Rijndael Encryption
Advanced Encryption Standard (AES)

Library Functions

aes_block_decrypt_CBC

Description:	Decrypt one or more blocks of cipher data according to the CBC encryption mode.
Location:	`crypt.lib`
Prototype:	`rijndael.h`
Syntax:	void **aes_block_decrypt_CBC** (cr_keyStruct *keyStr, cr_block initVec, cr_block *cipherText, int nBlocks, cr_block *plainText)
Parameter:	keyStr = pointer to a static cr_keyStruct filled by aes_key_init
	initVec = pointer to one cr_block containing the initial vector
	cipherText = pointer to one or more cr_block's with data to be decrypted
	nBlocks = number of cr_block's to be encrypted
	plainText = pointer to one or more cr_block's that will receive the decrypted data
Return value:	None

aes_block_encrypt_CBC

Description:	Encrypt one or more blocks of plain data according to the CBC encryption mode
Location:	`crypt.lib`
Prototype:	`rijndael.h`
Syntax:	void **aes_block_encrypt_CBC**(cr_keyStruct *keyStr, cr_block initVec, cr_block *plainText, int nBlocks, cr_block *cipherText)
Parameter:	keyStr = pointer to a static cr_keyStruct filled by aes_key_init
	initVec = pointer to one cr_block containing the initial vector
	plainText = pointer to one or more cr_block's with data to be encrypted
	nBlocks = number of cr_block's to be encrypted
	cipherText = pointer to one or more cr_block's that will receive the encrypted data.
Return value:	None

aes_byte_encrypt

Description:	Encrypt or decrypt one or more bytes of data according to the CFB mode.
Location:	`crypt.lib`
Prototype:	`rijndael.h`
Syntax:	void **aes_byte_encrypt**(cr_keyStruct *keyStr, cr_block *initVec, cr_block *input, int nBytes, cr_block *output, BYTE *left, enum cr_mode mode)
Parameter:	keyStr = pointer to a static cr_keyStruct filled by aes_key_init initVec = pointer to one cr_block containing the initial vector input = pointer to one or more bytes of data to be encrypted or decrypted nBytes = number of bytes to be encrypted or decrypted output = pointer to one or more cr_block's that will receive the calculated data. left = pointer to a global variable holding the amount of bytes left over from last time a cipher block was not completely used for encryption. The first ever call to this function *left needs to be 0. mode = encryption (1) or decryption (2) selection.
Return value:	None

aes_cipher_init

Description:	Initializes the data tables used by the encryption module. Called only once after boot.
Location:	`crypt.lib`
Prototype:	`rijndael.h`
Syntax:	void **aes_cipher_init** ()
Parameter:	None
Return value:	None

aes_key_init

Description:	Calculate roundkeys from encryption key. Called once for every stream with a separate encryption key.
Location:	`crypt.lib`
Prototype:	`rijndael.h`
Syntax:	void **aes_key_init**(cr_keyStruct *keyStr, BYTE *key, int keyLen)
Parameter:	keyStr = pointer to a static cr_keyStruct. Will be filled with roundkeys and other encryption info. key = pointer to a byte array containing the encryption key. keyLen = length of encryption key in bits. Only 128, 192 and 256 are allowed.
Return value:	None

Macros

CR_ BLOCK_DECRYPT_CBC

Description:	Decrypt one or more blocks of cipher data according to the CBC encryption mode.
Location:	`crypt.lib`
Prototype:	`security.h`
Syntax:	void **CR_BLOCK_DECRYPT_CBC**(cr_keyStruct *keyStr, cr_block initVec, cr_block *cipherText, int nBlocks, cr_block *plainText)
Parameter:	keyStr = pointer to a static cr_keyStruct filled by CR_KEY_PREP
	initVec = pointer to one cr_block containing the initial vector
	cipherText = pointer to one or more cr_block's with data to be decrypted
	nBlocks = number of cr_block's to be encrypted
	plainText = pointer to one or more cr_block's that will receive the decrypted data
Return value:	None

CR_BLOCK_ENCRYPT_CBC

Description:	Encrypt one or more blocks of plain data according to the CBC encryption mode
Location:	`crypt.lib`
Prototype:	`security.h`
Syntax:	void **CR_BLOCK_ENCRYPT_CBC**(cr_keyStruct *keyStr, cr_block initVec, cr_block *plainText, int nBlocks, cr_block *cipherText)
Parameter:	keyStr = pointer to a static cr_keyStruct filled by CR_KEY_PREP
	initVec = pointer to one cr_block containing the initial vector
	plainText = pointer to one or more cr_block's with data to be encrypted
	nBlocks = number of cr_block's to be encrypted
	cipherText = pointer to one or more cr_block's that will receive the encrypted data.
Return value:	None

CR_ BYTE_DECRYPT_CFB

Description:	Decrypt one or more bytes of plain data according to the CFB encryption mode.
Location:	`crypt.lib`
Prototype:	`security.h`
Syntax:	void **CR_BYTE_DECRYPT_CFB**(cr_keyStruct *keyStr, cr_block *initVec, cr_block *cipherText, int nBytes, cr_block *plainText, BYTE *left)
Parameter:	keyStr = pointer to a static cr_keyStruct filled by CR_KEY_PREP
	initVec = pointer to one cr_block containing the initial vector
	cipherText = pointer to one or more bytes of data to be decrypted
	nBytes = number of bytes to be encrypted
	plainText = pointer to one or more cr_block's that will receive the decrypted data.
	left = pointer to a global variable holding the amount of bytes left over from last time a cipher block was not completely used for encryption. The first ever call to this function *left needs to be 0.
Return value:	None

CR_ BYTE_ENCRYPT_CFB

Description:	Encrypt one or more bytes of plain data according to the CFB encryption mode.
Location:	`crypt.lib`
Prototype:	`security.h`
Syntax:	void **CR_BYTE_ENCRYPT_CFB**(cr_keyStruct *keyStr, cr_block *initVec, cr_block *plainText, int nBytes, cr_block *cipherText, BYTE *left)
Parameter:	keyStr = pointer to a static cr_keyStruct filled by CR_KEY_PREP
	initVec = pointer to one cr_block containing the initial vector
	plainText = pointer to one or more bytes of data to be encrypted
	nBytes = number of bytes to be encrypted
	cipherText = pointer to one or more cr_block's that will receive the encrypted data.
	left = pointer to a global variable holding the amount of bytes left over from last time a cipher block was not completely used for encryption. The first ever call to this function *left needs to be 0.
Return value:	None

CR_CIPHER_INIT

Description:	Initializes the data tables used by the encryption module. Called only once after boot.
Location:	`crypt.lib`
Prototype:	`security.h`
Syntax:	void **CR_CIPHER_INIT** ()
Parameter:	None
Return value:	None

CR_KEY_INIT

Description:	Calculate roundkeys from encryption key. Called once for every stream with a separate encryption key.
Location:	`crypt.lib`
Prototype:	`security.h`
Syntax:	void **CR_KEY_INIT**(cr_keyStruct *keyStr, BYTE *key, int keyLen)
Parameter:	keyStr = pointer to a static cr_keyStruct. Will be filled with roundkeys and other encryption info.
	key = pointer to a byte array containing the encryption key.
	keyLen = length of encryption key in bits. Only 128, 192 and 256 are allowed.
Return value:	None