On the Security of Data Markets and Private Function Evaluation

István Vajda
Dept. of Informatics, TU Budapest, Hungary
Email: vajda@hit.bme.hu

Abstract: The income of companies working on data markets steadily grows year by year. Private function evaluation (PFE) is a valuable tool in solving corresponding security problems. The task of Controlled Private Function Evaluation and its relaxed version was introduced in [Horvath et.al., 2019]. In this article, we propose and examine several different approaches for such tasks with computational and information theoretical security against static corruption adversary. The latter level of security implies quantum-security. We also build known techniques and constructions into our solution where they fit into our tasks. The main cryptographic primitive, naturally related to the task is 1-out-of-n oblivious transfer. We use Secure Multiparty Computation techniques and in one of the constructions functional encryption primitive. The analysis of the computational complexity of the constructions shows that the considered tasks can efficiently be implemented, however it depends on the range of parameter values (e.g. size of database, size of the set of permitted function), the execution environment (e.g. concurrency) and of course on the level of security.

Keywords: private function evaluation, secure database access, secure multiparty computation, functional encryption

1. Introduction

The growing importance of data is beyond question today [Cattaneo et.al., 2019], [Otto et.al., 2020]. Due to the current technological trends (proliferation of smart devices and the Internet of Things (IoT)), an increasing amount of data is waiting for utilization. According to the business model of a data market, a data broker buys data from the owners and sells the collected data (possibly in processed form) to a third party that provides value-added services to its users. Serious security concerns may arise in connection with the operation of this model. We briefly mention a few real-life examples.

DNA database contains information about the purpose of each gene. Such databases are extremely valuable and thus those are not sold on a whole, but rather users are charged per access to the database. On the other hand, the particular DNA sequences accessed by a user (e.g. a pharmaceutical company) reveal a lot of information about the interests of the user, e.g., for which disease it is developing a medicine. Similarly, requests sent to the stock quotes database can reveal information about the investment strategy of the requester or patent search patterns can reveal sensitive business information. The database owner wants to control which subscriptions allow access to which types of information (it is quite likely that subscription prices vary with the type of accessed information). However, this must be reconciled in some way with the user's need for privacy.

Private function evaluation (PFE) is a valuable tool in solving this problem. Private function evaluation (PFE) is a two-party computation, where the input of user U is an efficiently evaluable function f, $f \in F$, and the input x of server S is a value from the domain of f. The user receives output f(x). The corresponding functionality defined by its I/O behavior is the following: $(x, f) \rightarrow (-, f(x))$.

A refined related task is the Controlled PFE (CPFE) where server S as part of its input gives also a set of impermissible functions F_A, $F_A \subset F$, i.e. $((x, F_A), f) \rightarrow (-, f(x))$, if $f \notin F_A$, else abort. Note if we allowed the identity function as an input value of the user it would completely reveal the input of the server. A relaxed version of CPFE (rCPFE) is the task where user U shares his set of functions F_B with server S, i.e. $((x, F_A), (f, F_B)) \rightarrow (-, f(x))$, if $f \in F_B \backslash F_A$, else abort, [Horvath, 2019]. Note the server can control the type of information accessible by the user by choosing the set of impermissible functions F_A appropriately. Similarly, the user reveals only a set of functions (F_B, $F_B \subset F$), from which it wants to select its input without revealing the actually selected function (f).

In this paper, we propose and examine different solutions for such tasks. We consider both computationally and information-theoretically (IT) secure constructions. The main cryptographic components in our constructions are 1-out-of-n OT (Oblivious Transfer), SMC based on the honest majority as well as functional encryption (FE). We build also on known techniques and constructions where they fit into our task.

One of the computationally secure constructions is a straightforward one based on a computationally secure 1-out-of-n OT protocol. Briefly, instead of the original database, the user has (controlled) access to a derived database containing all permitted mappings of the original database items. In the two-phase version, the user first obliviously selects the records whose functions he wants to know. In this version, we assume the availability of auxiliary servers with an honest majority and we perform secret sharing based Secure Multiparty Computation (SMC).

The other computationally secure construction uses the FE-primitive. In nutshell, the principle of operation of the protocol is as follows. The user first obliviously selects a set of encrypted database records ($E(x_1),\ldots, E(x_n)$), $n \geq 1$ encrypted by the server. Next, using another oblivious transfer protocol user obtains a secret function key (sk_f) matching its input ($f, f \in F_B \backslash F_A$). Finally, the user computes its output $f(x_1),\ldots,f(x_n)$ by decrypting the ciphertexts.

In the IT-secure constructions, we adapt SMC techniques and constructions [Araki et.al., 2016], [Ishai et.al., 2013], and [Furukawa et.al., 2023].

The price of this high level of security is the required access to a set of servers with an honest majority or to a trusted correlated randomness setup server, furthermore access to perfect secure communication channels. An important advantage of IT-secure solutions (perfect or statistical) is that they provide quantum security. The main price factor is high randomness complexity.

As the communication media for the protocol will probably be a publicly accessible network (e.g. Internet), instances of other, potentially hostile protocols (i.e. protocols designed to attack our protocol) may also use the same communication space simultaneously. Therefore a desirable feature is a security in general concurrent execution environment ([Canetti, 2002], [Canetti et.al., 2002]), which is achieved by some of the presented constructions. Of course, this entails additional costs, in particular the availability of appropriate trusted setups.

Considering potential practical applications, the main related factors are (communication-, computation-, randomness-) complexity, setup, and trust assumptions. IT-secure or UC-secure constructions always require some kind of trusted setup (i.e. non-standard model) or honest majority. Deciding whether a construction can be practical depends on the size of the task (the number of items in the database, the binary length of database items, the size of the set of permitted functions), the circuit complexity of the functions, the speed of communication channels, the probability that an assumed honest majority exists. From this point of view, we do not carry out a comprehensive comparative analysis in this paper, and we mainly focus on the methods of construction. At the same time, we highlight the application-related advantages and disadvantages of each construction together with numerical examples of complexity values.

With this paper, we wanted to contribute to the development process of security technologies in the emerging field of data markets.

The structure of the paper is as follows. In Section 2 we present related works and we name our contributions. In Sections 3 and 4 we present the computationally and information-theoretically secure constructions, respectively. In Section 5 we discuss related issues on UC security as well and here we show a summary table of advantages and disadvantages of approaches. Finally, we derive conclusions.

2. Related works

Protocols presented in this article use secure multiparty computation (SMC) techniques, but we also show a construction based on functional encryption. We show protocols with both computational and unconditional security.

The two generic approaches for constructing SMC protocols are the secret-sharing and the garbled-circuit approaches. The protocols we present are based on the secret-sharing approach. Such protocols have a number of rounds that are linear in the depth of the circuit being computed (i.e. not constant). At the same time, their demand for communication bandwidth is more favorable.

SMC has the property that it can be made information-theoretically secure (IT-secure), which means that it does not rely on some computational hardness assumption. The level of IT security can be perfect or statistical. The security of such protocols holds against adversaries with unlimited computing power, be it classical or quantum computing.
The IT-secure solutions require an honest majority or correlated randomness setup. We show solutions with such assumptions.

[Furukawa et.al., 2023] show a generic approach for secure three-party (n=3) computation of any functionality, with an honest majority and a malicious adversary. It provides security against malicious adversaries by using Beaver's multiplication triple approach (a version of correlated randomness) together with the statistical cut-and-choose technique. Recall for comparison that by the classic result in the case of $t<n/2$ cheaters are just honest-but-curious plus we assume broadcast channels, in contrast in construction [Furukawa et.al., 2023] the adversary is malicious and no broadcast channel is assumed. This construction has both an IT-secure and computationally secure variant, we adapt both to our task, in particular to the SMC evaluation of 1-out-of-n oblivious transfer.

In case of the availability of an appropriate correlated randomness setup, secure computation can be achieved with no honest majority. We show a perfectly secure solution for

our task against malicious parties based on correlated randomness setup by adapting a construction technique from [Ishai et.al., 2013]. Here we cite [Ishai et.al., 2013]: "Any 2-party "sender-receiver" functionality, (which takes inputs from a sender and a receiver and delivers an output only to the receiver), can be perfectly realized against malicious parties given correlated randomness." Note oblivious transfer is such a kind of functionality.

We also consider an FE-based solution for our task. Functional encryption (FE) is naturally related to the PFE task. However, there are two main obstacles. The set of functions such that practically efficient FE construction is known is rather restricted. Fortunately, this is not a problem for linear functionalities and multivariate quadratic polynomials, see [Abdalla et.al., 2015] and [Elisabetta et.al., 2017], respectively. In our application, a quadratic polynomial can express many statistical functions for the statistical analysis of database items. Unfortunately, there is a bigger obstacle. In the case of non-adaptive FE, we can work in the standard cryptographic setting, however, in the case of adaptive input selection, the best we can hope for is a realization (of the FE primitive) in a random oracle setting [Boneh et.al., 2011]. Partly due to limited space, we show the cryptographic analysis of this construction in detail, while in the case of the other constructions we mainly examine the issue of computational complexity and efficiency concerning our task.

A database security task that is related to our goal is called an oblivious transfer with access control (AC-OT) [Camenisch et.al., 2009]. Their protocol provides the following security guarantees: "Only authorized users can access the record; the database provider does not learn which record the user accesses; the database provider does not learn which attributes or roles the user has when she accesses the database". In contrast, we allow the user to learn (any permitted) efficiently computable mapping of database records. We consider also IT-secure constructions, furthermore UC-security, and quantum safety. Furthermore, in several constructions, we allow active adversaries.

Report [Horvath, 2019] initiated our work. This report provides preliminary thoughts about the realization of this task with computational security, including a sketchy analysis in the stand-alone setting against a static semi-honest adversary.

The contributions of the paper are as follows: this paper

- presents several constructions in connection with the security problems of the data market, more specifically for the task of Controlled Private Function Evaluation of obliviously chosen database items, where we consider both the computational and the information theoretical security,
- defines ideal functionality (FrCPFE) for the task of relaxed Controlled Private Function Evaluation,
- shows versions with postquantum security as well as with UC security,
- presents numerical examples for the evaluation of computational complexity.

3. Computationally-secure constructions

First, we give a brief overview of the main steps of constructions. We consider three protocols with computational and two protocols with IT security. The adversary model, network model, and trust assumptions are given per construction.

Within the computationally secure constructions, we consider one- and two-phase approaches. The one-phase approach is conceptually the most straightforward. The database server computes a derived database such that it stores all permitted mappings ("statistics") of database items. The user chooses an item obliviously from the derived database. In the two-phase approach, first, the wanted database item x is SMC-selected so that (in this phase) x remains unknown for both the user and the server, and next mapping f(x) is SMC-computed so that x (in its entirety) remains unknown for the user and f remains unknown for the server.

The third computationally secure construction uses an FE-primitive. It is a two-phase approach. In the first phase user chooses encrypted database records $(E(x_1),\ldots, E(x_n))$, $n\geq 1$ obliviously to the server. In the second phase user obliviously chooses a secret function key (sk_f). Finally, the user computes output $f(x_1),\ldots,f(x_n)$ by decrypting the chosen ciphertexts.

In all of the constructions presented in this paper, an agreement on the subset of functions permitted by the server to use by the user is achieved by exchanging the related sets of functions (F_A, F_B). This beginning part of the protocols is omitted from the presentation of the constructions.

The two main issues related to the constructions are the security guarantee and the complexity. We show numerical examples for the calculation of the complexity to see the relationship between the size of the task (the size of the database and the set of permitted functions) and its practical applicability. If for a size, a construction seems to be inefficient the following straightforward technique may help:

We can decrease the computational overhead at the expense of a weakened privacy guarantee (for the user). We partition the database and a user first reveals to the server the identifier of the portion from which it wants to select an item. Random selection of portions may enhance privacy especially if the order of the items in the database is related to their content. This straightforward partitioning technique can be used for all constructions in this paper. Accordingly, when (in the subsequent chapters) we analyze the implementability of a protocol concerning the database size, we can consider those sizes as the size of a database portion. In this sense, all of the protocols can be implemented with practical efficiency if we can accept a privacy-error probability 1/S where S is the appropriate size of a portion (the latter probability is the probability of the event that a semi-honest server correctly blind-guesses the database item the user is interested in). So, essentially, a decision on the practical efficiency of a construction is transformed into a decision on the accepted level of privacy error.

In numerical examples, we want to know whether a protocol can be used in practice or not (for complexity reasons). These numerical examples aim to illustrate the evaluation of complexity, and in the case of a concrete application task, the applicability of a protocol must be decided based on the actual parameter values.

3.1 A straightforward construction

Database server generates a derived database: it stores $f_i(x_j)$ for all $i\in I$, $j\in X$, where $\{f_i, i\in I\}$ is the set permitted functions and $\{x_j, j\in X\}$ is the set of database items. Let m be the (uniform) size of database items in the derived database. A computationally secure 1-out-of-N, $N=|I|\cdot|X|$ OT protocol is executed.

An advantage of this solution (in addition to its simplicity) is that no auxiliary server is required. The obvious disadvantage is the significant increase in complexity caused by the |I|-times increase in the size of the (original) database.

Example: An 1-out-of-N OT protocol can be constructed from 1-out-of-2 OT protocol by an appropriate extension step. In the Naor-Pinkas's extension ([Naor and Pinkas, 1999]) the additional complexity of extension is the computation of decryption keys that requires number log N l-out of-2 OT executions plus kN evaluations of a PRF ($N=2^k$ for some k). Communication complexity is determined by the transmission of N encrypted database items plus log N decryption keys. Accordingly, the communication complexity is $mN + l \log N$ bits, where l is the length of the secret key of the PRF. The randomness complexity is small. For instance, if 1 second is the bound on latency then the required speed of communication channel between the server and the user is at least this complexity value. For instance, a big-size task with 1 million records, 256 permitted functions, and a record length of 1kb a communication channel with a speed of at least 256 Gbps is required between the server and (each) user. We can significantly reduce this complexity for the user by the two-stage approach detailed subsequently.

3.2 A construction with auxiliary servers

Instead of choosing an item from the derived database with a size of $N=|I|\cdot|X|$ items, first we choose an item from the (original) database of |X| elements, next from a set of |I| elements while keeping the privacy requirements.

This protocol assumes the availability of auxiliary servers with appropriate trust assumptions (detailed subsequently). For the simplicity of presentation first suppose that parties (database server, user) have access to two sets of three auxiliary servers in each. These servers are denoted servers D_1, D_2, and D_3 (auxiliary D-servers) in the first set and F_1, F_2, and F_3 (auxiliary F-servers) in the second set. We assume an honest majority in both of these sets of auxiliary servers. The main steps of the protocol are summarized in Fig. 1.

Step 1: Database server and user distribute shares of the (original) database and shares of the chooser index (pointing at a database item x), respectively, among auxiliary D-servers. D-servers run a 3-party secret-sharing type secure function evaluation algorithm for the evaluation of 1-out-of-|X| OT functionality as a circuit. D-servers compute shares $[x]_k$, k=1,2,3 of database item x.

Step 2: The set of F-servers 3-party SMC-evaluate function $f_i()$ on the shares of x, for all $i\in I$ as follows. Server F_k receives as input share $[x]_k$ from the output of the D-server assigned to it and SMC-computes share $[f_i(x)]_k$ of $f_i(x)$ for all $i\in I$ and stores these shares, k=1,2,3.

Step 3: User executes 1-out-of-I OT protocol with each F-server separately in a 2-party computation and learns shares $[f_{i*}(x)]_k$, k=1,2,3, where $f_{i*}()$ is the function chosen by the user. Finally, the user combines the shares and outputs $f_{i*}(x)$.

Fig. 1: Main steps of the two-stage protocol

This construction provides the following security guarantees. Dishonest auxiliary servers learn neither x nor f. The user can learn the mapping of a database item computed with a

permitted function only, while the entire database item x remains hidden from the user (assuming the identity function is not included in the set of permitted functions).

An important observation is that a single set of three auxiliary servers is sufficient. The above setup of servers can be reduced to a single set of auxiliary servers by using the servers in all three steps.

For an implementation of the protocol, below apply the computationally secure version of the generic protocol of [Furukawa, 2023] as a subprotocol. This is a protocol for secure three-party computation of any functionality, with an honest majority and a malicious adversary. We apply it to compute the OT functionalities in Steps 1 and 2. Accordingly, an honest majority is assumed among the auxiliary servers with at most one malicious server. Construction [Furukawa, 2023] uses the Beaver's multiplication triple correlated randomness technique [Beaver, 1995]. These triplets (together with a cut-and-choose statistical technique) are used to force a malicious party to compute correct (AND) values (and remain semi-honest). The AND-complexity of the circuit determines the computational complexity of the protocol. Accordingly, to assess the complexity of our protocol we need an upper bound on the AND complexity of 1-out-of-n OT (m=1). Our corresponding claim is as follows:

Claim 1: An upper bound on the AND complexity of 1-out-of-n OT (m=1) is $n\cdot\log_2(n)$. An upper bound on the circuit size (when both the AND and the XOR gates are considered) is twice this value plus n-1. ($n=2^k$, $k\geq 1$)

Proof: For the simplicity of presentation, we show the circuit complexity for an instance, but the general case can be reconstructed from this easily.

Let n=4 and we use the following notation: the input of the sender is denoted by $x_i \in \{0,1\}$, where $i\in\{0,1,2,3\}$, the input of the receiver (a 2-bit chooser string) is denoted by $(\sigma_0, \sigma_1) \in \{0,1\}^2$. The following circuit implements a 1-out-of-4 OT functionality:

$$x_0\cdot(\sigma_0+1)\cdot(\sigma_1+1)+ x_1\cdot(\sigma_0+1)\cdot(\sigma_1+0)+x_2\cdot(\sigma_0+0)\cdot(\sigma_1+1)+x_3\cdot(\sigma_0+0)\cdot(\sigma_1+0)$$

Accordingly an upper bound on the number of XOR and AND gates is $n(\log_2(n)+1) - 1$ and $n\log_2(n)$, respectively. ■

To feel the magnitude of the computational complexity of the protocol, now we estimate it for the following numerical values of parameters: 1 million database items with a size of 256 bits each and 256 different permitted functions (statistics), i.e. $|X|=2^{20}$, $|I|=2^8$, $N=2^{28}$, $m=2^8$. Let's consider the first phase with 1-out-of 2^{20} OT. We get an upper bound $2^8\cdot 2^{20}\cdot 20 \sim 5.2$ billion on the number of AND gates. For an estimate of the computational complexity measured in the time of execution of the OT protocol, we rely on related experimental data published in the paper [Furukawa, 2023]. We cite this paper: "On a cluster of three 20-core servers with a 10Gbs connection, the protocol achieves a rate of computation of 7 billion AND gates per second." Extrapolating from these figures to our case, the first phase of the protocol can be executed in time less than a second.

A relatively high-speed communication channel between the database server and the auxiliary servers is still required. Indeed, random shares of the original database is transmitted in Step 1. This requires a channel speed of at least $m|X|$ bps (at 1 sec latency), however it is significantly smaller than $m|X||I|$ bps we got in the case of a straightforward approach. The practically more important advantage is that the communication complexity for the user drastically decreases, requiring a channel speed of at least $m|I|$ bps between the

user and each auxiliary server (see Step 3). Notice the set of three auxiliary servers with an honest majority is on the cost side of this advantage.

The randomness complexity of the database server significantly increases compared to the straightforward solution, since the server computes three random shares of the database. Assuming this triplet of shares for a database item x is $(x+r_1+r_2, r_1, r_2)$ where x, r_1, $r_2 \in \{0,1\}^m$, and r_1, r_2 are randomly chosen, the number of random bits used by the database server is at least $2m|X|$. For instance, a large task when $|X|=2^{20}$ and $m=2^9$ requires ~ 1 billion truly random bits. The server must run a hardware random number generator (TRNG) speed of 1Gbps to keep 1 second latency. The aim of this latter numerical example is to show that the implementation of even such large tasks is not unrealistic (e.g. PEN-drive size random generators with the speed of the order of 100Mbps are commercially available). Medium-size tasks can be implemented with less extreme high-speed TRNG.

3.3 A construction with functional encryption

In this two-phase construction, no auxiliary servers are required. Informally speaking, we replace the SMC computation with FE encryption. Briefly, in the first phase we use a 1-out-of-|I| OT protocol (OT1) to choose the secret function key to function f_{i*}. In the second phase, we execute a 1-out-of-|X| OT protocol (OT2) to choose an FE-encrypted database item E(x). Finally, user decodes $f_{i*}(x)$.

As mentioned in Section 2 there are two serious drawbacks. Firstly, the set of functions for which practically efficient FE construction is known is rather restricted. Fortunately, this is not a problem for linear functionalities and multivariate quadratic polynomials, see [Abdalla et.al., 2015] and [Elisabetta et.al., 2017], respectively. Fortunately, the latter has several practical applications as a quadratic polynomial can express many statistical functions for statistical analysis of database items.

Secondly, in practical application the FE-adaptivity issue arises naturally, implying that it is necessary to assume the availability of a public random oracle (to achieve provable security). A public random oracle is a trusted external party accessible by all parties of the protocol and behaves like a truly random function. Such an oracle is a theoretical construct. However, there are special, interesting cases where the FE adaptivity issue does not arise. Such cases are batch processing and parallel processing scenarios. A frequent change of the FE master key pair may also resolve this problem.

We consider the case of passive (semi-honest) and active (malicious) adversaries, both. A passive adversary attacks the privacy of the parties. An active adversary that corrupts the server may cheat with the function keys to distort the output of the user. Accordingly, guarantees against a passive adversary are as follows: the entire x is hidden from the user, the function chosen by the user is hidden from the server and the server controls the set of permitted functions. In addition to the guarantees against a passive adversary, the protocol provides also correctness properties for the following type of active attack: a malicious server cheats with a function key in such a way that the function key provided by the server does not match the request

We do not consider a countermeasure against dishonest computation of encryptions of database items because this is more or less equivalent to changing (distorting) database items directly and we exclude such an attack. (We note that it is possible to expand the attacker's model with this attack, but this causes a significant increase in the complexity of a corresponding provably secure protocol.)

In the case of a passive adversary, we do the analysis in three steps as follows:

1. definition of ideal functionality F_{rCPFE}
2. definition of OT1, OT2 – hybrid protocol
3. definition of simulator for the hybrid protocol and the analysis of its success

For security against an active adversary, we extend the semi-honest protocol. The user verifies the correctness of the function keys by using the function keys on test ciphertexts generated by itself. We can modify the above steps of analysis accordingly.

Here we make a presentation-technical observation. The protocol can be viewed as a successive execution of two OT sub-protocols, so from space saving reasons we shorten the presentation of the ideas of analysis. Accordingly, subsequently, we focus on the first phase (on oblivious selection of function keys) and we simplify the second phase. The server simply transmits to the user the encryption of a subset $\{x_1,\ldots,x_n\}$ of database items, i.e. we omit the OT2 sub-protocol in the subsequent presentation of the analysis.

3.3.1 Single session ideal functionality F_{rCPFE}

Definition of single session ideal functionality F_{rCPFE} is shown in Fig. 2.

Parties: S (server), U (user). Parameters: F (base set of functions)

1. Upon receiving an input (**KeyGen**, sid) from party S, ideal functionality verifies that sid = (S, sid') for some fresh value sid'. If not so, then it ignores the input. Otherwise, it hands (**KeyGen**, sid) to the (ideal system) adversary (simulator, Sim). Sim generates public key pk. Upon receiving (**Pub_key**, sid, pk) from Sim, it outputs (**Pub_key**, sid, pk) to S.

2. Upon receiving an input message (***InicUser***, sid, F_B) from party U, it verifies that {sid=(S, sid')}. If not so, then it ignores the input. Else, it verifies that {$F_B \subseteq F$}. If so, then it stores (sid, F_B) and sends a private delayed message (***InicUser***, sid, F_B) to party *S*, else it sends message Abort1 to party S and aborts the session.

3. Upon receiving an input message (***InicServer***, sid, F_A) from party S, it verifies that {sid=(S, sid')}. If not so, then it ignores the input. Otherwise, it verifies that $F_A \cap F_B$ is not empty. If so, then it stores (sid, F_A) and sends a private delayed message (***InicServer***, sid, F_A) to party U, else it sends message Abort2 to party U and aborts the session.

4. Upon receiving an input message (***InputUser***, sid, f) from party U, it verifies that {sid=(S, sid'). If not, it ignores the input, else stores (sid, f).

5. Upon the condition that {"Inic" inputs have been set for both parties} AND {an input function f from party U has already been stored} it verifies that {$f \in F_B \backslash F_A$}. If not so, then it sends the message Abort3 to (honest) party S and aborts the session.

6. Upon receiving an input message (***InputServer***, sid, $x_1,\ldots,x_n$) from party *S*, it verifies if {sid=(S, sid')}. If not so, then it ignores the input, else it stores (sid, $x_1,\ldots,x_n$).

7. Upon the condition that {all inputs with id sid have been set for both parties}, it sends output (***Output***, sid, $f(x_1),\ldots,f(x_n)$) to party U and halts the session.

8. Upon receiving ***(Corrupt,*** sid, ssid, U) from the adversary followed by a pair (f', F'_B) and no output has been yet written to U in session sid, then it stores (sid, f, F_B), where f=f', $F_B = F'_B$).

Upon receiving ***(Corrupt,*** sid, S*)* from the adversary and an input (sid, $x_1,\ldots,x_n$, F_A) has already been received from S, ideal functionality discloses this input to the adversary. If the adversary tries to change honest output ($y_1=f(x_1),\ldots, y_n = f(x_n)$) to party U to a different (well-formatted) sequence ($y'_1=g(x'_1),\ldots, y'_n = g(x'_n)$), where g, $g \neq f$, $g \in F$ ideal functionality sends output Abort 4 to honest party U and terminates.

Fig. 2: Single-session ideal functionality F_{rCPFE}

Intuitively, session id sid=(S, sid') is associated with a FE key pair (PK, MK) generated by server S. A single user sends a single database request to the database.

3.3.2 F_{OT1}-hybrid realization of functionality F_{rCPFE}

F_{OT1}-hybrid protocol π is shown in Fig. 3. (Plain model of communication is assumed.)

Key setup:

Party S calls *KeyGen*(1^n) → (PK, MK) algorithm of the FE scheme and publishes PK.

The protocol:

1. U → S	: F_B; if not $F_B \subseteq F$, then S outputs Abort1 and aborts
2. S → U	: F_A; if not $F_A \subseteq F$ or $F_A \cap F_B$ is empty then U outputs Abort2 and aborts
3. U → F_{OT1}	: (the index of function) f
4. S → *func.KeyGen*	: $\{f_1,\ldots,f_m\}$ (= $F_B \setminus F_A$)
func.KeyGen → S	: $sk_{f1}, \ldots, sk_{fm}$
S → F_{OT1}	: $sk_{f1}, \ldots, sk_{fm}$
5. F_{OT1} → U	: sk_f , if $f \in F_B \setminus F_A$ else
	⊥ , when S outputs Abort3 and aborts
U	: if Ver(sk_f)=0, then U outputs Abort4 and aborts
6. S → *Encrypt*	: $x_1,\ldots,x_n$
Encrypt → S	: $c_1=E(x_1),\ldots,c_n=E(x_n)$
S → U	: $c_1,\ldots,c_n$
7. U	: outputs $D_{skf}(c_1),\ldots,D_{skf}(c_n)$

Fig. 3: The F_{OT1}-hybrid protocol π for realization of single-session functionality F_{rCPFE}

In the single-session scenario, the database serves a single request with a single input function f. The order of message sendings (i.e. the order of requesting secret function keys and ciphertexts) will guarantee a Non-Adaptivity (NA) setting for the FE-primitive. Specifically, the user has to send its request (function f) before it sees encrypted database records.

Assume we impose an (a priori) upper bound q1 on the number of secret function key queries from FE key setup functionality *func.KeyGen* (i.e. m≤q1 above), similarly an upper bound q2 on the number of encrypted messages (i.e. the number of database items ≤q2). Accordingly, we assume the availability of a (q1, q2)-NA-SIM (Non-Adaptive SIMulatably secure) functional encryption scheme. Such construction exists in the standard setting [Gorbunov et.al., 2012].

Theorem 1: Assuming (q1, q2)-NA-SIM FE scheme and simulation-secure 1-out-of-m string OT scheme, F_{OT1}-hybrid protocol π is a secure realization of single-session functionality F_{rCPFE}.

3.3.3 Analysis

First we define the simulator. Simulator Sim runs the FE-simulator. In particular, it can run the key generation (*KeyGen*: $1^n \rightarrow$(PK, MK)), the function key simulation (*func_KeyGen*: f $\rightarrow f_{sk^*}$, f∈F), and the ciphertext simulation (*Encrypt*: x → c*, x∈X) algorithms of the FE-simulator. Simulator Sim simulates also ideal functionality F_{OT}.

We show the steps of the simulation by the steps of the hybrid protocol.

Case 1: corrupt S

FE key setup phase:

Party S receives an input (**KeyGen**, sid) (from the calling environment Z). Sim forwards this input to party S. Upon receiving a call to key generation algorithm *KeyGen* from S, simulator Sim runs the algorithms and hands public key PK to party S. At the same time Sim forwards input (**KeyGen**, sid) also to functionality F_{rCPFE}, and for the corresponding call from the functionality Sim simulates the same public key PK. When F_{rCPFE} outputs public key PK to party S, simulator Sim outputs PK to the calling environment.

Simulation of the main part of F_{OTl} -hybrid protocol:

Step 1. Sim(U)→S:

Honest party U sends an input F_B to functionality F_{rCPFE}. The functionality may send output message F_B or Abort1 to (corrupted) party S. Sim learns this output. If the message is F_B (more precisely, not an abort message) then Sim forwards the message to party S via internal interaction. If the message is Abort1 then Sim forwards it to the environment via external interaction and halts.

Step 2. S→Sim(U):

Party S sends a message F_A to party U. Sim captures the message and sends it as input to ideal functionality F_{rCPFE} on behalf of S via external interaction. The functionality may send output message F_A or Abort2 to (honest) party U.

Step 3. U→Sim(F_{OTl}):

Honest party U sends an input $f \in F_B \backslash F_A$ to functionality F_{rCPFE}.

Step 4.
S → Sim(func.KeyGen):
Sim(func.KeyGen) → S:
Simulator Sim runs the FE-simulator, in this step it runs the function key simulator for party S. Simulator simulates function keys sk^*_{f1}, …, sk^*_{fm} for function requests $f_1,\ldots,f_m$.

S → Sim(F_{OTl}):

Party S sends as input message sk'_{f1}, …, sk'_{fm} to F_{OT}, Sim captures the message and verifies it. Sim randomly chooses $i \in \{1,\ldots,m\}$. If $sk'_{fi} \neq sk^*_{fi}$ then Sim induces Abort 4 for party via the ideal functionality.

Step 6.
S → Sim(*Encrypt*) : $x_1,\ldots,x_n$
Sim(*Encrypt*) → S : $c^*_1=E^*(x_1),\ldots,c^*_n=E^*(x_n)$

Simulator Sim runs the FE-simulator, in this step, it runs the encryption simulator for party S. The simulator simulates ciphertexts $c^*_1,\ldots,c^*_n$ for plaintexts $x_1,\ldots,x_n$. Simulator sends $x_1,\ldots,x_n$ as input to ideal functionality F_{rCPFE} on behalf of S via external interaction. Ideal functionality F_{rCPFE} outputs $f(x_1),\ldots,f(x_n)$ to party U.

Case 2: corrupt U

There is a preliminary step of key generation similar to the case of corrupt S.

Step 1. U→Sim(S):

Party U sends a message F_B to party U. Sim captures the message and sends it as input to ideal functionality F_{rCPFE} on behalf of U via external interaction.

Step 2. Sim(S)→U:

Party S sends an input F_A to functionality F_{rCPFE} that the functionality outputs to party U. Sim learns this output and forwards it to party U.

Step 3. U→ Sim(F_{OT1}):

Party U sends (the index of) a function f as input to functionality F_{OT}. Sim captures the message and sends it as input to ideal functionality F_{rCPFE} on behalf of U via external interaction.

Step 5. Sim(F_{OT1})→U:

(In this step simulation takes place under output constraint ($f(x_1),\ldots,f(x_n)$.)

Ideal functionality F_{rCPFE} outputs $f(x_1),\ldots,f(x_n)$ to party U. Simulator captures this message and learns the output constraint.

Simulator prepares and simulates ciphertexts $c''_1,\ldots,c''_n$ and a function key sk" such that the ciphertexts are decoded into the wanted outputs $f(x_1),\ldots,f(x_n)$ with decoding key sk"., respectively.

The simulator forwards sk" to U via internal interaction.

The simulator sends the simulated ciphertexts to party U via internal interaction. The simulator forwards messages between party U and the environment.

■

By definition the view of a corrupted party (adversary) at a step of a protocol: {input, random tape, messages received so far}. We perform (stand-alone) straight-line simulation so that message transmissions are in sync with the order of the messages in the real system. Therefore, it is sufficient to examine the indistinguishability of the views of the adversary only at the end of the run. We distinguish the executions with and without an abort event.

Case of no abort

Message views of corrupted parties are as follows (received messages are shown in time order of their arrival):
Ideal system (F_{rCPFE}):
Corrupted S: F_B, sk^*_{f1}, …, sk^*_{fm}, $c^*_1,\ldots,c^*_n$
Corrupted U: F_A, sk", $c''_1,\ldots,c''_n$, $f(x_1),\ldots,f(x_n)$
Real system (F_{OT1} -hybrid protocol):
Corrupted S: F_B, sk_{f1}, …, sk_{fm}, $c_1=E(x_1),\ldots,c_n=E(x_n)$
Corrupted U: F_A, sk_f, $c_1,\ldots,c_n$, $f(x_1),\ldots,f(x_n)$
The FE simulation guarantees that the matching views are indistinguishable.

Case of abort: We have to show the indistinguishability of the two systems in case of aborts. Concretely, we have to show that the corresponding probabilities of aborts are equal in the two systems. Note this is obvious for abort events caused by dishonestly chosen function sets and functions (Abort1-3).

In the real system, an abort event Abort4 happens when corrupted party S chooses function keys as input to functionality OT1 maliciously. The randomized test performed by the simulator results in equal probabilities of abort (Abort4) in the real and the ideal system. ■

3.3.4 Extensions to the single-session scenario

1. Batch processing of multiple database requests:

A frequent (i.e. per database query) change of the FE key pair (PK, MK) limits the application possibilities of the single-session scenario. We can significantly reduce this disadvantage by making the user interested in sending its requests to the database at the same time in a batch (e.g. by offering a more favourable price of service per request in an application scenario). Note that with such an extension, we are still in the non-adaptive FE scenario.

A further possibility for a natural extension within the NA setting is when different users are served synchronously (in parallel) by the database, where the requests from different users are collected into a single request message. The formal definition of the parallelized version of ideal functionality F_{rCPFE} is the same as the base functionality except for a few straightforward differences. The technical details of the analysis for the parallel version are essentially the same as for the base case.

2. Multiple session extension

2.1. Single database

The next natural step of extension is a multiple session scenario, where multiple users use the same database multiple times and we assume a single, stand-alone database. An arbitrary number of requests (to the database items and to the functions) may be sent from arbitrarily different users to the database. Such a scenario implies an adaptive scenario for the FE encryption, accordingly, it requires an AD-SIM FE scheme ([Matt and Maurer, 2015]). Such a scheme can be implemented only in a non-standard setting with the assumption of a programmable random oracle model. Such an oracle is a trusted external party running a truly random function for the parties of the protocol. It is a theoretical concept.

However, if we accept the Fiat-Shamir paradigm we can implement a non-provably secure protocol efficiently by modeling a hash function as a random oracle. It is based on the belief that a secure hash function is a practical (efficient) implementation of a pseudorandom function.

In the proof of security in the RO model, the simulator simulates (also) the corresponding F_{RO} ideal functionality, which enables the simulator to learn the queries of all involved parties and to program any "random-looking" (i.e. computationally indistinguishable from truly random) values as outputs.

2.2. Multiple databases

The definition of multi-session ideal functionality is very similar to the single-session functionality except for a few differences. The difference is that by the multi-session functionality F'_{rCPFE}, the database can be queried an arbitrary number of times by an arbitrary number of users under the same main session identifier sid but under different fresh sub-session identifiers (ssid). Intuitively, the (main) session id value sid=(S, sid') corresponds to a fresh FE key pair (PK, MK) generated by server S. We assume that a new pair of such keys is generated for a new database. Sub-session identifier ssid=(U, S, ssid') corresponds to the ssid'-th instance of the functionality run by parties U and S. A natural choice is to bind ssid to a fresh instance of communication keys (i.e. to a fresh instance of secret channel) between U

and S. (From space-saving reasons we omit the presentation of the definition of this ideal functionality.)

4. IT-secure constructions

In this section, we adapt two different generic approaches to our task. Both methods use correlated randomness technique and we apply them for the derived database. One of them ([Araki et.al., 2016], [Furukawa et.al., 2023]) requires an honest majority in a set of auxiliary servers and achieves IT security. We apply these constructions as subprotocols in Construction 1 below. In the other approach ([Ishai et.al., 2013]), no honest majority is assumed. In this case, secure function evaluation is carried out by a two-party protocol with correlated randomness setup. This protocol provides perfect security. We adapt this technique to our task in Construction 2 below.

IT security implies high (or extremely high) complexity cost, in general. A central question in multiparty computation is to understand the amount of communication needed to securely evaluate a circuit of size s, with the obvious aim of minimizing communication complexity. However, in the case of IT-secure designs, first of all, the high randomness complexity (the number of required genuine random bits) is the bottleneck, as we demonstrate below with numerical examples.

A fixed cost element of IT-secure protocols comes from the requirement of perfect secure communication channels. Such channels can be implemented using one-time pad encryption together with IT-secure authentication using a one-time message authentication code (MAC). The total randomness complexity consists of the amount of random bits used in the computation plus the amount used in communication.

4.1 Construction 1

Here we apply the IT-secure versions of the generic approach [Araki et.al., 2016], [Furukawa et.al., 2023] to our task, in particular to the evaluation of the circuit corresponding to the 1-out-of-N, $N=|I|\cdot|X|$ ($m=1$) oblivious transfer functionality. Using the terminology of Section 3, here we consider a one-phase approach. The database server and the user outsource this computation to a set of three auxiliary servers (SMC-servers). Database server distributes random shares of the derived database between the SMC-servers. User does the same for its chooser string. Finally, the user processes the outputs of the SMC-servers.

The perfect secure construction [Araki et.al., 2016] requires an honest majority in the set of SMC servers (2 honest servers + (at most) 1 semi-honest server). In the statistically secure version ([Furukawa et.al., 2023]) the dishonest server can be malicious.

In these constructions, for the evaluation of an AND gate, the servers generate a triplet of correlated random bits (x_0, x_1, x_2) such that $x_0+x_1+x_2=0$. The triplet is generated as follows: party P_i chooses a random bit b_i and sends it to party $P_{(i+1) \bmod 3}$, $i=0,1,2$. Party P_i computes x_i by adding b_i and the received bit mod 2. Consequently, an estimate of the number of random bits used in the computation (of the SMC servers) is $3\cdot mN\log(N)$.

For instance, for the task with parameter values $|X|=2^{15}$, $|I|=2^6$, $m=2^7$ execution of the protocol would require $21\cdot 2^{28}$ truly random bits per SMC-server. Considering random bit generation with a speed of 1Gbps per SMC-server we get ~ 5 second latency.

The statistically secure version secure against an active adversary consumes considerably more random bits [Furukawa et.al., 2023]. The driving force behind the high randomness complexity is the "wasteful" statistical algorithm (cut-and-choose algorithm) used for forcing semi-honest behavior in the computation of the AND gates. Therefore, it is suitable only for appropriately small database problems. More correctly, we should partition the database to such an extent that the size of a partition already enables efficient implementation. Of course, we can only do this at the expense of increasing privacy error.

4.2 Construction 2

In the previous section, a correlated triplet was used to securely evaluate an AND gate (small parts of the whole circuit) and such triplets were generated and used by the SMC-servers evaluating the circuit. Now, a setup server provides correlated randomness setup for the user and the server. We adapt construction technique from [Ishai et.al., 2013].

A brief outline of the protocol follows. For easier explanation, consider the ideal case when a single fully trusted auxiliary server (S*) is available. The execution of the protocol is divided into two stages, the preprocessing and the online stages. We represent the derived database as a matrix where rows correspond to different database items and columns to different permitted functions. The setup server computes a one-time permutation of this matrix. Given that a permutation will be only used once, a random cyclic shifts can be used (row-wise and column-wise) instead of a random permutation.

The corresponding protocol is denoted π_0 and shown in Fig. 5. Parties (user and data server) receive shares of the permuted matrix from the setup server. Additionally, the user receives the permutation applied by rows and columns. In the online phase, the user reconstructs both shares of the wanted (derived) database item.

Functionality:
– Parties: S, U and auxiliary server S*
– Server (S) has input matrix D' (derived database) .
– User (U) has input $x \in X$ (index of a data item) and $y \in Y$ (index of a function).
– U learns $z = D'(x, y)$.

Pre-processing: (performed by S*)
1. Sample random $r \in X$, $s \in Y$ and let D* be the permutation of D':
$D^*(x+r, y+s) = D'(x, y)$.
2. Sample a random matrix $M_1 \in Z^{X \times Y}$ and let $M_2 = D^* - M_1$.
3. Output (M_1, s, r) to U and M_2 to S.

Protocol:
1. U sends $u = x + r$, $v = y + s$ to S
2. S sends $M_2(u,v)$ to U
3. U outputs sum $M_1(u,v) + M_2(u,v)$

Fig. 5: Protocol π_0

Now we make a weaker assumption, by assuming the availability of a pair of auxiliary servers (S1*, S2*) such that one of these may be corrupted by a semi-honest adversary. The main steps of the corresponding protocol (π_1) are shown in Fig. 6.

1. Server S distributes shares D'1, D'2 of database D' between servers S1* and S2*,
2. Servers S1* and S2* perform the pre-processing phase with input D'1 and D'2, respectively, similarly to protocol π_0.
3. Parties S and U execute the online phase of protocol π_0 twice, first with S1* and next S2*,
4. Finally U combines (adds) the outputs received from these two executions.

Fig. 6: Main steps of protocol π_1

Protocol π_1 guarantees privacy for both parties. In the next extension, we allow one of the parties (database server, user) may maliciously execute the protocol. The construction follows the technique of MAC-enhanced OTTT in [Ishai et.al., 2013]. Accordingly, a pair of authentication tag matrices is computed by the setup servers. A one-time ε-secure MAC scheme is assumed (see Definition 1 in [Ishai et.al., 2013]). The modified protocol π_0 is denoted π_{0*} and is shown in Fig. 7. Now, protocol π_{0*} is used in protocol π_1 instead of π_0.

Preprocessing:
1. Sample random keys for one-time ε-secure MAC scheme with key space K and MAC space W. Choose randomly a pair of keys k_1, k_2 from space K and sample random $r \in X$, $s \in Y$. Let matrix $D^* \in (ZxWxW)^{XxY}$ be such that

$D^*(x+r, y+s) = \{D'(x, y) , Tag_{k1}(D'(x, y)), Tag_{k1}(D'(x, y))\}$,

2. Sample a random matrix $M_1 \in (ZxWxW)^{XxY}$ and let $M_2 = D^* - M_1$
3. Output (M_1, r, s, k_1) to U and (M_2, k_2) to S.

Protocol:
1. U sends $u = x + r$, $v = y + s$ to S
2. S sends $M_2(u,v)$ to U
3. Each party parses $M_1(u,v) + M_2(u,v)$ as (z, t_1, t_2)
4. If $Ver_{k1}(z, t_1) = 1$, party U outputs z, otherwise it outputs $\perp$
   If $Ver_{k2}(z, t_2) = 0$, party S outputs "malicious user"

Fig. 7: Protocol π_{0*}

This protocol guarantees perfect security. The randomness complexity is determined by the sampling of random matrix M_1, which means $(m+2t)|X||I|$ random bits, where t is the length of MAC. For instance, for concrete parameter values $|X|=2^{15}$, $|I|=2^6$, $m=2^7$, $t=64$ this means 2^{29} random bits per setup server. Considering random bit generation with a speed of 1Gbps per SMC-server we get ~ 0.5 second latency. The communication randomness complexity is of the same order of magnitude.

5. Discussion

Perfect security may imply UC security. In particular, the perfect secure constructions in Chapter 4. are UC-secure since those are based on protocols shown as UC-secure in papers [Araki et.al., 2016], [Ishai et.al., 2013]. [Furukawa et.al., 2023]. In particular, these latter protocols were proven secure with a black-box non-rewinding simulator in the classic stand-alone setting and since the inputs of all parties are fixed before the execution begins, such a proof automatically guarantees UC security.

Now we consider the computationally secure construction based on functional encryption (Sec. 3.3). The two computationally secure primitives used in the construction must be UC-secure. This means a fully adaptive AD-SIM-secure FE scheme [Matt, Maurer; 2015] and a UC-secure OT protocol. Both these components require a trusted setup. As usual, the difficulty comes primarily from the realization of the setup.

First, we consider the setup for the FE primitive. According to the ideal functionality defined in Section 3.3. different databases correspond to different sid values, i.e. different databases have their own independent random oracle instances that serve all user-server sessions with identifier (sid, ssid) for some ssid. Following the (ad-hoc) logic an assumption of independent random oracle instance per protocol instance (with identifier sid) is intuitively impractical as a large number of different secure hash functions do not exist. It seems that the only solution is to assume that different instances of the protocol have shared access to a global setup functionality, i.e. we consider global RO models. In this case, only a single secure hash function is required for an ad-hoc implementation of the global random oracle.

The question that arises here is whether it is possible to use a global oracle, serving different service providers, or a service provider with different databases. Here we face a proof-technical obstacle. Namely, in the GUC framework of Canetti, the (calling and distinguishing) environment also has direct access to the global oracle and this fact has serious consequences: the global version of (Canetti's) F_{RO} functionality is the strictest among all RO models. Indeed, the simulator is neither allowed to observe the environment's random-oracle queries nor to program its answers. Therefore, to give an advantage to the simulator seems impossible. However, it turned out (see [Camenisch et.al., 2018]) that even such strict functionality allows GUC-secure practical constructions for digital signatures and public-key encryption. The breakthrough observation was (see the details in [Camenisch et.al., 2018]) that the (classic) proof technique of rewinding (of the oracle) can be applied in the part of the proof about indistinguishability, instead of within the simulation algorithm (where the rewinding tool is usually applied). Our point here is that we can apply the mentioned proof technique from [Camenisch et.al., 2018] also in our task:

Claim 2: *There exists a global universally composable (GUC-secure) adaptively secure FE scheme in a programmable random GUC-oracle setting.*

Proof (Sketch): Boneh's AD-Sim-secure brute force construction [Boneh et.al., 2011] is GUC-secure in a programmable random GUC-oracle model defined in [Camenisch et.al., 2018]. To prove it a proof technique in Ch. 4.2 of [Camenisch et.al., 2018]) can be adapted. ■

Now we consider the oblivious transfer primitive. In the paper [Choi et.al., 2014] an efficient, universally composable oblivious transfer (OT) was proposed, where a single, 'global', common reference string (CRS) can be used for multiple invocations of oblivious transfer by arbitrary pairs of parties. It is also round-optimal (3 rounds) with static security. A complexity comparison of UC-secure OT constructions can also be found in Table 1. [Choi et.al., 2014].

The realization of the setups is most straightforward if the parties can find an external trusted party for this purpose. However, if the interests of the parties conflict, this is rarely the case. A more realistic scenario is that the setup is based on a set of external parties (servers) with an honest majority (like the auxiliary servers in the constructions of this paper).

6. Conclusions

A summary comparision of the pros and cons of the constructions is shown if Table 1.

Protocol	Type of security	Pros	Cons
Sec.3.1	computational	conceptually simple no aux. servers	large increase in database size
Sec.3.2	computational	balanced complexity (potential quantum-safe version)	3 aux. servers with honest majority
Sec.3.3	computational	balanced complexity no aux. servers	restricted set of functions non-adaptive scenarios
Sec.4.1	IT	quantum-safety low comm. complexity between aux. servers	3 aux. servers with honest majority high randomness complexity
Sec.4.2	IT	quantum-safety conceptually simple low online complexity	a pair of setup servers high randomness complexity

Table 1.

In this paper, we showed several constructions for the task of Controlled Private Function Evaluation of obliviously chosen database items. The practical efficiency of the constructions highly depends on the size of the task measured in the triplet of the number of database items, the bit-length of the items, and the number of permitted functions. The complexity problem can be mitigated by partitioning the database. Accordingly, the fourth parameter affecting the practical applicability is the tolerated level of privacy error.

We considered both computationally and information-theoretically secure solutions. Though IT-secure solution have typically very high randomness complexity, interestingly, even in the complex task we consider in this paper, even such level of security can be reached at realistic parameter values. Of course, this requires truly random generators with very high speeds (in our examples with the speed of 1Gbps). Furthermore, it is required that a set of three auxiliary servers with honest majority or a pair of setup servers with at most one is semi-honest is available. At the same time, at the cost of this increase in complexity, we "automatically" get a postquantum secure protocol.

These perfect secure solutions are also UC-secure, i.e. guaranteeing secure general composability. As for the computational secure solutions, we examined UC-security for one of the computational constructions that is based on functional encryption (FE) primitive. However, in this case, in contrast to perfect secure constructions, a UC-secure FE-based construction cannot be implemented in the standard setting, since it requires access to programmable random oracles.

6. References

[1] Abdalla, M., Bourse, F., De Caro, A. and Pointcheval, D., "Simple Functional Encryption Schemes for Inner Products", PKC 2015: Public-Key Cryptography, pp. 733–751.
[2] T. Araki, J. Furukawa, Y. Lindell, A. Nof and K. Ohara, "High-Throughput Semi-Honest Secure Three-Party Computation with an Honest Majority", 23rd ACM CCS, 2016, pp. 805–817.
[3] Beaver, D., "Precomputing Oblivious Transfer", In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, Springer, Heidelberg (1995), pp. 97–109.
[4] Boneh, D., Sahai, A. and Waters, B.,"Functional Encryption: Definitions and Challenges", Proceedings of Theory of Cryptography Conference (TCC) 2011, pp 253-273.
[5] Canetti, R., "Universally Composable Security: A New Paradigm for Cryptographic Protocols", In 34th STOC 2002, pp. 494-503.
[6] Canetti, R. et. al.,"Universally composable two-party and multi-party secure computation", In 34th ACM STOC, Montreal, Quebec, Canada, ACM Press, pp. 494–503.
[7] Camenisch, J., Dubovitskaya, M. and Neven, G., "Oblivious Transfer with Access Control", 16th ACM Conference on Computer and Communications Security (ACM CCS 2009), pp. 131-140.
[8] Camenisch, J. et al., "The Wonderful World of Global Random Oracles", in Advances in Cryptology – EUROCRYPT 2018, pp 280-312.
[9] Cattaneo, G. et. al., "The European data market monitoring tool, Key facts & figures, first policy conclusions, data landscape and quantified stories: d2.9 final study report". Publications Office of the EU, July 2020.
[10] Choi, S.G. et. al., "Efficient, Adaptively Secure, and Composable Oblivious Transfer with a Single, Global CRS", https://eprint.iacr.org/2012/700.pdf
[11] Elisabetta, C., Baltico, Z., Catalano, D., Fiore, D. and Romain G., "Practical Functional Encryption for Quadratic Functions with Applications to Predicate Encryption", July 2017, Annual International Cryptology Conference
[12] Furukawa, J., Lindell, Y, Nof, A. and Weinstein, O., "High-Throughput Secure Three-Party Computation for Malicious Adversaries and an Honest Majority", Journal of Cryptology, Volume 36, Issue 3, Jul 2023
[13] Gorbunov, S., Vaikuntanathan, V. and Wee, H. (2012), "Functional Encryption with Bounded Collusions via Multi-Party Computation", Advances in Cryptology – CRYPTO 2012, pp 162-179.
[14] Horvath, M. et. al., "There Is Always an Exception: Controlling Partial Information Leakage in Secure Computation", Cryptology ePrint Archive: Report 2019/1302
[15] Ishai,Y., Kushilevitz, E., Meldgaard, S., Orlandi, C. and Paskin-Cherniavsky, A., "On the Power of Correlated Randomness in Secure Computation", A. Sahai (Ed.): TCC 2013, LNCS 7785, pp. 600–620.
[16] Matt, C. and Maurer, U., "A definitional framework for functional encryption". In IEEE 28th Computer Security Foundations Symposium, CSF 2015, Verona, Italy, 13-17 July, 2015, pp. 217–231.
[17] Naor, M., B. Pinkas, B., "Oblivious transfer and polynomial evaluation", In Proceedings of the 31st ACM Symposium on Theory of Computing, 1999, pp.145-254.
[18] Otto, B. et. al., "Data ecosystems. Conceptual foundations, constituents and recommendations for action", Technical report, Fraunhofer Institute for Software and Systems Engineering ISST, 10 2019.