Encryption (encode/decode, mycrypt)
Data security (HTTPS)
XSS and SQL injection

One-way functions
Crypt(string, salt) <-- The salt parameter is optional. However, crypt() creates a weak password without the salt.
Encrypts only 1st 8 chars of string. Can alter output by 2 character salt (which is prepended onto result)
Md5(string) <-- this has been compromised. Does not encrypt. Returns a message digest. A 32-bit string returned. See https://en.wikipedia.org/wiki/MD5
Eg $pwdDigest=md5(trim($passwd));
Cannot retrieve a password; reset it. Md5(pwd) and compare

Alternatives:
bcrypt, sha512crypt or scrypt.
inPHP:
password_hash() uses a strong hash, generates a strong salt
string password_hash ( string $password , integer $algo [, array $options ] )
PASSWORD_DEFAULT - Use the bcrypt algorithm.
PASSWORD_BCRYPT - Use the CRYPT_BLOWFISH algorithm. Using the PASSWORD_BCRYPT as the algorithm, will result in the password parameter being truncated to a maximum length of 72 characters.
eg echo password_hash("mypassword", PASSWORD_DEFAULT)."\n";
output: > 60 characters (use a db column of size 255 characters ).

PHP: Use two way function: mcrypt()
MySQL: use encode(str, key) and decode(str, key) functions
Salt md5() to improve security against brute force attack & dictionary attacks
Eg md5('2@string#3');
Md5("$salt1$string$salt2");

SQL injection:
$user = $_POST['user'];
$pass = $_POST['pass'];
$query = "SELECT * FROM users WHERE user='$user' AND pass='$pass'";
Scenario:
User= admin' #
Pass= [blank]
SELECT * FROM users WHERE user='admin' #' AND pass=''

Scenario:
$query = "DELETE FROM users WHERE user='$user' AND pass='$pass'";
User= anything' OR 1=1 #
DELETE FROM users WHERE user='anything' OR 1=1 #' AND pass=''

Use mysqli_real_escape_string()
Do not rely on PHP's built-in magic quotes,
meant to replace or preface quotes ' or " with \
Use mysqli_real_escape_string ()
If magic_quotes turned on, using mysql_real_escape_string will corrupt string (will double escape)
if (get_magic_quotes_gpc())
$string = stripslashes($string); return mysql_real_escape_string($string);
You can only use mysql_real_escape_string () where there is an active mysql connection string

XSS --> XSS: cross site scripting
To protect user's privacy (not safety of your site)
3rd party may try to steal cookie information (Can glean user-passwd info); trick users to download malicious code
Can occur if users are allowed to type in html or javacript code into a form, which your site subsequently displays (eg forum type entries)
Use htmlentities () to parse input
Converts & to &amp, < to &lt etc

eg
Given user input as
<script src='http://x.com/hack.js'> </script><script>hack();</script>
Is converted to
<script src='http://x.com/hack.js'> </script><script>hack();</script>

Note: strip_tags($str) removes html tags
Source: Learning PHP, Mysql and Javascript p.249

Reverse with html_entity_decode()

Very good protection:
Before passing $user and $pass into a query,
1. use mysqli_real_escape_string($user)
//check if magic_quotes, then stripslashes().
2. pass result to htmlentities()
Then will be safe to use in normal query.
Eg $query = "SELECT * FROM users WHERE user='$user' AND pass='$pass'";

May also strip_tags() as step 3. to improve security

Prevent XSS and SQL inject
<?php
function mysqli_entities_fix_string($string)
{
return htmlentities(mysql_fix_string($string));
}
function mysql_fix_string($string)

```
{
if (get_magic_quotes_gpc()) $string = stripslashes($string);
return mysqli_real_escape_string($string); }
?>

Useful fxn to clean form input
function cleanInput($var)
{
$var = stripslashes($var);
$var = htmlentities($var);
$var = strip_tags($var);
return $var;
}
```

Using a previous lab,
Save data into a table. Ask for a password. Encrypt the password field.

Full LAB:
Use code that takes care of

SQL injection attack at log in (use placeholders)
Protect against SQL injection attack
Takes care of XSS at the time of submitting forum entries.
Encrypt (with salting) all passwords
At registration, ask for Real Name, but store this in an encrypted form in the DB. (you will need to display unencrypted name when viewing data)