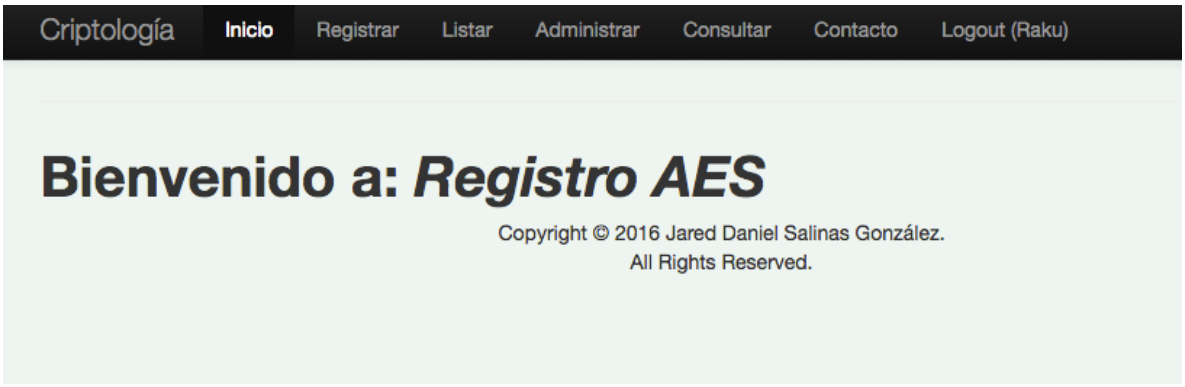
Proyecto 1: Criptografía.

Jared Daniel Salinas González.
David Rodríguez Robles.



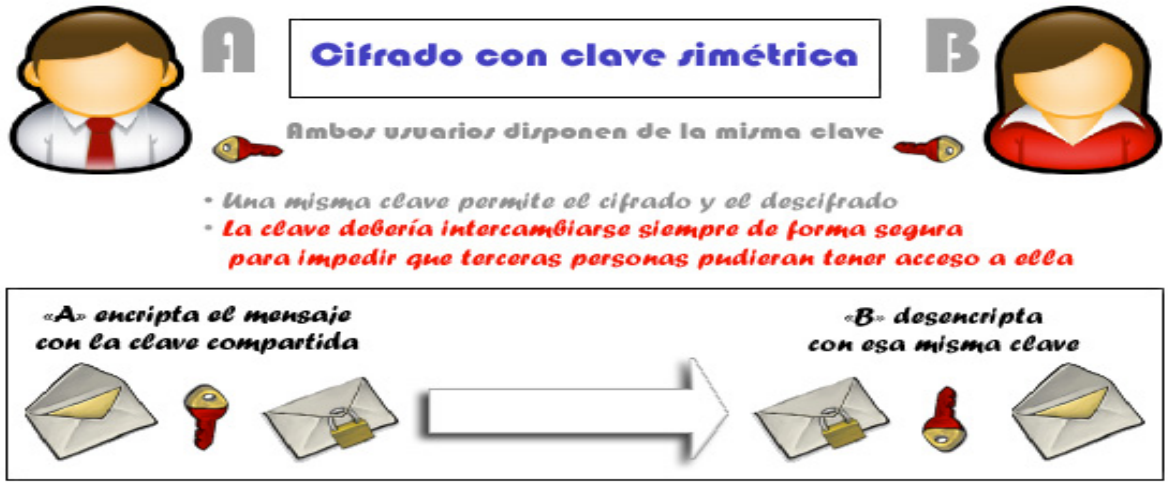
Mi proyecto que realicé fue el primero, la cual es el cifrado y descifrado de un log-in a partir de la criptografía simétrica para eso use las siguientes herramientas:

Nuestro objetivo era experimentar con las tecnologías que hemos estado usando actualmente y gracias a que mucha información pasa por internet considere que podría demostrar como una página web creada con php usando la base de datos de MySQL puede implementarse la criptografía simétrica usando la herramienta más interesante y más usada según en lo que se discutía en clase, el AES.

Como apoyo use el framework Yii y para las vistas la herramienta Bootstrap, gracias a esas herramientas el desarrollo fue más rápido y pude enfocarme en el problema del cifrado de datos.

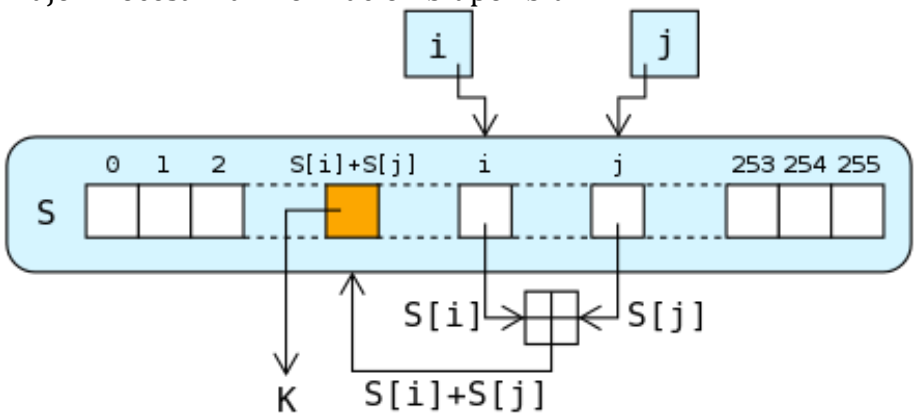
Marco Teórico:

La criptografía simétrica usa solo una clave privada, la cual se usa una llave con la cadena a cifrar para producir un mensaje cifrado.

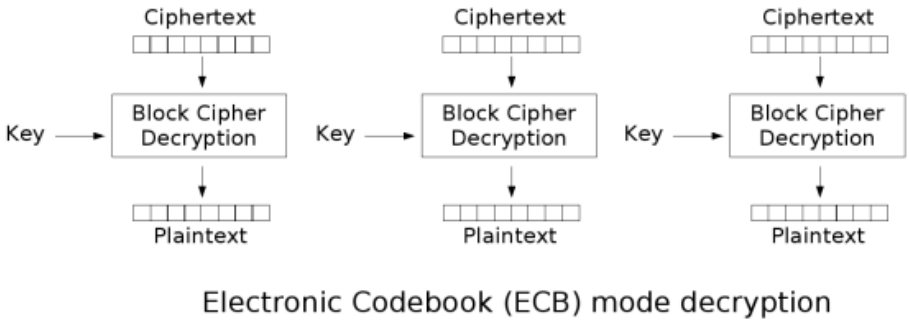


Como se conoce hay dos tipos de soluciones a la criptografía simétrica:

- Flujo: Procesan la información bit por bit.



- Bloques: Procesan grupos de bits.



Electronic Codebook (ECB) mode decryption

El AES el algoritmo que use para descifrar pertenece a la familia de los algoritmos de cifrado por bloques y es actualmente el estándar Americano de cifrado de datos, gracias a su popularidad, la seguridad que puede ofrecer, la documentación existente facilidad fue el escogido a usar.

Como Funciona:

Lo primero que hice al desarrollar el sistema fue escoger mis herramientas de codificación, como había mencionado quería hacer algo web, así que busque implementaciones del algoritmos AES en internet que usaran php y el la base de datos MySQL, pero gracias a la popularidad del algoritmo, dicha base de datos ya cuenta con una simple sentencia para cifrar como descifrar los datos la cual es la siguiente:

```
AES_ENCRYPT (cadena,clave)
AES_DECRYPT (cadena,clave)
```

Dentro del código se encuentra como en la siguiente sentencia:

```php
public function insertarAes($m, $pass) {
    $q = "insert into identificacion"
            . "(nick,"
            . "password,"
            . "nombre,"
            . "apellido_pat,"
            . "apellido_mat,"
            . "calle_numero,"
            . "colonia,"
            . "municipio,"
            . "estado,"
            . "pais,"
            . "telefono,"
            . "tarjeta_credito,"
            . "id) VALUES"
            . "(AES_ENCRYPT('$m->nick', '$pass'),"
            . " AES_ENCRYPT('$m->password', '$pass'),"
            . " AES_ENCRYPT('$m->nombre', '$pass'),"
            . " AES_ENCRYPT('$m->apellido_pat', '$pass'),"
            . " AES_ENCRYPT('$m->apellido_mat', '$pass'),"
            . " AES_ENCRYPT('$m->calle_numero', '$pass'),"
            . " AES_ENCRYPT('$m->colonia', '$pass'),"
            . " AES_ENCRYPT('$m->municipio', '$pass'),"
            . " AES_ENCRYPT('$m->estado', '$pass'),"
            . " AES_ENCRYPT('$m->pais', '$pass'),"
            . " AES_ENCRYPT('$m->telefono' , '$pass'),"
            . " AES_ENCRYPT('$m->tarjeta_credito' , '$pass'),"
            . " '$m->id')";
    $cmd = Yii::app()->db->createCommand($q);
    return $cmd->execute();
```

Para descifrar se encuentra de la siguiente manera:

```php
public function descifrarAes($name, $pass) {
    $q = "select "
            . " AES_DECRYPT(nick, '$pass'),"
            . " AES_DECRYPT(password, '$pass'),"
            . " AES_DECRYPT(nombre, '$pass'),"
            . " AES_DECRYPT(apellido_pat, '$pass'),"
            . " AES_DECRYPT(apellido_mat, '$pass'),"
            . " AES_DECRYPT(calle_numero, '$pass'),"
            . " AES_DECRYPT(colonia, '$pass'),"
            . " AES_DECRYPT(municipio, '$pass'),"
            . " AES_DECRYPT(estado, '$pass'),"
            . " AES_DECRYPT(pais, '$pass'),"
            . " AES_DECRYPT(telefono, '$pass'),"
            . " AES_DECRYPT(tarjeta_credito, '$pass') "
            . " from identificacion where AES_DECRYPT(nick, '$pass
    $cmd = Yii::app()->db->createCommand($q);
    return $res = $cmd->query();
}
```

En la aplicación el resultado se puede visualizar de la siguiente manera:

Ver Identificacion #1

Nick	¯ÃŸ{ø]»=m›—=
Password	‰ @p“†âoÍ CÃÀÜ
Nombre	ÂTÜ˜Ù{xËw¾}Zþá
Apellido Pat	>ÆÇœýYÇ>ž˜ãtã ¾
Apellido Mat	Ò²[ª]—¡søâ¯L×
Calle Numero	$óŠNàMöž]òñfÙä_~æ«Á9ï¨ü½Pc
Colonia	XE¸â(GÀ-`ê«ÛÛVŒ
Municipio	ñÇÀÿçàé¶`œfÃ+
Estado	ñÇÀÿçàé¶`œfÃ+
Pais	²Ÿ*ø¦¿~u¥Y–P
Telefono	“w¼_P™‰Ç‹K¯i4
Tarjeta Credito	ËLß-á‡E?‡²©îRm5Å9
ID	1

Datos Descifrados

Nick : Raku

Password : oracle

Nombre : Jared

aP. Pat : Salinas

aP. Mat : Gonzalez

Conclusiones:

Para mi es sorprendente ver que es muy sencillo implementar un algoritmo que provee mucha seguridad, aunque existen los SQL Injections, es un algoritmo que puede ser muy potente, pero descubrí ciertas dificultades al usarlo en diferentes tipos de cotejamientos de bases de datos como el UTF8 o el latin1_swedish_ci el cual resultaba con problemas de cifrar los datos, pues no en muchos cotejamientos y configuraciones de la base de datos puedan o no almacenar dichos valores.

Es bueno cifrar los datos, pero considero que deben en ser en mensajes o campos muy específicos, ya que considero que se puede considerar un problema después de tratar de cifrar varios o todos los campos.