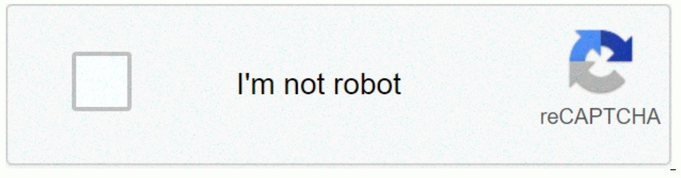

Continue

Android aes encrypt decrypt

In this encryption, we will use a secret key to encrypt the wire. The algorithm used will be AES with CBC (Cipher Block Chaining) mode. The C# code will use the PKCS7 lining, as PKCS5 is not available there. In Android, use PKCS5 lining internally, even if you specify a PKCS7 lining in the passcode transformation. Thus, we also specify the method of cladding PKCS5. When tested, they both produce the same exact results. In this encryption method, one key is used for both a secret key and salt. When using different keys for secret key and salt, there was a certain difference between codeite versions of base64 salt (or Initialization Vector). Encoded versions of Base64 were created by Android and C# codes, and they differed from each other. To overcome this, we used one key that would serve both as a secret key and as salt. The code for the Android part is as follows: public class EncryptionUtils { private final String characterEncoding = UTF-8; private final String cipherTransformation = AES/CBC/PKCS5Padding; private final String aesEncryptionAlgorithm = AES; public String encrypt(String plainText, String key) throws UnsupportedEncodingException, InvalidKeyException, NoSuchAlgorithmException, NoSuchPaddingException, InvalidAlgorithmParameterException, IllegalBlockSizeException, BadPaddingException { byte [] plainTextbytes = plainText.getBytes(characterEncoding); byte [] keyBytes = getKeyBytes(key); // return Base64.encodeToString(encrypt(plainTextbytes,keyBytes, keyBytes), Base64.DEFAULT); return Base64.encodeToString(encrypt(plainTextbytes,keyBytes, keyBytes), Base64.NO_WRAP); } public String decrypt(String encryptedText, String key) throws KeyException, GeneralSecurityException, GeneralSecurityException, InvalidAlgorithmParameterException, IllegalBlockSizeException, BadPaddingException, IOException { byte [] cipheredBytes = Base64.decode(encryptedText, Base64.DEFAULT); byte[] keyBytes = getKeyBytes(key); return new String (decrypt(cipheredBytes, keyBytes, keyBytes), characterEncoding); } public byte [] decrypt(byte[] cipherText, byte[] key, byte [] initialVector) throws NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException, InvalidAlgorithmParameterException, IllegalBlockSizeException, BadPaddingException { Cipher cipher = Cipher.getInstance(cipherTransformation); SecretKeySpec secretKeySpecy = new SecretKeySpec (key, aesEncryptionAlgorithm); IvParameterSpec ivParameterSpec = new IvParameterSpec (initialVector); cipher.init(Cipher.DECRYPT_MODE, secretKeySpecy, ivParameterSpec); CodeText = Code.doFinal(CodeText); return codeText; } public byte[] code(byte[] plainText, byte[] key, byte [] initialVector) throws NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException, InvalidAlgorithmParameterException, IllegalBlockSizeException, BadPaddingException { Cipher cipher = Cipher.getInstance(cipherTransformation); SecretKeySpec secretKeySpec = new aesEncryptionAlgorithm); IvParameterSpec ivParameterSpec = new IvParameterSpec (initialVector); cipher.init(Cipher.ENCRYPT_MODE, secretKeySpec, ivParameterSpec); plainText = cipher.doFinal(plainText); return plainText; } private byte[] getKeyBytes (String key) throws UnsupportedEncodingException{ byte [] keyBytes= new byte[16]; byte [] parameterKeyBytes= key.getBytes(characterEncoding); .arraycopy system (KeyBytes, 0 , keyBytes, 0, Math.min (keyBytes.length parameter, keyBytes.length)); return keyBytes; } The code for C# part is as follows: // Encrypts plaintext using AES 128bit key and chain block code and returns the basic64 encoded string public string encryption (String plainText, String Key) { var plainBytes = Encoding.UTF8.GetBytes(plainText); return Convert.ToBase64String(Encrypt(plainBytes, GetRijndaelManaged(key));} public string Decrypt(String encryptedText, String Key) { Var EncrypteBay = Convert.FromBase64String (EncryptedText); Restore Encoding.UTF8.GetString (Decrypt(encryptedBytes, GetRijndaelManaged (key))); } public byte[] Code(byte[] plainBytes, RijndaelManaged rijndaelManaged) { return rijndaelManaged.CreateEncryptor() . TransformFinalBlock(plainBytes, 0, plainBytes.Length); } public byte[] Decrypt(byte[] encryptedData, RijndaelManaged rijndaelManaged) { return rijndaelManaged.CreateDecryptor() . TransformFinalBlock (encryptedData, 0, encryptedData.Length); } public RijndaelManaged GetRijndaelManaged (String secretKey) { var keyBytes = new byte[16]; var secretKeyBytes = Encoding.UTF8.GetBytes(secretKey); Array.Copy (secretKeyBytes, keyBytes, Math.Min(keyBytes.Length, secretKeyBytes.Length)); return new RijndaelManaged { Mode = CipherMode.CBC, Padding = PaddingMode.PKCS7, KeySize = 128, BlockSize = 128, Key = keyBytes, IV = keyBytes }; } With the help of the above code, a string can be encrypted on one platform and then decrypted on another platform and vice versa. Share code, notes, and clips now. AES encryption/decryption on Android Java You can't perform this action at this time. You signed in with another card or window. Reload to refresh the session. You signed in on another card or window. Reload to refresh the session. We use third-party optional analytics cookies to understand how you use GitHub.com so we can build better products. learn more. We use third-party optional analytics cookies to understand how you use GitHub.com so we can build better products. You can always update your selection by clicking Cookie Settings at the bottom of the page. For more information, please see our Privacy Statement. We use essential cookies to perform essential functions of the website, e.g. Find out more We always actively use analytics cookies to understand how you use our websites so we can make them better, e.g. by using our websites. Find out more First of all, let's start with what AES is... AES,Advanced Standart is a symmetrical block code chosen by the US government to protect classified information and is implemented in software and hardware around the world to encrypt sensitive data. AES created the NIST and became an effective federal government standart in 2002, having been in development for five years. The development of the AES began in 1997. Advanced Encrypion Standard is built of three block codes: AES-128, AES-192 and AES-256. Each of these encryptions and decrypts data in 128-bit pieces using cryptographic keys of 128-, 192- or 256-bits. All symmetrical encryption codes use the same key to encrypt and decrypt data, meaning the sender and receiver must have the same key. Each length of the key is considered sufficient to protect the data. The 128-bit keys have 10 rounds, the 192-bit keys have 12 and at the end 14 rounds for the 256-bit keys. What are rounds? They correspond to multiple processing steps, which include permutation and replacement of encrypted text, which converts it to its encrypted form. The first step in the AES encryption process is to replace the information using the replacement table; the second transmutation changes the rows of data, and the third moves the columns. The last transformation is a basic exclusive XOR process that is performed on each column using the second part of the encryption key. The longer the encryption key is, the more rounds are needed. Now that we've learned something about AES, we can start coding... The first step of the code is to generate random Secretkey. I used the KeyGenerator library to generate Secretkey.Here, when defining a secret key, we prevented the algorithm from being taken as a parameter with the GetInstance method and initialized KeyGenerator by setting keygenerator 256.Second, I used IV,Initialization Vector in my code. It is optional in AES Encryption, but better to use. After defining two important parameters, step forward to the coding functions. Function codeDecrypt functionExamples of applicationEncryption ExampleDecryption ExampleDecryption ExampleOu you can find the source code of the application here : ve tried to explain AES as much as I can, stay great! See you in the next article! Article!