(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2024-0145654
(43) 공개일자 2024년10월07일

(51) 국제특허분류(Int. Cl.)
G06F 18/2413 (2023.01) *G06F 16/9032* (2019.01)
G06F 16/906 (2019.01)
(52) CPC특허분류
G06F 18/24147 (2023.01)
G06F 16/9032 (2019.01)
(21) 출원번호 10-2023-0040377
(22) 출원일자 2023년03월28일
심사청구일자 2023년03월28일

(71) 출원인
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)
(72) 발명자
박정수
서울특별시 동대문구 제기로4길 46
이동훈
서울특별시 종로구 사직로8길 34, 1404호(내수동, 경희궁의아침3단지아파트)
(74) 대리인
김등용

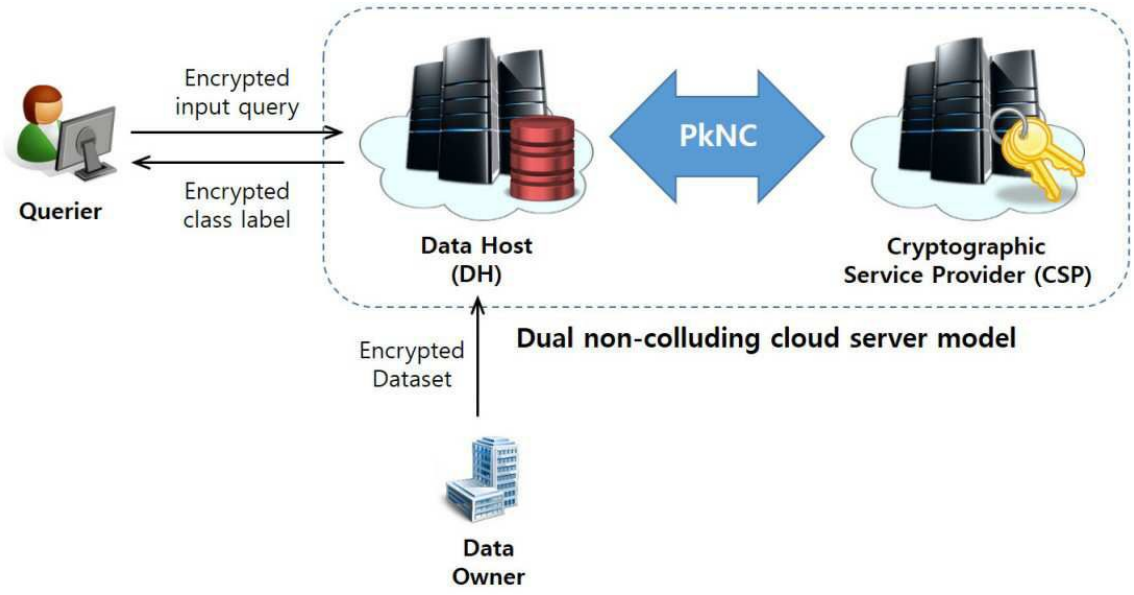
전체 청구항 수 : 총 6 항

(54) 발명의 명칭 **외부 클라우드 컴퓨팅 환경에서 병렬로 실행되고 프라이버시를 보호하는 kNN 분류 장치 및 방법**

(57) 요 약

외부 클라우드 컴퓨팅 환경에서 병렬로 실행되고 프라이버시를 보호하는 kNN(k-Nearest Neighbor) 분류 시스템 및 방법이 개시된다. 상기 kNN 분류 방법은 kNN 분류 시스템 상에서 수행되고, 질의자의 단말로부터 쿼리 데이터를 포함하는 입력 쿼리를 수신하는 단계, 데이터셋에 포함된 복수의 데이터들 각각과 상기 쿼리 데이터의 거리를 계산하는 단계, 상기 복수의 데이터들 중에서 가장 작은 거리를 갖는 k 개의 데이터들을 계산하는 단계, 및 상기 k 개의 데이터들 중에서 가장 많은 데이터의 클래스를 상기 쿼리 데이터의 클래스로 결정하는 단계를 포함한다.

대 표 도 - 도1



(52) CPC특허분류
G06F 16/906 (2019.01)

이 발명을 지원한 국가연구개발사업
과제고유번호 1345358724
과제번호 2022R1A6A3A01087466
부처명 교육부
과제관리(전문)기관명 한국연구재단
연구사업명 이공학학술연구기반구축
연구과제명 외부 클라우드 컴퓨팅 환경에서 병렬로 실행되고 증명 가능한 안전성을 제공하는 계층적 군집기법 개발
기 여 율 0.7/1
과제수행기관명 고려대학교 산학협력단
연구기간 2022.09.01 ~ 2024.08.31
이 발명을 지원한 국가연구개발사업
과제고유번호 1711152985
과제번호 2021-0-00518-002
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 데이터경제를위한블록체인기술개발
연구과제명 블록체인 데이터 암호화 기반의 프라이버시 보호 기술개발
기 여 율 0.3/1
과제수행기관명 한양대학교산학협력단
연구기간 2021.04.01 ~ 2025.12.31

명 세 서

청구범위

청구항 1

kNN(k-Nearest Neighbor) 분류 시스템 상에서 수행되는 kNN 분류 방법에 있어서,

질의자의 단말로부터 쿼리 데이터를 포함하는 입력 쿼리를 수신하는 단계;

데이터셋에 포함된 복수의 데이터들 각각과 상기 쿼리 데이터의 거리를 계산하는 단계;

상기 복수의 데이터들 중에서 가장 작은 거리를 갖는 k 개의 데이터들을 계산하는 단계; 및

상기 k 개의 데이터들 중에서 가장 많은 데이터의 클래스를 상기 쿼리 데이터의 클래스로 결정하는 단계를 포함하는 kNN 분류 방법.

청구항 2

제1항에 있어서,

상기 쿼리 데이터는 암호화된 상태로 수신되고,

상기 거리를 계산하는 단계는, 암호화된 상기 쿼리 데이터와 암호화된 상기 복수의 데이터들 각각 사이의 거리를 계산하는,

kNN 분류 방법.

청구항 3

제1항에 있어서,

상기 k 개의 데이터들을 계산하는 단계는, 알고리즘 2를 이용하여 수행되고, 상기 알고리즘 2는 계산된 거리의 1의 보수(1′ s complement)를 입력으로 수신하는,

kNN 분류 방법.

[알고리즘 2]

Algorithm 2: Secure version of SkLE ($SkLE_S$)

DH input : set of encrypted elements $\{\langle E(e_i)\rangle_B\}_{i\in[n]}$
CSP input : secret key SK
DH output: $\{E(K_i)\}_{i\in[n]}$ where $K_i = 1$ if an element e_i is included in the set of k largest elements; otherwise, $K_i = 0$.

DH locally performs all computations besides the interactive protocols for functionalities F_{SM}, F_{SBD}, and F_{SCI}.
for $i \leftarrow 1, \ldots, n$ **do**
  $E(K_i) \leftarrow E(0)$
  $E(C_i) \leftarrow E(1)$
end
for $j \leftarrow l-1, \ldots, 0$ **do**
  for $i \leftarrow 1, \ldots, n$ **do**
    $(E(u_i), \perp) \leftarrow F_{SM}(\{E(e_{i,j}), E(C_i)\}, SK)$
    $E(P_i) \leftarrow E(K_i) * E(u_i)$
  end
  $E(s) \leftarrow \prod_{i=1}^{n} E(P_i)$
  $(S', \perp) \leftarrow F_{SBD}(E(s), SK)$
  $(\{E(M), E(D)\}, \perp) \leftarrow F_{SCI}(\{S', K'\}, SK)$
  $(E(\alpha), \perp) \leftarrow F_{SM}(\{E(D), E(M)\}, SK)$
  $E(\beta) \leftarrow E(1) * E(D)^{N-1} * E(\alpha)$
  $E(\gamma) \leftarrow E(D) * E(\alpha)^{N-2}$
  for $i \leftarrow 1, \ldots, n$ **do**
    $(E(v_i), \perp) \leftarrow F_{SM}(\{E(u_i), E(\beta)\}, SK)$
    $E(K_i) \leftarrow E(K_i) * E(v_i)$
    $(E(w_i), \perp) \leftarrow F_{SM}(\{E(e_{i,j}, E(\gamma)\}, SK)$
    $E(x_i) \leftarrow E(M) * E(w_i)$
    $(E(C_i), \perp) \leftarrow F_{SM}(\{E(C_i), E(x_i)\}, SK)$
  end
end
for $i \leftarrow 1, \ldots, n$ **do**
  $E(K_i) \leftarrow E(K_i) * E(C_i)$
end
return $\{E(K_i)\}_{i\in[n]}$

청구항 4

제3항에 있어서,

상기 $F_{SCI}()$는 알고리즘 1을 이용하여 계산되는,

kNN 분류 방법.

[알고리즘 1]

Algorithm 1: Secure Comparison and Inequality (SCI)

DH input : $\{S', k'\}$ where $S' = \{\langle E(s)\rangle_B, \langle E(\overline{s})\rangle_B\}$ and $k' = \{\langle k\rangle_B, \langle \overline{k}\rangle_B\}$

CSP input : secret key SK

DH output: $\{E(M), E(D)\}$ where $E(M) = \begin{cases} E(1), \text{ if } S < k \\ E(0), \text{ otherwise} \end{cases}$ and

$$E(D) = \begin{cases} E(1), \text{ if } S \neq k \\ E(0), \text{ otherwise} \end{cases}$$

DH:
  toss a random coin $\alpha \in_R \{0, 1\}$
  for $j \leftarrow l-1, \ldots, 0$ **do**
    if $\alpha = 0$ **then**
      $E(w_j) \leftarrow E(\overline{s_j})^{k_j}$
    else
      $E(w_j) \leftarrow E(s_j)^{\overline{k_j}}$
    end
    $E(w_j) \leftarrow E(w_j)^{r_j}$ where $r_j \neq 0 \in_R \mathbb{Z}_N$
    $E(x_j) \leftarrow E(s_j \oplus k_j)$
    $E(y_j) \leftarrow E(y_{j+1})^{r'_j} * E(x_j)$ where $r'_j \neq 0 \in_R \mathbb{Z}_N$ and $E(y_l) = E(0)$
  end
DH, CSP:
  $(E(\gamma), \perp) \leftarrow F_{SZP}(\{E(x_j)\}_{j\in\{0,\ldots,l-1\}}, SK)$
DH:
  if $\alpha = 0$ **then**
    $E(y_0) \leftarrow E(y_0) * E(\gamma)$
  end
  for $j \leftarrow l-1, \ldots, 0$ **do**
    $E(z_j) \leftarrow E(y_j) * E(N-1)$
    $E(u_j) \leftarrow E(z_j)^{r''_j} * E(w_j)$ where $r''_j \in_R \mathbb{Z}_N$
  end
  $\{v'_j\}_{j\in\{0,\ldots,l-1\}} \leftarrow \sigma(\{E(u_j)\}_{j\in\{0,\ldots,l-1\}})$
  $DH \rightarrow CSP : \{v'_j\}_{j\in\{0,\ldots,l-1\}}$
CSP:
  $\{v_j\}_{j\in\{0,\ldots,l-1\}} \leftarrow \{D(v'_j)\}_{j\in\{0,\ldots,l-1\}}$
  if $\exists v_j = 0$ *in* $\{v_j\}_{j\in\{0,\ldots,l-1\}}$ **then**
    $E(\beta) \leftarrow E(0)$
  else
    $E(\beta) \leftarrow E(1)$
  end
  $CSP \rightarrow DH : E(\beta)$
DH:
  if $\alpha = 0$ **then**
    $E(M) \leftarrow E(\beta)$
  else
    $E(M) \leftarrow E(1) * E(\beta)^{N-1}$
  end
  $E(D) \leftarrow E(1) * E(\gamma)^{N-1}$
return $\{E(M), E(D)\}$

청구항 5

제4항에 있어서,

상기 $F_{SZP}()$는 알고리즘 A1을 이용하여 계산되는,

kNN 분류 방법.

[알고리즘 A1]

Algorithm A1: Secure Zero Protocol (SZP)

DH input : $\{E(x_i)\}_{i\in[l]}$
CSP input : secret key SK
DH output: $E(\gamma) = \begin{cases} E(1), \text{ if } x_i = 0 \text{ for all } i \in [l] \\ E(0), \text{ otherwise} \end{cases}$

DH:
  toss a random coin $\alpha \in_R \{0,1\}$
  if $\alpha = 0$ **then**
    for $i \leftarrow 1,\ldots,l$ **do**
      $E(s_i) \leftarrow E(-1) * E(x_i) * (\prod_{j=i+1}^{l} E(x_j))^2$
      $E(c_i) \leftarrow E(s_i)^{r_i}$ where $r_i \neq 0 \in_R \mathbb{Z}_N$
    end
  else
    $E(s_1) \leftarrow \prod_{i=1}^{l} E(x_i)$
    $E(c_1) \leftarrow E(s_1)^{r_1}$ where $r_1 \neq 0 \in_R \mathbb{Z}_N$
    for $i \leftarrow 2,\ldots,l$ **do**
      $E(c_i) \leftarrow E(r_i)$ where $r_i \neq 0 \in_R \mathbb{Z}_N$
    end
  end
  $\{d'_i\}_{i\in[l]} \leftarrow \sigma(\{E(c_i)\}_{i\in[l]})$
  $DH \rightarrow CSP : \{d'_i\}_{i\in[l]}$
CSP:
  $\{d_i\}_{i\in[l]} \leftarrow \{D(d'_i)\}_{i\in[l]}$
  if $\exists d_i = 0$ *in* $\{d_i\}_{i\in[l]}$ **then**
    $E(\beta) \leftarrow E(0)$
  else
    $E(\beta) \leftarrow E(1)$
  end
  $CSP \rightarrow DH : E(\beta)$
DH:
  if $\alpha = 0$ **then**
    $E(\gamma) \leftarrow E(\beta)$
  else
    $E(\gamma) \leftarrow E(1) * E(\beta)^{N-1}$
  end
return $E(\gamma)$

청구항 6

제5항에 있어서,

상기 질의자의 단말로 결정된 클래스를 송신하는 단계를 더 포함하는,

kNN 분류 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 입력 데이터의 정보를 보호하면서 효율적으로 분류하는 kNN(k-nearest neighbor) 분류 기법으로서, 보다 상세하게는 수많은 데이터를 기반으로 주어진 입력 쿼리 데이터를 효율적이고 안전하게 분류하는 기법에 관한 것이며, 의료 진단, 스팸 메일 탐지, 안면 인식, 금융 분야 등등 다양한 분야에서 사용될 수 있다.

배 경 기 술

[0002] 기하급수적으로 증가하는 데이터를 효율적으로 관리하기 위해서 대규모의 자원을 갖는 클라우드 서비스가 자주 사용된다. 데이터 소유자는 자신의 데이터를 아마존이나 구글 같은 대형 IT 회사의 클라우드 서비스 회사에 위탁함으로써 데이터 관리 비용을 줄일 수 있다. 또한, 클라우드 컴퓨팅 서비스는 사용자의 데이터를 데이터 마이닝(data mining) 기법을 사용하여 효율적으로 분석함으로써 의미 있는 정보를 제공할 수 있다.

[0003] 그러나, 외부 클라우드 컴퓨팅 서비스에서는 데이터와 쿼리의 심각한 프라이버시 문제가 발생될 수 있다. 데이

터 소유자가 데이터를 클라우드에 위탁하고 사용자가 쿼리를 전송하면 클라우드 서비스 제공자는 위탁된 데이터, 쿼리, 쿼리 결과를 노출함으로써 데이터 소유자와 사용자의 프라이버시를 침해할 수 있다. 이러한 데이터와 쿼리의 프라이버시를 보호하기 위해서 데이터 소유자와 사용자가 메시지를 클라우드에 전송하기 전에 암호화하더라도, 클라우드 서비스 제공자는 암호화된 데이터의 접근 패턴을 분석함으로써 추가적인 정보를 얻을 수 있다.

[0004] 외부 클라우드 컴퓨팅 환경에서 실행되는 데이터 마이닝 기법은 데이터, 쿼리, 쿼리 결과 및 데이터 접근 패턴의 프라이버시를 보장해야 한다. 본 발명에서는 프라이버시 보호 기술로서, 부분 동형 암호 기법(Partially Homomorphic Encryption, PHE)을 사용한다. 동형 암호 기법은 암호문 상태에서 평문 데이터의 연산이 가능한 암호 기법이다. 부분 동형 암호 기법은 연산 종류 중에서 한가지 연산(e.g. 덧셈 연산)을 무한한 횟수로 수행 가능한 암호 기법이다.

[0005] 데이터 마이닝 기법 중에서 분류(classification) 기법은 의료 진단, 스팸 메일 탐지, 안면 인식, 금융 분야 등등 다양한 분야에서 사용된다. 분류 작업을 위한 알고리즘 중에서 kNN(k-nearest neighbor) 분류 기법은 높은 성능과 효율적인 결과를 제공한다. kNN 분류 기법은 클래스가 분류된 데이터 집합에서 쿼리와 가장 유사한 k개의 데이터를 선택하고, 선택된 k개의 데이터 중에서 가장 많은 클래스를 쿼리의 클래스로 분류한다.

[0006] kNN 분류 기법에서 데이터의 개수가 많을수록 kNN 결과의 정확도는 향상되지만 연산량은 증가한다. 또한, 파라미터 k도 값이 높을수록 kNN 결과의 에러율은 낮아지지만 연산량은 증가한다. 그러므로, kNN 결과의 정확도와 연산의 효율성을 고려하여 k값은 적절하게 선택되어야 한다. 일반적으로, k값은 데이터 개수의 제곱근 값을 사용하지만, 빅데이터처럼 수많은 데이터를 사용할 경우에는 연산량이 폭발적으로 증가하는 문제점이 있다.

[0007] 본 발명의 시스템은 데이터 소유자, 사용자, kNN 연산을 수행하는 서로 공모하지 않는 두 클라우드 서버(DH와 CSP)로 구성된다. 데이터 소유자의 각 데이터는 여러 개의 원소값(attributes)과 클래스 레이블로 구성된다. 사용자의 쿼리는 여러 개의 원소값으로 구성되고, 사용자는 데이터 집합을 기반으로 쿼리가 어떤 클래스에 포함되는지 알기를 원한다.

[0008] 데이터 소유자는 데이터의 프라이버시를 보호하기 위해서 데이터를 암호화한 후에 DH에게 업로드한다. 사용자도 마찬가지로 쿼리의 프라이버시를 보호하기 위해서 쿼리를 암호화한 후에 DH에게 전송한다. DH는 CSP와 함께 kNN 분류 작업을 실행하고, kNN 분류 작업이 완료된 후에 DH는 쿼리의 분류 결과를 암호문 상태로 사용자에게 전송한다. 본 발명에 의할 경우, 데이터, 쿼리, 쿼리 결과 및 데이터 접근 패턴에 대한 어떠한 정보도 노출되지 않는다. 이렇게 프라이버시를 보호하는 kNN 분류 기법을 PkNC(Privacy Preserving k-Nearest Neighbor Classification)라고 한다.

[0009] PkNC 기법에 관한 기존 연구가 있지만, 기존 기법은 정보를 노출하거나 비효율적이었다. 기존 기법은 입력 데이터에 따라서 서브 프로토콜의 종료 시점이 달라지기 때문에 입력 데이터의 정보를 노출한다. 또한 기존 기법에서 사용된 비교 프로토콜은 비교하는 두 데이터가 같은지 아닌지에 관한 정보를 노출하는 문제점이 있다.

선행기술문헌

특허문헌

[0010] (특허문헌 0001) 대한민국 등록특허 제2010644호 (2019.08.14. 공고)

(특허문헌 0002) 대한민국 등록특허 제1953651호 (2019.03.04. 공고)

(특허문헌 0003) 대한민국 공개특허 제2010-0058697호 (2010.06.04. 공개)

발명의 내용

해결하려는 과제

[0011] 본 발명이 해결하고자 하는 첫 번째 과제는 kNN 분류 작업을 효율적으로 실행하면서 데이터의 프라이버시를 보호하는 기법(PkNC)을 제공하는 것이다. 본 발명은 각 데이터에 대해서 병렬로 실행하여 매우 효율적이면서도, 입력 데이터, 입력 쿼리, kNN 분류 결과 및 데이터 접근 패턴에 대한 프라이버시를 보호한다.

[0012] 본 발명이 해결하고자 하는 두 번째 과제는 데이터 집합에서 k개의 가장 큰/작은 데이터를 효율적으로 계산하면

서 데이터의 프라이버시를 보호하는 기법(SkLE/SkSE)을 제공하는 것이다.

[0013] 본 발명이 해결하고자 하는 세 번째 과제는 두 데이터를 비교하는 기존 기법의 보안 문제점을 해결한 기법(SCI)을 제공하는 것이다.

과제의 해결 수단

[0014] 본 발명의 일 실시예에 따른 kNN(k-Nearest Neighbor) 분류 방법은 kNN 분류 시스템 상에서 수행되고, 질의자의 단말로부터 쿼리 데이터를 포함하는 입력 쿼리를 수신하는 단계, 데이터셋에 포함된 복수의 데이터들 각각과 상기 쿼리 데이터의 거리를 계산하는 단계, 상기 복수의 데이터들 중에서 가장 작은 거리를 갖는 k 개의 데이터들을 계산하는 단계, 및 상기 k 개의 데이터들 중에서 가장 많은 데이터의 클래스를 상기 쿼리 데이터의 클래스로 결정하는 단계를 포함한다.

발명의 효과

[0015] 제안하는 PkNC의 실행시간은 기존 기법보다 효율적이다. 즉, 기존 연구의 실험에서 사용했던 데이터로 실험한 결과, 기존 기법은 12.02~55.5분 소요(k=5~25)되는 반면에, 본 발명은 4.38분 소요(k=1000)된다. 즉, 본 발명의 실행 시간은 기존 기법보다 약 1/3 이하로 감소된다.

[0016] PkNC의 연산량과 통신량의 복잡도(complexity)는 기존 기법보다 효율적이다. 기존 기법의 복잡도는 O(kln)이 되지만, 본 발명의 복잡도는 O(ln)으로서 더 효율적이다. 여기서 l은 데이터의 길이를 의미하고, n은 데이터의 개수를 의미한다.

[0017] 실행시간에 가장 큰 영향을 미치는 병렬로 전송되는 통신 횟수에 대해서 PkNC의 복잡도는 O(1)이다. 즉, PkNC의 통신 횟수는 이론적으로 데이터의 개수와 비례하지 않으므로, 수많은 데이터를 처리하는 빅데이터 분석에 적합하다.

[0018] PkNC는 각 데이터에 대해서 병렬로 실행될 수 있기 때문에, 수많은 병렬 연산이 가능한 환경(예를 들면, 클라우드 컴퓨팅 환경)에서 실행될 경우, 그 성능이 병렬 연산의 개수와 비례하여 향상된다.

[0019] PkNC는 연산을 수행하는 두 클라우드 서버에게 입력 데이터, 입력 쿼리, kNN 분류 결과 및 데이터 접근 패턴의 프라이버시를 보호하고, 본 발명의 안전성은 수학적으로 증명된다.

[0020] PkNC의 모든 연산은 두 클라우드 서버에게 모두 위탁된다. 즉, 데이터 소유자와 사용자는 각각 입력 데이터와 입력 쿼리를 두 클라우드 서버에게 전송한 후에 PkNC 연산에 개입되지 않으므로, 연산량이 부족한 사용자도 PkNC에 참여할 수 있다.

[0021] SkLE/SkSE는 각 데이터에 대해서 병렬로 실행되므로 매우 효율적이다. 기존 기법은 모든 데이터 및 k에 대해서 순차적(serially)으로 실행되므로, 실행시간이 데이터의 개수 및 k에 비례하였다. 그러나 제안 기법은 각 데이터에 대해서 병렬로 실행되고 한 번 실행으로 가장 큰/작은 k개의 데이터를 계산하기 때문에 매우 효율적이다.

[0022] 실행시간에 가장 큰 영향을 미치는 병렬로 전송되는 통신 횟수에 대해서, SkLE/SkSE의 복잡도는 데이터 개수 및 k와 무관하다. 이전 기법은 모든 데이터를 비교하는 최대값/최소값 프로토콜을 k번 순차적으로 실행하기 때문에 통신 횟수가 데이터의 개수 및 k에 비례하였다. 그러므로, SkLE/SkSE는 빅데이터 분석과 높은 k를 사용하는 분야에 적합하다.

[0023] SkLE/SkSE의 모든 연산은 두 클라우드 서버에서 실행된다.

[0024] SkLE/SkSE는 연산을 수행하는 두 클라우드 서버에게 데이터, 연산 결과 및 데이터 접근 패턴의 프라이버시를 보호하고, 본 발명의 안전성은 수학적으로 증명된다.

[0025] 동일한 데이터 때문에 가장 높은/낮은 데이터 개수가 k보다 큰 경우, SkLE/SkSE는 동일한 데이터를 모두 결과로 출력하므로 이전보다 더욱 정확한 결과를 출력한다. 이전 기법은 정렬 기법(sorting)을 이용하므로 결과는 동일한 데이터 중에서 임의로 선택되기 때문에, 부정확한 결과를 출력한다.

[0026] 프라이버시를 보호하면서 두 데이터를 비교하는 SCI 프로토콜은 기존 기법의 정보 노출 문제를 해결한다. 기존 기법에서 두 입력 데이터 중 하나의 데이터가 더 크거나 작을 경우(즉, 두 데이터의 값이 같지 않을 경우)와 비교해서, 두 입력 데이터가 같을 경우에는 이전과 다른 값을 출력하기 때문에 정보 노출 문제가 발생한다.

[0027] SCI 프로토콜은 연산을 수행하는 두 클라우드 서버에게 입력 데이터 및 결과의 프라이버시를 보호하고, 본 프로

토콜의 안전성은 수학적으로 증명된다.

도면의 간단한 설명

[0028] 본 발명의 상세한 설명에서 인용되는 도면을 보다 충분히 이해하기 위하여 각 도면의 상세한 설명이 제공된다.

도 1은 본 발명의 일 실시예에 따른 분류 시스템을 도시한다.

도 2는 프라이버시를 보호하면서 두 개의 데이터 (E(s), k)를 비교하는 SCI 프로토콜의 흐름도이다.

도 3은 프라이버시를 보호하면서 k개의 가장 큰 데이터를 계산하는 SkLE의 흐름도이다.

도 4는 PkNC 계산 과정의 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0029] 본 명세서에 개시되어 있는 본 발명의 개념에 따른 실시예들에 대해서 특정한 구조적 또는 기능적 설명들은 단지 본 발명의 개념에 따른 실시예들을 설명하기 위한 목적으로 예시된 것으로서, 본 발명의 개념에 따른 실시예들은 다양한 형태들로 실시될 수 있으며 본 명세서에 설명된 실시예들에 한정되지 않는다.

[0030] 본 발명의 개념에 따른 실시예들은 다양한 변경들을 가할 수 있고 여러 가지 형태들을 가질 수 있으므로 실시예들을 도면에 예시하고 본 명세서에서 상세하게 설명하고자 한다. 그러나, 이는 본 발명의 개념에 따른 실시예들을 특정한 개시 형태들에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물, 또는 대체물을 포함한다.

[0031] 제1 또는 제2 등의 용어는 다양한 구성 요소들을 설명하는데 사용될 수 있지만, 상기 구성 요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성 요소를 다른 구성 요소로부터 구별하는 목적으로만, 예컨대 본 발명의 개념에 따른 권리 범위로부터 벗어나지 않은 채, 제1 구성 요소는 제2 구성 요소로 명명될 수 있고 유사하게 제2 구성 요소는 제1 구성 요소로도 명명될 수 있다.

[0032] 어떤 구성 요소가 다른 구성 요소에 "연결되어" 있다거나 "접속되어" 있다고 언급된 때에는, 그 다른 구성 요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성 요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성 요소가 다른 구성 요소에 "직접 연결되어" 있다거나 "직접 접속되어" 있다고 언급된 때에는 중간에 다른 구성 요소가 존재하지 않는 것으로 이해되어야 할 것이다. 구성 요소들 간의 관계를 설명하는 다른 표현들, 즉 "~사이에"와 "바로 ~사이에" 또는 "~에 이웃하는"과 "~에 직접 이웃하는" 등도 마찬가지로 해석되어야 한다.

[0033] 본 명세서에서 사용한 용어는 단지 특정한 실시예를 설명하기 위해 사용된 것으로서, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 명세서에서, "포함하다" 또는 "가지다" 등의 용어는 본 명세서에 기재된 특징, 숫자, 단계, 동작, 구성 요소, 부분품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성 요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[0034] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가진다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 의미를 갖는 것으로 해석되어야 하며, 본 명세서에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.

[0035] 이하, 본 명세서에 첨부된 도면들을 참조하여 본 발명의 실시예들을 상세히 설명한다. 그러나, 특허출원의 범위가 이러한 실시예들에 의해 제한되거나 한정되는 것은 아니다. 각 도면에 제시된 동일한 참조 부호는 동일한 부재를 나타낸다.

[0037] 도 1은 본 발명의 일 실시예에 따른 분류 시스템을 도시한다.

[0038] 도 1을 참조하면, 본 발명의 일 실시예에 따른 분류 방법은 서로 공모하지 않은 두개의 (클라우드) 서버, 즉 제1 서버(데이터 호스트(data host, DH))와 제2 서버(암호 서비스 제공자(cryptographic service provider,

CSP))에 의해 수행된다. 따라서, 분류 시스템(kNN 분류 시스템 등으로 명명될 수도 있음)은 적어도 제1 서버 및 제2 서버를 포함한다. 또한, 분류 시스템은 데이터 소유자의 단말 및/또는 질의자의 단말을 더 포함할 수도 있다.

[0039] 제1 서버(DH)는 데이터 소유자(data owner, 데이터 소유자의 단말을 의미할 수 있음)로부터 암호화된 데이터셋, 질의자(querier, 질의자의 단말이나 사용자의 단말을 의미할 수 있음)로부터 암호화된 입력 쿼리를 수신하고, 제2 서버(CSP)로부터 암호화 키(encryption key)를 수신한다. 제1 서버(DH)는 제2 서버(CSP)와 함께 PkNC를 수행한 후 입력 쿼리의 클래스(분류 결과)를 암호화된 형태로 (질의자에게) 반환한다(송신한다). 제안하는 PkNC는 데이터셋, 입력 쿼리, 및 분류 결과(클래스)에 대한 어떠한 정보도 노출시키지 않는다.

[0040] 이하에서는, 제안하는 시스템 모델, 보안 정의(decurity definitions), 및 덧셈에 대한 동형 암호 기법으로 Paillier cryptosystem을 소개한다.

[0041] **System Model**

[0042] 제안하는 프로토콜은, 도 1에 도시된 바와 같이, 서로 공모하지 않은 두 개의 (클라우드) 서버들, 즉 제1 서버(DH)와 제2 서버(CSP)에 의해 수행된다. 제2 서버(CSP)는 암호화를 위한 공개키(public key)와 복호화를 위한 비밀키(secret key)를 생성하고, 생성된 공개키를 제1 서버(DH)로 전송한다. 암호화된 입력 데이터를 보유한 제1 서버(DH)는 제2 서버(CSP)와 프로토콜을 수행한다. 프로토콜을 완료한 후에, 제1 서버(DH)는 암호화된 형태로 결과를 사용자(질의자나 질의자의 단말을 의미할 수 있음)에게 반환한다.

[0043] 제안하는 시스템에서, 제1 서버(DH)와 제2 서버(CSP) 누구도 입력 데이터나 결과에 대한 정보를 유출하지 않는다. 자세히 설명하면, 제1 서버(DH)는 암호화된 형태로 프로토콜을 수행하기 때문에, 입력 데이터나 결과에 대한 어떠한 정보도 획득할 수 없다. 제2 서버(CSP)가 제1 서버(DH)로부터 수신한 암호화된 중간 결과를 복호화하기는 하지만, 암호화된 데이터는 임의의 값(random value)에 의해 블라인드되어(blinded) 있기 때문에, 제2 서버(CSP)는 입력 데이터나 결과에 대한 어떠한 정보도 획득할 수 없다. 따라서, 제1 서버(DH)가 제2 서버(CSP)와 공모하지 않는 한, 제안하는 프로토콜은 입력 데이터나 연산 결과가 유출되지 않음을 보장할 수 있다.

[0044] **Adversary Model and Security Definitions**

[0045] **Semi-Honest Adversary Model:** 본 발명에서, 제1 서버(DH)와 제2 서버(CSP)는 semi-honest adversary model로 동작하는 것으로 가정한다. 그러나 악의적인 운영자(예를 들면, 악의적인 제1 서버 운영자 또는 제2 서버 운영자)는 프로토콜에서 입력 데이터의 정보를 얻을 수 있다. 예컨대, 기존 연구들(Park, J.; Lee, D.H. Parallelly running k-nearest neighbor classification over semantically secure encrypted data in outsourced environments. IEEE Access 2020, 8, 64617-64633., Samanthula, B.K.; Elmehdwi, Y.; Jiang, W. K-nearest neighbor classification over semantically secure encrypted relational data. IEEE Trans. Knowl. Data Eng. 2014, 27, 1261-1273.)의 비교 프로토콜에서, 제2 서버(CSP)는, 제1 서버(DH)로부터 수신한 중간 결과를 복호화하고 분석하는 과정에서 두 입력 데이터가 동일한지에 대한 정보를 획득한다. $SkLE_E$/$SkSE_E$(Park, J.; Lee, D.H. Parallelly running k-nearest neighbor classification over semantically secure encrypted data in outsourced environments. IEEE Access 2020, 8, 64617-64633.)에서는 입력 데이터셋에 따라서 종료 시점이 달라지기 때문에, 입력 데이터셋에 대한 정보가 노출된다.

[0046] **Security Definition:** 제안하는 프로토콜에 보안성을 제공하기 위하여, semi-honest adversary model의 보안 정의를 이용한다. 시뮬레이터(simulator)는, 입력과 출력이 주어졌을 때, 실제 프로토콜 실행에서 악의적인 파티(corrupted party)에 대한 뷰(view)를 생성할 수 있음을 증명한다. 악의적인 파티에 대한 뷰는 입력, 내부적으로 생성된 임의의 값(internal coin tosses), 및 수신된 메시지로 구성된다. 시뮬레이터가 실제 실행의 악의적인 파티의 뷰와 구분되지 않는 값들(indistinguishable values)을 생성할 수 있다면, 프로토콜은 안전하다고 말한다. 정의는 다음과 같다.

[0047] 함수가 $f : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^* \times \{0,1\}^*$와 같고, $f_{DH}(x,y)$ (resp., $f_{CSP}(x,y)$)를 $f(x,y)$의 제1 서버(DH)의 (resp., 제2 서버(CSP)의) 요소(element)라 하자. π를 f를 계산하기 위한 2-파티 프로토콜(two-party protocol)이라 하자. (x,y)에 대한 π의 실행 동안의 제1 서버(DH) (resp., 제2 서버(CSP))의 뷰($\text{VIEW}^{\pi}_{DH}(x,y)$ (resp., $\text{VIEW}^{\pi}_{CSP}(x,y)$)로 표기됨)는 $(x, r, m_1, \ldots, m_t)$ (resp., $(y, r, m_1, \ldots, m_t)$)이다. 여기서, r은 제1 서버(DH)에서 생성된 임의의 값을 나타내고, m_i는 수신된 i-번째 메시지를 나타낸다. (x,y)에 대한 π의

실행 이후의 제1 서버(DH)의 (resp., 제2 서버(CSP)의) 출력($\text{OUTPUT}_{DH}^{\pi}(x,y)$ (resp., $\text{OUTPUT}_{CSP}^{\pi}(x,y)$)로 표기됨)은 실행에 대한 파티 자신의 뷰(party's own view) 내에서 암시적이고, $\text{OUTPUT}^{\pi}(x,y) = (\text{OUTPUT}_{DH}^{\pi}(x,y), \text{OUTPUT}_{CSP}^{\pi}(x,y))$이다.

[0048] **Definition 1** (Privacy with respect to semi-honest behavior-general case). 확률적 다항-시간 알고리즘(probabilistic polynomial-time algorithms, 수학식 1 및 수학식 2와 같이 S_{DH}와 S_{CSP}로 표기됨)이 존재하면 π는 비밀스럽게 f를 계산한다고 말하다.

[0049] [수학식 1]

[0050] $$\{(S_{DH}(x, f_{DH}(x,y)), f(x,y))\}_{x,y} \equiv_c \{(VIEW_{DH}^{\pi}(x,y), OUTPUT^{\pi}(x,y))\}_{x,y}$$

[0051] [수학식 2]

[0052] $$\{(S_{CSP}(y, f_{CSP}(x,y)), f(x,y))\}_{x,y} \equiv_c \{(VIEW_{CSP}^{\pi}(x,y), OUTPUT^{\pi}(x,y))\}_{x,y}$$

[0053] 위 수학식에서, $\equiv_c$는 두 분포가 산술적으로 구분불가능함(computationally indistinguishable)을 의미한다. 제안하는 프로토콜의 함수는 확률적이기 때문에, 본 발명에서는 상기 일반적인 케이스의 보안 정의를 이용한다. 이를 통해 제안하는 안전한 비교 프로토콜의 안전성을 증명할 수 있다.

[0054] **Sequential Modular Composition Theorem:**

[0055] sequential modular composition theorem(Canetti, R. Security and composition of multiparty cryptographic protocols. J. Cryptol. 2000, 13, 143-202.)은 프로토콜의 안전성을 분석하기 위해 이용되는 툴이다. π_f가, 함수 g를 계산하기 위해 서브프로토콜 π_g를 호출하는, f를 계산하는 프로토콜이라 가정하자. 정리(theorem)는, π_f의 안전성을 분석하기 위해서, 실제 서브프로토콜 π_g를 실행하는 파티 대신에 이상적으로 g를 계산하는 제3의 파티(third party)가 존재하는 하이브리드 모델에서 π_f의 실행을 고려하는 것으로 충분하다고 말한다. 따라서, 모듈 방식(modular way)으로 프로토콜의 안전성을 분석하기 위해서, π_g의 안전성을 증명한 후 π_f의 안전성을 증명한다.

[0056] **Paillier cryptosystem**

[0057] 부분 동형 암호 기법(partially homomorphic encryption scheme)으로, Paillier cryptosystem(Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Prague, Czech Republic, 2?6 May 1999; Springer: Berlin/Heidelberg, Germany, 1999; pp. 223-238.)이 이용될 수 있다. Paillier cryptosystem은 의미론적 안전성(semantical security)을 갖는 확률론적 비대칭 암호 기법(probabilistic asymmetric encryption scheme)이다. 이는, 주어진 암호화된 데이터에 대하여, 상대방은 오리지널 데이터에 대한 어떠한 정보도 알 수 없음을 의미한다. $E_{pk}(\cdot) = E(\cdot)$를 공개키 pk를 갖는 암호화 함수, $D_{sk}(\cdot) = D(\cdot)$를 비밀키 sk를 갖는 복호화 함수라 하자. Paillier cryptosystem은 오리지널 데이터의 덧셈이 암호화된 형태로 지역적으로 계산되도록 하는 동형 특성(homomorphic property)을 유지한다. 다시 말하면, 두 데이터 $a, b \in \mathbb{Z}_N$가 주어지면, 수학식 3과 수학식 4를 만족한다.

[0058] [수학식 3]

[0059] $$D(E(a) * E(b) mod N^2) = a + b mod N$$

[0060] [수학식 4]

[0061] $$D(E(a)^b mod N^2) = a \cdot b mod N$$

[0062] 간결하게, 본 명세서에서는 $mod N^2$와 $mod N$ 텀(terms)을 생략한다.

[0063] **Performance Evaluation**

[0064] 계산 비용(computational costs, 즉 암호화/복호화 및 지수 연산의 개수)과 통신 비용(communication costs, 즉 통신 라운드의 개수와 통신량)의 관점에서 프로토콜의 성능을 분석한다. 암호화/복호화 및 지수 연산 이외의 다른 연산(예컨대, 동형 덧셈)은 효율성에 미치는 영향이 적기 때문에, 계산 비용으로 고려하지 않는다. 통신량은 프로토콜을 완료하기 위해서 전송된 데이터의 총량을 의미한다. 이는 암호문의 크기인 C의 곱으로 나타난다. 통신 라운드의 개수는 병렬로 수행되는 통신 횟수를 의미한다.

[0065] **Notation**

[0066] $0 \le x < 2^l$인 데이터 x에 대하여, $\langle x \rangle_B = \langle x_{l-1}, \ldots, x_1, x_0 \rangle$는 데이터 x의 이진 표현(binary representation)이다. 여기서 x_0 (resp., x_{l-1})는 LSB(least significant bit)(resp., MSB(most significant bit))을 나타내고, $x_j \in \{0, 1\}$에 대하여 $x = \sum_{j=0}^{l-1} x_j \cdot 2^j$이다. 유사하게, $0 \le x < 2^l$인 암호문 E(x)에 대하여, $\langle E(x) \rangle_B = \langle E(x_{l-1}), \ldots, E(x_1), E(x_0) \rangle$를, 데이터 x의 개별 비트들에 대한 암호문으로 나타낸다. 여기서, $x_j \in \{0, 1\}$에 대하여, $x = \sum_{j=0}^{l-1} x_j \cdot 2^j$이다. $\bar{x}$를 데이터 x의 1의 보수(1's complement)라 하자. 이는 데이터의 모든 비트들을 토글링(toggling)함으로써 계산된다. 예컨대, 이진수 1010의 1의 보수는 0101이다. 유사하게, 비트 x_i에 대하여, x_i의 보수를 $\overline{x_i}$라 하자. 이는 $x_i \in \{0, 1\}$에 대하여, $\overline{x_i} = 1 - x_i$에 의해 계산된다.

[0067] $n \ge 1$에 대하여, [n]은 집합 {1, 2, …, n}을 의미한다. 집합 $I=\{i_1, i_2, \ldots, i_n\}$에 대하여, $\{d_i\}_{i \in I}$는 $\{d_{i_1}, d_{i_2}, \ldots, d_{i_n}\}$를 의미한다. $\{d_i\}_{i \in [l]}$는 벡터 d로 명명될 수 있다. 집합 S에 대하여, $r \in_R S$는 값 r이 집합 S에서 균일하게(uniformly) 임의로 선택되었음을 의미한다. DH → CSP : E(x)는 DH가 CSP로 암호문 E(x)를 전송함을 의미한다. a · b는 정수(integer)에서의 곱셈 연산(multiplication operation)을, E(a) * E(b)는 동형 덧셈을 의미한다. 본 명세서를 통하여, 데이터의 개수는 n으로, Paillier 암호문의 크기(size)는 C로, 그리고 데이터를 표현하기 위해 요구되는 비트의 상한 개수(upper-bound number)는 l로 표시하고, 이는 Paillier cryptosystem의 모듈러스 크기(modulus size) |N| 보다 작거나 같다(즉, $l \le |N|$).

[0068] 이하에서는, 제안하는 안전한 비교(Secure Comparison)와 SkLE/SkSE 프로토콜에 대해 설명한다.

[0069] 기존의 $SkLE_E$는 마지막 l-번째 라운드 이전에 k 개의 가장 큰 요소(elements, 데이터를 의미할 수 있음)를 검색한다(또는 찾는다). 다시 말하면, 기존의 $SkLE_E$는, 종료 시점(end points)이 입력 데이터에 따라 변화하기 때문에, 입력 데이터베이스에 대한 정보를 노출한다. 이하에서는, $SkLE_S$/$SkSE_S$를 이용하여, 이와 같은 정보 유출 문제을 해결한다. $SkLE_S$/$SkSE_S$를 구축하기 위하여, 어떠한 정보도 유출하지 않는 SCI(Secure Comparison and Inequality) 프로토콜을 제안한다.

[0070] **SCI(Secure Comparison and Inequality) Protocol**

[0071] 두 개의 입력 데이터를 비교하기 위한 SCI 프로토콜을 제안한다. 제안하는 SCI 프로토콜은 기존 비교 프로토콜에서 발생하는 정보 노출 문제를 해결할 수 있다. 기존 기법에 의할 경우, 두 입력 데이터가 같지 않을 때(즉, 하나의 입력 데이터가 다른 데이터보다 크거나 작을 때), 제1 서버(DH)는 0 또는 1을 포함하는 임의의 값(random values)으로 구성된 벡터를 제2 서버(CSP)로 전송한다. 그러나, 두 입력 데이터가 같을 때, 제1 서버(DH)는 임의의 값만으로 구성된 벡터를 제2 서버(CSP)로 전송한다. 따라서, 제2 서버(CSP)는 두 입력 데이터가 같은지 그렇지 않은지에 대한 정보를 획득할 수 있다. 제안하는 SCI 프로토콜은, 제1 서버(DH)의 두 입력 데이터가 같지 않을 때 뿐만 아니라 같을 때에도 제2 서버(CSP)로 랜덤 코인(random coin)에 따라 0을 포함하는 임의의 값(random values)으로 구성된 벡터 또는 임의의 값들로만 구성되는 벡터를 전송하기 때문에, 두 입력 데이터에 대한 어떠한 정보도 노출시키지 않는다.

[0072] SCI 함수 F_{SCI}는 제1 서버(DH)로부터 {S′ ,k′ }를, 제2 서버(SCP)로부터 비밀키(SK)를 수신한다. 여기서, $S' = \{\langle E(s) \rangle_B, \langle E(\bar{s}) \rangle_B\}$ 및 $k' = \{\langle k \rangle_B, \langle \bar{k} \rangle_B\}$이다. 또한, $\langle E(s) \rangle_B = \langle E(s_{l-1}), \ldots, E(s_1), E(s_0) \rangle$이고, $\bar{s}$는 데이터 s의 1의 보수(1's complement)이다. 그러면, F_{SCI}는 {E(M),E(D)}를 제1 서버(DH)로 전송한다. 여기서, s<k이면 E(M)=E(1), 그렇지 않으면 E(M)=E(0)이고, s≠k이면 E(D)=E(1)이거나 그렇지 않으면 E(D)=E(0)이다. F_{SCI}는 수

학식 1과 같이 정의된다.

[0073] [수학식 1]

[0074] $$F_{SCI}(\{S', k'\}, SK) \rightarrow (\{E(M), E(D)\}, \perp)$$

[0075] F_{SCI}는 알고리즘 1에 의해 연산되고, 이해를 돕기 위한 예는 표 1에 도시되어 있다. 표 1은 l이 5일 때 SCI 프로토콜을 위한 알고리즘 1의 예를 도시한다. 제안하는 SCI는 비교 결과(E(M)) 뿐만 아니라 비동일성 결과(inequality result, E(D))를 반환하고, 두 입력 데이터셋 중 하나는 평문 형식(plaintext form, k′)을 갖는다. 그러나, SCI 프로토콜의 약간의 수정을 가함으로써, 비교 결과만을 반환하는 일반적인 안전한 비교 프로토콜을 구축할 수 있을 뿐만 아니라 두 입력 데이터가 암호화된 형식을 갖도록 수정할 수도 있다.

[0076] [표 1]

Input	Functionality	j	$E(s_j)$	k_j	$E(w_j)$	$E(x_j)$	$E(y_j)$	$E(\gamma)$	$E(y_0)$	$E(z_j)$	$E(u_j)$	$E(\beta)$	$E(M)$	$E(D)$
(Case 1) $s < k$	$F: s < k$ ($\alpha = 0$)	4	$E(1)$	1	$E(0)$	$E(0)$	$E(0)$			$E(-1)$	$E(r)$			
		3	$E(1)$	1	$E(0)$	$E(0)$	$E(0)$			$E(-1)$	$E(r)$			
		2	$E(0)$	1	$E(r)$	$E(1)$	$E(1)$			$E(0)$	$E(r)$	$E(1)$	$E(1)$	$E(1)$
		1	$E(1)$	0	$E(0)$	$E(1)$	$E(r)$			$E(r)$	$E(r)$			
		0	$E(0)$	1	$E(r)$	$E(1)$	$E(r)$	$E(0)$	$E(r)$	$E(r)$	$E(r)$			
$s = 26$ $k = 29$	$F: s \geq k$ ($\alpha = 1$)	4	$E(1)$	1	$E(0)$	$E(0)$	$E(0)$			$E(-1)$	$E(r)$			
		3	$E(1)$	1	$E(0)$	$E(0)$	$E(0)$			$E(-1)$	$E(r)$			
		2	$E(0)$	1	$E(0)$	$E(1)$	$E(1)$			$E(0)$	$E(0)$	$E(0)$	$E(1)$	$E(1)$
		1	$E(1)$	0	$E(r)$	$E(1)$	$E(r)$			$E(r)$	$E(r)$			
		0	$E(0)$	1	$E(0)$	$E(1)$	$E(r)$	$E(0)$		$E(r)$	$E(r)$			
(Case 2) $s > k$	$F: s < k$ ($\alpha = 0$)	4	$E(1)$	1	$E(0)$	$E(0)$	$E(0)$			$E(-1)$	$E(r)$			
		3	$E(1)$	1	$E(0)$	$E(0)$	$E(0)$			$E(-1)$	$E(r)$			
		2	$E(1)$	0	$E(0)$	$E(1)$	$E(1)$			$E(0)$	$E(0)$	$E(0)$	$E(0)$	$E(1)$
		1	$E(0)$	1	$E(r)$	$E(1)$	$E(r)$			$E(r)$	$E(r)$			
		0	$E(1)$	0	$E(0)$	$E(1)$	$E(r)$	$E(0)$	$E(r)$	$E(r)$	$E(r)$			
$s = 29$ $k = 26$	$F: s \geq k$ ($\alpha = 1$)	4	$E(1)$	1	$E(0)$	$E(0)$	$E(0)$			$E(-1)$	$E(r)$			
		3	$E(1)$	1	$E(0)$	$E(0)$	$E(0)$			$E(-1)$	$E(r)$			
		2	$E(1)$	0	$E(r)$	$E(1)$	$E(1)$			$E(0)$	$E(r)$	$E(1)$	$E(0)$	$E(1)$
		1	$E(0)$	1	$E(0)$	$E(1)$	$E(r)$			$E(r)$	$E(r)$			
		0	$E(1)$	0	$E(r)$	$E(1)$	$E(r)$	$E(0)$		$E(r)$	$E(r)$			
(Case 3) $s = k$	$F: s < k$ ($\alpha = 0$)	4	$E(1)$	1	$E(0)$	$E(0)$	$E(0)$			$E(-1)$	$E(r)$			
		3	$E(1)$	1	$E(0)$	$E(0)$	$E(0)$			$E(-1)$	$E(r)$			
		2	$E(0)$	0	$E(0)$	$E(0)$	$E(0)$			$E(-1)$	$E(r)$	$E(0)$	$E(0)$	$E(0)$
		1	$E(1)$	1	$E(0)$	$E(0)$	$E(0)$			$E(-1)$	$E(r)$			
		0	$E(0)$	0	$E(0)$	$E(0)$	$E(0)$	$E(1)$	$E(1)$	$E(0)$	$E(0)$			
$s = 26$ $k = 26$	$F: s \geq k$ ($\alpha = 1$)	4	$E(1)$	1	$E(0)$	$E(0)$	$E(0)$			$E(-1)$	$E(r)$			
		3	$E(1)$	1	$E(0)$	$E(0)$	$E(0)$			$E(-1)$	$E(r)$			
		2	$E(0)$	0	$E(0)$	$E(0)$	$E(0)$			$E(-1)$	$E(r)$	$E(1)$	$E(0)$	$E(0)$
		1	$E(1)$	1	$E(0)$	$E(0)$	$E(0)$			$E(-1)$	$E(r)$			
		0	$E(0)$	0	$E(0)$	$E(0)$	$E(0)$	$E(1)$		$E(-1)$	$E(r)$			

[0077] r: random value.

[0078] [알고리즘 1]

Algorithm 1: Secure Comparison and Inequality (SCI)

DH input : $\{S', k'\}$ where $S' = \{\langle E(s)\rangle_B, \langle E(\overline{s})\rangle_B\}$ and $k' = \{\langle k\rangle_B, \langle \overline{k}\rangle_B\}$

CSP input : secret key SK

DH output: $\{E(M), E(D)\}$ where $E(M) = \begin{cases} E(1), \text{ if } S < k \\ E(0), \text{ otherwise} \end{cases}$ and

$$E(D) = \begin{cases} E(1), \text{ if } S \neq k \\ E(0), \text{ otherwise} \end{cases}$$

DH:
- toss a random coin $\alpha \in_R \{0, 1\}$
- **for** $j \leftarrow l-1, \ldots, 0$ **do**
  - **if** $\alpha = 0$ **then**
    - $E(w_j) \leftarrow E(\overline{s_j})^{k_j}$
  - **else**
    - $E(w_j) \leftarrow E(s_j)^{\overline{k_j}}$
  - **end**
  - $E(w_j) \leftarrow E(w_j)^{r_j}$ where $r_j \neq 0 \in_R \mathbb{Z}_N$
  - $E(x_j) \leftarrow E(s_j \oplus k_j)$
  - $E(y_j) \leftarrow E(y_{j+1})^{r'_j} * E(x_j)$ where $r'_j \neq 0 \in_R \mathbb{Z}_N$ and $E(y_l) = E(0)$
- **end**

DH, CSP:
- $(E(\gamma), \perp) \leftarrow F_{SZP}(\{E(x_j)\}_{j\in\{0,\ldots,l-1\}}, SK)$

DH:
- **if** $\alpha = 0$ **then**
  - $E(y_0) \leftarrow E(y_0) * E(\gamma)$
- **end**
- **for** $j \leftarrow l-1, \ldots, 0$ **do**
  - $E(z_j) \leftarrow E(y_j) * E(N-1)$
  - $E(u_j) \leftarrow E(z_j)^{r''_j} * E(w_j)$ where $r''_j \in_R \mathbb{Z}_N$
- **end**
- $\{v'_j\}_{j\in\{0,\ldots,l-1\}} \leftarrow \sigma(\{E(u_j)\}_{j\in\{0,\ldots,l-1\}})$
- $DH \rightarrow CSP : \{v'_j\}_{j\in\{0,\ldots,l-1\}}$

CSP:
- $\{v_j\}_{j\in\{0,\ldots,l-1\}} \leftarrow \{D(v'_j)\}_{j\in\{0,\ldots,l-1\}}$
- **if** $\exists v_j = 0 \; in \; \{v_j\}_{j\in\{0,\ldots,l-1\}}$ **then**
  - $E(\beta) \leftarrow E(0)$
- **else**
  - $E(\beta) \leftarrow E(1)$
- **end**
- $CSP \rightarrow DH : E(\beta)$

DH:
- **if** $\alpha = 0$ **then**
  - $E(M) \leftarrow E(\beta)$
- **else**
  - $E(M) \leftarrow E(1) * E(\beta)^{N-1}$
- **end**
- $E(D) \leftarrow E(1) * E(\gamma)^{N-1}$

return $\{E(M), E(D)\}$

[0079]

[0080] 직관적으로, 제1 서버(DH)는, 랜덤 코인 α에 의하여 함수 $F(F : s < k \text{ or } F : s \geq k)$를 선택하고, 두 입력 데이터에 대하여 계산한다. CSP는 계산 결과를 변환하고 변환된 값(β)을 제1 서버(DH)로 반환한다. 그러면, 제1 서버(DH)는, 랜덤 코인 α에 따라, 변환된 값을 결과로 출력한다. 이전 비교 프로토콜에서 제1 서버(DH)의 중간 결과는 두 입력 데이터가 같지 않을 때 0 또는 1을 포함하는 임의의 값으로 구성된 벡터이고, 두 입력 데이터가 같을 때 임의의 값으로만 구성된 벡터이다. 이전 비교 프로토콜을 수정하여 제안하는 SCI 프로토콜에서는 두 입력 데이터가 같지 않을 때 벡터 내 1이 임의의 값이 되도록 수정하였고, 두 입력 데이터가 같을 때 벡터 내의 임의의 값들 중 하나가 랜덤 코인에 따라 0이 되도록 수정하였다. 결과적으로, 제안하는 SCI 프로토콜에서 제1 서버(DH)의 중간 결과는 랜덤 코인에 따라 0을 포함하는 임의의 값들로 구성된 벡터 또는 임의의 값들로만 구성된 벡터가 된다.

[0081] 두 입력 데이터가 같을 때 제1 서버(DH)의 중간 결과가 임의의 값들로만 구성되는 벡터가 되는 시나리오를 피하기 위해서, 모든 입력 데이터가 0인지 또는 그렇지 않은지 여부를 계산하는 함수 F_{SZP}를 이용한다. 함수 F_{SZP}는 제1 서버(DH)로부터 $\{E(x_i)\}_{i\in[l]}$를 제2 서버(CSP)로부터 비밀키(SK)를 수신한 다음, $E(\gamma)$를 제1 서버(DH)로 송신

한다. 여기서, 모든 x_i=0이면 $E(\gamma) = E(1)$이고, 그렇지 않으면 $E(\gamma) = E(0)$이다. F_{SZP}는 수학식 2와 같이 정의된다.

[0082] [수학식 2]

[0083] $$F_{SZP}(\{E(x_i)\}_{i\in[l]}, SK) \rightarrow (E(\gamma), \perp)$$

[0084] F_{SZP}에 대한 알고리즘 A1은 다음과 같다.

[0085] [알고리즘 A1]

Algorithm A1: Secure Zero Protocol (SZP)

DH input : $\{E(x_i)\}_{i\in[l]}$
CSP input : secret key SK
DH output: $E(\gamma) = \begin{cases} E(1), \text{ if } x_i = 0 \text{ for all } i \in [l] \\ E(0), \text{ otherwise} \end{cases}$

1 **DH:**
2 toss a random coin $\alpha \in_R \{0,1\}$
3 **if** $\alpha = 0$ **then**
4 **for** $i \leftarrow 1, \ldots, l$ **do**
5 $E(s_i) \leftarrow E(-1) * E(x_i) * (\prod_{j=i+1}^{l} E(x_j))^2$
6 $E(c_i) \leftarrow E(s_i)^{r_i}$ where $r_i \neq 0 \in_R \mathbb{Z}_N$
7 **end**
8 **else**
9 $E(s_1) \leftarrow \prod_{i=1}^{l} E(x_i)$
10 $E(c_1) \leftarrow E(s_1)^{r_1}$ where $r_1 \neq 0 \in_R \mathbb{Z}_N$
11 **for** $i \leftarrow 2, \ldots, l$ **do**
12 $E(c_i) \leftarrow E(r_i)$ where $r_i \neq 0 \in_R \mathbb{Z}_N$
13 **end**
14 **end**
15 $\{d'_i\}_{i\in[l]} \leftarrow \sigma(\{E(c_i)\}_{i\in[l]})$
16 $DH \rightarrow CSP : \{d'_i\}_{i\in[l]}$
17 **CSP:**
18 $\{d_i\}_{i\in[l]} \leftarrow \{D(d'_i)\}_{i\in[l]}$
19 **if** $\exists d_i = 0$ *in* $\{d_i\}_{i\in[l]}$ **then**
20 $E(\beta) \leftarrow E(0)$
21 **else**
22 $E(\beta) \leftarrow E(1)$
23 **end**
24 $CSP \rightarrow DH : E(\beta)$
25 **DH:**
26 **if** $\alpha = 0$ **then**
27 $E(\gamma) \leftarrow E(\beta)$
28 **else**
29 $E(\gamma) \leftarrow E(1) * E(\beta)^{N-1}$
30 **end**
31 **return** $E(\gamma)$

[0086]

[0087] 다시 알고리즘 1을 참조하면, 제1 서버(DH)는, 랜덤 코인 α를 던짐으로써 함수 F를 선택한다(line 2). 여기서, α=0이면 F:s<k, 그렇지 않으면 F:s≥k이다. 제1 서버(DH)가 함수 F:s<k(resp., F:s≥k)를 선택할 때, (s_j,k_j)=(0,1) (resp., (s_j,k_j)=(1,0))이면 w_j는 랜덤이고, 그렇지 않으면 $w_j = 0$ (lines 4–9)이다. $s_j \neq k_j$이면 x_j=1이고, 그렇지 않으면 x_j=0이다(line 10). (l-1)번째 비트로부터 $s_j \neq k_j$인 첫번째 비트에서 y_j=1이고, 다른 y_j는 0 또는 임의의 값이다(line 11). (l-1)번째 비트로부터 $s_j \neq k_j$인 첫번째 비트를 t-번째 비트라 하자. 그러면, y_t=1, j= t+1, …, l-1에 대하여 y_j=0, 그리고 j=0, …, t-1에 대하여 y_j는 임의의 값을 갖는다.

[0088] s가 k와 같다면 $\gamma = 1$이고, 그렇지 않다면 $\gamma = 0$이다(line 14). line 10 에서, $s_j \neq k_j$이면 x_j=0이고, 그렇지 않으면 x_j=1이기 때문이다. 제1 서버(DH)가 $F : s < k\ (\alpha = 0)$를 선택한다면, 제1 서버(DH)는 γ을 y_0에 더함으로써, s=k일 때, CSP에 0을 포함하는 임의의 값들로 구성된 벡터를 전송할 수 있다. 그리고, 제2 서버(CSP)는 $\beta = 0$을 제1 서버(DH)로 다시 반환한다(line 14-18). γ가 y의 고정된 0-번째 위치에 더해짐에도, 제2

서버(CSP)는, 위치에 대한 정보는 치환(permutation) σ에 의해 제거되기 때문에(line 23), 위치를 알 수 없다. 제1 서버(DH)가 $F:S\geq k\,(\alpha=1)$를 선택한다면, 제1 서버(DH)는 γ를 더하지 않는다. 이에 따라 제1 서버(DH)는 CSP에 임의의 값들로만 구성된 벡터를 전송할 수 있다. 그리고, 제2 서버(CSP)는 β=1을 제1 서버(DH)로 다시 반환한다.

[0089] y_j=1인 위치를 t-번째 비트라 하자. 선택된 함수 F가 두 입력 데이터의 관계와 상이할 때, 즉 제1 서버(DH)가 $F:s<k$를 선택하고 두 입력 데이터셋의 관계가 s>k이거나, 또는 제1 서버(DH)가 $F:s\geq k$를 선택하고 두 입력 데이터셋의 관계가 s<k일 때, u_t=0이고 다른 u_j는 랜덤이다(lines 19-22). 반대로, 선택된 함수 F가 두 입력 데이터와 같을 때, 즉 제1 서버(DH)가 $F:s<k$를 선택하고 두 입력 데이터셋의 관계는 s<k이거나, 또는 제1 서버(DH)가 $F:s\geq k$를 선택하고 두 입력 데이터셋의 관계가 s>k일 때, w_j를 더함으로써 모든 u_j는 랜덤이다(line 21). 또한, 두 입력 데이터가 동일하고(s=k) 선택된 함수가 F:s<k일 때, 즉 선택된 함수 F가 두 입력 데이터의 관계와 상이하다면, u는 u_0=0을 포함하는 임의의 값들로 구성되는 벡터이다. 제1 서버(DH)가 $\gamma=1$을 y_0에 더했기 때문이다(lines 16-18). 선택된 함수가 F:s≥k이면, 즉 선택된 함수 F가 두 입력 데이터의 관계와 동일할 때, u는 임의의 값들로만 구성된 벡터이다. $\{u_j\}_{j\in\{0,\ldots,l-1\}}$의 치환 σ 이후에, 제1 서버(DH)는 치환된 벡터 v를 제2 서버(CSP)로 전송한다(lines 23-24).

[0090] 제2 서버(CSP)는, 제2 서버(CSP)가 0을 포함하는 임의의 값들로 구성된 벡터(resp., 임의의 값들로만 구성된 벡터)를 수신하면, β=0 (resp., β=1)를 제1 서버(DH)로 다시 전송한다. 자세히 설명하면, 제2 서버(CSP)는 v'_j를 복호화하여 v_j를 획득한다(line 26). 벡터 v 내에 0이 포함되었으면, 제2 서버(CSP)는 β=0를 제1 서버(DH)로 전송한다. 벡터 v가 임의의 값들로만 구성된다면, 제2 서버(CSP)는 β=1을 제1 서버(DH)로 전송한다(lines 27-32). 벡터 u 내에 0을 갖는 요소의 위치가 $s_j\neq k_j$인 첫번째 위치 또는 s=k일 때의 0-번째 비트에 대한 정보를 포함하고 있지만, 제2 서버(CSP)는 어떠한 정보도 획득할 수 없다. line 23에서 랜덤 치환에 의해 제거되기 때문이다. 제1 서버(DH)는 선택된 α에 따라서 β를 계산하여 결과 M을 계산한다. 즉, 제1 서버(DH)는 결과 M을, α=0이면 β(M←β)로, α=1이면 β(M←1-β)의 보수(complement)로 설정한다(lines 34-38). 결과적으로, 결과 M은 입력 데이터 s와 k에 대하여, 조건 $M=(s<k)=(\alpha\oplus\beta)$을 만족시킨다. 제1 서버(DH)는 입력 데이터 s와 k에 대하여, 비동일성 결과(inequality result) D를 계산한다(line 39).

[0091] **Computation and communication costs:** SCI 프로토콜은 2l의 암호화/복호화 및 (4l+3)의 지수 계산(exponential computation)을 요구하고, 2(l+1)·C 비트가 2 라운드 내에서 전송된다. 자세히 설명하면, SZP는 line 14에서 l의 암호화/복호화를 요구하고, line 14에서 제2 서버(CSP)는 j=0, …, l-1에 대하여 v'_j를 복호화한다. 지수 계산은 lines 9, 11, 및 21에서 3l회, line 37과 39에서 2회 수행된다. k_j는 {0, 1} 내의 공개값(public value)이기 때문에, line 5와 7에서 지수 계산은 제외된다. 따라서, SCI의 전체 지수 계산은, line 14에서 (l+1)회를 포함하는 (4l+3)이다. 통신량(amount of communication)은 line 14에서 (l+1)·C 비트를 포함하는 2(l+1)·C 비트이다. line 14에서의 1회를 포함하는 두 번의 통신 라운드가 존재한다.

[0092] **Secure Version of SkLE/SkSE ($SkLE_S$/$SkSE_S$)**

[0093] 어떠한 정보도 노출시키지 않으며, 어레이(array) 내에서 k 개의 가장 큰 요소들(즉, 데이터셋 내의 k 개의 가장 큰 데이터)를 연산하는 $SkLE_S$를 설명한다. $SkLE_S$의 장점은 매우 효율적이라는 것이다. 각 요소가 병렬로 실행되기 때문이다. k 개의 가장 큰 요소들을 연산하기 위해여, 기존 프로토콜의 통신 라운드(communication rounds)는 요소의 개수와 최근접 이웃(nearest neighbors)의 파라미터 k에 비례한다. 직렬적으로(serially) 맥시멈 프로토콜(maximum protocol)을 k 회 반복하기 때문이다. 그러나, 제안하는 $SkLE_S$는 각 요소에 대하여 병렬적으로 수행되고 k 개의 가장 큰 요소들을 단일의 실행으로 연산하기 때문에, 통신 라운드는 요소의 개수 및 파라미터 k와 독립적이다. 따라서, 많은 데이터를 처리하는 빅데이터 분석 및 높은 k값을 갖는 PkNC 응용에 적합하다. $SkLE_S$는 요소의 개수만큼 연산을 동시에 수행할 경우에 최상의 성능을 제공하기 때문에, 복수의 병렬 연산이 가능한 클라우드 컴퓨팅 환경에서 성능은 매우 개선된다. 또한, 제안하는 $SkLE_S$는 $SkLE_E$에서 발생하는 정보

공개 문제를 해결할 수 있다.

[0094] F_{SkLE_S}는 제1 서버(DH)로부터 암호화된 요소들의 집합 $\{\langle E(e_i)\rangle_B\}_{i\in[n]}$을, 제2 서버(CSP)로부터 비밀키 SK를 수신하고, $\{E(K_i)\}_{i\in[n]}$를 제1 서버(DH)로 송신한다. 이때, 요소 e_i가 k 개의 가장 큰 요소들의 집합에 속하면 K_i=1이고, 그렇지 않으면 K_i=0이다. F_{SkLE_S}는 수학식 11과 같이 정의된다.

[0095] [수학식 11]

[0096] $$F_{SkLE_S}(\{\langle E(e_i)\rangle_B\}_{i\in[n]}, SK) \rightarrow (\{E(K_i)\}_{i\in[n]}, \perp)$$

[0097] 입력 데이터셋 내의 각 요소 e_i는 $\{K_i, P_i, C_i\} \in \{0, 1\}^3$으로 구성된 보조 데이터(auxiliary data)를 갖고, $SkLE_S$는 각 라운드에 각 요소들의 보조 데이터를 계산함으로써 k 개의 가장 큰 요소들을 계산한다. K_i는 대응되는 요소 e_i가 k 개의 가장 큰 요소들의 집합에 포함되는지 여부를 나타내는 $SkLE_S$의 출력이다. P_i는 대응되는 라운드에서 요소 e_i가 k 개의 가장 큰 요소로 예상되는지 여부를 나타낸다. $SkLE_S$는 후보 요소들의 집합에서 k 개의 가장 큰 요소들과 예측된 k 개의 가장 큰 요소를 찾는다. 여기서 C_i는 대응되는 요소 e_i가 후보 요소들의 집합에 포함되는지 여부를 나타낸다. 일단 요소 e_i가 k 개의 가장 큰 요소들의 집합에 포함되면, 이는 철회될 수 없다(즉, Ki=0 → 1 but $1 \nrightarrow 0$). 유사하게, 일단 요소 e_i가 후보 요소들의 집합으로부터 배제되면, 이는 철회될 수 없다(즉, C_i=1 → but $0 \nrightarrow 1$).

[0098] $SkLE_S$는 (l-1)번째 비트(MSB) 부터 0번째 비트(LSB)까지 매 라운드마다 모든 요소들의 1 비트에 대하여 보조 데이터를 계산한다. 여기서, l은 요소(데이터)의 길이를 의미한다. 결과적인 k 개의 가장 큰 요소들은 K_i=1인 요소 e_i이고, 이는 요소들이 k 개의 가장 큰 요소들의 집합에 포함됨을 의미한다. 알고리즘 2에는 F_{SkLE_S}를 계산하는 프로토콜이 도시되어 있고, 표 3에는 이해의 편의를 위한 예제가 도시되어 있다. 제1 서버(DH)는 알고리즘 2의, F_{SM}, F_{SBD}, 및 F_{SCI}를 위한 인터랙티브 프로토콜들(interactive protocols)을 제외하고, 모든 계산을 지역적으로 수행한다. n은 모든 요소들의 개수이고 l은 요소 e_i를 나타내기 위해 요구되는 비트의 개수의 상한(upper-bound)이다.

[0099] $SkLE_S$의 아이디어는, k 개의 가장 큰 요소들을 찾은 후에, 결과가 계속 유지되도록 후보 요소들의 집합에 있는 모든 요소들을 제거하는 것이다. 기존의 $SkLE_E$는 k 개의 가장 큰 요소들을 찾은 이후에 종료되기 때문에, k 개의 가장 큰 요소들을 위한 보조 데이터의 연산을 고려할 필요가 없었다. 제안하는 $SkLE_S$는 k 개의 가장 큰 요소들을 찾은 이후에 종료되지 않고 일관적으로 l 라운드를 실행하기 때문에, k 개의 가장 큰 요소들에 대한 결과가 마지막까지 유지되어야 한다. 이를 위해, $SkLE_S$가 k 개의 가장 큰 요소들을 찾은 후에, 후보 집합에 있는 모든 요소들을 제거한다($C_i \leftarrow 0$). 왜냐하면, k 개의 가장 큰 요소들은 후보 요소들의 집합 내에서 찾기 때문이다.

[0100] [표 3]

j	$\{\langle E(e_i)\rangle_B\}_{i\in[5]}$ [1] / $\{E(K_i)\}_{i\in[5]}$	$\{E(P_i)\}_{i\in[5]}$ / $\{E(C_i)\}_{i\in[5]}$	$E(M)$ / $E(D)$
.	.	.	.
	$E(0),E(0),E(0),E(0),E(0)$	$E(1),E(1),E(1),E(1),E(1)$	.
7	$E(0),E(0),E(0),E(0),E(0)$	$E(0),E(0),E(0),E(0),E(0)$	$E(1)$
	$E(0),E(0),E(0),E(0),E(0)$	$E(1),E(1),E(1),E(1),E(1)$	$E(1)$
6	$E(1),E(0),E(0),E(0),E(0)$	$E(1),E(0),E(0),E(0),E(0)$	$E(1)$
	$E(1),E(0),E(0),E(0),E(0)$	$E(0),E(1),E(1),E(1),E(1)$	$E(1)$
5	$E(0),E(1),E(1),E(1),E(1)$	$E(1),E(1),E(1),E(1),E(1)$	$E(0)$
	$E(1),E(0),E(0),E(0),E(0)$	$E(0),E(1),E(1),E(1),E(1)$	$E(1)$
4	$E(0),E(1),E(0),E(0),E(0)$	$E(1),E(1),E(0),E(0),E(0)$	$E(1)$
	$E(1),E(1),E(0),E(0),E(0)$	$E(0),E(0),E(1),E(1),E(1)$	$E(1)$
3	$E(1),E(0),E(1),E(1),E(0)$	$E(1),E(1),E(1),E(1),E(0)$	$E(0)$
	$E(1),E(1),E(0),E(0),E(0)$	$E(0),E(0),E(1),E(1),E(0)$	$E(1)$
2	$E(0),E(1),E(1),E(0),E(1)$	$E(1),E(1),E(1),E(0),E(0)$	$E(0)$
	$E(1),E(1),E(1),E(0),E(0)$	$E(0),E(0),E(0),E(0),E(0)$	$E(0)$
1	$E(0),E(1),E(0),E(0),E(1)$	$E(1),E(1),E(1),E(0),E(0)$	$E(0)$
	$E(1),E(1),E(1),E(0),E(0)$	$E(0),E(0),E(0),E(0),E(0)$	$E(0)$
0	$E(1),E(0),E(1),E(1),E(0)$	$E(1),E(1),E(1),E(0),E(0)$	$E(0)$
	$E(1),E(1),E(1),E(0),E(0)$ [2]	$E(0),E(0),E(0),E(0),E(0)$	$E(0)$

Parameters : $k = 3$, $l = 8$; [1] input: $\{\langle E(e_i)\rangle_B\}_{i\in[5]} = \{\langle E(73)\rangle_B, \langle E(54)\rangle_B, \langle E(45)\rangle_B, \langle E(41)\rangle_B, \langle E(38)\rangle_B\}$; [2] output $\{E(K_i)\}_{i\in[5]} = \{E(1),E(1),E(1),E(0),E(0)\}$ means $\{73,54,45\}$ are the three largest elements in array $\{73,54,45,41,38\}$.

[0101]

[0102] 알고리즘 2에 대한 이해의 편의를 위해, 암호화 없이 데이터를 직관적으로 설명한다. 먼저, 제1 서버(DH)는, 모든 요소들 e_i의 보조 데이터 K_i 및 C_i를 초기화한다. 즉, k 개의 가장 큰 요소들의 집합 내에는 어떠한 요소가 존재하지 않고(즉, $K_i \leftarrow 0$), 모든 요소들은 후보 요소들의 집합에 포함된다(즉, $C_i \leftarrow 1$)(lines 2-5)고 설정한다. 매 라운드마다 제1 서버(DH)와 제2 서버(CSP)는 모든 요소 e_i의 1 비트에 대하여 보조 데이터 P_i, K_i, 및 $C_i \in \{0, 1\}^3$을 계산하고(lines 6-24), 계산 과정은 아래와 같이 4개의 스텝으로 구성된다. 우리는 제1 서버(DH)와 제2 서버(CSP)가 (l-j)-번째 라운드(j=l-1, …, 0)에서 모든 요소들의 j-번째 비트에 대해서 보조 데이터를 계산한다고 가정한다.

[0103] [알고리즘 2]

Algorithm 2: Secure version of SkLE ($SkLE_S$)

DH input : set of encrypted elements $\{\langle E(e_i)\rangle_B\}_{i\in[n]}$
CSP input : secret key SK
DH output: $\{E(K_i)\}_{i\in[n]}$ where $K_i = 1$ if an element e_i is included in the set of k largest elements; otherwise, $K_i = 0$.

1 DH locally performs all computations besides the interactive protocols for functionalities F_{SM}, F_{SBD}, and F_{SCI}.
2 **for** $i \leftarrow 1,\ldots,n$ **do**
3 $\quad E(K_i) \leftarrow E(0)$
4 $\quad E(C_i) \leftarrow E(1)$
5 **end**
6 **for** $j \leftarrow l-1,\ldots,0$ **do**
7 $\quad$ **for** $i \leftarrow 1,\ldots,n$ **do**
8 $\quad\quad (E(u_i),\perp) \leftarrow F_{SM}(\{E(e_{i,j}),E(C_i)\},SK)$
9 $\quad\quad E(P_i) \leftarrow E(K_i) * E(u_i)$
10 $\quad$ **end**
11 $\quad E(s) \leftarrow \prod_{i=1}^{n} E(P_i)$
12 $\quad (S',\perp) \leftarrow F_{SBD}(E(s),SK)$
13 $\quad (\{E(M),E(D)\},\perp) \leftarrow F_{SCI}(\{S',K'\},SK)$
14 $\quad (E(\alpha),\perp) \leftarrow F_{SM}(\{E(D),E(M)\},SK)$
15 $\quad E(\beta) \leftarrow E(1) * E(D)^{N-1} * E(\alpha)$
16 $\quad E(\gamma) \leftarrow E(D) * E(\alpha)^{N-2}$
17 $\quad$ **for** $i \leftarrow 1,\ldots,n$ **do**
18 $\quad\quad (E(v_i),\perp) \leftarrow F_{SM}(\{E(u_i),E(\beta)\},SK)$
19 $\quad\quad E(K_i) \leftarrow E(K_i) * E(v_i)$
20 $\quad\quad (E(w_i),\perp) \leftarrow F_{SM}(\{E(e_{i,j},E(\gamma)\},SK)$
21 $\quad\quad E(x_i) \leftarrow E(M) * E(w_i)$
22 $\quad\quad (E(C_i),\perp) \leftarrow F_{SM}(\{E(C_i),E(x_i)\},SK)$
23 $\quad$ **end**
24 **end**
25 **for** $i \leftarrow 1,\ldots,n$ **do**
26 $\quad E(K_i) \leftarrow E(K_i) * E(C_i)$
27 **end**
28 **return** $\{E(K_i)\}_{i\in[n]}$

[0104]

[0105] (Step 1: lines 7-10) privately computing predicted k-largest element (P_i): j-번째 비트에서 예상되는 k 개의 가장 큰 요소는 (l-1)-번째 비트로부터 j-번째 비트까지 비트 1이 적어도 한번 나타나는 요소 e_i이다. 즉, j-번째 비트에서 예측된 k 개의 가장 큰 요소는 j-번째 비트가 1($e_{i,j}$=1)인 후보 요소(C_i=1)이거나 이전 라운드에서 k 개의 가장 큰 요소(K_i=1)로 결정된 요소들이다. 따라서, $E(P_i)$는 수학식 12와 같이 계산된다.

[0106] [수학식 12]

[0107] $$E(P_i) \leftarrow E(e_{i,j} \cdot C_i + K_i)$$

[0108] (Step 2: line 11) privately computing the number of all predicted k-largest elements (s): 예측된 k 개의 가장 큰 요소를 위한 보조 데이터의 값은 0 또는 1이기 때문에(즉, $P_i \in \{0, 1\}$), 모든 예측된 k 개의 가장 큰 요소들의 개수 s는 수학식 13과 같이 값들을 모두 더함으로써 연산된다.

[0109] [수학식 13]

[0110] $$E(s) \leftarrow E(\textstyle\sum_{i=1}^{n} P_i) = \prod_{i=1}^{n} E(P_i)$$

[0111] (Step 3: lines 12-13) privately comparing the number s of all predicted k-largest elements to parameter k of nearest neighbors: s와 k의 비교를 위해, 제1 서버(DH)와 제2 서버(CSP)는, F_{SCI}를 계산하는 서버와 인터랙티브 프로토콜을 실행한다. F_{SCI}의 입력을 위한 S′ 을 계산하기 위하여, 제1 서버(DH)와 제2 서버(CSP)는 또한, F_{SBD}를 계산하기 위하여, 인터랙티브 프로토콜을 실행한다. 본 명세서에서 k′ ={$\langle k\rangle_B$, $\langle k\rangle_B$}를 계산하는 방법에 대한 상세한 설명은 생략한다. 왜냐하면, k는 공개 파라미터(public parameter)이고 k′ 은 용이하게 계산될 수 있기 때문이다.

[0112] (Step 4: lines 14-23) privately computing k largest elements (K_i) and candidate elements (C_i): 여기서, 우리는 예측된 k 개의 가장 큰 요소를 해당 라운드의 비트가 1($e_{i,j}$=1)인 후보 요소(C_i=1)라고 하고, 마찬가지로, 예측되지 않은 k 개의 가장 큰 요소는 해당 라운드의 비트가 0($e_{i,j}$=0)인 후보 요소(C_i=1)라 하자. 제1 서버(DH)와 제2 서버(CSP)는, 예측된 k 개의 가장 큰 요소들의 개수 s와 파라미터 k의 비교 결과(M 및 D)에 따라, 요소 (e_i)가 k 개의 가장 큰 요소들의 집합(K_i) 및 후보 요소들의 집합(C_i)에 포함되는지 여부를 계산한다. K_i와 C_i를 계산하기 위한 아이디어는, s<k 이면 예측된 k 개의 가장 큰 요소들을 k 개의 가장 큰 요소들의 집합에 포함시키고, s>k이면 예측되지 않은 k 개의 가장 큰 요소들을 후보 데이터의 집합에서 제외시키는 것이다. s=k이면 모든 예측된 k 개의 가장 큰 요소들은 k 개의 가장 큰 요소들의 집합에 포함되고, k 개의 가장 큰 요소들의 집합을 결과로 유지하기 위하여 다른 후보 요소들(즉, 예측되지 않은 k 개의 가장 큰 요소들)은 후보 요소들의 집합으로부터 배제된다. 왜냐하면, k 개의 가장 큰 요소들은 후보 요소들의 집합에서 발견되기 때문이다.

[0113] 표 4는, 각 케이스에 따른, 요소 e_i에 대한 C_i와 K_i의 값들을 도시한다. (case 1) s<k일 때(즉, 예측된 k 개의 가장 큰 요소들의 개수가 파라미터 k 보다 작을 때), 예측된 k 개의 가장 큰 요소(즉, $e_{i,j}$=1이고 후보 요소들의 집합 내에 포함된 e_i)는 k 개의 가장 큰 요소들의 집합 내에 포함되고($K_i \leftarrow 1$), 후보 요소들의 집합으로부터 배제된다($C_i \leftarrow 0$). (case 2) s>k일 때, 예측되지 않은 k 개의 가장 큰 요소(즉, $e_{i,j}$=0이고 후보 요소들의 집합 내에 포함된 e_i)는 후보 요소들의 집합으로부터 배제된다($C_i \leftarrow 0$). (case 3) s=k일 때, 예측된 k 개의 가장 큰 요소는 k 개의 가장 큰 요소들의 집합에 포함되고($K_i \leftarrow 1$), 후보 요소들의 집합으로부터 배제된다($C_i \leftarrow 0$). (case 4) 그리고, 후보 요소들의 집합 내의 다른 요소들(즉, 예측되지 않은 k 개의 가장 큰 요소들)은 집합으로부터 배제된다($C_i \leftarrow 0$). (case 5) 후보 요소들의 집합 내에 존재하지 않는 모든 요소들의 K_i와 C_i의 값들은 동일하게 유지된다. 표 4에 따라, $E(K_i)$와 $E(C_i)$는 수학식 14와 수학식 15와 같이 계산된다.

[0114] [수학식 14]

[0115] $$E(K_i) \leftarrow E(K_i + e_{i,j} \cdot C_i \cdot (1 - D + D \cdot M))$$

[0116] [수학식 15]

[0117] $$E(C_i) \leftarrow E(C_i \cdot (M + e_{i,j} \cdot D \cdot (1 - 2M)))$$

[0118] [표 4]

	Candidate $(K_i = 0, C_i = 1)$		**Non-Candidate** $(C_i = 0)$	
	Predicted $(e_{i,j} = 1)$	**Unpredicted** $(e_{i,j} = 0)$	$e_{i,j} = 1$	$e_{i,j} = 0$
$s > k$	(case 2) $K_i \rightarrow K_i$ $C_i \rightarrow C_i$	(case 2) $K_i \rightarrow K_i$ $C_i \rightarrow 0$		
$s < k$	(case 1) $K_i \rightarrow 1$ $C_i \rightarrow 0$	$K_i \rightarrow K_i$ $C_i \rightarrow C_i$	(case 5) $K_i \rightarrow K_i$ $C_i \rightarrow C_i$	
$s = k$	(case 3) $K_i \rightarrow 1$ $C_i \rightarrow 0$	(case 4) $K_i \rightarrow K_i$ $C_i \rightarrow 0$		

[0119]

[0120] 동일한 값들 때문에 마지막 라운드를 실행한 후에도 k 개의 가장 큰 요소들을 결정할 수 없을 때, $SkLE_S$는 k 개의 가장 큰 요소들의 집합과 후보 요소들의 집합 내의 요소들을 모두 결과로 반환한다(lines 25-27). 예컨대, $SkLE_S$가 어레이 {1, 2, 3, 3, 4, 5} 내의 3개의 가장 큰 요소들(k=3)을 찾을 때, 4 개의 가장 큰 요소들 {3, 3, 4, 5}를 결과로 반환한다.

[0121] Parallelism: 제안하는 $SkLE_S$는 각 요소에 대해서 병렬로 연산하거나 또는 모든 요소들에 대해서 연산을 수행한

다. 따라서, 실행 시간(running time)과 관련된 통신 라운드들은 요소들의 개수와 독립적이다. lines 7-10 및 lines 17-23에서의 연산은 각 요소에 대하여 독립적이고 병렬로 계산된다. lines 12-16에서의 연산은 요소들이 개수와 무관하게 공동으로 계산된다. line 11에서 요소들의 개수 만큼의 동형 덧셈(homomorphic additions)이 시리얼하게 계산되지만, 동형 덧셈은 효율성에 거의 영향을 미치지 않기 때문에 이를 연산 비용이나 통신 비용으로 고려할 필요는 없다.

[0122] Computation and communication costs: $SkLE_S$는 $(24n + 5l' + 7)\cdot l$의 암호화/복화화 및 $(8n + 8l' + 9)\cdot l$의 지수 계산을 요구하고, $(12n + 4l' + 7)\cdot l\cdot C$ 비트가 $(l' + 8)\cdot l$ 라운드에서 전송된다. 여기서 l은 요소 e_i의 크기(size)를, l' 은 요소들의 개수의 크기를 나타낸다. SM은 6회의 암호화/복호화 및 2회의 지수 연산을 요구하고, $3\cdot C$ 비트가 하나의 라운드에서 전송된다. 자세히 설명하면, line 8에서 SM은 6nl의 암호화/복호화 및 2nl의 지수 연산을 요구하고, $3nl\cdot C$ 비트가 전송된다. n 개의 요소들에 대한 SMs가 l 번 반복되기 때문이다. 그러나, n 개의 요소들을 위한 SMs은 병렬로 수행되기 때문에 통신 라운드의 개수는 l이다. 유사하게, lines 18-22 내의 SM 연산은 nl회 실행되고, lines 11-16 내의 연산은 l 번 수행된다.

[0123] Secure version of SkSE ($SkSE_S$): $SkSE_S$는, 입력 어레이 또는 결과에 대한 정보의 공개 없이, k 개의 가장 작은 요소들을 계산한다. 결과적으로, $SkSE_S$는 입력 어레이의 1의 보수(1's complement)를 아래와 같이 $SkLE_S$에 입력함으로써 구축된다.

[0124] [수학식 16]

[0125] $$SkSE(\{\langle E(e_i)\rangle_B\}_{i\in[n]}) = SkLE(\{\langle E(\overline{e_i})\rangle_B\}_{i\in[n]})$$

[0126] $SkSE_S$를 구축하기 위해, SkLES와 유사한 프로세스를 따른 결과 상기 결론에 도달하였다.

[0127] 도 2는 프라이버시를 보호하면서 두 개의 데이터 (E(s), k)를 비교하는 SCI 프로토콜의 흐름도이다. 여기서 l은 데이터의 길이를 나타내고, 아래 첨자를 추가한 값은 비트값을 나타낸다. 예를 들면, k_j는 k의 l개 비트 중에서 j번째 비트를 나타내고, $E(s_j)$는 s의 l개 비트 중에서 j번째 비트의 암호문을 나타낸다.

[0128] 110단계에서, 제1 서버(DH)는 $\alpha\in\{0, 1\}$를 임의로 선택한다.

[0129] 120단계에서, 제1 서버(DH)는 α의 값에 따라서 모든 l개의 비트 암호문 $E(w_j)$를 다음과 같이 계산한다.

[0130] $$E(w_j)\leftarrow\begin{cases} E(\overline{s_j}*k_j*r_j) & (\alpha=0) \\ E(s_j*\overline{k_j}*r_j) & (otherwise)\end{cases}$$ (r_j는 0이 아닌 랜덤값, j=1, …, l)

[0131] 직관적으로, 110단계에서 α=0(resp., α=1)을 선택했을 때, $(s_j,k_j)=(0,1)$(resp., $(s_j,k_j)=(1,0)$)인 경우 w_j는 랜덤값이고 그 외에는 $w_j\leftarrow 0$이 된다.

[0132] 130단계에서, 제1 서버(DH)는 모든 l개의 암호문 $E(x_j)$를 다음과 같이 계산한다.

[0133] $$E(x_j) \leftarrow E(s_j\oplus k_j)$$ (j=1, …, l)

[0134] 직관적으로, $s_j\neq k_j$인 경우 $x_j\leftarrow 1$이고 그 외에는 $x_j\leftarrow 0$이 된다.

[0135] 140단계에서, 제1 서버(DH)는 모든 l개의 암호문 $E(y_j)$를 다음과 같이 계산한다.

[0136] $$E(y_j) \leftarrow E(y_{j+1}*r_j'+x_j)$$ (r_j' 는 0이 아닌 랜덤값, j=1, …, l)

[0137] 최상위 비트부터 $s_j\neq k_j$인 첫 번째 비트를 t비트라고 하자. 직관적으로, $y_t\leftarrow 1$이고 $y_j\leftarrow 0$ (j>t), y_j(j<t)는 랜덤값이 된다.

[0138] 150단계에서는 서브 프로토콜 SZP를 이용하여 모든 l개의 암호문 $E(x_j)$에 대해서 $E(\gamma)$를 계산한다.

[0139] $$E(\gamma)\leftarrow SZP(\{E(x_j)\}_{j\in\{0,\ldots,l-1\}})$$

[0140] 직관적으로, 모든 l개의 x_j=0인 경우 $\gamma \leftarrow 1$ 이고 그 외에는 $\gamma \leftarrow 0$ 이 된다.

[0141] 160단계는 제1 서버(DH)가 110단계에서 α=0을 선택한 경우에만 다음과 같이 실행한다.

[0142] $E(y_0) \leftarrow E(y_0+\gamma)$

[0143] 170단계에서 제1 서버(DH)는 모든 l개의 암호문 $E(u_j)$를 다음과 같이 계산한다.

[0144] $E(u_j) \leftarrow E((y_j-1)*r_j''+w_j)$ (r_j'' 는 0이 아닌 랜덤값, j=1, …, l)

[0145] 직관적으로, s<k인 경우에 α=0(resp., α=1)이면 l개의 암호문 $E(u_j)$는 모두 랜덤값(resp., 0을 포함한 랜덤값)의 암호문이고, s>k인 경우에 α=0(resp., α=1)이면 l개의 암호문 $E(u_j)$는 0을 포함한 랜덤값(resp., 모두 랜덤값)의 암호문이다. s=k인 경우에는 α=0(resp., α=1)이면 l개의 암호문 $E(u_j)$는 0을 포함한 랜덤값(resp., 모두 랜덤값)의 암호문이다.

[0146] 180단계에서 제1 서버(DH)는 모든 l개의 암호문 $E(u_j)$의 순서를 임의로 변경(random permutation)한 후에 제2 서버(CSP)에게 전송한다.

[0147] 190단계에서 제2 서버(CSP)는 제1 서버(DH)로부터 전송된 모든 암호문을 복호화한다. CSP는 복호화된 값 중에서 0이 포함된 경우에는 $E(\beta) \leftarrow E(0)$ 으로 설정하고, 그 외에는 $E(\beta) \leftarrow E(1)$ 으로 설정한 후에, $E(\beta)$를 제1 서버(DH)에게 전송한다.

[0148] 200단계에서 제1 서버(DH)는 α에 따라서 다음과 같이 E(M), E(D)를 계산하고 결과로 출력한다.

$$E(M) \leftarrow \begin{cases} E(\beta) & (\alpha = 0) \\ E(1-\beta) & (otherwise) \end{cases}$$

[0149] $E(D) \leftarrow E(1-\gamma)$

[0150] 지금까지 설명한 SCI 프로토콜을 이용하여 SkLE/SkSE 기법을 설명한다. SkLE/SkSE는 프라이버시를 보호하면서 n개의 입력 데이터 중에서 k개의 가장 큰/작은 데이터를 계산하는 기법이다. 이 기법은 PkNC에서 쿼리와 가장 가까운 k개의 데이터를 찾기 위해서 사용된다.

[0151] SkLE/SkSE에서 각 데이터는 보조 데이터 K_i, P_i, $C_i \in \{0,1\}^3$를 갖는다. K_i는 해당 데이터가 k개의 가장 큰/작은 데이터인지를 의미한다. P_i는 매 라운드마다 해당 데이터가 예상된 k개의 가장 큰/작은 데이터인지를 의미한다. k개의 가장 큰/작은 데이터는 후보 데이터 중에서 선택되는데, C_i는 해당 데이터가 k개의 가장 큰/작은 후보 데이터인지를 의미한다.

[0152] SkLE/SkSE는 모든 n개의 데이터의 최상위 비트부터 최하위 비트까지 매 라운드마다 1비트에 대해서 보조 데이터 K_i, P_i, C_i를 계산하면서 k개의 가장 큰/작은 데이터를 계산한다.

[0153] 도 3은 프라이버시를 보호하면서 k개의 가장 큰 데이터를 계산하는 SkLE의 흐름도이다.

[0154] 210단계에서는 입력 데이터의 보조 데이터를 다음과 같이 초기화한다.

[0155] $E(K_i) \leftarrow E(0)$, $E(C_i) \leftarrow E(1)$ (i=1, …, n)

[0156] 보조 데이터의 의미는 어떤 데이터도 k개의 가장 큰 데이터 집합에 포함되지 않고(K_i=0), 모든 데이터가 후보 데이터 집합에 포함된다(C_i=1)는 것을 나타낸다.

[0157] 220단계에서는 k개의 가장 큰 데이터로 예상되는지 확인하기 위해 다음과 같이 계산한다. 여기서 $e_{i,j}$는 i번째 데이터의 j번째 비트값을 의미한다.

[0158] $E(P_i) \leftarrow E(K_i + C_i \cdot e_{i,j})$ (i=1, …, n)

[0159] 230단계에서는 k개의 가장 큰 데이터로 예상되는 데이터의 개수를 다음과 같이 계산한다.

[0160] $$\mathrm{E(s)} \leftarrow E\left(\sum_{i=1}^{n} P_i\right)$$

[0161] 240단계에서는 k개의 가장 큰 데이터로 예상되는 데이터의 개수를 파라미터 k와 비교한다. 이를 위해서 SCI 프로토콜을 이용한다.

[0162] $$\{E(M), E(D)\} \leftarrow SCI(E(s), k)$$

[0163] 즉, s<k인 경우 M←1이고 그 외에는 M←0이 된다. 또한, s≠k인 경우 D←1이고 그 외에는 D←0이 된다.

[0164] 250단계에서는 각 데이터에 대해서 k개의 가장 큰 데이터인지(K_i)와 후보 데이터인지(C_i)를 다음과 같이 계산한다.

[0165] $E(K_i) \leftarrow E(K_i + e_{i,j} \cdot C_i \cdot (1-D+D \cdot M))$ (i=1, …, n)

[0166] $E(C_i) \leftarrow E(C_i \cdot (M+e_{i,j} \cdot D \cdot (1-2M)))$ (i=1, …, n)

[0167] 220단계부터 250단계까지 데이터의 길이 l라운드를 실행한 후에, 260단계에서는 각 데이터가 k개의 가장 큰 데이터인지(K_i)를 다음과 같이 계산한다.

[0168] $E(K_i) \leftarrow E(K_i + C_i)$ (i=1, …, n)

[0169] SkLE가 종료되면, 모든 n개의 $E(K_i)$를 결과로 출력하고, K_i=1에 해당하는 데이터가 가장 큰 k개의 데이터가 된다.

[0170] k개의 가장 작은 데이터를 안전하게 계산하는 SkSE는 SkLE를 이용해서 계산된다. 즉, 앞에서 설명했던 k개의 가장 큰 데이터를 안전하게 계산하는 SkLE는 입력값으로서 데이터의 각 비트의 암호문을 입력한다. 그러나 SkLE에 데이터의 비트의 암호문 대신 보수 비트의 암호문을 입력하면, k개의 가장 작은 데이터를 결과로 출력한다. 즉, k개의 가장 작은 데이터를 안전하게 계산하는 SkSE는 SkLE를 이용해서 다음과 같이 계산한다.

[0171] $$SkSE\left(\{E(e_{i,j})\}_{i\in[n],j\in\{0,\ldots,l-1\}}\right) = SkLE\left(\{E(1-e_{i,j})\}_{i\in[n],j\in\{0,\ldots,l-1\}}\right)$$

[0172] 지금까지 설명한 SkSE를 이용하여 PkNC를 계산할 수 있다.

[0173] 도 4는 PkNC 계산 과정의 흐름도이다.

[0174] 310단계에서는 입력 쿼리와 각 데이터 사이의 거리(유사도)를 계산한다. 거리를 계산하는 방법으로는 유클리드 거리(Euclidean distance), 코사인 거리(cosine distance), 해밍 거리(Hamming distance) 등등 여러 가지 방법이 사용될 수 있다. 입력 쿼리와 비교되는 각 데이터는 데이터 소유자의 단말로부터 미리 수신되어 제1 서버(DH)에 저장되어 있을 수 있다. 또한, 입력 쿼리는 사용자 또는 질의자의 단말로부터 제1 서버(DH)로 전송될 수 있다. 또한, 입력 쿼리와 데이터는 암호화된 상태로 수신될 수 있다. 따라서, 암호화된 상태의 데이터로부터 거리를 산출하기 위해서 소정의 암호 기법, 예컨대 부분 동형 암호 기법 등이 이용될 수 있다.

[0175] 320단계에서는 앞에서 계산된 거리 중에서 k개의 가장 작은 거리를 계산한다. 이를 위해서 앞에서 설명한 SkSE를 이용한다.

[0176] 330단계에서는 앞에서 계산된 k개의 가장 가까운 거리에 해당하는 데이터들이 포함된 각 클래스의 개수를 계산한다. 각 클래스의 개수 중에서 가장 많은 개수의 클래스를 입력 쿼리의 클래스로 결정한다. 결정된 클래스에 대한 정보는 사용자 또는 질의자의 단말로 전송될 수 있다. 또한, 클래스에 대한 정보는 암호화된 상태로 전송될 수 있다.

[0177] 이상에서 설명된 장치는 하드웨어 구성 요소, 소프트웨어 구성 요소, 및/또는 하드웨어 구성 요소 및 소프트웨어 구성 요소의 집합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성 요소는, 예를 들어, 프로세서, 콘트롤러, ALU(Arithmetic Logic Unit), 디지털 신호 프로세서(Digital Signal Processor), 마이크로컴퓨터, FPA(Field Programmable array), PLU(Programmable Logic Unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(Operation System, OS) 및 상기 운영 체제 상에서

수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술 분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(Processing Element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(Parallel Processor)와 같은, 다른 처리 구성(Processing Configuration)도 가능하다.

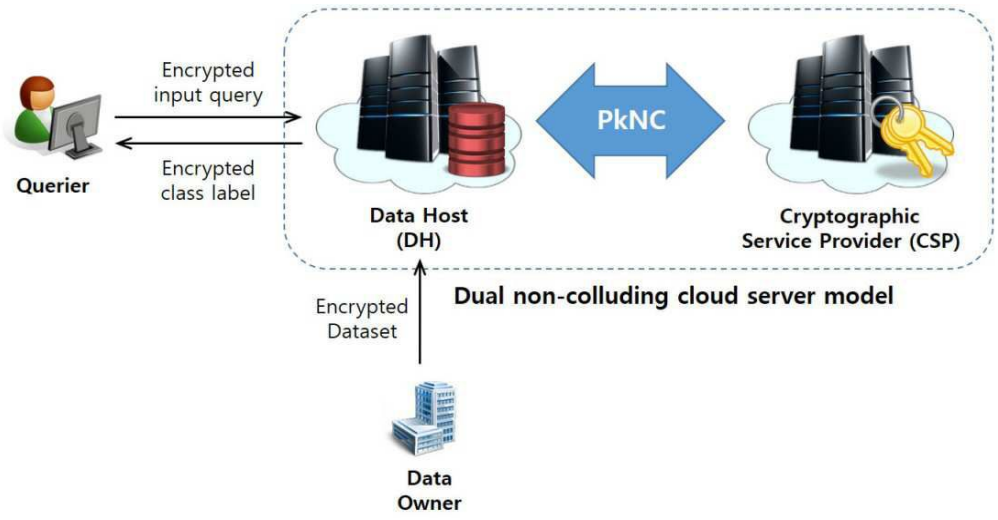
[0178] 소프트웨어는 컴퓨터 프로그램(Computer Program), 코드(Code), 명령(Instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(Collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성 요소(Component), 물리적 장치, 가상 장치(Virtual Equipment), 컴퓨터 저장 매체 또는 장치, 또는 전송되는 신호 파(Signal Wave)에 영구적으로, 또는 일시적으로 구체화(Embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

[0179] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(Magnetic Media), CD-ROM, DVD와 같은 광기록 매체(Optical Media), 플롭티컬 디스크(Floptical Disk)와 같은 자기-광 매체(Magneto-optical Media), 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

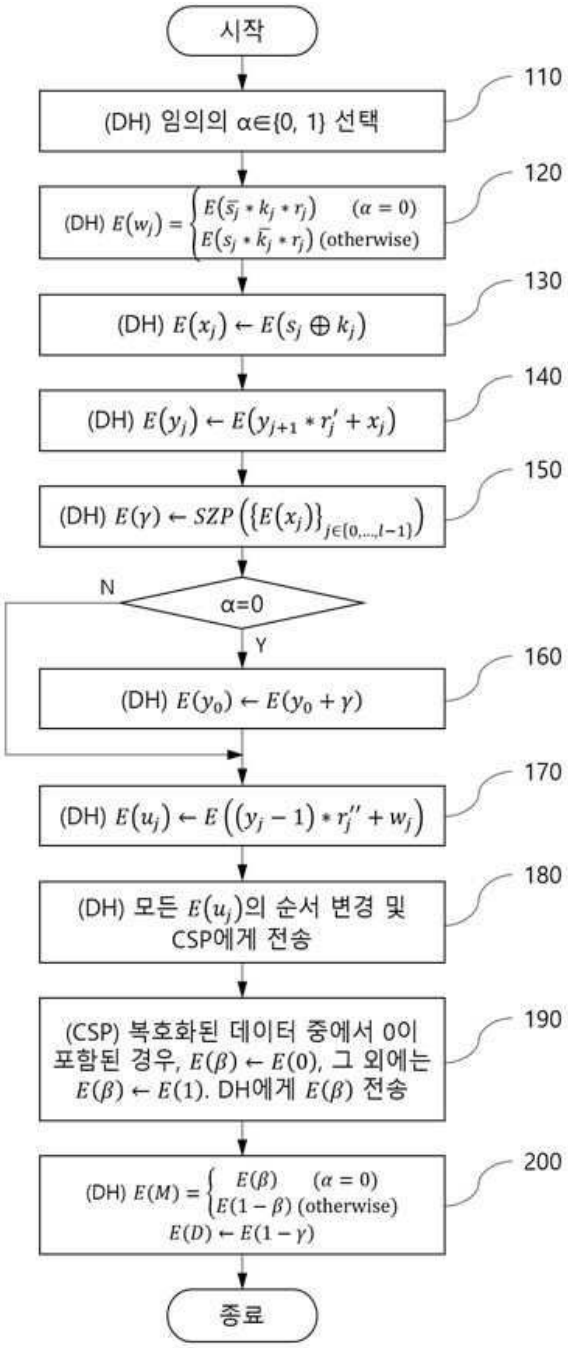
[0180] 본 발명은 도면에 도시된 실시예를 참고로 설명되었으나 이는 예시적인 것에 불과하며, 본 기술 분야의 통상의 지식을 가진 자라면 이로부터 다양한 변형 및 균등한 타 실시예가 가능하다는 점을 이해할 것이다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성 요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다. 따라서, 본 발명의 진정한 기술적 보호 범위는 첨부된 등록청구범위의 기술적 사상에 의해 정해져야 할 것이다.

도면

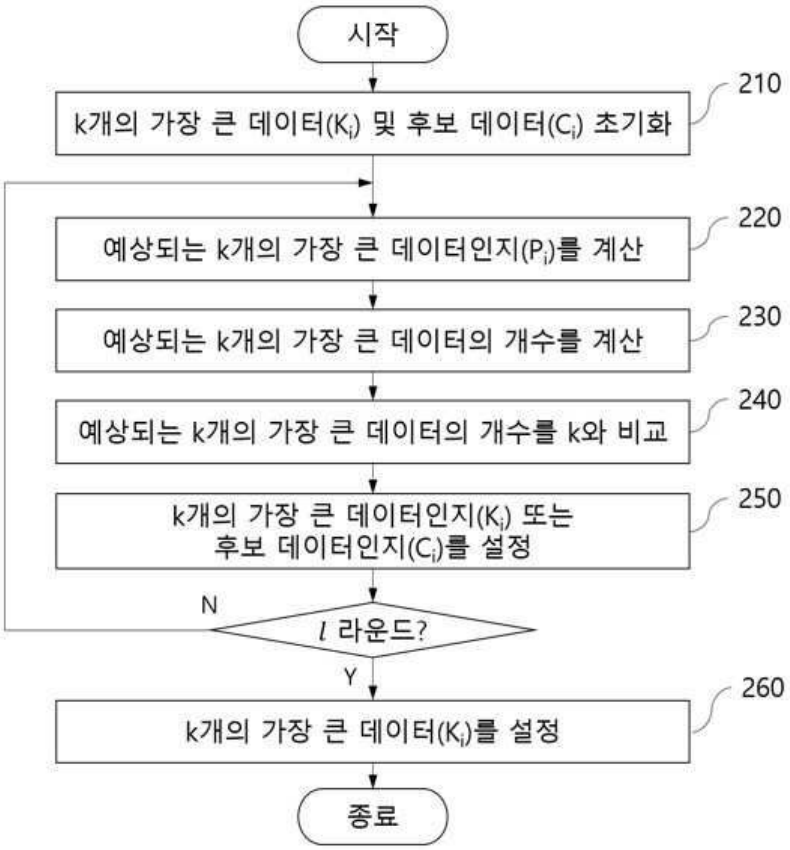
도면1



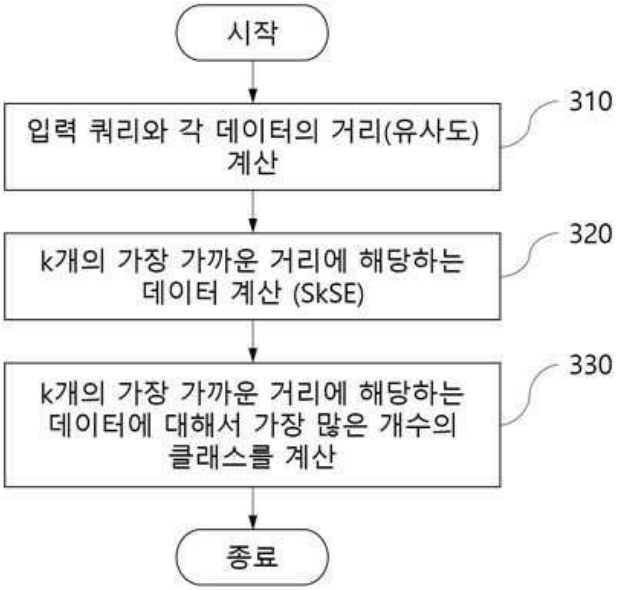
도면2


시작
110
(DH) 임의의 α∈{0, 1} 선택
120
(DH) $E(w_j) = \begin{cases} E(\bar{s}_j * k_j * r_j) & (\alpha = 0) \\ E(s_j * \bar{k}_j * r_j) & (\text{otherwise}) \end{cases}$
130
(DH) $E(x_j) \leftarrow E(s_j \oplus k_j)$
140
(DH) $E(y_j) \leftarrow E(y_{j+1} * r'_j + x_j)$
150
(DH) $E(\gamma) \leftarrow SZP\left(\{E(x_j)\}_{j\in\{0,\ldots,l-1\}}\right)$
N
α=0
Y
160
(DH) $E(y_0) \leftarrow E(y_0 + \gamma)$
170
(DH) $E(u_j) \leftarrow E\left((y_j - 1) * r''_j + w_j\right)$
180
(DH) 모든 $E(u_j)$의 순서 변경 및 CSP에게 전송
190
(CSP) 복호화된 데이터 중에서 0이 포함된 경우, $E(\beta) \leftarrow E(0)$, 그 외에는 $E(\beta) \leftarrow E(1)$. DH에게 $E(\beta)$ 전송
200
(DH) $E(M) = \begin{cases} E(\beta) & (\alpha = 0) \\ E(1-\beta) & (\text{otherwise}) \end{cases}$
$E(D) \leftarrow E(1-\gamma)$
종료

도면3


시작
k개의 가장 큰 데이터(K_i) 및 후보 데이터(C_i) 초기화
210
예상되는 k개의 가장 큰 데이터인지(P_i)를 계산
220
예상되는 k개의 가장 큰 데이터의 개수를 계산
230
예상되는 k개의 가장 큰 데이터의 개수를 k와 비교
240
k개의 가장 큰 데이터인지(K_i) 또는 후보 데이터인지(C_i)를 설정
250
N
l 라운드?
Y
k개의 가장 큰 데이터(K_i)를 설정
260
종료

도면4


시작
입력 쿼리와 각 데이터의 거리(유사도) 계산
310
k개의 가장 가까운 거리에 해당하는 데이터 계산 (SkSE)
320
k개의 가장 가까운 거리에 해당하는 데이터에 대해서 가장 많은 개수의 클래스를 계산
330
종료