Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.33, NO.3, Jun. 2023 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2023.33.3.511

볼륨 암호화 및 백업 응용프로그램에 대한 복호화 방안 연구*

박 귀 은,[1†] 이 민 정,[1] 강 수 진,[1] 김 기 윤,[1] 김 종 성[2‡]
[1,2]국민대학교 (대학원생, 교수)

A Study on the Decryption Method for Volume Encryption and Backup Applications*

Gwui-eun Park,[1†] Min-jeong Lee,[1] Soo-jin Kang,[1] Gi-yoon Kim,[1] Jong-sung Kim[2‡]
[1,2]Kookmin University (Graduate student, Professor)

요 약

개인정보보호에 대한 인식이 증가하면서 사용자 PC의 데이터를 보호하기 위해 실시간 암호화 또는 가상 드라이브 볼륨을 사용하는 다양한 Full Disk Encryption (FDE) 계열 응용프로그램이 개발되고 있다. FDE 계열 응용프로그램은 사용자의 주요 데이터가 담긴 볼륨을 암호화하여 보호한다. 그러나 디스크 암호화 기술이 발전함에 따라 일부 사용자들은 특정 범죄 행위와 관련된 증거를 암호화하는 등 이를 악용하여 포렌식 수사에 어려움을 주고 있다. 이에 대응하기 위해 FDE 계열 응용프로그램에 사용된 암호화 과정을 분석하여, 암호화된 데이터를 복호화하는 선행연구가 필요하다. 본 논문에서는 볼륨 암호화 및 백업 기능을 제공하는 Cryptomator와 Norton Ghost를 분석한다. 암호화된 데이터 구조와 암호화 과정을 분석하여 주요 데이터를 분류하고, 데이터 복호화에 사용되는 암호화 알고리즘을 식별한다. 해당 응용프로그램들의 암호화 알고리즘은 최근에 등장하고 있거나 커스텀된 암호화 알고리즘으로 이를 분석하여 주요 데이터를 복호화한다. 복호화에 사용되는 데이터 암호키를 생성하기 위해 사용자 패스워드가 필수적으로 요구되며, 각 응용프로그램의 기능을 사용하여 패스워드 획득 방안을 제시한다. 이는 패스워드 전수조사의 한계를 보완하였으며, 획득한 패스워드를 기반으로 암호화된 데이터를 복호화하여 사용자의 주요 데이터를 식별한다.

ABSTRACT

As awareness of personal data protection increases, various Full Disk Encryption (FDE)-based applications are being developed that real-time encryption or use virtual drive volumes to protect data on user's PC. FDE-based applications encrypt and protect the volume containing user's data. However, as disk encryption technology advances, some users are abusing FDE-based applications to encrypt evidence associated with criminal activities, which makes difficulties in digital forensic investigations. Thus, it is necessary to analyze the encryption process used in FDE-based applications and decrypt the encrypted data. In this paper, we analyze Cryptomator and Norton Ghost, which provide volume encryption and backup functions. We analyze the encrypted data structure and encryption process to classify the main data of each application and identify the encryption algorithm used for data decryption. The encryption algorithms of these applications are recently emergin gor customized encryption algorithms which are analyzed to decrypt data. User password is essential to generate a data encryption key used for decryption, and a password acquisition method is suggested using the function of each application. This supplemented the limitations of password investigation, and identifies user data by decrypting encrypted data based on the acquired password.

Keywords: FDE-based application, Digital Forensic, Encryption algorithm, User Password

Received(02. 21. 2023), Modified(04. 04. 2023), Accepted(04. 04. 2023)

* 본 연구는 2022년도 정부(법무부)의 재원으로 대검찰청의 지원을 받아 수행된 연구결과임.

† 주저자, dhnr16@kookmin.ac.kr
‡ 교신저자, jskim@kookmin.ac.kr(Corresponding author)

I. 서 론

개인정보 유출 사례가 지속적으로 증가함에 따라 개인정보보호에 대한 인식률이 높아지고 있다[1]. 사용자 PC는 개인정보를 포함한 다양한 데이터가 저장되어있다. 이와 같은 사용자의 민감한 데이터를 보호하기 위해 별도의 저장소에 보관하거나, 보호 기능을 제공하는 응용프로그램을 사용하는 경우가 존재한다. 특히 사용자 PC의 데이터를 안전하게 보관하기 위해 디스크 암호화 소프트웨어나 하드웨어를 이용하여 드라이브 볼륨에 있는 데이터를 암호화하는 기법인 FDE (Full Disk Encryption)가 많이 적용되고 있다. 예를 들어 BitLocker는 Windows에 기본적으로 탑재되어 있으며, FDE 기능을 제공하여 이동식 드라이브 및 시스템 드라이브를 암호화한다[2]. CrossCrypt는 실시간 암호화 프로그램으로, 사용자가 드라이브에 저장한 파일을 암호화하여 가상 드라이브를 생성한다[3].

FDE 기법이 적용된 응용프로그램은 디스크 암호화 기능뿐만 아니라 실시간 암호화 기능(On-The-Fly Encryption)[1]을 제공하여 가상 드라이브 볼륨 사용 등 다양한 방식으로 사용되고 있다. 최근에는 드라이브 볼륨에 무단 접근을 방지하기 위해 데이터가 저장되거나 사용될 때 실시간으로 암·복호화하는 방식을 사용한다[4]. 실시간 암호화 방식을 사용하면 올바른 암호키가 제공되는 즉시 파일 접근이 가능하며, 일반적으로 가상 볼륨이 사용된다. 볼륨이 가상화되어 사용되므로 암호화되지 않은 것처럼 보일 수 있지만 실제 볼륨은 암호화되어 저장된다. 볼륨 내 전체 데이터는 기타 메타 데이터를 포함하여 파일명, 디렉터리명과 파일 내용 등을 모두 암호화한다. 해당 기능을 통한 데이터 수정 시 복호화 이후 재 암호화할 필요가 없으며, 패스워드 또는 암호키가 존재할 경우 저장된 데이터를 즉시 확인할 수 있다.

본 논문에서는 FDE 기법 중 On-The-Fly Encryption 방식 또는 백업 기능을 사용하여 데이터를 암호화하는 응용프로그램을 분석한다. 이러한 프로그램들을 FDE 계열 응용프로그램이라 지칭한다. FDE 계열 응용프로그램은 다양한 방식으로 개발되고 있으며, 일부 사용자는 FDE 계열 응용프로그램을 악용하여 범죄 행위 관련 디지털 증거를 암호화하여 포렌식 수사에 어려움을 주고 있다.

실제로 국내 드루킹 사건과 브라질의 Satyagraha 작전 등에서 오픈소스 FDE 계열 응용프로그램인 TrueCrypt가 사용되어 주요 증거자료 수집에 어려움이 존재하였다[5][6].

이러한 범죄 사건의 피해를 줄이기 위해 FDE 응용프로그램 사용 시 생성되는 암호화된 데이터를 식별하고 암호화 과정을 분석하여 데이터를 복호화하는 연구가 선행되어야 한다. 특히, 데이터 복호화에 사용되는 암호화 알고리즘을 분석하고, 실제 암호화에 사용되는 패스워드나 암호키를 획득하는 방안에 대한 연구가 필요하다.

본 논문에서는 Windows 환경의 FDE 계열 응용프로그램인 Cryptomator와 Norton Ghost를 분석한다. 해당 응용프로그램의 주요 기능 사용으로 생성된 데이터 구조를 식별하고, 암호화된 데이터 구조와 암호키 생성 및 암호화 과정을 분석한다. 분석한 응용프로그램 모두 데이터 암호화 시 사용자 패스워드가 필수적으로 요구되므로, 사용자 입력 패스워드를 획득할 수 있는 방안을 제시한다.

II. 관련 연구

FDE 기법에 대한 안전성 분석 및 동향 연구와 FDE 계열 응용프로그램의 암호화 과정 분석, 패스워드와 암호키 획득을 위한 메모리 분석 등 다양한 연구가 존재한다.

Eoghan Casey 등은 범죄 수사 현장에서 FDE로 암호화된 데이터를 획득하기 위한 가이드를 제공하였다[7]. 해당 가이드를 통해 라이브 시스템 상태에서 암호키 또는 암호화된 데이터 복구 가능성을 제시한다. Tilo Muller 등은 FDE에 대한 표준 공격 및 취약점을 소개하며 FDE의 보안 안전성을 분석하였다[8]. 서승희 등은 안드로이드 상에서 FDE와 이를 보완한 FBE (File Based Encryption)의 특징 및 동작 과정을 정리하였으며, 이에 대한 복호화 방안과 안전성 연구 동향을 소개하였다[9]. 장성민 등은 FDE 환경에 따른 디지털 증거 수집 절차와 대응 방안을 제시하며, 실제 사용률이 높은 FDE 환경의 응용프로그램 탐지 및 세부 대응 방안을 설명하였다[10]. Lijun Zhang 등은 TrueCrypt 볼륨의 암호화 과정, 키 유도 방식과 패스워드 검증 과정 등 암호화된 데이터 구조와 암호화 원리를 분석하였다[11]. 이를 바탕으로 암호화된 볼륨을 식별하고, 헤더를 추출하여 패스워드를 획득하는 방안을 제시하였다[12]. 또한, Windows 운영체제에서 기본적으로

1) 실시간으로 데이터를 로드하여 권한 있는 사용자가 복호화된 데이터를 확인 가능하지만, 저장매체는 여전히 암호화된 상태로 존재하게 하는 기법

디스크 암호화를 제공하고 있는 BitLocker에 대한 연구도 존재한다. Cheng Tan 등은 BitLocker의 암호화 과정을 분석하였고, 시스템 파티션 암호화와 비시스템 파티션 암호화 케이스로 나누어 암호화된 볼륨 마스터 암호키의 복호화 방안을 제시하였다[13]. 그 외에도 FDE 기법이 적용된 응용프로그램의 메모리 분석을 통해 암호키가 로컬 영역에 저장되지 않거나 데이터 보호 기법이 적용된 암호화 데이터를 분석하는 연구도 존재한다. Abdullah Kazim 등은 메모리 포렌식을 통해 인스턴트 메시징 도구로 채팅 메시지를 복구하였으며, BitLocker와 TrueCrypt로 암호화된 볼륨의 마스터 암호키 또한 획득 가능성을 제시하였다[14]. Alex Halderman 등은 BitLocker, TrueCrypt와 FileVault 등을 대상으로 cold-boot 공격을 적용하여 응용프로그램별로 암호키 추출 및 복호화 가능성을 분석하였다[15]. Vipkas Al Hadid Firdaus 등은 Linux 운영체제에서 FDE 기법이 적용된 응용프로그램의 디스크 암호키 획득을 위해 라이브 상태에서 메모리를 분석하였다[16].

상기 연구들은 FDE 기법이 적용된 응용프로그램의 암호화된 데이터 구조 및 암호화 과정 분석 등 본 연구와 유사하나, 본 논문에서는 아직 분석되지 않은 FDE 계열 응용프로그램을 선정하여 세부적인 암호화된 데이터 구조 파악 및 암호화 과정을 분석한다. 또한, 메모리 분석을 통해 패스워드 획득 방안을 제시하는 연구 등과 달리 선정한 응용프로그램별로 기능 사용을 통해 사용자 패스워드를 획득하는 방안을 제시한다.

III. Cryptomator 분석

Skymatic社에서 개발한 오픈소스 소프트웨어로, 가상 드라이브(vault)를 생성하고, 드라이브에 저장되는 파일을 사용자 입력 패스워드를 기반으로 암호화한다. 패스워드는 서비스 이용에 있어 필수적으로 요구되며 최소 8자를 사용해야 한다. Cryptomator는 현재 다양한 플랫폼에서 서비스를 제공하며, 본 논문에서는 Windows 환경에서 2022년 12월 기준 최신 버전인 1.6.17 버전을 분석하였다[17].

3.1 암호화된 데이터 구조

Cryptomator는 실시간 파일 암호화를 지원하며, 사용자 PC에서 데이터가 암호화되어 클라우드에 전송한다. 생성된 vault와 사용자가 지정한 경로에 존재하는 주요 데이터 구조는 Fig. 1.과 같다.

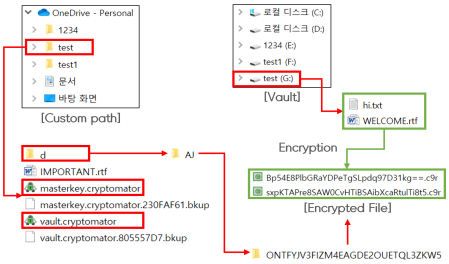

Fig. 1. Data structure of Cryptomator

새로운 vault 생성 시 'vault.cryptomator'와 'masterkey.cryptomator' 파일이 사용자 지정 경로에 저장되며, 이를 통해 암호화된 파일 저장 경로임을 파악할 수 있다. 2개의 파일은 생성한 vault에 따라 고정적으로 존재하는 파일이며, vault마다 세부 내용은 일부 상이하다. 그 외에 암호화된 파일의 경우 파일명과 파일의 내용이 암호화되어 저장된다.

3.1.1 vault.cryptomator 파일 구조

'vault.cryptomator' 파일에는 vault에 대한 기본 정보와 사용된 암호 알고리즘이 JWT (JSON Web Token)[18] 형식으로 저장된다. 해당 파일은 Header, Payload와 Signature로 구성되며, 각 영역은 JSON (JavaScript Object Notation) 형식의 데이터를 base64url로 인코딩한 것으로 마침표로 구분된다. 각 영역별로 세부 데이터는 다음 Table 1.과 같다.

Table 1. vault.cryptomator file data

Part	Value	Date
Header	kid	key filename
	typ	type (JWT)
	alg	Default: HS256
Payload	format	8
	shortening Threshold	220
	jti	UUID
	cipherCombo	Encryption algorithm
Signature	signature	signature data (HMACSHA256)

Header의 'kid'에 저장된 키 파일명을 통해 마스터 암호키(Master Encryption Key, MEK)와 마스터 MAC 키(Master MAC Key, MMK) 관련 정보가 'masterkey.cryptomator' 파일임을 확인할 수 있다. Payload에는 vault를 고유하게 식별 가능한 랜덤 UUID 값이 'jti'에 저장되어 있으며, 파일 암호화에 사용되는 알고리즘 정보가 'cipher Combo'에 저장된다. 현재 Cryptomator는 'SIV_CTRMAC'을 지원하므로 해당 값으로 고정되어 있다. 이는 org.cryptomator.cryptolib에서 제공하는 알고리즘으로 암호화 방식은 파일명 암호화에 AES-256-SIV[2][19]가 사용되며, 파일 내용 암호화에는 AES-256-CTR과 HMAC이 사용된다.

3.1.2 masterkey.cryptomator 파일 구조

'masterkey.cryptomator' 파일에는 암호화된 MEK, MMK와 마스터키 생성에 사용되는 Scrypt 알고리즘 인자가 JSON 형식으로 저장된다. Scrypt 알고리즘은 대량의 메모리를 사용한 패스워드 기반 키 유도 함수의 한 종류이다[20]. 해당 파일은 Fig. 2.와 같이 각 항목의 모든 데이터를 정수 혹은 base64로 인코딩하여 저장한다.

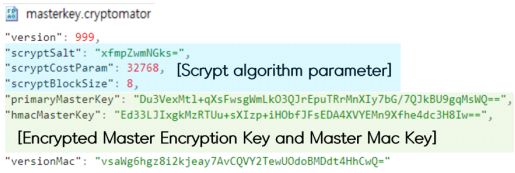

Fig. 2. masterkey.cryptomator file data

'version' 정보는 999로 고정되며, 'ScryptSalt', 'ScryptCostParam'과 'ScryptBlockSize'에는 MEK와 MMK 생성 과정에 적용되는 Scrypt 알고리즘 인자 값이 저장된다. 'primaryMasterKey'와 'hmacMasterKey'에는 암호화된 MEK와 MMK가 저장된다. 'versionMac'은 다운그레이드를 방지하기 위한 vault 버전의 HMAC-256 값이다. 'masterkey.cryptomator' 파일에 저장된 값들은 암호화 과정에 사용되는 암호키 생성에 필요하므로 해당 파일의 수집이 선행되어야 한다.

2) AES를 사용한 새로운 블록암호 운영모드로, RFC 5297에 정의되어 있으며, 추가 데이터와 평문을 통해 인증 및 암호화를 수행하는 알고리즘

3.1.3 암호화 대상 파일 구조

Cryptomator로 암호화된 파일의 데이터는 파일 헤더와 파일 내용으로 구성되어 있으며, 전체 암호화된 파일의 구조는 Fig. 3.과 같다.

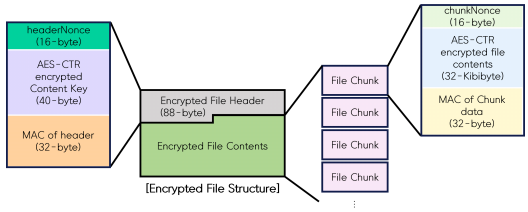

Fig. 3. Encrypted file structure

암호화된 파일의 상위 88-byte는 파일 헤더로, 파일 내용 암호화에 사용되는 암호키인 Content Key (CK)가 암호화되어 저장된다. 파일 헤더의 상세 구조는 Fig. 4.와 같다.

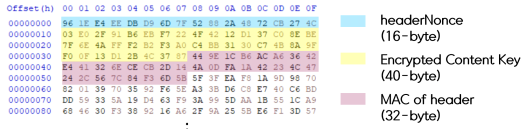

Fig. 4. Encrypted file header structure

파일 내용에 해당하는 데이터는 최대 32-KiB와 48-byte로 구성된 chunk 단위로 나누어 암호화한다. 하나의 chunk 기준으로 상위 16-byte는 chunkNonce, 32-KiB 크기의 암호화 대상 데이터와 32-byte 크기의 MAC 값이 순서대로 연접되어 저장된다. 파일 크기에 따라 여러 chunk가 사용되며, 암호화된 chunk는 순서대로 연접된다. chunk마다 존재하는 nonce 값은 서로 다르며, 마지막 chunk의 암호화된 파일 내용의 크기는 32-KiB보다 작을 수 있다. 파일 헤더와 파일 내용의 데이터 구조는 nonce, 암호화 영역과 MAC 값으로 동일하다. nonce와 MAC 값이 암호화 과정에 사용되므로 파일별 크기에 따라 데이터 구조를 파악해야 한다.

3.2 암호화 과정 분석

Cryptomator의 암호화 과정은 파일 데이터 암호화와 파일명 암호화로 구분된다. 파일 데이터는

AES-256-CTR로 암호화되며, 파일명은 AES-256-SIV로 암호화된다. 파일 데이터 중 헤더 영역과 파일명을 암호화하기 위해서는 MEK가 필요하며, 파일 내용에 해당하는 영역을 암호화하기 위해서는 CK가 필요하다. CK는 파일 헤더에 존재하므로 파일 내용을 복호화하기 위해서는 파일 헤더 복호화가 선행되어야 한다.

3.2.1 MEK/MMK 생성 과정

MEK는 파일 암호화에 사용되며, MMK는 파일, 파일 암호키 그리고 파일명 암호키의 MAC 값 검증에 사용된다. MEK와 MMK는 JAVA의 SecureRandom 알고리즘으로 랜덤하게 생성되며, 이후 AES Key Wrap[3)][21]으로 암호화된 뒤 base64로 인코딩되어 'masterkey.cryptomator' 파일에 저장된다. AES Key Wrap에 사용되는 암호키는 사용자 입력 패스워드를 기반으로 생성된다. 생성 알고리즘은 Scrypt이며, 사용되는 인자 또한 'masterkey.cryptomator' 파일에 저장된다. 마지막 인자인 ParallelizationFactor는 1로 사용되며, 이에 대한 상세 생성 및 저장 과정은 Fig. 5.와 같다.

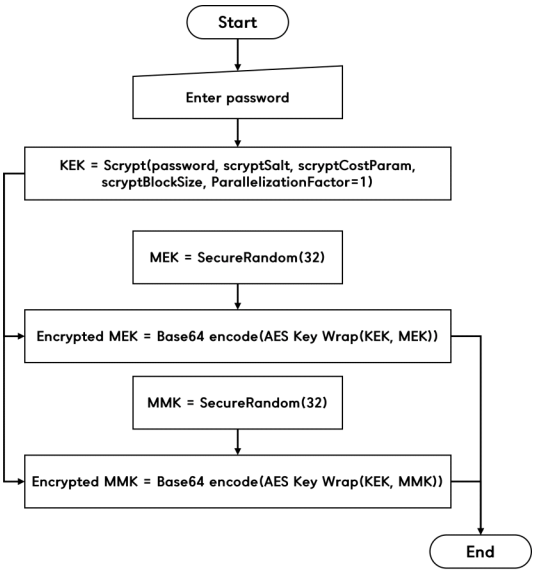

Fig. 5. Encryption process after generating MEK and MMK

3) RFC 3394에 정의된 알고리즘으로, 데이터 암호화에 사용된 암호키를 AES로 암호화하여 보호하는 알고리즘

3.2.2 파일 데이터 암호화 과정

파일 데이터의 암호화 과정은 파일 헤더와 파일 내용으로 구분된다.

파일 헤더의 상위 16-byte는 CK 암호화에 사용되는 nonce 값이며, 이후 40-byte가 CK를 포함한 파일 헤더의 암호화 대상이다. 해당 영역은 상위 8-byte의 0xFF FF FF FF FF FF FF FF 고정값 이후에 CK가 연접되어 있으며, 이를 MEK와 AES-256-CTR로 암호화한다. 암호화 영역을 복호화할 때 획득한 CK의 검증 방법은 8-byte 크기의 고정값의 존재 유무로 확인할 수 있다. 파일 헤더의 마지막 32-byte는 headerNonce와 CK를 포함한 암호화된 영역을 연접하여 MMK를 HMAC-SHA256으로 해싱한 값이다. 파일 헤더의 암호화 과정은 Fig. 6.과 같다.

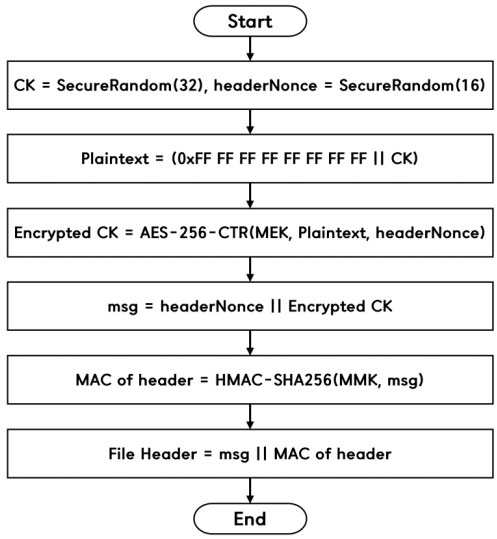

Fig. 6. Encryption process of file header

암호화에는 각 chunk에 해당하는 chunkNonce가 사용되며, 마지막 32-byte는 다음의 data를 연접하여 MMK를 HMAC-SHA256으로 해싱한 값이다.

o Data = headerNonce || blocknumber || chunkNonce || Encrypted data

headerNonce는 이전 파일 헤더 영역에 존재하는 nonce 값으로, chunkNonce와 다른 값이다. blocknumber는 0부터 시작하며, chunk의 개수에

따라 1씩 증가한다. 하나의 chunk를 기준으로 chunkNonce, 암호화된 파일 내용과 HMAC-SHA256 해싱 값을 순서대로 연접한다. 파일 크기에 따라 여러 chunk가 존재할 경우 위의 암호화 과정을 반복하며, chunk를 순서대로 연접한다. 파일 내용의 암호화 과정은 Fig. 7.과 같다.

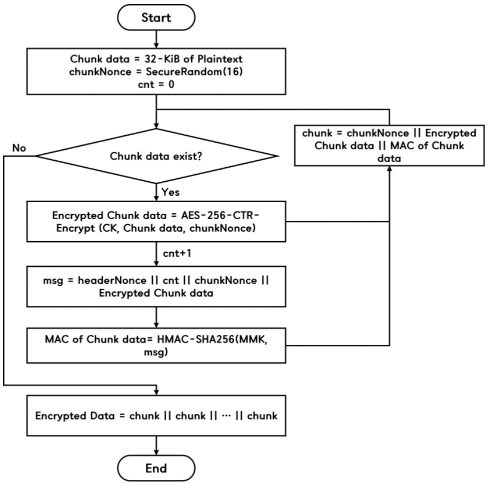

Fig. 7. Encryption process of file content

3.2.3 파일명 암호화 과정

Cryptomator는 파일명 또한 암호화되어 사용자 지정 경로에 저장된다. 암호화 방식은 MEK와 MMK를 AES-256-SIV로 암호화한 후 base64url로 인코딩되어 저장된다. 이때, AES-256-SIV에 사용되는 추가 데이터가 존재하며, Directory UUID 인자 값으로 해당 데이터는 암호화 대상 파일이 저장되는 경로 내 디렉터리의 UUID 값이다.

루트 디렉터리의 UUID는 NULL 값이며, 새로운 디렉터리 생성 시 해당 디렉터리의 UUID 값은 Fig. 8.과 같이 2가지 형태로 저장된다.

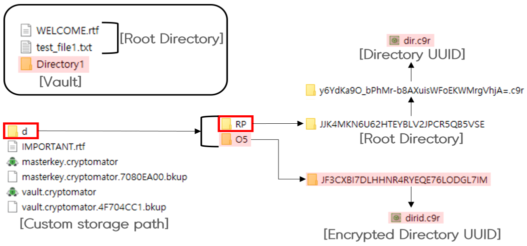

Fig. 8. UUID structure of directory

Vault 내에서 디렉터리 생성 시 Cryptomator의 루트 디렉터리인 'd' 내부에 각 디렉터리에 해당하는 UUID 값이 'dirid.c9r' 파일에 암호화되어 저장된다. 또한, 평문 형태의 UUID 값이 하위 디렉터리에 'dir.c9r' 파일에 평문 형태로 저장된다.

각 파일이 존재하는 디렉터리의 UUID 값은 상이하므로 파일명 암호화에 사용되는 디렉터리의 UUID 값은 동일한 경로에 존재하는 'dirid.c9r' 파일을 통해 UUID 값을 획득할 수 있다. 따라서 암호화된 디렉터리의 UUID 값을 복호화한 이후에 이를 사용하여 파일명을 복호화한다. 파일명의 암호화 과정은 Fig. 9.와 같다.

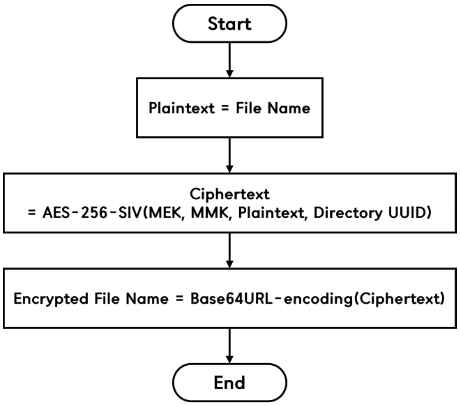

Fig. 9. Encryption process of file name

파일명은 암호화될 때 암호화된 파일명 뒤에 '.c9r' 확장자가 추가되어 저장된다. 암호화된 파일명과 확장자를 포함한 길이가 220-byte를 초과할 경우 암호화된 파일명과 파일 내용을 Fig. 10.과 같이 별도로 저장한다.

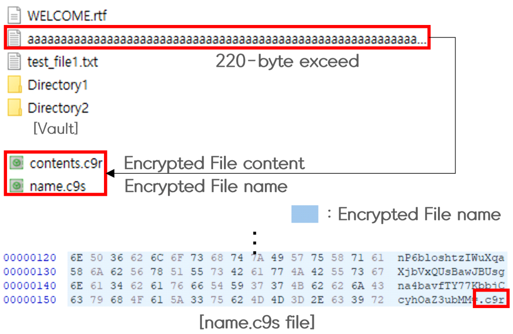

Fig. 10. Encrypted file name structure

암호화된 파일명은 'name.c9s' 파일에 저장되고, 암호화된 파일의 내용은 'contents.c9r' 파일에 저장된다.

IV. Norton Ghost 분석

Symantec社에서 개발한 백업 소프트웨어로 특정 디렉터리 및 파일 백업, 드라이브 전체 백업과 디스크 복제 기능 등을 제공한다. Windows 환경에서만 사용 가능하며, 본 논문에서는 Norton Ghost 개인용의 최종 버전인 버전 15.0.1.36526을 분석하였다. 2013년 4월을 기준으로 Norton Ghost 개인용의 판매는 중단되었지만, 기업용은 지속적으로 사용되고 있다[22].

Norton Ghost는 Fig. 11.과 같이 '선택한 파일 및 디렉터리 백업'과 'PC 전체 백업' 기능을 제공한다.

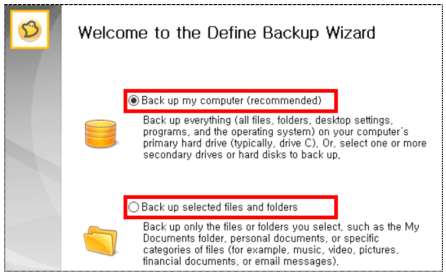

Fig. 11. Create Norton Ghost's Backup data

'선택한 파일 및 디렉터리 백업' 기능의 경우 선택한 파일 및 디렉터리뿐만 아니라 새로운 파일 및 디렉터리를 추가하여 백업할 수 있다.

'PC 전체 백업' 기능의 경우 선택한 드라이브 전체를 백업하여 사용자가 지정한 경로에 저장한다. 데이터 백업 시 옵션을 통해 Fig. 12.와 같이 패스워드와 암호화 여부를 설정할 수 있다.

패스워드를 설정하지 않고 데이터 백업이 가능하지만, 패스워드를 설정해야만 암호화 설정을 추가적으로 선택할 수 있다. 따라서 백업 데이터를 암호화하기 위해서는 패스워드 입력이 필수적으로 요구된다. 패스워드 설정을 통해 백업된 데이터에 대한 접근을 제어할 수 있다. Norton Ghost는 AES 암호 알고리즘을 사용한 데이터 암호화 기능만 지원하며, AES 키 길이를 선택할 수 있다. 이에 따라 최소로 입력해야 하는 패스워드 길이가 다르게 적용된다.

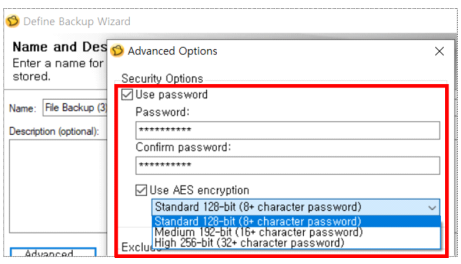

Fig. 12. Encryption options

4.1 암호화된 데이터 구조

Norton Ghost의 백업 기능 사용 시 생성되는 파일 구조는 Fig. 13.과 같다.

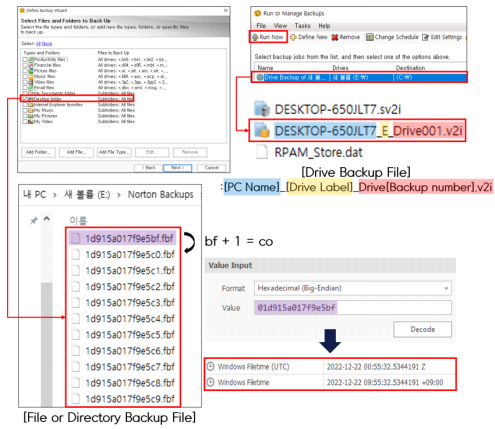

Fig. 13. Data structure Norton Ghost

파일 및 디렉터리 백업 기능을 사용하여 특정 데이터를 백업할 경우 사용자 지정 경로에 '.fbf' 확장자 파일로 암호화되어 저장된다. 해당 파일의 파일명은 백업 시작 시각을 Windows filetime[4]으로 인코딩하여 hex string 형태로 변경한다. 2개 이상의 파일을 동시에 백업하는 경우 첫 번째 파일의 hex string 값을 기준으로 1씩 증가되어 저장된다. '.fbf' 파일의 시그니처는 'SymBackup 1.0' 이후 0x0A 1A 01 값을 연접하여 파일 시작의 상위 16-byte에 저장된다.

전체 PC 백업 기능을 사용하여 특정 드라이브를 백업할 경우 사용자 지정 경로에 '[PC 이름]_[드라

4) Windows에서 지원하는 시간 포맷 중 하나로, 1,000만분의 1초 단위로 표현되며, 1601년 1월 1일을 기준으로 경과한 시간을 의미

이브 라벨]_Drive[백업 번호]'를 파일명으로 '.v2i' 확장자 파일이 생성된다. 백업 번호는 새롭게 백업할 때마다 1씩 증가한다. '.v2i' 파일의 시그니처는 '$CAN'이며, 파일 시작의 상위 4-byte에 저장된다.

2가지 기능을 통해 생성된 암호화된 파일은 암호화 과정 이후 변경된 파일명과 각 파일에 존재하는 시그니처 값으로 구분할 수 있다.

4.1.1 파일 및 디렉터리 백업 파일('.fbf' 확장자 파일)

암호화된 파일 및 디렉터리의 백업 데이터 구조는 크게 헤더와 바디 영역으로 구성되어 있다. 암호화된 헤더 영역에 대한 상세 구조는 Table 2.와 같다.

Table 2. Data structure of encrypted Backup file header

Offset	Field	Data
0~16	signature	'SymBakup 1.0' 0x0A 1A 01
17~24 25~32	segment ID	file name
32~33	header length	header length
34~35	header's ciphertext length	ciphertext length
36~39	flag	0x15: no password 0x0115: only set password 0x0215: AES128 encryption 0x0315: AES192 encryption 0x0415: AES256 encryption
40~47	salt2	salt2 salt: salt1||salt2
48~55	password check value	password check value
56~63	counter (for nonce)	counter
64~68	salt1	salt1 salt: salt1||salt2
69~84	backup PC name	backup PC name
85~N	header's ciphertext	header data
N~N+4	end signature	'FHT1'

헤더 영역에는 암호화된 파일의 시그니처 값, segment ID, 데이터 암호키 생성에 사용되는 salt 값, 데이터 암호화에 사용되는 카운터 값과 사용자 입력 패스워드 검증에 사용되는 값 등 암호화 과정에 사용되는 다양한 정보가 저장된다. 또한, 헤더 영역과 암호문의 크기 정보가 저장되며, 헤더 영역 존재하는 flag 값은 사용자가 백업 생성 시 패스워드 및 암호화 알고리즘 사용 여부를 확인할 수 있다.

헤더 영역의 마지막 시그니처인 'FHT1' 값을 통해 헤더 영역과 바디 영역을 세분화하는 용도로 사용된다. 바디 영역은 특정 크기에 따라 반복되는 구조로 다시 나누어질 수 있으며, 동일하게 'FHT1' 구분자가 사용된다. 암호화된 바디 영역에 대한 상세 구조는 Table 3.과 같다.

Table 3. Data Structure of encrypted Backup file body

Offset	Field	Data
0~4	file index	file index
5~8	compression flag	0: not compressed 1: deflate
9~12	length of decompressed result	length of decompressed result
13~16	length of compressed data	length of compressed data
N	body's ciphertext	body data
N+4	end signature	"FHT1"

File index는 바디 영역의 개수에 따라 0부터 1씩 증가하며, compressed flag 값은 파일의 압축 여부를 의미한다. Flag 값이 0인 경우 바디 영역의 암호문이 압축되지 않았으며, 1인 경우 deflate 압축[23] 방식으로 저장된다. 압축된 데이터와 Adler-32 체크섬이 존재할 경우 ZLIB으로, CRC32 체크섬이 존재할 경우 GZIP으로 압축된 것임을 확인할 수 있다. Norton Ghost 바디 영역에 존재하는 암호문에는 CRC32 체크섬 값을 포함한 값이 저장되므로 데이터가 압축되었을 경우 GZIP 방식으로 압축 해제를 해야한다.

데이터를 복호화하면 암호문 위치의 상위 4-byte에 CRC32 체크섬 값 이후에 압축 여부에 따라 압축 또는 원본 데이터가 존재한다. 따라서 암호문에 저장된 영역의 크기는 압축된 데이터 크기에 4-byte를 더한 값이다. 암호문 뒤에는 헤더 영역과 동일하

게 'FHT1' 시그니처 값이 저장된다. 복호화 이후 GZIP 압축 해제를 위해 다음의 데이터 형태로 저장해야 한다.

o GZIP file header signature(1F 8B 08 00 00 00 00 00 02 FF)||compressed data||CRC32 checksum||length of decompressed data

위의 데이터를 모두 연접하여 파일을 저장한 후 GZIP으로 데이터 압축을 해제하면 원본 데이터를 획득할 수 있다.

4.1.2 드라이브 백업 파일('.v2i' 확장자 파일)

암호화된 드라이브의 백업 데이터 구조는 크게 헤더와 암호화 영역으로 구성되어 있다. 암호화된 드라이브의 전체 데이터 구조는 Table 4.와 같다.

Table 4. Data structure of encrypted Drive Backup file

Offset	Field	Data
0~4	signature	"$CAN"
5~8	header and meta data length	header and meta data length
9~12	header length	header length
17~20	salt1	salt1 salt: salt1||salt2
25~28 37~40	password check value	password check value
29~32	block size	block size
33~36 49~52	counter generator	counter generator
41~48	salt2	salt2 salt: salt1||salt2
57~64	header start offset	header start offset
89~92	header checksum	CRC32
93~96	ciphertext checksum	CRC32
N	meta data and Ciphertext	backup info area

헤더 영역에는 암호화된 드라이브의 시그니처 값이 존재하며, 해당 값을 통해 데이터에 존재하는 각 블록을 구분할 수 있다. 첫 블록의 경우 헤더 영역 이후에 메타 데이터 영역이 존재하며, 암호화된 백업 드라이브의 GUID와 시그니처값 'Base', 백업된 PC 이름이 저장된다. 각 블록에 존재하는 헤더 영역의 크기, 메타 데이터 크기와 블록 크기 정보를 통해 데이터 위치를 확인할 수 있다. 또한, 데이터 암호키 생성에 사용되는 salt 값, 데이터 암호화에 사용되는 카운터 값과 사용자 입력 패스워드 검증에 사용되는 값 등 암호화 과정에 사용되는 다양한 정보가 저장된다. 암호화된 데이터 구조를 파악하여 암호화에 필요한 데이터와 실제 암호화하는 영역을 확인해야 한다.

4.2 암호화 과정 분석

Norton Ghost의 암호화 과정은 암호화에 사용되는 백업 데이터 암호키 생성과 백업 데이터 암호화 과정으로 구분된다. 데이터 암호화에는 AES 알고리즘과 커스텀된 AES-CTR 모드가 사용되며, 해당 모드에서 사용되는 카운터는 초기 카운터와 누적 카운터로 구성된다. 백업 데이터 암호화 과정은 초기 카운터와 누적 카운터 생성 이후에 이를 활용하여 데이터를 암호화한다. 누적 카운터 생성 과정을 제외하고 파일 및 디렉터리 백업 파일과 드라이브 백업 파일의 암호화 과정은 모두 동일하다.

4.2.1 커스텀된 AES-CTR 모드

일반적인 AES-CTR 모드의 암호화는 AES를 사용하여 데이터를 암호화하지 않고, 정해진 초기화 벡터(Initalization Vector, IV)에 카운터를 덧셈하여 암호화한 결과를 평문과 XOR 한다. 카운터는 한 블록을 암호화할 때마다 1씩 증가한다. 암호화되는 과정에는 블록끼리의 종속성이 없으며, 연산 형태는 다음과 같다.

$$c_i = m_i \oplus e_k(IV \oplus < i >_n) \quad (1)$$

각 블록에 해당하는 암호화 과정이며, 블록당 1씩 증가하는 값이 i이고, $< i >_n$는 i를 n-bit로 표현한 값이다. Norton Ghost는 이와 같은 AES-CTR 모드를 변형한 커스텀된 AES-CTR 모드를 사용하며, IV에 XOR되는 값에 차이가 일부 존재한다. n-bit로 표현한 기존의 방식을 $n/2$-bit로 나누어 $< i+1 >_{n/2} \,||\, < i >_{n/2}$로 계산하여 사용한다. $< i >_{n/2}$를 Norton Ghost의 0번째 블록이

라 할 때, 0번째 블록에서 2번째 블록까지 암호화하는 경우 $<i+1>_{n/2} || <i>_{n/2}$ 값의 변화는 Fig. 14.와 같다. 0번째 블록을 기준으로 다음 블록의 카운터는 little-endian 기준으로 1을 더하여 계산한다.

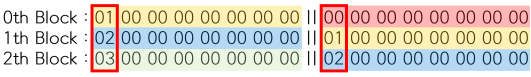

Fig. 14. Norton Ghost counter

4.2.2 백업 데이터 암호키 생성 과정

데이터 암호화에 사용되는 암호키는 사용자 입력 패스워드를 기반으로 하는 PBKDF2-HMAC-SHA1 알고리즘을 통해 생성한다. 사용자 입력 패스워드는 UTF-16으로 인코딩되며, salt 값은 암호화된 데이터 구조에 저장된 salt1과 salt2를 연접한 결과로 12-byte 크기의 데이터 값을 사용한다. 반복 횟수는 1,003으로 고정되며, 사용자가 지정한 암호화 강도에 따라 암호키 길이는 16, 24, 32 중 하나로 결정된다. 백업 데이터 암호키 생성 과정은 Fig. 15.와 같다.

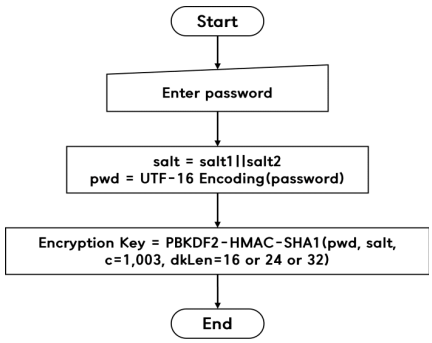

Fig. 15. Generating process of Backup data encryption key

4.2.3 백업 데이터 암호화 과정

Norton Ghost의 데이터 암호화에는 커스텀된 AES-CTR 모드가 사용된다. 해당 모드에서 사용되는 카운터는 초기 카운터라 칭하며, 암호키 생성 방식과 동일하게 초기 카운터 생성 과정에 PBKDF2-HMAC-SHA1 알고리즘이 사용된다. 또한, Norton Ghost는 누적 카운터가 사용되며, 이는 초기 카운터 생성 과정에 포함된다. Norton Ghost의 초기 카운터 생성 과정은 Fig. 16.과 같다.

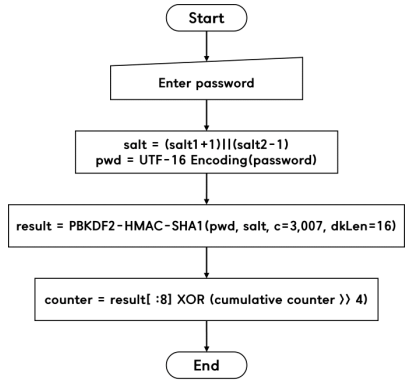

Fig. 16. Generating process of counter

UTF-16으로 인코딩된 사용자 입력 패스워드가 사용되며, salt 값은 little-endian 기준으로 salt1에는 1을 더하고, salt2에는 1을 뺀 값을 연접한 결과로 12-byte 크기의 데이터 값을 사용한다. 반복 횟수와 출력 결과는 3,007과 16-byte로 고정된다. PBKDF2-HMAC-SHA1 알고리즘으로 출력된 결과의 상위 8-byte와 누적된 카운터를 오른쪽으로 4만큼 shift 연산한 값을 XOR하면 초기 카운터가 생성된다.

누적 카운터 생성 방식은 파일 및 디렉터리 백업과 드라이브 백업에 따라 상이하다.

파일 및 디렉터리 백업 파일의 경우 헤더, 바디 영역의 구분자를 기준으로 이전 구분자까지 암호화한 블록 수×블록 크기(0x10)로 누적 카운터를 생성한다. 하나의 블록 크기는 16-byte로 헤더 및 바디 영역의 크기를 기준으로 계산한다. 헤더 영역을 복호화하는 경우 헤더 영역에 저장된 8-byte 크기의 카운터를 누적 카운터로 사용한다. 바디 영역을 복호화하는 경우 헤더 영역에 저장된 카운터 값, 헤더 영역의 블록 크기와 이전 바디 영역의 블록 크기 모두를 합한 값이 누적 카운터가 된다. 동시에 암호화하는 파일 개수와 각 파일의 헤더 및 바디 영역의 개수에 따라 누적 카운터 생성 방식이 결정되며, 순차적으로 카운터 값이 누적된다.

드라이브 백업 파일의 경우 헤더와 암호화 영역을 이루고 있는 데이터 구조가 '$CAN' 시그니처를 통해 구분된다. 따라서 개수에 따라 누적되는 카운터 값이 존재하지 않으며, 헤더 영역에 존재하는

8-byte 크기의 카운터를 4-byte 단위로 나누어 곱셈한 결과를 누적 카운터로 사용한다.

누적 카운터를 이용하여 초기 카운터를 생성한 후 각 블록 암호화에 사용되는 IV 값을 $(counter+1)||counter$ 형태로 생성한다. 초기 카운터는 8-byte 크기이며, 하나의 블록에 사용되는 IV 값은 16-byte 크기가 된다. 앞서 생성한 백업 데이터 암호키와 IV를 AES-CTR 모드로 암호화하여 생성된 키 스트림과 평문 또는 암호화 영역의 데이터를 XOR하는 것으로 암·복호화가 가능하다.

V. 사용자 패스워드 획득 방안

Cryptomator와 Norton Ghost는 데이터 암호화 시 사용자 패스워드를 필수적으로 요구한다. 이때 패스워드는 사용자 옵션에 따라 PC 내에 암호화되어 저장될 수 있다. 본 장에서는 암호키 생성 과정을 활용한 패스워드 검증방안과 DPAPI (Data Protection API)[24]를 활용한 암호화된 패스워드 획득 방안을 제시한다.

패스워드 검증 과정은 Nvidia 그래픽카드 Tesla A100-PCIE GPU 4대 기준으로 추정하였으며, DPAPI의 경우 DataProtectionDecryptor[25] 도구를 활용해 복호화를 진행하였다.

5.1 암호키 생성 과정을 활용한 패스워드 검증방안

Cryptomator와 Norton Ghost의 패스워드는 최소 8자리 이상이 요구된다. 본 논문에서는 [0-9a-zA-Z]의 62자와 자주 사용되는 특수문자 10가지를 선정하여 한 자리당 가능한 패스워드 수를 72자 기준으로 실험을 진행하였다.

Cryptomator는 사용자 입력 패스워드로부터 암호키를 생성하기 위해 Scrypt를 사용하며, Norton Ghost는 암호키와 카운터를 생성하기 위해 PBKDF2-HMAC-SHA1을 사용한다.

Cryptomaotr는 'masterkey.cryptomator' 파일을 통해 Scrypt 인자 값과 암호화된 MEK 값을 확인할 수 있다. ScryptCostParam과 Scrypt BlockSize는 32,768과 8로 고정된 값이 사용되며, salt는 vault에 따라 랜덤한 값이 사용된다. Scrypt를 통해 KEK가 생성되며, 암호화된 MEK를 KEK로 AES Key Unwrap 한다. 이때 unwrap된 결괏값이 해당 알고리즘에 사용되는 초기 IV 값 0xA6 A6 A6 A6 A6 A6 A6 A6일 경우 올바른 패스워드를 통해 KEK를 생성했음을 검증할 수 있다.

Norton Ghost는 반복 횟수가 1,003인 PBKDF2-HMAC-SHA1으로 암호키를 생성한다. 사용되는 salt는 헤더에 존재하는 salt1, 2를 연접한 값이며, 설정한 암호화 종류에 따라 암호키 길이가 달라진다. 이와 다르게 카운터 생성 과정에는 반복 횟수가 3,007인 PBKDF2-HMAC-SHA1이 적용되며, salt1에는 1을 더하고, satl2에는 1을 뺀 값을 연접하여 사용된다. 이때 생성되는 결괏값은 16-byte가 고정이며, 해당 데이터를 암호키를 사용하여 AES-CTR 모드로 암호화한다. 암호화한 결과의 상위 8-byte와 헤더에 저장되어 있는 패스워드 검증자와 비교하여 동일한 경우 올바른 패스워드임을 검증할 수 있다.

패스워드 전수조사에 필요한 시간은 가장 오래걸리는 연산인 Scrypt와 PBKDF2-HMAC-SHA1을 기반으로 하여 패스워드 복구 가능한 시간을 추정하였다. Norton Ghost의 경우 암호키와 카운터 생성의 반복 횟수가 다르므로 각각을 더하여 연산량을 계산한 후 시간을 추정하였다.

패스워드 측정에 드는 시간은 해시함수 기반 패스워드 복구 도구인 hashcat[26]을 활용하여 계산하였다.

검증 가능한 패스워드는 Cryptomator 기준 초당 $2^{27.4}$회이며, Norton Ghost 기준 초당 $2^{27.1}$회 연산이 가능하다[27]. 최소 패스워드 자릿수인 8자리와 9자리를 기준으로 하여 패스워드 복구에 필요한 시간은 다음 Table 5.와 같다.

Table 5. Estimation time for password recovery

Application	Password digits	Estimation time
Cryptomator	8	47 Days
	9	9.3 Years
Norton Ghost	8	58 Days
	9	11.5 Years

패스워드 자릿수가 8자리일 때 전수조사 하는 경우 약 47일과 58일의 시간이 필요하였으며, 한 자리가 증가할 때마다 약 72배의 시간이 증가하였다. 전수조사를 통한 패스워드 획득에 필요한 시간 계산량은 불가능한 수치이므로, 본 논문에서는 이와 다르게 각 응용프로그램의 기능을 사용한 패스워드 획득 방안을 제시한다.

5.2 Cryptomator 패스워드 획득 방안

Cryptomator는 vault 생성 이후, 잠금 해제 시 '비밀번호 기억' 옵션을 설정할 수 있다. 해당 옵션을 설정할 경우 '%APPDATA%\Cryptomator' 경로 내 'keychain.json' 파일에 사용자 패스워드가 DPAPI로 암호화되어 저장된다. 다음 Fig. 17.은 'keychain.json' 파일 내에 JSON 형태로 저장된 암호화된 패스워드다.

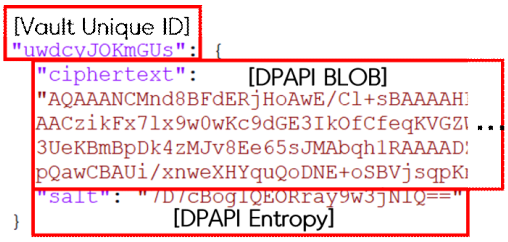

Fig. 17. keychain.json file structure

사용자가 생성한 vault에서 '비밀번호 기억' 옵션을 설정한 vault의 고유 ID가 저장되며, 해당 값은 9-byte로 랜덤하게 생성된다. 또한, 'ciphertext'와 'salt'에 해당하는 데이터가 존재하며, 각각 base64 디코딩 이후 DPAPI BLOB과 추가 엔트로피로 사용된다. DPAPI를 활용하여 암호화된 데이터를 복호화하면 Fig. 18.과 같이 저장된 사용자 패스워드를 획득할 수 있다.

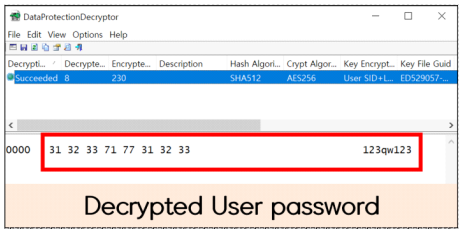

Fig. 18. Decrypted user password of Crypto mator using DPAPI

5.3 Norton Ghost 패스워드 획득 방안

Norton Ghost는 데이터 백업 시 '스케줄'을 설정해야 한다. 생성된 '스케줄'은 'ProgramData\Symantec\Norton Ghost\Schedule' 경로 내에 저장된다. 이때, 사용자 패스워드를 설정하는 경우 패스워드가 DPAPI로 암호화되어 '스케줄' 내에 저장된다. 다음 Fig. 19.는 스케줄 내에 저장된 암호화된 패스워드다.

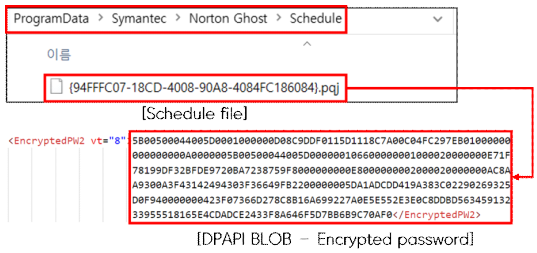

Fig. 19. Encrypted user password of Norton Ghost using DPAPI

DPAPI를 활용하여 암호화된 데이터를 복호화하면 Fig. 20.과 같이 암호화된 패스워드를 획득할 수 있다. DPAPI를 활용하기 위해서는 추가 엔트로피가 필요하며, 추가 엔트로피는 UTF-16으로 인코딩된 파일명이다.

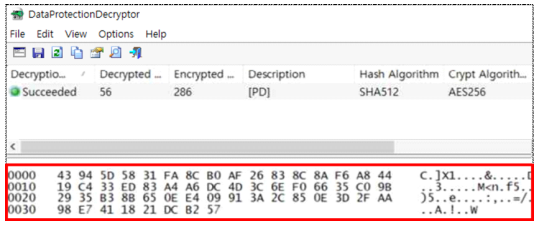

Fig. 20. Decrypted DPAPI BLOB of Norton Ghost using DPAPI

DPAPI로 복호화된 데이터는 RC4 스트림암호 알고리즘에 의해 암호화되어 있다. 암호화에 사용된 암호키는 '스케줄' 파일명을 UTF-16으로 인코딩하여 MD5 해싱한 결과의 상위 5-byte 값과 11-byte 크기의 NULL 값을 연접한 값이다. Fig. 20과 같이 DPAPI로 복호화한 데이터를 암호문으로 하여 RC4로 복호화한 후 UTF-16 디코딩한 결과 Fig. 21.과 같이 사용자 패스워드를 획득할 수 있다.

Fig. 21. Decrypted user password

VI. 결 론

디스크 암호화 기법을 제공하는 FDE 계열 응용프로그램은 실시간 암호화 또는 백업 기능을 제공하며 다양한 형태로 개발되고 있다. 이를 통해 사용자는 주요 데이터를 암호화하여 보호할 수 있다. 그러나 만일 FDE 계열 응용프로그램을 사용하여 범죄 사건에 관련된 주요 데이터를 암호화하는 경우, 디지털 증거 수집에 어려움이 발생한다. 따라서 FDE 계열 응용프로그램 사용 시 생성되는 암호화된 데이터 구조 파악 및 데이터 복호화를 위해 사용되는 암호화 알고리즘 분석과 암호키 생성 방안 연구가 선행되어야 한다.

본 논문에서는 볼륨 암호화 기능을 제공하는 2종의 FDE 계열 응용프로그램을 분석하였다. Cryptomator는 On-The-Fly Encryption 방식으로 데이터를 암호화하며, Norton Ghost는 데이터 백업 기능을 사용하여 데이터를 암호화한다. 각 애플리케이션에 따라 암호화된 데이터 구조를 파악하였고, 암호키 생성과 이를 이용한 암호화 과정을 분석하여 데이터 복호화 방안을 제시하였다. 분석 대상인 2개의 응용프로그램 모두 암호키 생성에 사용자 입력 패스워드가 필수적으로 요구된다. 이때, 특정 해시함수를 반복하는 사용자 패스워드 기반 데이터 암호키 생성 알고리즘을 사용하는 경우 패스워드 전수조사에 어려움이 존재한다. 이와 같이 암호화 과정에 필요한 패스워드 획득이 불가능하면 데이터를 복호화할 수 없으므로 이를 해결하기 위해 각 응용프로그램에 존재하는 기능을 사용하여 패스워드 획득 방안을 제시하였다. 획득한 사용자 패스워드와 분석한 암호화 과정을 통해 데이터 복호화가 가능하였다. 이와 같이 FDE 계열 응용프로그램에 따라 암호화된 데이터를 복호화하여 효율적인 데이터 수집이 가능하다. 또한, 증거 수집 및 분석에 소요되는 시간과 비용 절약으로 인해 디지털 포렌식 수사에 도움이 될 것으로 기대한다.

References

[1] South Korea Policy Briefing, "2022 Survey on Information Protection" https://www.korea.kr/briefing/pressReleaseView.do?newsId=156525374, 14 Jan 2023

[2] DIGITAL FORENSICS WIKIPEDIA, "Data Encryption" http://forensic.korea.ac.kr/DFWIKI/index.php/%EB%8D%B0%EC%9D%B4%ED%84%B0_%EC%95%94%ED%98%B8%ED%99%94#cite_note-5, 03 Feb 2023

[3] WIKIPEDIA, "CrossCrypt" https://en.wikipedia.org/wiki/CrossCrypt, 04 Feb. 2023

[4] WIKIPEDIA, "Disk encryption" https://en.wikipedia.org/wiki/Disk_encryption, 03 Feb. 2023

[5] "Unlock the code that the Druking group hung... Special prosecutor mobilizes all experts", YONHAP NEWS, 18 July 2018

[6] "Brazilian banker's crypto baffles FBI", The Register, 28 June 2010

[7] E. Casey, G. Fellows, M. Geiger and G. Stellatos, "The growing impact of full disk encryption on digital forensics," Digital Investigation, vol. 8, no. 2, pp. 129-134, Nov. 2011.

[8] T. Müller and F. C. Freiling, "A Systematic Assessment of the Security of Full Disk Encryption," IEEE Transactions on Dependable and Secure Comput

ing, vol. 12, no. 5, pp. 491-503, Sept.-Oct. 2015.
[9] Seunghee Seo, Changhoon Lee, "Trend of Android FDE·FBE Decryption Research," Journal of The Korea Institute of Information Security and Cryptology, 29(6), pp. 5-11, Dec. 2019.
[10] Sungmin Jang, Jungheum Park, Chanung and SangJin Lee, "The Research for Digital Evidence Acquisition Procedure within a Full Disk Encryption Environment," Journal of The Korea Institute of Information Security and Cryptology, 25(1), pp. 39-48, Feb. 2015.
[11] L. Zhang, Y. Zhou and J. Fan, "The forensic analysis of encrypted Truecrypt volumes," 2014 IEEE International Conference on Progress in Informatics and Computing, pp. 405-409, May. 2014.
[12] L. Zhang, X. Deng, C. Tan, "An Extensive Analysis of TrueCrypt Encryption Forensics," CSAE '19: Proceedings of the 3rd International Conference on Computer Science and Application Engineering, pp. 1-6, Oct. 2019.
[13] C. Tan, L. Zhang and L. Bao, "A Deep Exploration of BitLocker Encryption and Security Analysis," 2020 IEEE 20th International Conference on Communication Technology (ICCT), pp. 1070-1074, Oct. 2020.
[14] A. Kazim, F. Almaeeni, S. A. Ali, F. Iqbal and K. Al-Hussaeni, "Memory Forensics: Recovering Chat Messages and Encryption Master Key," 2019 10th International Conference on Information and Communication Systems (ICICS), pp. 58-64, June. 2019.
[15] J.Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten, "Lest We Remember: Cold-Boot Attacks on Encryption Keys," Communications of the ACM, vol. 52, no. 5, pp. 91-98, May. 2009.
[16] V. Firdaus, D. Suprianto, R. Agustina, "Analisis Forensik Digital Memori Volatile untuk Mendapatkan Kunci Enkripsi Aplikasi Dm-Crypt," Jurnal Sistem Komputer dan Informatike (JSON), vol. 2, no. 3, May 2021.
[17] CRYPTOMATOR, "Cryptomator" https://cryptomator.org/, 18 Dec. 2022
[18] M. Jones, J. Bradley and N. Sakimura, "JSON Web Token (JWT)," May. 2015.
[19] D. Harkins, "Synthetic Initialization vector (SIV) Authenticated Encryption Using the Advanced Encryption Standard (AES)," Oct. 2008.
[20] C. Percival and S. Josefsson. "The Scrypt Password-Based Key Derivation Function," Aug. 2016.
[21] S, Jim, and R. Housley. "Advanced Encryption Standard (AES) Key Wrap Algorithm," Sept. 2002.
[22] norton, "Norton Ghost has been discontinued" https://community.norton.com/en/forums/norton-ghost-has-been-discontinued, 18 Dec. 2022
[23] P.Deutsch, "DEFLATE Compressed Data Format Specification version 1.3," May. 1996.
[24] Microsoft, "How to: Enable data Protection" https://docs.microsoft.com/ko-kr/dotnet/standard/security/how-to-use-data-protection, 13 Jan. 2022
[25] NirSoft, "DataProtectionDecryptor", https://www.nirsoft.net/utils/dpapi_data_decryptor.html, 13 Jan. 2022
[26] hashcat, "advanced password recovery" https://hashcat.net/hashcat/, 16 Jan. 2022
[27] Github, "TESLA_A100_PCIE_v6.1.1" https://gist.github.com/Chick3nman/d65bcd5c137626c0fcb05078bba9ca89, 16 Jan. 2022

〈저자소개〉

박 귀 은 (Gwui-eun Park) 학생회원
2022년 2월: 국민대학교 정보보안암호수학과 졸업
2022년 3월~현재: 국민대학교 금융정보보안학과 석사과정
〈관심분야〉 디지털 포렌식, 정보보호

이 민 정 (Min-jeong Lee) 학생회원
2022년 2월: 국민대학교 정보보안암호수학과 졸업
2022년 3월~현재: 국민대학교 금융정보보안학과 석사과정
〈관심분야〉 디지털 포렌식, 정보보호

강 수 진 (Soo-jin Kang) 학생회원
2018년 2월: 국민대학교 정보보안암호수학과 졸업
2022년 2월: 국민대학교 금융정보보안학과 석사
2022년 3월~현재: 국민대학교 금융정보보안학과 박사과정
〈관심분야〉 디지털 포렌식, 정보보호

김 기 윤 (Gi-yoon Kim) 학생회원
2019년 2월: 국민대학교 정보보안암호수학과 졸업
2019년 3월~현재: 국민대학교 금융정보보안학과 석박통합과정
〈관심분야〉 디지털 포렌식, 정보보호

김 종 성 (Jong-sung Kim) 종신회원
2006년 11월: K.U.Leuven, ESAT/SCD-COSIC 정보보호 공학박사
2007년 2월: 고려대학교 정보보호대학원 공학박사
2009년 9월~2013년 2월: 경남대학교 e-비즈니스학과 교수
2013년 3월~2017년 2월: 국민대학교 수학과 교수
2017년 3월~현재: 국민대학교 정보보안암호수학과/금융정보보안학과 교수
〈관심분야〉 정보보호, 암호 알고리즘, 디지털 포렌식