# 每日一题 - 实现一个对称加密算法

### 信息卡片

* 时间：2019-07-29
* tag：非对称加密 加密算法

## 问题描述

我们在互联网中的操作如果没有经过加密算法的加密是非常危险的，使用HTTPS是一个很好的选择， 它基于TSL/SSL 对传输层的内容进行加密。 尽管其中的核心算法是非对称加密， 但是真正传输数据的过程仍然是对称加密，其原因就是非对称加密的复杂性。

那么如何实现一个对称加密算法呢？补充代码，使之符合预期。

function encrpt(msg, key) {  
 // 补充代码  
}  
  
function decrypt(secretMsg, key) {  
 // 补充代码  
}  
  
const msg = "hello";  
const key = null; // key can be anything  
  
const secretMsg = encrpt(msg, key);  
  
  
console.log(secretMsg); // secretMsg !== msg  
  
console.log(decrypt(secretMsg, key) === msg); // true

其实，服务端和客户端用对称加密之前，协商的过程就是确定算法（比如上述就是一种算法）以及key的过程

## 参考代码

function encrpt(msg, key) {  
 // 补充代码  
 const secretMsg = [];  
 for (let i = 0; i < msg.length; i++) {  
 secretMsg[i] = msg[i].charCodeAt() + key;  
 }  
  
 return secretMsg.join('\n');  
}  
  
function decrypt(msg, key) {  
 // 补充代码  
 const secretMsg = msg.split('\n')  
 const ret = [];  
 for (let i = 0; i < secretMsg.length; i++) {  
 ret[i] = String.fromCharCode(secretMsg[i] - key);  
 }  
  
 return ret.join("");  
}  
  
const msg = "hello";  
const key = 3;  
  
const secretMsg = encrpt(msg, key);  
  
decrypt(secretMsg, key) === msg;  
  
console.log(secretMsg); // secretMsg !== msg  
  
console.log(decrypt(secretMsg, key) === msg); // true

## 扩展

如何优化你的算法，使得加密之后的字符串尽可能没有规律。