# Task 1: Proof of Concept (PoC) Report

## Tool Name: KeyBTC Decrypting Tool

### History

#### 1 Description:

The KeyBTC Decrypting Tool was created as a response to the KeyBTC ransomware, a malicious software strain known for encrypting users' files and demanding Bitcoin ransom payments. The tool emerged from reverse engineering efforts by cybersecurity researchers and malware analysts aiming to help victims recover their data without paying the ransom.

KeyBTC itself was part of a broader trend of Bitcoin-themed ransomware families (like BTCWare, CryptoBTC, etc.), and likely surfaced around 2017–2019, although various variants or imitators may have followed in later years. The KeyBTC Decrypting Tool was released as an open-source or private utility depending on the context (e.g., CERTs, law enforcement, forensic analysts).

#### 2 What Is This Tool About?

The KeyBTC Decrypting Tool is a forensic recovery utility designed to decrypt files that have been locked by the KeyBTC ransomware.

It operates under the following assumptions:

i. The ransomware uses AES (symmetric) encryption to encrypt each file.
ii. The AES keys may be encrypted with RSA (asymmetric) and stored alongside or within the encrypted file or ransom note.
iii. Victims or analysts may have access to:
iv. The attacker's RSA private key (recovered or leaked).
v. Known static AES keys or IVs (in poorly coded variants).
vi. Unencrypted versions of some files (known-plaintext attacks).

The tool is written in Python, using well-known libraries such as PyCryptodome for secure crypto operations. It offers a modular and auditable way to decrypt KeyBTC-encrypted files, verify successful restoration, and handle file batch processing.

#### 3 Key Characteristics / Features:

| Feature | Description |
|---|---|
| AES & RSA Decryption Support | Supports AES-256-CBC for file decryption and RSA-OAEP for AES key decryption. |

| Feature | Description |
|---|---|
| PoC-Friendly | Designed as a proof-of-concept to test decryption under lab or CTF settings. |
| Safe & Offline | Fully offline tool; no need for internet connectivity or contacting servers. |
| File Recovery | Recovers files with .KEYBTC extension and restores original content. |
| Verbose Logging | Provides step-by-step status updates to track decryption progress. |
| Batch Support (optional) | Can be extended to handle multiple files in bulk. |
| Forensic Integration | Easily integrated into digital forensics workflows or malware sandboxes. |
| Open-Source Ready | Easily modifiable code for researchers or SOC teams to adapt. |

## 4 Types / Modules Available:

The tool is modular by design and can be broken down into the following components:

### ➢ AES Decryption Module

Purpose: Decrypts the file contents using the recovered AES key.

Functions:

decrypt_file(file_path, aes_key, iv)

Uses AES.MODE_CBC (or adjustable mode if variant differs).

Dependencies: Crypto.Cipher.AES, Crypto.Util.Padding.

### ➢ RSA Key Decryption Module

Purpose: Decrypts the AES key using a provided RSA private key.

Functions:

decrypt_aes_key(encrypted_key_path, private_key_path)

Uses PKCS1_OAEP for secure RSA decryption.

Dependencies: Crypto.PublicKey.RSA, Crypto.Cipher.PKCS1_OAEP.

### ➢ File I/O Handler

Purpose: Loads and saves encrypted and decrypted files.

Functions:

save_decrypted_file(original_path, decrypted_data)

Handles renaming and path safety (e.g., removing .KEYBTC suffix).

### ➢ Configuration Module (optional)

Purpose: Manages settings like IVs, default paths, batch modes.

Can be a .json file or Python dictionary.

Makes it easy to adapt to different KeyBTC variants.

### ➢ Batch Decryption (Extendable)

Purpose: Enables processing multiple encrypted files in directories.

Sample logic (user can implement):

```
for file in os.listdir("encrypted_folder"):
   if file.endswith(".KEYBTC"):
      decrypt_file(...)
```

## 5. How Will This Tool Help?

The KeyBTC Decrypting Tool provides significant value in incident response, forensic analysis, and malware reverse engineering:

| Scenario | How It Helps |
|---|---|
| Ransomware Recovery | Allows recovery of encrypted data without paying ransom, if keys are known. |
| Malware Analysis | Helps analysts understand encryption logic and flaws in ransomware design. |
| File Forensics | Recovers original file content for investigation, legal evidence, or backup. |
| Toolchain Integration | Can be integrated into forensic kits, SIEM tools, or automated decryptors. |
| Training / CTF Use | Ideal for demonstrating ransomware decryption in controlled environments. |
| Threat Containment | Helps neutralize impact of ransomware campaigns by enabling file recovery. |

6. Proof of Concept (PoC):

   While this is a code-based tool, here are conceptual visual representations.

   Encryption Flow Diagram

   i. Visual flow:

   Original File → AES Encryption → .KEYBTC File

   AES Key → RSA Public Key Encryption → Encrypted AES Key

   ii. 2. Decryption Workflow (Tool)

   [Encrypted File + Encrypted AES Key] + [RSA Private Key]

   ↓

   → Decrypt AES Key

   → Decrypt File

   → Restore Original File

iii. Before/After Snapshot

Encrypted: secret.txt.KEYBTC

Decrypted: secret.txt

7. 15-Liner Summary:

1. Name: KeyBTC Decrypting Tool
2. Purpose: Recover files encrypted by KeyBTC ransomware.
3. Language: Python 3 (with PyCryptodome).
4. Core Functions: AES file decryption + RSA key recovery.
5. Encrypted File Extension: .KEYBTC
6. AES Mode: AES-256-CBC (default; configurable).
7. Key Recovery: RSA private key used to decrypt AES key.
8. IV Handling: Static or extracted per variant.
9. Modular Design: Easy to extend or audit.
10. File Input: .KEYBTC files + encrypted AES key blob.
11. File Output: Original restored files.
12. Usage Context: Ransomware recovery, analysis, training.
13. Security: Offline use, verifiable steps, no external calls.
14. Batch Mode Support: Optional — decrypt folders recursively.
15. Open-Source Ready: Adaptable for different ransomware strains.

8. Time to Use / Best Case Scenarios:

| **Scenario** | **Best Use Case for Tool?** |
|---|---|
| Encrypted files only | Not helpful unless encrypted AES key or private key is available. |
| You have the RSA private key | Yes — full decryption is possible. |
| Known AES key or known plaintext | Yes — can adapt tool to brute or guess. |
| CTF / Ransomware Simulations | Excellent for demos, training, and red team scenarios. |

| Scenario | Best Use Case for Tool? |
|---|---|
| Malware analysis lab | Use to reverse crypto logic and analyze flaws. |
| Victim has partial key data / IV leak | Can customize tool to recover files partially or fully. |
| No keys available at all | Only useful to explore encryption mechanism, not recovery. |

## 9. When to Use During Investigation:

The KeyBTC Decrypting Tool is most effective in post-infection response or forensic recovery phases of a ransomware investigation.

| Investigation Phase | Use Tool | Reason |
|---|---|---|
| Initial Triage | No | Focus is on containment, process kill, and isolating infection. |
| Malware Reverse Engineering | Yes | Useful to understand crypto mechanics and test recovery. |
| Data Recovery | Yes | Actively used to recover files if key info is available. |
| Threat Intelligence | Maybe | Can extract static keys, IV patterns, or encryption fingerprints. |
| Evidence Collection | Yes | Demonstrates recovery capability or confirms ransomware behavior. |
| Lab Simulation / CTF | Yes | Safe environment to demonstrate ransomware decryption flow. |

10. Best Person to Use This Tool & Required Skills:

| Role | Fit? | Why / Skills Needed |
|---|---|---|
| **Incident Responder (IR)** | Yes | Needs Python basics, file handling, and crypto concepts. |
| **Malware Analyst** | Yes | Comfortable with encryption flow, reverse engineering, and analysing binary artifacts. |
| **Digital Forensics Expert** | Yes | Skilled in recovering and preserving data; tool aids evidence extraction. |
| **Security Researcher / Educator** | Yes | Great for demonstrations or studying ransomware behaviours. |
| **Sysadmin / IT Support** | Limited | Might struggle unless provided clear keys or a GUI wrapper. |
| **Beginner in Cybersec** | Basic | Needs basic Python knowledge and understanding of AES/RSA. |

i. Required Skills

a) Python 3 scripting knowledge
b) Understanding of:
c) AES and RSA encryption
d) File I/O operations
e) IV handling and padding (e.g., PKCS#7)
f) Optional: familiarity with ransomware behaviours

## 11.Flaws / Suggestions to Improve:

| Flaw | Suggestion to Improve |
|---|---|
| Static IV (hardcoded or missing) | Add logic to **extract IV dynamically** from file headers or sidecar files. |
| No batch decryption support | Implement **recursive folder decryption with error handling**. |
| No GUI interface | Add **Tkinter or PyQt** front-end for less technical users. |
| No error diagnostics | Implement **logging**, exception tracking, and file integrity verification. |
| Only RSA decryption path | Add support for variants using **hardcoded AES keys**, XOR, or stream ciphers. |
| Lacks integrity verification | Include **hash comparison** (e.g., SHA-256) to verify decrypted file validity. |

## 12. Good About the Tool:

| Strength | Description |
|---|---|
| Simple and modular | Easy to read, modify, and adapt for different ransomware cases. |
| Offline and safe | Doesn’t connect to internet, preserving confidentiality and forensics chain. |
| Educational value | Great for teaching crypto misuse in ransomware and PoC development. |
| Forensic-compatible | Maintains file structure and logs operations; easy to integrate into workflows. |

| Strength | Description |
| --- | --- |
| Low dependencies | Only needs Python and PyCryptodome; no bloat. |
| Open to extension | Can be upgraded with plugins for key scanning, key brute-force attempts, etc. |

Prepared By:

Intern Name: Kush Dewal

Intern ID: 360