



The PHP Company

Strong Cryptography in PHP

by Enrico Zimuel (enrico@zend.com)

Senior Software Engineer
Zend Framework Core Team
Zend Technologies Ltd

About me



- Software Engineer since 1996
- Enjoying PHP since 1999
- PHP Engineer at Zend Technologies, in the Zend Framework Team
- Author of two books on security and cryptography (in italian)
- B.Sc. (Hons) in Computer Science and Economics
- Blog on Programming in PHP:
<http://www.zimuel.it/blog>

Note about the source code

- The source code reported in this presentation is only for teaching purpose. The PHP code reported must be considered only as parts of a complete system.

DON'T USE IT IN A PRODUCTION ENVIRONMENT!

- In order to implement a secure software using cryptography you need much more information than the one reported in this presentation.
- Our advice is to involve always *cryptography engineers* if you need to implement a secure software, using cryptography, in a production environment.

Strong cryptography

Strong cryptography is the usage of cryptographic systems or components that are considered highly resistant to cryptanalytic attacks

A metric of security?

- How we can say that an encryption algorithm is considered highly resistant to cryptanalytic attacks?
- It's difficult to answer to this question. We don't have a simple metric of security.
- We have to consider:
 - ▶ Brute forcing attacks
 - ▶ Theoretical attacks
 - ▶ Implementation attacks

A metric of security? (2)

- **Brute forcing attacks**

- ▶ Space key is 2^n , where n is the byte size of the key. If $n=128$, $K= 3,4 * 10^{38}$

- **Theoretical attacks**

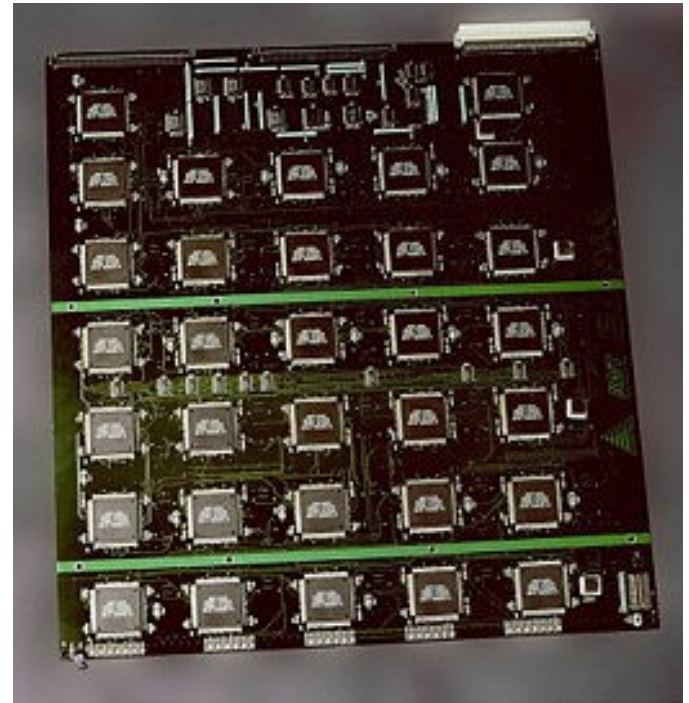
- ▶ Break the encryption with mathematical attacks.
- ▶ Reduce the space key, for AES 256bit, an attack can reduce K to $2^{99.5}$

- **Implementation attacks**

- ▶ Based on the implementation

Is DES still secure?

- **EFF DES cracker** ("Deep Crack") is a computer built by the Electronic Frontier Foundation (EFF) in 1998 to perform a brute force search of DES cipher's key space
- The Deep Crack decrypted a 56 bit DES cryptogram in only 56 hours of work. **In the 1998!**



Examples of strong cryptography

- **Strong:**
 - ▶ PGP, OpenPGP, GnuPG
 - ▶ AES, Blowfish, Twofish
 - ▶ RSA (key \geq 2048 bit)
- **Not strong:**
 - ▶ DES
 - ▶ WEP (Wired Equivalent Privacy)
 - ▶ SSL 40 bit, international version
 - ▶ All the classic ciphers (Enigma, ROT13, Vigenère, etc)

Not only encryption

- Strong cryptography is not only related to encryption.
- It can also be used to describe **hashing** and **unique identifier**
- In this usage, the term means **difficult to guess**

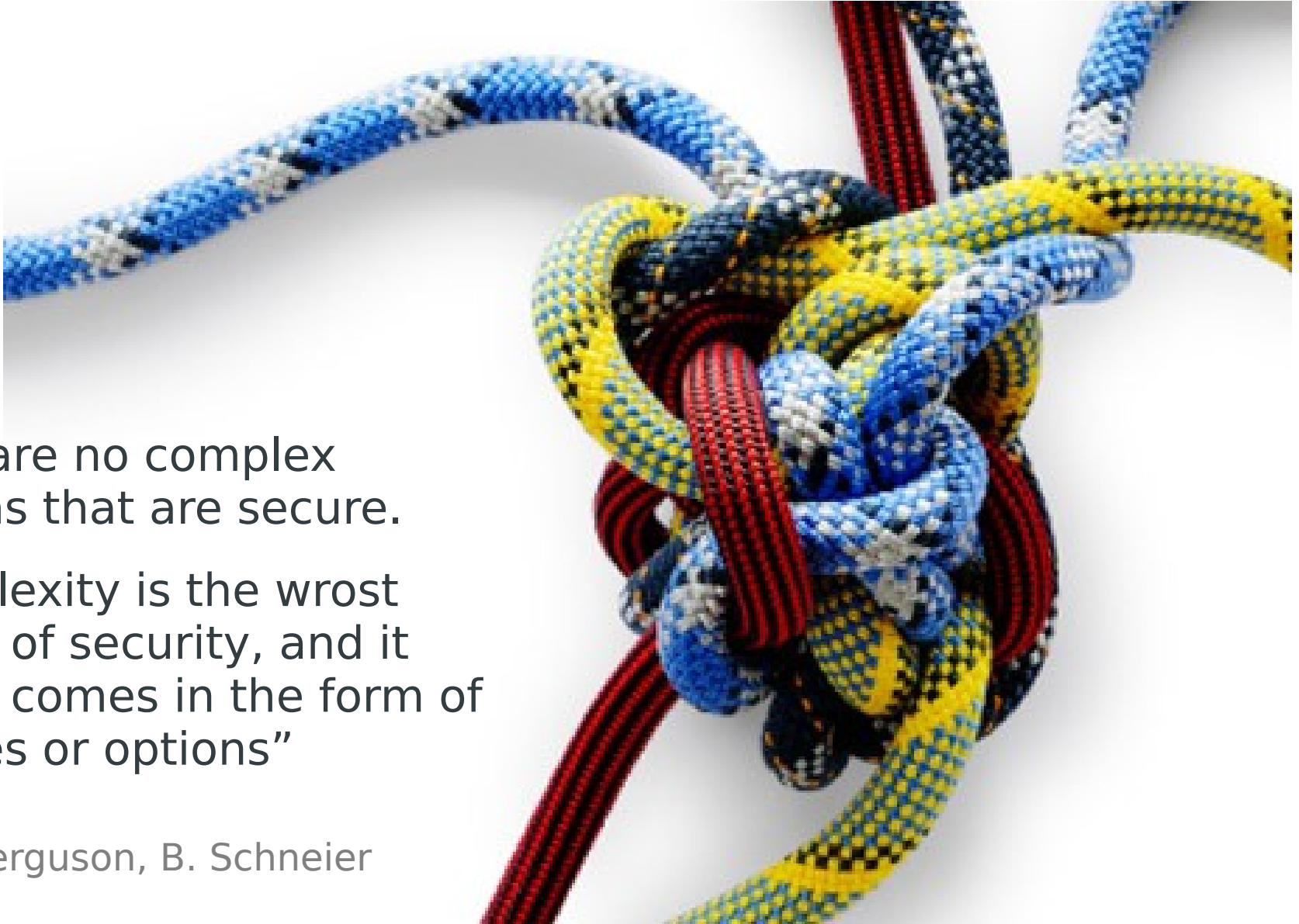
Cryptography vs. Security

- Cryptography doesn't means security
- Encryption is not enough
- “Security is a process, not a product”

Bruce Schneier



Complexity vs. Security



- There are no complex systems that are secure.
- “Complexity is the worst enemy of security, and it always comes in the form of features or options”

N. Ferguson, B. Schneier

Cryptography in PHP

Cryptography in PHP

- crypt()
- Mcrypt
- Hash
- OpenSSL

crypt()

- One-way string hashing
- Support strong cryptography
 - ▶ bcrypt, sha-256, sha-512
- PHP 5.3.0 – bcrypt support
- PHP 5.3.2 – sha-256/512

Mcrypt

- Mcrypt is an interface to the **mcrypt library**, which supports a wide variety of block algorithms
- It support the following encryption algorithms:
 - ▶ 3DES, ARCFOUR, BLOWFISH, CAST, DES, ENIGMA, GOST, IDEA (non-free), LOKI97, MARS, PANAMA, RIJNDAEL, RC2, RC4, RC6, SAFER, SERPENT, SKIPJACK, TEAN, TWOFISH, WAKE, XTEA

Hash

- The Hash extension requires no external libraries and is enabled by default as of PHP 5.1.2.
- This extension replace the old mhash extension
- With this extension you can generate hash values or HMAC (Hash-based Message Authentication Code)
- Supported hash algorithms: MD4, MD5, SHA1, SHA256, SHA384, SHA512, RIPEMD, RIPEMD, WHIRLPOOL, GOST, TIGER, HAVAL, etc

OpenSSL

- The OpenSSL extension uses the functions of the **OpenSSL project** for generation and verification of signatures and for sealing (encrypting) and opening (decrypting) data
- You can use OpenSSL to protect data using public key cryptography with the RSA algorithm.



Use standard algorithms



- AES (RIJNDAEL), FIST 197 standard since 2001
- BLOWFISH
- TWOFISH
- SHA-256, 384, 512
- RSA

Examples and Best practices

How build a key?

- **New key: pseudo-random**
 - ▶ Use `openssl_random_pseudo_bytes()`
(PHP 5.3.0)
 - ▶ DO NOT USE `rand()` or `mt_rand()`
- **Don't use the user password as a key**
 - ▶ Hash with a salt + iteration (stretching)
 - ▶ To prevent dictionary based attacks
Try <http://md5.rednoize.com/>

Pseudo random key

```
function pseudoRandomKey($size) {
    if (function_exists('openssl_random_pseudo_bytes')) {
        $rnd = openssl_random_pseudo_bytes($size, $strong);
        if($strong === TRUE)
            return $rnd;
    }
    $sha=''; $rnd='';
    for ($i=0;$i<$size;$i++) {
        $sha= hash('sha256', $sha.mt_rand());
        $char= mt_rand(0,62);
        $rnd.= chr(hexdec($sha[$char].$sha[$char+1]));
    }
    return $rnd;
}
```

Build a key from a user password

- Hash the password with a random salt + stretching

```
$salt= pseudoRandomKey(128);  
$hash='';  
for ($i=0;$i<HASH_CYCLE_LIMIT;$i++) {  
    $hash= hash('sha512',$hash.$salt.$password);  
}
```

- HASH_CYCLE_LIMIT depends on the CPU speed, should take between 200 to 1000 ms
 - ▶ Intel Core 2 at 2.1Ghz, LIMIT \approx 20'000 (500 ms)

Safely store a password (bcrypt)

- Hash the password using bcrypt (PHP 5.3.0)

```
$salt = substr(str_replace('+', '.',  
                    base64_encode($salt)), 0, 22);  
$hash= crypt($password, '$2a$'$cost.'$'.$salt);
```

- where **\$cost** is the base-2 logarithm of the iteration count (Blowfish). Must be in range 04-31.
- How to check if a password is valid or not:

```
$hash==crypt($password, $hash)
```

Safely store a password (sha256)

```
function securePassword ($password, $salt) {  
    $hash='';  
    for ($i=0;$i<SHA_LIMIT_LOOP;$i++) {  
        $hash= hash('sha256', $hash.$salt.$password);  
    }  
    return base64_encode($salt). '$' . $hash;  
}
```

- For instance, **\$password**= 'thisIsTheSecretPassword' and **\$salt**= 'hsjYeg/bxn()%3jdhsGHq0'

aHNqWWVnL2J4bigpJTNqZGhzR0hxMA==
\$a9c810e9c722af719adabcf5
0db8a0b4cd0d14e07eddbb43e5f47bde620a3c13

Green= salt, Red= encrypted password

Check if a password is valid

```
function validPassword ($password, $hash) {  
    $delimiter= strpos($hash, '$');  
    if ($delimiter===false) {  
        return false;  
    }  
    $salt= base64_decode(substr($hash,0,$delimiter));  
    $tHash='';  
    for ($i=0;$i<SHA_LIMIT_LOOP;$i++) {  
        $tHash= hash('sha256',$tHash.$salt.$password);  
    }  
    return (base64_encode($salt).'$'.$tHash==$hash);  
}
```

Example: encrypt with AES (CBC mode)

```
$ivSize= mcrypt_get_iv_size(MCRYPT_RIJNDAEL_128,  
                             MCRYPT_MODE_CBC);  
$iv= mcrypt_create_iv($ivSize, MCRYPT_RAND);  
  
$key= pseudoRandomKey(16); // 128 bit  
  
$encrypted= mcrypt_encrypt(  
    MCRYPT_RIJNDAEL_128,  
    $key,  
    $data,  
    MCRYPT_MODE_CBC,  
    $iv  
);
```

Example: decrypt with AES (CBC mode)

```
$data= mcrypt_decrypt(  
    MCRYPT_RIJNDAEL_128,  
    $key ,  
    $encrypted ,  
    MCRYPT_MODE_CBC ,  
    $iv  
);
```

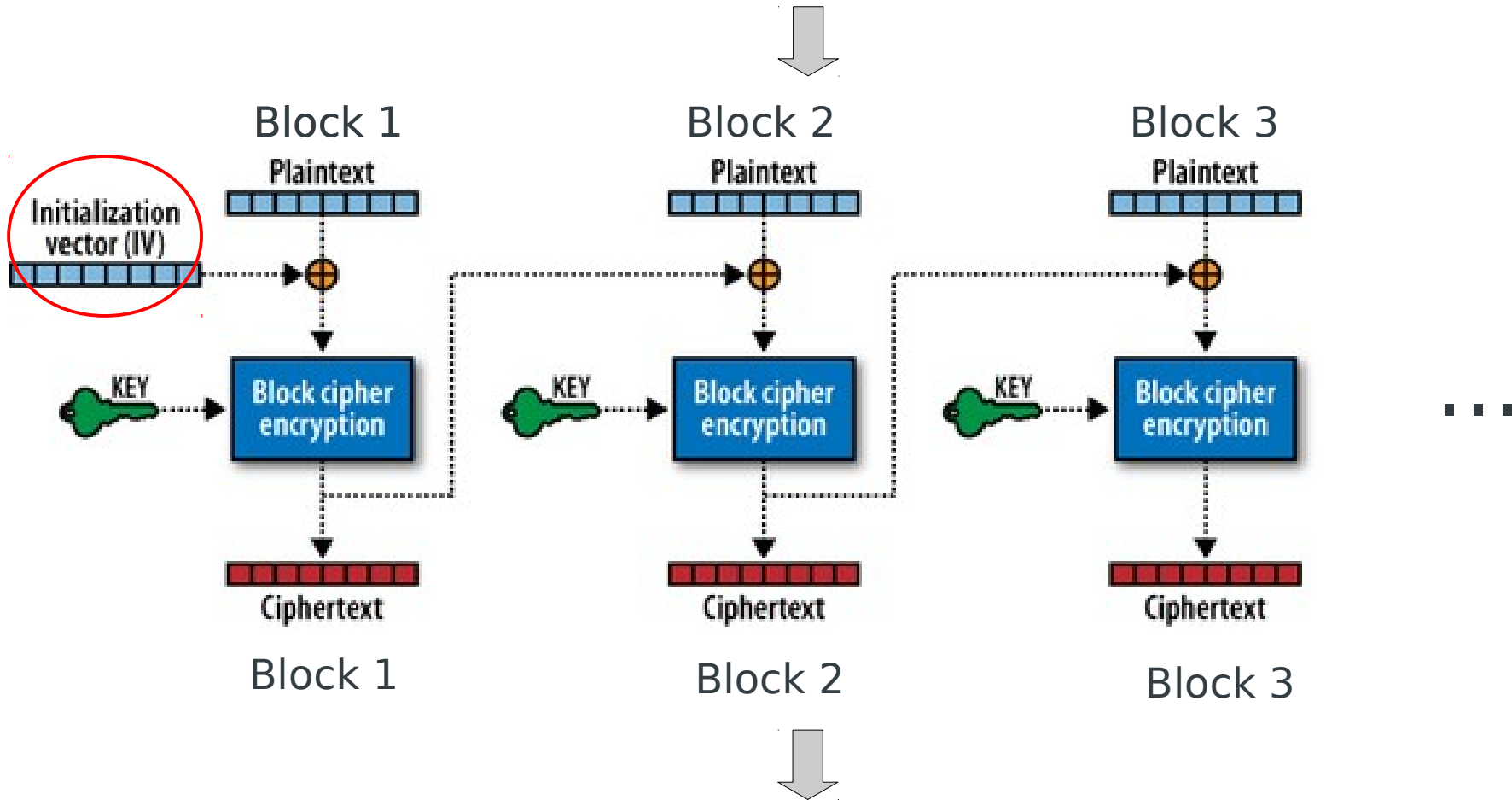
- Question: what is the \$iv?

Initialization vector (IV)

- In cryptography, an **Initialization Vector** (IV) is a fixed-size input to a cryptographic primitive that is typically required to be random or pseudorandom
- **The IV is not a secret**, you can send it in plaintext
- Usually IV is stored before the encrypted message

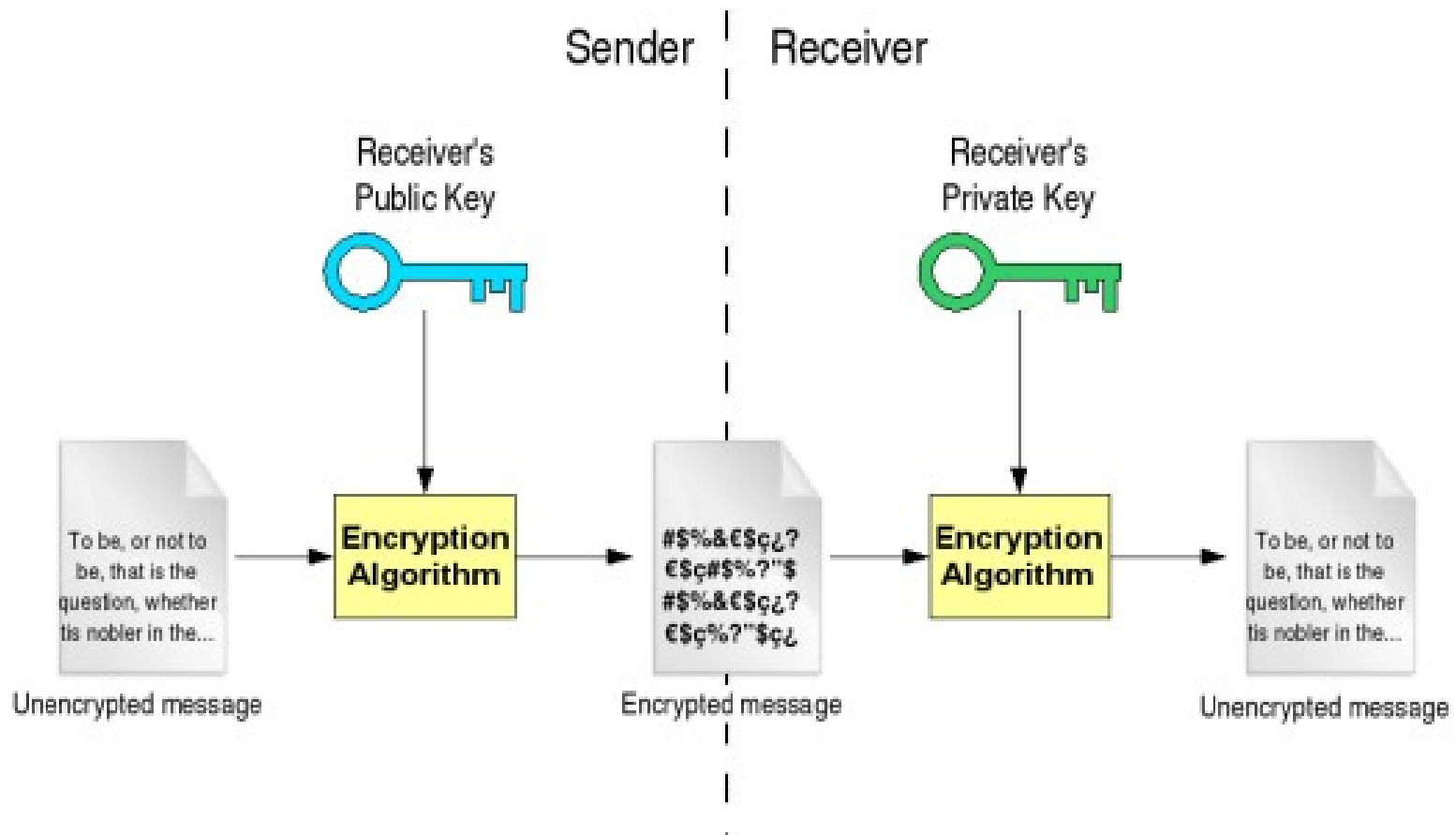
CBC needs IV

The Plaintext (input) is divided into blocks

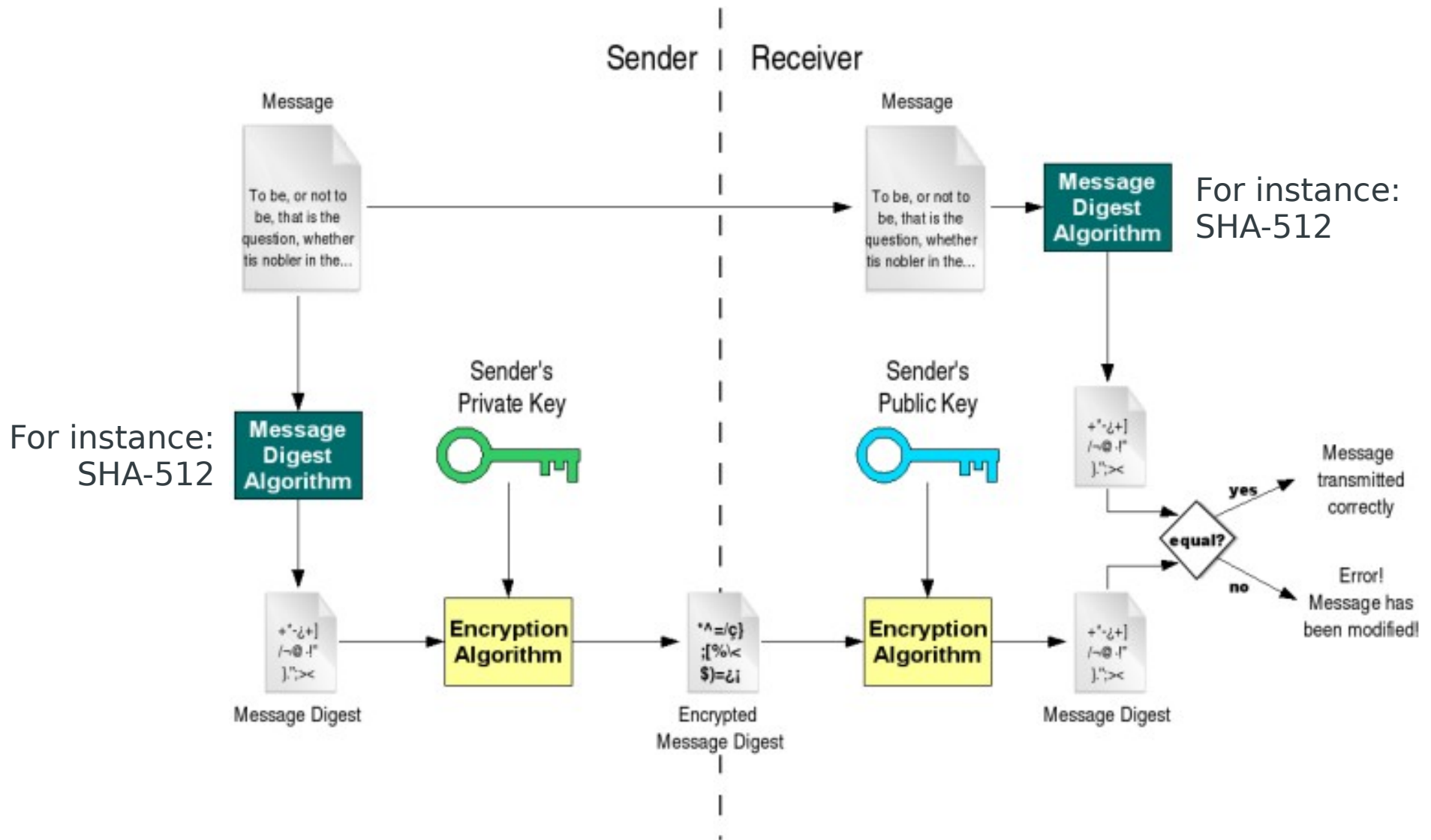


The Ciphertext (output) is the concatenation of the cipher-blocks

Public-key cryptography



Not only encryption: authentication



Generate public and private keys

```
$privateKey = openssl_pkey_new(array(  
    'private_key_bits' => 2048,  
    'private_key_type' => OPENSSL_KEYTYPE_RSA  
));  
  
openssl_pkey_export_to_file($privateKey,  
    '/path/to/privatekey', $passphrase);  
  
$keyDetails = openssl_pkey_get_details($privateKey);  
file_put_contents('/path/to/publickey',  
    $keyDetails['key']);
```

Encrypt/decrypt using RSA

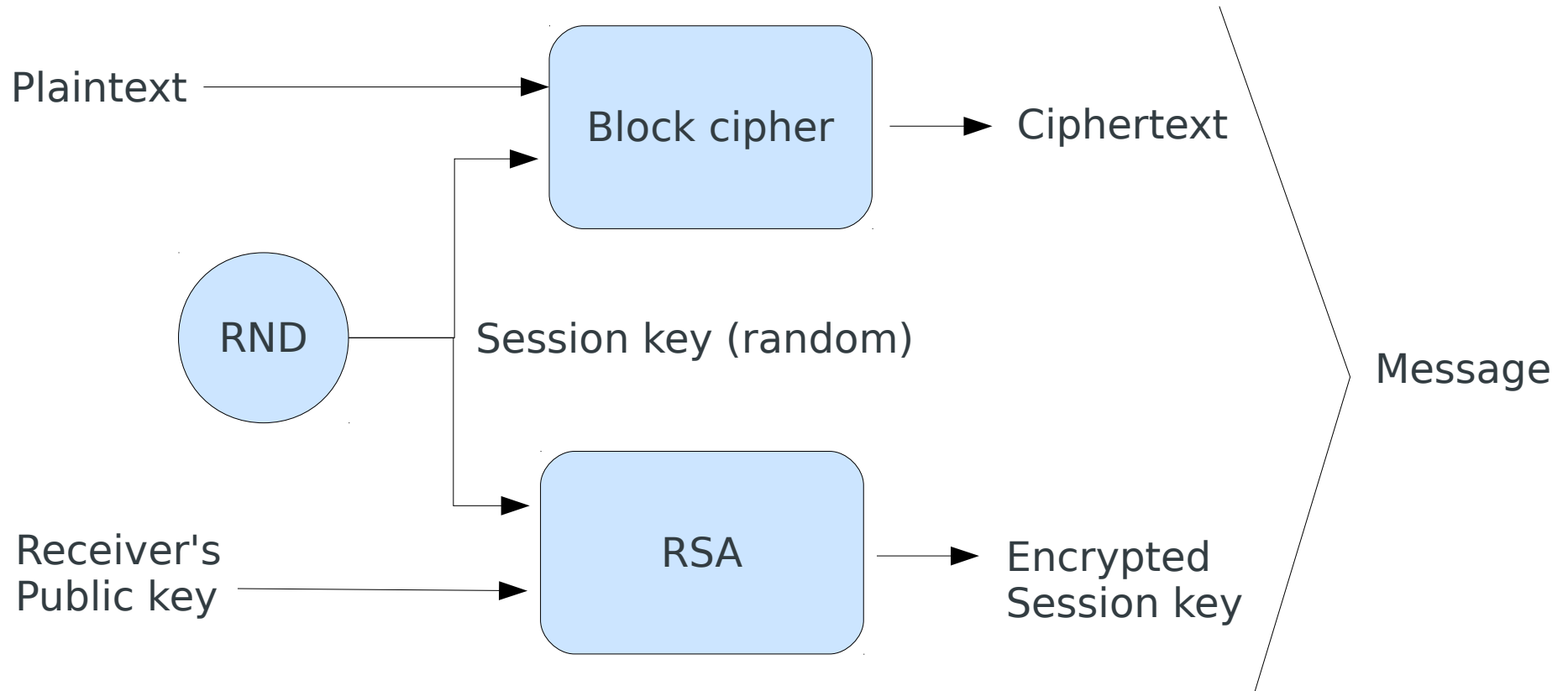
```
// encrypt
$pubKey= openssl_pkey_get_public('pubkeyfile');
openssl_public_encrypt($plaintext,
                      $encrypted,
                      $pubKey);

// decrypt
$privateKey= openssl_pkey_get_private('privkeyfile',
                                     $passphrase);
openssl_private_decrypt($encrypted,
                      $plaintext,
                      $privateKey);
```

Public-key cryptography to encrypt data

- In general, the public-key cryptography is not used directly to encrypt data.
- Public-key cryptography is computationally heavy, that means the algorithms are very slow!
- We can use hybrid systems:
 - ▶ **public key + block chipers**

Example of hybrid system: PGP



Some resources (books and papers)

- N.Ferguson, B.Schneier, and T. Kohno, “Cryptography Engineering” John Wiley & Sons, 2010
- N. Ferguson, B.Schneier, “Practical Cryptography” Wiley, 2003
- C. Snyder, M.Southwell, “Pro PHP Security”, Apress, 2005
- Chris Chiflett, “Essential PHP Security”, O'Reilly, 2006
- Norman D. Jorstad, Landgrave T. Smith, Jr. “Cryptographic Algorithm Metrics”, Institute for Defense Analyses, 1997
- Z. Benenson, U. Kühn, S.Lucks, “Cryptographic Attack Metrics” Dependability Metrics 2005

Some resources (web)

- PHP Cryptography Extensions,
<http://www.php.net/manual/en/refs.crypto.php>
- crypt(), <http://nl.php.net/manual/en/function.crypt.php>
- Cracking MD5 and SHA-1, <http://md5.rednoize.com/>
- A Guide to Cryptography in PHP,
<http://www.devx.com/webdev/Article/37821>
- How To Safely Store A Password,
<http://codahale.com/how-to-safely-store-a-password/>
- Zimuel's blog, [Strong Cryptography in PHP](#)
- Zimuel's blog, [Encrypt Session data in PHP](#)

Thank you!

- **Twitter:**

- ▶ **@ezimuel**

- **Blog:**

- ▶ **<http://www.zimuel.it/blog>**

- **GitHub:**

- ▶ **<https://github.com/ezimuel>**

© Copyright of the pictures used in this presentation:

- TotalFail.blogspot.com
- **Windows Azure: Building a Secure Backup System**
- Borja Sotomayor **“Globus Toolkit 4 Programmer's Tutorial”**

Questions?

