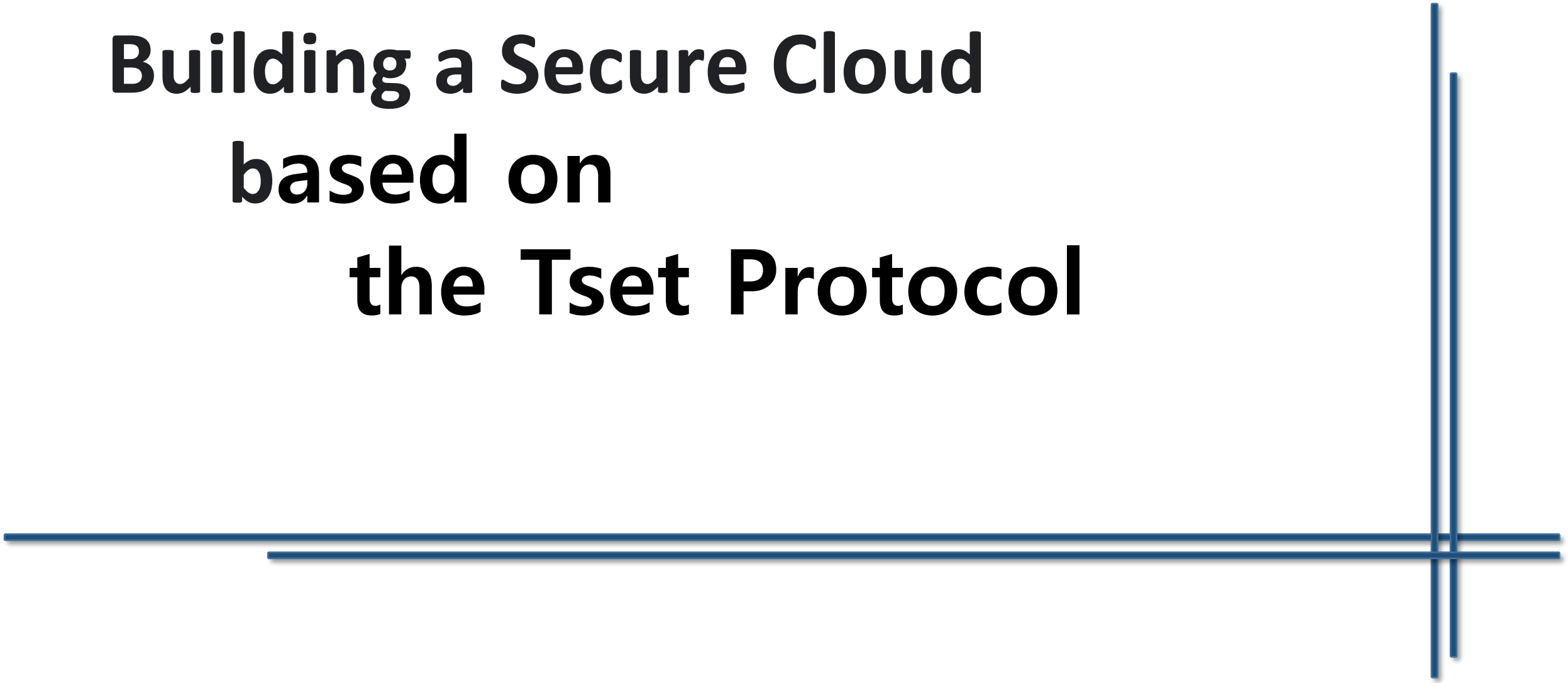
Building a Secure Cloud based on the Tset Protocol



Introduction

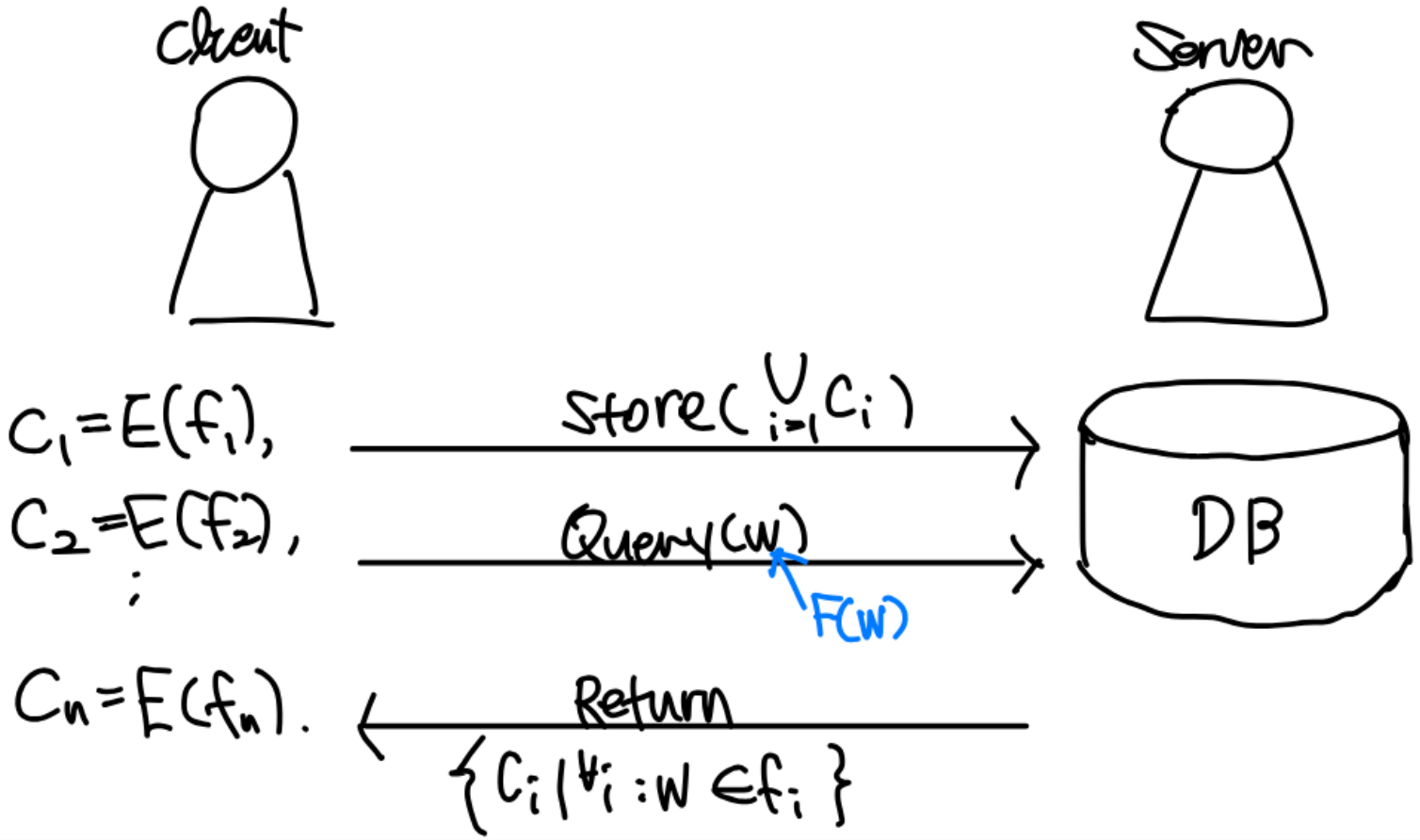
- **Highly-scalable searchable symmetric encryption with support for Boolean queries 논문의 SSE 시스템 구현 (Tset protocol)**
  - **OPENSSL(AES128, SHA256), BOOST/ASIO 라이브러리 사용**
  - **Socket Programming: 클라이언트, 서버-데이터베이스 구현**
  - **MariaDB, 패킷 설계, Serialize**

Why Tset ?

- 사람은 원하는 정보 검색할 때 키워드 위주 서칭을 한다 .
- 클라우드에 암호화 된 파일을 검색 시, 안전한 검색을 지원하는 방법
- Keyword w 제공->w 포함하는 모든 파일의 id찾음
- Ex) 경찰청 클라우드에서 사건 문서를 검색 - 익사, 노란색코트, 인상착의 등의 키워드로 검색 가능
- 이러한 키워드들은 파일 내에 직접적으로 존재할 수도 있지만 파일 내의 단어가 아닌, 주요 키워드 추출 알고리즘을 통해 얻은 독립적인 또 다른 데이터들이다.
- 중복되어 나타나는 키워드의 존재와 분포는 기밀하게 지켜야 할 대상이 된다.

파일 암호화

- Tset 구축 전, 파일 자체의 암호화 시행 후 전송



<Store>

$X_i \leftarrow AES.E(K_0, W_i)$ and $X_i = L_i \| R_i$

$S_i \leftarrow G(K_1, i)$ and $kappa_i \leftarrow G(K_2, L_i)$

$U_i \leftarrow F(kappa_i, S_i)$ and set $T_i \leftarrow S_i \| U_i$

$C_i \leftarrow X_i \oplus T_i$

# of words 만큼 반복, $C = (C_1 \| C_2 \| \cdots \| C_n)$

Send C to Server

<Restore>

$C_i^* = A_i \| B_i$

$S_i \leftarrow G(K_1, i)$ and $L_i \leftarrow S_i \oplus A_i$

$kappa_i \leftarrow G(K_2, L_i)$ and $U_i \leftarrow F(kappa_i, S_i)$

$T_i \leftarrow S_i \| U_i$

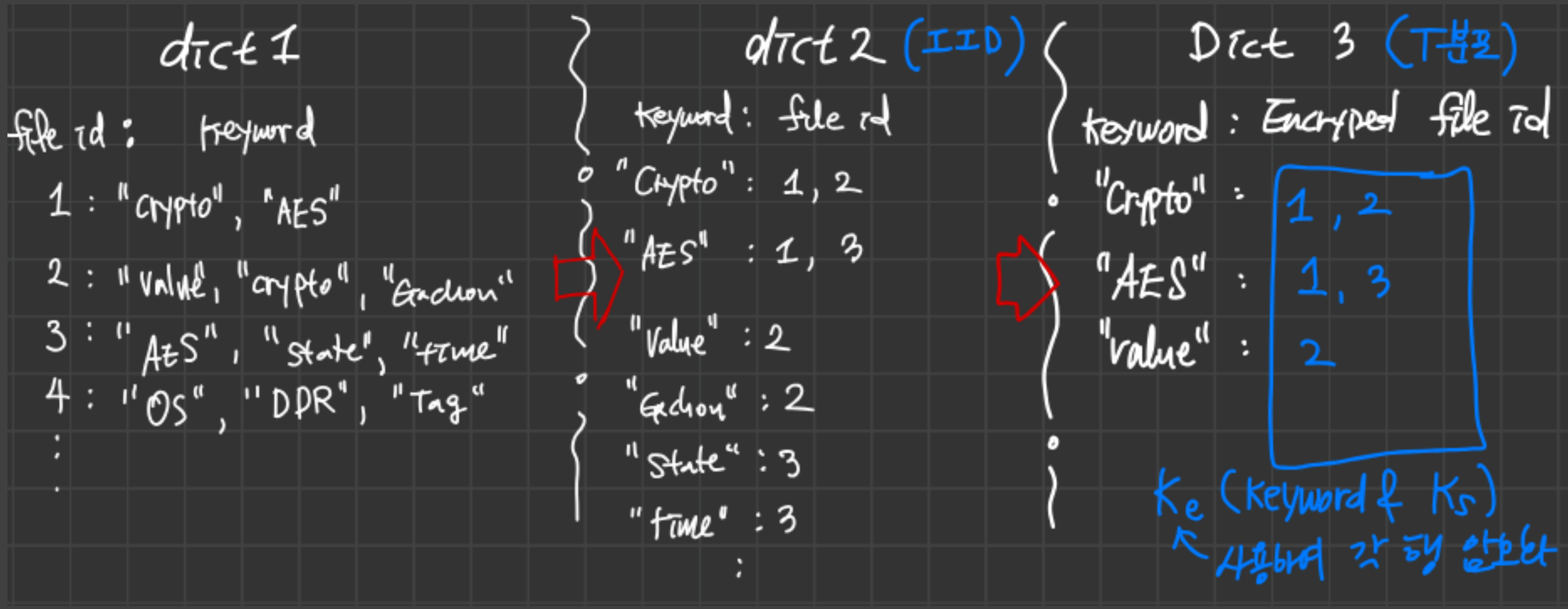
$X_i \leftarrow C_i^* \oplus T_i$ and $W_i \leftarrow AES.D(K_0, X_i)$

# of words 만큼 반복, $D = (W_1 \| W_2 \| \cdots \| W_n)$

Store D as a file.

Why Tset ?

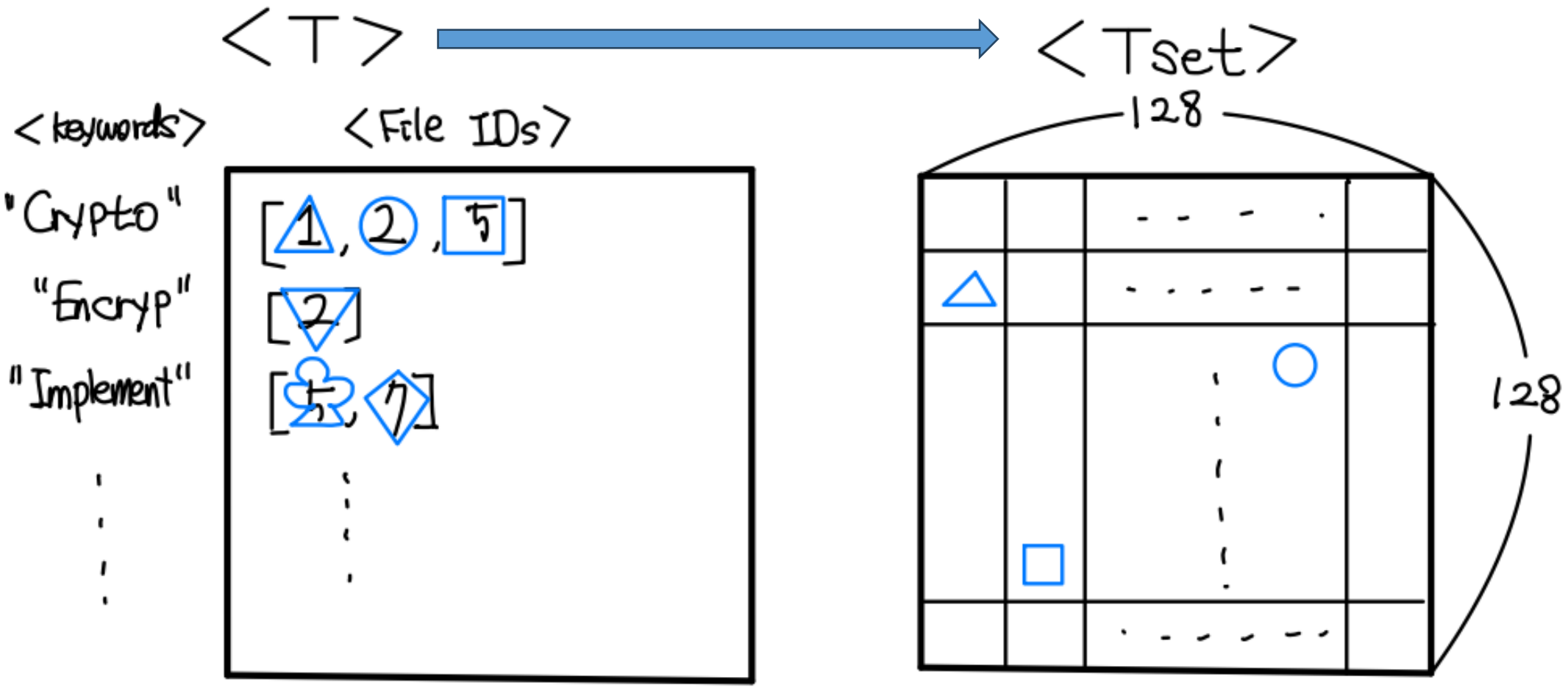
- 클라우드는 외부 서버이기 때문에, 모든 파일이 암호화 되어있는 것은 당연하고
- 1) 어떤 키를 받기 전까지는 Search 자체가 불가
- 2) Client가 키워드를 전송했을 때 Server는 아무것도 모르는 상태에서 키 만으로 검색을 수행해야 함



T 만으로는 키워드 & 파일 모두 암호화되어 있더라도 어떤 키워드가 갖는 파일 개수와 분포를 공격자가 쉽게 알아챌 수 있다.

⇒ TSet protocal 제안

A short explanation of a Tset



• Client 단에서 할일: 파일 당 키워드 추출해둔 , T 표로 만들기 → 이를 TSetup으로 만들어서 binary 형태로 서버로 전송.

(Documents는 이미 K_d로 암호화해야 전송)

필요한 key : K_d, K_S, $K_T \leftarrow rand(), \{0,1\}^{\lambda}$ ← 32

Document / keyword / stag

TSetSetup(T, W) → (TSet)

Type: pair(key, value) = α

Size: $S \times S$.

How to figure out S? 정수이고, 2^n

$\leftarrow \sqrt{\max(\#\text{ of keywords}) * \max(\#\text{ of Ids})}$

어떤 키워드가 가질 수 있는 파일 idx들의 최대값

/* $W = \bigcup_{i=1}^{n} w_i$

for each $w_i \in W$:

$stag \leftarrow AES.E(K_T, w_i)$

for $i = 1$ to t_w : // t_w: # of ids which contain w

for each $w_i \in W$:

어떤 키워드가 가질 수 있는 파일 더들의 최대값

$Stag \leftarrow AES.E(K_T, w_i)$

for $i = 1$ to t_w : // t_w: # of ids which contain w

$(b \| L \| k) \xleftarrow{256} SHA(AES.E(Stag, i))$ 7, 80, 129

If $Free[b] = \emptyset$, then abort ∵ $S=128$ $p=\frac{1}{2^{80}}$ $128+1$

$j \leftarrow Free[b] = \{1, 2, \cdots, S\}$ // pick a random column ⇒ $TSet[b, j]$

If $i < t_w$:

$\beta \leftarrow 1$ // 해당하는 파일 더가 더 있다는 Indicator

else

$\beta \leftarrow 0$

$\alpha.key \leftarrow L$

$\alpha.value \leftarrow (\beta \| e_i) \oplus k$

$TSet[b, j] \leftarrow \alpha$

b: row, L: Stag 만들때 서버에서 retrieve 시 L == TSet(i,j).first

↑ 서버에서 AES → parse ↑ 서버 TSet

β: 1 ← 더 찾아야 할 때, 0 ← 그만 찾아도 괜찮을 때

e_i: 암호화된 파일 데이터

- Server 에서 할일 : 클라이언트로부터 Stag(비밀키) 받음

  i) 미리 전송된 해당 클라이언트의 TSet 내에서 $\{e_1, e_2 \cdots e_w\}$ 찾아서 전송

  (client: $\{e_1, e_2, \cdots, e_w\}$ 복호화 후 $\{id_1, id_2, \cdots id_w\}$ 전송)

  ii) $\{id_1, id_2, \cdots id_w\}$를 받은 서버는 DB에서 해당 파일 id에 해당하는 Documents 모두 전송

  (client: Document 받아서 복호화 후, 파일로 저장)

TSetRetrieve(TSet, stag) $\longrightarrow$ ($\mathbb{V} = (\tilde{id}_1, \tilde{id}_2, \cdots, \tilde{id}_t)$)

$\mathbb{V} \leftarrow \emptyset$, $\beta \leftarrow 1$, $i \leftarrow 1$

while $\beta = 1$ do

  $(b \| L \| K) \leftarrow SHA(AES.E(stag, i))$

  $T \leftarrow TSet[b]$

  for $1 \le j \le S$ do

    if $T[b,j].label = L$ then // 해당하는 파일 아이디 찾음

      $v \leftarrow T[b,j].value \oplus K$ // b행에 1개 이상 있을 수 있음.

      $\beta \leftarrow v[0]$

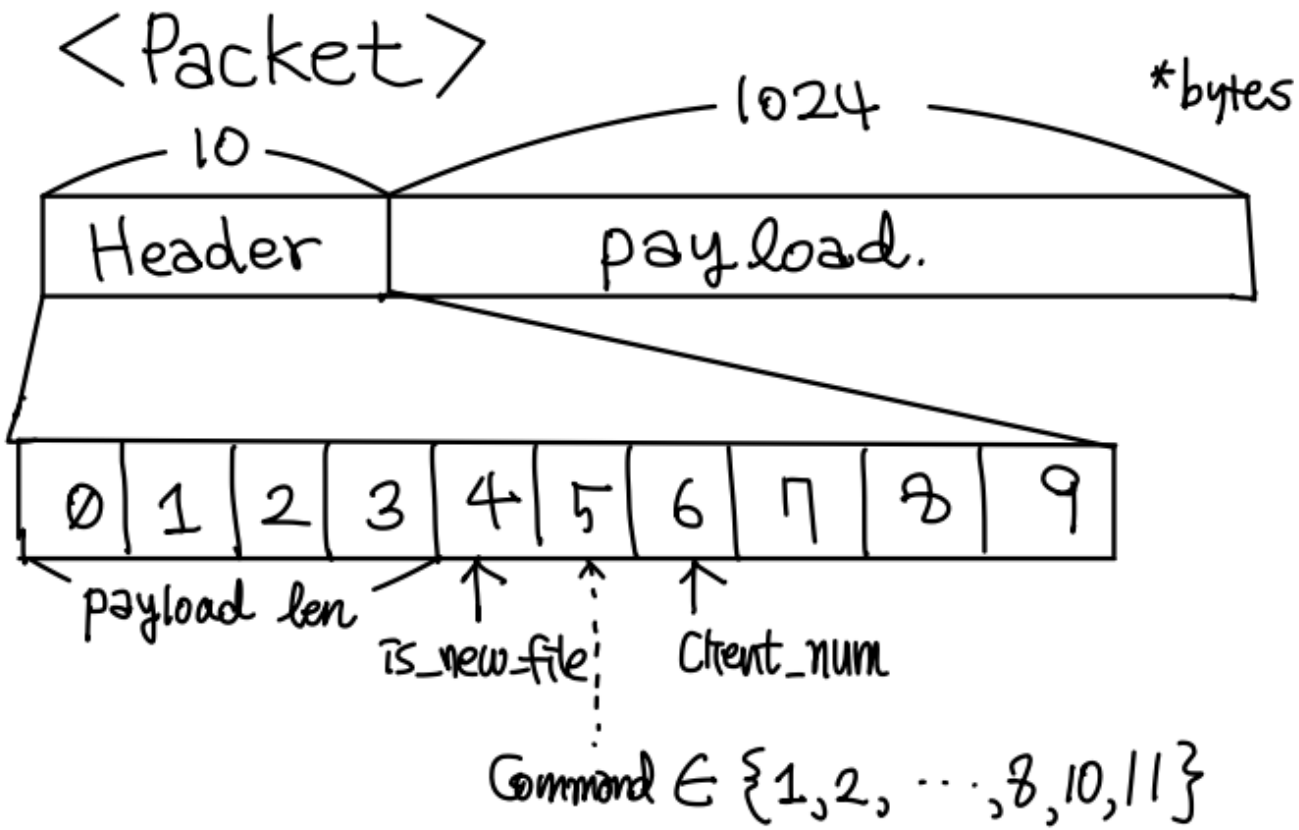
      $id \leftarrow v[1:129]$ // v의 0부터 128bit

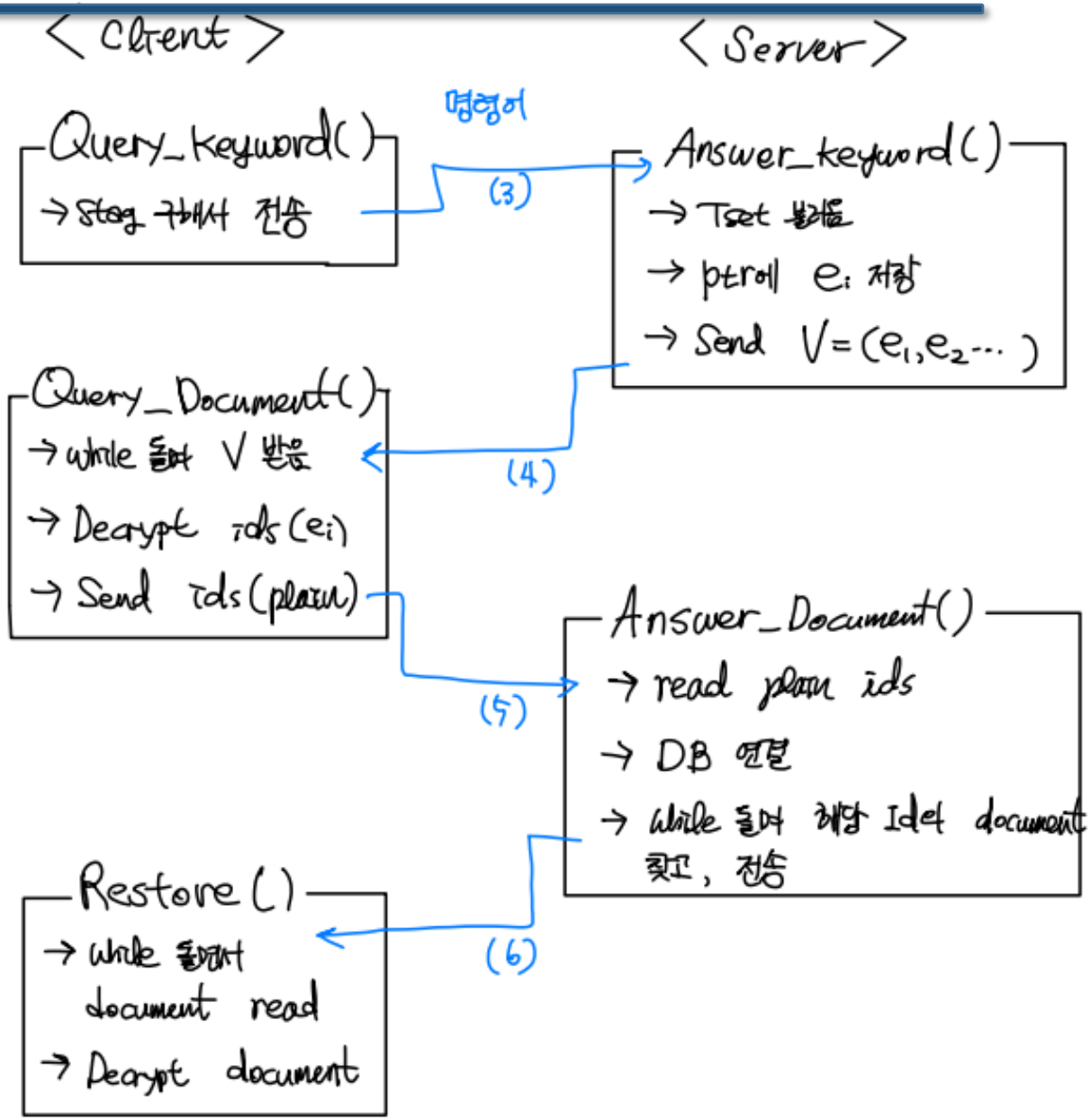
      $\mathbb{V} \leftarrow \mathbb{V} \cup \{id\}$

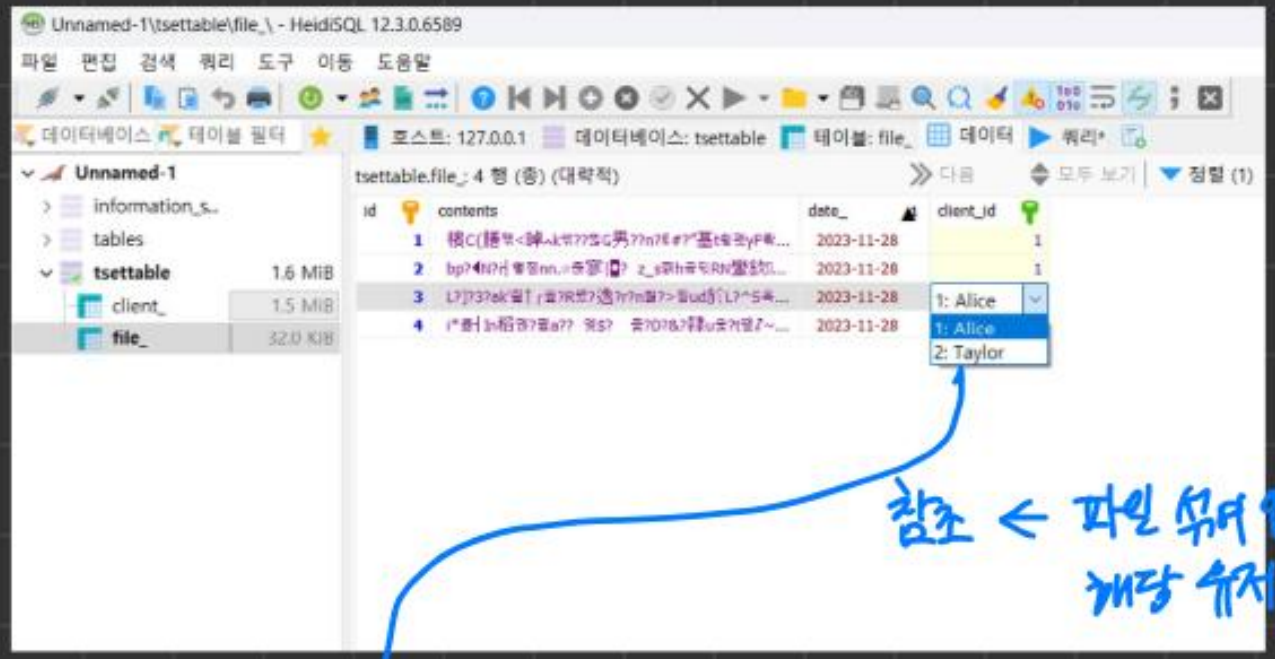
  $i \leftarrow i+1$

return $\mathbb{V}$

Network Communication



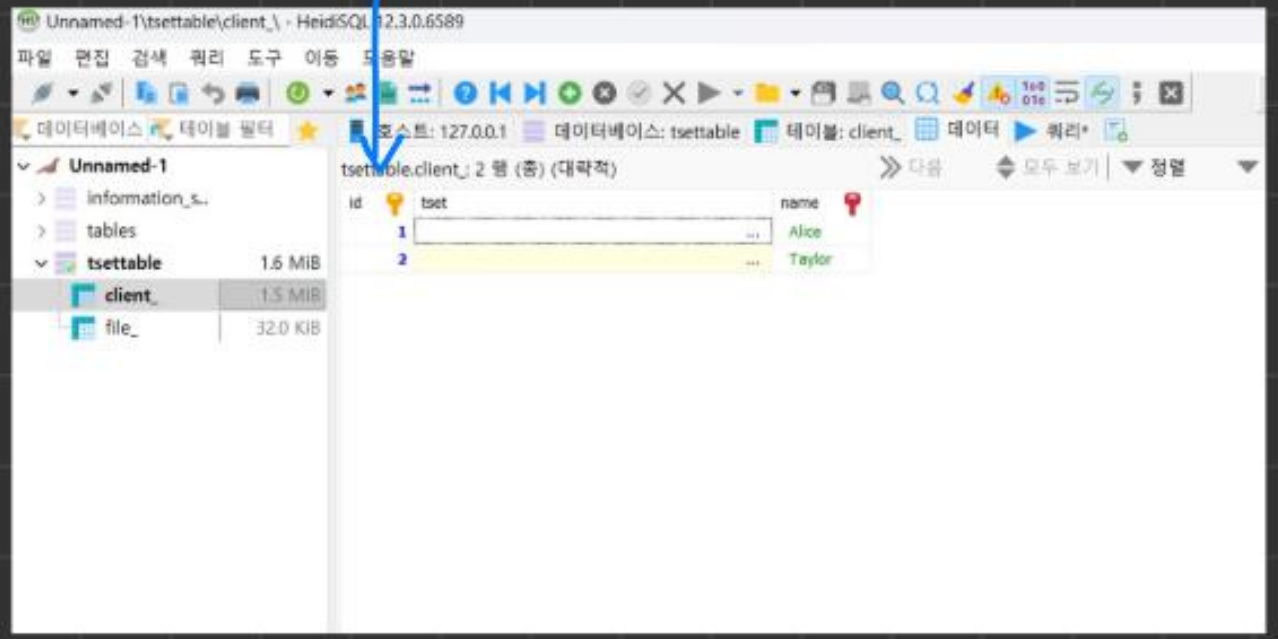




Document table

→ client_id : foreign key

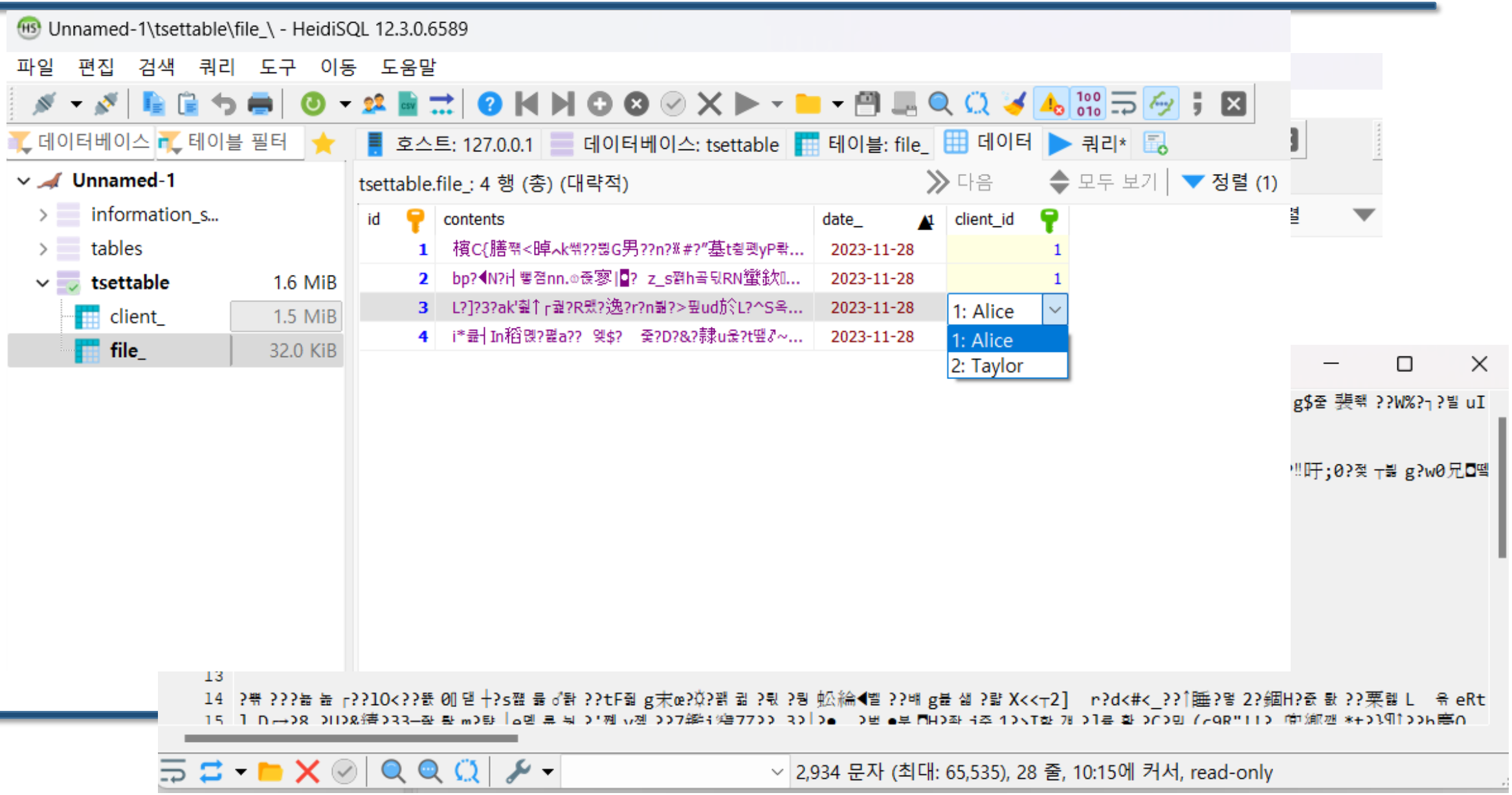
참조 ← 파일 업로드 시 업로드 되어진 해당 유저 파일만 접근해서 다운가능.



Client info table

→ tset : blob data

DataBase


Unnamed-1\tsettable\file_\ - HeidiSQL 12.3.0.6589
파일 편집 검색 쿼리 도구 이동 도움말
데이터베이스 테이블 필터
호스트: 127.0.0.1 데이터베이스: tsettable 테이블: file_ 데이터 쿼리*
Unnamed-1
information_s...
tables
tsettable 1.6 MiB
client_ 1.5 MiB
file_ 32.0 KiB
tsettable.file_: 4 행 (총) (대략적)
다음 모두 보기 정렬 (1)
id contents date_ client_id
1 2023-11-28 1
2 2023-11-28 1
3 2023-11-28 1: Alice
4 2023-11-28
1: Alice
2: Taylor
2,934 문자 (최대: 65,535), 28 줄, 10:15에 커서, read-only

The Limitations..

- **Improvements to the code itself**
  - Connecting with many clients
  - Packet processing structure

- **The Tset**
  - The server eventually knows the number of file IDs when you query.
  - Actual IDs exposed when you request documents.
  - As the queries are repeated, security continues to decrease.
  - Only Single keyword search provided

The ways to improve the system.

- **Improvements to the code itself**
  - Profiling needed
- **The Tset**
  - Search two different keywords at Once
  - Reset the Tset every time you query a keyword
  - Compare the cost of reduced security searching keywords & the cost of resetting the Tset

Thank you