KAES4 (4세대 표준 암호화) 형식 기술 설명서

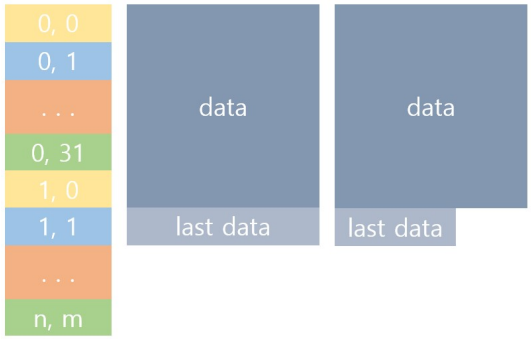
1 형식 구조

- 0/128B 배수
- 8B
- -> 5B KAES4
- -> 3B 헤더 사이즈
- 분할된 데이터들

- pre header는 png를 0으로 패딩한 것.
- magic num : b'KAES4' + main header 사이즈가 3 바이트 인코딩되어 들어감.
- 헤더는 '이름(타입)값\n'의 연속으로 되어 있으며, 전부 대문자로만 이루어진 문자열 기반 구조임.

일반 암호화에 쓰이는 전체 모드와 암호화 저장소에 쓰일 기능적 모드가 존재.
전체 모드의 헤더에는 MODE, SALT, PWH, CKDT, HINT, (TKDT, NMDT)가 존재.
기능적 모드의 헤더에는 MODE, CKDT가 존재.

비밀번호 흐름 : salt는 32바이트 난수. pw + kf + salt -> master key 48B, pw hash 256B. content key는 1536바이트 난수, title key는 48바이트 난수. mkey로 각각을 암호화해 ckeydata 1536B, tkeydata 48B를 생성. content key는 실제 내용을 암호화하고, title key는 파일 이름을 암호화함.



- 32스레드 병렬, 청크 크기는 128KB로 고정.
- 4MB 이상부터 32스레드 전체를 사용하기 시작.
- 데이터를 분할시켜 32개의 독립적인 암호화 흐름을 만들어 병렬화함.

core : 32, chunk size : 131072 고정.
AES의 16 바이트 패딩은 마지막 청크에만 적용.
마지막 청크의 크기는 chunk size와 같거나 작을 수 있고, 커질 수는 없음.
패딩된 마지막 청크의 최소 사이즈는 16 바이트.
각 core number 별로 독립된 암호화를 수행하는 것, iv는 number별 일렬로 이어짐.

2 모듈 사용법

===> 기본 제공 함수 (클래스 선언 없이 바로 사용 가능)

===> getkf(path) : (str) -> bytes
path 경로의 파일을 읽어 키 파일 바이트로 반환, 불가능하면 기본키파일 반환.

===> 클래스 제공 함수 (toolbox 클래스 선언하여 사용)

===> 클래스 내부변수 : self.valid - 모듈 유효성, self.assist - 네이티브 가속 여부

===> encode(number, length) : (int, int) -> bytes
number를 길이 length의 바이트로 리틀 엔디안 인코딩.

===> decode(binary) : (bytes) -> int
binary를 숫자로 리틀 엔디안 디코딩.

===> rdhead(header) : (str) -> dict
헤더 문자열을 읽고 {이름 : 값}의 딕셔너리를 반환. 이름으로는 대문자 영문만 사용 가능. 값의 타입은 문자열, 바이트, 정수형, 실수형 지원.

===> mkhead(values) : (dict) -> str
{이름 : 값}의 딕셔너리를 받아 헤더 문자열 생성. 이름과 값의 제한 조건은 위와 동일.

===> enshort(key, iv, data) : (bytes 32B, bytes 16B, bytes 16nB) -> bytes 16nB
입력된 key, iv를 이용하여 패딩 없이 data를 암호화. data는 16의 배수 길이여야 함.

===> deshort(key, iv, data) : (bytes 32B, bytes 16B, bytes 16nB) -> bytes 16nB
입력된 key, iv를 이용하여 패딩 없이 data를 복호화. data는 16의 배수 길이여야 함.

===> genrandom(size) : (int) -> bytes
입력된 사이즈 만큼의 암호학적으로 안전한 난수 바이트를 생성.

===> mkkey(salt, pw, kf) : (bytes, bytes, bytes) -> bytes 48B
salt는 그대로, pw는 kf + pw + pw + kf + pw으로 scrypt 해싱함. 라운드 16384, 워드 8, 스레드 1. 48바이트 크기의 해시 내보냄.

===> svkey(salt, pw, kf) : (bytes, bytes, bytes) -> bytes 256B
salt는 그대로, pw는 pw + pw + kf + kf + pw으로 scrypt 해싱함. 라운드 524288, 워드 8, 스레드 1. 256바이트 크기의 해시 내보냄.

===> view(data) : (bytes/str) -> bytes
bytes 입력 경우 이 데이터를 KAES4 암호화 파일로 보고 힌트를 추출해 반환, str 입력 경우 이 경로의 파일을 KAES4 암호화 파일로 보고 힌트를 추출해 반환. 잘못된 형식이거나 헤더 모드가 FUNC인 경우는 notvalidKAES4 오류 발생.

===> enwhole(pw, kf, hint, data) : (bytes, bytes, bytes, bytes/str) -> bytes/str
pw, kf, hint를 받아 WHOLE 모드로 암호화. data가 bytes면 암호화 결과물은 bytes로 반환. data가 str이면 이 경로의 파일을 암호화하여 같은 폴더상에 생성, 이름은 랜덤. 이름을 str로 반환. data가 지원하지 않는 형식이라면 wrongdata 오류 발생.

===> dewhole(pw, kf, data) : (bytes, bytes, bytes/str) -> bytes/str
pw, kf를 받아 WHOLE 모드로 복호화. data가 bytes면 암호화 결과물은 bytes로 반환. data가 str이면 이 경로의 파일을 복호화하여 같은 폴더상에 생성. 복호화된 이름을 str로 반환. data가 지원하지 않는 형식이라면 wrongdata 오류 발생. PWH와 입력값이 불일치하는 경우 invalidKEY 오류 발생. 암호화 파일의 모드가 WHOLE이 아니거나 잘못된 파일의 경우 notvalidKAES4 오류 발생.

===> enfunc(mkey, data, tgt='temp466') : (bytes 48B, bytes/str, str) -> bytes/str
master key를 받아 FUNC 모드로 암호화. data가 bytes면 암호화 결과물은 bytes로 반환. data가 str이면 이 경로의 파일을 암호화하여 tgt 경로에 생성. tgt의 기본값은 temp466. data가 지원하지 않는 형식이라면 wrongdata 오류 발생.

===> defunc(mkey, data, tgt='temp466') : (bytes 48B, bytes/str, str) -> bytes/str
master key를 받아 FUNC 모드로 복호화. data가 bytes면 복호화 결과물은 bytes로 반환. data가 str이면 이 경로의 파일을 복호화하여 tgt 경로에 생성. tgt의 기본값은 temp466. data가 지원하지 않는 형식이라면 wrongdata 오류 발생. PWH와 입력값이 불일치하는 경우 invalidKEY 오류 발생. 암호화 파일의 모드가 WHOLE이 아니거나 잘못된 파일의 경우 notvalidKAES4 오류 발생.

3 네이티브 가속

순수 python으로 작성된 버전과 go를 이용한 가속 버전이 있음.
가속 버전은 python에서 go dll로 전송하는 데이터량에 비례해 메모리 누수가 발생.
가속 버전은 스크립트와 같은 폴더의 kaes4na.dll을 확인하고 유효성을 검증한 후 실행됨.
가속 대상 함수 : enshort, deshort, mkkey, svkey, enwhole, dewhole, enfunc, defunc

간단 벤치마크 (i5-13600kf + 32GB, 1GB 0파일/3.66GB 동영상 파일), B : 바이트, F : 파일

BBFBBF	enshort	enwhole	enfunc	enshort/	enwhole/	enfunc/
시간 (초)	87.2	14.3	45.4	48.2	49.8	2.60
속도 (MB/s)	11.7	71.6	82.6	21.2	20.6	1441
상대속도	1.00	6.12	7.06	1.80	1.76	123.2

4 상세 벤치마크 / 자원 사용량

파이썬 버전 (bytes : 10MB 0파일, file : 100MB 0파일)

BBBFF	enshort	enwhole	dewhole	enfunc	defunc
속도 (MB/s)	11.3	5.53	5.42	34.6	34.8
상대속도	1.00	0.489	0.480	3.06	3.08
메모리 (MB)	46.3	231.5	545.6	527.3	564.0
상대 메모리	x4.63	x23.2	x54.6	고정	고정
누수량 (MB)	-	-	-	-	-
상대 누수량	-	-	-	-	-

파이썬 버전 (bytes : 1GB 0파일, file : 3.66GB 동영상 파일)

BBBFF	enshort	enwhole	dewhole	enfunc	defunc
속도 (MB/s)	11.2	71.4	71.7	82.6	82.6
상대속도	1.00	6.38	6.40	7.38	7.38
메모리 (MB)	9268	2607	3636	551.2	555.0
상대 메모리	x9.05	x2.55	x3.55	고정	고정
누수량 (MB)	-	-	-	-	-
상대 누수량	-	-	-	-	-

가속 버전 (bytes : 10MB 0파일, file : 100MB 0파일)

BBBFF	enshort/	enwhole/	dewhole/	enfunc/	defunc/
속도 (MB/s)	21.2	7.83	7.69	47.8	48.9
상대속도	1.00	0.369	0.363	2.25	2.31
메모리 (MB)	86.1	767.7	532.1	89.3	92.1
상대 메모리	x8.61	x76.8	x53.2	고정	고정
누수량 (MB)	81.2	670.3	532.1	86.3	90.0
상대 누수량	x8.12	x67.0	x53.2	고정	고정

가속 버전 (bytes : 1GB 0파일(1023MB), file : 3.66GB 동영상 파일)

BBBFF	enshort/	enwhole/	dewhole/	enfunc/	defunc/
속도 (MB/s)	21.3	20.9	21.4	1587	1818
상대속도	1.00	0.981	1.00	74.5	85.4
메모리 (MB)	14456	16030	15582	77.6	90.1
상대 메모리	x14.1	x15.7	x15.2	고정	고정
누수량 (MB)	4226	5798	4648	77.6	85.3
상대 누수량	x4.13	x5.67	x4.54	고정	고정

경고 : bytes 가속 암호화 모드는 메모리 누수가 심하게 발생하고, 1GB 이상의 데이터를 처리할 수 없습니다. 따라서 파일 암호화의 경우 가속 버전을 사용하고, 바이트 암호화의 경우 파이썬 버전을 사용하는 것을 추천합니다.	경고 : 파이썬 버전의 경우 메인 스크립트에 다음 문구 추가
	`import multiprocessing as mp`
	`if __name__ == '__main__':`
	`  mp.freeze_support()`

경고 : 이 벤치마크는 1회당 기준값 추정치이며, 사용자 환경에 따라 달라질 수 있음.