<h1>Connexion s&#65533;uris&#65533; sur un site web</h1> Cet article est un rappel concernant la s&#65533;urisation de la connexion &#65533;un site web. J'aborde bri&#65533;ement le stockage des mots de passe des usagers, la connexion des utilisateurs et le changement de leur mot de passe. Nous allons voir ici une technique simple &#65533;mettre en &#65533;uvre pour assurer un minimum de s&#65533;urit&#65533;&#65533;l'utilisateur et lui &#65533;iter des ennuis en prenant au s&#65533;ieux ces donn&#65533;s et leurs traitements. Tout d'abord, il faut savoir qu'un utilisateur a des comptes sur plusieurs sites internet et qu'il utilise souvent le m&#65533;e nom d'utilisateur et mot de passe sur les diff&#65533;ents sites. On doit donc s&#65533;uriser nos donn&#65533;s pour qu'une intrusion ou une faille ne permette &#65533;un tiers de mettre la main sur ces donn&#65533;s.&#65533; <h2><u>1) Stockage des mots de passe des usagers</u></h2> Le plus souvent, on utilise une base de donn&#65533;s pour stocker les mots de passe de tous les utilisateurs pour avoir un traitement automatis&#65533;de l'information le plus efficace et rapide que possible. En th&#65533;rie, seul vous pouvez vous connecter &#65533;votre base de donn&#65533;s pour la consulter. En r&#65533;lit&#65533; vous ne pouvez pas le savoir. En effet, si votre site ou le serveur sur lequel il se trouve a &#65533;&#65533;compromis, une autre personne peut avoir r&#65533;up&#65533;&#65533;des informations issues de votre base de donn&#65533;s ou avoir mis la main sur les codes de connexion &#65533;votre base. Il faut donc &#65533;iter de stocker directement les mots de passe dans la base. En r&#65533;lit&#65533; m&#65533;e un cryptage du mot de passe n'est pas possible puisque la cl&#65533;et l'algorithme seront disponibles pour l'intrus. Il ne reste plus qu'une seule solution, stocker une empreinte du mot de passe. Habituellement, on le fait avec une fonction de hachage qui va cr&#65533;r un r&#65533;um&#65533;du message et qui ne permettra pas de retrouver le mot de passe original. A ce niveau, plusieurs fonctions sont disponibles comme MD5 ou SHA1. Je vous conseille de rechercher si la s&#65533;urit&#65533;de la fonction que vous utiliserez est toujours bonne. En effet, les techniques &#65533;oluent et les puissances de calculs disponibles augmentent rapidement. Au moment o&#65533; j'ai r&#65533;ig&#65533;cet article, la fonction MD5 n'est plus s&#65533;uris&#65533; puisqu'il est maintenant possible de cr&#65533;r un mot de passe clair qui correspond &#65533;votre fonction. En ce qui concerne SHA1, il fait l'objet d'attaques r&#65533;urrentes et il devrait surement bient&#65533; tomber. {mooblock=Mise en place (cliquez pour voir)} Voici la partie serveur d'un code simple qui permet de mettre en pratique un GDS. Il vous reste &#65533;vous cr&#65533;r une fonction rand_str() bien &#65533;vous. Moi ici, j'ai une fonction qui va g&#65533;&#65533;er une chaine une cha&#65533;e al&#65533;toire de taille 40 caract&#65533;es. {codecitation class="brush: php; gutter: false;"}if (!isset($_SESSION['gds']))
 {
 $_SESSION['gds'] = rand_str(40);
 }
&#65533;if((isset($_POST['log_login'])) && (isset($_POST['hashpassword'])))
 {
 $f_login = $_POST['log_login'];
 $log_ok = 0;
 $reqUser = "SELECT * FROM USERS WHERE login = '".$f_login."';";
 foreach($bdd->query($reqUser,PDO::FETCH_ASSOC) as $user)
 {
 if(($user['LOGIN'] == $f_login) && ($_POST['hashpassword'] == sha1($user["MDP"] . $_SESSION['gds'])))
 {
 $_SESSION = $user;
 $log_ok = 1;
 }
 }
 if(!$log_ok)
 {
 include_once("logout.php");
 }
&#65533;}{/codecitation} Pour que l'utilisateur puisse valider la formulaire, j'ai ajout&#65533;ces scripts dans la page : {codecitation class="brush: php; gutter: false;"}<script src="js/utf8_encode.js" mce_src="js/utf8_encode.js" type="text/javascript"></script>
 <script src="js/sha1.js" mce_src="js/sha1.js" type="text/javascript"></script>
 <script type="text/javascript">
 <!--
 function valider(formulaire)
 {
 formulaire.hashpassword.value = sha1(sha1(formulaire.log_mdp.value) + '<?php echo $_SESSION['gds']; ?>');
 formulaire.log_mdp.value = '';
 return true;
 }
 -->
 </script>{/codecitation} Il faut pr&#65533;iser la fonction qui valide le formulaire et les 2 champs qui g&#65533;ent le mot de passe : {codecitation class="brush: php; gutter:

false;"}<form action="votrepage.php" method="post" onsubmit="return valider(this)" id="formlog">
<input type="hidden" name="hashpassword" id="" />
<input type="password" name="log_mdp" id="log_mdp" />{/codecitation}
&#65533; &#65533;
Il ne reste plus qu'&#65533;mettre dans le dossier js les 2 scripts que vous pouvez trouver &#65533;cette adresse. utf8_encode.js : {codecitation class="brush: jscript; gutter: false;"}function utf8_encode (argString) {
 // Encodes an ISO-8859-1 string to UTF-8
 //
 // version: 1008.1718
 // discuss at: http://phpjs.org/functions/utf8_encode
 // + original by: Webtoolkit.info (http://www.webtoolkit.info/)
 // + improved by: Kevin van Zonneveld (http://kevin.vanzonneveld.net)
 // + improved by: sowberry
 // + tweaked by: Jack
 // + bugfixed by: Onno Marsman
 // + improved by: Yves Sucaet
 // + bugfixed by: Onno Marsman
 // + bugfixed by: Ulrich
 // * example 1: utf8_encode('Kevin van Zonneveld');
 // * returns 1: 'Kevin van Zonneveld'
 var string = (argString+''); // .replace(/rn/g, "n").replace(/r/g, "n");

 var utftext = "";
 var start, end;
 var stringl = 0;

 start = end = 0;
 stringl = string.length;
 for (var n = 0; n < stringl; n++) {
 var c1 = string.charCodeAt(n);
 var enc = null;

 if (c1 < 128) {
 end++;
 } else if (c1 > 127 && c1 < 2048) {
 enc = String.fromCharCode((c1 >> 6) | 192) + String.fromCharCode((c1 & 63) | 128);
 } else {
 enc = String.fromCharCode((c1 >> 12) | 224) + String.fromCharCode(((c1 >> 6) & 63) | 128) + String.fromCharCode((c1 & 63) | 128);
 }
 if (enc !== null) {
 if (end > start) {
 utftext += string.substring(start, end);
 }
 utftext += enc;
 start = end = n+1;
 }
 }

 if (end > start) {
 utftext += string.substring(start, string.length);
 }

 return utftext;
}{/codecitation} &#65533;sha1.js : {codecitation class="brush: jscript; gutter: false;"}function sha1 (str) {
 // Calculate the sha1 hash of a string
 //
 // version: 1008.1718
 // discuss at: http://phpjs.org/functions/sha1
 // + original by: Webtoolkit.info (http://www.webtoolkit.info/)
 // + namespaced by: Michael White (http://getsprink.com)
 // + input by: Brett Zamir (http://brett-zamir.me)
 // + improved by: Kevin van Zonneveld (http://kevin.vanzonneveld.net)
 // - depends on: utf8_encode
 // * example 1: sha1('Kevin van Zonneveld');
 // * returns 1: '54916d2e62f65b3afa6e192e6a601cdbe5cb5897'
 var rotate_left = function (n,s) {
 var t4 = (n<<s) | (n>>>(32-s));
 return t4;
 };

 /*var lsb_hex = function (val) { // Not in use; needed?
 var str="";
 var i;
 var vh;
 var vl;

 for (i=0; i<=6; i+=2) {
 vh = (val>>>(i*4+4))&0x0f;
 vl = (val>>>(i*4))&0x0f;
 str += vh.toString(16) + vl.toString(16);
 }
 return str;
 };*/

 var cvt_hex = function (val) {
 var str="";
 var i;
 var v;

 for (i=7; i>=0; i--) {
 v = (val>>>(i*4))&0x0f;
 str += v.toString(16);
 }
 return str;
 };

 var blockstart;
 var i, j;
 var W = new Array(80);
 var H0 = 0x67452301;
 var H1 = 0xEFCDAB89;
 var H2 = 0x98BADCFE;
 var H3 = 0x10325476;
 var H4 = 0xC3D2E1F0;
 var A, B, C, D, E;
 var temp;

 str = this.utf8_encode(str);
 var str_len = str.length;

 var word_array = [];
 for (i=0; i<str_len-3; i+=4) {
 j = str.charCodeAt(i)<<24 | str.charCodeAt(i+1)<<16 |
 str.charCodeAt(i+2)<<8 | str.charCodeAt(i+3);
 word_array.push(j);
 }

 switch (str_len % 4) {
 case 0:

i = 0x080000000;
 break;
 case 1:
 i = str.charCodeAt(str_len-1)<<24 | 0x0800000;
 break;
 case 2:
 i = str.charCodeAt(str_len-2)<<24 | str.charCodeAt(str_len-1)<<16 | 0x08000;
 break;
 case 3:
 i = str.charCodeAt(str_len-3)<<24 | str.charCodeAt(str_len-2)<<16 | str.charCodeAt(str_len-1)<<8 | 0x80;
 break;
 }

 word_array.push(i);

 while ((word_array.length % 16) != 14) {word_array.push(0);}

 word_array.push(str_len>>>29);
 word_array.push((str_len<<3)&0x0ffffffff);

 for (blockstart=0; blockstart<word_array.length; blockstart+=16) {
 for (i=0; i<16; i++) {W[i] = word_array[blockstart+i];}
 for (i=16; i<=79; i++) {W[i] = rotate_left(W[i-3] ^ W[i-8] ^ W[i-14] ^ W[i-16], 1);}

 A = H0;
 B = H1;
 C = H2;
 D = H3;
 E = H4;

 for (i= 0; i<=19; i++) {
 temp = (rotate_left(A,5) + ((B&C) | (~B&D)) + E + W[i] + 0x5A827999) & 0x0ffffffff;
 E = D;
 D = C;
 C = rotate_left(B,30);
 B = A;
 A = temp;
 }

 for (i=20; i<=39; i++) {
 temp = (rotate_left(A,5) + (B ^ C ^ D) + E + W[i] + 0x6ED9EBA1) & 0x0ffffffff;
 E = D;
 D = C;
 C = rotate_left(B,30);
 B = A;
 A = temp;
 }

 for (i=40; i<=59; i++) {
 temp = (rotate_left(A,5) + ((B&C) | (B&D) | (C&D)) + E + W[i] + 0x8F1BBCDC) & 0x0ffffffff;
 E = D;
 D = C;
 C = rotate_left(B,30);
 B = A;
 A = temp;
 }

 for (i=60; i<=79; i++) {
 temp = (rotate_left(A,5) + (B ^ C ^ D) + E + W[i] + 0xCA62C1D6) & 0x0ffffffff;
 E = D;
 D = C;
 C = rotate_left(B,30);
 B = A;
 A = temp;
 }

 H0 = (H0 + A) & 0x0ffffffff;
 H1 = (H1 + B) & 0x0ffffffff;
 H2 = (H2 + C) & 0x0ffffffff;
 H3 = (H3 + D) & 0x0ffffffff;
 H4 = (H4 + E) & 0x0ffffffff;
 }

 temp = cvt_hex(H0) + cvt_hex(H1) + cvt_hex(H2) + cvt_hex(H3) + cvt_hex(H4);
 return temp.toLowerCase();
}{/codecitation}{/mooblock} <h2><u>2) Connexion des utilisateurs</u></h2> Sachant que vous ne contr�ez pas le r�eau entre votre serveur et l'utilisateur, on ne peut pas transmettre en clair un mot de passe. Celui-ci pourrait �re intercept�par une machine sur le r�eau. On appelle cette technique l'attaque �man in the middle� puisque c'est une personne qui s'intercale entre les 2 ordinateurs. Donc, le plus simple serait-il de calculer le hash du mot de passe cot�client en JavaScript et de l'envoyer ? La r�onse est NON! En effet, il suffit �la personne qui intercepte la communication de renvoyer les m�es donn�s pour �re identifi��la place de l'utilisateur. Le mot de passe est alors devenu une chaine qui sert �un g��ateur de mot de passe et le hash lui est devenu le vrai mot de passe. Alors comment faire pour identifier un utilisateur ? Il faut utiliser la technique du Grain De Sel. Le principe : 1- A l'affichage de la page de connexion par l'utilisateur, on va g��er une chaine le plus al�toire possible qui changera �chaque fois qu'un utilisateur affiche cette page. Pour en garder une trace, on peut sauvegarder ce GDS dans la session de l'utilisateur. 2- Hash en JavaScriptdu mot de passe c��client. 3- Concat�ation du hash obtenu avec le GDS.� 4- Hash de la�concat�ation obtenu en 3. 5- Envoi du r�ultat Ainsi, �chaque fois que l'utilisateur se connectera, la chaine servant �l'identification du mot de passe change et une personne �outant la ligne ne peut pas deviner

le mot de passe m&#65533;e s'il connait le GDS. <h2><u>3) Changement de mot de passe</u></h2> Si un utilisateur oublie son mot de passe ou si on veut lui laisser la possibilit&#65533;de le changer, le m&#65533;e probl&#65533;e se pose sauf que cette fois si, on ne pourra pas utiliser un hachage associ&#65533;&#65533;un Grain De Sel pour r&#65533;up&#65533;er le mot de passe puisque cette technique permet uniquement de v&#65533;ifi&#65533;si un mot de passe est correct. J'ai donc mis au point un syst&#65533;e qui g&#65533;&#65533;e des cl&#65533; de cryptage &#65533;chaque fois que l'utilisateur demande &#65533;modifier son mot de passe pour utiliser un chiffrement asym&#65533;rique RSA. e syst&#65533;e ne peut pas &#65533;re utilis&#65533;&#65533;chaque fois qu'un utilisateur se connecte car il demande beaucoup plus de ressource au serveur. Il doit &#65533;re utilis&#65533;avec parcimonie. {mooblock=Voici un exemple d'utilisation : (cliquez pour voir)} {codecitation class="brush: php; gutter: false;"}include_once("rsa.php");
$keys = generate_keys();&#65533;
$_SESSION['rsa_keys'] = $keys; ?>
<script src="js/rsa.js" type="text/javascript"></script>{/codecitation}&#65533; Ne pas oublier de mettre le code javascript qui va g&#65533;&#65533;er le mot de passe : {codecitation class="brush: php; gutter: false;"}<script type="text/javascript">
<!--
function valider(formulaire)
{
/* Mettre ici les v&#65533;ifications du formulaire et d&#65533;inir l'&#65533;at de "ok" */
 if (ok)
 {
&#65533; if (mdp.length >= 8)
 {
 var mdpcrypt = rsa_encrypt(formulaire.insc_mdp.value, <?php echo $keys[1].', '.$keys[0]; ?>);
 formulaire.insc_mdp.value = mdpcrypt;
 formulaire.insc_confirm_mdp.value = mdpcrypt;
 }
 return true;
 }
 return false;
}
-->
</script>{/codecitation} Fichier rsa.php :&#65533; {codecitation class="brush: php; gutter: false;"}<?php
/*

*v.1.3 [2 Sep 2002]

* Rivest/Shamir/Adelman (RSA) compatible functions
* to generate keys and encode/decode plaintext messages.
* Plaintext must contain only ASCII(32) - ASCII(126) characters.

*Send questions and suggestions to Ilya Rudev <www <at> polar-lights <dot> com> (Polar Lights Labs)

*most part of code ported from different
*C++, JS and Flash
*RSA examples found in books and in the net :)

*supplied with Hacker Hunter authentication system.
*http://www.polar-lights.com/hackerhunter/

*It 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.

*With a great thanks to:
*Glenn Haecker <ghaecker <at> idworld <dot> net>
*Segey Semenov <sergei2002 <at> mail <dot> ru>
*Suivan <ssuuii <at> gmx <dot> net>
*/

/*random generator seed */
mt_srand((double)microtime()*1000000);

/*
* Prime numbers table
* 570 prime numbers (Array can not be be enlarged)
* 4507, 4513 is the smallest and 9521, 9533 is the largest pair
* Still have no time to find why 9521, 9533 is the largest - sorry :)
*/
$primes = array (4507, 4513, 4517, 4519, 4523, 4547, 4549, 4561, 4567, 4583, 4591, 4597,
4603, 4621, 4637, 4639, 4643, 4649, 4651, 4657, 4663, 4673, 4679, 4691, 4703, 4721, 4723, 4729, 4733, 4751,
4759, 4783, 4787, 4789, 4793, 4799, 4801, 4813, 4817, 4831, 4861, 4871, 4877, 4889, 4903, 4909, 4919, 4931,
4933, 4937, 4943, 4951, 4957, 4967, 4969, 4973, 4987, 4993, 4999, 5003, 5009, 5011, 5021, 5023, 5039, 5051,
5059, 5077, 5081, 5087, 5099, 5101, 5107, 5113, 5119, 5147, 5153, 5167, 5171, 5179, 5189, 5197, 5209, 5227,
5231, 5233, 5237, 5261, 5273, 5279, 5281, 5297, 5303, 5309, 5323, 5333, 5347, 5351, 5381, 5387, 5393, 5399,
5407, 5413, 5417, 5419, 5431, 5437, 5441, 5443, 5449, 5471, 5477, 5479,

5483, 5501, 5503, 5507, 5519, 5521,
5527, 5531, 5557, 5563, 5569, 5573, 5581, 5591, 5623, 5639, 5641, 5647, 5651, 5653, 5657, 5659, 5669, 5683,
5689, 5693, 5701, 5711, 5717, 5737, 5741, 5743, 5749, 5779, 5783, 5791, 5801, 5807, 5813, 5821, 5827, 5839,
5843, 5849, 5851, 5857, 5861, 5867, 5869, 5879, 5881, 5897, 5903, 5923, 5927, 5939, 5953, 5981, 5987, 6007,
6011, 6029, 6037, 6043, 6047, 6053, 6067, 6073, 6079, 6089, 6091, 6101, 6113, 6121, 6131, 6133, 6143, 6151,
6163, 6173, 6197, 6199, 6203, 6211, 6217, 6221, 6229, 6247, 6257, 6263, 6269, 6271, 6277, 6287, 6299, 6301,
6311, 6317, 6323, 6329, 6337, 6343, 6353, 6359, 6361, 6367, 6373, 6379, 6389, 6397, 6421, 6427, 6449, 6451,
6469, 6473, 6481, 6491, 6521, 6529, 6547, 6551, 6553, 6563, 6569, 6571, 6577, 6581, 6599, 6607, 6619, 6637,
6653, 6659, 6661, 6673, 6679, 6689, 6691, 6701, 6703, 6709, 6719, 6733, 6737, 6761, 6763, 6779, 6781, 6791,
6793, 6803, 6823, 6827, 6829, 6833, 6841, 6857, 6863, 6869, 6871, 6883, 6899, 6907, 6911, 6917, 6947, 6949,
6959, 6961, 6967, 6971, 6977, 6983, 6991, 6997, 7001, 7013, 7019, 7027, 7039, 7043, 7057, 7069, 7079, 7103,
7109, 7121, 7127, 7129, 7151, 7159, 7177, 7187, 7193, 7207, 7211, 7213, 7219, 7229, 7237, 7243, 7247, 7253,
7283, 7297, 7307, 7309, 7321, 7331, 7333, 7349, 7351, 7369, 7393, 7411, 7417, 7433, 7451, 7457, 7459, 7477,
7481, 7487, 7489, 7499, 7507, 7517, 7523, 7529, 7537, 7541, 7547, 7549, 7559, 7561, 7573, 7577, 7583, 7589,
7591, 7603, 7607, 7621, 7639, 7643, 7649, 7669, 7673, 7681, 7687, 7691, 7699, 7703, 7717, 7723, 7727, 7741,
7753, 7757, 7759, 7789, 7793, 7817, 7823, 7829, 7841, 7853, 7867, 7873, 7877, 7879, 7883, 7901, 7907, 7919,
7927, 7933, 7937, 7949, 7951, 7963, 7993, 8009, 8011, 8017, 8039, 8053, 8059, 8069, 8081, 8087, 8089, 8093,
8101, 8111, 8117, 8123, 8147, 8161, 8167, 8171, 8179, 8191, 8209, 8219, 8221, 8231, 8233, 8237, 8243, 8263,
8269, 8273, 8287, 8291, 8293, 8297, 8311, 8317, 8329, 8353, 8363, 8369, 8377, 8387, 8389, 8419, 8423, 8429,
8431, 8443, 8447, 8461, 8467, 8501, 8513, 8521, 8527, 8537, 8539, 8543, 8563, 8573, 8581, 8597, 8599, 8609,
8623, 8627, 8629, 8641, 8647, 8663, 8669, 8677, 8681, 8689, 8693, 8699, 8707, 8713, 8719, 8731, 8737, 8741,
8747, 8753, 8761, 8779, 8783, 8803, 8807, 8819, 8821, 8831, 8837, 8839, 8849, 8861, 8863, 8867, 8887, 8893,
8923, 8929, 8933, 8941, 8951, 8963, 8969, 8971, 8999, 9001, 9007, 9011, 9013, 9029, 9041, 9043, 9049, 9059,
9067, 9091, 9103, 9109, 9127, 9133, 9137, 9151, 9157, 9161, 9173, 9181, 9187, 9199, 9203, 9209, 9221, 9227,
9239, 9241, 9257, 9277, 9281, 9283, 9293, 9311, 9319, 9323, 9337, 9341, 9343, 9349, 9371, 9377, 9391, 9397,
9403, 9413, 9419, 9421, 9431, 9433, 9437, 9439, 9461, 9463, 9467, 9473, 9479, 9491, 9497, 9511, 9521, 9533);

$maxprimes = count($primes) - 1;

/*Function for generating keys. Return array where
$array[0] -> modulo N
$array[1] -> public key E
$array[2] -> private key D
Public key pair is N and E
Private key pair is N and D
*/
function generate_keys ($show_debug=0){

 global $primes, $maxprimes;
 while (empty($e) || empty($d)) {
 /*finding 2 small prime numbers $p and $q
 $p and $q must be different*/
 $p = $primes[mt_rand(0, $maxprimes)];
 while (empty($q) || ($p==$q)) $q = $primes[mt_rand(0, $maxprimes)];
 //second part of public and private pairs - N
 $n = $p*$q;

 //$pi (we need it to calculate D and E)
 $pi = ($p - 1) * ($q - 1);

 // Public key E
 $e = tofindE($pi, $p, $q);

 // Private key D
 $d = extend($e, $pi);

 $keys = array ($n, $e, $d);
 if ($show_debug) {
 echo "P = $p
Q = $q
N = $n -

modulo
PI = $pi
E = $e - public key
D = $d - private key<p>";
 }
 }
 return $keys;
}

/* modulus function */
function mo ($g, $l) {
 return $g - ($l * floor ($g/$l));
}

/*
* Standard method of calculating D
* D = E-1 (mod N)
* It's presumed D will be found in less then 16 iterations
*/
function extend ($Ee, $Epi) {
 $u1 = 1;
 $u2 = 0;
 $u3 = $Epi;
 $v1 = 0;
 $v2 = 1;
 $v3 = $Ee;
 while ($v3 != 0) {
 $qq = floor($u3/$v3);
 $t1 = $u1 - $qq * $v1;
 $t2 = $u2 - $qq * $v2;
 $t3 = $u3 - $qq * $v3;
 $u1 = $v1;
 $u2 = $v2;
 $u3 = $v3;
 $v1 = $t1;
 $v2 = $t2;
 $v3 = $t3;
 $z = 1;
 }
 $uu = $u1;
 $vv = $u2;
 if ($vv < 0) {
 $inverse = $vv + $Epi;
 } else {
 $inverse = $vv;
 }
 return $inverse;
}

/* This function return Greatest Common Divisor for $e and $pi numbers */
function GCD($e, $pi) {
 $y = $e;
 $x = $pi;
 while ($y != 0) {
 $w = mo($x , $y);
 $x = $y;
 $y = $w;

 }
 return $x;
}

/*function for calculating E under conditions:
GCD(N, E) = 1 and 1<E<N
If each test E is prime, there will be much less loops in that fuction
and smaller E means less JS calculations on client side */
/*
* Calculating E under conditions:
* GCD(N, E) = 1 and 1<E<N
* If E is prime, there will be fewer loops in the function.
* Smaller E also means reduced JS calculations on client side.
*/
function tofindE($pi) {
 global $primes, $maxprimes;
 $great = 0;
 $cc = mt_rand (0, $maxprimes);
 $startcc = $cc;
 while ($cc >= 0) {
 $se = $primes[$cc];
 $great = GCD($se, $pi);
 $cc--;
 if ($great == 1) break;
 }
 if ($great == 0) {
 $cc = $startcc + 1;
 while ($cc <= $maxprimes) {
 $se = $primes[$cc];
 $great = GCD($se, $pi);
 $cc++;
 if ($great == 1) break;
 }
 }
 return $se;
}

/*
* ENCRYPT function returns
*, X = M^E (mod N)
* Please check http://www.ge.kochi-ct.ac.jp/cgi-bin-takagi/calcmodp
* and Flash5 RSA .fla by R.Vijay <rveejay0 <at> hotmail <dot> com> at
* http://www.digitalillusion.co.in/lab/rsaencyp.htm
* It is one of the simplest examples for binary RSA calculations
*
* Each letter in the message is represented as its ASCII code number - 30
* 3 letters in each block with 1 in the beginning and end.
* For example string
*, AAA
* will become
*, 13535351 (A = ASCII 65-30 = 35)
* we can build these blocks because the smalest prime available is 4507
*, 4507^2 = 20313049
* This means that
*, 1. Modulo N will always be < 19999991
*, 2. Letters > ASCII 128 must not occur in plain text message
*/

function rsa_encrypt ($m, $e, $n) {
 $asci = array ();
 for ($i=0; $i<strlen($m); $i+=3) {
 $tmpasci="1";
 for ($h=0; $h<3; $h++) {
 if ($i+$h <strlen($m)) {
 $tmpstr = ord (substr ($m, $i+$h, 1)) - 30;
 if (strlen($tmpstr) < 2) {
 $tmpstr ="0".$tmpstr;
 }
 } else {
 break;
 }
 $tmpasci .=$tmpstr;
 }
 array_push($asci, $tmpasci."1");
 }

 //Each number is then encrypted using the RSA formula: block ^E mod N
 $coded = "";
 for ($k=0; $k< count ($asci); $k++) {
 $resultmod = powmod($asci[$k], $e, $n);
 $coded .= $resultmod." ";
 }
 return trim($coded);
}

/*Russian Peasant method for exponentiation */
function powmod ($base, $exp, $modulus) {
 $accum = 1;
 $i = 0;
 $basepow2 = $base;
 while (($exp >> $i)>0) {
 if (($exp

>> $i) & 1) == 1) {
 $accum = mo(($accum * $basepow2) , $modulus);
 }
 $basepow2 = mo(($basepow2 * $basepow2) , $modulus);
 $i++;
 }
 return $accum;
}

/*
ENCRYPT function returns
M = X^D (mod N)
*/
function rsa_decrypt ($c, $d, $n) {
 //Strip the blank spaces from the ecrypted text and store it in an array
 $decryptarray = explode(" ", $c);
 for ($u=0; $u<count ($decryptarray); $u++) {
 if ($decryptarray[$u] == "") {
 array_splice($decryptarray, $u, 1);
 }
 }
 //Each number is then decrypted using the RSA formula: block ^D mod N
 $deencrypt = "";
 for ($u=0; $u< count($decryptarray); $u++) {
 $resultmod = powmod($decryptarray[$u], $d, $n);
 //remove leading and trailing '1' digits
 $deencrypt.= substr ($resultmod, 1, strlen($resultmod)-2);
 }
 $resultd = "";
 //Each ASCII code number + 30 in the message is represented as its letter
 for ($u=0; $u<strlen($deencrypt); $u+=2) {
 $resultd .= chr(substr ($deencrypt, $u, 2) + 30);
 }
 return $resultd;
}

/* Example */

/*echo"Keys:
";
$keys = generate_keys (1);
//or just
//$keys = generate_keys ();
$message="";
for ($i=32;$i<127;$i++) $message.=chr($i);
$encoded = rsa_encrypt ($message, $keys[1], $keys[0]);
$decoded = rsa_decrypt($encoded, $keys[2], $keys[0]);
echo "<pre>
Test ASCII(32) - ASCII(126):n";
echo "Message: $messagen";
echo "Encoded: $encodedn";
echo "Decoded: $decodedn";
echo "Success: ".(($decoded == $message) ? "True" : "False")."</pre>n";
*/
?>{/codecitation} rsa.js : {codecitation class="brush: html; gutter: false;"}function strlen (string) {
 // Get string length
 //
 // version: 1008.1718
 // discuss at: http://phpjs.org/functions/strlen
 // + original by: Kevin van Zonneveld (http://kevin.vanzonneveld.net)
 // + improved by: Sakimori
 // + input by: Kirk Strobeck
 // + improved by: Kevin van Zonneveld (http://kevin.vanzonneveld.net)
 // + bugfixed by: Onno Marsman
 // + revised by: Brett Zamir (http://brett-zamir.me)
 // % note 1: May look like overkill, but in order to be truly faithful to handling all Unicode
 // % note 1: characters and to this function in PHP which does not count the number of bytes
 // % note 1: but counts the number of characters, something like this is really necessary.
 // * example 1: strlen('Kevin van Zonneveld');
 // * returns 1: 19
 // * example 2: strlen('Aud87eudc04Z');
 // * returns 2: 3
 var str = string+";
 var i = 0, chr = '', lgth = 0;

 if (!this.php_js || !this.php_js.ini || !this.php_js.ini['unicode.semantics'] ||
 this.php_js.ini['unicode.semantics'].local_value.toLowerCase() !== 'on') {
 return string.length;
 }

 var getWholeChar = function (str, i) {
 var code = str.charCodeAt(i);
 var next = '', prev = '';
 if (0xD800 <= code && code <= 0xDBFF) { // High surrogate (could change last hex to 0xDB7F to treat high private surrogates as single characters)
 if (str.length <= (i+1)) {
 throw 'High surrogate without following low surrogate';
 }
 next = str.charCodeAt(i+1);
 if (0xDC00 > next || next > 0xDFFF) {
 throw 'High surrogate without following low surrogate';
 }
 return str.charAt(i)+str.charAt(i+1);
 } else if (0xDC00 <= code && code <= 0xDFFF) { // Low surrogate
 if (i === 0) {
 throw 'Low surrogate without preceding high surrogate';
 }
 prev = str.charCodeAt(i-1);
 if (0xD800 > prev || prev > 0xDBFF) { //(could change last hex

to 0xDB7F to treat high private surrogates as single characters)
 throw 'Low surrogate without preceding high surrogate';
 }
 return false; // We can pass over low surrogates now as the second component in a pair which we have already processed
 }
 return str.charAt(i);
 };

 for (i=0, lgth=0; i < str.length; i++) {
 if ((chr = getWholeChar(str, i)) === false) {
 continue;
 } // Adapt this line at the top of any loop, passing in the whole string and the current iteration and returning a variable to represent the individual character; purpose is to treat the first part of a surrogate pair as the whole character and then ignore the second part
 lgth++;
 }
 return lgth;
}

function ord (string) {
 // Returns the codepoint value of a character
 //
 // version: 1008.1718
 // discuss at: http://phpjs.org/functions/ord
 // + original by: Kevin van Zonneveld (http://kevin.vanzonneveld.net)
 // + bugfixed by: Onno Marsman
 // + improved by: Brett Zamir (http://brett-zamir.me)
 // * example 1: ord('K');
 // * returns 1: 75
 // * example 2: ord('uD800uDC00'); // surrogate pair to create a single Unicode character
 // * returns 2: 65536
 var str = string + '';

 var code = str.charCodeAt(0);
 if (0xD800 <= code && code <= 0xDBFF) { // High surrogate (could change last hex to 0xDB7F to treat high private surrogates as single characters)
 var hi = code;
 if (str.length === 1) {
 return code; // This is just a high surrogate with no following low surrogate, so we return its value;
 // we could also throw an error as it is not a complete character, but someone may want to know
 }
 var low = str.charCodeAt(1);
 if (!low) {

 }
 return ((hi - 0xD800) * 0x400) + (low - 0xDC00) + 0x10000;
 }
 if (0xDC00 <= code && code <= 0xDFFF) { // Low surrogate
 return code; // This is just a low surrogate with no preceding high surrogate, so we return its value;
// we could also throw an error as it is not a complete character, but someone may want to know
 }
 return code;
}

function substr (str, start, len) {
 // Returns part of a string
 //
 // version: 909.322
 // discuss at: http://phpjs.org/functions/substr
 // + original by: Martijn Wieringa
 // + bugfixed by: T.Wild
 // + tweaked by: Onno Marsman
 // + revised by: Theriault
 // + improved by: Brett Zamir (http://brett-zamir.me)
 // % note 1: Handles rare Unicode characters if 'unicode.semantics' ini (PHP6) is set to 'on'
 // * example 1: substr('abcdef', 0, -1);
 // * returns 1: 'abcde'
 // * example 2: substr(2, 0, -6);
 // * returns 2: false
 // * example 3: ini_set('unicode.semantics', 'on');
 // * example 3: substr('auD801uDC00', 0, -1);
 // * returns 3: 'a'
 // * example 4: ini_set('unicode.semantics', 'on');
 // * example 4: substr('auD801uDC00', 0, 2);
 // * returns 4: 'auD801uDC00'
 // * example 5: ini_set('unicode.semantics', 'on');
 // * example 5: substr('auD801uDC00', -1, 1);
 // * returns 5: 'uD801uDC00'
 // * example 6: ini_set('unicode.semantics', 'on');
 // * example 6: substr('auD801uDC00zuD801uDC00', -3, 2);
 // * returns 6: 'uD801uDC00z'
 // * example 7: ini_set('unicode.semantics', 'on');
 // * example 7: substr('auD801uDC00zuD801uDC00', -3, -1)
 // * returns 7: 'uD801uDC00z'
// Add: (?) Use unicode.runtime_encoding (e.g., with string wrapped in "binary" or "Binary" class) to
// allow access of binary (see file_get_contents()) by: charCodeAt(x) & 0xFF (see https://developer.mozilla.org/En/Using_XMLHttpRequest) or require conversion first?

 var i = 0, allBMP = true, es = 0, el = 0, se = 0, ret = '';
 str += '';
 var end =

str.length;

  // BEGIN REDUNDANT
  this.php_js = this.php_js || {};
 this.php_js.ini = this.php_js.ini || {};
  // END REDUNDANT
  switch(
 (this.php_js.ini['unicode.semantics'] &&
 this.php_js.ini['unicode.semantics'].local_value.toLowerCase())) {
  case 'on': // Full-blown Unicode including non-Basic-Multilingual-Plane characters
  // strlen()
  for (i=0; i < str.length; i++) {
  if (/[uD800-uDBFF]/.test(str.charAt(i)) && /[uDC00-uDFFF]/.test(str.charAt(i+1))) {
  allBMP = false;
 break;
  }
  }

  if (!allBMP) {
  if (start < 0) {
  for (i = end - 1, es = (start += end); i >= es; i--) {
  if (/[uDC00-uDFFF]/.test(str.charAt(i)) && /[uD800-uDBFF]/.test(str.charAt(i-1))) {
  start--;
  es--;
  }
  }
 }
  else {
  var surrogatePairs = /[uD800-uDBFF][uDC00-uDFFF]/g;
  while ((surrogatePairs.exec(str)) != null) {
  var li = surrogatePairs.lastIndex;
  if (li - 2 < start) {
 start++;
  }
  else {
  break;
  }
  }
  }

  if (start >= end || start < 0) {
  return false;
  }
  if (len < 0) {
  for (i = end - 1, el = (end += len); i >= el; i--) {
  if (/[uDC00-uDFFF]/.test(str.charAt(i)) && /[uD800-uDBFF]/.test(str.charAt(i-1))) {
  end--;
  el--;
  }
  }
 if (start > end) {
  return false;
  }
  return str.slice(start, end);
  }
  else {
  se = start + len;
 for (i = start; i < se; i++) {
  ret += str.charAt(i);
 if (/[uD800-uDBFF]/.test(str.charAt(i)) && /[uDC00-uDFFF]/.test(str.charAt(i+1))) {
 se++; // Go one further, since one of the "characters" is part of a surrogate pair
 }
  }
  return ret;
  }
 break;
  }
  // Fall-through
  case 'off': // assumes there are no non-BMP characters;
  // if there may be such characters, then it is best to turn it on (critical in true XHTML/XML)
  default:
  if (start < 0) {
 start += end;
  }
  end = typeof len === 'undefined' ? end : (len < 0 ? len + end : len + start);
  // PHP returns false if start does not fall within the string.
  // PHP returns false if the calculated end comes before the calculated start.
 // PHP returns an empty string if start and end are the same.
  // Otherwise, PHP returns the portion of the string from start to end.
  return start >= str.length || start < 0 || start > end ? !1 : str.slice(start, end);
  }
  return undefined; // Please Netbeans
}

function array_push (inputArr) {
  // Pushes elements onto the end of the array
  //
  // version: 1008.1718
  // discuss at: http://phpjs.org/functions/array_push
  // + original by: Kevin van Zonneveld (http://kevin.vanzonneveld.net)
  // + improved by: Brett Zamir (http://brett-zamir.me)
  // % note 1: Note also that IE retains information about property position even
  // % note 1: after being supposedly deleted, so if you delete properties and then
  // % note 1: add back properties with the same keys (including numeric) that had
  // % note 1: been deleted, the order will be as before; thus, this function is not
  // % note 1: really recommended with associative arrays (objects) in IE environments
  // * example 1: array_push(['kevin','van'], 'zonneveld');
  // * returns 1: 3
  var i=0, pr = '', argv = arguments, argc = argv.length,
  allDigits = /^d$/, size = 0, highestIdx = 0,

len = 0;
 if (inputArr.hasOwnProperty('length')) {
 for (i=1; i < argc; i++){
 inputArr[inputArr.length] = argv[i];
 }
 return inputArr.length;
 }

 // Associative (object)
 for (pr in inputArr) {
 if (inputArr.hasOwnProperty(pr)) {
 ++len;
 if (pr.search(allDigits) !== -1) {
 size = parseInt(pr, 10);
 highestIdx = size > highestIdx ? size : highestIdx;
 }
 }
 }
 for (i=1; i < argc; i++){
 inputArr[++highestIdx] = argv[i];
 }
 return len + i - 1;
}

function count (mixed_var, mode) {
 // Count the number of elements in a variable (usually an array)
 //
 // version: 1008.1718
 // discuss at: http://phpjs.org/functions/count
 // + original by: Kevin van Zonneveld (http://kevin.vanzonneveld.net)
 // + input by: Waldo Malqui Silva
 // + bugfixed by: Soren Hansen
 // + input by: merabi
 // + improved by: Brett Zamir (http://brett-zamir.me)
 // * example 1: count([[0,0],[0,-4]], 'COUNT_RECURSIVE');
 // * returns 1: 6
 // * example 2: count({'one' : [1,2,3,4,5]}, 'COUNT_RECURSIVE');
 // * returns 2: 6
 var key, cnt = 0;

 if (mixed_var === null){
 return 0;
 } else if (mixed_var.constructor !== Array && mixed_var.constructor !== Object){
 return 1;
 }

 if (mode === 'COUNT_RECURSIVE') {
 mode = 1;
 }
 if (mode != 1) {
 mode = 0;
 }

 for (key in mixed_var){
 if (mixed_var.hasOwnProperty(key)) {
 cnt++;
 if (mode==1 && mixed_var[key] && (mixed_var[key].constructor === Array || mixed_var[key].constructor === Object)){
 cnt += this.count(mixed_var[key], 1);
 }
 }
 }

 return cnt;
}

function trim (str, charlist) {
 // Strips whitespace from the beginning and end of a string
 //
 // version: 1008.1718
 // discuss at: http://phpjs.org/functions/trim
 // + original by: Kevin van Zonneveld (http://kevin.vanzonneveld.net)
 // + improved by: mdsjack (http://www.mdsjack.bo.it)
 // + improved by: Alexander Ermolaev (http://snippets.dzone.com/user/AlexanderErmolaev)
 // + input by: Erkekjetter
 // + improved by: Kevin van Zonneveld (http://kevin.vanzonneveld.net)
 // + input by: DxGx
 // + improved by: Steven Levithan (http://blog.stevenlevithan.com)
 // + tweaked by: Jack
 // + bugfixed by: Onno Marsman
 // * example 1: trim(' Kevin van Zonneveld ');
 // * returns 1: 'Kevin van Zonneveld'
 // * example 2: trim('Hello World', 'Hdle');
 // * returns 2: 'o Wor'
 // * example 3: trim(16, 1);
 // * returns 3: 6
 var whitespace, l = 0, i = 0;
 str += '';

 if (!charlist) {
 // default list
 whitespace = " nrtfx0bxa0u2000u2001u2002u2003u2004u2005u2006u2007u2008u2009u200au200bu2028u2029u3000";
 } else {
 // preg_quote custom list
 charlist += '';
 whitespace = charlist.replace(/([[]().?/*{}+$^:])/g, '$1');
 }

 l = str.length;
 for (i = 0; i < l; i++) {
 if (whitespace.indexOf(str.charAt(i)) === -1) {
 str = str.substring(i);
 break;
 }
 }

 l = str.length;
 for (i = l - 1; i >= 0; i--) {
 if (whitespace.indexOf(str.charAt(i)) === -1) {
 str = str.substring(0, i + 1);
 break;
 }
 }

 return whitespace.indexOf(str.charAt(0)) === -1 ? str : '';
}

function floor (value) {
 // Returns the next lowest integer value from the number
 //
 // version: 1008.1718
 // discuss at: http://phpjs.org/functions/floor
 // + original by: Onno Marsman
 // * example 1: floor(8723321.4);
 // * returns 1: 8723321

/>
  return Math.floor(value);
}

//**********
//**********
//**********
//**********

/*Function for generating keys. Return array where
$array[0] -> modulo N
$array[1] -> public key E
$array[2] -> private key D
Public key pair is N and E
Private key pair is N and D
*/

/* modulus function */
function mo (g, l) {
  return g - (l * floor (g/l));
}

function rsa_encrypt (m, e, n) {
  var asci = new Array();
  for (i=0; i<strlen(m); i+=3) {
  var tmpasci="1";
  for (h=0; h<3; h++) {
  if (i+h <strlen(m)) {
  var tmpstr = ord (substr(m, i+h, 1)) - 30;
  if (strlen(tmpstr) < 2) {
  tmpstr ="0".tmpstr;
  }
  } else {
  break;
  }
  tmpasci = tmpasci + tmpstr;
  }
  array_push(asci, tmpasci+"1");
 }

  //Each number is then encrypted using the RSA formula: block ^E mod N
 var coded = "";
 var temp = count(asci);
  for (k=0; k< count(asci); k++) {
 var resultmod = powmod(asci[k], e, n);
  coded = coded + resultmod + " ";
  }
  return trim(coded);
}

function powmod (base, exp, modulus) {
  accum = 1;
  i = 0;
  basepow2 = base;
  while ((exp >> i)>0) {
  if (((exp >> i) & 1) == 1) {
  accum = mo((accum * basepow2) , modulus);
  }
  basepow2 = mo((basepow2 * basepow2) , modulus);
  i++;
  }
  return accum;
}{/codecitation}{/mooblock} Je ne pr�ends pas que mes solutions sont parfaites mais c'est mieux qu'aucune protection. <h2><u>4) Et pour finir</u>�</h2> Vous n'avez plus qu'�adapter tout ce code �votre sauce. La 1�e modification utile est d'ajouter une cha�e au mot de passe avant de le hasher pour qu'au final, celui-ci soit unique dans votre base de donn�s m�e si l'utilisateur utilise le m�e ailleurs. Ensuite, pour que votre site soit vraiment s�uris� ne vous connectez pas en FTP pour envoyer vos fichiers sur le serveur. En effet, �ce moment l� le mot de passe transit en clair sur le r�eau. Si votre h�ergeur vous fourni un acc� SSH, utilisez le SFTP pour s�uriser votre transfert.� Vous pouvez aussi utiliser un certificat SSL (https) si vous en avez un.