

Playing Safe

Vancouver PHP Conference

January 22nd, 2004. Vancouver, Canada

Derick Rethans <derick@php.net>

?

o

"People who are willing to rely on the government to keep them safe are pretty much standing on Darwin's mat, pounding on the door, screaming, 'Take me, take me!'"

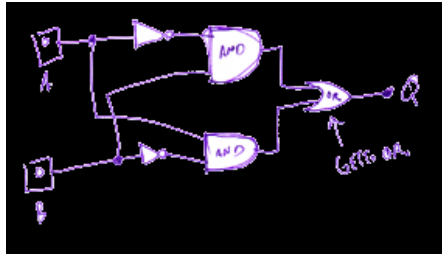
Carl Jacobs, Alt.Sysadmin.Recovery

- o Rotational (ceasar)
- o Transposition (xor)
- o Symmetric (DES, Blowfish, AES)
- o Asymmetric (RSA, Diffie Hellman)
- o Hashes

Plain text:	H	e	l	l	o	W	o	r	l	d	
ASCII value of each character:	72	101	108	108	111	32	87	111	114	108	100
Key:	K	e	y		K	e	y		K	e	y
ASCII value of each character:	75	101	121	32	75	101	121	32	75	101	121
Crypted text:	␣	Ê	å	␣	◌	␣	Đ	␣	œ	Ñ	Ý
ASCII value of each character:	147	202	229	140	186	133	208	143	189	209	221

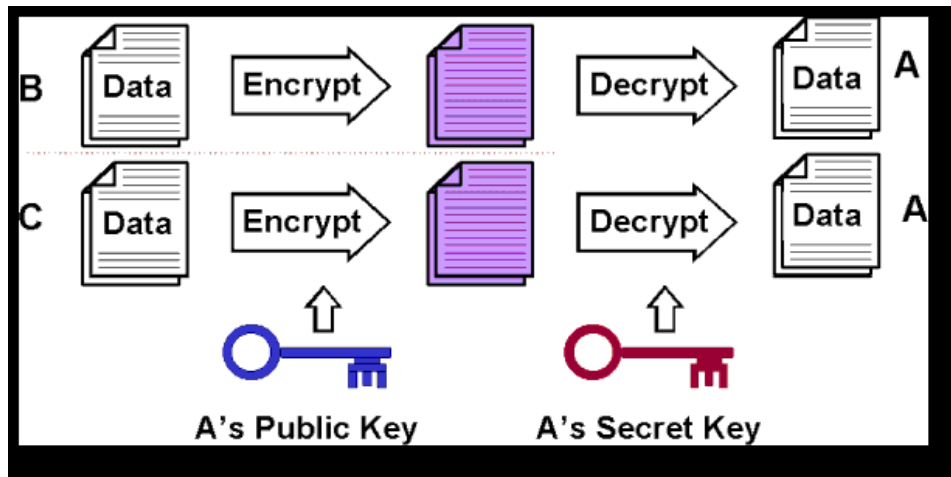
- o Rotate characters depending on key
- o Easy to crack unless key is same size as plain text
- o But we always have `str_rot13()` :-)

- o XOR plain text with key
- o Easy to crack with small key
- o Impossible to crack when $\text{strlen}(\text{plain}) == \text{strlen}(\text{key})$



"Perl - The only language that looks the same before and after RSA encryption."

Keith Bostic



- o Key to encrypt is different as the key to decrypt
- o ElGamal, Diffie Hellman, RSA
- o Used for signatures and key distribution



- o Not reversible
- o PHP: sha1(), md5(); mhash: GOST, HAVAL



- o Used for in signatures and validation

```
<?php
    if (isset($_COOKIE['stamp'])) {
        if ($_COOKIE['stamp'] == sha1($_COOKIE['data'] . 'SECRETKEY')) {
            echo "Validated!\n";
        } else {
            echo "<blink>Not validated</blink>";
        }
    } else {
        $_COOKIE['data'] = '00110100011';
        $_COOKIE['stamp'] = sha1($_COOKIE['data'] . 'SECRETKEY');
    }
?>
```

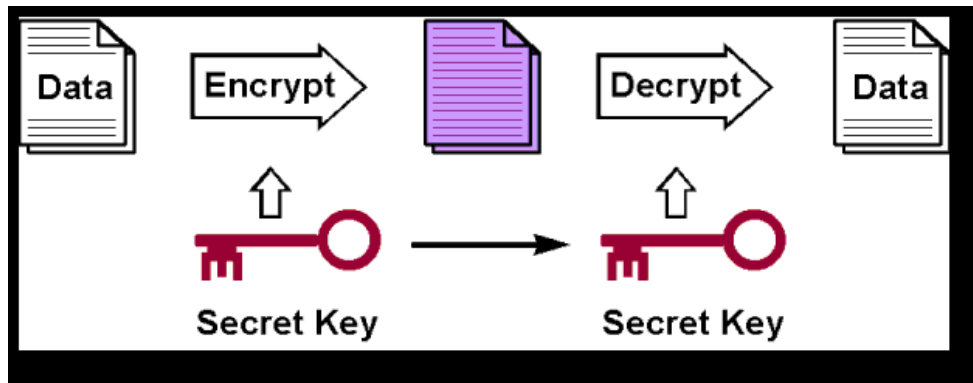
- o Easy validation of cookie data
- o But not the 'best' way

```
<?php
require_once 'Crypt/HMAC.php';

$hash = new Crypt_HMAC('SECRETKEY', 'sha1');

if (isset($_COOKIE['stamp'])) {
    if ($_COOKIE['stamp'] == $hash->hash($_COOKIE['data'])) {
        echo "Validated!\n";
    } else {
        echo "<blink>Not validated</blink>";
    }
} else {
    $_COOKIE['data'] = '00110100011';
    $_COOKIE['stamp'] = $hash->hash($_COOKIE['data']);
}
?>
```

- o pear install Crypt_HMAC
- o RFC 2104 implementation
- o MD5 and SHA1 supported



- o Key to encrypt is the same as the key to decrypt
- o mcrypt: DES, Blowfish, Rijndael (AES)
- o Used for session keys and in secure environment
- o Not used for key distribution

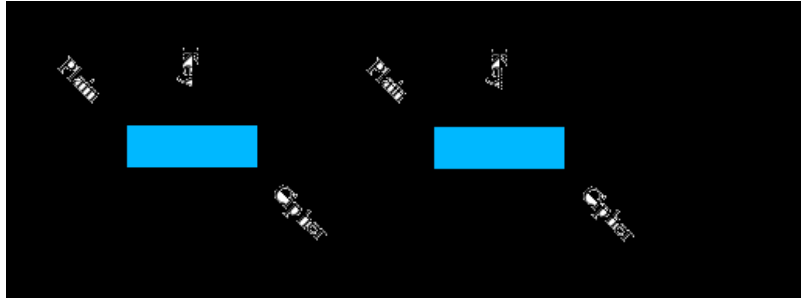
I've found one, and only one practical application for "13375p33k"

Secure passwords

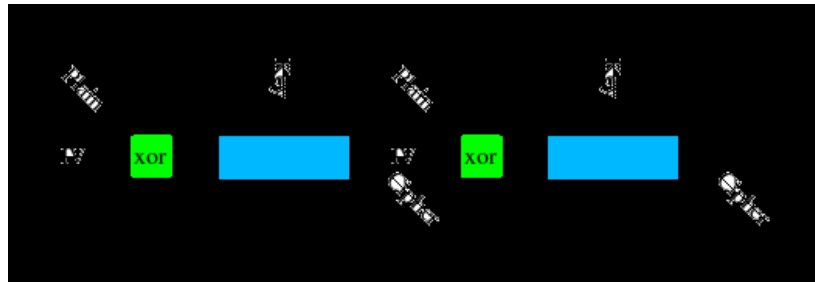
Jan Lehnardt

Different encryption modes:

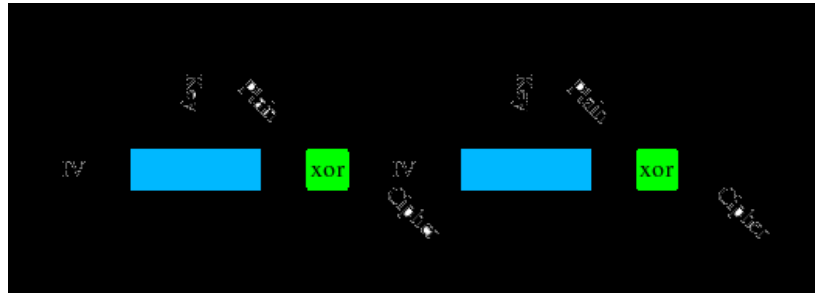
- o ECB (Electronic CodeBook)
each block encrypted independently
- o CBC (Cipher Block Chaining)
text block is xor'ed with previous encrypted block
- o CFB (Cipher FeedBack)
text is xor'ed with the encrypted output of the IV; cipher output is new IV for next block
- o OFB (Output FeedBack)
text is xor'ed with the encrypted output of the IV; encrypted output is new IV for next block



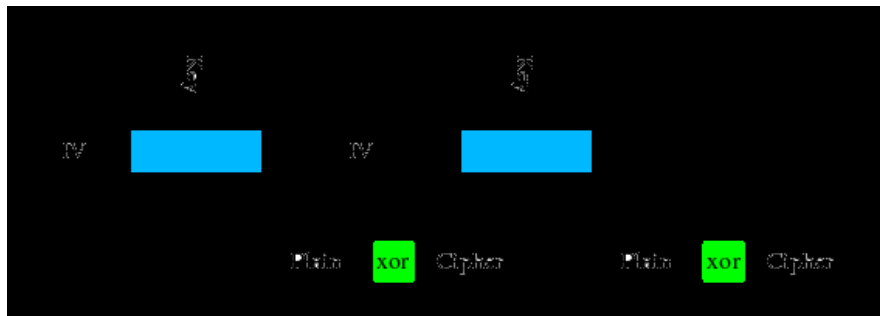
ECB: each block encrypted independently



CBC: text block is xor'ed with previous encrypted block



CFB: text is xor'ed with the encrypted output of the IV; cipher output is new IV for next block



OFB: text is xor'ed with the encrypted output of the IV; encrypted output is new IV for next block

"most security failures in its area of interest are due to failures in implementation, not failure in algorithms or protocols"

The NSA

"most security failures in its area of interest are due to failures in implementation, not failure in algorithms or protocols"

The NSA

Do [blink]not[/blink] store the key on the same machine as the encrypted data

Do never store passwords, unless hashed

```

<?php
error_reporting(E_ALL);
$td = @mcrypt_module_open('rijndael-256', '', 'cfb', '');

if ($td) {
    echo mcrypt_enc_get_algorithms_name($td). "/";
    echo mcrypt_enc_get_modes_name($td). "<br />";
    echo "IV Size:      ". mcrypt_enc_get_iv_size($td). "; ";
    echo "Block Size:   ". mcrypt_enc_get_block_size($td). "; ";
    echo "Key Size:     ". mcrypt_enc_get_key_size($td). "<br />";
    echo "Key Sizes:    ". join(", ", mcrypt_enc_get_supported_key_sizes($td)).
"<br />";

    echo "Is Block A/M: ". (mcrypt_enc_is_block_algorithm_mode($td) ? "Y" :
"N"). "; ";
    echo "Is Block A:   ". (mcrypt_enc_is_block_algorithm($td) ? "Y" : "N"). ";
";
    echo "Is Block M:   ". (mcrypt_enc_is_block_mode($td) ? "Y" : "N"). "<br />
";
    echo "Selftest:    ". (mcrypt_enc_self_test($td) ? "N" : "Y");
}
?>

```

Output:

```

Rijndael-256/CFBIV Size:    32; Block Size: 32; Key Size:    32Key Sizes: 16,
24, 32Is Block A/M: Y; Is Block A:    Y; Is Block M:    N Selftest:    Y

```

```
<?php
    $td = @mcrypt_module_open('rijndael-256', '', 'cfb', '');

    if ($td) {
        / Get key size and IV size /
        $iv_size = mcrypt_enc_get_iv_size($td);
        $key_size = mcrypt_enc_get_key_size($td);

        / Create IV and 'key' /
        $iv = mcrypt_create_iv($iv_size);
        $key = substr(shal('very secret key'), 0, $key_size);

        / Init encryption module /
        mcrypt_generic_init($td, $key, $iv);
    }
?>
```

```
<?php
    $td = @mdecrypt_module_open('rijndael-256', '', 'cfb', '');

    if ($td) {
        $iv_size = mdecrypt_enc_get_iv_size($td);
        $key_size = mdecrypt_enc_get_key_size($td);
        $iv = mdecrypt_create_iv($iv_size);
        $key = substr(sha1('very secret key'), 0, $key_size);

        / Init encryption module /
        mdecrypt_generic_init($td, $key, $iv);

        / Encrypt text /
        $crypt_text = mdecrypt_generic($td, "very important data");
        echo $crypt_text;

        / Clean up cipher /
        mdecrypt_generic_deinit($td);
        mdecrypt_module_close($td);
    }
?>
```

Output:

î68`½bĐçó÷ËJŒEDù-Qk

- o Encrypted data is binary 'crap'
- o Length and trailing zeros are important
- o Use `base64_encode()` or `binhex()`

"I have a string encrypted with mcrypt and encoded in base64. I unbase64 this string and I decrypt it using mcrypt. I looks like I got the exact same string but when I compare it with == it is not the same."

```
<?php
    $td = @mcrypt_module_open('rijndael-128', '', 'ecb', '');

    $iv_size = mcrypt_enc_get_iv_size($td);
    $key_size = mcrypt_enc_get_key_size($td);
    $key = substr(sha1('very secret key'), 0, $key_size);

    / Init encryption module /
    @mcrypt_generic_init($td, $key, '');

    / Decrypt text /
    $text = mdecrypt_generic($td, $crypt_text);
    echo $text;
?>
```

- o If you use a block method, mcrypt will pad with \0s.
- o So store the original length of the text too.

"Typical uses for something like libmccrypt is to encrypt a plaintext for decryption at some later date. All that should be required for the decryption is the key that was used during encryption. In particular, the values in the IV array cannot be expected to be the same during encryption and decryption.

[...] If however, I modify the test program as per the patch below, so that the IV array is different for encryption and decryption, then many of the tests fail."

The IV for encryption and decryption should always be the same.

But it does not need to be secret, only unique and random.

Wouldn't this be much nicer?

```
<?php
    $td = new Mcrypt('tripledes', 'cfb');

    echo $td->name;

    echo $td->is_block_algorithm;
    echo $td->block_size;

    echo $td->iv_size;
    echo $td->key_size;

    $iv = $td->generate_iv();
    / or /
    $iv = mcrypt::creat_iv($size);
    $td->set_iv($iv);

    $td->set_key('secret_key', MCRYPT_HASH); / or MCRYPT_PAD /

    $td->init(); / optional, only required for multiple encryptions /
    $crypt = $td->encrypt('secret text');

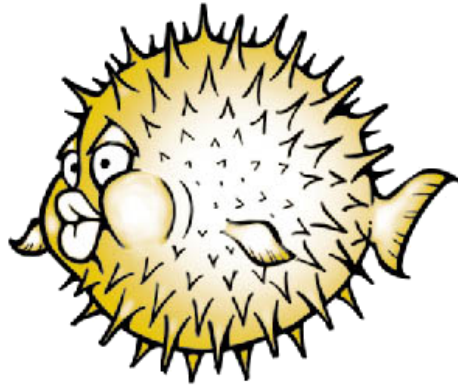
    $td = NULL;

?>
```

A false sense of security is worse than being unsure.

grc.com

?



These Slides: <http://pres.derickrethans.nl/mcrypt-vancouver>

PHP: <http://www.php.net>

Crypto Intro: <http://>

Questions?: <mailto:derick@php.net>

Index

Questions	2
Why use encryption?	3
Quote	4
Cipher groups	5
Rotational ciphers	6
Transposition ciphers	7
Quote	8
Asymmetric ciphers	9
Hash algorithms	10
Hash Example	11
Crypt_MHAC example	12
Symmetric ciphers	13
Quote	14
Modes	15
ECB and CBC	16
CFB and OFB	17
Quote	18
Example	19
Mcrypt Modules	20
Mcrypt Init	21
Encrypting	22
Q1: Storing crypttext	23
Q2: Bug #26733	24
Q3: IV troubles	25
Supported modes and ciphers	26
Future?	27
Quote	28
Questions	29
Resources	30