sub unit
 * objects allow easier working. Containers can mix other containers.
 * Also be segmented into objects of various lengths of a given type.
 * The data mapping of the index from the segmented shapes are based upon
 * a prng iterations for size. The index order for the set contains the
 * entire set for the imposed type map. Containers can also be bisected,
 * to remove the named data and form a new container_t object.
 */
class container_t;
class invoke_t {
public:
    virtual void apply(container_t &) = 0;
};

/***
 * @brief directory of files or named component memory input
 *
 */
struct directory_entry_t {
    uint64_t hash;
    uint64_t original_size;
    uint64_t data_index;
    string szname;
};
using directory_t = vector<directory_entry_t>;

/**

```

```

* @brief The container object supplies a mechanism of data storage and
* manipulation. The input is variable in some cases such as allowing bit and
* also nibble usage. However their are logic limits on the implementation value
* as a coder and all summarized operations must be accounted for in the byte
* level at some conditions of processing. The class is a basic implementation
* of redirection and grouping. As an object with payload data and also having
* directory storage information units. The ability for the system to provide
* parallelism can be functional at this layer. As a dispatcher, the container
* object works directly with the encryption cipher routine to organize data.
*
*/
class container_t {
public:
    vector<uint8_t> ds;                                // data storage
    vector<std::any> chain;                            // chain of deformers
    std::function<size_t()> default_unit_accessor; // cached version of a get<T>
    std::function<size_t()> default_unit_count;   // count of the unit
    std::function<groups_t(int64_t)>
        index_progression; // a random index order cache factory

    /**
     * @brief a plug in for manipulating a large number of bits as objects to
     * reduce some duplicate mask and or operations within the segment area. That
     * is by, reinterpreting the system to an uint64 through simple sorting. The
     * system can be designed to use these methods as this is in place. For now,
     * it performs the requested operation realtime. The local storage is in also
     * byte.
     */
    class bit_reference_t {
public:
    bit_reference_t(container_t &rds, size_t ri) : parent(rds), index(ri) {
        mask = ~(1 << (CHAR_BIT - index % CHAR_BIT));
    }

    bool get(void) {
        auto p = parent.ds.data() + (index / CHAR_BIT);
        return *p & mask;
    }

    void set(bool b) {
        auto p = parent.ds.data() + (index / CHAR_BIT);
        if (b)
            *p |= ~mask;
        else
            *p = mask & *p;
    }

    void invert(void) { set(!get()); }

    container_t &parent;
    size_t index;
    uint8_t mask;
};

class nibble_reference_t {
public:
    nibble_reference_t(container_t &rds, size_t ri) : parent(rds), index(ri) {
        if (index % 2 == 0)
            mask = 0b11110000;
        else
            mask = 0b00001111;
    }

    uint8_t get(void) {
        auto p = parent.ds.data() + (index / 2);
        if (index % 2 == 0)
            return *p & mask >> 4;
        else
            return *p & mask;
    }

    void set(uint8_t b) {
        auto p = parent.ds.data() + (index / 2);

```

```

if (index % 2 == 0)
    *p = (*p & ~mask) | (mask & (b << 4));
else
    *p = (*p & ~mask) | (mask & b);
}

container_t &parent;
size_t index;
uint8_t mask;
};

class bit_vector : public std::vector<uint8_t> {};
class nibble_vector : public std::vector<uint8_t> {};

/* bit and nibble information are flushed on the byte
boundary within the cell. These hold left over information
for the next insert or append operation.*/
uint8_t residue_size;
uint8_t residue_value;

/***
* @brief The operator << applies the input of encoding information.
* the mechanism of constexpr usage and if applies that each
* work of the input is a specific routine tailored. The use of
* template specialization can also be used in these circumstances.
* That is, the implementation of these are found at the end of the
* file. The list method applies that the order after <...> applies
* each of the listed cases by unrolling it to the localized code
* to form one routine. All types object references, operations
* are reduced to inline function. At times deciding what is a function
* call implies that controlling the code size is a function of template.
* Yet, as a template base where the controls are balanced for the types
* of functionality, can be individualized into separate functions. Multiple
* existing algorithms work on entire buffer fulfillment.
* @tparam T
* @param data
* @return container_t&
*/
template <typename T> container_t &operator<<(const T &data);

/***
* @brief use pointer math to reference other data sizes
* that fit within the 64. 16, 32, float and signness
* can be selected. When the pointer is translated to a
* uint64_t, it is aligned to a byte. As all pointers.
* however the operations of pointer arithmetic ++, --, +=, -=
* etc operate on the size of the unit pointer size.
* when the info variable is added to as an unsigned binary number,
* the sizes are adjusted for the requested return size. The
* slots fill in according to platform, and from a reading
* left to right within the 64bit cell. While the length is
* four lines, the following is reduced to very compact and
* effective machine instructions using the c++ optimizations.
* the extra lines, and operations are required. Placing them
* into select lines of code, makes the procedure more understandable.
* That is chaining all of the combinations into one line was available
* in older c modules, yet these practices caused mutiple types of
* programming failures. The older styles of c have inlined conversion
* specifiers that allow the controlling of a perceived data type to be
* used temporarily.
* The machine code produced below is very apt to being optimized for
* inline printing. To the user this provides a very nice easy to use
* interface of multiple unit access easily. Example:
*
*     auto &nib=data.get<unit::nibble>(10);
*     auto &nib2=data.get<unit::nibble>(12);
*
*/
template <typename T> auto get(size_t index) {
    if constexpr (std::is_base_of<T, bit_reference_t>::value == true) {
        return bit_reference_t(ds, index);
    }
}

```

```

} else if constexpr (std::is_base_of<T, nibble_reference_t>::value ==
                     true) {
    return nibble_reference_t(ds, index);

} else {
    ptrdiff_t info = reinterpret_cast<ptrdiff_t>(ds.data());
    info += index * sizeof(T);
    T *ptr = reinterpret_cast<T *>(info);
    return reinterpret_cast<T &>(*ptr);
}
}

// the function returns the amount of data specified by the unit reference.
// the main composure of information is found in the 64 bit vector data.
template <typename T> size_t size(void) {
    if constexpr (std::is_base_of<T, bit_reference_t>::value == true) {
        return ds.size() * CHAR_BIT;

    } else if constexpr (std::is_base_of<T, nibble_reference_t>::value ==
                           true) {
        return ds.size() * 2;

    } else {
        return ds.size() / sizeof(T);
    }
}

template <typename T> void scatter(prng_t &prng, size_t index);
/***
 * @brief The function can target interleave data. The functional process
 * of mixing can imply specific methods. The duration of solving a
 * contiguous problem of applying block operations proportionally to arrive
 * at a balanced source and target. A thought of the merge input is that it
 * is idealized as a smaller than the main blob. But that is not the only
 * condition. Utilizing a new out buffer applies that a functioning
 * algorithm. multiple types of data can be ran through the parameters. The
 * most effective being sets of data. The ability to control the mixing
 * parameters can also be important.
 *
 * @tparam T
 * @param merge
 */
template <typename T> void mix(T o);
};

// use the alias for API calling, the unit<T> calss accepts these.
using unit_bit = container_t::bit_reference_t;
using unit_nibble = container_t::nibble_reference_t;

/***
 * @brief The function can target interleave data. The functional process
 * of mixing can imply specific methods. The duration of solving a contiguous
 * problem of applying block operations proportionally to arrive at a balanced
 * source and target. A thought of the merge input is that it is idealized as
 * a smaller than the main blob. But that is not the only condition. Utilizing
 * a new out buffer applies that a functioning algorithm. multiple types of
 * data can be ran through the parameters. The most effective being sets of
 * data. The ability to control the mixing parameters can also be important.
 *
 * @tparam T
 * @param merge
 */
template <> void container_t::mix(container_t &mn) {
    container_t out;

    size_t len = ds.size() + mn.ds.size();
    out.ds.reserve(len);

    bool bdone = false;
    size_t idx1 = 0;
    size_t idx2 = 0;
    size_t cnt = 0;

```

```

while (!bdone) {
    cnt = len / 5;
    while (--cnt && idx1 < ds.size()) {
        out.ds.push_back(ds[idx1]);
        idx1++;
    }

    cnt = len / 3;
    while (--cnt && idx2 < mn.ds.size()) {
        out.ds.push_back(mn.ds[idx2]);
        idx2++;
    }
}

if (idx1 > ds.size() && idx2 > mn.ds.size())
    bdone = true;
}

template <> void container_t::mix(bit_vector &n) {
    container_t mn;
    mn << n;
    mix(mn);
}

template <> void container_t::mix(nibble_vector &n) {
    container_t mn;
    mn << n;
    mix(mn);
}

/***
 * @brief The bit entry provides a byte aligned flush using the residue
 * buffer. Nibble insert simply creates four bits and calls this routine. This
 * allows entering of access types and sizes. Data should be aligned within a
 * byte boundary as segments of bit and nibble. Otherwise stream insert logic
 * is not allotted for other data types. Typical uses are the nibbles and bit
 * stream operations are not file based, simply encoding base. The system does
 * not support dynamic bit packing of a variable number of types currently.
 */
* @tparam
* @param bv
* @return container_t&
*/
template <> container_t &container_t::operator<<(const container_t &rdata) {
    ds.insert(ds.end(), rdata.ds.begin(), rdata.ds.end());
    residue_size = rdata.residue_size;
    residue_value = rdata.residue_value;
    return *this;
}

template <> container_t &container_t::operator<<(const std::string &s) {
    for (auto ch : s)
        operator<<(static_cast<uint8_t>(ch));

    return *this;
}

template <> container_t &container_t::operator<<(const directory_t &rd) {
    for (auto n : rd) {
        operator<<(n.hash);
        operator<<(n.original_size);
        operator<<(n.data_index);
        operator<<(n.szname);
    }
    return *this;
}

template <typename T> container_t &container_t::operator<<(const T &rn) {
    if constexpr (is_instance_of_v<T, std::vector>) {
        for (auto n : rn)
            operator<<(n);

    } else if constexpr (std::is_base_of_v<T, bit_vector>) {

```

```

const T &bv = rn;
uint8_t b = 0x00;
uint8_t ch = 0;

auto prime_bit = 0;

if (residue_size) {
    prime_bit = residue_size;
    ch = residue_value;
}

size_t whole_bits = (bv.size() / CHAR_BIT) * CHAR_BIT;
residue_size = bv.size() % CHAR_BIT;

// send data one byte at a time.
size_t i = 0;
size_t j = prime_bit;

bool bdone = false;
while (!bdone) {
    while (j < CHAR_BIT) {
        if (bv[i])
            ch |= 1 << (CHAR_BIT - 1) - j;
        j++;
        i++;
        if (i > bv.size()) {
            bdone = true;
            break;
        }
    }
    if (bdone) {
        residue_value = ch;
        residue_size = j;
        break;
    }
}

operator<<(ch);
j = 0;
ch = 0;
}

} else if constexpr (std::is_base_of_v<T, nibble_vector>) {
    const T &nv = rn;

    bit_vector bv;
    for (auto n : nv) {
        bv.push_back(n & 8);
        bv.push_back(n & 4);
        bv.push_back(n & 2);
        bv.push_back(n & 1);
    }
    operator<<(bv);
    return *this;
} else {
    union bts {
        T o;
        uint8_t b[sizeof(T)];
    };

    bts t;
    t.o = rn;

    for (auto c = 0; c < sizeof(T); c++)
        operator<<(t.b[c]);
}

return *this;
}

/**
 * takes information and scatters the unit
 */
template <typename T> void container_t::scatter(prng_t &prng, size_t index) {

```

```

size_t cycles = 0;
size_t ratio_low = 0;
size_t ratio_high = 0;
size_t amt = 0;

union bts {
    T n;
    uint8_t b[sizeof(T)];
};

bts t;
t.n = get<T>(index);

for (size_t c = 0; c < cycles; c++) {

    switch (prng.generate(0, 2)) {
        case 0:
            ratio_low = sizeof(T) / 5;
            ratio_high = sizeof(T) / 3;
            amt = prng.generate(ratio_low, ratio_high);

            // bytes
            for (auto i = 0; i < amt; i++) {
                int id1 = prng.generate(0, sizeof(T));
                int id2 = prng.generate(0, sizeof(T));
                std::swap(t.b[id1], t.b[id2]);
            }
            break;

        // nibbles
        case 1:
            ratio_low = sizeof(T) * 2 / 5;
            ratio_high = sizeof(T) * 2 / 3;
            amt = prng.generate(ratio_low, ratio_high);

            for (auto i = 0; i < amt; i++) {
                int id1 = prng.generate(0, sizeof(T));
                int id2 = prng.generate(0, sizeof(T));
                uint8_t mask1, mask2;

                if (prng.generate(0, 1))
                    mask1 = 0b11110000;
                else
                    mask1 = 0b00001111;

                if (prng.generate(0, 1))
                    mask2 = 0b11110000;
                else
                    mask2 = 0b00001111;

                uint8_t v1 = t.n[id1] & mask1;
                uint8_t v2 = t.n[id2] & mask2;

                // swapping same nibble positions
                if (mask1 == mask2) {
                    t.n[id2] = (~mask1 & t.n[id2]) | (mask1 | v1);
                    t.n[id1] = (~mask1 & t.n[id1]) | (mask1 | v2);

                    // swapping high to low
                } else if (mask1 == 0b11110000) {
                    t.n[id2] = (~mask1 & t.n[id2]) | (mask2 & v1 >> 4);
                    t.n[id1] = (~mask2 & t.n[id1]) | (mask1 & v2 << 4);

                } else {
                    t.n[id2] = (~mask1 & t.n[id2]) | (mask2 & v1 << 4);
                    t.n[id1] = (~mask2 & t.n[id1]) | (mask1 & v2 >> 4);
                }
            }
            break;

        // Bits
        case 2:
            ratio_low = sizeof(T) * CHAR_BIT / 5;
            ratio_high = sizeof(T) * CHAR_BIT / 3;
    }
}

```

```

amt = prng.generate(ratio_low, ratio_high);

for (auto i = 0; i < amt; i++) {
    int id1 = prng.generate(0, sizeof(T) * CHAR_BIT);
    int id2 = prng.generate(0, sizeof(T) * CHAR_BIT);
    uint8_t mask1, mask2;

    mask1 = 1 << (CHAR_BIT - 1) - id1 % CHAR_BIT;
    mask2 = 1 << (CHAR_BIT - 1) - id2 % CHAR_BIT;

    bool v1 = t.n[id1 / CHAR_BIT] & mask1;
    bool v2 = t.n[id2 / CHAR_BIT] & mask2;

    // swapping two bits
    t.n[id2 / CHAR_BIT] = (~mask2 & t.n[id2 / CHAR_BIT]) |
        (v1 == true ? 1 << id2 % CHAR_BIT : 0);
    t.n[id1 / CHAR_BIT] = (~mask1 & t.n[id1 / CHAR_BIT]) |
        (v2 == true ? 1 << id1 % CHAR_BIT : 0);
}
}
}

class operation_t {
public:
    class noise_inject : public invoke_t {
public:
    void apply(container_t &t);
};

    class scatter : public invoke_t {
public:
    void apply(container_t &t) {}
};

    class shuffle : public invoke_t {
public:
    void apply(container_t &t) {}
};

    class segment : public invoke_t {
public:
    void apply(container_t &t) {}
};

    class compress : public invoke_t {
public:
    void apply(container_t &t) {}
};

    class rotate : public invoke_t {
public:
    void apply(container_t &t) {}
};

    class start_lock : public invoke_t {
public:
    void apply(container_t &t) {}
};
    class end_lock : public invoke_t {
public:
    void apply(container_t &t) {}
};
    class directory : public invoke_t {
public:
    void apply(container_t &t) {}
};
};

};

};

}; // namespace genetic_cipher

```

shape.h

```
/*
 * shape.h
 *
 * Created on: Apr 17, 2024
 *      Author: anthonym
 */

#pragma once

#include "stdbase.h"
#include "random.h"

namespace genetic_cipher {

using namespace std;

/***
 * @brief
 *
 */
class shape {
public:
    class shapeb {
public:
    shapeb(prng_t &rp) : prng(rp){}
    prng_t &prng;

    size_t x = 0, y = 0, w = 0, h = 0, a = 0;
    void rotate(group_t &g, int64_t iamt) {

        if (iamt < 0) {
            // rotate left
            int rotL = iamt * -1;
            std::rotate(g.begin(), g.begin() + rotL, g.end());
        } else {
            // rotate right
            int rotR = iamt;
            std::rotate(g.begin(), g.begin() + g.size() - rotR, g.end());
        }
    }
};

class square : public shapeb {
public:
    group_t &info;

    square(prng_t &pr, vector<int64_t> &rinfo) : shapeb(pr), info(rinfo) {
        a = info.size();
        w = std::sqrt(a);
        h = w;
    }

    group_t get_row(int64_t irow) { return get_line(irow, 0, irow, w); }

    void set_row(int64_t irow, group_t &g) { set_line(irow, 0, irow, w, g); }

    group_t get_col(int64_t icol) { return get_line(0, icol, h, icol); }

    void set_col(int64_t icol, group_t &g) { set_line(0, icol, h, icol, g); }

    // https://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm
    group_t get_line(int64_t x0, int64_t y0, int64_t x1, int64_t y1) {
        int64_t dx = std::abs(x1 - x0), sx = x0 < x1 ? 1 : -1;
        int64_t dy = -std::abs(y1 - y0), sy = y0 < y1 ? 1 : -1;
        int64_t err = dx / 2, e2 = dx * dx, y = y0;
        for (int64_t x = x0; x != x1; x += sx) {
            g[x][y] = true;
            err -= dy;
            if (err <= 0) {
                y += sy;
                err += e2;
            }
        }
    }
};
```

```

int64_t err = dx + dy, e2;
/* error value e_xy */

group_t out;

for (;;) {
    int64_t n = x0 * w * y0;
    out.push_back(n);

    e2 = 2 * err;
    if (e2 >= dy) {
        if (x0 == x1)
            break;
        err += dy;
        x0 += sx;
    }
    if (e2 <= dx) {
        if (y0 == y1)
            break;
        err += dx;
        y0 += sy;
    }
}
return out;
}

void set_line(int64_t x0, int64_t y0, int64_t x1, int64_t y1, group_t &g) {
    size_t dx = abs(x1 - x0), sx = x0 < x1 ? 1 : -1;
    size_t dy = -abs(y1 - y0), sy = y0 < y1 ? 1 : -1;
    size_t err = dx + dy, e2;
    /* error value e_xy */

    size_t idx = 0;
    for (;;) {
        int64_t n = x0 * w * y0;
        info[n] = g[idx];
        idx++;

        e2 = 2 * err;
        if (e2 >= dy) {
            if (x0 == x1)
                break;
            err += dy;
            x0 += sx;
        }
        if (e2 <= dx) {
            if (y0 == y1)
                break;
            err += dx;
            y0 += sy;
        }
    }
}

// mix up the interior of the box
void scatter(void) {
    auto ratio_low = w / 5;
    auto ratio_high = w / 2;
    int64_t amt;
    auto cycles = prng.generate(3, 5);

    for (auto c1 = 0; c1 < cycles; c1++) {
        switch (prng.generate(0, 5)) {
        case 0: // swap rows
            amt = prng.generate(ratio_low, ratio_high);
            for (auto l = 0; l < amt; l++) {
                auto i1 = prng.generate(0, w);
                auto i2 = prng.generate(0, w);
                auto o1 = get_row(i1);
                auto o2 = get_row(i2);
                set_row(i1, o1);
                set_row(i2, o2);
            }
        }
    }
}

```

```

        break;
case 1: // swap cols
amt = prng.generate(ratio_low, ratio_high);
for (auto l = 0; l < amt; l++) {
    auto i1 = prng.generate(0, h);
    auto i2 = prng.generate(0, h);
    auto o1 = get_col(i1);
    auto o2 = get_col(i2);
    set_col(i1, o1);
    set_col(i2, o2);

}
break;
case 2: // rotate rows
amt = prng.generate(ratio_low, ratio_high);
for (auto l = 0; l < amt; l++) {
    auto i1 = prng.generate(0, h);
    auto o1 = get_row(i1);

    auto op = prng.generate(ratio_low, ratio_high);
    if (prng.generate(0, 1))
        op *= -1;

    rotate(o1, op);
    set_row(i1, o1);
}
break;
case 3: // rotate cols
amt = prng.generate(ratio_low, ratio_high);
for (auto l = 0; l < amt; l++) {
    auto i1 = prng.generate(0, w);
    auto o1 = get_col(i1);

    auto op = prng.generate(ratio_low, ratio_high);
    if (prng.generate(0, 1))
        op *= -1;

    rotate(o1, op);
    set_col(i1, o1);
}
break;

case 4: // lines rotate
amt = prng.generate(ratio_low, ratio_high);
for (auto l = 0; l < amt; l++) {
    auto i1 = prng.generate(0, w);
    auto i2 = prng.generate(0, h);
    auto i3 = prng.generate(0, w);
    auto i4 = prng.generate(0, h);

    auto o1 = get_line(i1, i2, i3, i4);
    auto op = prng.generate(ratio_low, ratio_high);
    if (prng.generate(0, 1))
        op *= -1;
    rotate(o1, op);
    set_line(i1, i2, i3, i4, o1);
}
break;
case 5: // lines rotate
amt = prng.generate(ratio_low, ratio_high);
for (auto l = 0; l < amt; l++) {
    for (auto i = 0; i < w / 2; i++) {
        auto o1 = get_line(0 + i, 0, w - i, h);
        auto op = prng.generate(ratio_low, ratio_high);
        if (prng.generate(0, 1))
            op *= -1;
        rotate(o1, op);
        set_line(0 + i, 0, w - i, h, o1);

        o1 = get_line(0, i, w, h - i);
        op = prng.generate(ratio_low, ratio_high);
        if (prng.generate(0, 1))
            op *= -1;
    }
}

```

```

        rotate(o1, op);
        set_line(0, i, w, h - i, o1);
    }
}
break;
}
}
};

class rectangle : public shapeb {};

class polygon : public shapeb {};

class hexagon : public shapeb {};

class octagon : public shapeb {};

class triangle : public shapeb {};

class circle : public shapeb {};
};

using shape_list_t = variant<shape::square, shape::rectangle, shape::polygon,
                           shape::octagon, shape::triangle, shape::circle>;

}; // namespace genetic_cipher

```

random.h

```

/*
 * random.h
 *
 * Created on: Apr 16, 2024
 *      Author: anthonym
 */

#pragma once

#include "stdbase.h"

namespace genetic_cipher {
using namespace std;

using group_t = vector<int64_t>;
using groups_t = vector<group_t>;
class distribution : public vector<int64_t>{};

class prng_t {
public:
    prng_t() {}

    /* with the behavior dependent on the prng engine,
    having the most secretive uses of the locked password,
    and also samples of data must change the behavior of the engine.
    This dampens the effect of having a statistical single nature
    into several long term or temporary modifications in behavior.
    The opportunity to use more data for seeding input provides
    this as these are vector oriented. During data encryption,
    the transformed data becomes modified. Using the seed multiply
    also is used to further unlock the random behavior. Of which
    trees are formed in the permutation objects. The order most
    presumed to operate in such capacities are likened to the
    workings of hash codes. */
    void seed_secret(const vector<uint8_t> &rp) {}
    void seed_multiply(std::vector<uint8_t> &v) {}
}

```

```

std::random_device rd;
std::mt19937 gen;

int64_t generate(int64_t low, int64_t high) {
    std::uniform_int_distribution<> rng(low, high);
    return rng(gen);
}
/***
 * The class maintains an index and tolerant work of array for
 * a progression.
***/

public:
    /* as values are working in the ratio set, the value of one side
     can be altered to plus or minus this amount randomly. In that,
     the projected ration values given are varied slightly along another
     profile within the prng. Yet for for large data sets, this value
     changing slightly alters the amount of data by percentage weight
     within a group. */
    const int jitter_max = 5;

groups_t generate_progression(int64_t rn, const distribution &profile) {
    groups_t out;
    out.reserve(profile.size());

    std::vector<int64_t> onums(rn);

    std::iota(onums.begin(), onums.end() + rn, 0);
    std::shuffle(onums.begin(), onums.end(), gen);

    // when the profile does not exactly add up to one hundred percent,
    // remove the differences now from random locations.
    int left_out = 100 - std::accumulate(profile.begin(), profile.end(), 0);
    if (left_out != 0)
        onums.erase(onums.begin(),
                    onums.begin() + (onums.size() * (left_out / 100)));

    bool bdone = false;
    auto idx = onums.begin();
    auto cnt = 0;

    int ratio_low = (profile[0] / 100) * rn;
    int ratio_high = (profile[1] / 100) * rn;

    while (!bdone) {
        auto skip_amt = generate(ratio_low, ratio_high);
        out[cnt].reserve(skip_amt);
        out[cnt].emplace_back(idx, idx + skip_amt);
        idx += skip_amt;

        if (idx != onums.end())
            bdone = true;
    }
    return out;
}

};

}; // namespace genetic_cipher

```

```
/*
 * std_cpp.h
 *
 * Created on: Apr 14, 2024
 * Author: anthonym
 */
#pragma once

#include <algorithm>
#include <any>
#include <array>
#include <cctype>
#include <climits>
#include <cmath>
#include <cstdarg>
#include <cstdint>
#include <cstdio>

#include <cstdlib>

#include <cstring>
#include <fstream>
#include <functional>
#include <future>
#include <iomanip>
#include <iostream>
#include <iterator>
#include <map>
#include <memory>
#include <optional>
#include <regex>
#include <sstream>
#include <stdexcept>
#include <string>
#include <string_view>
#include <tuple>
#include <type_traits>
#include <typeindex>
#include <typeinfo>
#include <unordered_map>
#include <utility>
#include <variant>
#include <vector>

#include <bits/stdc++.h>
#include <cmath>
#include <random>
```

Conclusion

未知、焉得仁。【二節】崔子弑齊君、陳文子有馬十乘、棄而違之。至於他邦、則曰、猶吾大夫崔子也、違之、之一邦、則又曰、猶吾大夫崔子也、違之、何如。子曰、清矣。曰、仁矣乎。曰、未知、焉得仁。
【十九章】季文子三思而後行。子聞之曰、再、斯可矣。
【二十章】子曰、甯武子、邦有道、則

2. Tsze-chang proceeded, 'When the officer Ch'ui killed the prince of Ch'i, Ch'an Wan, though he was the owner of forty horses, abandoned them and left the country. Coming to another State, he said, "They are here like our great officer, Ch'ui," and left it. He came to a second State, and with the same observation left it also;-- what do you say of him?' The Master replied, 'He was pure.' 'Was he perfectly virtuous?' 'I do not know. How can he be pronounced perfectly virtuous?'

CHAP. XIX. Chi Wan thought thrice, and then acted. When the Master was informed of it, he said, 'Twice may do.'

CHAP. XX. The Master said, 'When good order prevailed in his country, Ning Wu acted the part of a wise man. When his country was in disorder, he acted the part of a stupid man. Others may equal his wisdom, but they cannot equal his stupidity.'

<https://sacred-texts.com/cfu/cfu.htm>
CONFUCIAN ANALECTS

As we learn to develop better methods, it is always to be entailed a dynamic capability of expansion. The C++ language to be harnessed in the future, is about security model concepts. Classification of program and base allowed. The capability for the abstraction of language to better be observed in planning is an important aspect of corporate planning. That is, if a trail of information exists such as taughting in legitimized technology namespace as cross reference argumentative points of security, many regulations of plans have to be managed. As a cycle of managing a government body, the computer and its intrinsics must be established to be frontal and not the reward of hacking. The largest yeail of humanity is the direction digit to thinking. The system here shows a genetic approach to creating an adaptable and self configured encryption per application requirement in technology. The placement of the interwoven implementation has to be strengthened to support industry production concepts. Logic in parallel processing and associated weight trees will almost always date with microprocessor speed, or number of processors. In summary, encryption so likened to holiness in that reflecting perfection of hiding the secret within a nest of words, meaning, and depth, its canonization is seen as a useful tool, propelling the medium of information security and truthful knowledge. God has even given and placed HASH codes in previous books.

The basis of a well formed and dynamic buffer communication system also entails that the instantiation of the template library handle communications. The browser code left to a smaller implementation for its landmine capacity of visualization, JIT, storage while connectivity and stream service is handled by the encryption and adaptive symbol dictionary compression algorithms. The Nebula OS is a system design that utilizes the C++ template language for syntax of the document object object in binary form. As a facility, the client and server are fashioned for a communicative sgml format, such as the HTML provides. The nibble dictionary format, or variable bit format, provides a narrow use of space and effective compression. As a system design, the encryption gives a new focus and freedom to market innovation. Imagine just a smart television with email system, or video phone capability. The format publication can very well change as a result. Or a device that is only a type writer for textual information input. As device, the entry of information is held as a distributed resource. A writing tool, or a software source code editor. An education only rugged computer. In summary, encryption and the template design of C++ lend it self to robust platform delivery of embedded devices, and personal computer, and computing clusters.

GNU License

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <<https://fsf.org/>>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of work.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to

your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated

conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no

permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means,

then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General

Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program. If not, see <<https://www.gnu.org/licenses/>>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

```
<program> Copyright (C) <year> <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License. Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<<https://www.gnu.org/licenses/>>.

The GNU General Public License does not permit incorporating your program
into proprietary programs. If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library. If this is what you want to do, use the GNU Lesser General
Public License instead of this License. But first, please read
<<https://www.gnu.org/licenses/why-not-lgpl.html>>.