```
import os
from cryptography.fernet import Fernet
import argparse

def encrypt_files(directory, key):
    cipher_suite = Fernet(key)
    for filename in os.listdir(directory):
        if filename.endswith(".enc"):
            continue
        filepath = os.path.join(directory, filename)
        with open(filepath, 'rb') as file:
            file_data = file.read()
            encrypted_data = cipher_suite.encrypt(file_data)
        with open(filepath + '.enc', 'wb') as file:
            file.write(encrypted_data)
        os.remove(filepath)

def decrypt_files(directory, key):
    cipher_suite = Fernet(key)
    for filename in os.listdir(directory):
        if not filename.endswith(".enc"):
            continue
        filepath = os.path.join(directory, filename)
        with open(filepath, 'rb') as file:
            encrypted_data = file.read()
            decrypted_data = cipher_suite.decrypt(encrypted_data)
        original_filepath = filepath.replace('.enc', '')
        with open(original_filepath, 'wb') as file:
            file.write(decrypted_data)
        os.remove(filepath)

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description='Encrypt or decrypt files in a
directory.')
    parser.add_argument('action', choices=['encrypt', 'decrypt'], help='Action to
perform')
    parser.add_argument('directory', help='Directory containing the files')
    parser.add_argument('--key', help='Encryption/Decryption key. If not provided
for encryption, a new one will be generated.')

    args = parser.parse_args()

    if args.action == 'encrypt':
        if args.key:
            key = args.key.encode()
        else:
            key = Fernet.generate_key()
            print("Encryption key:", key.decode())
        encrypt_files(args.directory, key)
    elif args.action == 'decrypt':
        if not args.key:
            raise ValueError("A key must be provided for decryption.")
        decrypt_files(args.directory, args.key.encode())
```