Garbled Searchable Symmetric Encryption

Kaoru Kurosawa

Ibaraki University, Japan
kurosawa@mx.ibaraki.ac.jp

Abstract. In a searchable symmetric encryption (SSE) scheme, a client can keyword search over symmetrically-encrypted files which he stored on the server (ideally without leaking any information to the server). In this paper, we show the first multiple keyword search SSE scheme such that even the search formula f (AND, OR and so on) is kept secret. Our scheme is based on an extended garbled circuit satisfying *label-reusable privacy* which is introduced in this paper.

Keywords: searchable symmetric encryption, multiple keyword search, garbled circuit

1 Introduction

1.1 Searchable Symmetric Encryption

Cloud storage service is a major industry trend in the Internet society. In the model of searchable symmetric encryption (SSE) schemes, a client first stores a set of encrypted files $\{C_i\}$ on the server. Later, in the search phase, he can efficiently retrieve the encrypted files which contain some specific keywords without any loss of data confidentiality. While single keyword search SSE schemes have been studies extensively so far [24, 12, 11, 7, 19, 22, 21, 20], there are only a few works that study multiple keyword search SSE schemes.

Conjunctive (AND) keyword search in the SSE setting was first considered by Golle et al. [17]. In their scheme, a client can specify at most one keyword in each keyword field. For example, the keyword fields consist of "To", "From" and "Subject" in emails. This framework was followed up by [4, 5]. In such schemes, however, the client cannot retrieve files which contain both Alice and Bob somewhere in all the keyword fields (for example, somewhere in "To", "From" and "Subject").

Wang et al. [25] showed a keyword field free conjunctive keyword search scheme. However, their scheme does not support any other search formulas (for example, OR).

Recently, Cash et al. [8] showed a keyword field free SSE scheme which can support any search formula in the random oracle model. However, the search formulas are leaked to the server [9, Page 16]. Further , their search phase requires four moves. Namely in the first two moves, the client receives the set of encrypted indexes $\{\mathtt{rind}\}$ of the files he wants to retrieve [9, Figure 3]. He then decrypts them to $\mathtt{DB} = \{\mathtt{ind}\}$. In the next two moves, the client sends $\mathtt{DB}$ to the server, and the server returns all encrypted files C_i such that $i \in \mathtt{DB}$.

1.2 Garbled Circuit

Garbled circuits were initially presented by Yao [26] in the context of secure two-party computation. They were proven secure by Lindell and Pinkas [23]. Recently, the notion has been formalized by Bellare et al. [3].

Over the years, garbled circuits have found many applications: two-party secure protocols [27], multi-party secure protocols [16], one-time programs [15], KDM-security [2], verifiable computation [13], homomorphic computations [14] and others.

A garbled circuit is an encoding $\mathtt{garble}(f)$ of a boolean circuit f such that one can compute $f(X)$ from $(\mathtt{garble}(f), \mathit{label}(X))$ without learning anything about (f, X) other than $f(X)$, where $\mathit{label}(X)$ is an encoding of X. This security notion is called *circuit and input privacy*.

Usually, $(\mathtt{garble}(f), \mathit{label}(X))$ is one-time use. Namely if $\mathtt{garble}(f)$ or $\mathit{label}(X)$ is reused, then some information on (f, X) is leaked. Very recently, Goldwasser et al. [18] constructed a reusable garbled circuit $\mathtt{garble}(f)$, which can be reused for multiple inputs $X_1, X_2, \ldots$.

1.3 Our Contribution

In this paper, we show the first multiple keyword search SSE scheme such that even the search formula f is kept secret. Also, (1) it is keyword field free, (2) it can support any search formula and (3) the search phase requires only two moves.

Table 1. Keyword field free SSE scheme.

	search formula	search phase	search formula secrecy
Wang et al. [25]	only AND	2 moves	no
Cash et al. [8]	any	4 moves	no
Proposed	any	2 moves	yes

Our scheme is based on an extended garbled circuit satisfying *label-reusable privacy* which is introduced in this paper. In such a scheme, one can compute $f_1(X), f_2(X), \ldots$ from $\mathit{label}(X), \mathtt{garble}(f_1), \mathtt{garble}(f_2), \ldots$ without learning anything about $(X, f_1, f_2, \ldots)$ other than $f_1(X), f_2(X), \ldots$.

Table 2. Reusable Garbled Circuit.

Goldwasser et al. [18]	$\mathtt{garble}(f)$ can be reused
This paper	$\mathit{label}(X)$ can be reused

We first formulate this security notion, and then present a simple scheme which satisfies it. We next construct a multiple keyword search SSE scheme by using an extended garbled circuit which satisfies *label-reusable privacy*. (In the first place, no SSE scheme is known which uses a garbled circuit.)

Suppose that a client wants to retrieve all files which contain two keywords w_1 AND w_2. Let $\mathtt{List}(w) = \{i \mid \text{a file } D_i \text{ contains a keyword } w\}$. Then in any multiple keyword search SSE scheme, the server learns at least $\mathtt{List}(w_1) \cap \mathtt{List}(w_2)$ because she must return all encrypted files C_i such that $i \in (\mathtt{List}(w_1) \cap \mathtt{List}(w_2))$. In addition to this, our scheme allows the server to learn only $\pi(1)$ and $\pi(2)$, where π is a random permutation.

On the other hand, in the scheme of Cash et al. [8], the server additionally learns (i) that the search formula is AND, (ii) $\mathtt{List}(w_1)$ or $\mathtt{List}(w_2)$, and some more information (see [9, Sec.5.3] for the details).

The communication overhead of our search phase is $c\lambda + 4ms\lambda$ bits, where λ is the security parameter (say $\lambda = 128$), m is the number of files, c is the input size of a search formula f (namely c is the number of search keywords) and s is the number of gates of f. We also present a more efficient variant for small c such that the communication overhead is $c\lambda + m2^c$ bits, which is $2\lambda + 4m$ bits for $2(= c)$ keyword search.[1] [2]

This paper is organized as follows. Sec. 2 is preliminaries. In Sec. 3, we introduce a notion of label-reusable privacy of garbled circuits. We then present a simple construction which satisfies this security notion. Sec. 4 defines multiple keyword query SSE schemes. In Sec. 5, we show how to construct a multiple keyword query SSE scheme from a label-reusable garbled circuit. Sec. 6 presents an example.

2 Preliminaries

PPT means probabilistic polynomial time. If A is an algorithm, then $y \leftarrow A(x_1, \ldots, x_n; r)$ represents the act of running the algorithm A with inputs $x_1, \ldots, x_n$ and coins r to get an output y, and $y \leftarrow A(x_1, \ldots, x_n)$ represents the act of picking r at random and letting $y \leftarrow A(x_1, \ldots, x_n; r)$.

If X is a set, then $x \stackrel{\$}{\leftarrow} X$ represents the act of choosing x randomly from X. $|X|$ denotes the cardinality of X.

If X is a string, then $|X|$ denotes the bit length of X, and $lsb(X)$ denotes the least significant bit of X.

For $X = (x_1, \ldots, x_n)$ and $U = (i_1, \ldots, i_c)$, we define

$$X|_U = (x_{i_1}, \ldots, x_{i_c}).$$

[1] Our scheme can be combined with an efficient single keyword search SSE scheme such as [22]. Then a single keyword search will be faster.

[2] The scheme of Cash et al. [8] achieves sublinear in m while their search phase requires 4 moves, and some amount of information is leaked to the server.

3 Label Reusable Garbled Circuit

In this section we introduce a notion of *label-reusable privacy* of garbled circuits.

3.1 Garbled Circuit

According to Bellare et al. [3], a boolean circuit is a 5-tuple $f = (n, s, A, B, G)$. Here $n \geq 2$ is the number of inputs, and $s \geq 1$ is the number of gates. We let Inputs $= \{1, ..., n\}$, Gates $= \{n+1, ..., n+s\}$, Wires $= \{1, ..., n+s\}$ and OutputWire $= \{n+s\}$. Then A: Gates $\rightarrow$ Wires\OutputWire is a function to identify each gate's first incoming wire, and B : Gates $\rightarrow$ Wires\OutputWire is a function to identify each gate's second incoming wire. We require $A(g) < B(g) < g$ for each gate $g \in$ Gates. Finally G : Gates $\times\{0,1\}^2 \rightarrow \{0,1\}$ is a function that determines the functionality of each gate. For example, if g is an AND gate, then $G_g(x, y) = x \wedge y$.

Each gate has two inputs and arbitrary functionality. The ith bit of the input is presented along wire i. Every non-input wire is the outgoing wire of some gate. The wires are numbered 1 to $n + s$, and the output wire is $n + s$. The outgoing wire of each gate serves as the name of that gate.

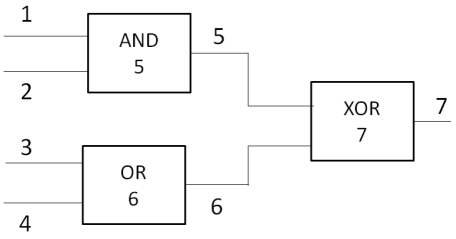

Fig. 1. A boolean circuit f with $n = 4$ and $s = 3$.

We say that $f^- = (n, s, A, B)$ is a topological circuit of $f = (n, s, A, B, G)$. Thus a topological circuit is like a circuit except that the functionality of the gates is unspecified.

We define a garbling scheme by a tuple of PPT algorithms (`GenLab`, `GenGC`, `EvalGC`) as follows:

- `GenLab`$(1^\lambda, n)$ chooses $v_i^0 \in \{0,1\}^\lambda$ and $v_i^1 \in \{0,1\}^\lambda$ such that

$$lsb(v_i^0) \neq lsb(v_i^1)$$

  for $i = 1, \ldots, n$ randomly, and outputs

$$V = ((v_1^0, v_1^1), \ldots, (v_n^0, v_n^1)).$$

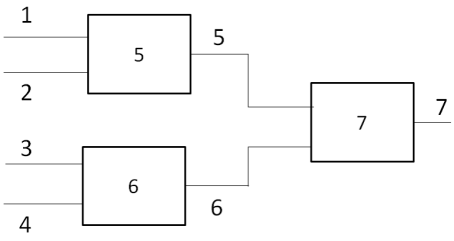

Fig. 2. The topological circuit f^- of Figure 1.

– $\texttt{GenGC}(f, V)$ outputs a garbled circuit Γ, where

$$f = (n, s, A, B, G) \text{ and } V = ((v_1^0, v_1^1), \ldots, (v_n^0, v_n^1)).$$

– $\texttt{EvalGC}(f^-, \Gamma, (v_1^{x_1}, \ldots, v_n^{x_n}))$ is a deterministic algorithm which outputs z such that

$$z = f(x_1, \ldots, x_n),$$

where $x_i \in \{0, 1\}$ for each i.

Correctness requires that if $V \leftarrow \texttt{GenLab}(1^\lambda, n)$ and $\Gamma \leftarrow \texttt{GenGC}(f, V)$, then

$$\texttt{EvalGC}(f^-, \Gamma, (v_1^{x_1}, \ldots, v_n^{x_n})) = f(x_1, \ldots, x_n).$$

for any $X = (x_1, \ldots, x_n)$.

A garbling scheme $(\texttt{GenLab}, \texttt{GenGC}, \texttt{EvalGC})$ is said to satisfy *circuit and input privacy* if $(f^-, \Gamma, (v_1^{x_1}, \ldots, v_n^{x_n}))$ leaks no information on f and $(x_1, \ldots, x_n)$ other than $z = f(x_1, \ldots, x_n)$ and f^-.

3.2 Label Reusable Privacy

We first extend a garbling scheme $(\texttt{GenLab}, \texttt{GenGC}, \texttt{EvalGC})$ to an extended garbling scheme $(\texttt{GenLab}, \texttt{eGenGC}, \texttt{eEvalGC})$. The difference is that $\texttt{eGenGC}$ and $\texttt{eEvalGC}$ take a positive integer $\texttt{counter}$ as an additional input. Namely

$$\Gamma \leftarrow \texttt{eGenGC}(\texttt{counter}, f, V),$$
$$z \leftarrow \texttt{eEvalGC}(\texttt{counter}, f^-, \Gamma, (v_1^{x_1}, \ldots, v_n^{x_n})).$$

The correctness requires that if $V \leftarrow \texttt{GenLab}(1^\lambda, n)$ and $\Gamma \leftarrow \texttt{eGenGC}(\texttt{counter}, f, V)$, then

$$\texttt{eEvalGC}(\texttt{counter}, f^-, \Gamma, (v_1^{x_1}, \ldots, v_n^{x_n})) = f(x_1, \ldots, x_n)$$

for any $X = (x_1, \ldots, x_n)$.

We next define *label-reusable privacy* for extended garbling schemes. Roughly speaking, it means that no information on X and $(f_1, f_2, \ldots)$ is leaked from

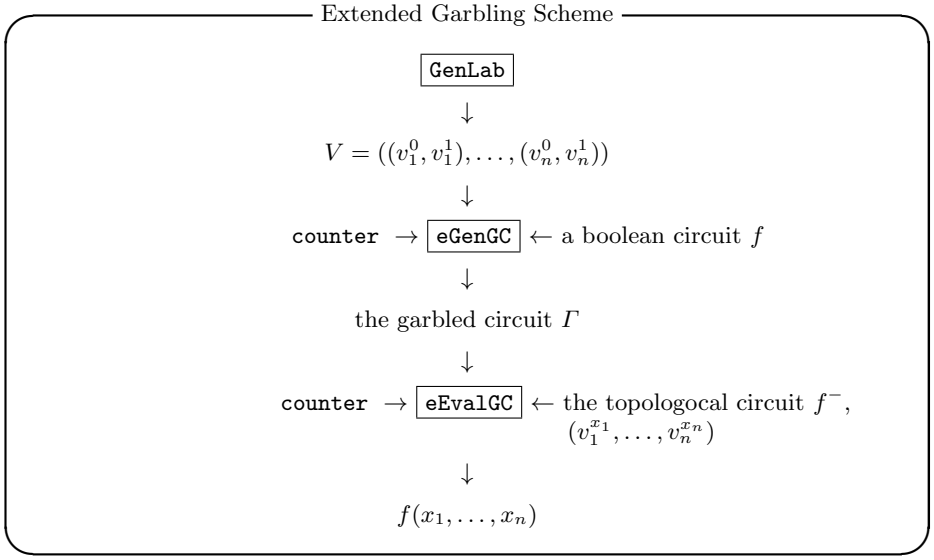

Fig. 3. Extended Garbling Scheme

$(v_1^{x_1}, \ldots, v_n^{x_n})$ and $(\Gamma_1, \Gamma_2, \ldots)$, where a fixed input X is reused for multiple boolean circuits $f_1, f_2, \ldots$.

To formally define this security notion, we consider a real game $\texttt{Garble}_{real}$ and a simulation game $\texttt{Garble}_{sim}$ as shown in Figure 4 and Figure 5. In both games, the adversary $\mathbf{A}$ chooses

- $X = (x_1, \ldots, x_n)$ in the setup phase, and
- (U_i, f_i) in the query phase for $i = 1, \ldots, q$, where $U_i \subseteq \{1, \ldots, n\}$ and $f_i = (|U_i|, s_i, A_i, B_i, G_i)$ is a boolean circuit,

and sends them to the challenger. In $\texttt{Garble}_{real}$, the challenger returns $(v_1^{x_1}, \ldots, v_n^{x_n})$ in the setup phase, and a garbled circuit Γ_i in the query phase. In $\texttt{Garble}_{sim}$, the simulator must return

- fake $(v_1^{x_1}, \ldots, v_n^{x_n})$ based solely on n in the setup phase, and
- fake Γ_i based solely on $|U_i|$, $z_i = f_i(X|_{U_i})$ and f_i^- in the query phase.

Our requirement is that $(v_1^{x_1}, \ldots, v_n^{x_n})$ and $\{\Gamma_i\}$ should not leak any information other than n, $\{z_i = f_i(X|_{U_i})\}$ and $\{(|U_i|, f_i^-)\}$[3]. Let

$$\texttt{Adv}^{\text{garble}}_{\text{real}}(A) = \Pr(\mathbf{A} \text{ outputs } b = 1 \text{ in } \texttt{Garble}_{real}),$$
$$\texttt{Adv}^{\text{garble}}_{\text{sim}}(A) = \Pr(\mathbf{A} \text{ outputs } b = 1 \text{ in } \texttt{Garble}_{sim}).$$

Definition 1. *We say that an extended garbling scheme* (GenLab, eGenGC, eEvalGC) *satisfies label-reusable privacy if there exists a PPT simulator*

[3] $\{(|U_i|, f_i^-)\}$ corresponds to the side-information function Φ_{topo} of [3].

Real Game for a Garbling Scheme ($\texttt{Garble}_{real}$)

- **Setup Phase:**
  1. An adversary **A** chooses $X = (x_1, \ldots, x_n)$ and sends it to the challenger.
  2. The challenger runs $\texttt{GenLab}(1^\lambda, n)$ to generate $V = ((v_1^0, v_1^1), \ldots, (v_n^0, v_n^1))$. He then returns $(v_1^{x_1}, \ldots, v_n^{x_n})$ to **A**.
- **Query Phase** ($i = 1, \ldots, q$)**:**
  1. **A** chooses $U_i \subseteq \{1, \ldots, n\}$ and $f_i = (|U_i|, s_i, A_i, B_i, G_i)$, and sends them to the challenger.
  2. The challenger returns a garbled circuit $\Gamma_i \leftarrow \texttt{eGenGC}(i, f_i, V|_{U_i})$ to **A**.
- Finally **A** outputs a bit b.

Fig. 4. Real Game for a Garbling Scheme: $\texttt{Garble}_{real}$.

Simulation Game for a Garbling Scheme ($\texttt{Garble}_{sim}$)

- In the setup phase,
  1. An adversary **A** chooses $X = (x_1, \ldots, x_n)$ and sends it to the challenger.
  2. The challenger sends n to a simulator **Sim**.
  3. **Sim** returns $(v_1, \ldots, v_n)$ to the challenger who relays it to **A**.
- In the query phase, for $i = 1, \ldots, q$,
  1. **A** chooses $U_i \subseteq \{1, \ldots, n\}$ and $f_i = (|U_i|, s_i, A_i, B_i, G_i)$, and sends them to the challenger.
  2. The challenger computes $z_i = f_i(X|_{U_i})$, and sends (i, U_i, f_i^-, z_i) to **Sim**.
  3. **Sim** returns Γ_i to the challenger who relays it to **A**.
- Finally **A** outputs a bit b.

Fig. 5. Simulation Game for a Garbling Scheme: $\texttt{Garble}_{sim}$.

Sim *such that* $|\texttt{Adv}^{\text{garble}}_{\text{real}}(A) - \texttt{Adv}^{\text{garble}}_{\text{sim}}(A)|$ *is negligible for any PPT adversary* **A**.

3.3 Construction

We present a simple construction of an extended garbling scheme which satisfies label-reusable privacy. Let $H_0 : \{0,1\}^* \to \{0,1\}^\lambda$ and $H_1 : \{0,1\}^* \to \{0,1\}$ be two hash functions. They will be treated as random oracles in the security proofs.

On input $\texttt{counter}$, $f = (n, s, A, B, G)$ and $V = ((v_1^0, v_1^1), \ldots, (v_n^0, v_n^1))$, $\texttt{eGenGC}$ behaves as follows.

1. For $i \in \{n+1, \ldots, n+s-1\}$, choose $\bar{v}_i^0$ and $\bar{v}_i^1$ from $\{0,1\}^\lambda$ such that $lsb(\bar{v}_i^0) \neq lsb(\bar{v}_i^1)$ randomly.
2. Define
$$L_i^x = \begin{cases} v_i^x \ if \ 1 \leq i \leq n \\ \bar{v}_i^x \ if \ n < i \leq n+s-1 \end{cases}$$

for $1 \leq i \leq n+s-1$ and $x \in \{0,1\}$.

3. For $(g,x,y) \in \{n+1,\ldots,n+s\} \times \{0,1\} \times \{0,1\}$, do

$$a \leftarrow A(g), b \leftarrow B(g), \ell_a \leftarrow lsb(L_a^x), \ell_b \leftarrow lsb(L_b^y),$$

$$P[g,\ell_a,\ell_b] = \begin{cases} H_0(\texttt{counter}, g, L_a^x, L_b^y) \oplus L_g^{G_g(x,y)} \; if \; g \neq n+s \\ H_1(\texttt{counter}, g, L_a^x, L_b^y) \oplus G_g(x,y) \; if \; g = n+s \end{cases}$$

4. Output a garbled circuit

$$\Gamma = [P(n+1,\cdot,\cdot),\ldots,P(n+s,\cdot,\cdot)]. \tag{1}$$

On input `counter`, f^-, $\Gamma = [P(n+1,\cdot,\cdot),\ldots,P(n+s,\cdot,\cdot)]$ and $(v_1,\ldots,v_n)$, `eEvalGC` behaves as follows. Let $L_i = v_i$ for $i = 1,\ldots,n$.

1. For $g = n+1,\ldots,n+s$, do

$$a \leftarrow A(g), b \leftarrow B(g), \ell_a \leftarrow lsb(L_a), \ell_b \leftarrow lsb(L_b),$$

$$v_g = P[g,\ell_a,\ell_b] \oplus H_0(\texttt{counter}, g, L_a, L_b) \text{ if } g \neq n+s$$

$$z = P[g,\ell_a,\ell_b] \oplus H_1(\texttt{counter}, g, L_a, L_b) \text{ if } g = n+s$$

2. Output z.

Namely our extended garbling scheme is almost the same as the usual garbling scheme except for that the additional input `counter` is included in the inputs to H_0 and H_1, and each value of $\bar{v}_i^x$ is chosen freshly for each value of `counter`.

Also $P(n+s,\cdot,\cdot)$ encrypts a bit $G_{n+s}(x,y)$ instead of a string $L_{n+s}^{G_{n+s}(x,y)}$ (by one-time pad) because this is enough for our application to searchable symmetric encryption. (We must encrypt $L_{n+s}^{G_{n+s}(x,y)}$ by one-time pad in secure two-party computation, though.)

(Example 1) Let $n = 2$. In `GenLab`, we choose

$$v_1^0, v_1^1, v_2^0, v_2^1 \xleftarrow{\$} \{0,1\}^\lambda.$$

For simplicity, assume that

$$lsb(v_1^0) = lsb(v_2^0) = 0, \; lsb(v_1^1) = lsb(v_2^1) = 1.$$

In `eGenGC`, for a boolean circuit $f(\cdot,\cdot)$, the garbled circuit Γ is constructed as

$$\Gamma = [P(3,0,0),\ldots,P(3,1,1)],$$

where

$$P(3,0,0) = H_1(\texttt{counter}, 3, v_1^0, v_2^0) \oplus f(0,0) \tag{2}$$

$$P(3,0,1) = H_1(\texttt{counter}, 3, v_1^0, v_2^1) \oplus f(0,1) \tag{3}$$

$$P(3,1,0) = H_1(\texttt{counter}, 3, v_1^1, v_2^0) \oplus f(1,0) \tag{4}$$

$$P(3,1,1) = H_1(\texttt{counter}, 3, v_1^1, v_2^1) \oplus f(1,1) \tag{5}$$

In $\texttt{eEvalGC}$, we are given Γ, $\texttt{counter}$ and some (v_1, v_2). Suppose that

$$(v_1, v_2) = (v_1^0, v_2^0).$$

Then we first compute

$$(lsb(v_1^0), lsb(v_2^0)) = (0, 0)$$

We next compute

$$f(0,0) = P(3,0,0) \oplus H_1(\texttt{counter}, 3, v_1^0, v_2^0).$$

Theorem 1. *The above extended garbling scheme* ($\texttt{GenLab}, \texttt{eGenGC}$, $\texttt{eEvalGC}$) *satisfies label-reusable privacy in the random oracle model.*

3.4 Proof

We construct a simulator **Sim** as follows. In the setup phase, **Sim** is given n. Then, **Sim** runs $\texttt{GenLab}(1^\lambda, n)$ to generate $V = ((v_1^0, v_1^1), \ldots, (v_n^0, v_n^1))$. It then returns $(v_1^0, \ldots, v_n^0)$ to the challenger.

In the ith query phase, **Sim** is given (i, U_i, f_i^-, z_i). Let $f_i^- = (c_i, s_i, A_i, B_i)$.

1. **Sim** chooses G such that $f_i' = (c_i, s_i, A_i, B_i, G)$ is a boolean circuit and $z_i = f_i'(0, \ldots, 0)$ arbitrarily.
2. **Sim** computes $\Gamma_i \leftarrow \texttt{eGenGC}(i, f_i', V|_{U_i})$ and returns Γ_i.

For $i = 1, \ldots, q$, we say that (i, g, L_a, L_b) is visible if we must query (i, g, L_a, L_b) to the H_0-oracle or to the H_1-oracle when computing

$$z_i \leftarrow \texttt{EvalGC}(i, f_i^-, \Gamma_i, (v_1^{x_1}, \ldots, v_n^{x_n})).$$

Otherwise we say that (i, g, L_a, L_b) is invisible.

Then, consider a game $\texttt{Garble}_1$ which is the same as $\texttt{Garble}_{real}$ except for that each $H_0(i, g, L_a, L_b)$, such that (i, g, L_a, L_b) is invisible, is replaced by a random string, and each $H_1(i, g, L_a, L_b)$, such that (i, g, L_a, L_b) is invisible, is replaced by a random bit. Define

$$p_1 = \Pr(\mathbf{A} \text{ outputs } b = 1 \text{ in } \texttt{Garble}_1).$$

Lemma 1. $|\texttt{Adv}_{\text{real}}^{\text{garble}}(A) - p_1|$ *is negligible.*

Proof. Let **BAD** be the event that an adversary $\mathbf{A}$ queries some invisible (i, g, L_a, L_b) to the H_0-oracle or to the H_1-oracle. Until **BAD** occurs, $\texttt{Garble}_{real}$ and $\texttt{Garble}_1$ are the same because H_0 and H_1 are random oracles. Therefore

$$|\texttt{Adv}_{\text{real}}^{\text{garble}}(A) - p_1| \leq \Pr(\mathbf{BAD}).$$

Next $\Pr(\mathbf{BAD})$ is negligible because $\mathbf{A}$ has no information on $v_i^{1-x_i}$ for $i = 1, \ldots, n$. Therefore, we can see that Hence $|\texttt{Adv}_{\text{real}}^{\text{garble}}(A) - p_1|$ is negligible. □

Similarly, let $\texttt{Garble}_2$ be a game which is the same as $\texttt{Garble}_{sim}$ except for that each $H_0(i, g, L_a, L_b)$, such that (i, g, L_a, L_b) is invisible, is replaced by a random string, and each $H_1(i, g, L_a, L_b)$, such that (i, g, L_a, L_b) is invisible, is replaced by a random bit. Let

$$p_2 = \Pr(\mathbf{A} \text{ outputs } b = 1 \text{ in } \texttt{Garble}_2).$$

Then, $|\texttt{Adv}^{\text{garble}}_{\text{sim}}(A) - p_2|$ is negligible similarly to Lemma 1. Finally, it is easy to see that $\texttt{Garble}_1$ and $\texttt{Garble}_2$ are identical. Therefore $p_1 = p_2$. Consequently $|\texttt{Adv}^{\text{garble}}_{\text{real}}(A) - \texttt{Adv}^{\text{garble}}_{\text{sim}}(A)|$ is negligible.

4 Multiple Keyword Query SSE

Let $\mathcal{D} = \{D_1, \ldots, D_m\}$ be a set of documents and $\mathcal{W} = \{w_1, \ldots, w_n\}$ be a set of keywords. Let $\texttt{Index} = \{e_{i,j}\}$ be an $m \times n$ binary matrix such that

$$e_{i,j} = \begin{cases} 1 \text{ if } D_i \text{ contains } w_j \\ 0 \textit{ otherwise} \end{cases}. \tag{6}$$

For a list of keywords $\bar{w} = (w_{j_1}, \ldots, w_{j_c})$ and a boolean circuit $f = (c, s, A, B, G)$, we write $\mathbf{IB}(f, \bar{w})$ for the set of identities of documents that satisfy f. Namely this means that $i \in \mathbf{IB}(f, \bar{w})$ if and only if

$$f(e_{i,j_1}, \ldots, e_{i,j_c}) = 1.$$

For example, suppose that $\bar{w} = (w_1, w_2)$ and $f_1(x_1, x_2) = x_1 \wedge x_2$. Then $i \in \mathbf{IB}(f_1, \bar{w})$ if and only if D_i contains w_1 AND w_2.

4.1 Model

A multiple keyword search SSE scheme is a protocol between a client and a server as follows.

(Store Phase) On input $(\mathcal{D}, \mathcal{W}, \texttt{Index})$, the client sends $(\mathcal{C}, \mathcal{I})$ to the server, where $\mathcal{C} = (C_1, \ldots, C_m)$ is the set of encrypted documents, and $\mathcal{I}$ is an encrypted $\texttt{Index}$.

(Search Phase)

1. The client chooses a list of keywords $\bar{w} = (w_{j_1}, \ldots, w_{j_c})$ and a boolean circuit $f = (c, s, A, B, G)$. He then sends a trapdoor information $t(f, \bar{w})$ to the server.
2. The server somehow computes $\mathbf{IB}(f, \bar{w})$ and returns $\mathbf{CB}(f, \bar{w}) = \{C_j \mid j \in \mathbf{IB}(f, \bar{w})\}$ to the client.
3. The client decrypts each $C_i \in \mathbf{CB}(f, \bar{w})$ and outputs $\mathbf{DB}(f, \bar{w}) = \{D_j \mid j \in \mathbf{IB}(f, \bar{w})\}$.

Real Game for Multiple Keywords ($\mathtt{SSE}_{real}$)

- **Store Phase:**
  1. An adversary **A** chooses $(\mathcal{D}, \mathcal{W}, \mathtt{Index})$ and sends them to the challenger.
  2. The challenger returns $(\mathcal{I}, \mathcal{C})$.
- **Search Phase ($i = 1, \ldots, q$):**
  1. **A** chooses $\bar{w} = (w_{j_1}, \ldots, w_{j_c})$ and f, and sends $(f, \bar{w})$ to the challenger.
  2. The challenger returns $t(f, \bar{w})$ to **A**.
- Finally **A** outputs a bit b.

Fig. 6. Real Game for Multiple Keywords: $\mathtt{SSE}_{real}$.

4.2 Security

We consider a real game $\mathtt{SSE}_{real}$ and a simulation game $\mathtt{SSE}_{sim}$ as shown in Figure 6 and Figure 7. In both games, the adversary **A** chooses

- $(\mathcal{D}, \mathcal{W}, \mathtt{Index})$ in the setup phase, and
- $\bar{w} = (w_{j_1}, \ldots, w_{j_c})$ and f in the query phase for $i = 1, \ldots, q$,

and sends them to the challenger. In $\mathtt{SSE}_{real}$, the challenger returns $(\mathcal{I}, \mathcal{C})$ in the setup phase, and $t(f, \bar{w})$ in the query phase to **A**. In $\mathtt{SSE}_{sim}$, on the other hand, the simulator must return

- fake $(\mathcal{I}, \mathtt{C})$ based solely on $|D_1|, \ldots, |D_m|$ and $n = |\mathcal{W}|$ in the setup phase,
- and fake $t(f, \bar{w})$ based solely on $\mathbf{IB}(f, \bar{w})$, f^- and $U = (\sigma(j_1), \ldots, \sigma(j_c))$ in the query phase, where σ is a random permutation chosen by the challenger at the beginning of the query phase.

In any multiple keyword search SSE scheme, the server learns $|D_1|, \ldots, |D_n|$ and $|\mathcal{W}|$ in the store phase, and $\mathbf{IB}(f, \bar{w})$[4] and f^- in the query phase. In addition to these, our definition will allow the server to learn only $U = (\sigma(j_1), \ldots, \sigma(j_c))$. Let

$$\mathtt{Adv}^{\mathrm{sse}}_{\mathrm{real}}(A) = \Pr(\mathbf{A} \text{ outputs } b = 1 \text{ in } \mathtt{SSE}_{real}),$$
$$\mathtt{Adv}^{\mathrm{sse}}_{\mathrm{sim}}(A) = \Pr(\mathbf{A} \text{ outputs } b = 1 \text{ in } \mathtt{SSE}_{sim}).$$

Definition 2. *We say that a multiple keyword search SSE scheme is secure if there exists a PPT simulator* **Sim** *such that*

$$|\mathtt{Adv}^{\mathrm{sse}}_{\mathrm{real}}(A) - \mathtt{Adv}^{\mathrm{sse}}_{\mathrm{sim}}(A)|$$

is negligible for any PPT adversary **A**.

[4] This is because the server must be able to return $\mathbf{CB}(f, \bar{w}) = \{C_j \mid j \in \mathbf{IB}(f, \bar{w})\}$.

Simulation Game for Multiple Keywords ($\mathtt{SSE}_{sim}$)

- In the store phase,
  1. **A** chooses $(\mathcal{D}, \mathcal{W}, \mathtt{Index})$ and sends them to the challenger.
  2. The challenger sends $|D_1|, \ldots, |D_m|$ and $n = |\mathcal{W}|$ to a simulator **Sim**.
  3. **Sim** returns $(\mathcal{I}, \mathtt{C})$ to the challenger who relays them to **A**.
- In the search phase, the challenger first chooses a random permutation σ on $\{1, \ldots, n\}$. Then for $i = 1, \ldots, q$,
  1. **A** chooses $\bar{w} = (w_{j_1}, \ldots, w_{j_c})$ and f, and sends $(f, \bar{w})$ to the challenger.
  2. The challenger sends $\mathbf{IB}(f, \bar{w})$, $U = (\sigma(j_1), \ldots, \sigma(j_c))$ and f^- to **Sim**.
  3. **Sim** returns $t(f, \bar{w})$ to the challenger who relays it to **A**.
- Finally **A** outputs a bit b.

Fig. 7. Simulation Game for Multiple Keywords: $\mathtt{SSE}_{sim}$.

5 How to Construct Multiple Keyword SSE

In this section we construct a multiple keyword search SSE scheme by using an extended garbling scheme which satisfies label usable privacy. Define $\mathcal{D}, \mathcal{W}, \mathtt{Index} = (e_{ij})$ and $\mathbf{IB}(f, \bar{w})$ as shown in Sec. 4.

5.1 Construction

Let $(\mathtt{GenLab}, \mathtt{eGenGC}, \mathtt{eEvalGC})$ be an extended garbling scheme. Let $\mathtt{SKE} = (\mathtt{Gen}, E, D)$ be a CPA-secure symmetric-key encryption scheme [1], where $\mathtt{Gen}$ is a key generation algorithm, E is an encryption algorithm and D is a decryption algorithm. Let $\mathtt{PRF} : \{0,1\}^\ell \times \{0,1\}^* \to \{0,1\}^\lambda$ be a pseudorandom function, where ℓ is the size of keys.

(Store Phase)

1. The client generates (k_e, k_0) randomly, where k_e is a key of $\mathtt{SKE}$, and k_0 is a key of the $\mathtt{PRF}$. He also chooses a random permutation π on $\{1, \ldots, n\}$.
2. He computes $C_i = E_{k_e}(D_i)$ for $i = 1, \ldots, m$, and
$$\mathtt{kw}_j = \mathtt{PRF}_{k_0}(w_j) \tag{7}$$
for $j = 1, \ldots, n$.
3. For $i = 1, \ldots, m$ and $j = 1, \ldots, n$, do:
   (a) Compute
   $$v^0_{i,j} = \mathtt{PRF}_{k_0}(i, w_j, 0),\ v^1_{i,j} = \mathtt{PRF}_{k_0}(i, w_j, 1). \tag{8}$$
   (b) If $lsb(v^0_{i,j}) = lsb(v^1_{i,j})$, then let
   $$v^1_{i,j} \leftarrow v^1_{i,j} \oplus (0, \ldots, 0, 1). \tag{9}$$

(c) Let

$$v_{i,j} = v_{i,j}^{e_{i,j}}, \tag{10}$$

where $e_{i,j}$ is defined by Eq.(6).

4. For $i = 1, \ldots, m$, let

$$Y_i = (v_{i,\pi(1)}, \ldots, v_{i,\pi(n)}). \tag{11}$$

5. He then stores $\mathcal{C} = (C_1, \ldots, C_m)$ and $\mathcal{I} = (\mathtt{kw}_{\pi(1)}, \ldots, \mathtt{kw}_{\pi(n)}, Y_1, \ldots, Y_m)$ to the server. (See Table 3.)
6. Let $\mathtt{counter} \leftarrow 0$. He holds $(\mathtt{counter}, m)$, and keeps (k_e, k_0) secret.

(Example 2) Consider `Index` such that

$$\mathtt{Index} = \begin{pmatrix} e_{1,1}, e_{1,2}, e_{1,3} \\ e_{1,1}, e_{1,2}, e_{1,3} \end{pmatrix} = \begin{pmatrix} 1, 1, 0 \\ 1, 0, 1 \end{pmatrix}, \tag{12}$$

where $m = 2$ and $n = 3$. Suppose that $\pi(i) = i$ for $i = 1, 2, 3$. Then the client stores the following table to the server, where $v_{i,j}^0$ and $v_{i,j}^1$ are computed according to Eq.(8) and Eq.(9). After this, he holds $(\mathtt{counter} = 0, m = 2)$, and keeps (k_e, k_0) secret.

	$\mathtt{kw}_1$	$\mathtt{kw}_2$	$\mathtt{kw}_3$
C_1	$v_{1,1}^1$	$v_{1,2}^1$	$v_{1,3}^0$
C_2	$v_{2,1}^1$	$v_{2,2}^0$	$v_{2,3}^1$

Table 3. Example of the store phase.

(Search Phase) The client chooses $\bar{w} = (w_{j_1}, \ldots, w_{j_c})$ and $f = (c, s, A, B, G)$. Then he does the following.

1. Let $\mathtt{counter} \leftarrow \mathtt{counter} + 1$.
2. Compute $\mathtt{kw}_{j_1} = \mathtt{PRF}_{k_0}(w_{j_1}), \ldots, \mathtt{kw}_{j_c} = \mathtt{PRF}_{k_0}(w_{j_c})$.
3. For $i = 1, \ldots, m$, do:
   (a) Compute $(v_{i,j_1}^0, v_{i,j_1}^1), \ldots, (v_{i,j_c}^0, v_{i,j_c}^1)$ as in the store phase.
   (b) Let $V_i = ((v_{i,j_1}^0, v_{i,j_1}^1), \ldots, (v_{i,j_c}^0, v_{i,j_c}^1))$
4. For $i = 1, \ldots, m$, compute $\Gamma_i \leftarrow \mathtt{eGenGC}(\mathtt{counter}, f, V_i)$.
5. Send

$$t(f, \bar{w}) = [\mathtt{counter}, f^-, (\mathtt{kw}_{j_1}, \ldots, \mathtt{kw}_{j_c}), (\Gamma_1, \ldots, \Gamma_m)]$$

to the server.

The server does the following.

1. For $i = 1, \ldots, m$, do
   Find $(v_{i,j_1}, \ldots, v_{i,j_c})$ from Y_i by using $(\mathtt{kw}_{j_1}, \ldots, \mathtt{kw}_{j_c})$.
   Compute $z_i \leftarrow \mathtt{eEvalGC}(\mathtt{counter}, f^-, \Gamma_i, (v_{j_1}, \ldots, v_{j_c}))$.
2. Return all C_i such that $z_i = 1$.

5.2 Security

Theorem 2. The above multiple keyword search SSE scheme is secure if the extended garbling scheme ($\mathtt{GenLab}, \mathtt{eGenGC}, \mathtt{eEvalGC}$) satisfies label-reusable privacy and $\mathtt{SKE} = (\mathtt{Gen}, E, D)$ is CPA-secure.

Proof. Since ($\mathtt{GenLab}, \mathtt{eGenGC}, \mathtt{eEvalGC}$) satisfies label-reusable privacy, there exists a simulator $\mathbf{Sim}_g$ which satisfies Def. 1. We will construct a simulator $\mathbf{Sim}_{sse}$ which satisfies Def. 2 by using $\mathbf{Sim}_g$ as a subroutine (see Figure 8).

Let $\mathbf{A}$ be an adversary against our multiple keyword search SSE scheme. Let $\mathtt{SSE}_1$ be a game which is the same as $\mathtt{SSE}_{real}$ except for the fact that all the outputs of $\mathtt{PRF}$ are replaced by random strings. Define

$$p_1 = \Pr(\mathbf{A} \text{ outputs } b = 1 \text{ in } \mathtt{SSE}_1).$$

Then, $|\mathtt{Adv}^{\mathrm{sse}}_{\mathrm{real}}(A) - p_1|$ is negligible because $\mathtt{PRF}$ is a pseudorandom function.

Let $\mathbf{Sim}_g^1, \ldots, \mathbf{Sim}_g^m$ be m copies of $\mathbf{Sim}_g$ such that each $\mathbf{Sim}_g^i$ has independent random coins. Then, our simulator $\mathbf{Sim}_{sse}$ behaves as follows.

(Store Phase) $\mathbf{Sim}_{sse}$ receives $|D_1|, \ldots, |D_m|$ and $n = |\mathcal{W}|$ from the challenger.

1. $\mathbf{Sim}_{sse}$ chooses k_e randomly, where k_e is a key of $\mathtt{SKE}$. Then, it computes $C_i = E_{k_e}(0^{|D_i|})$ for $i = 1, \ldots, m$. Also let $\mathtt{kw}_i \xleftarrow{\$} \{0,1\}^\lambda$ for $i = 1, \ldots, n$.
2. For $i = 1, \ldots, m$, $\mathbf{Sim}_{sse}$ sends n to $\mathbf{Sim}_g^i$, and receives $Y_i = (v_{i,1}, \ldots, v_{i,n})$ from $\mathbf{Sim}_g^i$.
3. $\mathbf{Sim}_{sse}$ returns $\mathcal{C} = (C_1, \ldots, C_m)$ and $\mathcal{I} = (\mathtt{kw}_1, \ldots, \mathtt{kw}_n, Y_1, \ldots, Y_m)$.

(Search Phase) For $ctr = 1, \ldots, q$, $\mathbf{Sim}_{sse}$ receives $\mathbf{IB}(f, \bar{w})$, $U = (\sigma(j_1), \ldots, \sigma(j_c))$ and f^- from the challenger.

1. For $i = 1, \ldots, m$, let
$$z_i = \begin{cases} 1 \; if \; i \in \mathbf{IB}(f, \bar{w}) \\ 0 \; if \; i \notin \mathbf{IB}(f, \bar{w}). \end{cases}$$
2. For $i = 1, \ldots, m$, $\mathbf{Sim}_{sse}$ sends (ctr, U, f^-, z_i) to $\mathbf{Sim}_g^i$, and receives Γ_i from $\mathbf{Sim}_g^i$.
3. $\mathbf{Sim}_{sse}$ returns
$$t(f, \bar{w}) = [ctr, f^-, (\mathtt{kw}_{\sigma(j_1)}, \ldots, \mathtt{kw}_{\sigma(j_c)}), (\Gamma_1, \ldots, \Gamma_n)].$$

Then, we can show that $|\mathtt{Adv}^{\mathrm{sse}}_{\mathrm{sim}}(A) - p_1|$ is negligible by using a hybrid argument because $\mathbf{Sim}_g$ is a simulator of ($\mathtt{GenLab}, \mathtt{eGenGC}, \mathtt{eEvalGC}$). Otherwise, we can construct an adversary B against ($\mathtt{GenLab}, \mathtt{eGenGC}$, $\mathtt{eEvalGC}$) by using $\mathbf{A}$ and $\mathbf{Sim}_{sse}$ as subroutines.

Consequently, $|\mathtt{Adv}^{\mathrm{sse}}_{\mathrm{real}}(A) - \mathtt{Adv}^{\mathrm{sse}}_{\mathrm{sim}}(A)|$ is negligible.

□

Corollary 1. *There exists a secure multiple keyword search SSE scheme in the random oracle model if there exists a pseudorandom function and a CPA-secure symmetric-key encryption scheme.*

Proof. The proof follows from Theorem 1 and 2. □

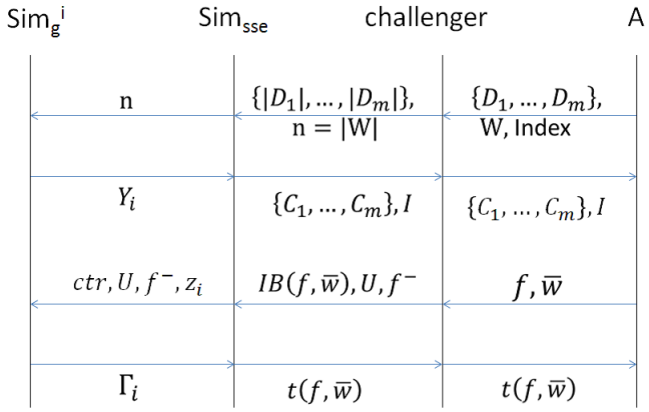

Fig. 8. Proof of Theorem 2.

5.3 Efficiency

Suppose that we use our extended garbling scheme given in Sec. 3.3. Then, from Eq.(7), (8), (10) and (11), we have

$$|\mathtt{kw}_i| = \lambda \text{ and } |Y_i| = n\lambda$$

for all i. Also from Eq.(1), we have

$$|\Gamma_i| = 4(s-1)\lambda + 4$$

for all i.

Therefore, in the store phase, the communication overhead is

$$|\mathcal{I}| = \sum_{i=1}^{m} |\mathtt{kw}_i| + \sum_{i=1}^{m} |Y_i| = m(n+1)\lambda.$$

In the search phase, suppose that the client chooses a list of keywords $\bar{w} = (w_{j_1}, \ldots, w_{j_c})$ and a boolean circuit $f = (c, s, A, B, G)$. Then, the communication overhead is

$$\begin{aligned} &|\mathtt{counter}| + |f^-| + \sum_{i=1}^{c} |\mathtt{kw}_{j_i}| + \sum_{i=1}^{m} |\Gamma_i| \\ &= |\mathtt{counter}| + |f^-| + c\lambda + 4m((s-1)\lambda + 1) \\ &\simeq |\mathtt{counter}| + |f^-| + (c + 4m(s-1))\lambda \end{aligned}$$

where s is the number of gates of f.

5.4 More Efficient Variant

In the search phase, let c be the input size of f, i.e, the number of search keywords. If c is small, then we can consider a more efficient variant such as follows. We can naturally extend our garbling scheme to f which consists of a single gate whose fan-in is c. Then, while the communication overhead of the store phase remains the same, that of the search phase is reduced to

$$|\mathtt{counter}| + c\lambda + 2^c \cdot m.$$

(For example, if $c = 2$, then $|\Gamma_i| = 4$ as can be seen from the next section.)

Suppose that $\lambda = 128$. Then this, variant is more efficient for $c \leq 7$. Further, no information on even f^- is leaked in this variant. (Namely no information on f is leaked at all.)

6 Example

Consider an example of the store phase shown in Sec. 5.1. After the store phase, the client holds $(\mathtt{counter} = 0, m = 2)$, and keeps (k_e, k_0) secret. In the search phase, suppose that the he wants to retrieve the documents which contain w_1 AND w_2. Namely in eq.(12), he wants to know if $e_{i,1} \wedge e_{i,2} = 1$ for $i = 1, 2$. Then, the client does the following.

1. Let $\mathtt{counter} \leftarrow \mathtt{counter} + 1$.
2. Compute $\mathtt{kw}_1 = \mathtt{PRF}_{k_0}(w_1)$ and $\mathtt{kw}_2 = \mathtt{PRF}_{k_0}(w_2)$.
3. For $i = 1, 2$, compute $(v^0_{i,1}, v^1_{i,1})$ and $(v^0_{i,2}, v^1_{i,2})$ according to Eq.(8) and Eq.(9).
4. For simplicity, suppose that
$$lsb(v^0_{i,1}) = lsb(v^0_{i,2}) = 0,\ lsb(v^1_{i,1}) = lsb(v^1_{i,2}) = 1$$
for $i = 1, 2$.
5. For $i = 1, 2$ and $(x, y) = (0, 0), \ldots, (1, 1)$, compute
$$P_i(3, x, y) \leftarrow H_1(\mathtt{counter}, 3, v^x_{i,1}, v^y_{i,2}) \oplus (x \wedge y).$$
(See Eq.(2) $\sim$ Eq.(5).)
6. For $i = 1, 2$, let
$$\Gamma_i \leftarrow [P_i(3, 0, 0), P_i(3, 0, 1), P_i(3, 1, 0), P_i(3, 1, 1)].$$
7. Send $[\mathtt{counter}, (\mathtt{kw}_1, \mathtt{kw}_2), (\Gamma_1, \Gamma_2)]$ to the server.

The communication cost is $|\mathtt{counter}| + 2\lambda + 4 \times 2$ bits. If there are m documents, then the communication cost is $|\mathtt{counter}| + 2\lambda + 4m$ bits.

The server has the table of Table 3. She now receives

$$[\mathtt{counter}, (\mathtt{kw}_1, \mathtt{kw}_2), (\Gamma_1, \Gamma_2)]$$

from the client. Then she does the following.

1. From $(\mathtt{kw}_1, \mathtt{kw}_1)$ and Table 3, find $(v^1_{1,1}, v^1_{1,2})$ and $(v^1_{2,1}, v^0_{2,2})$.
2. Compute

$$(lsb(v^1_{1,1}), lsb(v^1_{1,2})) = (1, 1)$$
$$(lsb(v^1_{2,1}), lsb(v^0_{2,2})) = (1, 0)$$

3. Compute

$$z_1 = P_1(3, 1, 1) \oplus H_1(\mathtt{counter}, 3, v^1_{1,1}, v^1_{1,2}) = 1 \wedge 1 = 1$$
$$z_2 = P_2(3, 1, 0) \oplus H_1(\mathtt{counter}, 3, v^1_{2,1}, v^0_{2,2}) = 1 \wedge 0 = 0$$

4. Return only C_1 because $z_1 = 1$ and $z_2 = 0$.

References

1. M. Bellare, A. Desai, E. Jokipii, P. Rogaway: A Concrete Security Treatment of Symmetric Encryption. FOCS 1997: pp.394–403 (1997)
2. Boaz Barak, Iftach Haitner, Dennis Hofheinz, and Yuval Ishai. Bounded key-dependent message security. In EUROCRYPT, pp.423–444, 2010.
3. Mihir Bellare, Viet Tung Hoang, Phillip Rogaway: Foundations of garbled circuits. ACM Conference on Computer and Communications Security 2012: 784-796
4. Lucas Ballard, Seny Kamara, Fabian Monrose: Achieving Efficient Conjunctive Keyword Searches over Encrypted Data. ICICS 2005, pp.414-426 (2005)
5. J. W. Byun, D. H. Lee, and J. Lim: Efficient conjunctive keyword search on encrypted data storage system. EuroPKI, pp.184–196 (2006)
6. N. Baric and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Advances in Cryptology: Proc. EUROCRYPT, volume 1233 of LNCS, pages 480–494. Springer-Verlag, 1997.
7. R.Curtmola, J.A. Garay, S.Kamara, R.Ostrovsky: Searchable symmetric encryption: improved definitions and efficient constructions. ACM Conference on Computer and Communications Security 2006: pp.79–88 (2006)
8. David Cash, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel Rosu, Michael Steiner: Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries. CRYPTO 2013.
9. ePrint version of the above paper: Cryptology ePrint Archive, Report 2013/169, http://eprint.iacr.org/
10. M. Chase and S. Kamara: Structured encryption and controlled disclosure. ASIACRYPT 2010, pp. 577–594 (2010)
11. Y.Chang and M.Mitzenmacher: Privacy Preserving Keyword Searches on Remote Encrypted Data. ACNS 2005: pp.442–455 (2005)
12. Eu-Jin Goh: Secure Indexes. Cryptology ePrint Archive, Report 2003/216, http://eprint.iacr.org/ (2003)
13. Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In CRYPTO, pp.465–482, 2010.
14. Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. A simple BGN-type cryptosystem from LWE. In EUROCRYPT, pp.506–522, 2010.
15. Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In CRYPTO, pp.39–56, 2008.

16. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In STOC, pp.218–229, 1987.
17. Philippe Golle, Jessica Staddon, Brent R. Waters: Secure Conjunctive Keyword Search over Encrypted Data. ACNS 2004, pp.31-45 (2004)
18. Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, Nickolai Zeldovich: Reusable garbled circuits and succinct functional encryption. STOC 2013: pp.555-564
19. Kaoru Kurosawa, Yasuhiro Ohtaki: UC-Secure Searchable Symmetric Encryption. Financial Cryptography 2012: 285-298
20. Kaoru Kurosawa, Yasuhiro Ohtaki: How to Update Documents Verifiably in Searchable Symmetric Encryption. CANS 2013: 309-328
21. Seny Kamara and Charalampos Papamanthou: Parallel and Dynamic Searchable Symmetric Encryption. FC 2013
22. Seny Kamara, Charalampos Papamanthou, Tom Roeder: Dynamic searchable symmetric encryption. ACM Conference on Computer and Communications Security 2012: 965-976
23. Yehuda Lindell and Benny Pinkas. A proof of security of Yao ’s protocol for two-party computation. J. Cryptol., 22:161–188, April 2009.
24. D.Song, D.Wagner, A.Perrig: Practical Techniques for Searches on Encrypted Data. IEEE Symposium on Security and Privacy 2000: pp.44–55 (2000)
25. Peishun Wang, Huaxiong Wang, Josef Pieprzyk: Keyword Field-Free Conjunctive Keyword Searches on Encrypted Data and Extension for Dynamic Groups. CANS 2008: 178-195
26. Andrew C. Yao. Protocols for secure computations. In FOCS, pp.160–164, 1982.
27. Andrew C. Yao. How to generate and exchange secrets (extended abstract). In FOCS, pp.162–167, 1986.