

# Forensic Detection of Timestamp Manipulation for Digital Forensic Investigation

By JUNGHOON OH, SANGJIN LEE, And HYUNUK HWANG

*Digital Object Identifier 10.1109/ACCESS.2024.3395644*

2024.11.11 (Mon)

Min-hyuk Cho in Mobile System Eng.

# INDEX

- 
- 01 Introduction
  - 02 Background Knowledge
  - 03 Proposed Detection Algorithm
  - 04 Implementation and Performance Evaluation
  - 05 Conclusion
-



**01**

# Introduction

# Introduction

## ✓ 디지털 포렌식?

: 정보의 무결성을 보존하고 데이터에 대한 엄격한 보관 체계를 유지하면서 데이터를 식별, 수집, 검사 및 분석하는 데 **과학적 방법**을 적용 하는 것 - NIST SP 800-86



## ✓ 안티 포렌식?

: 데이터를 은폐 또는 파괴하여 접근하지 못하도록 하는 기술 및 행위 - NIST SP 800-86



## ✓ 타임스탬프?

: 로그 생성 시 언제 (몇 시, 몇 분, 몇 초에) 생성되었는 지 알려주는 정보

: 디지털 포렌식 분석에 있어서 가장 기본적이면서 중요한 정보

=> “범죄자는 타임스탬프를 조작하여 포렌식 수사를 어렵게 할 수 있다”



## 본 논문의 필요성

“파일 타임스탬프 조작을 효과적으로 탐지하기 위한 알고리즘 제안 필요”

## 본 논문의 목표

“NTFS 파일 시스템에서 발생하는 타임스탬프 조작을 효과적이고 정확하게 탐지하는 알고리즘을 개발한다”

“개발한 알고리즘을 바탕으로 많은 시나리오에 대해 타임스탬프 조작 탐지 성능을 검증한다”



**02**

# Background Knowledge

# Background Knowledge

## ✓ NTFS (New Technology File System)

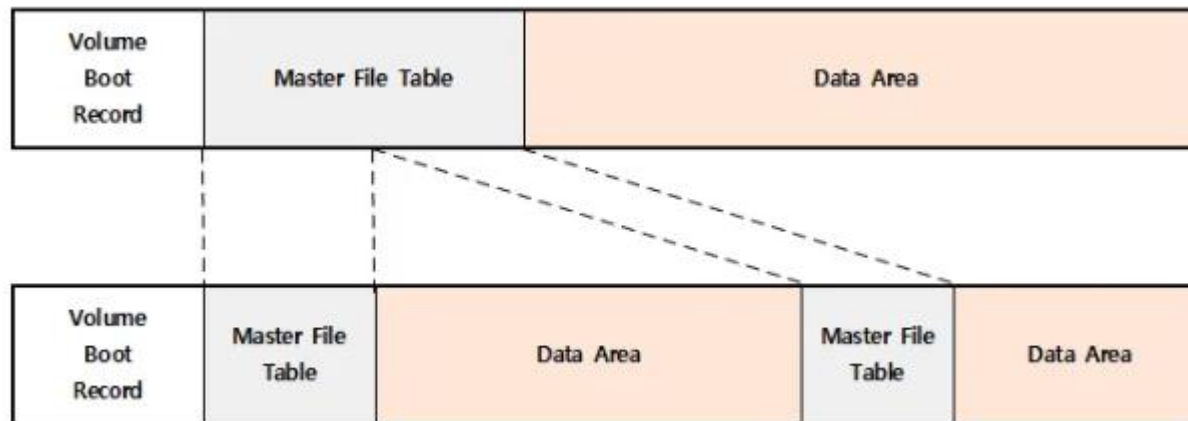
: Windows의 표준 파일 시스템

: 파일 시스템에 존재하는 모든 파일과 디렉터리에 대한 메타데이터 정보 저장 - MFT

: 저널링 기능을 통해 파일 시스템의 무결성을 유지 - \$LogFile, \$UsnJrnl

: 타임스탬프 정보

- M (Modified) : 파일 내용이 마지막으로 수정된 시간
- A (Accessed) : 파일이 마지막으로 접근된 시간
- C (Created) : 파일이 생성된 시간
- E (Entry Modified) : \$MFT 엔트리가 업데이트된 시간





# Background Knowledge

## ✓ MFT (Master File Table)

: 파일 시스템에 존재하는 모든 파일과 디렉터리에 대한 정보 저장

: MFT는 엔트리 단위로 구성

: 파일과 폴더의 메타데이터는 하나 이상의 엔트리에 저장

: 각 엔트리는 속성 단위로 구성

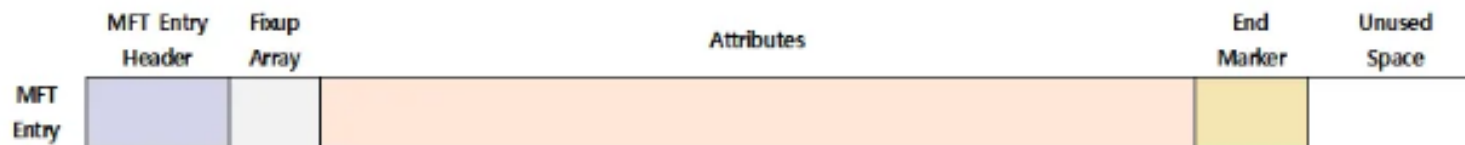
## ✓ \$STANDARD\_INFORMATION

: 윈도우 탐색기에서 확인 가능한 파일 및 폴더의 시간 정보

## ✓ \$FILE\_NAME

: 파일 이름 또는 위치가 변경될 때 시간 정보 갱신

Entry Number	Entry Name	Description
0	\$MFT	NTFS에서의 모든 파일 들의 MFT Entry 정보를 담고 있다.
1	\$MFTMirr	\$MFT 파일의 일부 백업 데이터를 담고 있다.
2	\$LogFile	Meta Data의 트랜잭션 저널 정보를 담고 있다.
3	\$Volume	볼륨의 레이블, 식별자, 버전 등의 여러 정보를 담고 있다.
4	\$AttrDef	속성의 식별자, 이름, 크기, 등의 정보를 담고 있다.
5	.	볼륨의 루트 디렉터리 정보를 담고 있다.
6	\$Bitmap	볼륨의 클러스터 할당 정보를 담고 있다.
7	\$Boot	볼륨이 부팅 가능할 경우 부트 섹터 정보를 담고 있다.
8	\$BadClus	배드 섹터를 가지는 클러스터의 정보를 담고 있다.
9	\$Secure	파일의 보안, 접근 제어와 관련된 정보를 담고 있다.
10	\$Upcase	모든 유니코드 문자의 대문자
11	\$Extend	\$ObjID, \$Quota, \$Reparse points, \$UsnJrnl 등의 추가적인 파일의 정보를 기록하기 위해서 사용하는 엔트리
12~15		미래를 위해서 예약된 영역으로 남겨둠
16~		포맷 후 생성되는 파일의 정보를 위해서 사용
-	\$ObjID	파일 고유의 ID정보를 담고 있다.
-	\$Quota	사용량의 정보를 담고 있다.
-	\$Reparse	Reparse Point에 대한 정보를 담고 있다.
-	\$UsnJrnl	파일, 디렉터리의 변경 정보를 담고 있다.



## ✓ \$LogFile

: 어떤 \$MFT 엔트리의 어느 속성의 어느 위치에서 어느 정도의 데이터가 업데이트 되었는지 반영

: 변경 전/후를 기록

## ✓ \$UsnJrnl

: NTFS 변경 로그를 기록

: 파일 시스템에서 발생하는 모든 변경 이벤트를 기록

: 변경 원인에 대한 기록

### 배경 지식 정리

“\$LogFile & \$UsnJrnl : 타임스탬프 변경 작업이 발생 했는 지 확인 가능”

“\$STANDARD\_INFORMATION & \$FILE\_NAME : 타임스탬프 조작 가능성 확인 가능”



**03**

# Proposed Detection Algorithm

# Proposed Detection Algorithm - 1

---

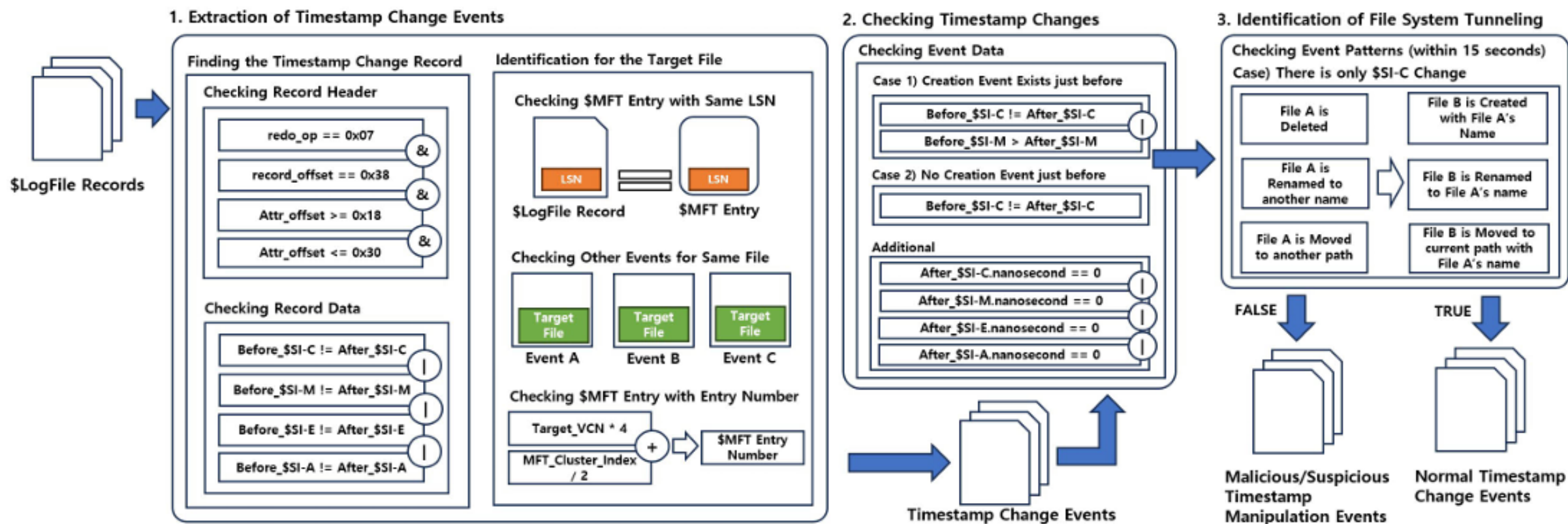
## ✓ Detection Method with \$LogFile

STEP 1) \$LogFile에서 타임스탬프 변경 이벤트 발생 파일 찾아 추출

STEP 2) 추출 내용이 타임스탬프 조작에 의한 것인지, 단순 변경인 것인지 파악

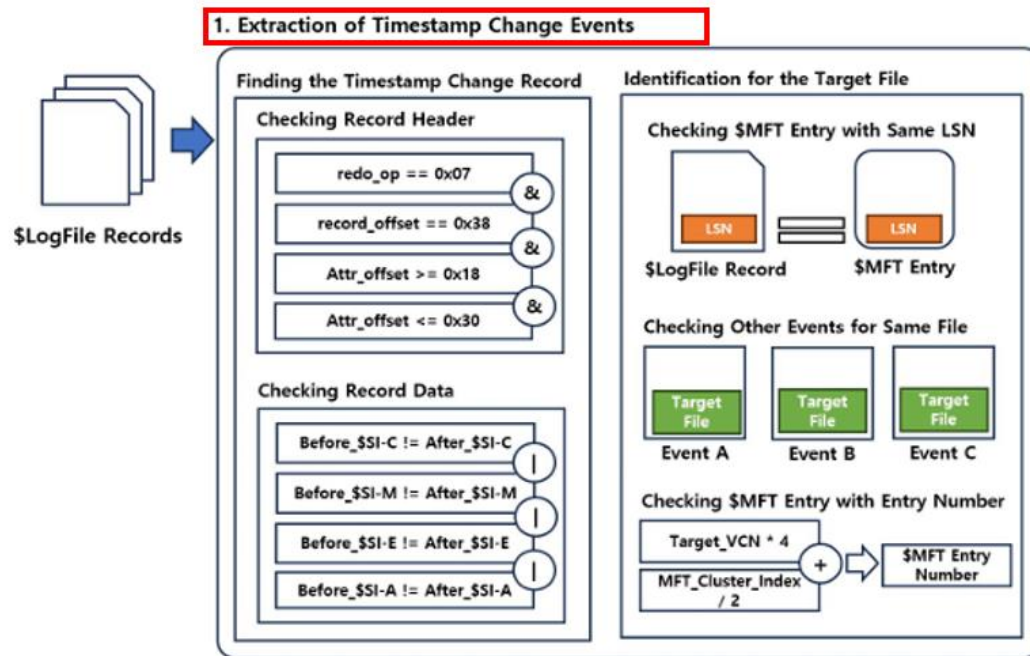
STEP 3) 파일 시스템 터널링 여부 확인

# Proposed Detection Algorithm - 1



**FIGURE 4.** Overall procedure of the advanced \$LogFile-based timestamp manipulation detection method.

# STEP 1 of Proposed Detection Algorithm



STEP 1) \$LogFile에서 타임스탬프 변경 이벤트 발생 파일 찾아 추출

: Timestamp 기록 부분 가져오기

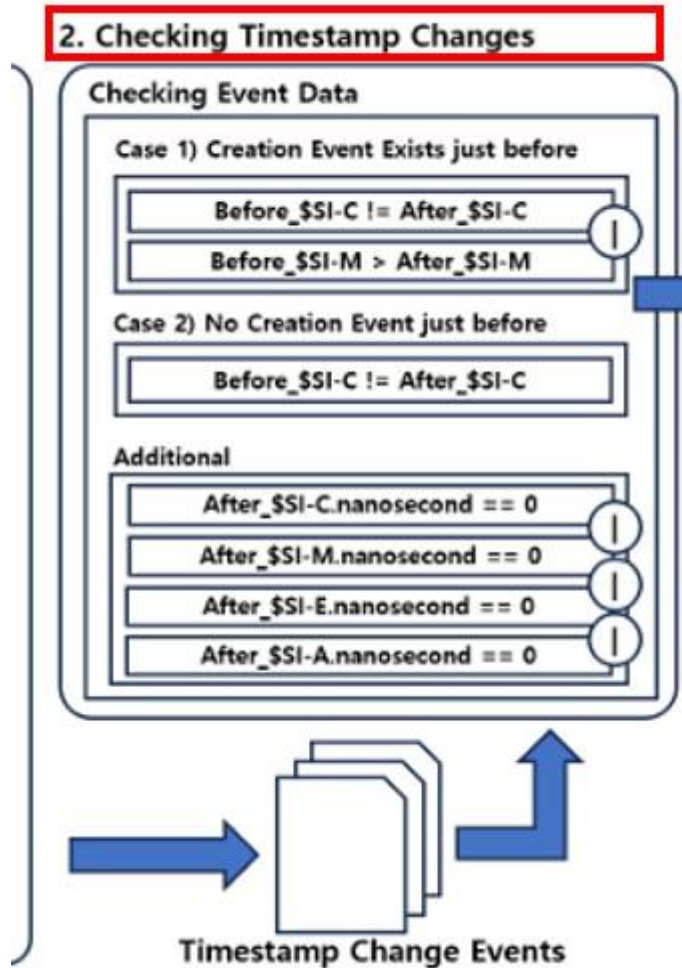
: Timestamp 변경 이력 있나 확인

: 이후, LSN을 확인하여 해당 파일의 MFT Entry 확인

: 해당 파일에 대해 관련 이벤트 수집

\* 또는 Entry Number 계산하여 MFT Entry 확인 가능

## STEP 2 of Proposed Detection Algorithm



STEP 2) 추출 내용이 타임스탬프 조작에 의한 것인지, 단순 변경인 것인지 파악

\* CASE 1) 직전에 파일 생성 이벤트가 있는 경우

: 이전 생성 시간과 이후 생성 시간 비교

: 이전 수정 시간과 이후 수정 시간 비교

\* CASE 2) 직전에 파일 생성 이벤트 없는 경우

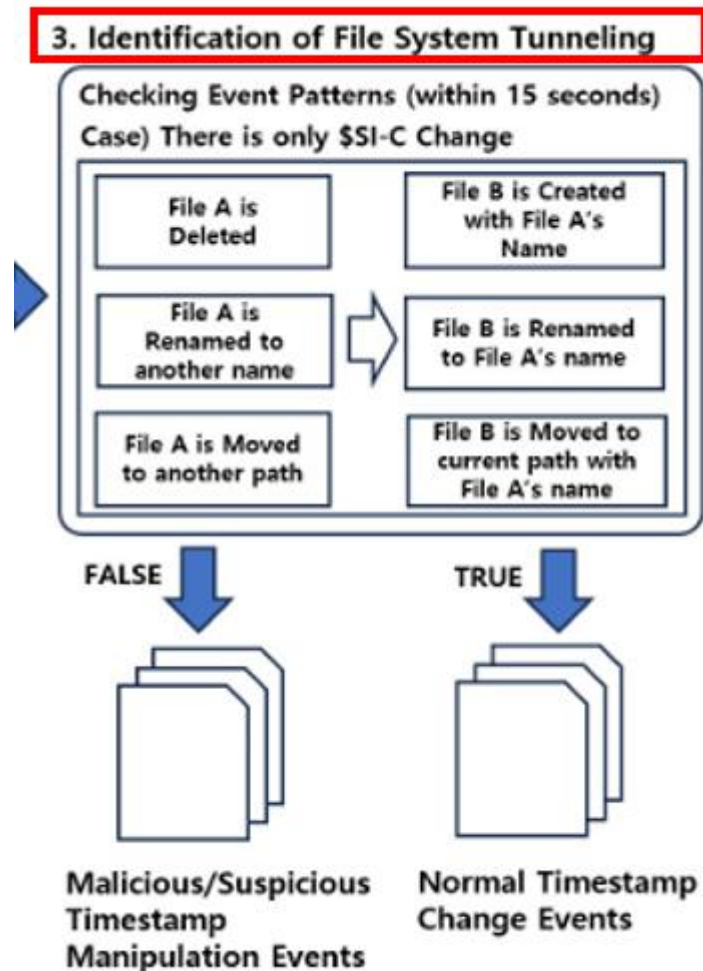
: 이전 생성 시간과 이후 생성 시간 비교

\* 추가 고려 사항

: \$\$I의 모든 시간 정보의 nanosecond가 0으로 세팅 시 시간 조작으로 판단



# STEP 3 of Proposed Detection Algorithm



## STEP 3) 파일 시스템 터널링 여부 확인

\* 3가지 실험을 각각 15초 이내로 수행

- 1) File A 삭제 후 File B를 File A의 이름으로 생성
- 2) File A의 이름 변경 후 File B를 File A의 이름으로 변경
- 3) File A의 경로 변경 후 File B를 File A의 이전 경로로 이동

=> 만약 True 시, 터널링에 의한 것으로 시간 조작 x

=> 만약 False 시, 시간 조작에 의한 것으로 판단

## Proposed Detection Algorithm - 2

---

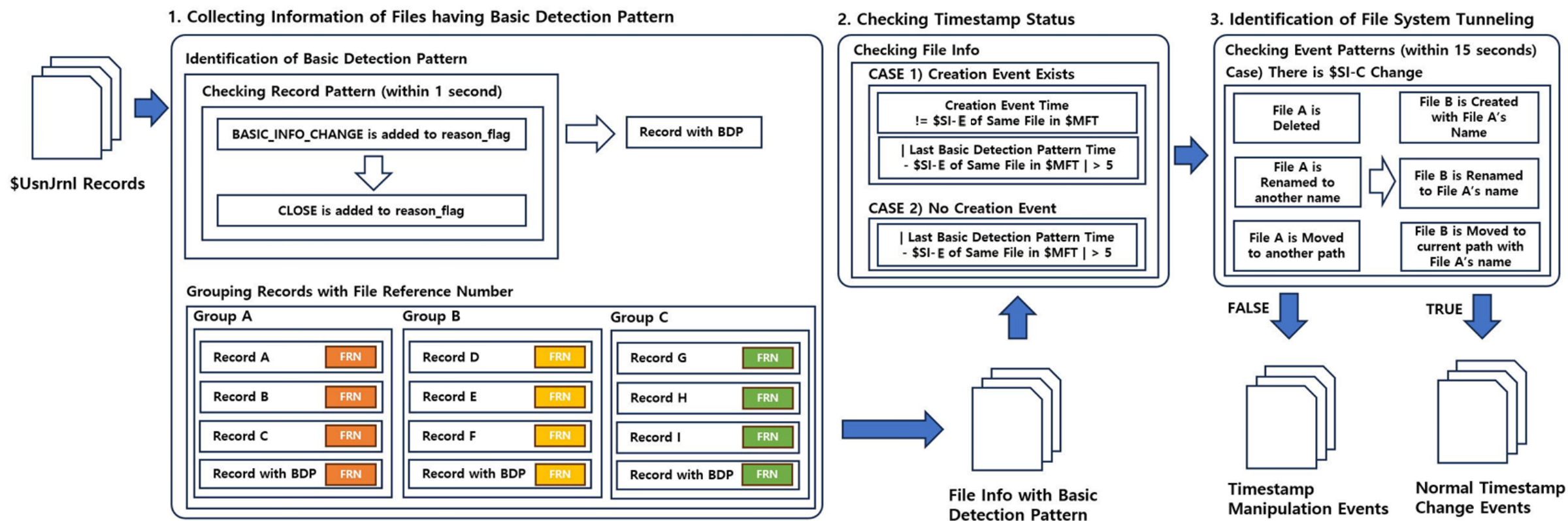
### ✓ Detection Method with \$UsnJrnl

STEP 1) 기본 탐지 패턴을 가진 파일 정보 수집

STEP 2) 비이상적 패턴 발생하는지 파악

STEP 3) 파일 시스템 터널링 여부 확인

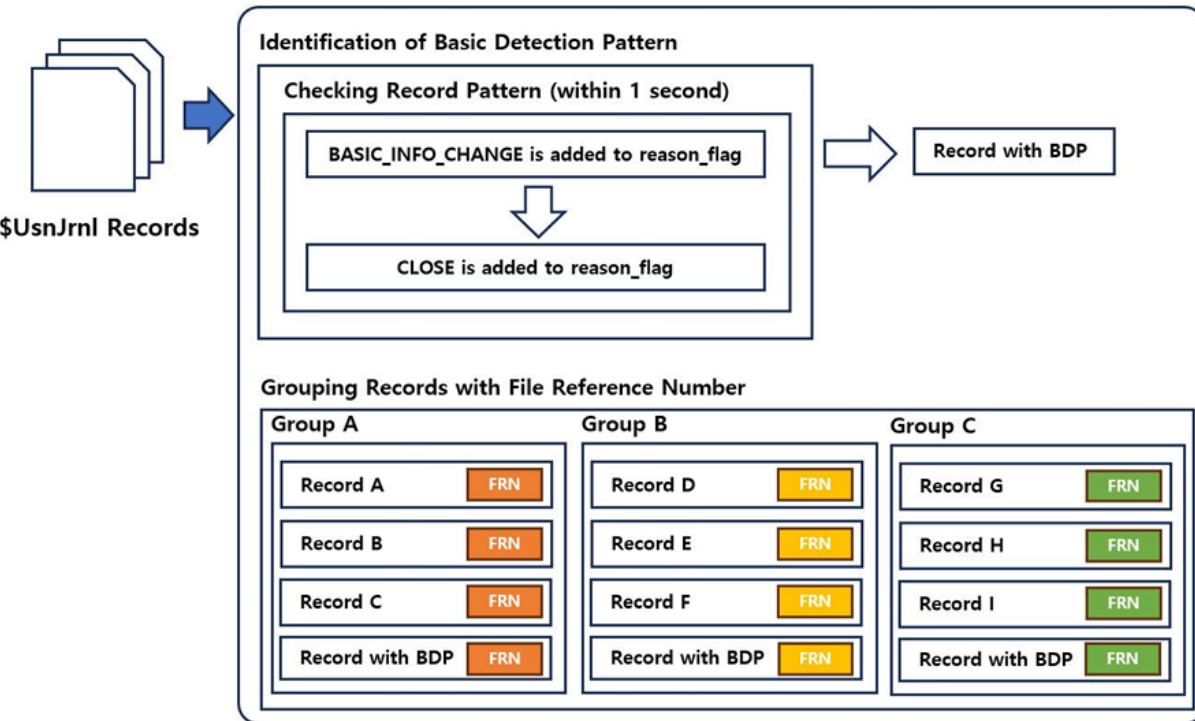
# STEP 1 of Proposed Detection Algorithm - 2



**FIGURE 5.** Overall procedure of the advanced \$UsnJrnl-based timestamp manipulation detection method.

## STEP 2 of Proposed Detection Algorithm - 2

### 1. Collecting Information of Files having Basic Detection Pattern



STEP 1) 기본 탐지 패턴을 가진 파일 정보 수집

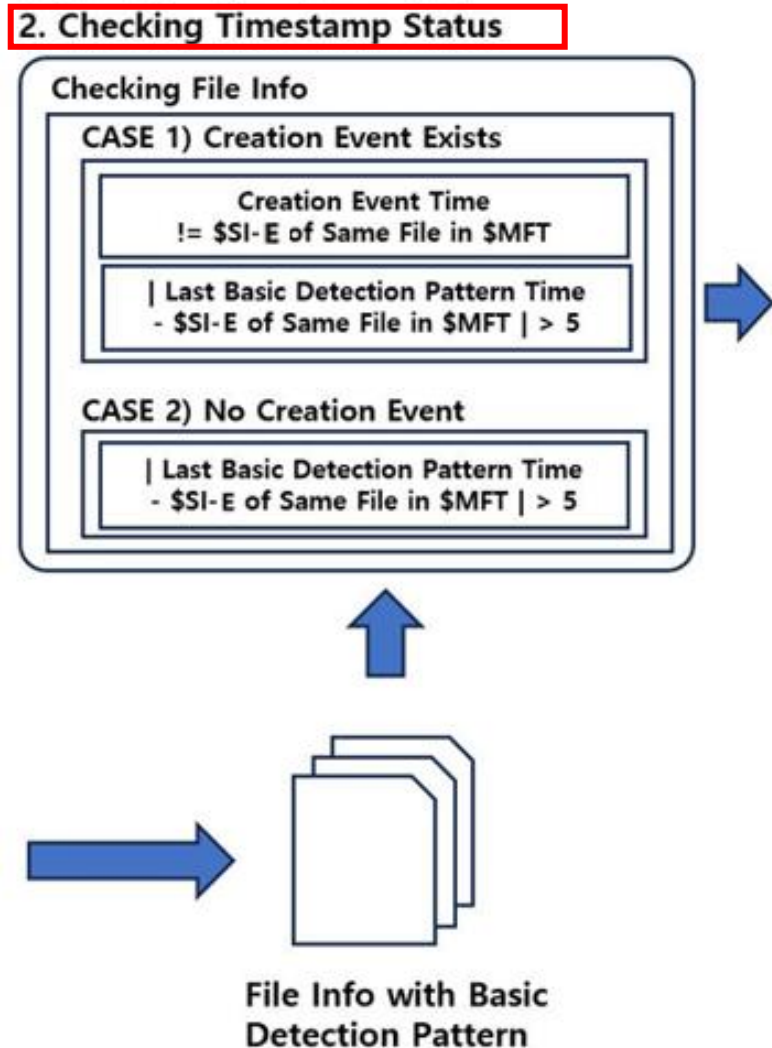
: BASIC\_INFO\_CHANGE가 reason\_flag에 추가되는 시간과

: CLOSE가 reason\_flag에 추가되는 시간 비교

=> 1초 이내면 기본 탐지 패턴에 추가

: 이후, 해당 패턴들의 정보 수집

## STEP 3 of Proposed Detection Algorithm - 2



### STEP 2) 비이상적 패턴 발생 파악

\* CASE 1) 파일 생성 이벤트 존재하는 경우

-> 해당 이벤트 타임스탬프와 \$SI-E를 비교

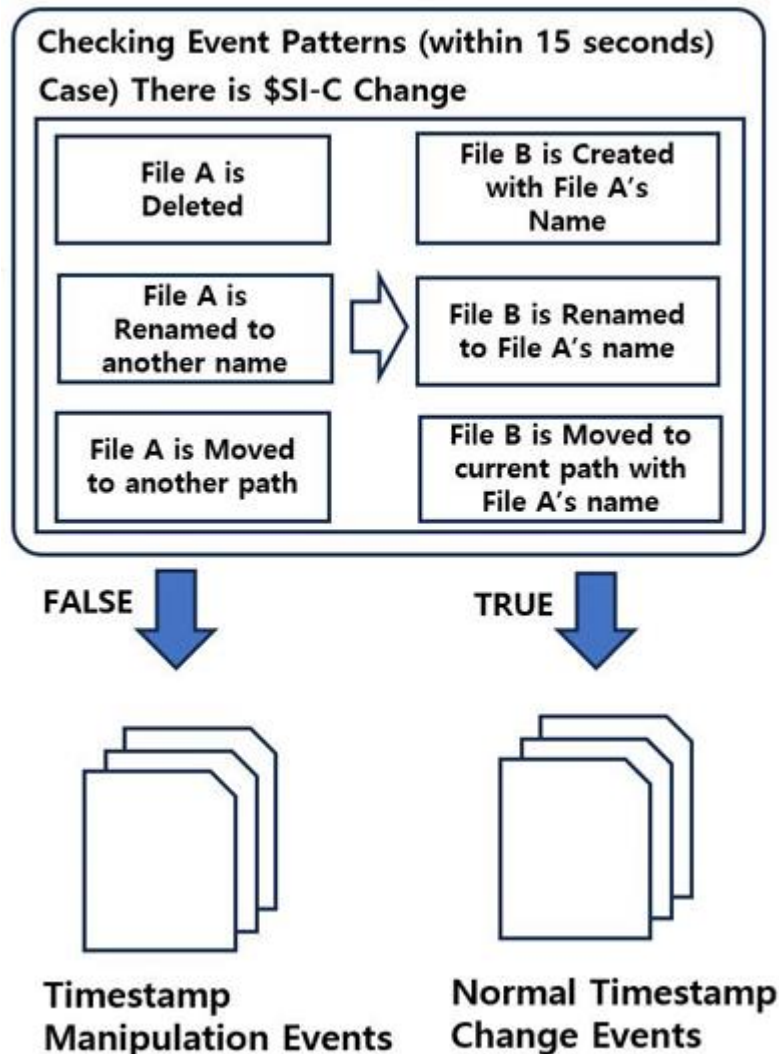
-> 기본 탐지 패턴 발생 시간과 \$SI-E의 차이가 5초를 초과하는지 확인

\* CASE 2) 파일 생성 이벤트 존재하지 않는 경우

-> 기본 탐지 패턴 발생 시간과 \$SI-E를 비교

# Proposed Detection Algorithm - 2

## 3. Identification of File System Tunneling



### STEP 3) 파일 시스템 터널링 여부 확인

\* 3가지 실험을 각각 15초 이내로 수행

- 1) File A 삭제 후 File B를 File A의 이름으로 생성
- 2) File A의 이름 변경 후 File B를 File A의 이름으로 변경
- 3) File A의 경로 변경 후 File B를 File A의 이전 경로로 이동

=> 만약 True 시, 터널링에 의한 것으로 시간 조작 x

=> 만약 False 시, 시간 조작에 의한 것으로 판단

# 04

## Implementation and Performance Evaluation



# Implementation – NTFS log tracker v1.9

\$LogFile

\$UsnJrnl:\$J

\$LogFile(Search Result)

\$UsnJrnl:\$J(Search Result)

Suspicious Behavior Detection

Detection Overview

No	Category	Count	Source
1	Timestamp Manipulation	3	\$UsnJrnl:\$J

Detection Detail

No	Detection Location (USN)	Detail
1	9208803944	[Malicious] The timestamp of "timestamp_all time_change.txt" have been manipulated after executing TimeStor
<		>

Detection Location in Log

TimeStamp(UTC+9)	USN	File/Directory Name	Full Path(from \$MFT)	Event
2022-07-04 01:04:27	9208803824	timestamp_all time_change.txt	\\Users\\[REDACTED]\\Desktop\\Time Change Test\\timestamp_all time_change.txt	Basic_In
2022-07-04 01:04:27	9208803944	timestamp_all time_change.txt	\\Users\\[REDACTED]\\Desktop\\Time Change Test\\timestamp_all time_change.txt	Basic_In
2022-07-04 01:04:27	9208804064	TIMESTAMP_EYE_C092E0B1_6F	\\Users\\[REDACTED]\\Desktop\\Time Change Test\\TIMESTAMP_EYE_C092E0B1_6F	Basic_In
<		>		

**FIGURE 8.** Result of timestamp manipulation detection in NTFS log tracker v1.9.



# Evaluation – Data Set


TABLE 6. Data sets applied by test scenarios.

No	Tools / Method	Test Scenarios
1	NewFileTime v6.77	Manipulate \$SI-C of NewFileTime_SI_C_Manipulation.dll to the past (0 in 100-nanosecond unit)
2	(SetFileTime() API)	Manipulate \$SI-M of NewFileTime_SI_M_Manipulation.dll to the past (0 in 100-nanosecond unit)
3		Manipulate \$SI-MAC of NewFileTime_SI_MAC_Manipulation.dll to the past (0 in 100-nanosecond unit)
4	Windows PowerShell	Manipulate \$SI-C of PowerShell_SI_C_Manipulation.dll to the past
5	(Get-Item Cmdlet)	Manipulate \$SI-M of PowerShell_SI_M_Manipulation.dll to the past
6		Manipulate \$SI-MAC of PowerShell_SI_MAC_Manipulation.dll to the past
7	nTimestamp	Manipulate \$SI-C of nTimestamp_SI_C_Manipulation.dll to the past
8	(NtSetInformationFile() API)	Manipulate \$SI-M of nTimestamp_SI_M_Manipulation.dll to the past
9		Manipulate \$SI-MACE of nTimestamp_SI_MACE_Manipulation.dll to the past
10	File System Tunneling	Change \$SI-C of FileSystemTunneling_SI_C_Change.dll by causing file system tunneling
11	SetMACE v1.0.0.4 (NtSetInformationFile() API)	Manipulate \$SI-MACE of SetMACE_SI_MACE_Copy_Manipulation.dll using the \$SI-MACE of another file in the same path
12		Manipulate \$SI-MACE and \$FN-MACE of SetMACE_SI_FN_MACE_Manipulation.dll to the past

- 1) No.1~3: \$SI-C, \$SI-M, \$SI-MAC (현재 -> 과거, 100-나노초 = 0)
- 2) No.4~6: \$SI-C, \$SI-M, \$SI-MAC (현재 -> 과거)
- 3) No.7~9: \$SI-C, \$SI-M, \$SI-MACE (현재 -> 과거)
- 4) No.10: 파일 시스템 터널링으로 인한 조작
- 5) No.11: 동일 경로에 있는 \$SI-MACE로 실험 파일 조작 (현재 -> 과거)
- 6) No.12: \$SI와 \$FN도 조작 (현재 -> 과거)

**TABLE 7.** Performance evaluation results.

Datasets	Detection Target	Program A	Program B	NTFS Log Tracker v1.9 with \$LogFile	Program C	NTFS Log Tracker v1.9 with \$UsnJrnl
1	SSI-C manipulation with SetFileTime()	O	O	O	X	O
2	SSI-M manipulation with SetFileTime()	X	X	O	X	X
3	SSI-MAC manipulation with SetFileTime()	O	O	O	X	O
4	SSI-C manipulation with PowerShell	O	O	O	X	O
5	SSI-M manipulation with PowerShell	X	X	O	X	X
6	SSI-MAC manipulation with PowerShell	O	O	O	X	O
7	SSI-C manipulation with NtSetInformationFile()	O	O	O	O	O
8	SSI-M manipulation with NtSetInformationFile()	X	X	O	O	O
9	SSI-MACE manipulation with NtSetInformationFile()	O	O	O	O	O
10	File System Tunneling	X	X	O	X	O
1-3	Manipulation with 0 in 100-nanoseconds	X	X	O	X	O
11	Manipulation using another file's timestamp	X	X	O	X	O
12	\$FN timestamp manipulation using file move	X	X	O	X	O
1-3, 7-9, 11-12	Execution of well-known timestamp manipulation tool	X	X	O	X	O



**05**

# Conclusion

# Conclusion

---

## ✓ Conclusion

: 세계적으로 많이 쓰이는 NTFS에서의 타임스탬프 조작 탐지 알고리즘을 제안

: 제안된 알고리즘을 기반으로 도구 제작 후 다양한 시나리오에 대해 검증

: 제작된 도구는 다양한 시나리오에 대해 올바른 탐지를 할 수 있었고, 오탐을 줄일 수 있었음

=> “제시된 알고리즘을 통해 디지털 포렌식 조사 과정에서 조사관들이 파일 타임스탬프 조작을 탐지하는 데 기여”

감사합니다

---

# QnA

---