**Ngee Ann Polytechnic**

**School of ICT**

**CTG Assignment Outline**

**Crypt****ographic Algorithms**

**Han Xihe, Ho Quang Minh, Dominic Lee, Neo Jia Chen**

**CSF02 Team Cotton**

**Introduction**

The rise of the internet, comes the rise of many online businesses and users which results in an enormous amount of data being transferred across the internet. Some of these data are confidential such as passwords and credit card information and must be protected from hackers. This is when cryptographic algorithms come into play.

There are many cryptographic algorithms currently being used and they all aim to fulfil P.A.I.N as much as possible. When P.A.I.N (privacy, authenticity, integrity, non-repudiation) is ensured, all aspects of cryptography are we can be confident that data is secured.

In this report, we will be examining 3 cryptographic algorithms that we think are worth learning about. They are Blowfish, Scrypt ,and RC5 and we will be examining their notable features, how they work, their intended purposes, their strengths and weaknesses, and any relevant information that will help you have a better understanding of these algorithms.

**Scrypt**

**Features/Characteristics**

Scrypt is a memory intensive algorithm that makes it costly to perform large scale attacks. As the memory in Scrypt is accessed in strong dependent order at each step, the memory access is the algorithm’s bottleneck.

**Detail Operation:**

Parameters:

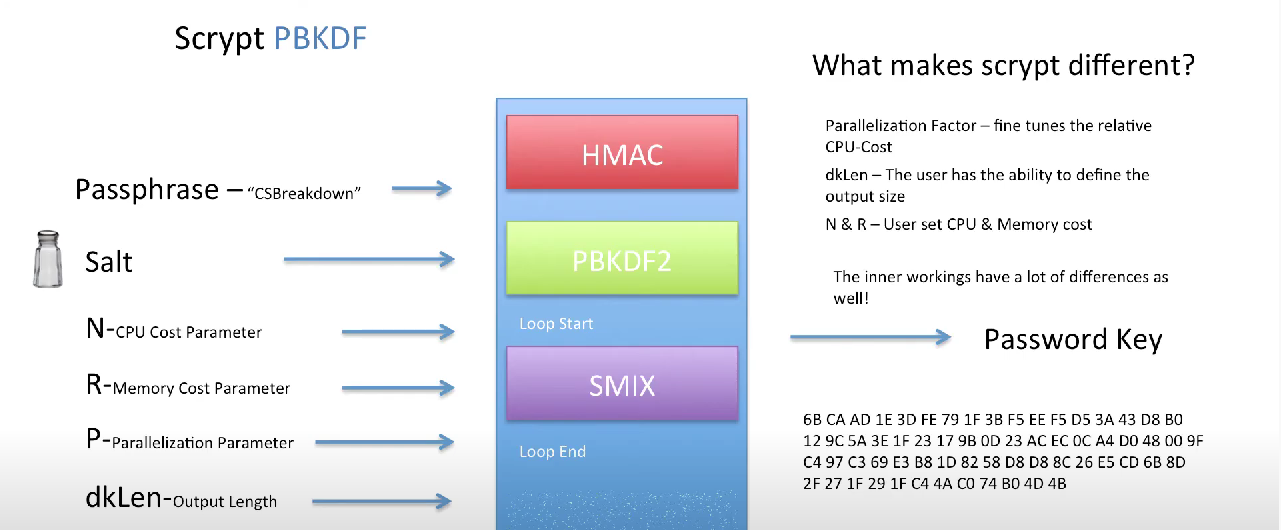
* N = iterations count
* r = block size
* P = parallelism factor
* Password = The input password
* Salt = securely-generated random bytes
* Derived-key-length = how many bytes to generate as output

Memory required to compute scrypt key = 128 \* N \* r \* p bytes

To make things clear, parallelism factor is responsible for fine-tuning the relative CPU-Cost, the derived key length allows the user to derive the output size, N sets the CPU cost, and R sets the memory cost.

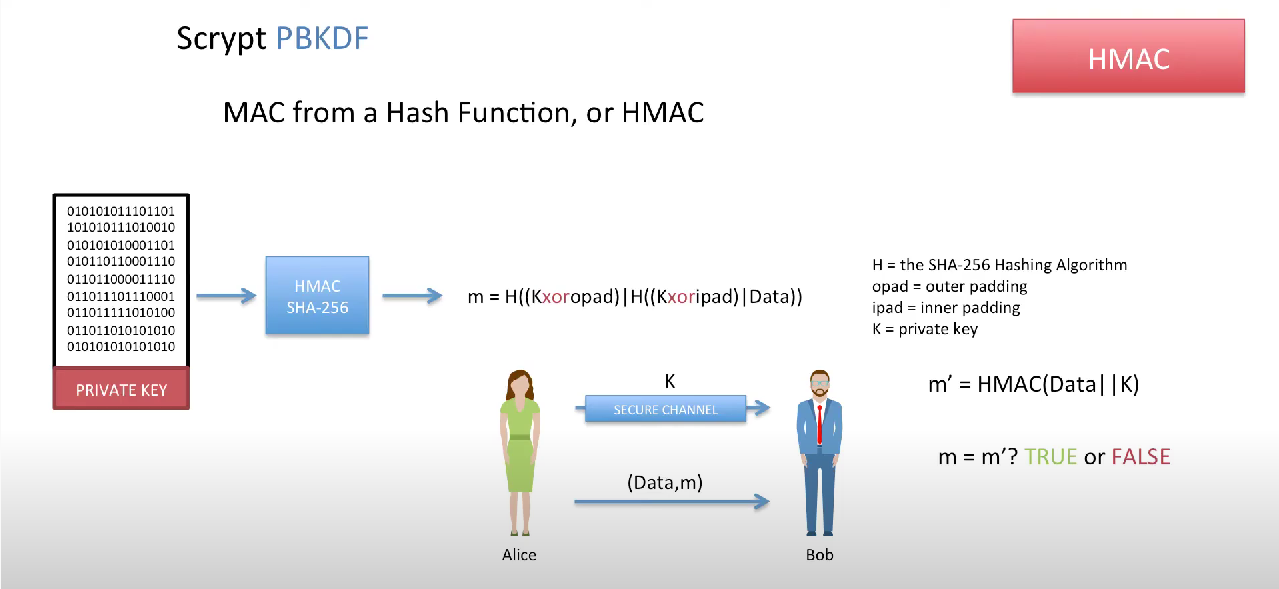
The overview of Scrypt

1. It will use the user input to create a HMAC
2. It then runs PBKDF2 function on the hashed value
3. Running a loop, for P iterations, it will apply a memory hard function called SMIX
4. It will then do the PBKDF2 function again on the value that is outputted when the loop ends.



Step 1: HMAC

The HMAC object is created with its key and data both being null. The HMAC object is initialized with a key that is hashed from the input password. The data’s value is still null and will only be added during the PBKDF2 function.



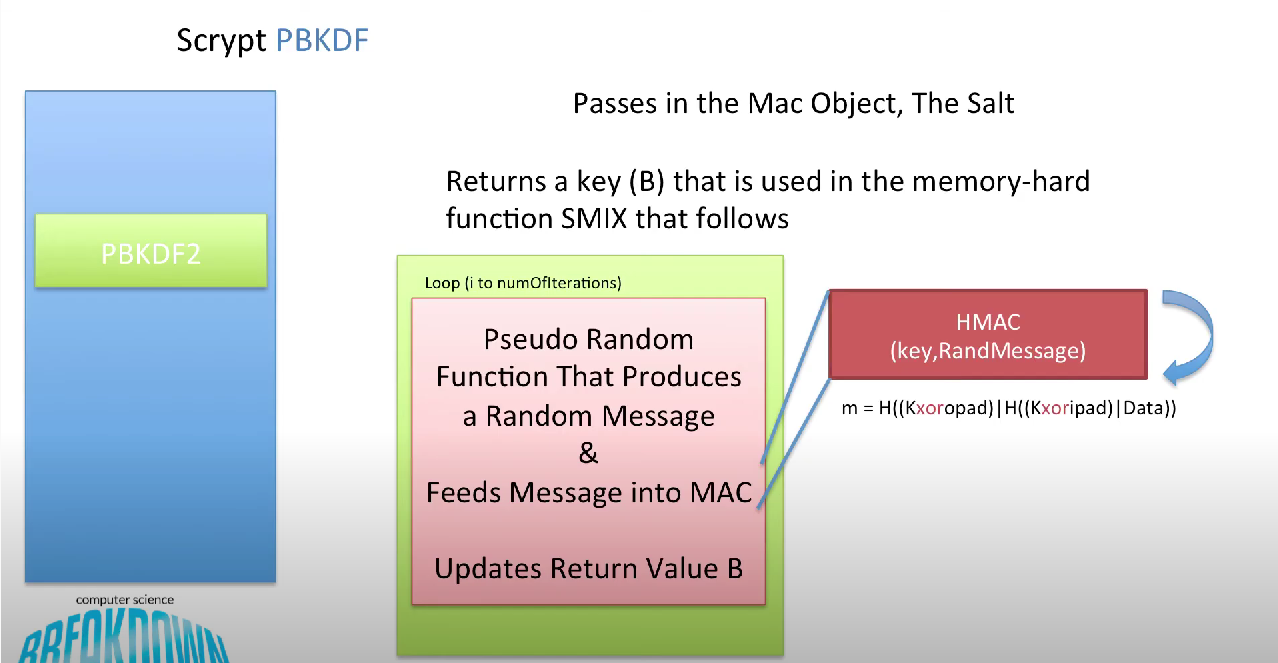
Note HMAC(key,data)

Step 2 and 4: PBKDF2

The PBKDF2 passes in the Mac object and the salt and returns a key (K) that will be used by the SMIX in the next step.

PBKDF2 is a pseudo-random function that produces a random message and inputs the message into the HMAC function that has its key as the hash of the user password HMAC(h(password),RandMsg). This will be done for several iterations that is defined by various variables such as the MAC length. Each iteration will update the final value of the return value K.

Step 4: For the final PBKDF2 in Scrypt, K’ which is computed from the SMIX algorithm is used as the salt which will output the final password key.



Step 3: SMIX

The number of iterations of SMIX is defined by the parallelization factor, P.

The SMIX take in the input K which was the key produced by the PBKDF2 function earlier. It then sets X = K and creates two arrays Y[ ] and Z[ ].

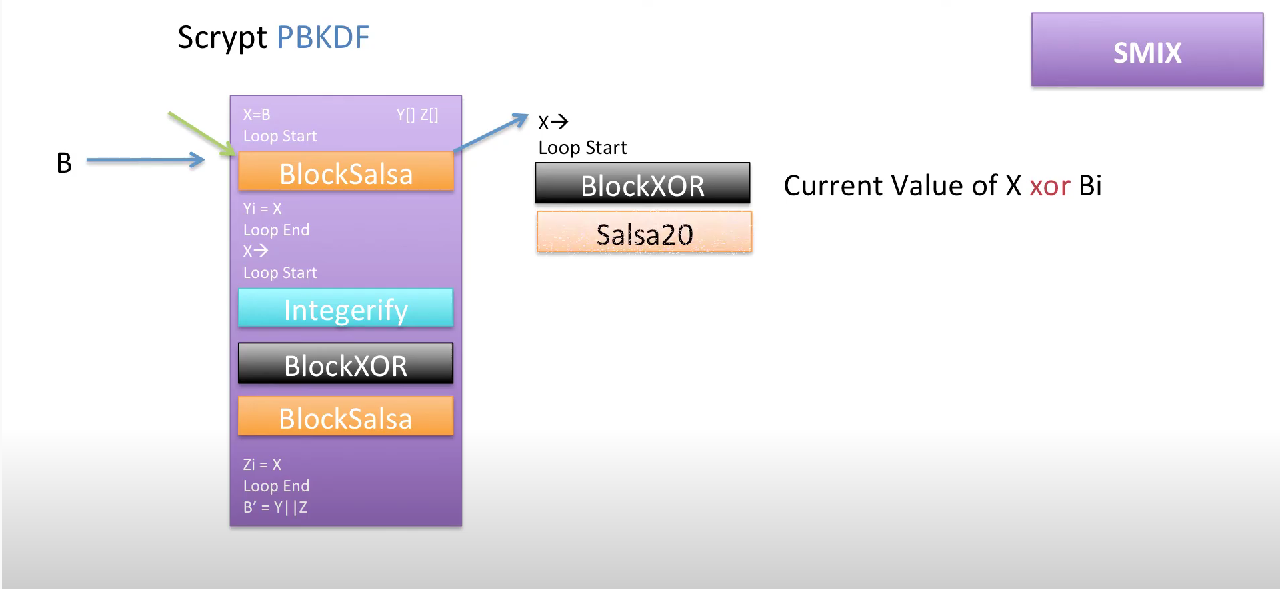
SMIX will first perform Block Salsa which takes in the value X and begin its own loops. In this loop, BlockXOR is performed on X which also means the current value of X xor Ki where i is the current iteration of the loop that it is in.

It then uses salsa 20 to hash the value X and appends this hashed value into array Y where X = Yi where i is the current iteration of the loop. The loop repeats until the iteration limit before going back to the SMIX algorithm.

X is then parsed into a new loop using Integerify which takes the current value of X and converts it into an integer value. This integer value of X is then parsed into another BlockXOR which xors the current value of X with Ki.

It then performs Block Salsa again where this time X is appended to Z each iteration where X = Zi until the loop ends.

A variable K’ is then created by appending array Y to array Z where the evens come before the odds.



**Purposes:**

Scrpyt was created by Colin Percival orgiannly used for the Tarsnap online backup service. Published in 2012 by the Internet Engineering Task Force in 2012.

Currently, scrypt is currently used in may cryptocurrencies as a proof of work algorithm. Notable companies include Dogecoin and Litecoin.

**Advantages**

1. Unlike PBKDF2 which has a glaring weakness because it uses a small amount of RAM, Scrypt which is an updated version of Bcrypt, takes a lot of CPU time and RAM memory.

This makes it more secure against GPU (parallel password cracking using video cards) and ASIC (specialized password cracking software) attack. It also does what PBDKF2 does well which is protecting against brute- force, rainbow, and dictionary attacks.

1. Lastly, it is a simple, efficient, and secure algorithm requiring little resources compared to its counterpart SHA-256.

**Disadvantages**

1. Argon2 is considered as a more secure modern key derivation function
2. The fact that it takes up a lot of CPU time and RAM memory also makes it disadvantages in certain situations. For example, scrypt is not used on larger files as it takes too long.

**RC5 (Ron Cipher 5)**

**Features/Characteristics**

RC5, short for Ron Cipher 5, is a word-oriented algorithm that is fast, has a variable key-length, and is a low-memory symmetric block cipher. It has a focus on speed, simplicity and compatibility across a wide range of computing hardware. It features variable rounds and key sizes that can be increased according to computation power that is available for increased security.

**Detail operations**

RC5 has three stages: key expansion, encryption and decryption.

Parameters:

* w = Word size: consists of a two-word input and two-word output, where both are 16-bits or higher
* r = number of rounds.
* B = length of key K bytes
* Pw = magic constant Odd((e – 2) \* 2 ^ w), with Odd here being the closest odd integer to (e – 2) \* 2 ^ w
* Qw = magic constant Odd((ϕ - 1) \* 2 ^ w), with Odd here being the closest odd integer to (ϕ - 1) \* 2 ^ w

Parameters B and r are directly proportionate to security as the higher/longer they are, the higher the security.

But the higher r is, the lower the speed of the algorithm

The nominal parameters for RC5 are, a 32-bit word, 16 round, and 7-byte key length.

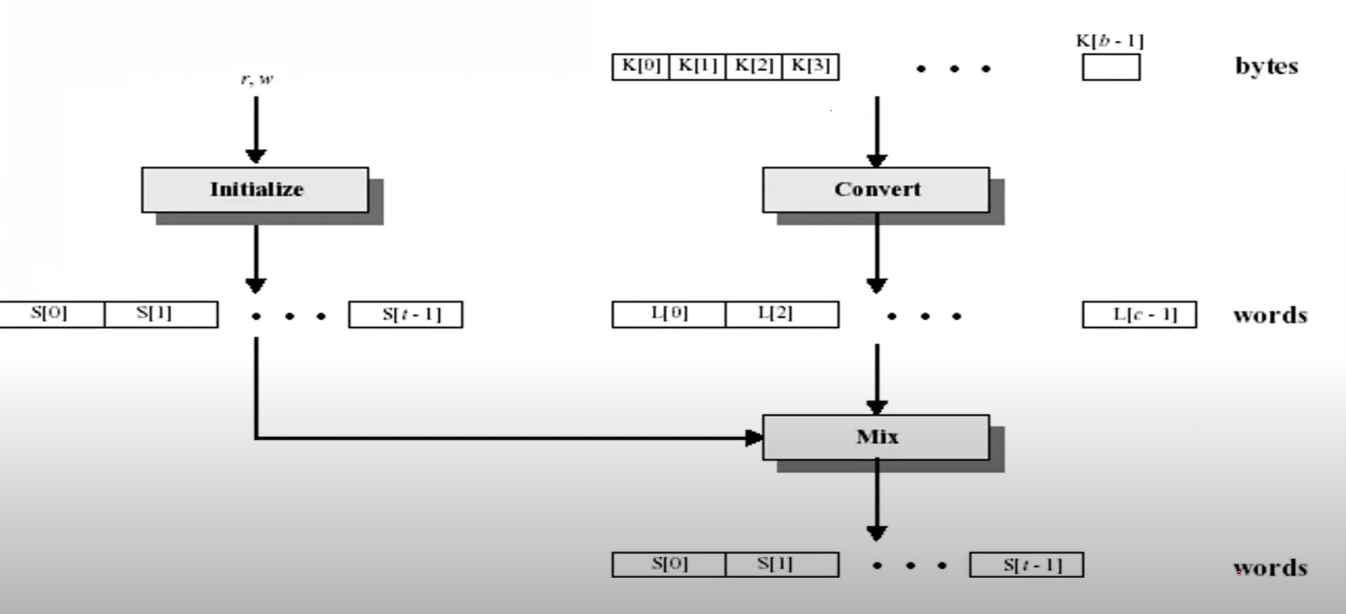
Step 1: Key expansion:

Key expansion is the step before encryption and decryption to obtain the expanded key S for future use in the encryption and decryption algorithms. The secret key will be processed through bitwise XOR, left circular shift and addition to produce t = 2r + 2 subkeys, where each subkey is w bits long.

In the algorithm, the array of secret key K is K [0], K [1], K [2]... K [B - 1] is converted into L [0], L [1], L [2] ... L [C – 1]. To convert the arrays, we first calculate one-eighth of w, which we will call u. Next, we calculate the length of array L by comparing values B and 1, and then use the larger value and divide it by u. We then round the results of the operation to the nearest integer, the obtained result known as C. Array L will therefore be C in length. To calculate array L, we loop with B iterations from B-1 to 0. L[i / u] is then equivalent to key array K bytes K[i], added to the left circular shift rotation of L[i] and 8, where (L[i / u] <<< 8) + K[i]. This provides all values in array L from L[0] to L[C – 1]

In a separate operation, the subkey array S, S[0], S[1], S[2], ... S[t – 1] is initialised. To calculate S[i], take the previous index i of S ( or S[0] for the first iteration) and minus the index i by 1 to find the new index. Take the new index of S and add it to magic constant Qw, where S[i] will now be equal to S[i – 1] + Qw. This operation is iterated t – 2 times, from 1 to t – 1, starting from S[0] being equal to magic constant Pw. This iteration provides all values in subkey array S.

In the following operation, the word array L is mixed with subkey array S, where i = j = X = Y = 0, taking the larger of t or C and multiplying the value by 3, the obtained result would be the number of iterations, i of the mixing operation. In the mixing operation, X is equal to S[i], which is also equivalent to the left circular rotation of S[i] + X + Y to obtain a value which we will call G. A left circular shift of G and 3 is then performed, where G <<< 3 = X. In the next operation, a similar calculation is performed, where Y is equal to L[i], equivalent to L[j] + X + Y to obtain a value which we will call P. A left circular shift of P and the results of X + Y is performed, where P <<< (X + Y) = Y. Afterwards, the iterative values i and j have a modulus performed on them to “prepare” them for the next iteration of key expansion, where i is equivalent to i + 1 modulus t and j is equivalent to j + 1 modulus c. These operations produces the final expanded key array S, S[0], S[1], S[2] to S[t - 1]. This expanded key array S is then passed onto the encryption/decryption algorithms.



Step 2: Encryption

After the key expansion, the next step for RC5 is the encryption of the data which uses various primitive operations to do so. They are bitwise XOR, left/ right circular rotation, and addition/ subtraction.

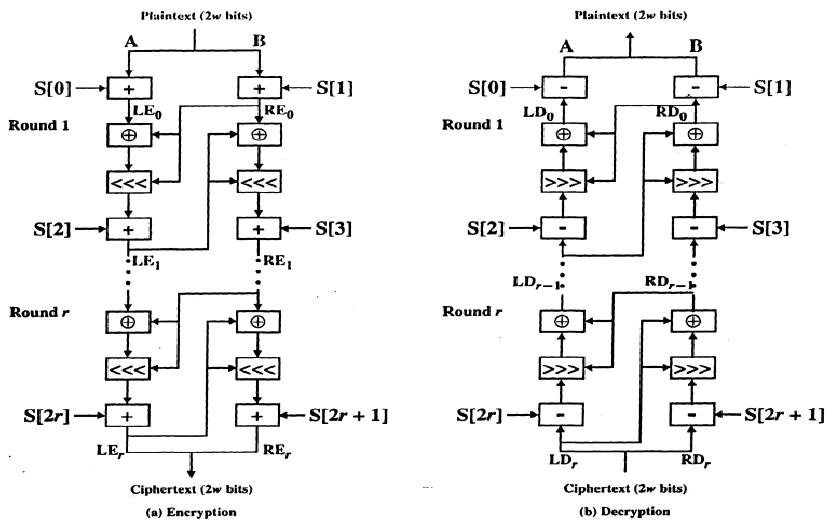
The plaintext resides in two w-bit registers as A and B. LE refers to the left half of the data and RE refers to the right half of the data after the round has been completed. There will be r rounds occurring during encryption and in each round, there will be a substitution using both words of data, a permutation using both words of data, and a substitution that depends on the key.

What really happens in each round?

Assuming this is round 1, LE = S [0] + A and RE = S [1] + B. LE is then XORed with RE to get a value which we will call K. A left circular rotation is then performed on K and RE, where K <<< RE, and this value is then added to S [2] which forms LE1, the left half of the data of the first round. LE1 is then XORed with RE to form a result which we will call L. Similarly, a left circular rotation is then carried out on L and LE1, where L <<< LE1 and the result is added to S [3] to form RE1, the right half of the data after round 1. This process is repeated for r rounds.

The final cipher text is contained in the two variables LEr and REr which is of length = 2w.

Note: As both halves of data are updated in 1 round, 1 round of RC5 = 2 rounds of DES.



Step 3: Decryption

Decryption in RC5 is extremely similar to its encryption except that some of the operations are reversed. Like encryption, there are two w-bits of ciphertext assigned to one-word variables LDr and RDr and r round in total.

What happens in each round?

Unlike the encryption where it starts with LE and RE, decryption starts with LDr and RDr. S [2r+1] is subtracted from RDr and right circular rotation is then performed on the LDr and this result to obtain a result which we will name M. M is the XORed with LDr to obtain RDr-1. Following, S [2r] is then subtracted from LDr and the obtained value then combined with RDr-1 goes through a right circular rotation and this value N, is then XORed with RDr-1 to obtain LDr-1.

This will be done for r round to obtain the plain text which will also be 2w bits which will be contained in LD and RD.

**Purposes**

RC5 was designed by Ronald Rivet in 1994 as a successor to RC4 with a focus on simplicity. One design point around RC5 was to encourage the study and cryptoanalysis of simplistic, low-level algorithms targeted for use in computing applications.

**Advantages**

**Fast**

* RC5 is a fast algorithm as it only uses primitive computer operations which takes little computing power.

**Not memory intensive**

* Due to the natures of its operations, RC5 is not memory intensive and can be used in systems with severe memory shortages such as checkout kiosks.

**Simple**

* It is known for its simplicity as a symmetric block cipher
* The encryption and decryption routines can be written in a few lines of code

**Secure**

* RC5 is known to be a very secure algorithm which is extremely difficult to break.
* A reason it is so secure is that its key expansion is a one-way function as it uses the golden ratio in Qw and logarithmic e in Pw which makes it impossible to reverse the key expansion process.
* Also, using powerful distributed computing systems or parallel computing systems will be too time-costly and thus unfeasible if the parameters were high.
* No obvious way RC5 is weak other than the key length being too short or number of rounds is set too low

**Disadvantages**

**Security Trade-offs**

* A high number of rounds are needed for increased security in the applications of RC5. 18 or more rounds are recommended for sufficient protection against attacks.
* However, a higher number of rounds would make the algorithm slower in less powerful computing platforms, while using a lower number of rounds will make it more susceptible to attacks, such as differential attacks using 12 rounds only.
* This means that for increased security, there would be the trade-off of having decreased speed on less powerful computing platforms, and vice versa.

**Implementation Limits**

* If the length of the key is too short and there are too little rounds, it would be possible to break it.
* For example, using 64-bit blocks and 12 rounds, RC5 can be broken with a 2^44 chosen plaintexts via a differential attack.

**Blowfish**

**Features/Characteristics**

Blowfish is a 64-bit block size symmetric-key block cipher. It uses a variable key length ranging from 32 bits to 448 bits with an S-box design, featuring a 16-round Feistel structure. It was designed as a replacement for DES (Data Encryption Standard) for general use.

**Detail Operations**

Blowfish is a symmetric algorithm. Thus, the same secret key has to be shared between the sender and the receiver.

**Encryption/ Decryption**

Step 1: Generate Subkeys

there are 18 subkeys that are needed in both the encryption and decryption process, and the same subkeys are used for both.

These 18 subkeys are then stored in a P-array, each array element is a 32-bit entry, and is initialized with the digits of pi.

Step 2: initialize substitution boxes

There are 4 substitution boxes that are needed in both encryption as well as the decryption process with each s-box having 256 entries where each entry is 32-bits.

Step 3: Encryption

The encryption consists of 16 rounds with each taking inputs from the plaintext from the previous round and corresponding subkeys.

**Purposes of Blowfish:**

Blowfish was designed as a generic replacement for DES to alleviate some of DES’s flaws, such as slower computation and small keys that are vulnerable to brute-forcing. Because of such characteristics, it is often used in computing applications with software. For example, OpenBSD uses a password-hashing algorithm that incorporates blowfish. Its low memory footprint makes also makes it compatible with low-end computing hardware, where can provide decently fast encryption rates. It is popular in embedded systems and also on hardware where increased rounds of computationally difficult encryption is not an issue.

**Advantages of Blowfish:**

Blowfish’s inherent nature makes it a fast block cipher algorithm, as it takes lesser operations to complete compared to other cryptographic algorithms. Increasing the rounds will make encryption take a longer time, but increase resistance against brute forcing. It’s public domain release also mean that anyone can use Blowfish in different applications.

**Disadvantages of Blowfish**

Each key has to be pre-process, causing encryption to take a longer time to complete, assuming that 16 rounds equivalent to 4 kilobytes of text is used. This prevents blowfish from being used in certain time-sensitive applications, although this is not a problem for certain applications such as SplashID. Thus, Schneier has recommended TwoFish, the successor of blowfish, over blowfish.

**References and image sources:**

Blowfish resources:

<https://www.ukessays.com/essays/computer-science/blowfish-algorithm-history-and-background-computer-science-essay.php>

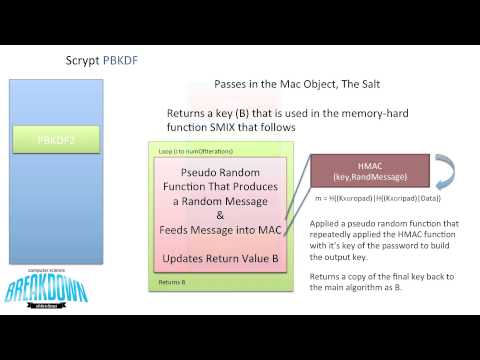
<https://www.encryptionconsulting.com/education-center/what-is-blowfish/>

<https://www.embedded.com/encrypting-data-with-the-blowfish-algorithm/>

Scrypt resources:

<https://www.linkedin.com/pulse/hashing-algorithms-sha256-vs-scrypt-kalana-wijenayake/>

[Scrypt Password Based Key Derivation Function - Cryptography](https://www.youtube.com/watch?v=TkWAgeSYL_Q&t=1161s)



RC5 resources:

<https://people.csail.mit.edu/rivest/Rivest-rc5rev.pdf>

<https://www.youtube.com/watch?v=BKQoRBezanY&t=1s>