Adaptive Oblivious Transfer and Generalization

Olivier Blazy[1], Céline Chevalier[2], and Paul Germouty[1]

[1] Université de Limoges, XLim, France
[2] CRED, Université Panthéon-Assas, Paris, France

Abstract. Oblivious Transfer (OT) protocols were introduced in the seminal paper of Rabin, and allow a user to retrieve a given number of lines (usually one) in a database, without revealing which ones to the server. The server is ensured that only this given number of lines can be accessed per interaction, and so the others are protected; while the user is ensured that the server does not learn the numbers of the lines required. This primitive has a huge interest in practice, for example in secure multi-party computation, and directly echoes to Symmetrically Private Information Retrieval (SPIR).

Recent Oblivious Transfer instantiations secure in the UC framework suffer from a drastic fallback. After the first query, there is no improvement on the global scheme complexity and so subsequent queries each have a global complexity of $\mathcal{O}(|DB|)$ meaning that there is no gain compared to running completely independent queries. In this paper, we propose a new protocol solving this issue, and allowing to have subsequent queries with a complexity of $\mathcal{O}(\log(|DB|))$ while keeping round optimality, and prove the protocol security in the UC framework with adaptive corruptions and reliable erasures.

As a second contribution, we show that the techniques we use for Oblivious Transfer can be generalized to a new framework we call *Oblivious Language-Based Envelope* (OLBE). It is of practical interest since it seems more and more unrealistic to consider a database with uncontrolled access in access control scenarios. Our approach generalizes Oblivious Signature-Based Envelope, to handle more expressive credentials and requests from the user. Naturally, OLBE encompasses both OT and OSBE, but it also allows to achieve Oblivious Transfer with fine grain access over each line. For example, a user can access a line if and only if he possesses a certificate granting him access to such line.

We show how to generically and efficiently instantiate such primitive, and prove them secure in the Universal Composability framework, with adaptive corruptions assuming reliable erasures. We provide the new UC ideal functionalities when needed, or we show that the existing ones fit in our new framework.

The security of such designs allows to preserve both the secrecy of the database values and the user credentials. This symmetry allows to view our new approach as a generalization of the notion of Symmetrically PIR.

Keywords. Adaptive Oblivious Transfer, Oblivious Signature-Based Envelope, UC Framework, Private Information Retrieval.

1 Introduction

Oblivious Transfer (OT) is a notion introduced by Rabin in [53]. In its classical 1-out-of-n version, it allows a user $\mathcal{U}$ to access a single line of a database while interacting with the server $\mathcal{S}$ owning the database. The user should be oblivious to the other line values, while the server should be oblivious to which line was indeed received. Oblivious transfer has a fundamental role for achieving secure multi-party computation: It is for example needed for every bit of input in Yao's protocol [59] as well as for Oblivious RAM ([56] for instance), for every AND gate in the Boolean circuit computing the function in [35] or for almost all known garbled circuits [6].

Private Information Retrieval (PIR) schemes [25] allow a user to retrieve information from a database, while ensuring that the database does not learn which data were retrieved. With the increasing need for user privacy, these schemes are quite useful in practice, be they used for accessing records for email repositories, collection of webpages, music... But while protecting the privacy of the user, it is equally important that the user should not learn more information than he is allowed to. This is called database privacy and the corresponding protocol is called a Symmetrically Private Information Retrieval (SPIR), which could be employed in practice, for medical data or biometric information. This notion is closely related to Oblivious Transfer.

Due to their huge interest in practice, it is important to achieve low communication on these Oblivious Transfer protocols. A usual drawback is that the server usually has to send a message equivalent to the whole database each time the user requests a line. If it is logical, in the UC framework, that an OT protocol requires a cost linear in the size of the database for the first line queried. One may then hope to amortize the cost for further queries between the same server and the same user (or even another user, if possible), reducing the efficiency gap between Private Information Retrieval schemes and their stronger equivalent Oblivious Transfer schemes. We thus deal in this paper with a more efficient way, which is to achieve *Adaptive* Oblivious Transfer, in which the user can adaptively ask several lines of the database. In such schemes, the server only sends his database once at the beginning of the protocol, and all the subsequent communication is in $o(n)$, more precisely logarithmic. The linear cost is batched once and for all in this preprocessing phase, achieving then a logarithmic complexity similar to the best PIR schemes.

Smooth Projective Hash Functions (SPHF), used in conjunction with *Commitments* have become the standard way to deal with such secret message transfers. In a commitment scheme, the sender is going to commit to the line required (*i.e.* to give the receiver an analogue of a sealed envelope containing his value i) in such a way that he should not be able to open to a value different from the one he committed to (*binding* property), and that the receiver cannot learn anything about i (*hiding* property) before a potential opening phase. During the opening phase, however, the committer would be asked to reveal i in such a way that the receiver can verify it was indeed i that was contained in the envelope.

But, in our applications, there cannot be an opening phase, due to the *oblivious* requirements on the protocols and the secrecy of the database line i sent. The decommitment (opening phase) will thus be implicit, which means that the committer does not really open its commitment, but rather convinces the receiver that it actually committed to the value it pretended to. We achieve this property thanks to *Smooth Projective Hash Functions* [26, 33], which have been widely used in such circumstances (see [1, 2, 8, 9, 45] for instance). These hash functions are defined in such a way that their value can be computed in two different ways if the input belongs to a particular subset (the *language*), either using a private hashing key or a public projection key along with a private witness ensuring that the input belongs to the language. The hash value obtained is indistinguishable from random in case the input does not belong to the language (*smoothness*) and in case the input does belong to the language but no witness is known (*pseudo-randomness*).

In a nutshell, to ensure implicit decommitment, the sender will thus simply mask the database line with this hash value computed using the private hashing key. He will then send it along with the public projection key to the user, who will be able to compute the same hash value thanks to the randomness of the commitment of this line he sent in the first place (the randomness is the witness of the membership of the commitment to the language of commitments of this specific line). In order to ensure adaptive security in the universal composability framework, the commitments used are usually required to be both *extractable* (meaning that a simulator can recover the value i committed to thanks to a trapdoor) and *equivocable* (meaning that a simulator can open a commitment to a value i' different from the value i it committed to thanks to a trapdoor).

In order to simplify these commitments, which can be quite technical, we choose here to rely on words in more complex languages rather than on simple line numbers. More precisely, the user will first compute an equivocable commitment on the line number required, which will be his word w in the language. This word will then be encrypted under a CCA encryption scheme, and the SPHF will be constructed for this word (rather than for the line number), which will be simpler. Furthermore, this abstraction consisting in encoding line numbers as words in more complex languages will reveal useful in more general contexts, not only Oblivious Transfer, the simplest of which being Oblivious Signature Based Envelope.

Oblivious Signature-Based Envelope (OSBE) was introduced by Li, Du and Boneh in [49]. OSBE schemes consider the case where Alice (the receiver) is a member of an organization and possesses a certificate produced by an authority attesting she actually belongs to this organization. Bob (the sender) wants to send a private message P to members of this organization. However due to the sensitive nature of the organization, Alice does not want to give Bob neither her certificate nor a proof she belongs to the organization. OSBE lets Bob send an obfuscated version of this message P to Alice, in such a way that she will be able to find P if and only if she is in the required organization. In the process, Bob cannot decide whether Alice does really belong to the organization. We even

manage to construct a more general framework to capture many protocols around trust negotiation, where the user receives a message if and only if he possesses some credentials or specific accreditations. As a reference to OSBE, we call this framework *Oblivious Language-Based Envelope* (OLBE).

1.1 Related Work

Since the original paper [53], several instantiations and optimizations of OT protocols have appeared in the literature [23, 51], including proposals in the UC framework. More recently, new instantiations have been proposed, trying to reach round-optimality [41], and/or low communication costs [52]. Recent schemes like [1,9] manage to achieve round-optimality while maintaining a small communication cost. Choi *et al.* [24] also propose a generic method and an efficient instantiation secure against adaptive corruptions in the CRS model with erasures, but it is only 1-out-of-2 and it does not scale to 1-out-of-n OT, for $n > 2$. As far as adaptive versions of those protocols are concerned, this problem was first studied by [37,47,50], and more recently UC secure instantiations were proposed, but unfortunately either under the Random Oracle, or under not so standard assumptions such as q-Hidden LRSW or later on q-SDH [17,20,39,43,54], but without allowing adaptive corruptions.

Concerning automated trust negotiation, two frameworks have been proposed to encompass the symmetric protocols (Password-based Authenticated Key-Exchange, Secret Handshakes and Verfier-Based PAKE): The Credential Authenticated Key Exchange [16], and Language-based Authenticated Key Exchange (LAKE) [7], in which two parties establish a common session key if and only if they hold credentials that belong to specific (and possibly independent) languages chosen by the other party. As for OSBE, the authors in [13] improved the security model initially proposed in [49], showing how to use Smooth Projective Hash Functions to do implicit proof of knowledge, and proposed the first efficient instantiation of OSBE, under a standard hypothesis. It fits, as well as Access Controlled Oblivious Transfer [18, 19], Priced Oblivious Transfer [4, 54]) and Conditional Oblivious Transfer [28], into the generic notion of Conditional Disclosure of Secrets (see for instance [4, 5, 15, 32, 34, 42, 48, 58]).

1.2 Contributions

Our first contribution is to give the first round-optimal adaptive Oblivious Transfer protocol secure in the UC framework with adaptive corruptions under standard assumptions (MDDH) and assuming reliable erasures. We show how to instantiate the needed building blocks using standard assumptions, using or extending various basic primitives in order to fit the MDDH framework introduced in [30]. In our scheme, the server first preprocesses its database in a time linear in the length of the database and transfers it to the receiver. After that, the receiver and the sender can run many instances of the protocol on the same database as input and adaptively chosen inputs from the receiver, with a cost sublinear in the database.

It is interesting to note that our resulting adaptive Oblivious Transfer scheme has an amortized complexity in $\mathcal{O}(\log |DB|)$, which is similar to current Private Information Retrieval instantiations [46], that have weaker security prerequisites, and much better than current UC secure Oblivious Transfer under standard assumptions (as they are in $\mathcal{O}(|DB|)$. For a fair comparison it should be stated that the PIR schemes allow this complexity directly from the first query while in our case due to the preprocessing, this amortized cost is only reached after a high number of queries. However, it is interesting to see this convergence in spite of hugely different security models and expectation. Compared to existing versions cited above (either proven in classical security models, or in the UC framework but only with static corruptions and under non-standard assumptions), we manage to prove its security under standard assumptions, like SXDH, and allow UC security with adaptive user corruptions.

As a side result, it is worth noting that we follow some ideas developed in the construction explained in [37] around Blind Identity-Based Encryption and provide techniques in order to transform IBE schemes into blind ones, applying them to revisit the one given in [12], in order to show how we can answer blind user secret key-retrieval, which can be of independent interest.

As a second contribution, we propose our new notion, that we call Oblivious Language-Based Envelope. We provide a security model by giving a UC ideal functionality, and show that this notion supersedes the classical asymmetric automated trust negotiation schemes recalled above such as Oblivious Transfer and Oblivious Signature-Based Envelope. We show how to choose the languages in order to obtain from our framework all the corresponding ideal functionalities, recovering the known ones (such as OT) and providing the new ones (such as OSBE, to the best of our knowledge). We then give a generic construction scheme fulfilling our ideal functionality, which directly gives generic constructions for the specific cases (OT, OSBE). Finally, we show how to instantiate the different simple building blocks in order to recover the standard efficient instantiations of these schemes from our framework. In addition to the two cases most studied (OT, OSBE), we also propose what we call *Conditioned Oblivious Transfer*, which encompasses Access Controlled Oblivious Transfer, Priced Oblivious Transfer and Conditional Oblivious Transfer, and in which the access to each line of the database is hidden behind some possibly secret restriction, be it a credential, a price, or an access policy. The advantage of the OLBE framework on the notion of Conditional Disclosure of Secrets is to allow generic constructions of a large subclass of schemes, as long as two participants are involved. It can be easily applied to any language expressing some new access control policy. Furthermore, those instantiations fit into a global security model, allowing to uniformize (for the better) the security expectations for such schemes. In particular, we allow security in the UC framework with adaptive corruptions for all our constructions (which was already known for some primitives cited above, but not all), and manage to achieve this level of security while staying in the standard model with standard hypothesis.

2 Definitions and Building Blocks

2.1 Notations for Classical Primitives

Throughout this paper, we use the notation $\mathfrak{K}$ for the security parameter.

Digital Signature. A digital signature scheme $\mathcal{S}$ [29, 36] allows a signer to produce a verifiable proof that he indeed produced a message. It is described through four algorithms $\sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. The formal definitions are given in the paper full version [10].

Encryption. An encryption scheme $\mathcal{C}$ is described through four algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$. The formal definitions are given in the paper full version [10].

Commitment and Chameleon Hash. Commitments allow a user to commit to a value without revealing it, but without the possibility to later change his mind. It is composed of four algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Commit}, \mathsf{Decommit})$. Informally, it is extractable if a simulator knowing a certain trapdoor can recover the value committed to, and it is equivocable if a simulator, knowing another trapdoor, can open the commitment to another value than the one it actually committed to. This directly echoes to Chameleon Hashes, traditionally defined by three algorithms $\mathsf{CH} = (\mathsf{KeyGen}, \mathsf{CH}, \mathsf{Coll})$. The formal definitions are given in the paper full version [10].

2.2 Identity-Based Encryption, Identity-based Key Encapsulation

Identity Based encryption was first introduced by Shamir in [55] who was expecting an encryption scheme where no public key will be needed for sending a message to a precise user, defined by his identity. Thus any user wanting to send a private message to a user only need this user's identity and a master public key. It took 17 years for the cryptographic community to find a way to realize this idea. The first instantiation was proposed in [14] by Boneh and Franklin. It can be described as an identity-based key encapsulation (IBKEM) scheme IBKEM which consists of four algorithms $\mathsf{IBKEM} = (\mathsf{Gen}, \mathsf{USKGen}, \mathsf{Enc}, \mathsf{Dec})$. Every IBKEM can be transformed into an ID-based encryption scheme IBE using a (one-time secure) symmetric cipher.

Definition 1 (Identity-based Key Encapsulation Scheme).
An identity-based key encapsulation scheme IBKEM *consists of four PPT algorithms* $\mathsf{IBKEM} = (\mathsf{Gen}, \mathsf{USKGen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following properties.*

- $\mathsf{Gen}(\mathfrak{K})$*: returns the (master) public/secret key* $(\mathsf{mpk}, \mathsf{msk})$*. We assume that* mpk *implicitly defines an identity space* $\mathcal{ID}$*, a key space* $\mathcal{KS}$*, and ciphertext space* CS*.*
- $\mathsf{USKGen}(\mathsf{msk}, \mathsf{id})$*: returns the user secret-key* $\mathsf{usk}[\mathsf{id}]$ *for identity* $\mathsf{id} \in \mathcal{ID}$*.*
- $\mathsf{Enc}(\mathsf{mpk}, \mathsf{id})$*: returns the symmetric key* $\mathsf{K} \in \mathcal{KS}$ *together with a ciphertext* $\mathsf{C} \in \mathsf{CS}$ *with respect to identity* id*.*

– $\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C})$*: returns the decapsulated key* $\mathsf{K} \in \mathcal{KS}$ *or the reject symbol* $\perp$.

For perfect correctness we require that for all $\mathfrak{K} \in \mathbb{N}$*, all pairs* $(\mathsf{mpk}, \mathsf{msk})$ *generated by* $\mathsf{Gen}(\mathfrak{K})$*, all identities* $\mathsf{id} \in \mathcal{ID}$*, all* $\mathsf{usk}[\mathsf{id}]$ *generated by* $\mathsf{USKGen}(\mathsf{msk}, \mathsf{id})$ *and all* (K, C) *output by* $\mathsf{Enc}(\mathsf{mpk}, \mathsf{id})$*:* $\Pr[\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C}) = \mathsf{K}] = 1$.

The security requirements for an IBKEM we consider here are indistinguishability and anonymity against chosen plaintext and identity attacks (IND-ID-CPA and ANON-ID-CPA). Instead of defining both security notions separately, we define pseudorandom ciphertexts against chosen plaintext and identity attacks (PR-ID-CPA) which means that challenge key and ciphertext are both pseudorandom. Note that PR-ID-CPA trivially implies IND-ID-CPA and ANON-ID-CPA. We define PR-ID-CPA-security of IBKEM formally via the games given in Figure 1.

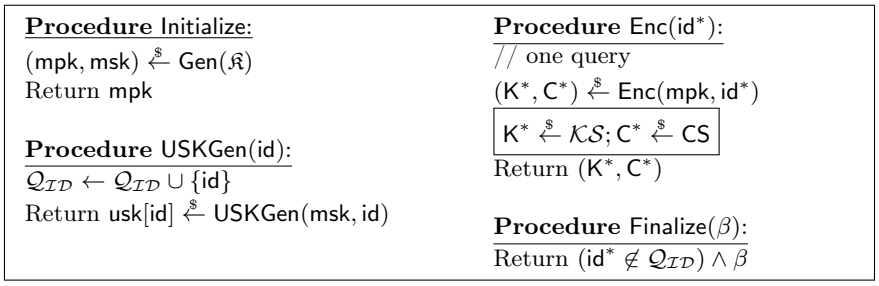


Fig. 1. Security Games $\mathsf{PR\text{-}ID\text{-}CPA}_{\mathsf{real}}$ and $\mathsf{PR\text{-}ID\text{-}CPA}_{\mathsf{rand}}$ (boxed) used for defining PR-ID-CPA-security.

Definition 2 (PR-ID-CPA Security). *An ID-based key encapsulation scheme* IBKEM *is* PR-ID-CPA*-secure if for all PPT* $\mathscr{A}$*, the following advantage is negligible:* $\mathsf{Adv}^{\mathsf{pr\text{-}id\text{-}cpa}}_{\mathsf{IBKEM}}(\mathscr{A}) := |\Pr[\mathsf{PR\text{-}ID\text{-}CPA}^{\mathscr{A}}_{\mathsf{real}} \Rightarrow 1] - \Pr[\mathsf{PR\text{-}ID\text{-}CPA}_{\mathsf{rand}}{}^{\mathscr{A}} \Rightarrow 1]|$.

2.3 Smooth Projective Hashing and Languages

Smooth projective hash functions (SPHF) were introduced by Cramer and Shoup in [26] for constructing encryption schemes. A projective hashing family is a family of hash functions that can be evaluated in two ways: using the (secret) hashing key, one can compute the function on every point in its domain, whereas using the (public) *projected* key one can only compute the function on a special subset of its domain. Such a family is deemed *smooth* if the value of the hash function on any point outside the special subset is independent of the projected key. The notion of SPHF has already found applications in various contexts in cryptography (*e.g.* [2, 33, 44]). A Smooth Projective Hash Function over a language $\mathfrak{L} \subset X$, onto a set $\mathcal{G}$, is defined by five algorithms (Setup, HashKG, ProjKG, Hash, ProjHash):

– $\mathsf{Setup}(1^{\mathfrak{K}})$ where $\mathfrak{K}$ is the security parameter, generates the global parameters param of the scheme, and the description of an $\mathcal{NP}$ language $\mathfrak{L}$;

– $\mathsf{HashKG}(\mathfrak{L}, \mathsf{param})$, outputs a hashing key hk for the language $\mathfrak{L}$;
– $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$, derives the projection key hp from the hashing key hk and the word W.
– $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$, outputs a hash value $v \in \mathcal{G}$, using the hashing key hk and the word W.
– $\mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), W, w)$, outputs the hash value $v' \in \mathcal{G}$, using the projection key hp and the witness w that the word $W \in \mathfrak{L}$.

In the following, we assume $\mathfrak{L}$ is a hard-partitioned subset of X, *i.e.* it is computationally hard to distinguish a random element in $\mathfrak{L}$ from a random element in $X \setminus \mathfrak{L}$. An SPHF should satisfy the following properties:

– *Correctness*: Let $W \in \mathfrak{L}$ and w a witness of this membership. Then, for all hashing keys hk and associated projection keys hp we have
$$\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) = \mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), W, w).$$
– *Smoothness*: For all $W \in X \setminus \mathfrak{L}$ the following distributions are statistically indistinguishable:

$$\Delta_0 = \left\{ (\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{param} = \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{hk} = \mathsf{HashKG}(\mathfrak{L}, \mathsf{param}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W), \\ v = \mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) \end{array} \right\}$$
$$\Delta_1 = \left\{ (\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{param} = \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{hk} = \mathsf{HashKG}(\mathfrak{L}, \mathsf{param}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W), v \xleftarrow{\$} \mathcal{G} \end{array} \right\}.$$

This is formalized by: $\mathsf{Adv}^{\mathsf{smooth}}_{\mathsf{SPHF}}(\mathfrak{K}) = \sum_{V \in \mathbb{G}} |\Pr_{\Delta_1}[v = V] - \Pr_{\Delta_0}[v = V]|$ is negligible.

– *Pseudo-Randomness*: If $W \in \mathfrak{L}$, then without a witness of membership the two previous distributions should remain computationally indistinguishable. For any adversary $\mathcal{A}$ within reasonable time, this advantage is negligible:

$$\mathsf{Adv}^{\mathsf{pr}}_{\mathsf{SPHF},\mathcal{A}}(\mathfrak{K}) = |\Pr_{\Delta_1}[\mathcal{A}(\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) = 1] - \Pr_{\Delta_0}[\mathcal{A}(\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) = 1]|$$

Languages. The language $\mathfrak{L} \subset X$ used in the definition of an SPHF should be a hard-partitioned subset of X, *i.e.* it is computationally hard to distinguish a random element in $\mathfrak{L}$ from a random element not in $\mathfrak{L}$ (see formal definition in [3, 33]). The languages used here are more complex and should fulfill the following properties[1]:

– *Publicly Verifiable:* Given a word x in X, anyone should be able to decide in polynomial time whether $x \in \mathfrak{L}$ or not.
– *Self-Randomizable*: Given a word in the language, anyone should be able to sample a new word in the language[2], and the distribution of this resampling should be indistinguishable from an honest distribution. This will be used in order to prevent an adversary, or the authority in charge of distributing the words, to learn which specific form of the word was used by the user.

[1] We here mainly consider languages which are hard-partitioned subsets, for instance, encryptions of publicly verifiable languages.
[2] It should be noted that this property is not incompatible with the potential secret key of the language in case it is keyed (see below).

In case we consider several languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$, we also assume it is a *Trapdoor Collection of Languages*: It is computationally hard to sample an element in $\mathfrak{L}_1 \cap \cdots \cap \mathfrak{L}_n$, except if one possesses a trapdoor tk (without the knowledge of the potential secret keys)[3]. For instance, if for all i, $\mathfrak{L}_i$ is the language of the equivocable commitments on words in an inner language $\widetilde{\mathfrak{L}}_i = \{i\}$ (as we will consider for OT), the common trapdoor key can be the equivocation trapdoor.

Depending on the applications, we can assume a *Keyed Language*, which means that it is set by a trusted authority, and that it is hard to sample fresh elements from scratch in the language without the knowledge of a secret language key $\mathsf{sk}_{\mathfrak{L}}$. In this case, the authority is also in charge of giving a word in the language to the receiver.

In case the language is keyed, we assume it is also a *Trapdoor Language*: We assume the existence of a trapdoor tk_L allowing a simulator to sample an element in $\mathfrak{L}$ (without the knowledge of the potential secret key $\mathsf{sk}_{\mathfrak{L}}$). For instance, for a language of valid Waters signatures of a message M (as we will consider for OSBE), one can think of $\mathsf{sk}_{\mathfrak{L}}$ as being the signing key, whereas the trapdoor $\mathsf{tk}_{\mathfrak{L}}$ can be the discrete logarithm of h in basis g.[4]

2.4 Security Assumptions

Due to lack of space, instantiations of the primitives recalled above are given in the paper full version [10] and we only give here the security assumptions.

Security Assumption: Pairing groups and Matrix Diffie-Hellman Assumption. Let GGen be a probabilistic polynomial time (PPT) algorithm that on input $1^{\mathfrak{K}}$ returns a description $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ of asymmetric pairing groups where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are cyclic groups of order p for a $\mathfrak{K}$-bit prime p, g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2$ is an efficiently computable (non-degenerated) bilinear map. Define $g_T := e(g_1, g_2)$, which is a generator in $\mathbb{G}_T$.

We use implicit representation of group elements as introduced in [30]. For $s \in \{1, 2, T\}$ and $a \in \mathbb{Z}_p$ define $[a]_s = g_s^a \in \mathbb{G}_s$ as the *implicit representation* of a in $\mathbb{G}_s$. More generally, for a matrix $\boldsymbol{A} = (a_{ij}) \in \mathbb{Z}_p^{n \times m}$ we define $[\boldsymbol{A}]_s$ as the implicit representation of $\boldsymbol{A}$ in $\mathbb{G}_s$:

$$[\boldsymbol{A}]_s := \begin{pmatrix} g_s^{a_{11}} & \ldots & g_s^{a_{1m}} \\ & & \\ g_s^{a_{n1}} & \ldots & g_s^{a_{nm}} \end{pmatrix} \in \mathbb{G}_s^{n \times m}$$

[3] This implicitly means that the languages are compatible, in the sense that one can indeed find a word belonging to all of them.

[4] As another example, one may think of more expressive languages which may not rely directly on generators fixed by the CRS. In this case, one can assume that the CRS contains parameters for an encryption and an associated NIZK proof system. The description of such a language is thus supplemented with an encryption of the language trapdoor, and a non-interactive zero-knowledge proof that the encrypted value is indeed a trapdoor for the said language. Using the knowledge of the decryption key, the simulator is able to recover the trapdoor.

We will always use this implicit notation of elements in $\mathbb{G}_s$, i.e., we let $[a]_s \in \mathbb{G}_s$ be an element in $\mathbb{G}_s$. Note that from $[a]_s \in \mathbb{G}_s$ it is generally hard to compute the value a (discrete logarithm problem in $\mathbb{G}_s$). Further, from $[b]_T \in \mathbb{G}_T$ it is hard to compute the value $[b]_1 \in \mathbb{G}_1$ and $[b]_2 \in \mathbb{G}_2$ (pairing inversion problem). Obviously, given $[a]_s \in \mathbb{G}_s$ and a scalar $x \in \mathbb{Z}_p$, one can efficiently compute $[ax]_s \in \mathbb{G}_s$. Further, given $[a]_1, [b]_2$ one can efficiently compute $[ab]_T$ using the pairing e. For $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^k$ define $e([\boldsymbol{a}]_1, [\boldsymbol{b}]_2) := [\boldsymbol{a}^\top \boldsymbol{b}]_T \in \mathbb{G}_T$. We recall the definition of the matrix Diffie-Hellman (MDDH) assumption [30].

Definition 3 (Matrix Distribution). *Let $k \in \mathbb{N}$. We call $\mathcal{D}_k$ a matrix distribution if it outputs matrices in $\mathbb{Z}_p^{(k+1)\times k}$ of full rank k in polynomial time.*

Without loss of generality, we assume the first k rows of $\boldsymbol{A} \xleftarrow{\$} \mathcal{D}_k$ form an invertible matrix, we denote this matrix $\overline{\boldsymbol{A}}$, while the last line is denoted $\underline{A}$. The $\mathcal{D}_k$-Matrix Diffie-Hellman problem is to distinguish the two distributions $([\boldsymbol{A}], [\boldsymbol{Aw}])$ and $([\boldsymbol{A}], [\boldsymbol{u}])$ where $\boldsymbol{A} \xleftarrow{\$} \mathcal{D}_k$, $\boldsymbol{w} \xleftarrow{\$} \mathbb{Z}_p^k$ and $\boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$.

Definition 4 ($\mathcal{D}_k$-Matrix Diffie-Hellman Assumption $\mathcal{D}_k$-MDDH). *Let $\mathcal{D}_k$ be a matrix distribution and $s \in \{1, 2, T\}$. We say that the $\mathcal{D}_k$-Matrix Diffie-Hellman ($\mathcal{D}_k$-MDDH) Assumption holds relative to* GGen *in group $\mathbb{G}_s$ if for all PPT adversaries $\mathcal{D}$,*

$$\mathbf{Adv}_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}) := |\Pr[\mathcal{D}(\mathcal{G}, [\boldsymbol{A}]_s, [\boldsymbol{Aw}]_s) = 1] - \Pr[\mathcal{D}(\mathcal{G}, [\boldsymbol{A}]_s, [\boldsymbol{u}]_s) = 1]| = \mathsf{negl}(\lambda),$$

where the probability is taken over $\mathcal{G} \xleftarrow{\$} \mathsf{GGen}(1^\lambda)$, $\boldsymbol{A} \xleftarrow{\$} \mathcal{D}_k, \boldsymbol{w} \xleftarrow{\$} \mathbb{Z}_p^k, \boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$.

For each $k \geq 1$, [30] specifies distributions $\mathcal{L}_k$, $\mathcal{U}_k$, ... such that the corresponding $\mathcal{D}_k$-MDDH assumption is the k-Linear assumption, the k-uniform and others. All assumptions are generically secure in bilinear groups and form a hierarchy of increasingly weaker assumptions. The distributions are exemplified for $k = 2$, where $a_1, \ldots, a_6 \xleftarrow{\$} \mathbb{Z}_p$.

$$\mathcal{L}_2 : \boldsymbol{A} = \begin{pmatrix} a_1 & 0 \\ 0 & a_2 \\ 1 & 1 \end{pmatrix} \quad \mathcal{U}_2 : \boldsymbol{A} = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \\ a_5 & a_6 \end{pmatrix}.$$

It was also shown in [30] that $\mathcal{U}_k$-MDDH is implied by all other $\mathcal{D}_k$-MDDH assumptions.

Lemma 5 (Random self reducibility [30]). *For any matrix distribution $\mathcal{D}_k$, $\mathcal{D}_k$-MDDH is random self-reducible. In particular, for any $m \geq 1$ and for all PPT adversaries $\mathcal{D}$ and $\mathcal{D}'$,*

$$\mathbf{Adv}_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}) + \tfrac{1}{q-1} \geq \mathbf{Adv}^m_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}')$$

where $\mathbf{Adv}^m_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}') := \Pr[\mathcal{D}'(\mathcal{G}, [\boldsymbol{A}], [\boldsymbol{AW}]) \Rightarrow 1] - \Pr[\mathcal{D}'(\mathcal{G}, [\boldsymbol{A}], [\boldsymbol{U}]) \Rightarrow 1]$, with $\mathcal{G} \leftarrow \mathsf{GGen}(1^\lambda)$, $\boldsymbol{A} \xleftarrow{\$} \mathcal{D}_k, \boldsymbol{W} \xleftarrow{\$} \mathbb{Z}_p^{k\times m}, \boldsymbol{U} \xleftarrow{\$} \mathbb{Z}_p^{(k+1)\times m}$.

Remark: It should be noted that $\mathcal{L}_1, \mathcal{L}_2$ are respectively the SXDH and DLin assumptions.

2.5 Security Models

UC Framework. The goal of the UC framework [21] is to ensure that UC-secure protocols will continue to behave in the ideal way even if executed in a concurrent way in arbitrary environments. It is a simulation-based model, relying on the indistinguishability between the real world and the ideal world. In the ideal world, the security is provided by an ideal functionality $\mathcal{F}$, capturing all the properties required for the protocol and all the means of the adversary. In order to prove that a protocol Π emulates $\mathcal{F}$, one has to construct, for any polynomial adversary $\mathscr{A}$ (which controls the communication between the players), a simulator $\mathscr{S}$ such that no polynomial environment $\mathcal{Z}$ can distinguish between the real world (with the real players interacting with themselves and $\mathscr{A}$ and executing the protocol π) and the ideal world (with dummy players interacting with $\mathscr{S}$ and $\mathcal{F}$) with a significant advantage. The adversary can be either *adaptive*, *i.e.* allowed to corrupt users whenever it likes to, or *static*, *i.e.* required to choose which users to corrupt prior to the execution of the session sid of the protocol. After corrupting a player, $\mathscr{A}$ has complete access to the internal state and private values of the player, takes its entire control, and plays on its behalf.

Simple UC Framework. Canetti, Cohen and Lindell formalized a simpler variant in [22], that we use here. This simplifies the description of the functionalities for the following reasons (in a nutshell): All channels are automatically assumed to be authenticated (as if we worked in the $\mathcal{F}_{\text{AUTH}}$-hybrid model); There is no need for *public delayed outputs* (waiting for the adversary before delivering a message to a party), neither for an explicit description of the corruptions. We refer the interested reader to [22] for details.

3 UC-secure Adaptive Oblivious Transfer

As explained in the introduction, the classical OT constructions based on the commitment/SPHF paradigm (with so-called implicit decommitment), among the latest in the UC framework [1, 9, 24], require the server to send an encryption of the complete database for each line required by the user (thus $O(n)$ each time). We here give a protocol requiring $O(\log(n))$ for each line (except the first one, still in $O(n)$), in the UC framework with adaptive corruptions under classical assumptions (MDDH). This protocol builds upon the more efficient known scheme secure in the UC framework [9] and we use ideas from [37] to make it adaptive.

Using implicit decommitment in the UC framework implies a very strong commitment primitive (formalized as SPHF-friendly commitments in [1]), which is both extractable and equivocable. Our idea is here to split these two properties by using on the one hand an equivocable commitment and on the other hand an (extractable) CCA encryption scheme by generalizing the way to access a line in the database. But this is infeasible with simple line numbers. Indeed, we suggest here not to consider anymore the line numbers as numbers in $\{1, \ldots, n\}$ but rather to "encode" them (the exact encoding will depend on the protocol):

For every line i, a word W_i in the language $\mathfrak{L}_i$ will correspond to a representation of line i. This representation must be publicly verifiable, in the sense that anyone can associate i to a word W_i. We formalize this in the following definition of oblivious transfer[5], given without loss of generality[6] (the classical notion of OT being easily captured using $\mathfrak{L}_i = \{i\}$).

3.1 Definition and Security Model for Oblivious Transfer

In such a protocol, a server $\mathcal{S}$ possesses a database of n lines $(m_1, \ldots, m_n) \in (\{0,1\}^{\mathfrak{K}})^n$. A user $\mathcal{U}$ will be able to recover m_k (in an oblivious way) as soon as he owns a word $W_k \in \mathfrak{L}_k$. The languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$ will be assumed to be a trapdoor collection of languages, publicly verifiable and self-randomizable. As we consider simulation-based security (in the UC framework), we allow a simulated setup SetupT to be run instead of the classical setup Setup in order to allow the simulator to possess some trapdoors. Those two setup algorithms should be indistinguishable.

Definition 6 (Oblivious Transfer). *An* OT *scheme is defined by five algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{DBGen}, \mathsf{Samp}, \mathsf{Verify})$*, along with an interactive protocol* $\mathsf{Protocol}\langle\mathcal{S},\mathcal{U}\rangle$*:*

- $\mathsf{Setup}(1^{\mathfrak{K}})$*, where* $\mathfrak{K}$ *is the security parameter, generates the global parameters* param*, among which the number* n*;*

or $\mathsf{SetupT}(1^{\mathfrak{K}})$*, where* $\mathfrak{K}$ *is the security parameter, additionally allows the existence*[7] *of a trapdoor* tk *for the collection of languages* $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$*.*

- $\mathsf{KeyGen}(\mathsf{param}, \mathfrak{K})$ *generates, for all* $i \in \{1, \ldots, n\}$*, the description of the language* $\mathfrak{L}_i$ *(as well as the language key* $\mathsf{sk}_{\mathfrak{L}_i}$ *if need be). If the parameters* param *were defined by* SetupT*, this implicitly also defines the common trapdoor* tk *for the collection of languages* $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$*.*
- $\mathsf{Samp}(\mathsf{param})$ *or* $\mathsf{Samp}(\mathsf{param}, (\mathsf{sk}_{\mathfrak{L}_i})_{i\in\{1,\ldots,n\}})$ *generates a word* $W_i \in \mathfrak{L}_i$*;*
- $\mathsf{Verify}_i(W_i, \mathfrak{L}_i)$ *checks whether* W_i *is a valid word in the language* $\mathfrak{L}_i$*. It outputs* 1 *if the word is valid,* 0 *otherwise;*
- $\mathsf{Protocol}\langle\mathcal{S}((\mathfrak{L}_1, \ldots, \mathfrak{L}_n), (m_1, \ldots, m_n)), \mathcal{U}((\mathfrak{L}_1, \ldots, \mathfrak{L}_n), W_i)\rangle$*, which is executed between the server* $\mathcal{S}$ *with the private database* $(m_1, \ldots, m_n)$ *and corresponding languages* $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$*, and the user* $\mathcal{U}$ *with the same languages and the word* W_i*, proceeds as follows. If the algorithm* $\mathsf{Verify}_i(W_i, \mathfrak{L}_i)$ *returns* 1*, then* $\mathcal{U}$ *receives* m_k*, otherwise it does not. In any case,* $\mathcal{S}$ *does not learn anything.*

The ideal functionality of an Oblivious Transfer (OT) protocol was given in [1,21,24], and an adaptive version in [38]. We here combine them and rewrite

[5] The adaptive version only implies that the database $(m_1, \ldots, m_n)$ is sent only once in the interaction, while the user can query several lines (*i.e.* several words), in an adaptive way.

[6] This formalization furthermore encompasses the variants of OT, such as conditioned OT, where a user accesses a line only if he knows a credential for this line.

[7] The specific trapdoor will depend on the languages and be computed in the KeyGen algorithm.

The functionality $\mathcal{F}_{\mathrm{OT}}^{\mathfrak{L}}$ is parametrized by a security parameter $\mathfrak{K}$ and a set of languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$ along with the corresponding public verification algorithms $(\mathsf{Verify}_1, \ldots, \mathsf{Verify}_n)$. It interacts with an adversary $\mathscr{S}$ and a set of parties $\mathfrak{P}_1, \ldots, \mathfrak{P}_N$ via the following queries:

- **Upon receiving from party $\mathfrak{P}_i$ an input $(\mathtt{NewDataBase}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \ldots, m_n))$** , with $m_k \in \{0,1\}^{\mathfrak{K}}$ for all k: record the tuple $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \ldots, m_n))$ and reveal $(\mathtt{Send}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to the adversary $\mathscr{S}$. Ignore further $\mathtt{NewDataBase}$-message with the same ssid from $\mathfrak{P}_i$.
- **Upon receiving an input $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, W_k)$ from party $\mathfrak{P}_j$**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \ldots, m_n))$ is not recorded. Otherwise, reveal $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to the adversary $\mathscr{S}$ and send the message $(\mathtt{Received}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, m'_k)$ to $\mathfrak{P}_j$ where $m'_k = m_k$ if $\mathsf{Verify}_k(W_k, \mathfrak{L}_k)$ returns 1, and $m'_k = \perp$ otherwise.

(Non-Adaptive case: Ignore further $\mathtt{Receive}$-message with the same ssid from $\mathfrak{P}_j$.)

Fig. 2. Ideal Functionality for (Adaptive) Oblivious Transfer $\mathcal{F}_{\mathrm{OT}}^{\mathfrak{L}}$

it in simple UC and using our language formalism (instead of directly giving a number line s to the functionality, the user will give it a word $W_s \in \mathfrak{L}_s$). The resulting functionality $\mathcal{F}_{\mathrm{OT}}^{\mathfrak{L}}$ is given in Figure 2. Recall that there is no need to give an explicit description of the corruptions in the simple version of UC [22].

3.2 High Level Idea of the Construction of the Adaptive Oblivious Transfer Scheme

Our construction builds upon the UC-secure OT scheme from [9], with ideas inspired from [37], who propose a neat framework allowing to achieve *adaptive* Oblivious Transfer (but not in the UC framework). Their construction is quite simple: It requires a *blind Identity-Based Encryption*, in other words, an IBE scheme in which there is a way to query for a user key generation without the authority (here the server) learning the targeted identity (here the line in the database). Once such a Blind IBE is defined, one can conveniently obtain an oblivious transfer protocol by asking the database to encrypt (once and for all) each line for an identity (the j-th line being encrypted for the identity j), and having the user do a blind user key generation query for identity i in order to recover the key corresponding to the line i he expects to learn.

This approach is round-optimal: After the database preparation, the first flow is sent by the user as a commitment to the identity i, and the second one is sent by the server with the blinded expected information. But several technicalities arise because of the UC framework we consider here. For instance, the blinded expected information has to be masked, we do this here thanks to an SPHF. Furthermore, instead of using simple line numbers as identities, we have to commit to words in specific languages (so as to ensure extractability and equivocability) as well as to *fragment* the IBE keys into bits in order to achieve $O(\log n)$ in both flows. This allows us to achieve the first UC-secure adaptive OT protocol allowing adaptive corruptions. More details follow in the next sessions.

3.3 Main Building Block: Constructing a Blind Fragmented IBKEM from an IBKEM

Definition and Security Properties of a Blind IBKEM Scheme. Following [12], we recalled in Section 2.2 page 6 the definitions, notations and security properties for an IBE scheme, seen as an Identity-Based Key Encapsulation (IBKEM) scheme. We continue to follow the KEM formalism by adapting the definition of a Blind IBE scheme given in [37] to this setting.

Definition 7 (Blind Identity-Based Key Encapsulation Scheme). *A Blind Identity-Based Key Encapsulation scheme* $\mathsf{BlindIBKEM}$ *consists of four PPT algorithms* $(\mathsf{Gen}, \mathsf{BlindUSKGen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following properties:*

- Gen, Enc *and* Dec *are defined as for a traditional* IBKEM *scheme.*
- $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(\mathcal{U}, \mathsf{id}, \ell; \boldsymbol{\rho})\rangle)$ *is an interactive protocol, in which an honest user* $\mathcal{U}$ *with identity* $\mathsf{id} \in \mathcal{ID}$ *obtains the corresponding user secret key* $\mathsf{usk}[\mathsf{id}]$ *from the master authority* $\mathcal{S}$ *or outputs an error message, while* $\mathcal{S}$*'s output is nothing or an error message (*ℓ *is a label and* $\boldsymbol{\rho}$ *the randomness).*

Defining the security of a $\mathsf{BlindIBKEM}$ requires two additional properties, stated as follows (see [37, pages 6 and 7] for the formal security games):

1. **Leak-free Secret Key Generation** (called Leak-free Extract for Blind IBE security in the original paper): A potentially malicious user cannot learn anything by executing the $\mathsf{BlindUSKGen}$ protocol with an honest authority which he could not have learned by executing the USKGen protocol with an honest authority; Moreover, as in USKGen, the user must know the identity for which he is extracting a key.
2. **Selective-failure Blindness**: A potentially malicious authority cannot learn anything about the user's choice of identity during the $\mathsf{BlindUSKGen}$ protocol; Moreover, the authority cannot cause the $\mathsf{BlindUSKGen}$ protocol to fail in a manner dependent on the user's choice.

   For our applications, we only need a weakened property for blindness:[8]
3. **Weak Blindness**: A potentially malicious authority cannot learn anything about the user's choice of identity during the $\mathsf{BlindUSKGen}$ protocol.

High-Level Idea of the Transformation. We now show how to obtain a $\mathsf{BlindIBKEM}$ scheme from any IBKEM scheme. From a high-level point of view, this transformation mixes two pre-existing approaches.

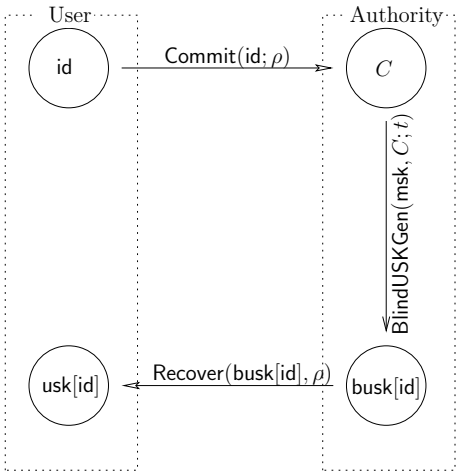
First, we are going to consider a reverse Naor transform [14, 27]: He drew a parallel between Identity-Based Encryption schemes and signature schemes, by showing that a user secret key on an identity can be viewed as the signature

[8] Two things to note: First, Selective Failure would be considered as a Denial of Service in the Oblivious Transfer setting. Then, we do not restrict ourselves to schemes where the blindness adversary has access to the generated user keys, as reliable erasures in the OT protocol provide us a way to forget them before being corrupted (otherwise we would need to use a randomizable base IBE).

on this identity, the verification process therefore being a test that any chosen valid ciphertext for the said identity can indeed be decrypted using the *signature* scheme.

Then, we are going to use Fischlin [31] round-optimal approach to blind signatures, where the whole interaction is done in one pass: First, the user commits to the message, then he recovers a signature linked to his commitment. For sake of simplicity, instead of using a Non-Interactive Zero-Knowledge Proof of Knowledge of a signature, we are going to follow the [11, 13] approach, where thanks to an additional term, the user can extract a signature on the identity from a signature on the committed identity.

Omitting technical details described more precisely in the following sections, the main idea of the transformation of the IBKEM scheme in order to blind a user key request is described in Figure 3.



1. A user commits to the targeted identity id using some randomness ρ.
2. The authority possesses an algorithm allowing it to generate keys for committed identities using its master secret key msk, and some randomness t, in order to obtain a blinded user secret key busk[id].
3. The user using solely the randomness used in the initial commitment is able to recover the requested secret key from the authority's generated value.

Fig. 3. Generic Transformation of an IBKEM into a Blind IBKEM (naive approach)

Generic Transformation of an IBKEM into a Blind IBKEM. It now remains to explain how one can fulfill the idea highlighted in Figure 3. The technique to blind a user key request uses a smooth projective hash function (see Section 2.3), and is often called *implicit decommitment* in recent works: the IBKEM secret key is sent hidden in such a way that it can only be recovered if the user knows how to open the initial commitment on the correct identity. We assume the existence of a labeled CCA-encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\mathrm{cca}}, \mathsf{KeyGen}_{\mathrm{cca}}, \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}, \mathsf{Decrypt}^{\ell}_{\mathrm{cca}})$ compatible with an SPHF defined by (Setup, HashKG, ProjKG, Hash, ProjHash) onto a set G (where ℓ is a label defined by the global protocol). By "compatible", we mean that the SPHF can be defined over a language $\mathfrak{L}^c_{\mathsf{id}} \subset X$, where $\mathfrak{L}^c_{\mathsf{id}} = \{C \mid \exists \boldsymbol{\rho} \text{ such that } C = \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}(\mathsf{id}; \boldsymbol{\rho})\}$. In the KeyGen algorithm, the description of the language $\mathfrak{L}_{\mathsf{id}} = \{\mathsf{id}\}$ thus implic-

itly defines the language $\mathfrak{L}^c_{\mathsf{id}}$ of CCA-encryptions of elements of $\mathfrak{L}_{\mathsf{id}}$. We additionally use a key derivation function KDF to derive a pseudo-random bit-string $K \in \{0,1\}^{\mathfrak{K}}$ from a pseudo-random element $v \in G$. One can use the Leftover-Hash Lemma [40], with a random seed defined in param during the global setup, to extract the entropy from v, then followed by a pseudo-random generator to get a long enough bit-string. Many uses of the same seed in the Leftover-Hash Lemma just lead to a security loss linear in the number of extractions. This gives the following protocol for BlindUSKGen, described in Figure 4.

- The user computes an encryption of the expected identity id and keeps the randomness ρ: $C = \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}(\mathsf{id}; \boldsymbol{\rho})\}$.
- For every identity id', the server computes the key $\mathsf{usk}[\mathsf{id}']$ along with a pair of (secret, public) hash keys $(\mathsf{hk}_{\mathsf{id}'}, \mathsf{hp}_{\mathsf{id}'})$ for a smooth projective hash function on the language $\mathfrak{L}^c_{\mathsf{id}'}$:
  $\mathsf{hk}_{\mathsf{id}'} = \mathsf{HashKG}(\ell, \mathfrak{L}^c_{\mathsf{id}'}, \mathsf{param})$ and $\mathsf{hp}_{\mathsf{id}'} = \mathsf{ProjKG}(\mathsf{hk}_{\mathsf{id}'}, \ell, (\mathfrak{L}^c_{\mathsf{id}'}, \mathsf{param}), \mathsf{id}')$.
  He also compute the corresponding hash value
  $$H_{\mathsf{id}'} = \mathsf{Hash}(\mathsf{hk}_{\mathsf{id}'}, (\mathfrak{L}^c_{\mathsf{id}'}, \mathsf{param}), (\ell, C)).$$
  Finally, he sends $(\mathsf{hp}_{\mathsf{id}'}, \mathsf{usk}[\mathsf{id}'] \oplus \mathtt{KDF}(H_{\mathsf{id}'}))$ for every id', where $\oplus$ is a compatible operation.
- Thanks to $\mathsf{hp}_{\mathsf{id}}$, the user is able to compute the corresponding projected hash value $H'_{\mathsf{id}} = \mathsf{ProjHash}(\mathsf{hp}_{\mathsf{id}}, (\mathfrak{L}^c_{\mathsf{id}}, \mathsf{param}), (\ell, C), \boldsymbol{\rho})$. He then recovers $\mathsf{usk}[\mathsf{id}]$ for the initially committed identity id since $H_{\mathsf{id}} = H'_{\mathsf{id}}$.

Fig. 4. Summary of the Generic Construction of $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(\mathcal{U}, \mathsf{id}, \ell; \boldsymbol{\rho})\rangle)$ for a blind IBE

Theorem 8. *If* IBKEM *is a* PR-ID-CPA*-secure identity-based key encapsulation scheme and $\mathcal{E}$ a labeled* CCA*-encryption scheme compatible with an* SPHF*, then* BlindIBKEM *is leak free and weak blind.*

Proof. First, BlindIBKEM satisfies leak-free secret key generation since it relies on the CCA security on the encryption scheme, forbidding a user to open it to another identity than the one initially encrypted. Furthermore, the pseudo-randomness of the SPHF ensures that the blinded user key received for id is indistinguishable from random if he encrypted $\mathsf{id}' \neq \mathsf{id}$. Finally, the weak blindness also relies on the CCA security on the encryption scheme, since an encryption of id is indistinguishable from a encryption of $\mathsf{id}' \neq \mathsf{id}$. □

Using a Blind IBKEM in our Application to Adaptive Oblivious Transfer. The previous approach allows to transform an IBKEM into a Blind IBKEM, but it has a huge drawback in our context: Since we assume an exponential identity space, it requires an exponential number of answers from the authority, which cannot help us to fulfill logarithmic complexity in our application. However, if we focus on the special case of affine IBE with bitwise function[9], a user

[9] They were defined in [12]. Affine IBE derive their name from the fact that only affine operations are done on the identity bits (no hashing, square rooting, inverting... are allowed).

key can be described as the list $(\mathsf{usk}[0], \mathsf{usk}[0, \mathsf{id}_0], \ldots, \mathsf{usk}[m-1, \mathsf{id}_{m-1}])$ if id_i is the i-th bit of the identity id. One can thus manage to be much more efficient by sending each "bit" evaluation on the user secret key, hidden with a smooth projective hash value on the language "the i-th bit of the identity is a 0 (or 1)", which is common to all identities. We can thus reduce the number of languages from the number of identities (which is exponential) to the length of an identity (which is polynomial). For security reasons, one cannot give directly the evaluation value, but as we are considering the sum of the evaluations for each bit, we simply add a Shamir-like secret sharing, by adding randomness that is going to cancel out at the end.

- The user computes a bit-per-bit encryption of the expected identity id and keeps the randomness ρ: $C = \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}(\mathsf{id}; \boldsymbol{\rho})\}$.
- The server computes a fragmented version of all the keys $\mathsf{usk}[\mathsf{id}']$, *i.e.* all the values $\mathsf{usk}[i, b]$ for i from 0 up to the length m of the keys and $b \in \{0, 1\}$. He also computes a pair of (secret, public) hash keys $(\mathsf{hk}_{i,b}, \mathsf{hp}_{i,b})$ for a smooth projective hash function on the language $\mathfrak{L}^c_{i,b}$: "The i-th bit of the value encrypted into C is b", *i.e.* $\mathsf{hk}_{i,b} = \mathsf{HashKG}(\ell, \mathfrak{L}^c_{i,b}, \mathsf{param})$ and $\mathsf{hp}_{i,b} = \mathsf{ProjKG}(\mathsf{hk}_{i,b}, \ell, (\mathfrak{L}^c_{i,b}, \mathsf{param}))$. He also computes the corresponding hash value $H_{i,b} = \mathsf{Hash}(\mathsf{hk}_{i,b}, (\mathfrak{L}^c_{i,b}, \mathsf{param}), (\ell, C))$ and chooses random values z_i. Finally, he sends, for each (i, b), $(\mathsf{hp}_{i,b}, \mathsf{busk}[i, b])$, where $\mathsf{busk}[i, b] = \mathsf{usk}[i, b] \oplus \mathtt{KDF}(H_{i,b}) \oplus z_i$, together with $Z = \mathsf{usk}_0 \ominus \left(\bigoplus_i z_i\right)$, where $\oplus$ is a compatible operation and $\ominus$ its inverse.
- Thanks to the $\mathsf{hp}_{i,\mathsf{id}_i}$ for the initially committed identity id, the user is able to compute the corresponding projected hash value
$$H'_{i,\mathsf{id}_i} = \mathsf{ProjHash}(\mathsf{hp}_{i,\mathsf{id}_i}, (\mathfrak{L}^c_{i,\mathsf{id}_i}, \mathsf{param}), (\ell, C), \boldsymbol{\rho}),$$
that should be equal to H_{i,id_i} for all i. From the values $\mathsf{busk}[i, \mathsf{id}_i]$, he then recovers $\mathsf{usk}[i, \mathsf{id}_i] \oplus z_i$. Finally, with the operation $\left(\bigoplus_i (\mathsf{usk}[i, \mathsf{id}_i] \oplus z_i)\right) \oplus Z$, he recovers the expected $\mathsf{usk}[\mathsf{id}]$.

Fig. 5. Summary of the Generic Construction of $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(\mathcal{U}, \mathsf{id}, \ell; \boldsymbol{\rho})\rangle)$ for a Blind affine IBE

As a last step, we finally need to make our construction compatible with the UC framework with adaptive corruptions. In this context, interactions should make sense for any possible input chosen by the environment and learnt a posteriori in the simulation during the corruption of an honest party. From the user side, this implies that the last flow should contain enough recoverable information so that a simulator, having sent a commitment to an incorrect identity, can extract the proper user secret key corresponding to the correct identity recovered after the corruption. From the server side, this implies that the IBKEM scheme is defined such as one is able to adapt the user secret keys in order to correspond to the new database learnt a posteriori. Of course, not all schemes allow this property, but this will be the case in the pairing scenario considered in our concrete instantiation.

To deal with corruptions of the user, recall that a simulated server (knowing the secret key of the encryption scheme) is already able to extract the identity

committed to. But we now consider that, for all id, $\mathfrak{L}_{\mathsf{id}}$ is the language of the equivocable commitments on words in the inner language $\widetilde{\mathfrak{L}}_{\mathsf{id}} = \{\mathsf{id}\}$. We assume them to be a *Trapdoor Collection of Languages*, which means that it is computationally hard to sample an element in $\mathfrak{L}_1 \cap \cdots \cap \mathfrak{L}_n$, except for the simulator, who possesses a trapdoor tk (the equivocation trapdoor) allowing it to sample an element in the intersection of languages. This allows a simulated user (knowing this trapdoor) not to really bind to any identity during the commitment phase. The only difference with the algorithm described in Figure 5 is that the user now encrypts this word W (which is an equivocable commitment on his identity id) rather than directly encrypting his identity id: $C = \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}(W; \boldsymbol{\rho})$. This technique is also explained as an application of our OLBE framework, in the paper full version [10]. We will directly prove this protocol during the proof of the oblivious transfer scheme.

3.4 Generic Construction of Adaptive OT

We derive from here our generic construction of OT (depicted in Figure 6). We additionally assume the existence of a Pseudo-Random Generator (PRG) F with input size equal to the plaintext size, and output size equal to the size of the messages in the database and an $\mathsf{IND\text{-}CPA}$ encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\mathrm{cpa}}, \mathsf{KeyGen}_{\mathrm{cpa}}, \mathsf{Encrypt}_{\mathrm{cpa}}, \mathsf{Decrypt}_{\mathrm{cpa}})$ with plaintext size at least equal to the security parameter. First, the owner of the database generates the keys for such an IBE scheme, and encrypts each line i of the database for the identity i. Then when a user wants to request a given line, he runs the blind user key generation algorithm and recovers the key for the expected given line. This leads to the following security result, proven in the paper full version [10].

Theorem 9. *Assuming that* $\mathsf{BlindUSKGen}$ *is constructed as described above, the adaptive Oblivious Transfer protocol described in Figure 6 UC-realizes the functionality* $\mathcal{F}_{OT}^{\mathfrak{L}}$ *presented in Figure 2 with adaptive corruptions assuming reliable erasures.*

3.5 Pairing-Based Instantiation of Adaptive OT

Affine Bit-Wise Blind IBE. In [12], the authors propose a generic framework to move from affine Message Authentication Code to IBE, and they propose a tight instantiation of such a MAC, giving an affine bit-wise IBE, which seems like a good candidate for our setting (making it blind and fragmented).

We are thus going to use the family of IBE described in the following picture (Figure 7), which is their instantiation derived from a Naor-Reingold MAC[10]. In the following, $h_i()$ are injective deterministic public functions mapping a bit to a scalar in $\mathbb{Z}_p$.

[10] For the reader familiar with the original result, we combine $x, \boldsymbol{y}$ into a bigger $\boldsymbol{y}$ to lighten the notations, and compact the (x'_i, y'_i) values into a single y' as this has no impact on their construction.

CRS generation:
$\mathsf{crs} \xleftarrow{\$} \mathsf{SetupCom}(1^{\mathfrak{K}}), \mathsf{param}_{\mathrm{cpa}} \xleftarrow{\$} \mathsf{Setup}_{\mathrm{cpa}}(1^{\mathfrak{K}})$.
Database Preparation:

1. Server runs $\mathsf{Gen}(\mathfrak{K})$, to obtain $\mathsf{mpk}, \mathsf{msk}$.
2. For each line t, he computes $(D_t, K_t) = \mathsf{Enc}(\mathsf{mpk}, t)$, and $L_t = K_t \oplus DB(t)$.
3. He also computes $\mathsf{usk}[i, b]$ for all $i = 1 \ldots, m$ and $b = 0, 1$ and erases msk.
4. Server generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by KeyGen.
5. He then publishes $\mathsf{mpk}, \{(D_t, L_t)\}_t, \mathsf{pk}$.

Index query on s:

1. User chooses a random value S, computes $R \leftarrow F(S)$ and encrypts S under pk: $c \xleftarrow{\$} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, S)$
2. User computes $\mathcal{C}$ with the first flow of $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(\mathcal{U}, s, \ell; \boldsymbol{\rho})\rangle)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{U}, \mathcal{S})$ (see Figure 5).
3. User stores the random $\boldsymbol{\rho}_s = \{\boldsymbol{\rho}_*\}$ needed to open $\mathcal{C}$ to s, and completely erases the rest, including the random coins used by $\mathsf{Encrypt}_{\mathrm{cpa}}$ and sends $(c, \mathcal{C})$ to the Server

IBE input msk:

1. Server decrypts $S \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and computes $R \leftarrow F(S)$
2. Server runs the second flow of $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(\mathcal{U}, s, \ell; \boldsymbol{\rho})\rangle)$ on $\mathcal{C}$ (see Figure 5).
3. Server erases every new value except $(\mathsf{hp}_{i,b})_{i,b}, (\mathsf{busk}[i, b])_{i,b}, Z \oplus R$ and sends them over a secure channel.

Data recovery:

1. User then using, $\boldsymbol{\rho}_s$ recovers $\mathsf{usk}[s]$ from the values received from the server.
2. He can then recover the expected information with $\mathsf{Dec}(\mathsf{usk}[s], s, D_s) \oplus L_s$ and erases everything else.

Fig. 6. Adaptive UC-Secure 1-out-of-n OT from a Fragmented Blind IBE

A property that was not studied in this paper was the blind user key generation: How to generate and answer blind user secret key queries? We answer to this question by proposing the $k-\mathsf{MDDH}$-based variation presented in Figure 8. To fit the global framework we are going to consider the equivocable language of each chameleon hash of the identity bits $(\boldsymbol{a}_i, \boldsymbol{b}_{i,m_i})$, and then a Cramer-Shoup like encryption of $\boldsymbol{b}$ into $\boldsymbol{d}$ (more details in the paper full version [10]). We denote this process as Har in the following protocol, and by $\mathfrak{L}_{\mathsf{Har},i,\mathsf{id}_i}$ the language on identity bits. We thus obtain the following security results.

Theorem 10. *This construction achieves both the weak Blindness, and the leak-free secret key generation requirements under the $k-\mathsf{MDDH}$ assumption.*

The first one is true under the indistinguishability of the generalized Cramer-Shoup encryption recalled in the paper full version [10], as the server learns nothing about the line requested during the first flow. It should even be noted

$\underline{\mathsf{Gen}(\mathfrak{K})}$:

$\boldsymbol{A} \xleftarrow{\$} \mathcal{D}_k, \boldsymbol{B} = \overline{\boldsymbol{A}}$
For $i \in [\![0, \ell]\!] : \boldsymbol{Y}_i \xleftarrow{\$} \mathbb{Z}_p^{k+1}; \boldsymbol{Z}_i = \boldsymbol{Y}_i^\top \cdot \boldsymbol{A} \in \mathbb{Z}_p^k$
$\boldsymbol{y}' \xleftarrow{\$} \mathbb{Z}_p^{k+1}; \boldsymbol{z}' = \boldsymbol{y}'^\top \cdot \boldsymbol{A} \in \mathbb{Z}_q^k$
$\mathsf{mpk} := (\mathcal{G}, [\boldsymbol{A}]_1, ([\boldsymbol{Z}_i]_1)_{i \in [\![0,\ell]\!]}, [\boldsymbol{z}']_1)$
$\mathsf{msk} := (\boldsymbol{Y}_i)_{i \in [\![0,\ell]\!]}, \boldsymbol{y}'$
Return $(\mathsf{mpk}, \mathsf{msk})$

$\underline{\mathsf{USKGen}(\mathsf{msk}, \mathsf{id})}$:

$\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^k, \boldsymbol{t} = \boldsymbol{B}\boldsymbol{s}$
$\boldsymbol{w} = (\boldsymbol{Y}_0 + \sum_{i=1}^{\ell} h_i(\mathsf{id}_i)\boldsymbol{Y}_i)\boldsymbol{t} + \boldsymbol{y}' \in \mathbb{Z}_p^{k+1}$
Return $\mathsf{usk}[\mathsf{id}] := ([\boldsymbol{t}]_2, [\boldsymbol{w}]_2) \in \mathbb{G}_2^{k+k+1}$

$\underline{\mathsf{Enc}(\mathsf{mpk}, \mathsf{id})}$:

$\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_p^k$
$\boldsymbol{c}_0 = \boldsymbol{A}\boldsymbol{r} \in \mathbb{Z}_p^{k+1}$
$\boldsymbol{c}_1 = (\boldsymbol{Z}_0 + \sum_{i=1}^{\ell} h_i(\mathsf{id}_i)\boldsymbol{Z}_i) \cdot \boldsymbol{r} \in \mathbb{Z}_p$
$K = \boldsymbol{z}' \cdot \boldsymbol{r} \in \mathbb{Z}_p$.
Return $[K]_T$
and $\mathsf{C} = ([\boldsymbol{c}_0]_1, [\boldsymbol{c}_1]_1) \in \mathbb{G}_1^{k+1+1}$

$\underline{\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C})}$:

Parse $\mathsf{usk}[\mathsf{id}] = ([\boldsymbol{t}]_2, [\boldsymbol{w}]_2)$
Parse $\mathsf{C} = ([\boldsymbol{c}_0]_1, [\boldsymbol{c}_1]_1)$
$K = e([\boldsymbol{c}_0]_1, [\boldsymbol{w}]_2) \cdot e([\boldsymbol{c}_1]_1, [\boldsymbol{t}]_2)^{-1}$
Return $K \in \mathbb{G}_T$

Fig. 7. A fragmentable affine IBKEM.

– First flow: $\underline{\mathcal{U} \text{ starts by computing}}$
  $\boldsymbol{\rho} \xleftarrow{\$} \mathbb{Z}_p^{1+4\times\ell}$,
  $\boldsymbol{a}, \boldsymbol{d} = \mathsf{Har}(\mathsf{id}, \ell; \rho) \in \mathbb{Z}_p^\ell \times \mathbb{Z}_p^{2\times(k+3)\ell}$,
  Sends $\mathcal{C} = ([\boldsymbol{a}]_1, [\boldsymbol{d}]_2)$ to S
– Second Flow: $\underline{\mathcal{S} \text{ then proceeds}}$
  $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^k, \boldsymbol{t} = \boldsymbol{B}\boldsymbol{s}$, $\boldsymbol{f} \xleftarrow{\$} \mathbb{Z}_p^{\ell\times k+1}$,
  For each $i \in [\![1, \lceil \log n \rceil]\!], b \in [\![0, 1]\!]$:
    $\mathsf{hk}_{i,b} = \mathsf{HashKG}(\mathfrak{L}_{\mathsf{Har},i,b}, \boldsymbol{C})$
    $\mathsf{hp}_{i,b} = \mathsf{ProjKG}(\mathsf{hk}_{i,b}, \mathfrak{L}_{\mathsf{Har},i,b}, \boldsymbol{C})$
    $H_{i,b} = \mathsf{Hash}(\mathsf{hk}_{i,b}, \mathfrak{L}_{\mathsf{Har},i,b}, \boldsymbol{C})$
    $\boldsymbol{\omega}_{i,b} = (b\boldsymbol{Y}_i)\boldsymbol{t} + \boldsymbol{f}_i + H_{i,b}$

Then sets $\boldsymbol{w}_0 = \boldsymbol{Y}_0\boldsymbol{t} + \boldsymbol{y}' - \sum_{i=1}^{\ell} \boldsymbol{f}_i \in \mathbb{Z}_p^{k+1}$
Returns $\mathsf{busk} := ([\boldsymbol{t}]_2, [\boldsymbol{w}_0]_2, \{[\boldsymbol{\omega}_{i,b}]_2\}, \{[\mathsf{hp}_{i,b}]_2\})$
– $\mathsf{BlindUSKGen}_3$:
  $\underline{\mathcal{U} \text{ then recovers his key}}$
  For each $i \in [\![1, \ell]\!]$:
    $H'_i = \mathsf{ProjHash}(\mathsf{hp}_{i,\mathsf{id}_i}, \mathfrak{L}_{\mathsf{Har},i,\mathsf{id}_i}, \boldsymbol{C}, \rho_i)$
    $\boldsymbol{w}_i = \omega_{i,\mathsf{id}_i} - H'_i$
  $\boldsymbol{w} = \boldsymbol{w}_0 + \sum_{i=1}^{\ell} \boldsymbol{w}_i$
  And then recovers $\mathsf{usk}[\mathsf{id}] := [\boldsymbol{t}]_2, [\boldsymbol{w}]_2$

Fig. 8. $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(\mathcal{U}, \mathsf{id}, \ell; \boldsymbol{\rho})\rangle)$.

that because of the inner chameleon hash, a simulator is able to use the trapdoor to do a commitment to every possible words of the set of languages at once, and so can adaptively decide which id he requested. The proof of the second result is delayed to the paper full version [10].

For sake of generality, any bit-wise affine IBE could work (like for example Waters IBE [57]), the additional price paid for tightness here is very small and allows to have a better reduction in the proof, but it it not required by the framework itself.

Adaptive UC-Secure Oblivious Transfer. We finally get our instantiation by combining this $k - \mathsf{MDDH}$-based blind IBE with a $k - \mathsf{MDDH}$ variant of El Gamal for the CPA encryption needed (see the paper full version [10] for details). The requirement on the IBE blind user secret key generation (being able to adapt the key if the line changes) is achieved assuming that the server knows the discrete logarithms of the database lines. This is quite easy to achieve by assuming that for all line s, $DB(s) = [db(s)]_1$ where $db(s)$ is the real line (thus known). It implies a few more computation on the user's side in order to recover $db(s)$ from $DB(s)$, but this remains completely feasible if the lines belong to a small space. For practical applications, one could imagine to split all 256-bit lines into 8 pieces for a decent/constant trade-off in favor of computational efficiency.

For $k = 1$, so under the classical SXDH assumption, the first flow requires $8 \log |DB|$ elements in $\mathbb{G}_1$ for the CCA encryption part and $\log(|DB| + 1)$ in $\mathbb{G}_2$ for the chameleon one, while the second flow would now require $1 + 4 \log |DB|$ elements in $\mathbb{G}_1$, $1 + 2 \log |DB|$ for the fragmented masked key, and $2 \log |DB|$ for the projection keys.

4 Oblivious Language-Based Envelope

The previous construction opens new efficient applications to the already known Oblivious-Transfer protocols. But what happens when someone wants some additional access control by requesting extra properties, like if the user is only allowed to ask two lines with the same parity bits, the user can only request lines for whose number has been signed by an authority, or even finer control provided through credentials?

In this section we propose to develop a new primitive, that we call Oblivious Language-Based Envelope (OLBE). The idea generalizes that of Oblivious Transfer and OSBE, recalled right afterwards, for n messages (with n polynomial in the security parameter $\mathfrak{K}$) to provide the best of both worlds.

4.1 Oblivious Signature-Based Envelope

We recall the definition and security requirements of an OSBE protocol given in [13, 49], in which a sender $\mathcal{S}$ wants to send a private message $m \in \{0,1\}^{\mathfrak{K}}$ to a recipient $\mathcal{R}$ in possession of a valid certificate/signature on a public message M (given by a certification authority).

Definition 11 (Oblivious Signature-Based Envelope). *An* OSBE *scheme is defined by four algorithms* (Setup, KeyGen, Sign, Verify)*, and one interactive protocol* Protocol$\langle\mathcal{S},\mathcal{R}\rangle$*:*

- Setup$(1^{\mathfrak{K}})$*, where* $\mathfrak{K}$ *is the security parameter, generates the global parameters* param*;*
- KeyGen$(\mathfrak{K})$ *generates the keys* (vk, sk) *of the certification authority;*
- Sign(sk, M) *produces a signature* σ *on the input message* M*, under the signing key* sk*;*
- Verify(vk, M, σ) *checks whether* σ *is a valid signature on* M*, w.r.t. the public key* vk*; it outputs 1 if the signature is valid, and 0 otherwise.*
- Protocol$\langle\mathcal{S}(\mathsf{vk}, M, P), \mathcal{R}(\mathsf{vk}, M, \sigma)\rangle$ *between the sender* $\mathcal{S}$ *with the private message* P*, and the recipient* $\mathcal{R}$ *with a certificate* σ*. If* σ *is a valid signature under* vk *on the common message* M*, then* $\mathcal{R}$ *receives* m*, otherwise it receives nothing. In any case,* $\mathcal{S}$ *does not learn anything.*

The authors of [13] proposed some variations to the original definitions from [49], in order to prevent some interference by the authority. Following them, an OSBE scheme should fulfill the following security properties. The formal security games are given in [13]. No UC functionality has already been given, to the best of our knowledge.

- *correct*: the protocol actually allows $\mathcal{R}$ to learn P, whenever σ is a valid signature on M under vk;
- *semantically secure*: the recipient learns nothing about $\mathcal{S}$'s input m if it does not use a valid signature σ on M under vk as input. More precisely, if $\mathcal{S}_0$ owns P_0 and $\mathcal{S}_1$ owns P_1, the recipient that does not use a valid signature cannot distinguish an interaction with $\mathcal{S}_0$ from an interaction with $\mathcal{S}_1$ even if he has eavesdropped on several interactions $\langle\mathcal{S}(\mathsf{vk}, M, P), \mathcal{R}(\mathsf{vk}, M, \sigma)\rangle$ with valid signatures, and the same sender's input P;
- *escrow-free* (*oblivious with respect to the authority*): the authority (owner of the signing key sk), playing as the sender or just eavesdropping, is unable to distinguish whether $\mathcal{R}$ used a valid signature σ on M under vk as input.
- *semantically secure w.r.t. the authority*: after the interaction, the authority (owner of the signing key sk) learns nothing about m from a passive access to a challenge transcript.

4.2 Definition of an Oblivious Language-Based Envelope

In such a protocol, a sender $\mathcal{S}$ wants to send one or several private messages (up to $n_{\max} \leq n$) among $(m_1, \ldots, m_n) \in (\{0,1\}^{\ell})^n$ to a recipient $\mathcal{R}$ in possession of a tuple of words $W = (W_{i_1}, \ldots, W_{i_{n_{\max}}})$ such that some of the words W_{i_j} may belong to the corresponding language $\mathfrak{L}_{i_j}$. More precisely, the receiver gets each m_{i_j} as soon as $W_{i_j} \in \mathfrak{L}_{i_j}$ with the requirement that he gets at most $n_{\max}$ messages. In such a scheme, the languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$ are assumed to be a trapdoor collection of languages, publicly verifiable and self-randomizable (see Section 2.3 for the definitions of the properties of the languages).

The collections of words can be a single certificate/signature on a message M (encompassing OSBE, with $n = n_{\max} = 1$), a password, a credential, a line number (encompassing 1-out-of-n oblivious transfer[11], with $n_{\max} = 1$), k line numbers (encompassing k-out-of-n oblivious transfer, with $n_{\max} = k$), etc. (see the paper full version [10] for detailed examples). Following the definitions for OSBE recalled above and given in [13, 49], we give the following definition for OLBE. As we consider simulation-based security (in the UC framework), we allow a simulated setup SetupT to be run instead of the classical setup Setup in order to allow the simulator to possess some trapdoors. Those two setup algorithms should be indistinguishable.

Definition 12 (Oblivious Language-Based Envelope). *An* OLBE *scheme is defined by four algorithms* (Setup, KeyGen, Samp, Verify)*, and one interactive protocol* Protocol$\langle \mathcal{S}, \mathcal{R} \rangle$*:*

- Setup$(1^{\mathfrak{K}})$*, where* $\mathfrak{K}$ *is the security parameter, generates the global parameters* param*, among which the numbers* n *and* $n_{\max}$*;*

or SetupT$(1^{\mathfrak{K}})$*, where* $\mathfrak{K}$ *is the security parameter, additionally allows the existence*[12] *of a trapdoor* tk *for the collection of languages* $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$*.*

- KeyGen(param, $\mathfrak{K}$) *generates, for all* $i \in \{1, \ldots, n\}$*, the description of the language* $\mathfrak{L}_i$ *(as well as the language key* $\mathsf{sk}_{\mathfrak{L}_i}$ *if need be). If the parameters* param *were defined by* SetupT*, this implicitly also defines the common trapdoor* tk *for the collection of languages* $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$*.*
- Samp(param, I) *or* Samp(param, $I, (\mathsf{sk}_{\mathfrak{L}_i})_{i \in I}$) *such that* $I \subset \{1, \ldots, n\}$ *and* $|I| = n_{\max}$*, generates a list of words* $(W_i)_{i \in I}$ *such that* $W_i \in \mathfrak{L}_i$ *for all* $i \in I$*;*
- Verify$_i(W_i, \mathfrak{L}_i)$ *checks whether* W_i *is a valid word in the language* $\mathfrak{L}_i$*. It outputs* 1 *if the word is valid,* 0 *otherwise;*
- Protocol$\langle \mathcal{S}((\mathfrak{L}_1, \ldots, \mathfrak{L}_n), (m_1, \ldots, m_n)), \mathcal{R}((\mathfrak{L}_1, \ldots, \mathfrak{L}_n), (W_i)_{i \in I}) \rangle$*, which is executed between the sender* $\mathcal{S}$ *with the private messages* $(m_1, \ldots, P_n)$ *and corresponding languages* $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$*, and the recipient* $\mathcal{R}$ *with the same languages and the words* $(W_i)_{i \in I}$ *with* $I \subset \{1, \ldots, n\}$ *and* $|I| = n_{\max}$*, proceeds as follows. For all* $i \in I$*, if the algorithm* Verify$_i(W_i, \mathfrak{L}_i)$ *returns* 1*, then* $\mathcal{R}$ *receives* m_i*, otherwise it does not. In any case,* $\mathcal{S}$ *does not learn anything.*

4.3 Security Properties and Ideal Functionality of OLBE

Since we aim at proving the security in the universal composability framework, we now describe the corresponding ideal functionality (depicted in Figure 9). However, in order to ease the comparison with an OSBE scheme, we first list the security properties required, following [49] and [13]:

- *correct*: the protocol actually allows $\mathcal{R}$ to learn $(m_i)_{i \in I}$, whenever $(W_i)_{i \in I}$ are valid words of the languages $(\mathfrak{L}_i)_{i \in I}$, where $I \subset \{1, \ldots, n\}$ and $|I| = n_{\max}$;

[11] Even if, as explained in the former section, we would rather consider equivocable commitments of line numbers than directly line numbers, in order to get adaptive UC security.

[12] The specific trapdoor will depend on the languages and be computed in the KeyGen algorithm.

– *semantically secure* (*sem*): the recipient learns nothing about the input m_i of $\mathcal{S}$ if it does not use a word in $\mathfrak{L}_i$. More precisely, if $\mathcal{S}_0$ owns $m_{i,0}$ and $\mathcal{S}_1$ owns $m_{i,1}$, the recipient that does not use a word in $\mathfrak{L}_i$ cannot distinguish between an interaction with $\mathcal{S}_0$ and an interaction with $\mathcal{S}_1$ even if the receiver has seen several interactions
$$\langle \mathcal{S}((\mathfrak{L}_1,\ldots,\mathfrak{L}_n),(m_1,\ldots,m_n)), \mathcal{R}((\mathfrak{L}_1,\ldots,\mathfrak{L}_n),(W'_j)_{j\in I})\rangle$$
with valid words $W'_i \in \mathfrak{L}_i$, and the same sender's input m_i;
– *escrow free* (*oblivious with respect to the authority*): the authority corresponding to the language $\mathfrak{L}_i$ (owner of the language secret key $\mathsf{sk}_{\mathfrak{L}_i}$ – if it exists), playing as the sender or just eavesdropping, is unable to distinguish whether $\mathcal{R}$ used a word W_i in the language $\mathfrak{L}_i$ or not. This requirement also holds for anyone holding the trapdoor key tk.
– *semantically secure w.r.t. the authority* (*sem**): after the interaction, the trusted authority (owner of the language secret keys if they exist) learns nothing about the values $(m_i)_{i\in I}$ from the transcript of the execution. This requirement also holds for anyone holding the trapdoor key tk.

Moreover, the Setups should be indistinguishable and it should be infeasible to find a word belonging to two or more languages without the knowledge of tk.

The functionality $\mathcal{F}_{\text{OLBE}}$ is parametrized by a security parameter $\mathfrak{K}$ and a set of languages $(\mathfrak{L}_1,\ldots,\mathfrak{L}_n)$ along with the corresponding public verification algorithms $(\mathsf{Verify}_1,\ldots,\mathsf{Verify}_n)$. It interacts with an adversary $\mathscr{S}$ and a set of parties $\mathfrak{P}_1,\ldots,\mathfrak{P}_N$ via the following queries:

– **Upon receiving from party $\mathfrak{P}_i$ an input of the form $(\mathtt{Send}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1,\ldots,m_n))$** , with $m_k \in \{0,1\}^{\mathfrak{K}}$ for all k: record the tuple $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1,\ldots,m_n))$ and reveal $(\mathtt{Send}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to the adversary $\mathscr{S}$. Ignore further `Send`-message with the same ssid from $\mathfrak{P}_i$.
– **Upon receiving an input of the form $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (W_i)_{i\in I})$ with the conditions $I \subset \{1,\ldots,n\}$ and $|I| = n_{\max}$ from party $\mathfrak{P}_j$**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1,\ldots,m_n))$ is not recorded. Otherwise, reveal $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to the adversary $\mathscr{S}$ and send the message $(\mathtt{Received}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m'_k)_{k\in I})$ to $\mathfrak{P}_j$ where $m'_k = m_k$ if $\mathsf{Verify}_k(W_k, \mathfrak{L}_k)$ returns 1, and $m'_k = \perp$ otherwise. Ignore further `Received`-message with the same ssid from $\mathfrak{P}_j$.

Fig. 9. Ideal Functionality for Oblivious Language-Based Envelope $\mathcal{F}_{\text{OLBE}}$

The ideal functionality is parametrized by a set of languages $(\mathfrak{L}_1,\ldots,\mathfrak{L}_n)$. Since we show in the following sections that one can see OSBE and OT as special cases of OLBE, it is inspired from the oblivious transfer functionality given in [1, 21, 24] in order to provide a framework consistent with works well-known in the literature. As for oblivious transfer (Figure 2), we adapt them to the simple UC framework for simplicity (this enables us to get rid of `Sent` and `Received` queries from the adversary since the delayed outputs are automatically considered in this simpler framework: We implicitly let the adversary determine if it wants

to acknowledge the fact that a message was indeed sent). The first step for the sender (`Send` query) consists in telling the functionality he is willing to take part in the protocol, giving as input his intended receiver and the messages he is willing to send (up to $n_{\max}$ messages). For the receiver, the first step (`Receive` query) consists in giving the functionality the name of the player he intends to receive the messages from, as well as his words. If the word does belong to the language, the receiver recovers the sent message, otherwise, he only gets a special symbol $\perp$.

4.4 Generic UC-Secure Instantiation of OLBE with Adaptive Security

For the sake of clarity, we now concentrate on the specific case where $n_{\max} = 1$. This is the most classical case in practice, and suffices for both OSBE and 1-out-of-n OT. In order to get a generic protocol in which $n_{\max} > 1$, one simply has to run $n_{\max}$ protocols in parallel. This modifies the algorithms Samp and Verify as follows: $\mathsf{Samp}(\mathsf{param}, \{i\})$ or $\mathsf{Samp}(\mathsf{param}, \{i\}, \{\mathsf{sk}_{\mathfrak{L}_i}\})$ generates a word $W = W_i \in \mathfrak{L}_i$ and $\mathsf{Verify}_j(W, \mathfrak{L}_j)$ checks whether W is a valid word in $\mathfrak{L}_j$.

Let us introduce our protocol OLBE: we will call $\mathcal{R}$ the receiver and $\mathcal{S}$ the sender. If $\mathcal{R}$ is an honest receiver, then he knows a word $W = W_i$ in one of the languages $\mathfrak{L}_i$. If $\mathcal{S}$ is an honest sender, then he wants to send up a message among $(m_1, \ldots, m_n) \in (\{0,1\}^{\mathfrak{K}})^n$ to $\mathcal{R}$. We assume the languages $\mathfrak{L}_i$ to be self-randomizable and publicly verifiable. We also assume the collection of languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$ possess a trapdoor, that the simulator is able to find by programming the common reference string. As recalled in the previous section, this trapdoor enables him to find a word lying in the intersection of the n languages. This should be infeasible without the knowledge of the trapdoor. Intuitively, this allows the simulator to commit to all languages at once, postponing the time when it needs to choose the exact language he wants to bind to. On the opposite, if a user was granted the same possibilities, this would prevent the simulator to extract the chosen language.

We assume the existence of a labeled CCA-encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\mathrm{cca}}, \mathsf{KeyGen}_{\mathrm{cca}}, \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}, \mathsf{Decrypt}^{\ell}_{\mathrm{cca}})$ compatible with an SPHF onto a set G. In the KeyGen algorithm, the description of the languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$ thus implicitly defines the languages $(\mathfrak{L}^c_1, \ldots, \mathfrak{L}^c_n)$ of CCA-encryptions of elements of the languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$. We additionally use a key derivation function `KDF` to derive a pseudo-random bit-string $K \in \{0,1\}^{\mathfrak{K}}$ from a pseudo-random element $v \in G$. One can use the Leftover-Hash Lemma [40], with a random seed defined in param during the global setup, to extract the entropy from v, then followed by a pseudo-random generator to get a long enough bit-string. Many uses of the same seed in the Leftover-Hash Lemma just lead to a security loss linear in the number of extractions. We also assume the existence of a Pseudo-Random Generator (PRG) F with input size equal to the plaintext size, and output size equal to the size of the messages in the database and an IND-CPA encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\mathrm{cpa}}, \mathsf{KeyGen}_{\mathrm{cpa}}, \mathsf{Encrypt}_{\mathrm{cpa}}, \mathsf{Decrypt}_{\mathrm{cpa}})$ with plaintext size at least equal to the security parameter.

We follow the ideas of the oblivious transfer constructions given in [1, 9], giving the protocol presented on Figure 10. For the sake of simplicity, we only give the version for adaptive security, in which the sender generates a public key pk and ciphertext c to create a somewhat secure channel (they would not be used in the static version).

CRS: $\mathsf{param} \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}})$, $\mathsf{param}_{\mathrm{cca}} \xleftarrow{\$} \mathsf{Setup}_{\mathrm{cca}}(1^{\mathfrak{K}})$, $\mathsf{param}_{\mathrm{cpa}} \xleftarrow{\$} \mathsf{Setup}_{\mathrm{cpa}}(1^{\mathfrak{K}})$.
Pre-flow:

1. Sender generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by KeyGen.
2. Sender sends pk to User.

Flow From the Receiver $\mathcal{R}$:

1. User chooses a random value J, computes $R \leftarrow F(J)$ and encrypts J under pk: $c \xleftarrow{\$} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, J)$.
2. User computes $C \xleftarrow{\$} \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}(W; r)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{R}, \mathcal{S})$.
3. User completely erases J and the random coins used by $\mathsf{Encrypt}_{\mathrm{cpa}}$ and sends C and c to Sender. He also checks the validity of his words: the receiver only keeps the random coins used by $\mathsf{Encrypt}_{\mathrm{cca}}$ for the j such that $\mathsf{Verify}_j(W, \mathfrak{L}_j) = 1$ (since he knows they will be useless otherwise).

Flow From the Sender $\mathcal{S}$:

1. Sender decrypts $J \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and then $R \leftarrow F(J)$.
2. For all $j \in \{1, \ldots, n\}$, sender computes $\mathsf{hk}_j = \mathsf{HashKG}(\ell, \mathfrak{L}^c_j, \mathsf{param})$, $\mathsf{hp}_j = \mathsf{ProjKG}(\mathsf{hk}_j, \ell, (\mathfrak{L}^c_j, \mathsf{param}))$, $v_j = \mathsf{Hash}(\mathsf{hk}_j, (\mathfrak{L}^c_j, \mathsf{param}), (\ell, C))$, $Q_j = m_j \oplus \mathtt{KDF}(v_j) \oplus R$.
3. Sender erases everything except $(Q_j, \mathsf{hp}_j)_{j \in \{1,\ldots,n\}}$ and sends them over a secure channel.

Message recovery:
Upon receiving $(Q_j, \mathsf{hp}_j)_{j \in \{1,\ldots,n\}}$, $\mathcal{R}$ can recover m_i by computing $m_i = Q_i \oplus \mathsf{ProjHash}(\mathsf{hp}_i, (\mathfrak{L}^c_i, \mathsf{param}), (\ell, C), r) \oplus R$.

Fig. 10. UC-Secure OLBE for One Message (Secure Against Adaptive Corruptions)

Theorem 13. *The oblivious language-based envelope scheme described in Figure 10 is* UC*-secure in the presence of adaptive adversaries, assuming reliable erasures, an* IND-CPA *encryption scheme, and an* IND-CCA *encryption scheme admitting an* SPHF *on the language of valid ciphertexts of elements of $\mathfrak{L}_i$ for all i, as soon as the languages are self-randomizable, publicly-verifiable and admit a common trapdoor. The proof is given in the paper full version [10].*

4.5 Oblivious Primitives Obtained by the Framework

Classical oblivious primitives such as Oblivious Transfer (both 1-out-of-n and k-out-of-n) or Oblivious Signature-Based Envelope directly lie in this framework and can be seen as examples of Oblivious Language-Based Envelope. We provide in the paper full version [10] details about how to describe the languages and choose appropriate smooth projective hash functions to readily achieve current instantiations of Oblivious Signature-Based Envelope or Oblivious Transfer from our generic protocol. The framework also enables us to give a new instantiation of Access Controlled Oblivious Transfer under classical assumptions. In such a primitive, the user does not automatically gets the line he asks for, but has to prove that he possesses one of the credential needed to access this particular line.

For the sake of simplicity, all the instantiations given are pairing-based but techniques explained in [9] could be used to rely on other families of assumptions, like decisional quadratic residue or even LWE.

Acknowledgments

This work was supported in part by the French ANR EnBid Project (ANR-14-CE28-0003).

References

1. Abdalla, M., Benhamouda, F., Blazy, O., Chevalier, C., Pointcheval, D.: SPHF-friendly non-interactive commitments. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 214–234. Springer (Dec 2013)
2. Abdalla, M., Chevalier, C., Pointcheval, D.: Smooth projective hashing for conditionally extractable commitments. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 671–689. Springer (Aug 2009)
3. Abdalla, M., Pointcheval, D.: A scalable password-based group key exchange protocol in the standard model. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 332–347. Springer (Dec 2006)
4. Aiello, W., Ishai, Y., Reingold, O.: Priced oblivious transfer: How to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer (May 2001)
5. Attrapadung, N.: Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 557–577. Springer (May 2014)
6. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 12. pp. 784–796. ACM Press (Oct 2012)
7. Ben Hamouda, F., Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: Efficient UC-secure authenticated key-exchange for algebraic languages. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 272–291. Springer (Feb / Mar 2013)

8. Benhamouda, F., Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: New techniques for SPHFs and efficient one-round PAKE protocols. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 449–475. Springer (Aug 2013)
9. Blazy, O., Chevalier, C.: Generic construction of uc-secure oblivious transfer. In: Malkin, T., Kolesnikov, V., Lewko, A.B., Polychronakis, M. (eds.) Applied Cryptography and Network Security - 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9092, pp. 65–86. Springer (2015), http://dx.doi.org/10.1007/978-3-319-28166-7_4
10. Blazy, O., Chevalier, C., Germouty, P.: Adaptive oblivious transfer and generalizations. Cryptology ePrint Archive, Report 2016/259 (2016), http://eprint.iacr.org/2016/259
11. Blazy, O., Fuchsbauer, G., Pointcheval, D., Vergnaud, D.: Signatures on randomizable ciphertexts. In: Gennaro, R. (ed.) Proceedings of PKC 2011. Lecture Notes in Computer Science, Springer (2010), full version available from the web page of the authors
12. Blazy, O., Kiltz, E., Pan, J.: (hierarchical) identity-based encryption from affine message authentication. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 408–425. Springer (Aug 2014)
13. Blazy, O., Pointcheval, D., Vergnaud, D.: Round-optimal privacy-preserving protocols with smooth projective hash functions. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 94–111. Springer (Mar 2012)
14. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer (Aug 2001)
15. Boneh, D., Goh, E.J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer (Feb 2005)
16. Camenisch, J., Casati, N., Groß, T., Shoup, V.: Credential authenticated identification and key exchange. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 255–276. Springer (Aug 2010)
17. Camenisch, J., Dubovitskaya, M., Haralambiev, K.: Efficient structure-preserving signature scheme from standard assumptions. In: Visconti, I., Prisco, R.D. (eds.) SCN 12. LNCS, vol. 7485, pp. 76–94. Springer (Sep 2012)
18. Camenisch, J., Dubovitskaya, M., Neven, G.: Oblivious transfer with access control. In: Al-Shaer, E., Jha, S., Keromytis, A.D. (eds.) ACM CCS 09. pp. 131–140. ACM Press (Nov 2009)
19. Camenisch, J., Dubovitskaya, M., Neven, G., Zaverucha, G.M.: Oblivious transfer with hidden access control policies. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 192–209. Springer (Mar 2011)
20. Camenisch, J., Neven, G., shelat, a.: Simulatable adaptive oblivious transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer (May 2007)
21. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS. pp. 136–145. IEEE Computer Society Press (Oct 2001)
22. Canetti, R., Cohen, A., Lindell, Y.: A simpler variant of universally composable security for standard multiparty computation. In: CRYPTO 2015, Part II. pp. 3–22. LNCS, Springer (Aug 2015)
23. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: 34th ACM STOC. pp. 494–503. ACM Press (May 2002)

24. Choi, S.G., Katz, J., Wee, H., Zhou, H.S.: Efficient, adaptively secure, and composable oblivious transfer with a single, global CRS. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 73–88. Springer (Feb / Mar 2013)
25. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: 36th FOCS. pp. 41–50. IEEE Computer Society Press (Oct 1995)
26. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer (Apr / May 2002)
27. Cui, Y., Fujisaki, E., Hanaoka, G., Imai, H., Zhang, R.: Formal security treatments for signatures from identity-based encryption. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 218–227. Springer (Nov 2007)
28. Di Crescenzo, G., Ostrovsky, R., Rajagopalan, S.: Conditional oblivious transfer and timed-release encryption. In: Stern, J. (ed.) EUROCRYPT'99. LNCS, vol. 1592, pp. 74–89. Springer (May 1999)
29. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
30. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for Diffie-Hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 129–147. Springer (Aug 2013)
31. Fischlin, M.: Round-optimal composable blind signatures in the common reference string model. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 60–77. Springer (Aug 2006)
32. Gay, R., Kerenidis, I., Wee, H.: Communication complexity of conditional disclosure of secrets and attribute-based encryption. In: CRYPTO 2015, Part II. pp. 485–502. LNCS, Springer (Aug 2015)
33. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 524–543. Springer (May 2003), http://eprint.iacr.org/2003/032.ps.gz
34. Gertner, Y., Ishai, Y., Kushilevitz, E., Malkin, T.: Protecting data privacy in private information retrieval schemes. In: 30th ACM STOC. pp. 151–160. ACM Press (May 1998)
35. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A. (ed.) 19th ACM STOC. pp. 218–229. ACM Press (May 1987)
36. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing 17(2), 281–308 (Apr 1988)
37. Green, M., Hohenberger, S.: Blind identity-based encryption and simulatable oblivious transfer. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 265–282. Springer (Dec 2007)
38. Green, M., Hohenberger, S.: Universally composable adaptive oblivious transfer. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 179–197. Springer (Dec 2008)
39. Guleria, V., Dutta, R.: Lightweight universally composable adaptive oblivious transfer. In: Au, M., Carminati, B., Kuo, C.C. (eds.) Network and System Security, Lecture Notes in Computer Science, vol. 8792, pp. 285–298. Springer International Publishing (2014), http://dx.doi.org/10.1007/978-3-319-11698-3_22
40. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM Journal on Computing 28(4), 1364–1396 (1999)

41. Horvitz, O., Katz, J.: Universally-composable two-party computation in two rounds. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 111–129. Springer (Aug 2007)
42. Ishai, Y., Wee, H.: Partial garbling schemes and their applications. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) ICALP 2014, Part I. LNCS, vol. 8572, pp. 650–662. Springer (Jul 2014)
43. Jarecki, S., Liu, X.: Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 577–594. Springer (Mar 2009)
44. Kalai, Y.T.: Smooth projective hashing and two-message oblivious transfer. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 78–95. Springer (May 2005)
45. Katz, J., Vaikuntanathan, V.: Round-optimal password-based authenticated key exchange. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 293–310. Springer (Mar 2011)
46. Kiayias, A., Leonardos, N., Lipmaa, H., Pavlyk, K., Tang, Q.: Optimal rate private information retrieval from homomorphic encryption. PoPETs 2015(2), 222–243 (2015), `http://www.degruyter.com/view/j/popets.2015.2015.issue-2/popets-2015-0016/popets-2015-0016.xml`
47. Kurosawa, K., Nojima, R., Phong, L.T.: Generic fully simulatable adaptive oblivious transfer. In: Lopez, J., Tsudik, G. (eds.) ACNS 11. LNCS, vol. 6715, pp. 274–291. Springer (Jun 2011)
48. Laur, S., Lipmaa, H.: A new protocol for conditional disclosure of secrets and its applications. In: Katz, J., Yung, M. (eds.) ACNS 07. LNCS, vol. 4521, pp. 207–225. Springer (Jun 2007)
49. Li, N., Du, W., Boneh, D.: Oblivious signature-based envelope. In: Borowsky, E., Rajsbaum, S. (eds.) 22nd ACM PODC. pp. 182–189. ACM (Jul 2003)
50. Naor, M., Pinkas, B.: Visual authentication and identification. In: Kaliski Jr., B.S. (ed.) CRYPTO'97. LNCS, vol. 1294, pp. 322–336. Springer (Aug 1997)
51. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: Kosaraju, S.R. (ed.) 12th SODA. pp. 448–457. ACM-SIAM (Jan 2001)
52. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer (Aug 2008)
53. Rabin, M.O.: How to exchange secrets with oblivious transfer. Technical Report TR81, Harvard University (1981)
54. Rial, A., Kohlweiss, M., Preneel, B.: Universally composable adaptive priced oblivious transfer. In: Shacham, H., Waters, B. (eds.) PAIRING 2009. LNCS, vol. 5671, pp. 231–247. Springer (Aug 2009)
55. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO'84. LNCS, vol. 196, pp. 47–53. Springer (Aug 1984)
56. Wang, X.S., Huang, Y., Chan, T.H.H., Shelat, A., Shi, E.: SCORAM: Oblivious RAM for secure computation. In: Ahn, G.J., Yung, M., Li, N. (eds.) ACM CCS 14. pp. 191–202. ACM Press (Nov 2014)
57. Waters, B.R.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer (May 2005)
58. Wee, H.: Dual system encryption via predicate encodings. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 616–637. Springer (Feb 2014)
59. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS. pp. 162–167. IEEE Computer Society Press (Oct 1986)