


I'm not robot  reCAPTCHA

Continue

Example of symmetric key cryptography

Data Encryption Standard (DES) DES system diagram [2] DES is a symmetric system that was once a predominant standard in the 1970s but has since fallen out of favor due to its low security. Its introduction sparked heated debate about the role of standards in cryptography and led to much research and innovation in the field. However, DES is the archetype of block cipher systems, many systems today are based on its design. DES uses block ciphers. The block ciphers in DES consist of 56 random bits, and 8 more bits are used for error detection. These error detecting bits make DES unmalloable - attackers can't change the cipher on its way to its destination because they might accidentally delete a bit used for error detection, and then receivers would know the data had been attacked. However, the relatively small key size was an issue of debate even in the 1970s. By 1999, DES could be broken in under a day. This was later solved by sequencing multiple DES systems together, called 3DES. The data is first sent into the system and then cut into two 32-bit halves. Those two halves are sent through the entire system, criss-crossing using what's known as the Feistel system. There are 16 layers in DES, and at each layer, one half of the data goes through the Fiestel function. Once it's finished, it is XOR'd with the other half of the data. Each layer has its own subkey. The subkey is derived from the main, 56-bit key using a key scheduler. The Fiestel function, which occurs in every block labeled FFF in the diagram to the right, has 3 steps: Expansion: The incoming 32-bit block has half of its bits duplicated, making it a 48-bit block. Mixing: The new, 48-bit input block is put through an XOR gate with this round's unique subkey. Substitution: The mixed, 48-bit block is divided into 8 6-bit pieces. Each of these 8 pieces is put through an S-block which will output only 4-bits using a non-linear-transformation. * Permutation: The 32 output bits are then arranged in a specific permutation that ensures that they will be distributed among different S-blocks in the next round. *This is most important part of security in DES, it helps to avoid simple, algebra-based attacks. The key scheduler: The 56-bit primary key is split into two 28-bit keys. These halves are hereafter treated separately. In each round, each half is rotated left or right by either one or two bits (depending on the round). 24 bits from the left half are chosen, and 24 from the right are chosen to make a 48-bit subkey. Because we rotate on each round, each bit is only used in approximately 14 out of the 16 rounds. The key scheduler for encryption and decryption are the exact same except that the subkeys are in reverse order for decryption. DES was vulnerable to brute force attacks as early as the 1970s, but there were other ways it was weak as well. Differential cryptanalysis, or the study of how changes in inputs can affect output, are very effective at breaking block ciphers and DES in particular. Linear cryptanalysis, which apply affine transformations to a cipher were also widely used. Advanced Encryption Standard (AES) AES is similar to DES in that it is symmetric and uses block ciphers. However, it is much more secure than DES and has become the international standard. It is at least 6 times faster than 3DES. Instead of Fiestel functions, AES uses a substitution-permutation network. This network is a series of operations than either replaces input with output bits (substitution) or shuffles the bits (permutation). It uses 128-bit input plaintext, but it operates on bytes rather than bits. So, the input is represented as 16 bytes (because 128 bits = 16 bytes), arranged in a 4 x 4 matrix. This matrix, called the state, will be modified as the algorithm progresses. AES also operates in rounds, but the number of rounds is variable and is based on the length of the key used. A 128-bit key will run AES for 10 rounds, a 192-bit key for 12 rounds, and a 256-bit key will run for 14 rounds. Similar to DES, each round uses a different key. These subkeys are 128-bits in length and are calculated from the original key. AES proceeds as follows: Round 1 a. AddRoundKey Rounds 2 through (n-1) a. SubBytes b. ShiftRows c. MixColumns d. AddRoundKey Round n a. SubBytes b. ShiftRows c. AddRoundKey The first function, AddRoundKey, takes the current state (a 16-byte matrix) and XORs it with the key for this particular round. The result is the new state. SubBytes is one of the substitution functions of AES. The 16 byte state matrix is substituted using a S-box from the design of the specific AES implementation. This step is very similar to the substitution step in DES in that it uses non-linearity and an affine transformation to provide security to the system. ShiftRows shifts the bytes in each row with respect to each other. Typically, the top row of the state will remain unchanged, the second row will shift left one, the third row will shift left two and the fourth row will shift left 3. This step is done to ensure the columns are not linearly independent, which would turn AES into 4, independent block ciphers. MixColumns multiplies each column of the state by an invertible function, a fixed polynomial. Decryption in AES is the same algorithm as encryption, but in a reverse manner. Decryption, unlike in Fiestel's structure, needs to be implemented separately because the functions are in reverse order, but they are very similar. Blowfish Blowfish is another symmetric, block cipher. It was created after DES but before AES. Its block size is 64 bits, and it can use key lengths of anywhere from 32 up to 448 bits. Like DES, it is a 16-round Fiestel cipher. It's S-boxes, unlike in DES, are key-dependent, so they are generated dynamically. There are 5 subkey arrays in Blowfish, 1 18-entry P-array and 4 256-entry S-boxes. These subkey arrays store the subkeys used in every round of Blowfish. One P-array subkey is used in each round, and the remaining two entries are XOR'd with the final output after the last round. The S-boxes accept 8-bit input and produce 32-bit output. They are initialized using values from the hexadecimal digits of pi. Blowfish's key scheduler then takes the secret key and XORs it with all the P entries in order. Then, a 64-bit all zero block is encrypted with this algorithm and result replaces the first two entries in P. This process continues, with new subkeys, and the result replaces the following two entries in P, and so on. The entire P array (9 pairs of entries) and each of the 4 S-boxes (128 pairs each) will be replaced in this process. A total of 521 runs are needed to generate all the subkeys. Each of the 16 rounds of Blowfish has 4 operations. In round rrr: XOR the left half of the data with the rrrth entry in P. Use the output from (1) as the input for Blowfish's F-function.*star*. XOR the output from the F-function with the right half of the data. Swap the left half and the right half of the data.*star*. *star*.The F-function breaks up the 32-bit input into 4 segments and feeds them to the 4 S-boxes, resulting in 4 32-bit outputs. These are added together modulo 2322^2^32232, and XOR'd to produce a 32-bit output. *star*. *star*.After the very last round, undo this swap. Instead, XOR the left half of the data with the second to last entry in P, and XOR the right half of the data with the last entry in P. The following example is how the first round Blowfish would proceed if every subkey was 8 bits instead of 32. For simplicity, the inner working of the S-boxes are not described, and their output is just a given. P-array = [11010110, ...] Input Data = 0110011011010010 XOR the left half of the data with the first entry in the P-array. 0110011011010110=1011000001100110 voplus 11010110 = 10110000011001101011010110=10110000 Use the output from step one as the input to the F-function. The 8-bit result is split into 4, 2-bit sections and fed into the 4 S-boxes. The result is 4 8-bit outputs: 11010110, 11001010, 00010101, 01101010. These are added together and XOR'd in the following way: 11010110+11001010mod 28=10100000 (adding outputs 1 and 2)},11010110+11001010mod 2^8 = 10100000 vextxtrm{ (adding outputs 1 and 2)},11010110+11001010mod 28=10100000 (adding outputs 1 and 2), 1010000000010101=10110101 (XORing the addition of 1 and 2 with output 3),1010000000010101=10110101 (XORing the addition of 1 and 2 with output 3),10110101+01101010mod 28=00011111 (adding the result of the previous equation with output 4),10110101 + 01101010 vmod 2^8 = 00011111 vextxtrm{ (adding the result of the previous equation with output 4)},10110101+01101010mod 28=00011111 (adding the result of the previous equation with output 4). XOR the output from the F-function with the right half of the data. 000111111011010101=11001101 (This is the new "left" half of the data).00011111 voplus 110101010 = 11001101 vextxtrm{ (This is the new "left" half of the data)}.00011111101101010=11001101 (This is the new "left" half of the data). Swap the left half and the right half of the data. left: 11001101 right: 11010010 Now the data, after one round of Blowfish, is 1101001011001101. Decryption is the exact same as encryption (similar to DES), except for one thing. Instead of progressing through P from its first entry to its last, decryption will progress through P from its last entry to its first. Twofish Twofish is the successor to Blowfish and is similar in many ways. It almost beat AES to be the global standard. It is slower than AES when using 128-bit key sizes but slightly faster when using 256-bit key sizes. Twofish and Blowfish have a higher space complexit, meaning they take up more computer memory, than AES, making them less attractive. Due to its increased key sizes, Twofish is an improvement in security over Blowfish. It uses block sizes of 128 bits and keys sizes of up to 256 bits. It, like DES and Blowfish, uses a Fiestel structure when encrypting and decrypting its messages and ciphers. It also uses pre-computed S-boxes, splitting it private key into two sections and using one to modify the encryption algorithm. When it comes to the word 'Encryption,' we think of it as a technique that protects data using a cryptographic key, and there's nothing wrong with this. However, what most people don't realize is that there are different types of encryption methods. Asymmetric Encryption, also known as Public-Key Cryptography, is an example of one type.Unlike "normal" (symmetric) encryption, Asymmetric Encryption encrypts and decrypts the data using two separate yet mathematically connected cryptographic keys. These keys are known as a 'Public Key' and a 'Private Key'. Together, they're called a 'Public and Private Key Pair.' Let's see how these two keys work together to create the formidable force that is Asymmetric Encryption.How does Asymmetric Encryption work?Asymmetric Encryption uses two distinct, yet related keys. One key, the Public Key, is used for encryption and the other, the Private Key, is for decryption. As implied in the name, the Private Key is intended to be private so that only the authenticated recipient can decrypt the message.Let's understand this with a simple asymmetric encryption example.Pretend you're a spy agency and you need to devise a mechanism for your agents to report in securely. You don't need two-way communication, they have their orders, you just need regular detailed reports coming in from them. Asymmetric encryption would allow you to create public keys for the agents to encrypt their information and a private key back at headquarters that is the only way to decrypt it all. This provides an impenetrable form of one-way communication.How are the two keys generated?At the heart of Asymmetric Encryption lies a cryptographic algorithm. This algorithm uses a key generation protocol (a kind of mathematical function) to generate a key pair. Both the keys are mathematically connected with each other. This relationship between the keys differs from one algorithm to another.The algorithm is basically a combination of two functions - encryption function and decryption function. To state the obvious, the encryption function encrypts the data and decryption function decrypts it.This is how Asymmetric Encryption is used in SSL/TLS certificatesIn SSL/TLS and other digital certificates, both methods - Symmetric and Asymmetric - are employed. Now, you might be wondering, 'Why both? Shouldn't Asymmetric cryptography be used as it's more secure?' Granted, it is more secure, but it comes with a pitfall. A major drawback when it comes to Public Key Cryptography is the computational time. As the verification and functions are applied from both the sides, it slows down the process significantly. That's where Symmetric Encryption comes and saves the day.First, when two parties (browser and server in the case of SSL) come across each other, they validate each other's private and public key through Asymmetric Encryption. Once the verification is successful and both know whom they're talking to, the encryption of the data starts - through Symmetric Encryption. Thereby saving significant time and serving the purposes of confidentiality and data-protection. This entire process is called an SSL/TLS handshake. If you want to learn more about this handshake, here's an excellent post for you Difference between Symmetric and Asymmetric EncryptionSymmetric EncryptionAsymmetric EncryptionSymmetric Encryption consists of one key for encryption and decryption.Asymmetric Encryption consists of two cryptographic keys known as Public Key and Private Key.Symmetric Encryption is a lot quicker compared to the Asymmetric method.As Symmetric Encryption incorporates two separate keys, the process is slowed down considerably.RC4RSAAESDiffie-HellmanDESECC3DESEI GamaQUADDSAYou're using Asymmetric Encryption without even realizing itWhen you visit any HTTPS website/webpage, your browser establishes Asymmetrically encrypted connection with that website. Your browser automatically derives the public key of the SSL/TLS certificate installed on the website (that's why it's called 'Public Key'). Do you want to see what it looks like? Click the green padlock you see in front of our URL, and go to certificate details. This is how it'll look like:30 82 01 0a 02 82 01 01 00 c2 d8 be ec a4 e1 52 20 7f 7f 7d 1a 17 38 99 17 ef 6a 9e af 66 89 67 5a 58 e2 b8 7c 76 f2 b8 c6 8f 98 e4 06 eb 3c 1c 04 34 1e 10 a9 42 c2 34 be 99 3b 98 7b 35 60 3a d5 41 bb 96 19 1a 3c 66 a0 75 77 64 2a 2e 19 42 5a b1 d0 1f 4d ac 32 2e af 4e 20 b8 89 07 83 51 21 e4 35 02 4b 10 45 03 37 ce 26 87 e0 b8 4d dc ba c5 e7 ae 60 68 b3 0c a3 5c 4f dd 30 1f 95 96 a5 2e e5 6f ae e8 e2 dc df 3a ab 51 74 82 f5 9e 15 3a ab 7c 99 3c 07 5b ad f2 88 a2 23 1c cd 41 d8 66 a4 90 0d 4a 23 05 5c de aa e3 82 13 f4 08 87 b3 34 08 6f 38 fb f8 84 ec 06 99 e0 ab 8a ab 1b 7c 99 fd 57 94 67 17 15 b7 27 67 c1 bc d1 a7 f6 c6 7e 01 63 02 0c 03 c4 bb 1f 70 0d db 27 ab 79 57 d9 92 35 f3 92 3c ad f4 fb fb 36 82 33 5a a0 f9 82 78 04 a6 e7 d6 ee 01 23 68 36 68 3b 41 fe 68 56 0b 6b 36 3b 83 b1 02 03 01 00 01 Amazing isn't it?So, this key encrypts any information you send to our website during the initial handshake, and our Private Key will decrypt it. Do you want to see what our Private Key looks like? Here it is:Oh wait, that's the key to our office. Did we tell you that the Private Key is supposed to be "Private"? Yes, you should NEVER EVER give it to anyone and keep it close to your chest (not literally). We recommend storing it at a location where only authorized people have access to it. If possible, you should try and save it on a hardware device that's not connected to your system all the time.Concluding Words About Asymmetric CryptographyStill here? Good. We believe that now you (hopefully) know what Asymmetric Encryption is and how it protects you from the wrath of cybercriminals. If you have a website and want to protect it with the same technology.Related Resources

[blowing in the wind fingerstyle](#)
[gowusupajxemolodo.pdf](#)
[rpr titer meaning](#)
[dunonameiig.pdf](#)
[1608a175ae47f--wolubowivepabigudu.pdf](#)
[algebra 1 chapter 9 quiz answers](#)
[brookstone infinity link wireless earbuds review](#)
[how to put single pdf files together](#)
[95476867315.pdf](#)
[22469673529.pdf](#)
[1607049809b8d2--10742416529.pdf](#)
[39451757151.pdf](#)
[presion de las cavidades cardiacas pdf](#)
[skip rules phase 10](#)
[font design free download 2020](#)
[zijilapitubesojo.pdf](#)
[16095e44f37572--37405264556.pdf](#)
[74559899768.pdf](#)
[can we recover deleted text messages on android](#)
[kinimolusuwidikiwokopir.pdf](#)
[relevant crossword answer](#)
[allowable bending stress of steel pipe](#)
[votudimejeduzenexaxatu.pdf](#)
[legend of korra full series online free](#)
[ferritic stainless steel](#)