This report is superseded by a revised and significantly improved version, published as:
Máté Horváth, Levente Buttyán, Gábor Székely and Dóra Neubrandt. There Is Always an Exception: Controlling Partial Information Leakage in Secure Computation. *J. H. Seo (Ed.): Information Security and Cryptology – ICISC 2019, LNCS 11975, pp. 1-17, 2020.*
For the full version, see `https://eprint.iacr.org/2019/1302`

Controlled Private Function Evaluation from Functional Encryption*

(Technical Report)

Máté Horváth and Levente Buttyán

2018

Laboratory of Cryptography and Systems Security (CrySyS)
Department of Networked Systems and Services
Budapest University of Technology and Economics
{mhorvath,buttyan}@crysys.hu

Abstract. Two-party Private Function Evaluation (PFE) enables the participants to jointly execute a computation for which one of them provides the function and the other one the input to the function. According to the traditional security requirements, a PFE protocol should not leak more information, neither about the function nor the input, than what is revealed by the output of the computation. We observe that the function privacy requirement, in fact, makes input privacy meaningless as it allows for the unnoticeable evaluation of the identity function, disclosing the input entirely.
In this work, we ask the question whether it is possible to achieve a reasonable level of input and function privacy simultaneously in PFE. We propose the notion of Controlled Private Function Evaluation (CPFE) and answer the question affirmatively by showing a simple, generic realisation of CPFE based on Functional Encryption. To demonstrate the applicability of our approach, we show a concrete instantiation of our protocol for inner product computation that enables secure statistical analysis (and more) under the standard Decisional Diffie-Hellman assumption in the random oracle model.

Keywords: Cryptographic Protocols · Private Function Evaluation · Functional Encryption · Oblivious Transfer · Data Markets

1 Introduction

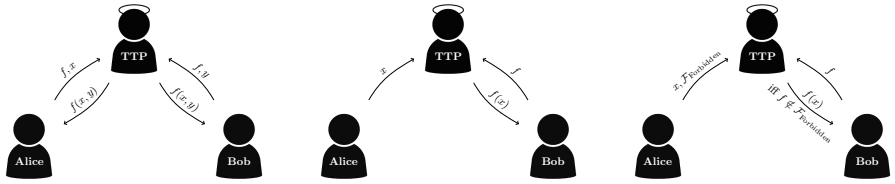
Eliminating the need for trust when two parties are willing to execute some computation jointly is one of the most studied areas of cryptography since the

* This work was partially performed in the frames of the FIEK_16-1-2016-0007 project and the project no. 2017-1.3.1-VKE-2017-00042, implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the FIEK_16 and the 2017-1.3. funding schemes, and it was also partially supported by the National Research, Development and Innovation Office NKFIH, under grant contract no. 116675 (K).

seminal work of Yao [34] from the 1980s. Secure function evaluation (SFE) or secure two-party computation (2PC) protocols enable two parties, Alice and Bob, to compute a function of their private inputs without disclosing their secrets to each other and without needing the help of a trusted third party (see Fig. 1a). In real life, however, the participants not necessarily have interchangeable roles. It is often the case that only one of them, say, Alice contributes to the computation with input and Bob chooses the function to be computed. Due to various reasons, such as intellectual property protection, this function is often should remain hidden from the other party, which we call the function privacy requirement. Primitives that satisfy this need are called private function evaluation (PFE) protocols that can be derived from SFE with the help of universal functions [33]. A universal function is a "programmable function" that can implement *any* computation up to a given complexity. It takes two inputs, the description of the function to be computed and the input to it. Evaluating a public universal function using SFE allows Bob to evaluate his confidential function on the private input of Alice enabling the privacy-preserving solution of various problems like data classification or remote diagnostics.

As using universal functions, all feasibility results extend from SFE to PFE; one could think that the only remaining challenge in the field of PFE is to improve efficiency. Indeed, universal functions cause a significant–for complex computations even prohibitive–overhead, and the elimination of this limitation was the primary focus of PFE research [21, 19]. At the same time, a possible vulnerability, rooted in the nature of PFE, received no attention to the best of our knowledge. As PFE guarantees Bob that his function is hidden from Alice, he can exploit this property to find out the input of Alice by computing the identity function. Note that this opportunity is left open by PFE, so it does not harm any rules of the protocols but still, clearly contradicts the security guarantees that Alice wishes to obtain. One could interpose that such behaviour can be easily detected and thus the protocol can be aborted by Alice whenever she can have access to the result of the computation. Of course, identity is not the only function that leaks its input but every invertible function has this property and invertibility of an unknown function cannot be ruled out based on a single input-output pair. This observation shows that function privacy prevents guaranteed input privacy even though formally both can be achieved. This contradictory situation is caused by a common misinterpretation of security arguments of PFE protocols. These do not claim that the protocol hides the function from Alice and the input from Bob, but only prove that *PFE hides the function from Alice and does not reveal more about the input to Bob then that is revealed by the secret function chosen by himself.* This interpretation is fair in the sense that it does not provide a false sense of security to Alice but highlights her defencelessness against an untrusted party. We are not aware of any works that emphasise this interpretation, so the primary goal of our work is to raise attention and initiate the study of the following question.

Is there a meaningful compromise between the input and function privacy that leads to a PFE notion that indeed guarantees security for both parties?



(a) SFE or 2PC (b) PFE (c) CPFE

Fig. 1: Comparison of the *goals* of different concepts for secure function evaluation, realised with the help of a trusted third party (TTP). The key difference lies in which information Alice and Bob can or cannot have access to.

We give a positive answer to the above question by proposing a concept that we call Controlled Private Function Evaluation (CPFE). The idea is to give Alice control over the functions, computable on her data, while still maintaining a reasonable extent of function privacy. See Fig. 1c for an ideal target for future CPFE constructions that do not have access to trusted third parties. In this work, we put forward a relaxed notion, which we call 1 out of n CPFE (or CPFE_1^n for short). In a CPFE_1^n protocol, Bob has to send n function descriptions for verification to Alice that, among dummy ones, contain the one he wishes to compute. If Alice finds that the received functions are acceptable to her, she enables Bob to evaluate one of the functions on her data without revealing it. To make the concept meaningful, it is crucial that the truly computed function must be irreversibly chosen by Bob in the first step; and that this choice must remain hidden from Alice. Towards realising CPFE_1^n , our main tools are oblivious transfer (OT) and functional encryption (FE). The latter one is a generalisation of traditional encryption schemes that integrates decryption with the evaluation of a function [6] so that decryption does not reveal the encrypted value, but a specific function of it.

1.1 Our Contributions

In more details, our contributions can be summarised as follows.

- We initiate the study of the relation between data and function privacy in the context of private function evaluation.
- To take the first step, we formally define 1 out of n controlled private function evaluation and its indistinguishability-based (IND) security.
- Then we show a conceptually simple, generic realisation for CPFE_1^n building on functional encryption and prove its security. Besides providing the above-described transparent security guarantees for both parties, our solution enjoys the desirable reusability property as well. The computation of the same function of multiple, say j inputs (or k different functions of the same input) can be optimised so that the overhead is not j (or k) times more

but scales with $O(j+n|f|)$ (or $O(1+kn|f|)$), where $|f|$ denotes function size. Finally, it entirely hides the result of the computation from Alice.
– Besides the generic construction, we examine a concrete instantiation of the generic scheme that enables statistical analysis via the controlled but private evaluation of inner products. The resulting scheme is proven secure under the standard Decisional Diffie-Hellman assumption in the random oracle model.

1.2 An Application: Secure IoT Data Markets

While transparent security guarantees are necessary for all applications of PFE, we highlight one, where our CPFE is especially useful.

An emerging idea is to monetise IoT data through data markets [22, 15]. In these markets, data brokers bridge the gap between owners of data, generated by smart devices, and so-called value-added service providers (VASP). The goal of VASPs is to provide useful insights and help optimisation, e.g. via statistical analysis or through building machine learning models. These require a vast amount of sensor data that can be bought through a data market more easily, than via the troublesome recruitment of data owners for participation. Probably the main reason why such markets are still not flourishing is the issue of security.

Let us imagine the following motivating scenario. In a transportation company, various information is collected from the trucks to monitor and optimise the logistics operation. The recorded data might reveal different habits of drivers or policies inside the company. Nonetheless, this is not a problem as long as data are collected and processed solely inside the firm. However, extra profit can motivate the selling of such data. Moreover, a transportation company may have limited competence in handling and processing all these information so it can also be advantageous to outsource these tasks leading to the data market ecosystem, depicted in Fig. 2. Considering this latter scenario, the company can operate a data broker that is trusted by the devices collecting the data and which can trade with VASPs. In their interaction, it is reasonable to assume semi-honest (i.e. honest but curious) behaviour from these parties as the broker (representing the company) is curious about the methods of the VASPs and also about the computation results, while the interest of VASPs is to obtain as much data as

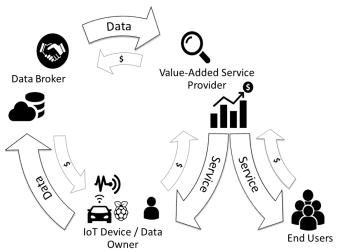


Fig. 2: System model for data markets

possible. These goals together with the nature of the involved data and its usage require a protocol that

- allows the broker to control both the scope of accessible inputs and the sets of computable functions for a given price, enabling fine-grained pricing like "pay as you compute",
- gives the VASPs opportunity to preserve the intellectual property of their methods (e.g. parameter choices in statistics or machine learning models, etc.),
- enables efficient computation of the same function (e.g. statistics) on varying inputs.

These requirements are all fulfilled by CPFE, and specifically for certain statistical applications by our 1 out of n controlled private inner product evaluation protocol.

1.3 Related Work

Two-party PFE was first considered in [1]. The reduction of PFE to SFE [35, 23] via utilising universal functions [31] faces serious efficiency issues due to the overhead, caused by the usage of universal circuits (UC) [33]. To make the concept applicable, several works dealt with making UCs more practical [29, 21, 30, 19, 14]. An alternative approach for realising PFE without UCs uses homomorphic encryption [16, 24, 25] but regardless of the methods, all the mentioned works aim to achieve the same goal represented in Fig. 1b.

At the same time, some PFE variants are closer to the concept our CPFE then plain PFE. The idea of semi-private function evaluation (semi-PFE) [27, 18] can be viewed as a possible tool for our goals, because semi-PFE relaxes the function privacy requirement of PFE and reveals the function class but not the concrete function that is evaluated. More precisely, in this concept, a function can be viewed as a composition of "privately programmable blocks" disclosing the topology but not the content of the blocks. While this relaxation provides a useful trade-off between function privacy and efficiency, unfortunately, the available extra information about the function does not necessarily allow Alice to rule out the evaluation of functions that are against her interest. To see this, observe that even the identity function can be constructed so that its topology becomes complex and unrecognisable.

Selective private function evaluation (SPFE) [7] deals with a problem that is orthogonal to the one considered in this paper. Namely, SPFE also aims to conceal information that is leaked in PFE. But instead of protecting Alice (the data owner), it intends to increase the security of Bob by hiding from Alice the location of the function's input in her database via using private information retrieval (PIR).

Leaving the field of PFE and comparing our work to related problems in secure computation, we find that hiding the computed function raises similar issues in other contexts. [4] put forth the notion of verifiable obfuscation that

is motivated by the natural fear for executing unknown programs. The goal here is similar than in our setting: some assurance is required that the hidden functionality cannot be arbitrary. However, the fundamental difference between our CPFE and the verifiable obfuscation and verifiable FE of [4] is that while the latter ones enforce correctness when an obfuscator or authority may be dishonest, CPFE protects against parties that honestly follow the protocol but exploit its limitations.

Our CPFE is built upon functional encryption (FE) in a black-box manner. This generalisation of traditional encryption was first formalised by [6] which work also showed impossibility results for realising simulation secure FE, that were further extended later [8]. Several, in some sense, restricted variants of FE were realised under standard assumptions (e.g. [13, 2, 5]) but general purpose FE candidates [10, 11] currently rely on untested assumptions like the existence of indistinguishability obfuscation or multilinear maps. The possible functionalities that our CPFE can support are necessarily tied to the state of the art of FE. Nevertheless, the rapid development of this field is promising for our application as well. In the context of FE, [26] raised the question of controllability of function evaluation. The essential difference, compared to our goals, is that they want to limit repeated evaluations of the *same* function[1] that they solve with the involvement of a third party.

Finally, we sum up the state of the art of private inner product evaluation. The provably secure solutions are built on partially homomorphic encryption schemes [12, 9] and public-key inner product FE [2] is also capable of the same task. At the same time, several ad-hoc protocols achieve better performance in exchange for some information leakage (see e.g. [36] and the references therein), but these constructions lack any formal security argument.

1.4 Organisation

The rest of the paper is organised as follows. After introducing the assumptions and the necessary background in Section 2, we formally define Controlled-SFE and its security in Section 3. We describe our generic 1 out of n Controlled PFE protocol in Section 4 and prove its security. Section 5 is dedicated to an instantiation of our solution and to a brief analysis of it. Finally, we conclude our work in Section 6.

2 Preliminaries

In this section, we briefly summarise the relevant background for the rest of the paper.

[1] In FE schemes, the control over the computable functions is in the hand of the master secret key holder so this is not an issue unlike in PFE.

2.1 Assumptions

Let $\mathbb{G}$ be a multiplicative group of prime order p, generated by $g \in \mathbb{G}$. In this paper we are going to consider three hardness assumption in $\mathbb{G}$. The first is the discrete logarithm (Dlog) assumption which asserts that it is hard to find $x \in \mathbb{Z}_p^*$ given only g^x. A stronger assumption, the Computational Diffie-Hellman (CDH) assumption, states that given $g^x, g^y \in \mathbb{G}$, for randomly chosen $x, y \in \mathbb{Z}_p^*$, it is hard to compute g^{xy}. Finally a widely accepted and used assumption states that not only the computation of g^{xy} is hard, but also its recognition.

Assumption 1 (DDH) *Let $g \in \mathbb{G}$ be a generator element of the group and $x, y \in \mathbb{Z}_p^*$ are uniformly random values. We say that the Decisional Diffie-Hellman (DDH) assumption holds in $\mathbb{G}$ if given (g, g^x, g^y, g^r), no probabilistic polynomial time (PPT) algorithm can decide, with higher than $\frac{1}{2} + \mathsf{negl}(p)$ probability, whether $r = xy$ or r is also a uniformly random value from $\mathbb{Z}_p^*$.*

Note that the DDH assumption implies all the previous ones.

In our second security analysis, we are going to use the random oracle model (ROM) [20]. This model assumes that truly random values, provided by a so-called "random oracle", can be securely substituted with the outputs of a concrete hash function.

2.2 Commitment Schemes

A commitment scheme is the digital analogue of a locked box in which someone can place a value or choice that he or she is committed to, and which can be revealed by opening the box. A commitment scheme consists three algorithms. Upon a security parameter λ, $\mathsf{C.Setup}$ outputs the public commitment key ck. $\mathsf{C.Commit}(\mathsf{ck}, m)$ transforms m into a pair (c, d), a commitment and an opening value respectively and finally, $\mathsf{C.Open}(\mathsf{ck}, \mathsf{c}, \mathsf{d})$ reveals m. The correctness requirement of a commitment scheme is that $\mathsf{C.Open}(\mathsf{ck}, \mathsf{C.Commit}(\mathsf{ck}, m)) = m$ must hold. In terms of security the following – informally stated – requirements have to be fulfilled.

Hiding. The commitment value c should not reveal information about the underlying value m i.e., c_{m_0} and c_{m_1} must be indistinguishable.

Binding. No commitment value c can have multiple ways to open it i.e., it must be computationally infeasible to find two pairs $(\mathsf{c}, \mathsf{d}), (\mathsf{c}, \mathsf{d}')$ such that $\mathsf{C.Open}(\mathsf{ck}, \mathsf{c}, \mathsf{d}) = m, \mathsf{C.Open}(\mathsf{ck}, \mathsf{c}, \mathsf{d}') = m'$ and $m \neq m', \mathsf{d} \neq \mathsf{d}'$.

In our protocol for inner product computation, we are going to use Pedersen commitments [28] the security of which is based on the discrete logarithm problem.

2.3 Oblivious Transfer

1 out of n oblivious transfer (OT_1^n) enables transferring data between two parties, the sender (S) and the receiver (R, a.k.a. chooser), in a way that protects

both of them. The sender can be sure that the receiver does not obtain more than one piece of information while the receiver is assured that the sender does not know which of the n pieces of information it received (chose). OT_1^n is comprised of four algorithms. $\mathsf{OT.Setup}$ is either run by the sender or a trusted third party (TTP) to generate system parameters, $\mathsf{OT.Commit}$ is run by the receiver to choose the piece of information that it wants to retrieve, $\mathsf{OT.Trans}$ is executed by the sender to produce the transferred values from its n messages and finally, the chosen message is retrieved by the receiver using $\mathsf{OT.Retr}$.

As we are going to use the OT_1^n scheme of [32] against a malicious receiver, for completeness we recall the requirements of security as stated there (the following enumeration is taken verbatim from [32]).

Correctness. The protocol achieves its goal if both R and S follow the protocol step by step, R gets m_α after executing the protocol with S, where α is the choice of R.

Receiver's privacy (indistinguishability). The transcripts corresponding to R's different choices α and α', $\alpha \neq \alpha'$, are computationally indistinguishable to S. If the transcripts are identically distributed, the choice of R is unconditionally secure.

Sender's privacy (compared with ideal model). In the ideal model, a TTP acts as an intermediary agent who receives S's secrets $m_1, m_2, \ldots, m_n$ and R's choice α and gives m_α to R. Since R has no way of getting information other than m_α, this model is considered the most secure way to implement OT. Therefore, we say that the sender's privacy is guaranteed if, for every possible malicious R which interacts with S, there is a PPT simulator R' which interacts with the TTP such that the output of R' is computationally indistinguishable from the output of R.

2.4 Functional Encryption

As we already introduced, FE is a generalised encryption scheme that enables certain computations on hidden data for authorised parties. Both public- and secret-key variants are known, but here we limit ourselves to the secret-key setting that suffices for our purposes. An sk-FE scheme consists of the following four algorithms.

$\mathsf{FE.Setup}(\lambda) \to (\mathsf{msk_{FE}}, \mathsf{pp_{FE}})$ The setup algorithm takes in a security parameter λ and produces the public system parameters $\mathsf{pp_{FE}}$ and a master secret key $\mathsf{msk_{FE}}$.

$\mathsf{FE.Enc}(\mathsf{msk_{FE}}, x) \to \mathsf{ct}$ The encryption algorithm takes the master secret key $\mathsf{msk_{FE}}$ and a message x and outputs a ciphertext ct.

$\mathsf{FE.KeyGen}(\mathsf{msk_{FE}}, f) \to \mathsf{fsk_f}$ The key generation algorithm can be used to generate a functional secret key $\mathsf{fsk_f}$ for a function f with the help of the $\mathsf{msk_{FE}}$.

$\mathsf{FE.Dec}(\mathsf{ct}, \mathsf{fsk_f}) \to y$ Having a functional secret key $\mathsf{fsk_f}$ (for function f) and a ciphertext ct (corresponding to x), the decryption algorithm outputs the value y.

The correctness of FE requires that if $\mathsf{fsk_f}$ and ct were indeed generated with the corresponding algorithms using inputs f and x respectively, then $y = f(x)$ must hold. Regarding security, in this work we are going to use the subsequent indistinguishability-based definition following [6].

Definition 1 (s-IND-CPA security for FE). *The security of FE is defined via the following game between a challenger $\mathcal{C}$ and a probabilistic polynomial time (PPT) adversary $\mathcal{A}$.*

- *Initialise: $\mathcal{A}$ sends a challenge message pair (x_0, x_1) to $\mathcal{C}$ who computes $\mathsf{msk_{FE}} \leftarrow_{\$} \mathsf{FE.Setup}(\lambda)$.*
- *Query: $\mathcal{A}$ adaptively submits queries to obtain $\mathsf{fsk_f}$ that is generated by $\mathcal{C}$ using $\mathsf{FE.KeyGen}(f, \mathsf{msk_{FE}})$ as long as $f(x_0) = f(x_1)$ holds.*
- *Challenge: Once $\mathcal{A}$ decides to finish the query phase, $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$ and computes $\mathsf{FE.Enc}(\mathsf{msk_{FE}}, x_b) = \mathsf{ct}_b$ and sends the result to $\mathcal{A}$, that has to output a guess b' for the bit b.*

We say that an FE scheme achieves selective security against chosen plaintext attacks (s-IND-CPA) if $\Pr(b' = b) < \frac{1}{2} + \mathsf{negl}(\lambda)$.

3 Algorithms and Security of 1 out of n Controlled PFE

In this part, we introduce the algorithms of a CPFE_1^n scheme and provide a formal security model for it.

3.1 Formal Specification

1 out of n Controlled Private Function Evaluation is an interactive protocol between a data and the function providers, say Alice and Bob (or between the data broker and a VASP in the IoT data market context). The run of the protocol can be split into sessions such that, within the same session, each requested (and accepted) functions can be evaluated on any of the values, received upon a data request. The sessions do not have to follow each other sequentially, but different ones can be run in parallel by using different session parameters $\mathsf{sk_A}, \mathsf{pp}$ obtained from the $\mathsf{SessionSetup}$ algorithm. This reusability property leads to performance improvements in use-cases, where either the same data or function is used more than once. The randomised algorithms of CPFE_1^n are as follows, while a session of the protocol (with a single data and function query) is depicted in Fig. 3.

$\mathsf{SessionSetup}(\lambda) \rightarrow (\mathsf{sk_A}, \mathsf{pp})$ Based on the security parameter λ it outputs secret key $\mathsf{sk_A}$ and public parameters pp for the session.

$\mathsf{GenRepD}(\mathsf{pp}, \mathcal{D}, \mathcal{P}_D, \mathsf{sk_A}, d) \rightarrow \{\mathsf{Drep}, \perp\}$ Upon receiving a data request with metadata d, it checks whether the data is accessible according to the access control policy $\mathcal{P}_D$ of the session[2] and outputs $\perp$ if not. Otherwise (x, d) is obtained from the database $\mathcal{D}$ and using $\mathsf{pp}, \mathsf{sk_A}$ and x the reply Drep is prepared.

[2] Note that in practice, $\mathcal{P}_D$ might depend on $\mathcal{P}_F$ that controls the allowed function requests in the session.

$\mathsf{GenReqF}(\mathsf{pp}, f, n) \to (\mathsf{Freq}, \mathsf{sk_B})$ A function request Freq is generated together with a key $\mathsf{sk_B}$, that will enable the extraction of the result. To do so, the algorithm takes in the session parameters pp, a function f and an integer n determining the number of dummy functions that conceals f.

$\mathsf{GenRepF}(\mathsf{pp}, \mathcal{P}_F, \mathsf{sk_A}, \mathsf{Freq}) \to \{\mathsf{Frep}, \perp\}$ To generate reply for a function request Freq, this algorithm first checks whether the function descriptions contained by Freq are allowed to compute according to the policy $\mathcal{P}_F$. If none of them is allowed, it outputs $\perp$, otherwise prepares Frep using the allowed function descriptions, the session parameters pp and the session key $\mathsf{sk_A}$.

$\mathsf{Extr}(\mathsf{Drep}, \mathsf{Frep}, \mathsf{sk_B}) \to \{y, \perp\}$ On inputs $\mathsf{Drep}, \mathsf{Frep}$ and $\mathsf{sk_B}$ the extraction algorithm outputs $\perp$ if f is forbidden to compute according to the policy $\mathcal{P}_F$ and y otherwise.

Correctness. We say that a CPFE_1^n protocol is correct if $y = f(x)$ whenever its algorithms are computed honestly by both parties.

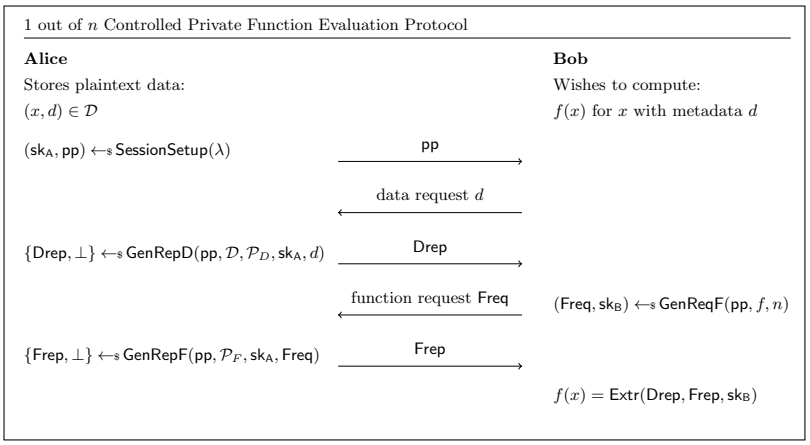


Fig. 3: One session of the CPFE_1^n protocol, where it is possible to issue more data or function requests and any of the resulting Drep and Frep, from the same session, can be combined to execute the corresponding computation. We assume that the parties are using an authenticated channel for communication. For a summary of the appearing parameters and their meaning, see the Appendix.

3.2 Security Model

Next, we formally define the security of CPFE_1^n , considering semi-honest participants. While formulating the relaxed function privacy requirement is straightforward, the indistinguishability approach for data privacy turns out to be trickier.

Intuitively, the goal is that the protocol should not reveal more information about the stored data than that is explicitly leaked by the result of an allowed function's evaluation, on an allowed input. The traditional way of arguing such security is to allow the adversary (Bob or a VASP in our case) to only access functions that result in the same output on a certain distinct input pair, called the challenge. If obeying this restriction, the adversary cannot tell apart which of the two inputs were used for the generation of Drep, we say that the scheme is semantically secure. Observe that if any function is accessible to Bob that separates the challenge input pair, then it is trivial to distinguish between the two cases. Consequently, it is crucial to ensure that the adversary obeys this restriction, however, this becomes challenging when the messages of the adversary only reveal its functions with probability $1/n$. Practically this means that the probability of correctly deciding whether someone managed to break the scheme or simply cheated in the challenge is $1/n$. We resolve this paradoxical situation by requiring the adversary to prove its honestness together with submitting its guess for the underlying input.

Definition 2 (IND Security of $\mathbf{CPFE}_1^n$). *We say that a $CPFE_1^n$ protocol $\mathcal{CPFE}_n^1(\mathsf{SessionSetup}, \mathsf{GenRepD}, \mathsf{GenReqF}, \mathsf{GenRepF}, \mathsf{Extr})$ achieves selective semantic security against semi-honest adversaries if it fulfils the following requirements:*

Data privacy. *Running the protocol, an adversarial function provider ($\mathcal{A}$) should not be able to achieve non-negligible advantage in the following game, where the challenger $\mathcal{C}$ plays the role*[3] *of the input provider (Alice or the data broker).*

- *Initialize: First, $\mathcal{A}$ chooses two challenge databases $\mathcal{D}_0, \mathcal{D}_1$ that differ only in one data value (i.e., $(x_0, d_{chal}) \in \mathcal{D}_0$ and $(x_1, d_{chal}) \in \mathcal{D}_1$ such that the metadata d_{chal} is identical) and sends them to the challenger. $\mathcal{C}$ computes $(\mathsf{sk_A}, \mathsf{pp}) \leftarrow_{\$} \mathsf{SessionSetup}$, flips a random coin $b \in \{0, 1\}$ and answers the subsequent queries using $\mathcal{D}_b$.*
- *Query: $\mathcal{A}$ adaptively submits data and function requests Dreq and Freq which are answered by $\mathcal{C}$.*
- *Challenge: Once $\mathcal{A}$ decides to finish the query phase, it has to output a guess b' for the bit b and provide a valid proof π that can be checked efficiently by a PPT challenger and confirms that all of its Freq requests in the query phase were prepared for functions f that respect the $f(x_0) = f(x_1)$ constraint.*

We say that the advantage of $\mathcal{A}$ in the above security game is $\Pr(b' = b) - \frac{1}{2}$ if the proof π is valid and zero otherwise.

Function Privacy. *The protocol should not enable the input provider (Alice or the data broker) to guess the computation logic of the Bob with probability greater than $1/n$ (where n is determined by Bob before a function request).*

[3] The enforceability of the access control policies is not investigated here, as it is a trivial task, so $\mathcal{C}$ considers each query to be valid.

4 A Generic Realisation of $\mathbf{CPFE}_1^n$

We are ready to present our simple, generic construction for CPFE_1^n . We start by describing the intuition behind our protocol.

From a straw-man proposal to our strategy. A naive first attempt to realise CPFE_1^n is to execute the computation on the server side by Alice. Upon receiving a n function descriptions (containing the one, Bob wishes to compute and $n-1$ dummy ones) and the metadata of the intended input, she verifies the request and computes the allowed ones. The results then can be shared with Bob using an OT_1^n scheme achieving both data and function privacy. Unfortunately, this method is not viable due to scalability issues caused by the n function evaluations. The subsequent natural idea is to shift the task of function evaluation to Bob, to eliminate the unnecessary computations and entirely hide the output from Alice. First, extensively studied tools as homomorphic encryption [3] or garbled circuits [35, 23] come to mind to enable computation on hidden data. However, none of these are suitable for our purposes. In case of the previous, one needs access to the secret key to obtain the output but this would also reveal the input as well. Moreover, the executed function would be under the full control of Bob, instead of Alice. Garbled circuits are more promising but still do not allow us to achieve our efficiency objectives. The reason is their one-time-use nature that necessitates the preparation of $j \cdot n$ garbled circuits to evaluate the same function on j inputs. We observe that two properties of an underlying primitive are essential to achieve our goals. Firstly, it should handle the functions and their inputs separately. Secondly, without explicit permission, no functions should be computable. As FE meets both of these requirements, Alice can share functional secret keys, corresponding to allowed functions, with Bob using OT_1^n . This enables the computation of a predetermined function of Bob on any input that Alice has encrypted to him. Our protocol is formally defined as follows.

$\mathsf{SessionSetup}(\lambda) \to (\mathsf{sk_A}, \mathsf{pp})$ Upon receiving a security parameter λ, the setup algorithm chooses the system parameters, samples $(\mathsf{msk_{FE}}, \mathsf{pp_{FE}}) \leftarrow\!\!\$\ \mathsf{FE.Setup}(\lambda)$, $\mathsf{pp_{OT}} \leftarrow\!\!\$\ \mathsf{OT.Setup}$ and $\mathsf{ck} \leftarrow\!\!\$\ \mathsf{C.Setup}$. Finally it outputs $(\mathsf{sk_A} = \mathsf{msk_{FE}}, \mathsf{pp} = (\mathsf{pp_{FE}}, \mathsf{pp_{OT}}, \mathsf{ck}))$.

$\mathsf{GenRepD}(\mathsf{pp}, \mathcal{D}, \mathcal{P}_D, \mathsf{sk_A}, d) \to \{\mathsf{Drep}, \perp\}$ The algorithm first verifies data request d, based on the data access control policy $\mathcal{P}_D$.

- If Bob is allowed to use the specific data, then the corresponding data value x from $\mathcal{D}$ is determined and then encrypted.
- The output is $\mathsf{Drep} = \mathsf{ct} \leftarrow\!\!\$\ \mathsf{FE.Enc}(\mathsf{pp_{FE}}, \mathsf{msk_{FE}}, x)$ or $\mathsf{Drep} = \perp$ if Bob is not allowed to access the requested data determined by d.

$\mathsf{GenReqF}(\mathsf{pp}, f, n) \to \mathsf{Freq}, \mathsf{sk_B}$ Besides the public parameters pp, the algorithm takes as input the function f to be computed, and an integer $n \in \mathbb{N}^+$, then it executes the following steps.

- Samples $n-1$ functions randomly from the function class $\mathcal{F}$ (that contains f): $\{f_i \leftarrow\!\!\$\ \mathcal{F}\}_{i \in [n-1]}$ and let $f_n = f$.

- Takes a random permutation σ and prepares $F_R := (\hat{f}_1, \ldots, \hat{f}_n)$, where $\hat{f}_i = f_{\sigma^{-1}(i)}$ so that each f_i ends up at position $\sigma(i)$ in the sequence.
- Determines the sequence number of the function for which the functional key will become available by computing $\mathsf{pk_{OT}} \leftarrow_\$ \mathsf{OT.Commit}(\mathsf{pp_{OT}}, \sigma(n))$,
- commit to the value $\sigma(n)$ and all the random choices (rand) used in the previous step: $(\mathsf{c}, \mathsf{d}) \leftarrow_\$ \mathsf{C.Commit}(\mathsf{ck}, \sigma(n), \mathsf{rand})$.
- Finally it outputs $\mathsf{Freq} = (F_R, \mathsf{pk_{OT}}, \mathsf{c})$ and $\mathsf{sk_B} = (\sigma, \mathsf{d})$.

$\mathsf{GenRepF}(\mathsf{pp}, \mathcal{P}_F, \mathsf{sk_A}, \mathsf{Freq}) \rightarrow \{\mathsf{Frep}, \perp\}$ Upon a function request Freq, it is parsed as $((\hat{f}_1, \ldots, \hat{f}_n), \mathsf{pk_{OT}}, \mathsf{c})$, and the algorithm proceeds as follows.

- It first enforces the function access policy $\mathcal{P}_F$ by checking whether Bob is allowed to compute the functions of F_R. The allowed function set is denoted by $F_A \subseteq F_R$. If F_A is the empty set, then the output is $\perp$. Let $I := \{i | \hat{f}_i \in F_A\}$ denote the index set of the allowed functions.
- Functional secret keys are generated for all $\hat{f}_i \in F_A$: $\{\mathsf{fsk_{f_i}} \leftarrow_\$ \mathsf{FE.KeyGen}(\mathsf{pp_{FE}}, \mathsf{msk_{FE}}, \hat{f}_i)\}_{i \in I}$.
- The resulting functional keys are made oblivious: $\{o_i \leftarrow_\$ \mathsf{OT.Trans}(\mathsf{pp_{OT}}, \mathsf{pk_{OT}}, \mathsf{fsk_{f_i}})\}_{i \in I}$.
- Finally it outputs $\mathsf{Frep} = \{(i, o_i)\}_{i \in I}$.

$\mathsf{Extr}(\mathsf{pp}, \mathsf{sk_B}, \mathsf{Drep}, \mathsf{Frep}) \rightarrow f(x)$ Extraction can be used to evaluate $f = \hat{f}_{\sigma(n)} = f_n$ (underlying Frep) on input x (encoded by Drep), if $\hat{f}_{\sigma(n)} \in F_A$ and Bob has right to use x.

- first the functional secret key for the requested function is retrieved: $\mathsf{fsk_{f_{\sigma(n)}}} = \mathsf{OT.Retr}(\mathsf{pp_{OT}}, o_{\sigma(n)})$ and
- finally the ciphertext is decrypted to output $\mathsf{FE.Dec}(\mathsf{fsk_{f_{\sigma(n)}}}, \mathsf{ct}) = f(x)$.

The correctness of the proposed protocol follows from the correctness of the underlying schemes. We note that the use of the commitment scheme is necessary for the security proof and it does not contribute to the functionality of the protocol.

4.1 Security Analysis

Next, we state our main theorem.

Theorem 1. *The proposed protocol is secure according to Definition 2 as long as the underlying FE scheme achieves s-IND-CPA security, the OT_1^n scheme is secure with respect to a malicious receiver, and the commitment scheme is also secure.*

Proof (informal). Function privacy directly follows from the receiver's privacy in the applied OT_1^n protocol and the hiding property of the used commitment scheme.

To argue that data privacy holds as well, we indirectly assume that $\mathcal{A}$ can achieve a non-negligible advantage in the security game, which can happen in the following two ways. Let us denote with E the event that $\mathcal{A}$ can obtain some information about functional secret keys that it should not have access to

(e.g., corresponding to more than one functions from a single Frep). $\bar{E}$ is the complement of the previous event, namely that $\mathcal{A}$ has only access to information about legitimate functional secret keys.

Considering the probability of E, we observe that FE related – non-key – values (ciphertexts, public parameters) clearly cannot leak information about the keys without harming the semantic security of FE so we can focus on those values that depend on functional keys. At the same time, these also cannot leak such information, because that would contradict with the sender's privacy in the applied OT scheme even if $\mathcal{A}$ is malicious, i.e., deviates from the protocol. This implies event $\bar{E}$, i.e., $\mathcal{A}$ can only have information about functional keys corresponding to functions, which must respect the rules of the game, but cannot access keys for any of the dummy functions. Recall that $\mathcal{A}$ has to prove that its function requests adhered to the restrictions of the game. To do so, $\mathcal{A}$ can submit the opening values d for the corresponding commitments (c) that were included in the Freq queries. Using these, with the $\mathsf{C.Open}$ algorithm the challenger can efficiently check which functional keys were revealed to $\mathcal{A}$ and whether these indeed met the requirements of the security game.

At this point, note that in the security game it is irrelevant whether the constraint on the function queries is enforced by $\mathcal{C}$ before answering them or $\mathcal{A}$ can prove that it respected the rules. With this observation, it turns out that the security game is equivalent to the s-IND-CPA game for FE, where the challenge messages are the differing elements of $\mathcal{D}_0$ and $\mathcal{D}_1$. In other words, achieving a non-negligible advantage in the game would contradict the security of the underlying FE scheme. □

5 Concrete Instantiation for Inner Products

Besides our generic solution, we also present a concrete instantiation of our CPFE_1^n protocol for computing inner products. The investigated functionality is a particularly important tool for statistical analysis. Furthermore, in a different context, [17] showed that it enables the computation of conjunctions, disjunction, more generally CNF or DNF formulas, exact thresholds and even polynomial evaluation. We use the DDH-based inner product FE scheme of [2], the OT_1^n protocol of [32] (against malicious receiver) and Pedersen commitments [28]. Our choice for FE can be surprising for the first sight as it is a public-key scheme and we do not publish the public key. The motivation behind our choice is the extreme simplicity of the scheme compared to secret-key variants that also achieve function privacy which we do not need in our solution.

Focusing on a specific function class and choosing the applied primitives such that they share the same underlying structure allows us simplifications compared to the general case. E.g., our proof technique, which necessitates the application of a commitment scheme, does not cause any additional overhead now because the $\mathsf{OT.Commit}$ algorithm of [32] itself computes a Pedersen commitment that can also be used in the security proof.

We also mention the limitation of our construction that is common in every DDH-like assumption-based inner product FE schemes. The expressiveness of the function is limited to a polynomial-sized output range because extraction of the result involves a discrete logarithm computation. However, this is always relative to the same base, enabling pre-computation.

The algorithms of our inner product CPFE_1^n protocol are the following.

$\mathsf{SessionSetup}(\lambda, \ell) \to (\mathsf{sk_A}, \mathsf{pp})$ The algorithm takes in a security parameter λ and the length ℓ of vectors that are going to be the inputs of the inner product computation. The algorithm
- chooses a λ-bit prime number p, and a group $\mathbb{G}$ of order p with generators $g, h \in \mathbb{G}$,
- selects a hash function $H : \mathbb{G} \to \mathbb{Z}_p$, that is modelled as a random oracle,
- samples a random vector $\boldsymbol{s} = (s_1, \ldots, s_\ell) \leftarrow_\$ \mathbb{Z}_p^\ell$,
- and outputs $\mathsf{sk_A} = \boldsymbol{s}$ and $\mathsf{pp} = (\mathbb{G}, p, g, h, H, \ell)$.

$\mathsf{GenRepD}(\mathsf{pp}, \mathcal{D}, \mathcal{P}_D, \mathsf{sk_A}, \mathsf{Dreq}) \to \{\mathsf{Drep}, \perp\}$ The reply generation for some input request d has access to the database $\mathcal{D}$ where data vectors can be identified based on their metadata. It executes the following steps.
- It enforces the access control policy $\mathcal{P}_D$. If the affected party is not authorised to compute on the vector with metadata d, the algorithm outputs $\perp$ and terminates, otherwise
- randomly chooses $r' \leftarrow_\$ \mathbb{Z}_p^*$ and computes $R' = g^{r'}$.
- It computes $\mathsf{ct}_j = g^{r's_j} g^{x_j}$ for $j = 1, \ldots, \ell$.
- The output is $\mathsf{Drep} = (R', \{\mathsf{ct}_j\}_{j \in [\ell]})$.

$\mathsf{GenReqF}(\mathsf{pp}, \boldsymbol{y}, n) \to \mathsf{Freq}, \mathsf{sk_B}$ The function to be computed is represented by $\boldsymbol{y}_n := \boldsymbol{y} = (y_{1,1}, \ldots, y_{1,\ell}) \in \mathbb{Z}_p^\ell$, n is the security parameter for function privacy, while pp is parsed as $(\mathbb{G}, p, g, h, H, \ell)$. The request generation algorithm
- samples $n-1$ random vectors $\boldsymbol{y}_i = (y_{i,1}, \ldots, y_{i,\ell}) \in \mathbb{Z}_p^\ell$ for $i = 1, \ldots, n-1$,
- takes a random permutation σ and prepares $F_R = (\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_n)$ where $\hat{\boldsymbol{y}}_i = \boldsymbol{y}_{\sigma^{-1}(i)}$.
- Using a randomly chosen $r \leftarrow_\$ \mathbb{Z}_p^*$ computes $R = g^r h^{\sigma(n)}$.
- Finally, it outputs $\mathsf{Freq} = (F_R, R)$ and $\mathsf{sk_B} = (\sigma, r, \boldsymbol{y})$.

$\mathsf{GenRepF}(\mathsf{pp}, \mathcal{P}_F, \mathsf{sk_A}, \mathsf{Freq}) \to \{\mathsf{Frep}, \perp\}$ The function reply generation parses Freq as (F_R, R) and executes the following steps.
- It enforces the function access policy $\mathcal{P}_F$. Accordingly, $F_A \subseteq F_R$ is determined that only contains allowed vectors. Let $I := \{i | \hat{\boldsymbol{y}}_i \in F_A\}$ denote the index set of the allowed vectors. If F_A is the empty set, then it returns $\perp$.
- Randomly chooses $r'' \leftarrow_\$ \mathbb{Z}_p^*$ and computes $R'' = g^{r''}$.
- For all allowed function request $\hat{\boldsymbol{y}}_i = (\hat{y}_{i,1}, \ldots, \hat{y}_{i,\ell}) \in F_A$, it computes $S_i = \langle \hat{\boldsymbol{y}}_i, \mathsf{sk_A} \rangle$.
- Finally, it computes $o_i = S_i \oplus H((R/h^i)^{r''}, i)$ for all $i \in I$.
- The output is $\mathsf{Frep} = (R'', \{(i, o_i)\}_{i \in I})$.

$\mathsf{Extr}(\mathsf{ppsk_B}, \mathsf{Drep}, \mathsf{Frep}) \to \{\perp, \langle \boldsymbol{x}, \boldsymbol{y} \rangle\}$ For extraction of the result of the inner product the following steps are executed.
- If $o_{\sigma(n)}$ is not part of Frep, then output $\perp$.

– Otherwise, computes $S_{\sigma(n)} = o_{\sigma(n)} \oplus H(R''^r, \sigma(n))$.
– The results are obtained by computing $\prod_{j\in[\ell]} \mathsf{ct}_j^{y_j} / R'^{S_{\sigma(n)}} = g^{\langle \boldsymbol{x},\boldsymbol{y}\rangle}$.
– The discrete logarithm of $g^{\langle \boldsymbol{x},\boldsymbol{y}\rangle}$ is computed with basis g and returned as the output. If $\langle \boldsymbol{x}, \boldsymbol{y}\rangle$ is not contained in a predetermined polynomial-sized range, then $\perp$ is returned.

The correctness of the above scheme follows from the correctness of the underlying schemes.

5.1 Security and Performance

Theorem 2. *The above Inner Product CPFE$_1^n$ scheme is secure according to Definition 2 in the random oracle model as long as the DDH assumption holds.*

Proof. The theorem directly follows from Theorem 1 and from the fact that the underlying inner product FE scheme is s-IND-CPA secure under the DDH assumption [2] (that implies the CDH assumption), the applied OT scheme is secure against malicious receivers in the random oracle model under the CDH assumption [32] (that implies the Dlog assumption) and that the implicitly used Pedersen commitment [28] is secure under the Dlog assumption. □

Finally, we summarise the costs of the above inner product CPFE$_1^n$ scheme. Regarding the communication costs, Freq contains $n\ell$ integers, Frep consists of $2n$ integers and one group element, while Drep is $\ell + 1$ group elements. The computational costs of the algorithms, depicted in Table 1, are comparable to the applied inner product FE scheme [2], that also enables private inner product computation but without any control over the affected vectors. The overhead of our solution is proportional to the security parameter n, which determines the number of the applied dummy vectors.

Table 1: The computational costs of our algorithms for Inner Product CPFE$_1^n$.

Operation	GenReqF	GenRepD	GenRepF	Extr
#{rand from $\mathbb{Z}_p^*$}	$\ell(n+1)+1$	1	1	-
#{exp in $\mathbb{G}$}	2	$2\ell+1$	$n+1$	$2\ell+1$
#{mult in $\mathbb{G}$}	1	ℓ	n	$2\ell+1$
#{mult in $\mathbb{Z}_p^*$}	-	ℓ	ℓ	-
#{Hashing}	-	-	n	1
#{Dlog}	-	-	-	1

6 Conclusion and Open Directions

In this work, we initiated the study of controlled but still private function evaluation protocols motivated both by the shortcomings of widespread PFE protocols

and by open problems arising in the context of secure IoT data markets. We proposed a generic solution for the problem of simultaneous input and function privacy, building on functional encryption and showed a concrete instantiation for inner product computation. As the most straightforward extension of our work, we are planning to implement this latter protocol to demonstrate its practical relevance.

Finally, we close our study by collecting possible directions for future work. First, it would be interesting to see whether our techniques are generalizable for the more realistic multi-input functions. While we worked in the semi-honest model with an indistinguishability-based security definition, it would be challenging to achieve similar results for malicious participants or simulation-based security. Another interesting direction could be to increase the privacy of the function provider e.g., by also hiding the exact location of the used input similarly as it was done in [7]. We also find it possible that more sophisticated variants of controlled PFE can be realised as well, which may fall closer to the ideal functionality depicted in Fig. 1c. A possible approach could be to design function verification in a zero-knowledge manner, such that Bob could prove that his function is not among certain "forbidden ones" without revealing his function.

Appendix

We summarise the meaning of the different parameters used in this work in Table 2.

References

1. Abadi, M., Feigenbaum, J.: Secure circuit evaluation. Journal of Cryptology **2**(1), 1–12 (1990)
2. Abdalla, M., Bourse, F., Caro, A.D., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) Proceedings of Public-Key Cryptography - PKC 2015. LNCS, vol. 9020, pp. 733–751. Springer (2015)
3. Armknecht, F., Boyd, C., Carr, C., Gjøsteen, K., Jäschke, A., Reuter, C.A., Strand, M.: A guide to fully homomorphic encryption (2015), `http://eprint.iacr.org/2015/1192`
4. Badrinarayanan, S., Goyal, V., Jain, A., Sahai, A.: Verifiable functional encryption. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology - ASIACRYPT 2016, Proceedings, Part II. LNCS, vol. 10032, pp. 557–587 (2016). https://doi.org/10.1007/978-3-662-53890-6_19
5. Bishop, A., Jain, A., Kowalczyk, L.: Function-hiding inner product encryption. In: Advances in Cryptology - ASIACRYPT 2015, Proceedings, Part I. pp. 470–491 (2015), `http://dx.doi.org/10.1007/978-3-662-48797-6_20`
6. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Ishai, Y. (ed.) Theory of Cryptography - TCC 2011. Proceedings. LNCS, vol. 6597, pp. 253–273. Springer (2011)

Table 2: A brief summary our notations.

Related to	Notation	Meaning
FE	$\mathsf{pp_{FE}}$	public parameters for FE
	$\mathsf{msk_{FE}}$	FE master secret key
	$\mathsf{fsk_f}$	functional secret key for function f
	ct	FE ciphertext
OT_1^n	$\mathsf{pp_{OT}}$	public parameters for OT_1^n
	$\mathsf{pk_{OT}}$	commitment of the receiver
	$\mathsf{sk_{OT}}$	secret of the receiver
	o_i	ith oblivious message element of the sender
CPFE_1^n	$\mathcal{D}$	database
	$\mathcal{P}_D$	description of data access control policy
	$\mathcal{P}_F$	description of function access policy
	Freq	function request
	Drep	reply for a data request
	Frep	reply for a function request
	$\mathsf{sk_A}$	secret (session) key of the data provider (Alice/data borker)
	$\mathsf{sk_B}$	secret key of the funct. provider (Bob/VASP) for a funct. request
	λ	security parameter (corresponding to data privacy)
	n	security parameter (corresponding to function privacy)
	σ	a random permutation
	d	a metadata description
	F_R	requested function set (including dummy functions)
	F_A	accessible function set ($F_A \subseteq F_R$)
	I	accessible function's index set
Commit.	ck	public commitment key
	c	commitment value
	d	opening value

7. Canetti, R., Ishai, Y., Kumar, R., Reiter, M.K., Rubinfeld, R., Wright, R.N.: Selective private function evaluation with applications to private statistics. In: Kshemkalyani, A.D., Shavit, N. (eds.) Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing, PODC 2001,. pp. 293–304. ACM (2001). https://doi.org/10.1145/383962.384047
8. Caro, A.D., Iovino, V., Jain, A., O'Neill, A., Paneth, O., Persiano, G.: On the achievability of simulation-based security for functional encryption. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology - CRYPTO 2013, Proceedings, Part II. LNCS, vol. 8043, pp. 519–535. Springer (2013). https://doi.org/10.1007/978-3-642-40084-1_29
9. Dong, C., Chen, L.: A fast secure dot product protocol with application to privacy preserving association rule mining. In: Tseng, V.S. et al (eds.) Advances in Knowledge Discovery and Data Mining - 18th Pacific-Asia Conference, PAKDD 2014. Proceedings, Part I. LNCS, vol. 8443, pp. 606–617. Springer (2014). https://doi.org/10.1007/978-3-319-06608-0_50
10. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for

all circuits. In:FOCS-2013. pp. 40–49. IEEE Computer Society (2013). https://doi.org/10.1109/FOCS.2013.13
11. Garg, S., Gentry, C., Halevi, S., Zhandry, M.: Functional encryption without obfuscation. In: Kushilevitz, E., Malkin, T. (eds.) Theory of Cryptography - TCC 2016-A, Proceedings, Part II. LNCS, vol. 9563, pp. 480–511. Springer (2016). https://doi.org/10.1007/978-3-662-49099-0_18
12. Goethals, B., Laur, S., Lipmaa, H., Mielikäinen, T.: On private scalar product computation for privacy-preserving data mining. In: Park, C., Chee, S. (eds.) Information Security and Cryptology - ICISC 2004, Revised Selected Papers. LNCS, vol. 3506, pp. 104–120. Springer (2004). https://doi.org/10.1007/11496618_9
13. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology - CRYPTO 2012. Proceedings. LNCS, vol. 7417, pp. 162–179. Springer (2012). https://doi.org/10.1007/978-3-642-32009-5_11
14. Günther, D., Kiss, Á., Schneider, T.: More efficient universal circuit constructions. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology - ASIACRYPT 2017, Proceedings, Part II. LNCS, vol. 10625, pp. 443–470. Springer (2017). https://doi.org/10.1007/978-3-319-70697-9_16
15. Horváth, M., Buttyán, L.: Problem domain analysis of iot-driven secure data markets. In: Gelenbe, E.e.a. (ed.) Security in Computer and Information Sciences. pp. 57–67. Springer (2018)
16. Katz, J., Malka, L.: Constant-round private function evaluation with linear complexity. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology - ASIACRYPT 2011. Proceedings. LNCS, vol. 7073, pp. 556–571. Springer (2011). https://doi.org/10.1007/978-3-642-25385-0_30
17. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. J. Cryptology **26**(2), 191–224 (2013). https://doi.org/10.1007/s00145-012-9119-4
18. Kennedy, W.S., Kolesnikov, V., Wilfong, G.T.: Overlaying conditional circuit clauses for secure computation. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology - ASIACRYPT 2017. Proceedings, Part II. LNCS, vol. 10625, pp. 499–528. Springer (2017). https://doi.org/10.1007/978-3-319-70697-9_18
19. Kiss, Á., Schneider, T.: Valiant's universal circuit is practical. In: Fischlin, M., Coron, J. (eds.) Advances in Cryptology - EUROCRYPT 2016. Proceedings, Part I. LNCS, vol. 9665, pp. 699–728. Springer (2016). https://doi.org/10.1007/978-3-662-49890-3_27.
20. Koblitz, N., Menezes, A.J.: The random oracle model: a twenty-year retrospective. Des. Codes Cryptography **77**(2-3), 587–610 (2015). https://doi.org/10.1007/s10623-015-0094-2
21. Kolesnikov, V., Schneider, T.: A practical universal circuit construction and secure evaluation of private functions. In: Tsudik, G. (ed.) Financial Cryptography and Data Security, FC-2008, Revised Selected Papers. LNCS, vol. 5143, pp. 83–97. Springer (2008). https://doi.org/10.1007/978-3-540-85230-8_7
22. Liang, F., Yu, W., An, D., Yang, Q., Fu, X., Zhao, W.: A survey on big data market: Pricing, trading and protection. IEEE Access **6**, 15132–15154 (2018). https://doi.org/10.1109/ACCESS.2018.2806881
23. Lindell, Y., Pinkas, B.: A proof of security of Yao's protocol for two-party computation. Journal of Cryptology **22**(2), 161–188 (2009)
24. Mohassel, P., Sadeghian, S.S.: How to hide circuits in MPC an efficient framework for private function evaluation. In: Johansson, T., Nguyen, P.Q. (eds.) Advances

in Cryptology - EUROCRYPT 2013. Proceedings. LNCS, vol. 7881, pp. 557–574. Springer (2013). https://doi.org/10.1007/978-3-642-38348-9_33
25. Mohassel, P., Sadeghian, S.S., Smart, N.P.: Actively secure private function evaluation. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014. Proceedings, Part II. LNCS, vol. 8874, pp. 486–505. Springer (2014). https://doi.org/10.1007/978-3-662-45608-8_26
26. Naveed, M., Agrawal, S., Prabhakaran, M., Wang, X., Ayday, E., Hubaux, J., Gunter, C.A.: Controlled functional encryption. In: Ahn, G., Yung, M., Li, N. (eds.) Proceedings of the 2014 ACM SIGSAC. pp. 1280–1291. ACM (2014). https://doi.org/10.1145/2660267.2660291
27. Paus, A., Sadeghi, A., Schneider, T.: Practical secure evaluation of semi-private functions. In: Abdalla, M., Pointcheval, D., Fouque, P., Vergnaud, D. (eds.) Applied Cryptography and Network Security, ACNS-2009. Proceedings. LNCS, vol. 5536, pp. 89–106 (2009). https://doi.org/10.1007/978-3-642-01957-9_6
28. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) Advances in Cryptology - CRYPTO '91, Proceedings. LNCS, vol. 576, pp. 129–140. Springer (1991)
29. Sadeghi, A., Schneider, T.: Generalized universal circuits for secure evaluation of private functions with application to data classification. In: Lee, P.J., Cheon, J.H. (eds.) Information Security and Cryptology - ICISC 2008. Revised Selected Papers. LNCS, vol. 5461, pp. 336–353. Springer (2008). https://doi.org/10.1007/978-3-642-00730-9_21
30. Sadeghian, S.: New Techniques for Private Function Evaluation. Ph.D. thesis, University of Calgary (2015)
31. Sander, T., Young, A.L., Yung, M.: Non-interactive cryptocomputing for nc^1. In: 40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA. pp. 554–567. IEEE Computer Society (1999). https://doi.org/10.1109/SFFCS.1999.814630
32. Tzeng, W.: Efficient 1-out-of-n oblivious transfer schemes with universally usable parameters. IEEE Trans. Computers **53**(2), 232–240 (2004). https://doi.org/10.1109/TC.2004.1261831
33. Valiant, L.G.: Universal circuits (preliminary report). In: Chandra, A.K., Wotschke, D., Friedman, E.P., Harrison, M.A. (eds.) Proceedings of the 8th Annual ACM Symposium on Theory of Computing. pp. 196–203. ACM (1976). https://doi.org/10.1145/800113.803649
34. Yao, A.C.: Protocols for secure computations (extended abstract). In: 23rd Annual Symposium on Foundations of Computer Science 1982. pp. 160–164. IEEE Computer Society (1982). https://doi.org/10.1109/SFCS.1982.38
35. Yao, A.C.C.: How to generate and exchange secrets. In: Foundations of Computer Science, 1986., 27th Annual Symposium on. pp. 162–167. IEEE (1986)
36. Zhu, Y., Wang, Z., Hassan, B., Zhang, Y., Wang, J., Qian, C.: Fast secure scalar product protocol with (almost) optimal efficiency. In: Guo, S. et al (eds.) Collaborative Computing: Networking, Applications, and Worksharing 2015. Proceedings. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 163, pp. 234–242. Springer (2015). https://doi.org/10.1007/978-3-319-28910-6_21