Encryption / Decryption files

Overview

This code provides functions for encrypting and decrypting files using AES-256 CTR encryption. It supports multi-threaded encryption and decryption of files in a specified directory. The encryption is performed using RSA public-key cryptography, where the AES key is encrypted with the recipient's public key before being stored in the encrypted file.

The code is organized into several functions and helper methods. Here's a brief summary of each component:

- `aes_256_ctr_encrypt_decrypt`: This function performs AES-256 CTR encryption or decryption on a given ciphertext using the provided key and nonce.
- `gen_aes_key`: This function generates a random AES key of the specified size.
- `inc_counter`: This helper function increments the given nonce, used in AES-CTR mode, by 1.
- `get_dst_file_path`: This function returns the destination file path for the encrypted file based on the source file path.
- `FileEncryptionDecryptionError`: This is an enum that represents possible errors that can occur during file encryption or decryption.
- `encrypt_decrypt_file`: This function encrypts or decrypts a file based on the specified parameters. It uses AES-CTR encryption for the file data and RSA encryption for the AES key.
- `multi_threaded_encrypt_decrypt_files`: This function performs multi-threaded encryption or decryption on multiple files within a directory. It distributes the files among multiple threads for parallel processing.

Usage

To use this code, you need to import the necessary dependencies:

```
use aes::{
    cipher::{NewCipher, StreamCipher},
    Aes256Ctr,
};
use rand::{distributions::Uniform, thread_rng, Rng};
```

```
use rsa::{
    pkcs1::{DecodeRsaPrivateKey, DecodeRsaPublicKey},
    Pkcs1v15Encrypt, RsaPrivateKey, RsaPublicKey,
};
use walkdir::WalkDir;

use std::{
    fmt::Error as FmtError,
    fs::{remove_file, File, OpenOptions},
    io::{BufReader, Read, Seek, SeekFrom, Write},
    path::{Path, PathBuf},
    sync::mpsc::{channel, Sender},
    thread::{self, JoinHandle},
};

use crate::c2::api::C2API;
```

Note: Some dependencies may need to be added to your project's `Cargo.toml` file.

To encrypt or decrypt a file, you can use the `encrypt_decrypt_file` function:

```
pub fn encrypt_decrypt_file(
    file_src_path: &str,
    private_public_key: String,
    is_encryption: u8,
) -> Result<usize, FileEncryptionDecryptionError>
```

- `file_src_path`: The path to the source file to be encrypted or decrypted.
- `private_public_key`: The RSA private or public key used for encryption or decryption.
- `is_encryption`: A flag indicating whether encryption or decryption should be performed. Set it to `1` for encryption and `0` for decryption.

The function returns the total number of bytes read from the file if successful, or an error of type `FileEncryptionDecryptionError` if an error occurs.

To perform multi-threaded encryption or decryption on multiple files within a directory, you can use the `multi_threaded_encrypt_decrypt_files` function:

```
pub fn multi_threaded_encrypt_decrypt_files(
    directory: &str,
    private_public_key: String,
    user_id: String,
```

```
    is_encryption: u8,
)
```

- `directory`: The directory containing the files to be encrypted or decrypted.
- `private_public_key`: The RSA private or public key used for encryption or decryption.
- `user_id`: An identifier for the user or recipient of the encrypted files.
- `is_encryption`: A flag indicating whether encryption or decryption should be performed. Set it to `1` for encryption and `0` for decryption.

This function performs multi-threaded processing on the files in the specified directory, distributing the workload among multiple threads for faster execution.

Limitations

- The code assumes the use of AES-256 CTR mode for encryption and decryption. Other modes or key sizes are not supported.
- The RSA encryption and decryption operations use the PKCS#1 v1.5 padding scheme. Other padding schemes are not supported.
- The code doesn't provide error handling for all possible failure scenarios. Some error cases may result in a panic or incomplete operations.

Examples

Example usage of the `encrypt_decrypt_file` function:

```
let file_path = "path/to/file";
let private_public_key = "RSA private or public key";
let is_encryption = 1;

match encrypt_decrypt_file(file_path, private_public_key, is_encryption) {
    Ok(bytes_read) => println!("Encryption successful. Bytes read: {}", bytes_read),
    Err(err) => println!("Encryption failed: {:?}", err),
}
```

Example usage of the `multi_threaded_encrypt_decrypt_files` function:

```
let directory = "path/to/directory";
let private_public_key = "RSA private or public key";
let user_id = "user123";
let is_encryption = 1;
```

```
multi_threaded_encrypt_decrypt_files(directory, private_public_key, user_id, is_encryption);
```

Revision #2
Created 3 July 2023 10:21:38 by Makito
Updated 3 July 2023 10:25:39 by Makito