# A Composable Look at Updatable Encryption

Françoise Levy-dit-Vehel[1] and Maxime Roméas[2]

[1] LIX, ENSTA Paris, INRIA, Institut Polytechnique de Paris, 91120 Palaiseau, France. levy@ensta.fr
[2] LIX, École polytechnique, INRIA, Institut Polytechnique de Paris, 91120 Palaiseau, France. romeas@lix.polytechnique.fr

**Abstract.** Updatable Encryption (UE), as originally defined by Boneh *et al.* in 2013, addresses the problem of key rotation on outsourced data while maintaining the communication complexity as low as possible. The security definitions for UE schemes have been constantly updated since then. However, the security notion that is best suited for a particular application remains unclear.
To solve this problem in the ciphertext-independent setting, we use the Constructive Cryptography (CC) framework defined by Maurer *et al.* in 2011. We define and construct a resource that we call *Updatable Server-Memory Resource* **USMR**, and study the confidentiality guarantees it achieves when equipped with a UE protocol, that we also model in this framework. With this methodology, we are able to construct resources tailored for each security notion. In particular, we prove that IND-UE-RCCA is the right security notion for many practical UE schemes. As a consequence, we notably rectify a claim made by Boyd *et al.* , namely that their IND-UE security notion is better than the IND-ENC+UPD notions, in that it hides the age of ciphertexts. We show that this is only true when ciphertexts can leak *at most one time per epoch*.
We stress that UE security is thought of in the context of adaptive adversaries, and UE schemes should thus bring post-compromise confidentiality guarantees to the client. To handle such adversaries, we use an extension of CC due to Jost *et al.* and give a clear, simple and composable description of the post-compromise security guarantees of UE schemes. We also model semi-honest adversaries in CC.
Our adaption of the CC framework to UE is generic enough to model other interactive protocols in the outsourced storage setting.

## 1 Introduction

Updatable Encryption (UE) allows a client, who outsourced his encrypted data, to make an untrusty server update it. The huge real-life applications of such a functionality explains the recent renew of interest on the subject [6,11,10,7,3]. The concept and definition of UE first appeared in a paper by Boneh et al. in 2013 [2], as an application of key homomorphic PRFs to the context of rotating keys for encrypted data stored in the cloud.

In this work, we are interested in the *ciphertext-independent* variant of UE, *i.e.*, the one in which a token dependent only on the old and new keys is used

to update all ciphertexts. This variant minimizes the communication complexity needed when doing key-rotation in the outsourced storage setting. Lehmann *et al.* [11] gave the first rigorous security treatment of those schemes by introducing the ENC-CPA and UPD-CPA security games alongside an Elgamal based secure UE scheme called RISE. Klooß *et al.* [10] strengthened these security notions in the context of chosen-ciphertext security by introducing (R)CCA games and generic constructions on how to achieve UE schemes meeting their new definitions. Boyd *et al.* [3], introduced a strictly stronger security notion, in that it hides the age of the data, called IND-UE and showed relations between all existing UE security notions. They also propose a generic construction for CCA secure schemes alongside the SHINE family of fast UE schemes based on random looking permutations and exponentiation. Finally, Jiang [7] showed that whether the token allowed to only upgrade or also downgrade ciphertexts did not make a difference in terms of security. Moreover, they propose the first post-quantum, LWE based, UE scheme.

*Constructive Cryptography.* This composable framework was first introduced by Maurer *et al.* [12]. Its *top-down* approach makes it very intuitive and nice to work with. Badertscher *et al.* [1] first used it in the outsourced storage setting to clarify some security models and build protocols in a modular fashion. Jost *et al.* [9] introduced the notion of *global event history* to the theory and give a first treatment of adaptive security. This work was then pursued [8] by introducing interval-wise security guarantees in order to find a way to overcome impossibility results such as the commitment problem.

**Contributions to Updatable Encryption** We go further in asserting the exact security guarantees one needs for practical UE protocols. We achieve this by considering UE in the Constructive Cryptography framework introduced by Maurer *et al.* [12].

This allows us to provide a detailed treatment of the existing notions of UE security that appeared previously, namely : IND-UE-atk, IND-ENC+UPD-atk, atk $\in$ {CPA, CCA, RCCA}[3], thus unveiling the differences between them.

We start from a basic Updatable Server Memory Resource (**USMR**) that we define and construct. We then define an encryption protocol based on UE for the **USMR**. In CC, this translates into describing our protocol by means of a pair of converters (one for the client and one for the server). Studying the effects of a real execution of such a protocol plugged in the **USMR**, we are able to give the exact security properties obtained.

We match the various adversary capabilities, in our CC setting to the well-known CPA or CCA adversarial contexts. Considering two leakage scenarii, this methodology yields the following main results.

[3] Recall that RCCA captures the idea according to which the ability of the adversary to generate a new ciphertext whose decryption is equal to that of the challenged ciphertext does not help him.

In the "any number of leaks" model, IND-(ENC+UPD)-CPA is necessary and sufficient to construct a confidential **USMR** (**cUSMR**). This means that IND-UE-CPA security is not always sufficient to hide the age of ciphertexts, clarifying an assertion of Boyd et al. This hiding property holds true only in a particular leakage model, namely "one leak per entry per epoch". In the same leakage model, the IND-UE-CCA notion is sufficient but not necessary to construct a **cUSMR** that allows injections (**ciUSMR**). Thus, for many practical applications, IND-UE-RCCA is the right security notion to construct a **ciUSMR**.

Furthermore, considering the enhanced interval-wise CC model proposed in [8], we give a precise treatment of the post-compromise security guarantees of UE schemes.

It is to be noted that all previous UE protocol instantiations whose updates are ciphertext-indepedent fit in our study. Whether those updates are deterministic or not does not matter for CPA security, whereas for CCA security we focus only on deterministic updates.

**Contributions to CC** Our work on the CC framework generalizes the work of [5] and extends [1], where the basic Server Memory Resource (**SMR**) is first defined. We define and construct a resource tailored for UE, namely the Updatable **SMR** (**USMR**). The main difference with the basic **SMR** is that its adversarial interface, S, is subdivided into two sub-interfaces S.1 and S.2 : S.1 guarantees honest behavior of the server, in that it participates in the protocol by means of a converter branched at this sub-interface; whereas at S.2, which is directly branched to the database, there is no guarantee on the behavior of the server. In this sense, we call the server a *semi-honest adversary*. This subtle difference with **SMR** permits us to make the server *interactive* : indeed, at S.1, the (converter of the) server can handle the client requests (e.g. update, fetchToken). The leakage behavior is described at S.2.

Another contribution to CC is a new improvement of a method of proof by Maurer *et al.* [5]: to prove the security of the different constructions, we define a sequence of systems, the first being the real one equipped with the protocol, the last the ideal one it is supposed to achieve. Our proof thus evolves from a conjunction of small security-game proofs. Moreover, this allows us to handle an asymmetry : indeed, to prove IND-UE-atk security, we here have to handle game pairs of the form $(m, c)$, *i.e.* (plaintext, ciphertext)[4]. Those pairs cannot be handled by the classical proofs by security games proposed in [5], which only treats requests of the form $(m, m')$ (mimicing their proofs in our case, the distinguishing advantage would be 1).

**Organization of the paper** The next section presents basic notions about UE and CC that are needed to understand our work. In section 3 we present

[4] Where the oracle either answers an encryption of $m$ or an update of $c$.

our Updatable Server Memory Resource, with associate resources and protocols : updatable key resource (and tokens) and updatable encryption protocols. The security properties of UE schemes w.r.t. the different leakage contexts are analyzed and proved in sections 4.1 to 4.3. In section 4.4, we give an exact description of the post-compromise security guarantees given by UE schemes. Section 5 concludes the paper.

## 2 Preliminaries

### 2.1 Updatable Encryption

A UE scheme is given by a tuple of algorithms (KG, TG, Enc, Dec, Upd) operating in epochs over message space $\mathcal{M}$, ciphertext space $\mathcal{C}$, key space $\mathcal{K}$ and token space $\mathcal{T}$. For a *correct* UE scheme, these algorithms work as follows :

- On input $1^\lambda$ where $\lambda$ is a security parameter, $\mathsf{KG}(1^\lambda) \in \mathcal{K}$ returns a key.
- On input $(k_e, k_{e+1}) \in \mathcal{K}^2$, $\mathsf{TG}(k_e, k_{e+1}) \in \mathcal{T}$ returns a token $\Delta_{e+1}$.
- On input a plaintext $m \in \mathcal{M}$ and $k_e \in \mathcal{K}$, $\mathsf{Enc}_{k_e}(m) \in \mathcal{C}$ returns an encryption of $m$ under the key $k_e$ of epoch $e$.
- On input a ciphertext $c \in \mathcal{C}$ encrypting a message $m \in \mathcal{M}$ under a key $k_e \in \mathcal{K}$, $\mathsf{Dec}_{k_e}(c) \in \mathcal{M}$ returns the message $m$. If $c$ is an invalid ciphertext, a decryption error $\diamond$ is returned instead.
- On input a ciphertext $c_e \in \mathcal{C}$ encrypting a message $m \in \mathcal{M}$ under a key $k_e \in \mathcal{K}$ and a token $\Delta_{e+1} \in \mathcal{T}$ computed with the pair $(k_e, k_{e+1}) \in \mathcal{K}^2$, $\mathsf{Upd}_{\Delta_{e+1}}(c) \in \mathcal{C}$ returns an encryption of $m$ under the key $k_{e+1}$.

### 2.2 Security notions

In this work, we study what confidentiality guarantees are exactly brought by the use of UE schemes. In previous works, the security of UE is described using security games. We will analyze the differences between the IND-ENC+IND-UPD security notion of [11] and the IND-UE one of [3]. We recall the different game oracles described in [3] and [7] in fig.8 of the appendix, while the oracles available to the adversary for each security notion are shown in fig. 9 of the appendix.

In IND-UE security, when given a plaintext $m$ and a ciphertext $c$ from a previous epoch encrypting a message of length $|m|$, the game challenges the adversary with either an encryption of $m$ or an update of $c$.

In IND-ENC security, when given two plaintexts $m_0$ and $m_1$ of same length, the game challenges the adversary with an encryption of one of them.

In IND-UPD security, when given two old ciphertexts $c_0$ and $c_1$ encrypting two messages of the same length, the game challenges the adversary with an update of one of them.

We build upon the work of [3] by adding an RCCA variant to the IND-UE security notion. In IND-UE security, the challenge is either an encryption of a plaintext $m_0$ or an update of a ciphertext $c_1$ sent by the adversary. On a challenge

request, our IND-UE-RCCA security game behaves in the same way except it remembers $m_0$ and $m_1$ (the decryption of $c_1$). When the decryption oracle is called on ciphertext $c$, if $c$ decrypts to a plaintext $m$, and not a decryption error $\diamond$, the oracle returns `test` if $m \in \{m_0, m_1\}$ and $m$ otherwise. We use the same RCCA variant, introduced in [10], for the IND-ENC and IND-UPD notions.

### 2.3 Constructive Cryptography

We give the required material to understand how we use the Constructive Cryptography framework by Maurer *et al.* [13,12,14,9,8]. We follow the presentation made by Jost and Maurer in [8].

**Global Event History** This work uses the globally observable events introduced in [9]. Formally, we consider a *global event history* $\mathcal{E}$ which is a list of event without duplicates. An event is defined by a name $n$, and triggering the event $n$ corresponds to the action of appending $n$ to $\mathcal{E}$, denoted by $\mathcal{E} \xleftarrow{+} \mathcal{E}_n$. For short, we use the notation $\mathcal{E}_n$ to say that event $n$ happened.

**Resources and Converters** A *resource* $\mathbf{R}$ is a system that interacts, in a black-box manner, at one or more of its *interfaces*, by receiving an input at a given interface and subsequently sending an output at the same interface. The behavior of the resource depends on the global event history $\mathcal{E}$ and it can append events to it. Do note that a resource only defines the observable behavior of a system and not how it is defined internally. We use the notation $[\mathbf{R}_1, \ldots, \mathbf{R}_k]$ to denote the parallel composition of resources. It corresponds to a new resource and, if $\mathbf{R}_1, \ldots, \mathbf{R}_k$ have disjoint interfaces sets, the interface set of the composed resource is the union of those.

In CC, *converters* are used to link resources and reprogram interfaces, thus expressing the local computations of the parties involved. A converter is plugged on a set of interfaces at the inside and provides a set of interfaces at the outside. When it receives an input at its outside interface, the converter uses a bounded number of queries to the inside interface before computing a value and outputting it at its outside interface.

A converter $\pi$ connected to the interface set $\mathcal{I}$ of a resource $\mathbf{R}$ yields a new resource $\mathbf{R}' := \pi^{\mathcal{I}}\mathbf{R}$. The interfaces of $\mathbf{R}'$ inside the set $\mathcal{I}$ are the interfaces emulated by $\pi$. Note that converter application at disjoint interface sets commutes since only the final system behavior matters. A protocol can be modelled as a tuple of converters with pairwise disjoint interface sets.

A *distinguisher* $\mathbf{D}$ is a special kind of environment that connects to all interfaces of a resource $\mathbf{R}$ and sends queries to them. $\mathbf{D}$ has access to the global event history and can append events that cannot be added by $\mathbf{R}$. At any point, the distinguisher can end its interaction by outputting a bit. The advantage of a distinguisher is defined as

$$\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{S}) := |\Pr[\mathbf{D}^{\mathcal{E}}(\mathbf{R}) = 1] - \Pr[\mathbf{D}^{\mathcal{E}}(\mathbf{S}) = 1]|,$$

$\mathbf{D}^{\mathcal{E}}$ meaning that the distinguisher has oracle access to the global event history $\mathcal{E}$.

**Specifications** An important concept of CC is the one of *specifications*. Systems are grouped according to desired or assumed properties that are relevant to the user, while other properties are ignored on purpose. A specification $\mathcal{S}$ is a set of resources that have the same interface set and share some properties, for example confidentiality. In order to construct this set of confidential resources, one can use a specification of assumed resources $\mathcal{R}$ and a protocol $\pi$, and show that the specification $\pi\mathcal{R}$ satisfies confidentiality. Proving security is thus proving that $\pi\mathcal{R} \subseteq \mathcal{S}$, sometimes written as $\mathcal{R} \xrightarrow{\pi} \mathcal{S}$, and we say that the protocol $\pi$ constructs the specification $\mathcal{S}$ from the specification $\mathcal{R}$. The composition property of the framework comes from the transitivity of inclusion. Formally, for specifications $\mathcal{R}, \mathcal{S}$ and $\mathcal{T}$ and protocols $\pi$ for $\mathcal{R}$ and $\pi'$ for $\mathcal{S}$, we have

$$\mathcal{R} \xrightarrow{\pi} \mathcal{S} \wedge \mathcal{S} \xrightarrow{\pi'} \mathcal{T} \Rightarrow \mathcal{R} \xrightarrow{\pi' \circ \pi} \mathcal{T}$$

We use the real-world/ideal-world paradigm, and often refer to $\pi\mathcal{R}$ and $\mathcal{S}$ as the real and ideal-world specifications respectively, to understand security statements. Those statements say that the real-world is "just as good" as the ideal one, meaning that it does not matter whether parties interact with an arbitrary element of $\pi\mathcal{R}$ or one of $\mathcal{S}$. This means that the guarantees of the ideal specification $\mathcal{S}$ also apply in the real world where an assumed resource is used together with the protocol.

In this work, we use *simulators*, *i.e.*, converters that translate behaviors of the real world to the ideal world, to make the achieved security guarantees obvious. For example, we will model confidential servers as a specification $\mathcal{S}$ that only leaks the data length, combined with an arbitrary simulator $\sigma$, and we will show that $\pi\mathcal{R} \subseteq \sigma\mathcal{S}$. It is then clear that the adversary cannot learn anything more that the data length.

**Relaxations** In order to talk about computational assumptions, post-compromise security or other security notions, the CC framework relies on *relaxations* which are mappings from specifications to larger, and thus weaker, *relaxed specifications*. The idea of relaxation is that, if we are happy with constructing specification $\mathcal{S}$ in some context, then we are also happy with constructing its relaxed variant. One common example of this is computational security. Let $\epsilon$ be a function that maps distinguishers $\mathbf{D}$ to the winning probability, in $[0, 1]$, of a modified distinguisher $\mathbf{D'}$ (the reduction) on the underlying computational problem. Formally,

**Definition 1.** *Let $\epsilon$ be a function that maps distinguishers to a value in $[0, 1]$. Then, for a resource $\mathbf{R}$, the reduction relaxation $\mathbf{R}^{\epsilon}$ is defined as*

$$\mathbf{R}^{\epsilon} := \{\mathbf{S} \mid \forall \mathbf{D}, \Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{S}) \leq \epsilon(\mathbf{D})\}$$

*This (in fact any) relaxation can be extended to a specification $\mathcal{R}$ by defining $\mathcal{R}^\epsilon := \cup_{\mathbf{R}\in\mathcal{R}}\mathbf{R}^\epsilon$.*

The other relaxation that we will use is the *interval-wise relaxation* introduced in [8]. Given two predicates $P_1(\mathcal{E})$ and $P_2(\mathcal{E})$ on the global event history, the interval-wise relaxation $\mathbf{R}^{[P_1,P_2]}$ is the set of all resources that must behave like $\mathbf{R}$ in the time interval starting when $P_1(\mathcal{E})$ becomes true and ending when $P_2(\mathcal{E})$ becomes true. Outside of this interval, we have no guarantees on how the resources behave.

The two relaxations we use have nice composition properties, mainly they are compatible together and with parallel and sequential protocol applications, as shown in [8]. This means that all the constructions presented in this work can be used in a modular fashion inside bigger constructions, without needing to write a new security proof.

## 3 The Updatable Server Memory Resource

### 3.1 The basic USMR

In [1], Maurer and Badertscher propose an instantiation of the client-server setting in CC. They notably introduce the basic Server Memory Resource (**SMR** for short), where an honest client can read and write data on an outsourced memory owned by a potentially dishonest server. We adapt their ideas to the use cases of $\mathsf{UE}$ schemes by introducing a different type of **SMR**, namely the Updatable Server-Memory Resource (**USMR** for short). One of the main differences is that, in the **USMR**, the server is now considered as semi-honest (through its interface $\mathsf{S}$), since it has to participate in the (interactive) protocol when receiving tokens and updating its database. In Maurer's work, the client was the only party to take part in the protocol, the role of the server being limited to storing data. In our case, the server agrees to apply a converter to control its behavior for a subset of its capabilities. We gather these capabilities in a sub-interface of $\mathsf{S}$ denoted by $\mathsf{S}.1$. Since the server is only semi-honest, it gives no guarantees about how it will use its remaining capabilities. We gather those under the sub-interface $\mathsf{S}.2$.

Another difference between the **USMR** and the **SMR** is that the client, through its interface $\mathsf{C}$, can interact with the server by switching the value of a boolean NeedUpdate. The server is given the capability of checking this boolean to see if it needs to take actions and then switch the value back, all of this being done at the server's interface. While the **USMR** needs an update, *i.e.* as long as NeedUpdate is set to `true`, the capabilities of the client at its interface are disabled. This addition is helpful to model $\mathsf{UE}$ schemes since they are inherently interactive : the client needs to send a token to the server which, upon reception, needs to use it to update the database. This approach is general enough for the **USMR** to be used when modeling other interactive protocols in the outsourced storage setting. A formal description of the **USMR** is given in figure 1.

**Resource USMR$_{\Sigma,n}$**

**Initialization**

NeedUpdate $\leftarrow$ `false`
$\mathbb{M} \leftarrow [\,]$

**Interface** C

**Input:** $(\texttt{read}, i) \in [n]$
  **if not** NeedUpdate **then**
    **return** $\mathbb{M}[i]$
**Input:** $(\texttt{write}, i, x) \in [n] \times \Sigma$
  **if not** NeedUpdate **then**
    $\mathbb{M}[i] \leftarrow x$
**Input:** `askUpdate`
  **if not** NeedUpdate **then**
    NeedUpdate $\leftarrow$ `true`
**Input:** `getStatus`
  **return** needUpdate

**Sub-Interface** S.1 of **Interface** S

**Input:** $(\texttt{read}, i) \in [n]$
  **return** $\mathbb{M}[i]$
**Input:** $(\texttt{write}, i, x) \in [n] \times \Sigma$
  $\mathbb{M}[i] \leftarrow x$
**Input:** `update`
  **if** needUpdate **then**
    needUpdate $\leftarrow$ `false`

**Sub-Interface** S.2 of **Interface** S

**Input:** $(\texttt{leak}, i) \in [n]$
  **return** $\mathbb{M}[i]$
**Input:** `getStatus`
  **return** needUpdate

**Fig. 1.** The updatable server-memory resource **USMR** with finite alphabet $\Sigma$ and memory size $n$. Interface S guarantees that it will endorse an honest behavior, through the application of a converter, at its sub-interface S.1. However, no such guarantees are offered at its sub-interface S.2.

### 3.2 The Updatable Key Resource UpdKey

We have so far described the **USMR**, a Server-Memory Resource to which we want to apply a $\mathsf{UE}$ scheme to strengthened the security guarantees of the client. To do so, we need to model the use of cryptographic keys and update tokens needed by $\mathsf{UE}$ schemes. This is why we introduce an Updatable Key Resource, named **UpdKey**, whose role is to model the existence, the operations and the availability of keys and update tokens. In the following, let $\mathcal{K}$ be the key space of $\mathsf{UE}$ schemes. Given $k$ and $k'$ two keys in $\mathcal{K}$, the notation $\Delta \leftarrow \mathcal{T}(k, k')$ denotes the assignation, to the variable $\Delta$, of the token that updates ciphertexts encrypted under the key $k$ to ones encrypted under $k'$.

At interface $\mathsf{S}$, we also distinguish between honestly reading the memory - through `read` requests at sub-interface $\mathsf{S}.1$ - to update a ciphertext without trying to use this information against the client; and maliciously reading (we prefer to say *leaking*) the memory - through `leak` requests at sub-interface $\mathsf{S}.2$ - to gain information and try to break the confidentiality guarantees of the client. Similarly, the `fetchToken` request is always accessible at sub-interface $\mathsf{S}.1$, since the protocol used at this interface will prevent the information it provides to be maliciously used, while the request `leakToken` at interface $\mathsf{S}.2$ requires that a special event $\mathcal{E}^{\texttt{leaked}}_{\mathsf{Token},i}$ has been triggered before returning the update token to epoch $i$, which can be used for malicious purposes. This event is triggered by the environment which, in CC, can be given an interface that is usually denoted by $\mathsf{W}$ for world interface.

This separation between `read`/`leak` and `fetchToken`/`leakToken` requests is important because it allows us to describe the security guarantees of the system more precisely. Indeed, since the server needs to retrieve the token to update the ciphertexts, if we consider that this token "leaked", it becomes impossible to express the post-compromise security guarantees brought by $\mathsf{UE}$ schemes. This is because if all tokens leak, a single key exposure compromises the confidentiality of ciphertexts for all subsequent epochs.

### 3.3 An Updatable Encryption protocol for the USMR

The main focus of this work being the study of the security guarantees brought by the use of $\mathsf{UE}$ schemes, we have to define an encryption protocol based on $\mathsf{UE}$ for the **USMR** and study its effects. Since we work in the CC framework, we describe our protocol as a pair of converters $(\mathsf{ue}_{\mathsf{cli}}, \mathsf{ue}_{\mathsf{ser}})$ where $\mathsf{ue}_{\mathsf{cli}}$ will be plugged in interface $\mathsf{C}$ and $\mathsf{ue}_{\mathsf{ser}}$ in sub-interface $\mathsf{S}.1$ of the **USMR**. A formal description of $\mathsf{ue}_{\mathsf{cli}}$ (resp. $\mathsf{ue}_{\mathsf{ser}}$) can be found in figure 3 (resp. figure 4).

### 3.4 The confidential USMR

The security guarantees of the **USMR** can be improved by requiring confidentiality for the client's data. The resulting resource is called confidential **USMR** and we will refer to it as **cUSMR**. In practice, this means that, on a $(\texttt{leak}, i)$ request at interface $\mathsf{S}$, only the length of $\mathbb{M}[i]$ is returned to the adversary and

**Resource UpdKey**

**Initialization**

$e \leftarrow 1, \mathbb{K} \leftarrow [\,], \mathbb{T} \leftarrow [\perp]$
$k \twoheadleftarrow \mathcal{K}, \mathbb{K} \leftarrow \mathbb{K} \,||\, k$
$\mathcal{E} \xleftarrow{+} \mathcal{E}_e^{\texttt{epoch}}$

**Interface** C

**Input:** `fetchKey`
  **return** $\mathbb{K}[e]$
**Input:** `nextEpoch`
  $k_{e+1} \twoheadleftarrow \mathcal{K}$
  $\mathbb{K} \leftarrow \mathbb{K} \,||\, k_{e+1}$
  $\Delta_{e+1} \leftarrow \mathcal{T}(k_e, k_{e+1})$
  $\mathbb{T} \leftarrow \mathbb{T} \,||\, \Delta_{e+1}$
  $e \leftarrow e + 1$
  $\mathcal{E} \xleftarrow{+} \mathcal{E}_e^{\texttt{epoch}}$
  **return** $k_{e+1}$

**Sub-Interface** S.1 of **Interface** S

**Input:** $(\texttt{fetchToken}, i)$
  **if** $2 \leq i \leq e$ **then**
    **return** $\mathbb{T}[i]$
  **else**
    **return** $\perp$

**Sub-Interface** S.2 of **Interface** S

**Input:** $(\texttt{leakKey}, i)$
  **if** $i \leq e$ **and** $\mathcal{E}_{\mathsf{Key},i}^{\texttt{leaked}}$ **then**
    **return** $\mathbb{K}[i]$
  **else**
    **return** $\perp$
**Input:** $(\texttt{leakToken}, i)$
  **if** $2 \leq i \leq e$ **and** $\mathcal{E}_{\mathsf{Token},i}^{\texttt{leaked}}$ **then**
    **return** $\mathbb{T}[i]$
  **else**
    **return** $\perp$

**Fig. 2.** The updatable key (with its associated token) resource **UpdKey**. For interface S, we use the same distinction between its sub-interfaces S.1 and S.2 as in the **USMR** description.

**Converter** $\mathsf{ue}_{\mathsf{cli}}$

**Initialization**

$k \leftarrow$ `fetchKey` at interface $\mathsf{C}$ of **UpdKey**

**Interface** `out`

**Input:** $(\texttt{read}, i) \in [n]$
  $c \leftarrow (\texttt{read}, i)$ at interface $\mathsf{C}$ of **USMR**
  **if** $c \neq \perp$ **then**
    $m \leftarrow \mathsf{Dec}_k(c)$
    **if** $m \neq \diamond$ **then**
      **return** $m$
**Input:** $(\texttt{write}, i, x) \in [n] \times \Sigma$
  $c \leftarrow \mathsf{Enc}_k(x)$
  Send $(\texttt{write}, i, c)$ at interface $\mathsf{C}$ of **USMR**
**Input:** `askUpdate`
  $u \leftarrow$ `getStatus` at interface $\mathsf{C}$ of **USMR**
  **if not** $u$ **then**
    $k \leftarrow$ `nextEpoch` at interface $\mathsf{C}$ of **UpdKey**
    Send `askUpdate` at interface $\mathsf{C}$ of **USMR**

**Fig. 3.** The client's converter $\mathsf{ue}_{\mathsf{cli}}$ for $\mathsf{UE}$ scheme $(\mathsf{KG}, \mathsf{TG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Upd})$ with decryption error symbol $\diamond$.

**Converter** $\mathsf{ue}_{\mathsf{ser}}$

**Initialization**

$\Delta \leftarrow \perp$, $e \leftarrow 1$

**Interface** `out`

**Input:** `update`
  $u \leftarrow$ `getStatus` at interface $\mathsf{S}$ of **USMR**
  **if** $u$ **then**
    $e \leftarrow e + 1$
    $\Delta \leftarrow (\texttt{fetchToken}, e)$ at interface $\mathsf{S}$ of **UpdKey**
    **for** $i = 1 \ldots n$ **do**
      $c \leftarrow (\texttt{read}, i)$ at interface $\mathsf{S}$ of **USMR**
      Send $(\texttt{write}, i, \mathsf{Upd}_\Delta(c))$ at interface $\mathsf{S}$ of **USMR**
    Send `update` at interface $\mathsf{S}$ of **USMR**
**Input:** $(\texttt{read}, i)$, $(\texttt{write}, i, x)$, or $(\texttt{fetchToken}, e)$
  **return** $\perp$

**Fig. 4.** The server's converter $\mathsf{ue}_{\mathsf{ser}}$ for $\mathsf{UE}$ scheme $(\mathsf{KG}, \mathsf{TG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Upd})$. Since the server is semi-honest, the converter monitors the behavior of the requests at subinterface $\mathsf{S}.1$. The server guarantees that updates are done correctly (using the $\mathsf{UE}$ scheme) through the `update` request, and that the requests `read`, `write` and `fetchToken` are only used to update the ciphertexts and not to gain information to break the confidentiality of the data. These requests are thus disabled at its interface `out` but they can still be used internally by the converter.

**Resource cUSMR**

**Sub-Interface** S.1 of **Interface** S

**Input:** `update`
  **if** NeedUpdate **then**
    NeedUpdate $\rightarrow$ `false`

**Sub-Interface** S.2 of **Interface** S

**Input:** $(\texttt{leak}, i) \in [n]$
  **return** $|\mathbb{M}[i]|$
**Input:** `getStatus`
  **return** NeedUpdate

**Fig. 5.** The confidential and updatable server-memory resource **cUSMR**. Only differences with **USMR** are shown.

not the $i$-th entry itself. The `read` and `write` capabilities of sub-interface S.1 are removed. The resource **cUSMR** is described in figure 5.

In order to build the **cUSMR** we will use the construction notion of the CC framework, as used in [1]. This notion is illustrated in figure 6. One difference with the work of [1] is that, although there is a protocol plugged in interface S, this interface is only semi-honest and doesn't belong to the so-called "honest parties". We thus need to plug in a simulator at this interface in the ideal world if we hope to achieve any meaningful construction. One way to see this is to consider the `read` capabilities at interfaces C and S when a UE scheme is used. Since the client holds the decryption key, the interface C is able to retrieve the plaintexts corresponding to the ciphertexts stored in memory. This is not true for the interface S since it has only access to the update token but not to the key under normal circumstances. Since there is no encryption in the ideal world, we need a simulator to simulate ciphertexts when S sends `read` requests at its interface, as otherwise distinguishing between the two worlds would be trivial.

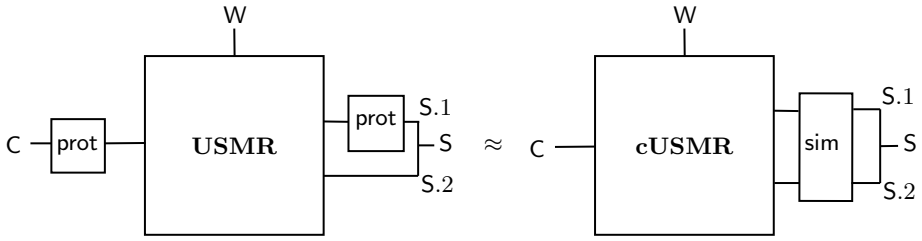


**Fig. 6.** The construction notion for **USMR**. On the left, the weaker **USMR** equipped with a protocol. On the right, the stronger **cUSMR** equipped with a simulator. The construction is deemed secure if there exists a simulator for which the two systems are indistinguishable.

It is important to stress that the security of UE schemes is thought of in the context of adaptive adversaries, where the use of a UE scheme should bring post-compromise confidentiality guarantees to the client. In this work, we thus

consider adaptive adversaries and we will use the CC framework to give a clear, simple and composable description of the post-compromise security guarantees given by UE schemes. To this end, the extension of CC developed by Jost and Maurer in [8] will prove to be particularly useful.

At this point, we precise the claim made by Boyd *et al.* [3], namely that their IND-UE security notion is better than the IND-ENC+UPD notions, in that it hides the age of ciphertexts. By age, we mean the last epoch in which the ciphertext was freshly written to the database. We show that this is only true when ciphertexts can leak at most one time per epoch. Indeed, if a ciphertext can leak at least two times per epoch, the adversary can use its first (resp. second) leak at the start (resp. end) of each epoch. If the ciphertext has changed between the two leaks, it must have been rewritten during this epoch and its age is now 0. If it has not changed, then its age is incremented. We see that in this setting, the age of ciphertexts cannot be protected.

In the rest of this work, we will distinguish between resources that only allow one leak per ciphertext per epoch, denoted by a 1 in the exponent (*e.g.* $\mathbf{USMR}^1$), and resources that allow two or more leaks per ciphertext per epoch, denoted by a + in the exponent (*e.g.* $\mathbf{USMR}^+$). If the number of leaks doesn't matter we will omit the exponent.

## 4 Exact security of UE schemes

### 4.1 At most one leak per entry per epoch : the CPA case

In this section, we work with $\mathbf{USMR}^1$ where the attacker can leak entries of the database at its interface S. This capability allows the distinguisher (which is connected to every interface of the system) to build an encryption oracle. Indeed, the distinguisher can use the client interface C to write messages of its choice into the database, and then leak the ciphertexts associated to these messages by sending a `leak` request at interface S. This fact motivates the use of a CPA security notion since it's tailored to bring security in the presence of such an encryption oracle.

In this case, the simulator $\sigma_{\mathsf{CPA}}$ works as follows. It simulates the epoch keys and tokens and, on a $(\texttt{leakKey}, i)$ or $(\texttt{leakToken}, i)$ request, it checks if the event $\mathcal{E}^{\texttt{leaked}}_{\mathsf{Key},i}$ (respectively $\mathcal{E}^{\texttt{leaked}}_{\mathsf{Token},i}$) exists in the Global Event History, and leaks the corresponding epoch key (respectively token) to the adversary if it is the case, and $\perp$ otherwise. On a $(\texttt{leak}, i)$ request, the simulator forwards it to the ideal resource to get a length $\ell$ and returns a fresh encryption of a random plaintext of length $\ell$ under the current epoch key.

**IND-UE-CPA security is sufficient for a secure construction of cUSMR that hides the age** The fact that IND-UE-CPA is sufficient to construct $\mathbf{cUSMR}^1$ from $\mathbf{USMR}^1$ is expressed in the following theorem.

**Theorem 1.** *Let $\Sigma$ be a finite alphabet and $n \in \mathbb{N}$. The protocol $\mathsf{ue} := (\mathsf{ue}_{\mathsf{cli}}, \mathsf{ue}_{\mathsf{ser}})$ described in figures 3 and 4 based on a $\mathsf{UE}$ scheme constructs the $\mathbf{cUSMR}^1_{\Sigma,n}$ from the basic $\mathbf{USMR}^1_{\Sigma,n}$, with respect to the simulator $\sigma_{\mathsf{CPA}}$ described in 4.1 and the dummy converter $\mathsf{honSrv}$ (that disables any adversarial behavior). More specifically, we construct reductions such that, for all distinguishers $\mathbf{D}$ in a set of distinguishers $\mathcal{D}$,*

$$\Delta^{\mathbf{D}}(\mathsf{honSrv}^{\mathsf{S}}\mathsf{ue}^{\mathsf{C}}_{\mathsf{cli}}\mathsf{ue}^{\mathsf{S}}_{\mathsf{ser}}\,\mathbf{USMR}^1_{\Sigma,n}, \mathsf{honSrv}^{\mathsf{S}}\,\mathbf{cUSMR}^1_{\Sigma,n}) = 0$$

$$\Delta^{\mathbf{D}}(\mathsf{ue}^{\mathsf{C}}_{\mathsf{cli}}\mathsf{ue}^{\mathsf{S}}_{\mathsf{ser}}\,\mathbf{USMR}^1_{\Sigma,n}, \sigma^{\mathsf{S}}_{\mathsf{CPA}}\,\mathbf{cUSMR}^1_{\Sigma,n}) \leq (2q{+}r) \sup_{\mathbf{D}' \in \mathcal{D}} \Delta^{\mathbf{D}'}(\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{UPD}}, \mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{ENC}})$$

*where $q$ (resp. $r$) is an upper bound on the number of writes (resp. updates) made by the distinguisher.*

The first condition, called *availability*, checks if the two systems behave in the same way when no adversary is present. It rules out trivial protocols that would ensure confidentiality by not writing data in memory for example. In all of this work, *availability* follows from the correctness of the $\mathsf{UE}$ schemes used. For clarity and conciseness, we will omit it in the rest of this work.

*Proof.* Let $\mathbf{R} := \mathsf{enc}^{\mathsf{C}}\,\mathsf{upd}^{\mathsf{S}}\,\mathbf{USMR}^1$ be the the real system and $\mathbf{I} := \sigma^{\mathsf{S}}_{\mathsf{CPA}}\,\mathbf{cUSMR}^1$ be the ideal system. In order to determine the advantage of a distinguisher in distinguishing $\mathbf{R}$ from $\mathbf{I}$, denoted by $\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{I})$, we proceed with a sequence of systems. We introduce a hybrid system $\mathbf{S}$, then we determine the distinguishing advantages $\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{S})$ and $\Delta^{\mathbf{D}}(\mathbf{S}, \mathbf{I})$, the triangular inequality allowing us to bound $\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{I})$ by the sum of those two advantages.

• Let $\mathbf{S}$ be a resource that behaves just like $\mathbf{R}$ except on query $(\texttt{leak}, i)$ where it leaks an encryption of a random plaintext of length $|\mathbb{M}[i]|$ instead of an encryption of $\mathbb{M}[i]$, if $\mathbb{M}[i]$ contains a fresh encryption and not an updated one. This happens if a query $(\texttt{write}, i, x)$ has been issued by the client in the current epoch. In the case when $\mathbb{M}[i]$ contains an updated version of a ciphertext, the two resources behave in the exact same way.

Let $q$ be an upper bound on the number of write queries issued to the systems. We define a hybrid resource $\mathbf{H}_i$ that behaves just like $\mathbf{R}$ on the first $i$ write queries and like $\mathbf{S}$ afterwards. Then we define a reduction $\mathbf{C}_i$ that behaves like $\mathbf{H}_i$ except it uses the game $\mathbf{G}^{\mathsf{ENC\text{-}CPA}}_b$ oracles instead of doing the $\mathsf{UE}$ operations by itself and on the $i$-th write request (of the form $(\texttt{write}, j, x)$) it challenges the game with input $(x, \bar{x})$, with $\bar{x}$ random of length $|x|$, to receive the ciphertext. We have

$$\mathbf{R} \equiv \mathbf{H}_q \text{ and } \mathbf{S} \equiv \mathbf{H}_0$$

and

$$\mathbf{H}_i \equiv \mathbf{G}^{\mathsf{ENC\text{-}CPA}}_0 \mathbf{C}_i \equiv \mathbf{G}^{\mathsf{ENC\text{-}CPA}}_1 \mathbf{C}_{i+1}$$

Indeed, this can be seen on the following timeline (1).

| $j$-th write query | $j < i$ | $j = i$ | $j = i+1$ | $j > i+1$ |
|---|---|---|---|---|
| $\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}}\mathbf{C}_i$ | $\mathsf{Enc}(x)$ | $\mathsf{Enc}(x)$ | $\mathsf{Enc}(\bar{x})$ | $\mathsf{Enc}(\bar{x})$ |
| $\mathbf{G}_1^{\mathsf{ENC\text{-}CPA}}\mathbf{C}_{i+1}$ | $\mathsf{Enc}(x)$ | $\mathsf{Enc}(x)$ | $\mathsf{Enc}(\bar{x})$ | $\mathsf{Enc}(\bar{x})$ |

**Table 1.** Leakage behavior of both systems for each $(\texttt{write}, \cdot, x)$ requests.

Let $\mathbf{C}_I$ be a reduction that samples $i \in [1, q]$ at random and behaves like $\mathbf{C}_i$ and define $\mathbf{D}' := \mathbf{D}\mathbf{C}_I$. We have,

$$\Pr[\mathbf{D}'(\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}}) = 1] = \frac{1}{q}\sum_{i=1}^{q} \Pr[\mathbf{D}(\mathbf{C}_i\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}}) = 1]$$

and

$$\begin{aligned}\Pr[\mathbf{D}'(\mathbf{G}_1^{\mathsf{ENC\text{-}CPA}}) = 1] &= \frac{1}{q}\sum_{i=1}^{q} \Pr[\mathbf{D}(\mathbf{C}_i\mathbf{G}_1^{\mathsf{ENC\text{-}CPA}}) = 1] \\ &= \frac{1}{q}\sum_{i=0}^{q-1} \Pr[\mathbf{D}(\mathbf{C}_i\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}}) = 1]\end{aligned}$$

Finally, the advantage of the distinguisher in distinguishing system $\mathbf{R}$ from $\mathbf{S}$ is

$$\begin{aligned}\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{S}) &= \Delta^{\mathbf{D}}(\mathbf{H}_q, \mathbf{H}_0) \\ &= \Delta^{\mathbf{D}}(\mathbf{C}_q\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}}, \mathbf{C}_0\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}}) \\ &= |\Pr[\mathbf{D}(\mathbf{C}_q\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}}) = 1] - \Pr[\mathbf{D}(\mathbf{C}_0\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}}) = 1]| \\ &= |\sum_{i=1}^{q} \Pr[\mathbf{D}(\mathbf{C}_i\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}}) = 1] - \sum_{i=0}^{q-1} \Pr[\mathbf{D}(\mathbf{C}_i\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}}) = 1]| \\ &= q \cdot |\Pr[\mathbf{D}'(\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}}) = 1] - \Pr[\mathbf{D}'(\mathbf{G}_1^{\mathsf{ENC\text{-}CPA}}) = 1]| \\ &= q \cdot \Delta^{\mathbf{D}'}(\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}}, \mathbf{G}_1^{\mathsf{ENC\text{-}CPA}})\end{aligned}$$

• Let us consider the systems $\mathbf{S}$ and $\mathbf{I}$. By definition, $\mathbf{S}$ behaves just like $\mathbf{I}$ except on query $(\texttt{leak}, i)$ where it leaks an updated ciphertext of an encryption of $\mathbb{M}[i]$ (instead of a fresh encryption of a random $\bar{x}$ in the ideal system) if $\mathbb{M}[i]$ contains an updated encryption and not a fresh one. In the case when $\mathbb{M}[i]$ contains a fresh ciphertext, the two resources behave in the exact same way. Namely, they leak an encryption of a random $\bar{x}$.

Let $r$ be an upper bound on the number of update queries issued to the systems. We define a hybrid resource $\mathbf{H}'_i$ that behaves just like $\mathbf{S}$ on the first $i$ update queries and like $\mathbf{I}$ afterwards. Then we define a reduction $\mathbf{C}'_i$ that behaves like $\mathbf{H}'_i$ except it uses the game $\mathbf{G}_{\mathsf{xxx}}^{\mathsf{IND\text{-}UE\text{-}CPA}}$, where $\mathsf{xxx} \in \{\mathsf{ENC}, \mathsf{UPD}\}$, oracles instead of doing the $\mathsf{UE}$ operations by itself and on the $i$-th update computation it challenges the game with input $(\bar{x}, c)$ to receive either a fresh encryption of the random $\bar{x}$ or the updated version of the ciphertext $c$. We have

$$\mathbf{S} \equiv \mathbf{H}'_r \text{ and } \mathbf{I} \equiv \mathbf{H}'_0$$

and

$$\mathbf{H}'_i \equiv \mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{UPD}}\mathbf{C}'_i \equiv \mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{ENC}}\mathbf{C}'_{i+1}$$

Indeed, this can be seen on the following timeline (2)

| $j$-th update query | $j < i$ | $j = i$ | $j = i+1$ | $j > i+1$ |
|---|---|---|---|---|
| $\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{UPD}}\mathbf{C}'_i$ | $\mathsf{Upd}(c)$ | $\mathsf{Upd}(c)$ | $\mathsf{Enc}(\bar{x})$ | $\mathsf{Enc}(\bar{x})$ |
| $\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{ENC}}\mathbf{C}'_{i+1}$ | $\mathsf{Upd}(c)$ | $\mathsf{Upd}(c)$ | $\mathsf{Enc}(\bar{x})$ | $\mathsf{Enc}(\bar{x})$ |

**Table 2.** Leakage behavior of both systems for each update requests.

Let $\mathbf{C}'_I$ be a reduction that samples $i \in [1, r]$ at random and behaves like $\mathbf{C}'_i$ and define $\mathbf{D}'' := \mathbf{D}\mathbf{C}'_I$. We have,

$$\Pr[\mathbf{D}''(\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{UPD}}) = 1] = \frac{1}{r}\sum_{i=1}^{r}\Pr[\mathbf{D}(\mathbf{C}'_i\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{UPD}}) = 1]$$

and

$$\Pr[\mathbf{D}''(\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{ENC}}) = 1] = \frac{1}{r}\sum_{i=0}^{r-1}\Pr[\mathbf{D}(\mathbf{C}'_i\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{UPD}}) = 1]$$

Finally, the advantage of the distinguisher in distinguishing system $\mathbf{S}$ from $\mathbf{I}$ is

$$\begin{aligned}
\Delta^{\mathbf{D}}(\mathbf{S},\mathbf{I}) &= \Delta^{\mathbf{D}}(\mathbf{H}'_r,\mathbf{H}'_0)\\
&= \Delta^{\mathbf{D}}(\mathbf{C}'_r\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{UPD}},\mathbf{C}'_0\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{UPD}})\\
&= |\Pr[\mathbf{D}(\mathbf{C}'_r\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{UPD}}) = 1] - \Pr[\mathbf{D}(\mathbf{C}'_0\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{UPD}}) = 1]|\\
&= |\sum_{i=1}^{r}\Pr[\mathbf{D}(\mathbf{C}'_i\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{UPD}}) = 1] - \sum_{i=0}^{r-1}\Pr[\mathbf{D}(\mathbf{C}'_i\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{UPD}}) = 1]|\\
&= r\cdot|\Pr[\mathbf{D}''(\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{UPD}}) = 1] - \Pr[\mathbf{D}''(\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{ENC}}) = 1]|\\
&= r\cdot\Delta^{\mathbf{D}''}(\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{UPD}},\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{ENC}})
\end{aligned}$$

• We use the triangular inequality to conclude. Let $q$ be an upper bound on the number of writes and $r$ be an upper bound on the number of updates. The advantage of the distinguisher in distinguishing the real system $\mathbf{R}$ from the ideal one $\mathbf{I}$ is

$$\begin{aligned}
\Delta^{\mathbf{D}}(\mathbf{R},\mathbf{I}) &\le \Delta^{\mathbf{D}}(\mathbf{R},\mathbf{S}) + \Delta^{\mathbf{D}}(\mathbf{S},\mathbf{I})\\
&= q\cdot\Delta^{\mathbf{D}'}(\mathbf{G}^{\mathsf{ENC\text{-}CPA}}_0,\mathbf{G}^{\mathsf{ENC\text{-}CPA}}_1) + r\cdot\Delta^{\mathbf{D}''}(\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{UPD}},\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{ENC}})\\
&= 2q\cdot\Delta^{\mathbf{D}'\mathbf{C}''}(\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{UPD}},\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{ENC}}) + r\cdot\Delta^{\mathbf{D}''}(\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{UPD}},\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{ENC}})\\
&\le (2q+r)\cdot\Delta^{\mathcal{D}}(\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{UPD}},\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{ENC}})^{5}
\end{aligned}$$

Where the reduction $\mathbf{C}''$ is given by Boyd *et al.* in [3] to prove the following

[5] Since we can also split the games the other way and consider $\mathsf{UPD\text{-}CPA}$ security first, we can replace $2q + r$ with $\min\{2q + r, q + 2r\}$.

**Proposition 1.** *Let $\Pi$ be a* UE *scheme. For any* ENC-CPA *adversary* $\mathbf{A}$ *against* $\Pi$*, there exists a reduction* $\mathbf{C}''$ *such that*

$$\Delta^{\mathbf{A}}(\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}}, \mathbf{G}_1^{\mathsf{ENC\text{-}CPA}}) \leq 2 \cdot \Delta^{\mathbf{AC}''}(\mathbf{G}_{\mathsf{ENC}}^{\mathsf{IND\text{-}UE\text{-}CPA}}, \mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}CPA}})$$

We also use the notation $\Delta^{\mathcal{D}}(\mathbf{X}, \mathbf{Y}) = \sup_{\mathbf{D} \in \mathcal{D}} \Delta^{\mathbf{D}}(\mathbf{X}, \mathbf{Y})$. □

**IND-UE-CPA security is necessary for a secure construction of cUSMR** Recall that we are working in the context of restricted leakage, where the adversary is only allowed to leak each entry of the database at most one time per epoch. In order to show that the IND-UE-CPA security notion is necessary to securely construct the $\mathbf{cUSMR}^1$ from a $\mathbf{USMR}^1$ using a UE scheme, we are going to use a technique introduced by Maurer *et al.* in [5]. Keeping our notation, we will start from the real system $\mathbf{R} := \mathsf{ue}_{\mathsf{cli}}^{\mathsf{C}} \mathsf{ue}_{\mathsf{ser}}^{\mathsf{S}}\, \mathbf{USMR}^1$ and the ideal one $\mathbf{I} := \sigma^{\mathsf{S}}\, \mathbf{cUSMR}^1$, where this time an arbitrary simulator $\sigma$ is used, and use reductions to construct the resources $\mathbf{G}_{\mathsf{xxx}}^{\mathsf{IND\text{-}UE\text{-}CPA}}$ implementing the IND-UE-CPA games. Formally, we will describe two reductions $\mathbf{C}_0$ and $\mathbf{C}_1$ such that

$$\mathbf{C}_0\mathbf{R} \equiv \mathbf{G}_{\mathsf{ENC}}^{\mathsf{IND\text{-}UE\text{-}CPA}},\ \mathbf{C}_1\mathbf{R} \equiv \mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}CPA}} \text{ and } \mathbf{C}_0\mathbf{I} \equiv \mathbf{C}_1\mathbf{I}.$$

Let $\mathbf{A}$ be an adversary against the IND-UE-CPA security notion. Using the triangular inequality, we have

$$\begin{aligned}
\Delta^{\mathbf{A}}(\mathbf{G}_{\mathsf{ENC}}^{\mathsf{IND\text{-}UE\text{-}CPA}}, \mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}CPA}}) &= \Delta^{\mathbf{A}}(\mathbf{C}_0\mathbf{R}, \mathbf{C}_1\mathbf{R}) \\
&\leq \Delta^{\mathbf{A}}(\mathbf{C}_0\mathbf{R}, \mathbf{C}_0\mathbf{I}) + \underbrace{\Delta^{\mathbf{A}}(\mathbf{C}_0\mathbf{I}, \mathbf{C}_1\mathbf{I})}_{=0} + \\
&\quad \Delta^{\mathbf{A}}(\mathbf{C}_1\mathbf{I}, \mathbf{C}_1\mathbf{R}) \\
&= \Delta^{\mathbf{AC}_0}(\mathbf{R}, \mathbf{I}) + \Delta^{\mathbf{AC}_1}(\mathbf{R}, \mathbf{I})
\end{aligned}$$

Thus, if a UE scheme securely construct (through a protocol) $\mathbf{I}$ from $\mathbf{R}$, then for any distinguisher $\mathbf{D}$, the distinguishing advantage $\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{I})$ is "small" and using the above inequality, the advantage of any adversary $\mathbf{A}$ in distinguishing the IND-UE-CPA games will also be "small". We conclude that, under the above hypothesis, the UE scheme will be IND-UE-CPA secure.

• We are going to describe two reductions $\mathbf{C}_0$ and $\mathbf{C}_1$ that connect to a $\mathbf{USMR}^1$ (or $\mathbf{cUSMR}^1$) resource and implement the oracles of the $\mathbf{G}_{\mathsf{xxx}}^{\mathsf{IND\text{-}UE\text{-}CPA}}$ games. Let $n$ be an upper bound on the number of encryption queries issued to the games by the adversary. In the following, we consider resources of memory size $n+1$. This is done to have enough space to store the ciphertexts returned by the encryption oracle as well as the challenge in case the challenger asks for updates later on. Here is how the reductions implement the game oracles :

– On the $i$-th $\mathcal{O}.\mathsf{Enc}(x)$ query : $\mathbf{C}_b$ sends $(\mathtt{write}, i, x)$ at interface $\mathsf{C}$, then sends $(\mathtt{leak}, i)$ at interface $\mathsf{S}$ and outputs the result. $\mathbf{C}_b$ also keeps a copy of the result in its own internal memory.

– On $\mathcal{O}.\mathsf{Next}()$ : $\mathbf{C}_b$ sends `askUpdate` at interface $\mathsf{C}$ and `update` at interface $\mathsf{S}$.
– On $\mathcal{O}.\mathsf{Upd}(c)$ : $\mathbf{C}_b$ checks if $c$ is the challenge or an updated version of it, in which case it returns $\perp$, and if $c$ was present in memory in the previous epoch. $\mathbf{C}_b$ can do this by keeping track of the entries it leaks and, at the end of each epoch, $\mathbf{C}_b$ sends $(\texttt{leak}, i)$ requests for every position $i$ that has not been leaked in the current epoch yet and writes the results in its own internal memory. If $c$ was not present, it returns $\perp$. If it was, let $i$ be the index where $c$ was written, $\mathbf{C}_b$ sends $(\texttt{leak}, i)$ at interface $\mathsf{S}$ and outputs the result (and writes it in its own internal memory).
– On $\mathcal{O}.\mathsf{Corr}(\mathsf{elt}, \hat{e})$ where $\mathsf{elt} \in \{\mathsf{Key}, \mathsf{Token}\}$ : $\mathbf{C}_b$ triggers the event $\mathcal{E}^{\texttt{leaked}}_{\mathsf{elt},\hat{e}}$, sends $(\texttt{fetchelt}, \hat{e})$ at interface $\mathsf{S}$ and outputs the result.
– On $\mathcal{O}.\mathsf{Chall}(m, c)$ : Both reductions check that $c$ was present in memory in the previous epoch and that its updated version has not been leaked yet in the current epoch. They also check that $m' := \mathsf{Dec}_k(c)$ and $m$ have the same length, this can be done with a `read` request at interface $\mathsf{C}$. They return $\perp$ if it is not the case. Otherwise, let $i$ be the position where $c$ was written. The reductions then proceed like this :
  – $\mathbf{C}_0$ sends $(\texttt{write}, n+1, m)$ at interface $\mathsf{C}$ then it sends $(\texttt{leak}, n+1)$ at interface $\mathsf{S}$ and outputs the result.
  – $\mathbf{C}_1$ sends $(\texttt{leak}, i)$ at interface $\mathsf{S}$ and outputs the result.
– On $\mathcal{O}.\mathsf{UpdC}()$ : Let $i_b$ be the position where the challenge is written for $\mathbf{C}_b$. If a challenge has already been issued, $\mathbf{C}_b$ sends $(\texttt{leak}, i_b)$ at interface $\mathsf{S}$ and returns the result. If not, it returns $\perp$.

Moreover, the reductions have all the information to check for trivial wins. When connected to the real system $\mathbf{R}$, the reductions $\mathbf{C}_b$ exactly implements the games $\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{xxx}}$ since the system leaks the same encryptions and updates as the ones produce by the game and correctly implements all of its oracles. Their observable behaviors being the same, we have $\mathbf{C}_0\mathbf{R} \equiv \mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{ENC}}$ and $\mathbf{C}_1\mathbf{R} \equiv \mathbf{G}^{\mathsf{IND\text{-}UE\text{-}CPA}}_{\mathsf{UPD}}$.

Now, if we connect the reductions to the ideal system $\mathbf{I}$, where an arbitrary simulator $\sigma$ is used, their behaviors can only differ on the challenge call. When asked for the challenge on the pair $(m, c)$, reduction $\mathbf{C}_0$ writes $m$ at position $n+1$ then it outputs the result of $(\texttt{leak}, n+1)$ meanwhile $\mathbf{C}_1$ outputs the result of $(\texttt{leak}, i)$ where $i$ is the position where the ciphertext $c$ was already written. In the ideal system $\mathbf{I}$, the result of a `leak` request is always $\ell$, the length of the message stored. Since, letting $m'$ be the underlying plaintext of $c$, the reductions checked that the two messages $m$ and $m'$ had the same length, the inputs of the simulator $\sigma$ are the same and the behaviors are thus identical. In the following epochs, this remains true when the adversary calls the oracle $\mathcal{O}.\mathsf{UpdC}$ to get an updated version of the challenge ciphertext. We can check that we never go beyond the leakage condition, it is never the case that a reduction leaks the content of a position $j$ at more than one occasion in a given epoch. We conclude that the systems $\mathbf{C}_b\mathbf{I}$ are indistinguishable from one another. Written differently, we have $\mathbf{C}_0\mathbf{I} \equiv \mathbf{C}_1\mathbf{I}$ and the theorem is proven in the $\mathsf{IND\text{-}UE\text{-}CPA}$ case.

In the context of restricted leakage, we proved that the IND-UE-CPA security notion for UE schemes was the correct one to consider if clients want to protect the confidentiality and the age of the data they store on a remote server in the presence of an adversary only able to leak the contents of the server at most once per epoch. This use case is realistic since it includes the common scenario of data breaches : when attackers are able to dump the content of a whole database and run away with the data. In the context of such an intrusion we showed, through the use of Constructive Cryptography, how to assess the situation, *i.e.*, we are able to describe precisely the remaining security guarantees for the confidentiality of the content and the age of the data depending on what keys and tokens are available to the adversaries.

### 4.2 Any number of leaks : the CPA case

In this case the simulator $\sigma_{\mathsf{CPA}^+}$ works as follows. It simulates the epoch keys and tokens and, on a $(\texttt{leakKey}, i)$ or $(\texttt{leakToken}, i)$ request, it checks if the event $\mathcal{E}^{\texttt{leaked}}_{\mathsf{Key},i}$ (respectively $\mathcal{E}^{\texttt{leaked}}_{\mathsf{Token},i}$) exists in the Global Event History and leaks the corresponding epoch key (respectively token) to the adversary if it is the case and $\perp$ otherwise. The simulator uses the ideal history to know which entries of the database correspond to fresh encryptions or updated encryptions. Together with its simulated epoch keys and tokens, this allows the simulator to maintain a simulated memory (and a simulated history) where fresh encryptions (in the real world) are replaced with fresh encryptions of random plaintexts and updated ciphertexts (in the real world) are replaced with updates of ciphertexts of random plaintexts. On a $(\texttt{leak}, i)$ request, the simulator returns the $i$-th entry of its simulated memory.

#### ENC-CPA and UPD-CPA security are sufficient for a secure construction of cUSMR$^+$

**Theorem 2.** *Let $\Sigma$ be a finite alphabet and $n \in \mathbb{N}$. The protocol $\mathsf{ue} := (\mathsf{ue}_{\mathsf{cli}}, \mathsf{ue}_{\mathsf{ser}})$ described in figures 3 and 4 based on a* UE *scheme constructs the $\mathbf{cUSMR}^+_{\Sigma,n}$ from the basic $\mathbf{USMR}^+_{\Sigma,n}$, with respect to the simulator $\sigma_{\mathsf{CPA}^+}$ described in 4.2. More specifically, we construct reductions $\mathbf{C}'$ and $\mathbf{C}''$ such that, for all distinguishers $\mathbf{D}$,*

$$\Delta^{\mathbf{D}}(\mathsf{ue}^{\mathsf{C}}_{\mathsf{cli}}\mathsf{ue}^{\mathsf{S}}_{\mathsf{ser}}\,\mathbf{USMR}_{\Sigma,n}, \sigma^{\mathsf{S}}_{\mathsf{CPA}^+}\,\mathbf{cUSMR}_{\Sigma,n}) \leq q \cdot \Delta^{\mathbf{DC}'}(\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}}, \mathbf{G}_1^{\mathsf{ENC\text{-}CPA}}) + r \cdot \Delta^{\mathbf{DC}''}(\mathbf{G}_0^{\mathsf{UPD\text{-}CPA}}, \mathbf{G}_1^{\mathsf{UPD\text{-}CPA}})$$

*where $q$ (resp. $r$) is an upper bound on the number of writes (resp. updates) made by the distinguisher.*

The proof is very similar to the one detailed in the IND-UE-CPA case. It can be found in appendix B.

In [3] the authors argued that IND-UE-CPA security is stronger that both ENC-CPA and UPD-CPA combined. Since we show that the combination of

ENC-CPA and UPD-CPA security is sufficient to securely construct the $\mathbf{cUSMR}^+$ from the $\mathbf{USMR}^+$ equipped with the $(\mathsf{enc}, \mathsf{upd})$ converters, we conclude that IND-UE-CPA security cannot be necessary for this secure construction in the unrestricted leakage model. This notion is thus too strong in this setting and brings nothing more than the combination of ENC-CPA and UPD-CPA security. Moreover, we are now going to show that ENC-CPA and UPD-CPA security is the perfect security notion to protect the confidentiality of the data in the context of unrestricted leakage (where the age of the data cannot be protected).

**ENC-CPA and UPD-CPA security are necessary for a secure construction of $\mathbf{cUSMR}^+$** We use the same proof technique as in the IND-UE-CPA case. The proof can be found in appendix C.

We showed that, in the context of unrestricted leakage, the ENC-CPA and UPD-CPA security notions are together necessary and sufficient to construct a $\mathbf{cUSMR}^+$ from a $\mathbf{USMR}^+$ using a protocol using UE. This means that, for UE schemes, the combination of these two notions is the correct one to consider and aim for in this specific context. In particular, the use of a strictly stronger security notion like IND-UE-CPA does not provide stronger security guarantees to clients in the setting of unrestricted leakage.

### 4.3 At most one leak per entry per epoch : the CCA case

We now consider Updatable Server Memory Resources where the attacker can inject messages into the database at his interface S and more precisely at its sub-interface S.2. Formally, S has access to the $(\texttt{inject}, i, x)$ request which assigns the value $x$ to the $i$-th entry of the database $\mathbb{M}[i]$. Moreover, each message leaked by S through $(\texttt{leak}, i)$ requests, can be replayed at any time through a $(\texttt{replay}, i, j)$ request which replaces the $i$-th entry of the database by the $j$-th message leaked by S. This **USMR** with injections will be denoted by **iUSMR**, and its confidential variant **ciUSMR**. The resource **iUSMR** is described in figure 7. The description of resource **ciUSMR** is the same except for the return value of $(\texttt{leak}, i)$ requests which is replaced by $|\mathbb{M}[i]|$. When dealing with injections, we only consider UE schemes with deterministic updates.

Injections enhance the power of the distinguisher (which is connected to every interface of the system) in the following manner. Just like before, the distinguisher can use the client interface C to write messages of its choice into the database and then leak a ciphertext associated to this message by sending a leak request at interface S, essentially building an encryption oracle for itself, which motivated the use of a CPA security notion. In the present case, the distinguisher can also use injections at interface S to write ciphertexts of its choice directly into the database then use the read request at interface C to get the message associated to the injected ciphertext, essentially building a decryption oracle for itself. This new capability of the distinguisher motivates the use of a CCA security notion since it is tailored to deal with situations of the sort.

In this case the simulator $\sigma_{\mathsf{CCA}}$ works as follows. It simulates the epoch keys and tokens and, on a $(\texttt{leakKey}, i)$ or $(\texttt{leakToken}, i)$ request, it checks if the event

**Resource iUSMR**

**Initialization**

$L \leftarrow [\,], \ell \leftarrow 0$

**Sub-Interface** S.2 of **Interface** S

**Input:** $(\texttt{leak}, i) \in [n]$
  $L \leftarrow L \,||\, \mathbb{M}[i]$
  $\ell \leftarrow \ell + 1$
  **return** $\mathbb{M}[i]$
**Input:** $(\texttt{inject}, i, x) \in [n] \times \Sigma$
  $\mathbb{M}[i] \leftarrow x$
**Input:** $(\texttt{replay}, i, j) \in [n] \times \mathbb{N}^*$
  **if** $j \leq \ell$ **then**
    $\mathbb{M}[i] \leftarrow L[j]$

**Fig. 7.** The updatable server-memory resource with injections **iUSMR**. Only differences with **USMR** are shown.

$\mathcal{E}^{\texttt{leaked}}_{\mathsf{Key},i}$ (respectively $\mathcal{E}^{\texttt{leaked}}_{\mathsf{Token},i}$) exists in the Global Event History and leaks the corresponding epoch key (respectively token) to the adversary if it is the case and $\perp$ otherwise. On the $j$-th $(\texttt{leak}, i)$ request, the simulator forwards it to the ideal resource to get a length $\ell$, computes a fresh encryption of a random $\bar{x}$ of length $\ell$ under the current epoch key, denoted by $c$, records the pair $(j, c)$ and returns $c$. Finally, on an $(\texttt{inject}, i, c)$ request, $\sigma_{\mathsf{CCA}}$ checks if it recorded a pair $(j, c')$ where $c' = c$ or $c$ is an updated version of $c'$ and, if it is the case, it sends $(\texttt{replay}, i, j)$ at interface S of $\mathbf{ciUSMR}^1$. If not, $\sigma_{\mathsf{CCA}}$ tries to decrypt $c$ with the current epoch key to get a message $m$ or a decryption error $\diamond$. If there is no decryption error, $\sigma_{\mathsf{CCA}}$ sends $(\texttt{inject}, i, m)$ to the resource.

**The CCA notion is not necessary** A important result, originally given by Canetti in [4] and exploited by Maurer *et al.* in [5], is that the CCA security is unnecessarily strict for most applications. We show that it is the case for UE by showing that this notion is not necessary to bring confidentiality to the **iUSMR**. A detailed proof can be found in appendix D. With this in mind, we focus on studying the weaker RCCA notion in the following sections.

**IND-UE-RCCA is necessary for a secure construction of $\mathbf{ciUSMR}^1$ that hides the age**

*Proof.* We show that the IND-UE-RCCA security notion is necessary to securely construct the $\mathbf{ciUSMR}^1$ from a $\mathbf{iUSMR}^1$ using a UE scheme. Keeping our notations, we will start from the real system $\mathbf{R} := \mathsf{ue}^{\mathsf{C}}_{\mathsf{cli}}\mathsf{ue}^{\mathsf{S}}_{\mathsf{ser}}\,\mathbf{iUSMR}^1$ and the ideal one $\mathbf{I} := \sigma^{\mathsf{S}}\,\mathbf{ciUSMR}^1$, where this time an arbitrary simulator $\sigma$ is used, and use reductions to construct the resources $\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}RCCA}}_{\mathsf{xxx}}$ implementing the IND-UE-RCCA games. Formally, we will describe two reductions $\mathbf{C}_0$ and $\mathbf{C}_1$ such that

$$\mathbf{C}_0\mathbf{R} \equiv \mathbf{G}^{\mathsf{IND\text{-}UE\text{-}RCCA}}_{\mathsf{ENC}},\ \mathbf{C}_1\mathbf{R} \equiv \mathbf{G}^{\mathsf{IND\text{-}UE\text{-}RCCA}}_{\mathsf{UPD}} \text{ and } \mathbf{C}_0\mathbf{I} \equiv \mathbf{C}_1\mathbf{I}$$

Let $\mathbf{A}$ be an adversary against the IND-UE-RCCA security notion. Using the triangular inequality, we have

$$\begin{aligned}\Delta^{\mathbf{A}}(\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}RCCA}}_{\mathsf{ENC}}, \mathbf{G}^{\mathsf{IND\text{-}UE\text{-}RCCA}}_{\mathsf{UPD}}) &= \Delta^{\mathbf{A}}(\mathbf{C}_0\mathbf{R}, \mathbf{C}_1\mathbf{R}) \\ &\leq \Delta^{\mathbf{A}}(\mathbf{C}_0\mathbf{R}, \mathbf{C}_0\mathbf{I}) + \Delta^{\mathbf{A}}(\mathbf{C}_0\mathbf{I}, \mathbf{C}_1\mathbf{I}) + \\ &\quad \Delta^{\mathbf{A}}(\mathbf{C}_1\mathbf{I}, \mathbf{C}_1\mathbf{R}) \\ &= \Delta^{\mathbf{AC}_0}(\mathbf{R}, \mathbf{I}) + \Delta^{\mathbf{AC}_1}(\mathbf{R}, \mathbf{I})\end{aligned}$$

• We are going to describe two reductions $\mathbf{C}_0$ and $\mathbf{C}_1$ that connect to a $\mathbf{iUSMR}^1$ (or $\mathbf{ciUSMR}^1$) resource and implement the oracles of the $\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}RCCA}}_{\mathsf{xxx}}$ games. Let $n$ (resp. $q$) be an upper bound on the number of encryption (resp. decryption) queries issued to the games by the adversary. In the following, we consider resources of memory size $n + q + 1$. This is done to have enough space to store the ciphertexts returned by the encryption oracle, the ciphertexts sent to the decryption oracle, as well as the challenge plaintext in case the challenger ask for updates to the update oracle later on. We use the same reductions as in the CPA case, the only difference being the addition of the decryption oracle. Since its behavior depends on the challenge oracle, we also describe this oracle below.

- On $\mathcal{O}.\mathsf{Chall}(m_0, c_1)$ : Both reductions check that $c_1$ was present in memory in the previous epoch and that its updated version has not been leaked yet in the current epoch. They also check that $m_1 := \mathsf{Dec}_k(c_1)$ and $m_0$ have the same length, this can be done with a `read` request at interface $\mathsf{C}$. They return $\perp$ if it is not the case. Otherwise, the reductions keep $m_0$ and $m_1$ in memory and let $i$ be the position where $c_1$ was written. The reductions then proceed like this :
  - $\mathbf{C}_0$ sends $(\texttt{write}, n+q+1, m_0)$ at interface $\mathsf{C}$ then it sends $(\texttt{leak}, n+q+1)$ at interface $\mathsf{S}$ and outputs the result.
  - $\mathbf{C}_1$ sends $(\texttt{leak}, i)$ at interface $\mathsf{S}$ and outputs the result.
- On the $j$-th $\mathcal{O}.\mathsf{Dec}(c)$ :
  - Before the challenge call, $\mathbf{C}_b$ sends an $(\texttt{inject}, n + j, c)$ request at interface $\mathsf{S}$. Then, it sends a $(\texttt{read}, n + j)$ request at interface $\mathsf{C}$ to get a result $m$ and returns $m$.
  - After the challenge call, $\mathbf{C}_b$ sends the same requests as before to get the result $m$. Only this time, if $m \in \{m_0, m_1\}$ they return `test` and if not they return $m$.

Moreover, the reductions have all the information to check for trivial wins. When connected to $\mathbf{R}$, the reductions $\mathbf{C}_b$ exactly implements the games $\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}RCCA}}_{\mathsf{xxx}}$. Their observable behaviors being the same, we have $\mathbf{C}_0\mathbf{R} \equiv \mathbf{G}^{\mathsf{IND\text{-}UE\text{-}RCCA}}_{\mathsf{ENC}}$ and $\mathbf{C}_1\mathbf{R} \equiv \mathbf{G}^{\mathsf{IND\text{-}UE\text{-}RCCA}}_{\mathsf{UPD}}$.

Now, if we connect the reductions to the ideal system $\mathbf{I}$, where an arbitrary simulator $\sigma$ is used, their behaviors can only differ on the challenge call. When

asked for the challenge on the pair $(m_0, c_1)$, reduction $\mathbf{C}_0$ writes $m_0$ at position $n+q+1$ then it outputs the result of $(\texttt{leak}, n+q+1)$ meanwhile $\mathbf{C}_1$ outputs the result of $(\texttt{leak}, i)$ where $i$ is the position where the ciphertext $c_1$ was already written. In the ideal system $\mathbf{I}$, the result of a `leak` request is always the length of the message stored. Since, letting $m_1$ be the underlying plaintext of $c_1$, the reductions checked that the two messages $m_0$ and $m_1$ had the same length, the inputs of the simulator $\sigma$ are the same and the behaviors are thus identical. We can check that we never go beyond the leakage condition, it is never the case that a reduction leaks the content of a position $j$ at more than one occasion in a given epoch. We conclude that the systems $\mathbf{C}_b\mathbf{I}$ are indistinguishable from one another. Written differently, we have $\mathbf{C}_0\mathbf{I} \equiv \mathbf{C}_1\mathbf{I}$ and the theorem is proven in the IND-UE-RCCA case. □

In the context of restricted leakage, we proved that the IND-UE-RCCA security notion for UE schemes is the correct one to consider if clients want to protect the confidentiality and the age of the data they store on a remote server in the presence of an adversary who can inject arbitrary data and is only able to leak the contents of the server at most once per epoch.

**IND-UE-RCCA is sufficient for a secure construction of ciUSMR$^1$ that hides the age with sufficiently large message space** We show the following theorem which states that IND-UE-RCCA security is sufficient to ensure confidentiality when injections are possible and the message space is sufficiently large.

**Theorem 3.** *Let $\Sigma$ be a finite alphabet and $n \in \mathbb{N}$. The protocol $\mathsf{ue} := (\mathsf{ue}_{\mathsf{cli}}, \mathsf{ue}_{\mathsf{ser}})$ described in figures 3 and 4 based on a* UE *scheme constructs the $\mathbf{ciUSMR}^1_{\Sigma,n}$ from the basic $\mathbf{iUSMR}^1_{\Sigma,n}$, with respect to the simulator $\sigma_{\mathsf{CCA}}$ described in 4.3. More specifically, we construct reductions such that, for all distinguishers $\mathbf{D}$,*

$$\Delta^{\mathbf{D}}(\mathsf{ue}^{\mathsf{C}}_{\mathsf{cli}}\mathsf{ue}^{\mathsf{S}}_{\mathsf{ser}}\,\mathbf{iUSMR}^1_{\Sigma,n}, \sigma^{\mathsf{S}}_{\mathsf{CCA}}\,\mathbf{ciUSMR}^1_{\Sigma,n}) \leq$$

$$(2q+r)\Delta^{\mathbf{D}}(\mathbf{G}^{\mathsf{IND\text{-}UE\text{-}RCCA}}_{\mathsf{UPD}}, \mathbf{G}^{\mathsf{IND\text{-}UE\text{-}RCCA}}_{\mathsf{ENC}}) + \frac{(q+r+1)s}{|\mathcal{M}|}$$

*where $q$ (resp. $r, s$) is an upper bound on the number of writes (resp. updates, injections) made by the distinguisher.*

*Proof.* Let $\mathbf{R} := \mathsf{enc}^{\mathsf{C}}\,\mathsf{upd}^{\mathsf{S}}\,\mathbf{iUSMR}^1$ be the the real system and $\mathbf{I} := \sigma^{\mathsf{S}}_{\mathsf{CCA}}\,\mathbf{ciUSMR}^1$ be the ideal system, where $\sigma_{\mathsf{CCA}}$ is the simulator given in section 4.3. In order to determine the advantage of a distinguisher in distinguishing $\mathbf{R}$ from $\mathbf{I}$, denoted by $\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{I})$, we proceed with a sequence of systems.

• Let $\mathbf{S}$ be a resource that behaves just like $\mathbf{R}$ except on query $(\texttt{leak}, i)$ where it leaks an encryption of a random plaintext if $\mathbb{M}[i]$ contains a fresh encryption. On the $j$-th `leak` query, $\mathbf{S}$ records the pair $(j, c)$. On query $(\texttt{inject}, i, c)$, if $c$ is an encryption of a random plaintext already recorded in a pair $(j, c)$ or an updated version of such a pair, then $\mathbf{S}$ behaves like a $(\texttt{replay}, i, j)$ request

was issued instead. In other words, $\mathbf{S}$ behaves just like $\mathbf{R}$ except when leaking fresh encryptions and when injecting ciphertexts that have already leak where it behaves just like $\mathbf{I}$.

Let $q$ (resp. $s$) be an upper bound on the number of write (resp. injection) queries issued to the systems. We define a hybrid resource $\mathbf{H}_i$ that behaves just like $\mathbf{R}$ on the first $i$ write queries and like $\mathbf{S}$ afterwards. Then we define a reduction $\mathbf{C}_i$ that behaves like $\mathbf{H}_i$ except it uses the game $\mathbf{G}_b^{\mathsf{ENC\text{-}RCCA}}$ oracles instead of doing the $\mathsf{UE}$ operations by itself except on the $i$-th write request (of the form $(\texttt{write}, j, x)$) it challenges the game with input $(x, \bar{x})$, with $\bar{x}$ random of length $|x|$ to receive the ciphertext $\tilde{c}$. On a $(\texttt{inject}, j, c)$ request, if $c \neq \tilde{c}$, $\mathbf{C}_i$ sends $c$ to the game decryption oracle. If the answer is $\texttt{test}$, it assigns the value $x$ to the $j$-th database entry $\mathbb{M}[j]$. We have

$$\mathbf{R} \equiv \mathbf{H}_q \text{ and } \mathbf{S} \equiv \mathbf{H}_0$$

and, if $\mathbf{D}$ is a distinguisher, we have

$$\Delta^{\mathbf{D}}(\mathbf{C}_i\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}, \mathbf{C}_{i+1}\mathbf{G}_1^{\mathsf{ENC\text{-}RCCA}}) \leq \frac{s}{|\mathcal{M}|}$$

The two systems behave in the same way on write requests. However, a difference in behaviors comes up after the $i$-th write request and before the $i+1$-th one. Indeed, in this case, the system $\mathbf{C}_i\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}$ has issued a challenge on $(x, \bar{x})$ but $\mathbf{C}_{i+1}\mathbf{G}_1^{\mathsf{ENC\text{-}RCCA}}$ has not. This means that, before the $i+1$-th write request, on an $(\texttt{inject}, j, c)$ where $c$ is an encryption of $\bar{x}$, the decryption oracle of $\mathbf{C}_i\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}$ returns $\texttt{test}$ and $x$ is written at position $j$. Meanwhile, since there is no challenge yet, the decryption oracle of $\mathbf{C}_{i+1}\mathbf{G}_1^{\mathsf{ENC\text{-}RCCA}}$ returns $\bar{x}$ and $\bar{x}$ is written at position $j$. A $(\texttt{read}, j)$ request then leads to a distinction of the two systems. Finding such a ciphertext $c$ with $s$ possible injections happens with probability $s/|\mathcal{M}|$.

Moreover, we have

$$\Delta^{\mathbf{D}}(\mathbf{H}_q, \mathbf{C}_q\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}) = \Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{C}_q\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}) \leq \frac{s}{|\mathcal{M}|}$$

This is due to the fact that, after the challenge is issued on the $q$-th write query, the behavior of the two systems differs after an $(\texttt{inject}, j, c)$ request when $c := \mathsf{Enc}(\bar{x})$ where $\bar{x}$ was the random plaintext used by the reduction during the challenge call. The system $\mathbf{R}$ returns $\bar{x}$ on a subsequent $(\texttt{read}, j)$ request while $\mathbf{C}_q\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}$ returns the challenge plaintext $x$. Finding such a ciphertext with $s$ injections queries happens with probability $s/|\mathcal{M}|$.

We also point out that

$$\Delta^{\mathbf{D}}(\mathbf{C}_0\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}, \mathbf{H}_0) = \Delta^{\mathbf{D}}(\mathbf{C}_0\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}, \mathbf{S}) = 0$$

This is because we are using $\mathbf{C}_0$, and not $\mathbf{C}_q$ like above, so no challenge is issued and the two systems behave in the exact same way.

Now let $\mathbf{C}_I$ be a reduction that samples $i \in [1, q]$ at random and behaves like $\mathbf{C}_i$ and define $\mathbf{D}' := \mathbf{D}\mathbf{C}_I$. We have,

$$\Pr[\mathbf{D}'(\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}) = 1] = \frac{1}{q}\sum_{i=1}^{q} \Pr[\mathbf{D}(\mathbf{C}_i\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}) = 1]$$

and

$$\begin{aligned}\Pr[\mathbf{D}'(\mathbf{G}_1^{\mathsf{ENC\text{-}RCCA}}) = 1] =& \frac{1}{q}\sum_{i=1}^{q} \Pr[\mathbf{D}(\mathbf{C}_i\mathbf{G}_1^{\mathsf{ENC\text{-}RCCA}}) = 1] \\ &\leq \frac{s}{|\mathcal{M}|} + \frac{1}{q}\sum_{i=0}^{q-1} \Pr[\mathbf{D}(\mathbf{C}_i\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}) = 1]\end{aligned}$$

Finally, the advantage of the distinguisher in distinguishing system $\mathbf{R}$ from $\mathbf{S}$ is

$$\begin{aligned}\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{S}) &= \Delta^{\mathbf{D}}(\mathbf{H}_q, \mathbf{H}_0) \\ &\leq \Delta^{\mathbf{D}}(\mathbf{H}_q, \mathbf{C}_q\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}) + \Delta^{\mathbf{D}}(\mathbf{C}_q\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}, \mathbf{C}_0\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}) \\ &\quad + \Delta^{\mathbf{D}}(\mathbf{C}_0\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}, \mathbf{H}_0) \\ &\leq \Delta^{\mathbf{D}}(\mathbf{C}_q\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}, \mathbf{C}_0\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}) + \frac{s}{|\mathcal{M}|} \\ &\leq |\Pr[\mathbf{D}(\mathbf{C}_q\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}) = 1] - \Pr[\mathbf{D}(\mathbf{C}_0\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}) = 1]| + \frac{s}{|\mathcal{M}|} \\ &\leq |\sum_{i=1}^{q} \Pr[\mathbf{D}(\mathbf{C}_i\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}) = 1] - \sum_{i=0}^{q-1} \Pr[\mathbf{D}(\mathbf{C}_i\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}) = 1]| + \frac{s}{|\mathcal{M}|} \\ &\leq q \cdot |\Pr[\mathbf{D}'(\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}) = 1] - \Pr[\mathbf{D}'(\mathbf{G}_1^{\mathsf{ENC\text{-}RCCA}}) = 1]| + \frac{s(q+1)}{|\mathcal{M}|} \\ &\leq q \cdot \Delta^{\mathbf{D}'}(\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}, \mathbf{G}_1^{\mathsf{ENC\text{-}RCCA}}) + \frac{s(q+1)}{|\mathcal{M}|}\end{aligned}$$

• Let us consider the systems $\mathbf{S}$ and $\mathbf{I}$. By definition, $\mathbf{S}$ behaves just like $\mathbf{I}$ except on query $(\texttt{leak}, i)$ where it leaks an updated ciphertext of an encryption of $\mathbb{M}[i]$ if $\mathbb{M}[i]$ contains an updated encryption. In the case when $\mathbb{M}[i]$ contains a fresh ciphertext, the two resources behave in the exact same way. Namely, they leak an encryption of a random $\bar{x}$ of length $|\mathbb{M}[i]|$.

Let $r$ (resp. $s$) be an upper bound on the number of update (resp. injection) queries issued to the systems. We define a hybrid resource $\mathbf{H}'_i$ that behaves just like $\mathbf{S}$ on the first $i$ update queries and like $\mathbf{I}$ afterwards. Then we define a reduction $\mathbf{C}'_i$ that behaves like $\mathbf{H}'_i$ except it uses the game $\mathbf{G}_{\mathsf{xxx}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}$, where $\mathsf{xxx} \in \{\mathsf{ENC}, \mathsf{UPD}\}$, oracles instead of doing the $\mathsf{UE}$ operations by itself and on the $i$-th update, for ciphertext $c$, it challenges the game with input $(\bar{x}, c)$ to receive either a fresh encryption of a random $\bar{x}$ or the updated version of $c$. We have

$$\mathbf{S} \equiv \mathbf{H}'_r \text{ and } \mathbf{I} \equiv \mathbf{H}'_0$$

and, if $\mathbf{D}$ is a distinguisher, we have

$$\Delta^{\mathbf{D}}(\mathbf{C}'_i\mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}, \mathbf{C}'_{i+1}\mathbf{G}_{\mathsf{ENC}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}) \leq \frac{s}{|\mathcal{M}|}$$

The two systems behave in the same way on update requests. However, a difference in behaviors comes up after the $i$-th update request and before the $i+1$-th one. Indeed, in this case, the system $\mathbf{C}'_i\mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}$ has issued a challenge on $(\bar{x}, c)$ but $\mathbf{C}'_{i+1}\mathbf{G}_{\mathsf{ENC}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}$ has not. This means that, before the $i+1$-th update request, on an $(\texttt{inject}, j, c')$ where $c'$ is an encryption of $\bar{x}$, the decryption oracle of $\mathbf{C}'_i\mathbf{G}_0^{\mathsf{UPD\text{-}RCCA}}$ returns $\texttt{test}$ and $x$ (the underlying plaintext of $c$) is written at position $j$. Meanwhile, since there is no challenge yet, the decryption oracle of $\mathbf{C}'_{i+1}\mathbf{G}_{\mathsf{ENC}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}$ returns $\bar{x}$ and $\bar{x}$ is written at position $j$. A $(\texttt{read}, j)$ request then leads to a distinction of the two systems. Finding such a ciphertext $c'$ with $s$ possible injections happens with probability $s/|\mathcal{M}|$.

Let $\mathbf{C}'_I$ be a reduction that samples $i \in [1, r]$ at random and behaves like $\mathbf{C}'_i$ and define $\mathbf{D}'' := \mathbf{D}\mathbf{C}'_I$. We have,

$$\Pr[\mathbf{D}''(\mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}) = 1] = \frac{1}{r}\sum_{i=1}^{r} \Pr[\mathbf{D}(\mathbf{C}'_i\mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}) = 1]$$

and

$$\Pr[\mathbf{D}''(\mathbf{G}_{\mathsf{ENC}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}) = 1] \leq \frac{s}{|\mathcal{M}|} + \frac{1}{r}\sum_{i=0}^{r-1} \Pr[\mathbf{D}(\mathbf{C}'_i\mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}) = 1]$$

This time, since the injection behaviors of both systems are the same, we have

$$\mathbf{H}'_r \equiv \mathbf{C}'_r\mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}RCCA}} \quad \text{and} \quad \mathbf{H}'_0 \equiv \mathbf{C}'_0\mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}$$

Finally, the advantage of the distinguisher in distinguishing system $\mathbf{S}$ from $\mathbf{I}$ is

$$\begin{aligned}
\Delta^{\mathbf{D}}(\mathbf{S}, \mathbf{I}) &= \Delta^{\mathbf{D}}(\mathbf{H}'_r, \mathbf{H}'_0) \\
&= \Delta^{\mathbf{D}}(\mathbf{C}'_r\mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}, \mathbf{C}'_0\mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}) \\
&= |\Pr[\mathbf{D}(\mathbf{C}'_r\mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}) = 1] - \Pr[\mathbf{D}(\mathbf{C}'_0\mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}) = 1]| \\
&= |\sum_{i=1}^{r} \Pr[\mathbf{D}(\mathbf{C}'_i\mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}) = 1] - \sum_{i=0}^{r-1} \Pr[\mathbf{D}(\mathbf{C}'_i\mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}) = 1]| \\
&\leq r \cdot |\Pr[\mathbf{D}''(\mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}) = 1] - \Pr[\mathbf{D}''(\mathbf{G}_{\mathsf{ENC}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}) = 1]| + \frac{rs}{|\mathcal{M}|} \\
&\leq r \cdot \Delta^{\mathbf{D}''}(\mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}, \mathbf{G}_{\mathsf{ENC}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}) + \frac{rs}{|\mathcal{M}|}
\end{aligned}$$

• We use the triangular inequality to conclude. Let $q, r$ and $s$ be upper bounds on the number of writes, updates and injections, respectively. The advantage of the distinguisher in distinguishing the real system $\mathbf{R}$ from the ideal one $\mathbf{I}$ is

$$
\begin{aligned}
\Delta^{\mathbf{D}}(\mathbf{R},\mathbf{I}) &\leq \Delta^{\mathbf{D}}(\mathbf{R},\mathbf{S}) + \Delta^{\mathbf{D}}(\mathbf{S},\mathbf{I}) \\
&\leq q \cdot \Delta^{\mathbf{D}'}(\mathbf{G}_0^{\mathsf{ENC\text{-}RCCA}}, \mathbf{G}_1^{\mathsf{ENC\text{-}RCCA}}) + \\
&\quad r \cdot \Delta^{\mathbf{D}''}(\mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}, \mathbf{G}_{\mathsf{ENC}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}) + \frac{(q+r+1)s}{|\mathcal{M}|} \\
&\leq 2q \cdot \Delta^{\mathbf{D}'\mathbf{C}''}(\mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}, \mathbf{G}_{\mathsf{ENC}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}) + \\
&\quad r \cdot \Delta^{\mathbf{D}''}(\mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}, \mathbf{G}_{\mathsf{ENC}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}) + \frac{(q+r+1)s}{|\mathcal{M}|} \\
&\leq (2q+r) \cdot \Delta^{\mathcal{D}}(\mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}, \mathbf{G}_{\mathsf{ENC}}^{\mathsf{IND\text{-}UE\text{-}RCCA}}) + \frac{(q+r+1)s}{|\mathcal{M}|}
\end{aligned}
$$

Where the reduction $\mathbf{C}''$ is a straightforward adaptation, in the RCCA case, of the one given by Boyd *et al.* in [3] to prove the following in the CCA case

**Proposition 2.** *Let $\Pi$ be a* UE *scheme. For any* ENC-CCA *adversary* $\mathbf{A}$ *against $\Pi$, there exists a reduction* $\mathbf{C}''$ *such that*

$$
\Delta^{\mathbf{A}}(\mathbf{G}_0^{\mathsf{ENC\text{-}CCA}}, \mathbf{G}_1^{\mathsf{ENC\text{-}CCA}}) \leq 2 \cdot \Delta^{\mathbf{A}\mathbf{C}''}(\mathbf{G}_{\mathsf{ENC}}^{\mathsf{IND\text{-}UE\text{-}CCA}}, \mathbf{G}_{\mathsf{UPD}}^{\mathsf{IND\text{-}UE\text{-}CCA}})
$$

We also use the notation $\Delta^{\mathcal{D}}(\mathbf{X},\mathbf{Y}) = \sup_{\mathbf{D}\in\mathcal{D}} \Delta^{\mathbf{D}}(\mathbf{X},\mathbf{Y})$. □

### 4.4 Handling post-compromise security

In the previous sections, we precisely described the security guarantees brought by the use of UE schemes in the presence of an adversary that didn't leak keys. The goal of this section is to give an exact description of the post-compromise security guarantees given by UE schemes. Said differently, we want to explain how the security guarantees evolve after a key exposure.

When dealing with situations such as key exposures, composable frameworks usually stumble on an impossibility result called the *commitment problem*. This problem is the following : given a message $m$, how can an online simulator explain a simulated ciphertext $c$, generated without knowledge of $m$, with a key $k$ such that $c$ decrypts to $m$ under this key.

Thanks to a recent work of Jost and Maurer [8], the CC framework is well equipped to deal with this impossibility result. This is done through the use of interval-wise security guarantees. In CC, the interval-wise relaxation describes security guarantees within an interval delimited by predicates on the global event history. For example, we can describe security guarantees before and after the key leaks.

A UE scheme is said to have *uni-directional* updates if an update token can only move a ciphertext from the old key to the new key. A UE scheme supports *bi-directional* updates if the update token can additionally downgrade ciphertexts from the new key to the old key. Jiang [7] recently proved that schemes supporting uni-directional updates do not bring more security compared to those with bi-directional updates, in the sense that the security notions for uni

and bi-directional updates are proved to be equivalent. Thus, in what follows, we only focus on schemes with bi-directional updates.

In $\mathsf{UE}$ schemes, it is clear that the confidentiality of the user data is lost when an epoch key leaks. This security breach remains in subsequent epochs if the keys continue to leak or if successive update tokens leak. However, as soon as we encounter an epoch where neither the key nor the update token leaks, the confidentiality is restored. This remains true until a future epoch, where either a key leaks or consecutive update tokens leak until a key is finally exposed. This is due to the fact that ciphertexts can be upgraded and downgraded with update tokens to an epoch where a key is exposed.

In the epoch timeline, the areas where confidentiality is preserved are called *insulated regions*. They have been studied and used in previous works [11,10,3,7]. We describe those regions with their extreme left and right epochs. These pairs of epochs are called *firewalls*. We recall the definition used in [3].

**Definition 2.** *An insulated region with firewalls* $\mathsf{fwl}$ *and* $\mathsf{fwr}$ *is a consecutive sequence of epochs* $(\mathsf{fwl}, \ldots, \mathsf{fwr})$ *for which :*

1. *No key in the sequence of epochs* $(\mathsf{fwl}, \ldots, \mathsf{fwr})$ *is corrupted.*
2. *The tokens* $\Delta_{\mathsf{fwl}}$ *and* $\Delta_{\mathsf{fwr}+1}$ *are not corrupted, if they exist.*
3. *All tokens* $(\Delta_{\mathsf{fwl}+1}, \ldots, \Delta_{\mathsf{fwr}})$ *are corrupted.*

*The set of all firewall pairs is denoted by* $\mathcal{FW}$*. The set of all insulated regions is denoted by* $\mathcal{IR} := \cup_{(\mathsf{fwl},\mathsf{fwr})\in\mathcal{FW}}\{\mathsf{fwl}, \ldots, \mathsf{fwr}\}$.

The epochs where the confidentiality guarantees do not hold are the ones not found in $\mathcal{IR}$. The set of firewalls $\mathcal{FW}$ can easily be described using predicates on the global event history. This is done in the following manner.

$$\begin{aligned}\mathcal{FW} = \{(\mathsf{fwl}, \mathsf{fwr}) \mid & \mathsf{fwl} \leq \mathsf{fwr}, \\ & \forall e \in \{\mathsf{fwl}, \ldots, \mathsf{fwr}\}, \neg\mathcal{E}^{\mathtt{leaked}}_{\mathsf{Key},e}, \\ & \neg\mathcal{E}^{\mathtt{leaked}}_{\mathsf{Token},\mathsf{fwl}} \text{ and } \neg\mathcal{E}^{\mathtt{leaked}}_{\mathsf{Token},\mathsf{fwr}+1}, \\ & \forall e \in \{\mathsf{fwl}+1, \ldots, \mathsf{fwr}\}, \mathcal{E}^{\mathtt{leaked}}_{\mathsf{Token},e}\}\end{aligned}$$

With insulated regions defined and clearly identifiable through the global event history, we can modify our definitions of the **cUSMR** and **ciUSMR** in the following way. On a $(\mathtt{leak}, i)$ request, these resources now return $|\mathbb{M}[i]|$ if they are inside an insulated region and $\mathbb{M}[i]$ otherwise. That being said, we have everything we need to state our main theorem.

**Theorem 4.** *Let* $\pi_{\mathsf{UE}} := (\mathsf{ue}_{\mathsf{cli}}, \mathsf{ue}_{\mathsf{ser}})$ *be the protocol securing the data using a* $\mathsf{UE}$ *scheme. Then, for the Updatable Server-Memory Resources defined throughout this work, there exists simulators* $\sigma_1$ *and* $\sigma_2$ *such that*

$$[\mathbf{UpdKey}, \mathbf{USMR}^1] \xrightarrow{\pi_{\mathsf{UE}}} \bigcap_{(P_1,P_2,\epsilon,\sigma)\in\Omega} (\sigma\, \mathbf{cUSMR}^1)^{[P_1,P_2]:\epsilon}$$

[6] The set of insulated regions.
[7] The set of all other regions, where confidentiality cannot be guaranteed.

*for,*

$$\Omega := \{(\mathcal{E}^{\mathtt{epoch}}_{\mathsf{fwl}}, \mathcal{E}^{\mathtt{epoch}}_{\mathsf{fwr}+1}, \epsilon_{\mathsf{CPA}}, \sigma_1) \mid (\mathsf{fwl}, \mathsf{fwr}) \in \mathcal{FW}\}^6$$
$$\cup \{(\mathcal{E}^{\mathtt{epoch}}_{\mathsf{fwr}+1}, \mathcal{E}^{\mathtt{epoch}}_{\mathsf{fwl}}, 0, \sigma_2) \mid (\mathsf{fwl}, \mathsf{fwr}) \in \mathcal{FW}\}^7$$

*where* $\epsilon_{\mathsf{CPA}}$ *denotes the reduction, described in theorem 1, from distinguishing the* $\mathbf{cUSMR}^1$ *(without key leakage) to the* IND-UE-CPA *game.*

In the above theorem, we can replace $(\mathbf{USMR}^1, \mathbf{cUSMR}^1, \epsilon_{\mathsf{CPA}}, \mathsf{IND\text{-}UE\text{-}CPA})$ with $(\mathbf{USMR}^+, \mathbf{cUSMR}^+, \epsilon_{\mathsf{CPA}^+}, \mathsf{ENC\text{-}CPA} + \mathsf{UPD\text{-}CPA})$ when we deal with unrestricted leakage, with $(\mathbf{iUSMR}^1, \mathbf{ciUSMR}^1, \epsilon_{\mathsf{CCA}}, \mathsf{IND\text{-}UE\text{-}RCCA})$ when we deal with injections and restricted leakage and with $(\mathbf{iUSMR}^+, \mathbf{ciUSMR}^+, \epsilon_{\mathsf{CCA}^+}, \mathsf{ENC\text{-}RCCA} + \mathsf{UPD\text{-}RCCA})$ when we deal with injections and unrestricted leakage. This includes all of our results.

*Proof.* We already proved the theorem inside the first type of intervals, the ones inside the insulated regions. In this case, the simulator $\sigma_1$ and the distinguishing advantage $\epsilon$ are given in our previous theorems and their proofs.

For the second type of intervals, the ones outside the insulated regions, consider the simulator $\sigma_2$ that encrypts the real messages and updates the real ciphertexts. Since there is no confidentiality, the simulator can do just that, its simulation is thus perfect and the correctness of the encryption scheme is enough to conclude. □

## 5 Conclusion

Our work was motivated by finding the most realistic security goal of UE. From this viewpoint, our contribution can be wrapped up saying that we give a composable version of UE, that permits to exhibit the right security notion sufficient for practical UE schemes, namely IND-UE-RCCA security.

From a CC viewpoint, we introduced an enhanced proof method, that notably allows to handle "asymmetric" challenges. We also proposed a UE protocol framework using interval-wise guarantees, that models the server as a semi-honest adversary.

## References


1. Badertscher, C., Maurer, U.: Composable and Robust Outsourced Storage, pp. 354–373. Lecture Notes in Computer Science, Springer International Publishing (2018), https://doi.org/10.1007/978-3-319-76953-0_19
2. Boneh, D., Lewi, K., Montgomery, H., Raghunathan, A.: Key Homomorphic PRFs and Their Applications, pp. 410–428. Advances in Cryptology - CRYPTO 2013, Springer Berlin Heidelberg (2013), https://doi.org/10.1007/978-3-642-40041-4_23
3. Boyd, C., Davies, G.T., Gjøsteen, K., Jiang, Y.: Fast and Secure Updatable Encryption, pp. 464–493. Advances in Cryptology - CRYPTO 2020, Springer International Publishing (2020), https://doi.org/10.1007/978-3-030-56784-2_16

4. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing Chosen-Ciphertext Security, pp. 565–582. Advances in Cryptology - CRYPTO 2003, Springer Berlin Heidelberg (2003), https://doi.org/10.1007/978-3-540-45146-4_33
5. Coretti, S., Maurer, U., Tackmann, B.: Constructing Confidential Channels from Authenticated Channels-Public-Key Encryption Revisited, pp. 134–153. Advances in Cryptology - ASIACRYPT 2013, Springer Berlin Heidelberg (2013), https://doi.org/10.1007/978-3-642-42033-7_8
6. Everspaugh, A., Paterson, K., Ristenpart, T., Scott, S.: Key Rotation for Authenticated Encryption, pp. 98–129. Advances in Cryptology - CRYPTO 2017, Springer International Publishing (2017), https://doi.org/10.1007/978-3-319-63697-9_4
7. Jiang, Y.: The Direction of Updatable Encryption Does Not Matter Much, pp. 529–558. Advances in Cryptology - ASIACRYPT 2020, Springer International Publishing (2020), https://doi.org/10.1007/978-3-030-64840-4_18
8. Jost, D., Maurer, U.: Overcoming Impossibility Results in Composable Security Using Interval-Wise Guarantees, pp. 33–62. Advances in Cryptology - CRYPTO 2020, Springer International Publishing (2020), https://doi.org/10.1007/978-3-030-56784-2_2
9. Jost, D., Maurer, U., Mularczyk, M.: A Unified and Composable Take on Ratcheting, pp. 180–210. Theory of Cryptography, Springer International Publishing (2019), https://doi.org/10.1007/978-3-030-36033-7_7
10. Klooß, M., Lehmann, A., Rupp, A.: (R)CCA Secure Updatable Encryption with Integrity Protection, pp. 68–99. Advances in Cryptology - EUROCRYPT 2019, Springer International Publishing (2019), https://doi.org/10.1007/978-3-030-17653-2_3
11. Lehmann, A., Tackmann, B.: Updatable Encryption with Post-Compromise Security, pp. 685–716. Advances in Cryptology - EUROCRYPT 2018, Springer International Publishing (2018), https://doi.org/10.1007/978-3-319-78372-7_22
12. Maurer, U.: Constructive Cryptography - A New Paradigm for Security Definitions and Proofs, pp. 33–56. Theory of Security and Applications, Springer Berlin Heidelberg (2012), https://doi.org/10.1007/978-3-642-27375-9_3
13. Maurer, U., Renner, R.: Abstract cryptography. In: In Innovations In Computer Science. Tsinghua University Press (2011)
14. Maurer, U., Renner, R.: From indifferentiability to constructive cryptography and back. In: Proceedings, Part I, of the 14th International Conference on Theory of Cryptography - Volume 9985. p. 3–24. Springer-Verlag, Berlin, Heidelberg (2016), https://doi.org/10.1007/978-3-662-53641-4_1

## Appendix A Security Games

Figure 8 is a description, taken from [3], of the oracles used in the security games studied throughout this work.

The oracles available for each security game, are shown in figure 9 :

## Appendix B IND-ENC+UPD-CPA security is sufficient for constructing cUSMR$^+$

*Proof.* Let $\mathbf{R} := \mathsf{enc}^{\mathsf{C}}\,\mathsf{upd}^{\mathsf{S}}\,\mathbf{USMR}$ be the the real system and $\mathbf{I} := \sigma^{\mathsf{S}}_{\mathsf{CPA}^+}\,\mathbf{cUSMR}$ be the ideal system. In order to determine the advantage of a distinguisher in

$\mathsf{Setup}(1^\lambda)$

$k_0 \leftarrow \mathsf{UE}.\mathsf{KG}(1^\lambda)$
$\Delta_0 \leftarrow \perp, e \leftarrow 0$
$\mathcal{L} \leftarrow [\,], \textsc{Chall} \leftarrow \texttt{false}$

$\mathcal{O}.\mathsf{Enc}(m)$

$c \leftarrow \mathsf{UE}.\mathsf{Enc}_{k_e}(m)$
$\mathcal{L} \leftarrow \mathcal{L} \mid\mid (c, e, m)$
**return** $c$

$\mathcal{O}.\mathsf{Dec}(c)$

$m$ or $\diamond \leftarrow \mathsf{UE}.\mathsf{Dec}_{k_e}(c)$
**if** Chall **and** $c = \tilde{c}_e$ **then**
  **return** `test`
**else**
  **return** $m$ or $\diamond$

$\mathcal{O}.\mathsf{Next}()$

$e \leftarrow e + 1$
$k_e \leftarrow \mathsf{UE}.\mathsf{KG}(1^\lambda)$
$\Delta_e \leftarrow \mathsf{UE}.\mathsf{TG}(k_{e-1}, k_e)$
**if** $\textsc{Chall}$ **then**
  $\tilde{c}_e \leftarrow \mathsf{UE}.\mathsf{Upd}_{\Delta_e}(\tilde{c}_{e-1})$

$\mathcal{O}.\mathsf{Upd}(c_{e-1})$

**if** $(c_{e-1}, e-1, m) \notin \mathcal{L}$ **then**
  **return** $\perp$
$c_e \leftarrow \mathsf{UE}.\mathsf{Upd}_{\Delta_e}(c_{e-1})$
$\mathcal{L} \leftarrow \mathcal{L} \mid\mid (c_e, e, m)$
**return** $c_e$

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{e})$

**if** $\hat{e} > e$ **then**
  **return** $\perp$
**if** $\mathsf{inp} = \mathsf{key}$ **then**
  **return** $k_{\hat{e}}$
**if** $\mathsf{inp} = \mathsf{token}$ **then**
  **return** $\Delta_{\hat{e}}$

$\mathcal{O}.\mathsf{Chall}(\bar{m}, \bar{c})$

**if** $\textsc{Chall}$ **then**
  **return** $\perp$
$\textsc{Chall} \leftarrow \texttt{true}$
**if** $(\bar{c}, e-1, \cdot) \notin \mathcal{L}$ **then**
  **return** $\perp$
**if** $b = 0$ **then**
  $\tilde{c}_e \leftarrow \mathsf{UE}.\mathsf{Enc}_{k_e}(\bar{m})$
**else**
  $\tilde{c}_e \leftarrow \mathsf{UE}.\mathsf{Upd}_{\Delta_e}(\bar{c})$
**return** $\tilde{c}_e$

$\mathcal{O}.\mathsf{UpdC}$

**if not** $\textsc{Chall}$ **then**
  **return** $\perp$
**return** $\tilde{c}_e$

**Fig. 8.** Description of the oracles used in UE security games.

| Notion | $\mathcal{O}.\mathsf{Enc}$ | $\mathcal{O}.\mathsf{Dec}$ | $\mathcal{O}.\mathsf{Next}$ | $\mathcal{O}.\mathsf{Upd}$ | $\mathcal{O}.\mathsf{Corr}$ | $\mathcal{O}.\mathsf{Chall}$ | $\mathcal{O}.\mathsf{UpdC}$ |
|---|---|---|---|---|---|---|---|
| IND-yy-CPA | ✓ | × | ✓ | ✓ | ✓ | ✓ | ✓ |
| IND-yy-CCA | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| IND-yy-RCCA | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

**Fig. 9.** Oracles available to the adversary in different security games, where $\mathsf{yy} \in \{\mathsf{ENC}, \mathsf{UPD}, \mathsf{UE}\}$. ✓ (resp. ×) means the adversary does (resp. does not) have access to the oracle.

distinguishing $\mathbf{R}$ from $\mathbf{I}$, denoted by $\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{I})$, we proceed with a sequence of systems. We introduce a hybrid system $\mathbf{S}$, then we determine the distinguishing advantages $\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{S})$ and $\Delta^{\mathbf{D}}(\mathbf{S}, \mathbf{I})$, the triangular inequality allows us to bound $\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{I})$ by the sum of those two advantages.

• Let $\mathbf{S}$ be a resource that behaves just like $\mathbf{R}$ when leaking updated ciphertexts and just like $\mathbf{I}$ when leaking fresh encryptions. Concretely, $\mathbf{S}$ maintains the same database as $\mathbf{R}$ using the $\mathsf{UE}$ scheme and, when $\mathbf{S}$ is asked to leak an updated ciphertext it returns it as is but when $\mathbf{S}$ is asked to leak a fresh ciphertext, it returns an encryption of a random$\bar{x}$.

Let $q$ be an upper bound on the number of write queries issued to the systems. We define a hybrid resource $\mathbf{H}_i$ that behaves just like $\mathbf{R}$ on the first $i$ write queries and like $\mathbf{S}$ afterwards. Then we define a reduction $\mathbf{C}_i$ that behaves like $\mathbf{H}_i$ except it uses the game $\mathbf{G}_b^{\mathsf{ENC\text{-}CPA}}$ oracles instead of doing the $\mathsf{UE}$ operations by itself and on the $i$-th write request (of the form $(\texttt{write}, j, x)$) it challenges the game with input $(x, \bar{x})$ to receive the ciphertext. We have

$$\mathbf{R} \equiv \mathbf{H}_q \text{ and } \mathbf{S} \equiv \mathbf{H}_0$$

and

$$\mathbf{H}_i \equiv \mathbf{G}_0^{\mathsf{ENC\text{-}CPA}}\mathbf{C}_i \equiv \mathbf{G}_1^{\mathsf{ENC\text{-}CPA}}\mathbf{C}_{i+1}$$

Indeed, this can be seen on the following timeline (3)

| $j$-th write query | $j < i$ | $j = i$ | $j = i+1$ | $j > i+1$ |
|---|---|---|---|---|
| $\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}}\mathbf{C}_i$ | $\mathsf{Enc}(x)$ | $\mathsf{Enc}(x)$ | $\mathsf{Enc}(\bar{x})$ | $\mathsf{Enc}(\bar{x})$ |
| $\mathbf{G}_1^{\mathsf{ENC\text{-}CPA}}\mathbf{C}_{i+1}$ | $\mathsf{Enc}(x)$ | $\mathsf{Enc}(x)$ | $\mathsf{Enc}(\bar{x})$ | $\mathsf{Enc}(\bar{x})$ |

**Table 3.** Leakage behavior of both systems for each $(\texttt{write}, \cdot, x)$ requests.

Let $\mathbf{C}_I$ be a reduction that samples $i \in [1, q]$ at random and behaves like $\mathbf{C}_i$ and define $\mathbf{D}' := \mathbf{D}\mathbf{C}_I$. We have,

$$\Pr[\mathbf{D}'(\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}}) = 1] = \frac{1}{q}\sum_{i=1}^{q} \Pr[\mathbf{D}(\mathbf{C}_i\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}}) = 1]$$

and

$$\begin{aligned}\Pr[\mathbf{D}'(\mathbf{G}_1^{\mathsf{ENC\text{-}CPA}}) = 1] &= \frac{1}{q}\sum_{i=1}^{q} \Pr[\mathbf{D}(\mathbf{C}_i\mathbf{G}_1^{\mathsf{ENC\text{-}CPA}}) = 1] \\ &= \frac{1}{q}\sum_{i=0}^{q-1} \Pr[\mathbf{D}(\mathbf{C}_i\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}}) = 1]\end{aligned}$$

Finally, the advantage of the distinguisher in distinguishing system $\mathbf{R}$ from $\mathbf{S}$ is

$$\begin{aligned}
\Delta^{\mathbf{D}}(\mathbf{R},\mathbf{S}) &= \Delta^{\mathbf{D}}(\mathbf{H}_q,\mathbf{H}_0)\\
&= \Delta^{\mathbf{D}}(\mathbf{C}_q\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}},\mathbf{C}_0\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}})\\
&= |\Pr[\mathbf{D}(\mathbf{C}_q\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}})=1]-\Pr[\mathbf{D}(\mathbf{C}_0\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}})=1]|\\
&= |\sum_{i=1}^{q}\Pr[\mathbf{D}(\mathbf{C}_i\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}})=1]-\sum_{i=0}^{q-1}\Pr[\mathbf{D}(\mathbf{C}_i\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}})=1]|\\
&= q\cdot|\Pr[\mathbf{D}'(\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}})=1]-\Pr[\mathbf{D}'(\mathbf{G}_1^{\mathsf{ENC\text{-}CPA}})=1]|\\
&= q\cdot\Delta^{\mathbf{D}'}(\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}},\mathbf{G}_1^{\mathsf{ENC\text{-}CPA}})
\end{aligned}$$

• Let us consider the systems $\mathbf{S}$ and $\mathbf{I}$. By definition, $\mathbf{S}$ behaves just like $\mathbf{I}$ when leaking fresh encryptions but, when asked to leak what should be an updated ciphertext, $\mathbf{S}$ returns this updated ciphertext while $\mathbf{I}$ simply returns an update of an encryption of a random $\bar{x}$.

Let $r$ be an upper bound on the number of update queries issued to the systems. We define a hybrid resource $\mathbf{H}'_i$ that behaves just like $\mathbf{S}$ on the first $i$ update queries and like $\mathbf{I}$ afterwards. Then we define a reduction $\mathbf{C}'_i$ that behaves like $\mathbf{H}'_i$ except it uses the game $\mathbf{G}_b^{\mathsf{UPD\text{-}CPA}}$ oracles instead of doing the $\mathsf{UE}$ operations by itself and on the $i$-th update computation it challenges the game with input $(c,\bar{c})$ to receive either an updated version of $c$ or $\bar{c}$, the encryption of the random $\bar{x}$. We have

$$\mathbf{S}\equiv\mathbf{H}'_r \text{ and } \mathbf{I}\equiv\mathbf{H}'_0$$

and

$$\mathbf{H}'_i\equiv\mathbf{G}_0^{\mathsf{UPD\text{-}CPA}}\mathbf{C}'_i\equiv\mathbf{G}_1^{\mathsf{UPD\text{-}CPA}}\mathbf{C}'_{i+1}$$

Indeed, this can be seen on the following timeline (4)

| $j$-th update query | $j<i$ | $j=i$ | $j=i+1$ | $j>i+1$ |
|---|---|---|---|---|
| $\mathbf{G}_0^{\mathsf{UPD\text{-}CPA}}\mathbf{C}'_i$ | $\mathsf{Upd}(c)$ | $\mathsf{Upd}(c)$ | $\mathsf{Upd}(\bar{c})$ | $\mathsf{Upd}(\bar{c})$ |
| $\mathbf{G}_1^{\mathsf{UPD\text{-}CPA}}\mathbf{C}'_{i+1}$ | $\mathsf{Upd}(c)$ | $\mathsf{Upd}(c)$ | $\mathsf{Upd}(\bar{c})$ | $\mathsf{Upd}(\bar{c})$ |

**Table 4.** Leakage behavior of both systems for each update requests.

Let $\mathbf{C}'_I$ be a reduction that samples $i\in[1,r]$ at random and behaves like $\mathbf{C}'_i$ and define $\mathbf{D}'':=\mathbf{D}\mathbf{C}'_I$. We have,

$$\Pr[\mathbf{D}''(\mathbf{G}_0^{\mathsf{UPD\text{-}CPA}})=1]=\frac{1}{r}\sum_{i=1}^{r}\Pr[\mathbf{D}(\mathbf{C}'_i\mathbf{G}_0^{\mathsf{UPD\text{-}CPA}})=1]$$

and

$$\Pr[\mathbf{D}''(\mathbf{G}_1^{\mathsf{UPD\text{-}CPA}}) = 1] = \frac{1}{r}\sum_{i=1}^{r}\Pr[\mathbf{D}(\mathbf{C}'_i\mathbf{G}_1^{\mathsf{UPD\text{-}CPA}}) = 1]$$
$$= \frac{1}{r}\sum_{i=0}^{r-1}\Pr[\mathbf{D}(\mathbf{C}'_i\mathbf{G}_0^{\mathsf{UPD\text{-}CPA}}) = 1]$$

Finally, the advantage of the distinguisher in distinguishing system $\mathbf{S}$ from $\mathbf{I}$ is

$$\begin{aligned}
\Delta^{\mathbf{D}}(\mathbf{S},\mathbf{I}) &= \Delta^{\mathbf{D}}(\mathbf{H}'_r,\mathbf{H}'_0)\\
&= \Delta^{\mathbf{D}}(\mathbf{C}'_r\mathbf{G}_0^{\mathsf{UPD\text{-}CPA}},\mathbf{C}'_0\mathbf{G}_0^{\mathsf{UPD\text{-}CPA}})\\
&= |\Pr[\mathbf{D}(\mathbf{C}'_r\mathbf{G}_0^{\mathsf{UPD\text{-}CPA}}) = 1] - \Pr[\mathbf{D}(\mathbf{C}'_0\mathbf{G}_0^{\mathsf{UPD\text{-}CPA}}) = 1]|\\
&= |\sum_{i=1}^{r}\Pr[\mathbf{D}(\mathbf{C}'_i\mathbf{G}_0^{\mathsf{UPD\text{-}CPA}}) = 1] - \sum_{i=0}^{r-1}\Pr[\mathbf{D}(\mathbf{C}'_i\mathbf{G}_0^{\mathsf{UPD\text{-}CPA}}) = 1]|\\
&= r\cdot|\Pr[\mathbf{D}''(\mathbf{G}_0^{\mathsf{UPD\text{-}CPA}}) = 1] - \Pr[\mathbf{D}''(\mathbf{G}_1^{\mathsf{UPD\text{-}CPA}}) = 1]|\\
&= r\cdot\Delta^{\mathbf{D}''}(\mathbf{G}_0^{\mathsf{UPD\text{-}CPA}},\mathbf{G}_1^{\mathsf{UPD\text{-}CPA}})
\end{aligned}$$

• We use the triangular inequality to conclude. Let $q$ be an upper bound on the number of writes and $r$ be an upper bound on the number of updates. The advantage of the distinguisher in distinguishing the real system $\mathbf{R}$ from the ideal one $\mathbf{I}$ is

$$\begin{aligned}
\Delta^{\mathbf{D}}(\mathbf{R},\mathbf{I}) &\leq \Delta^{\mathbf{D}}(\mathbf{R},\mathbf{S}) + \Delta^{\mathbf{D}}(\mathbf{S},\mathbf{I})\\
&= q\cdot\Delta^{\mathbf{D}'}(\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}},\mathbf{G}_1^{\mathsf{ENC\text{-}CPA}}) + r\cdot\Delta^{\mathbf{D}''}(\mathbf{G}_0^{\mathsf{UPD\text{-}CPA}},\mathbf{G}_1^{\mathsf{UPD\text{-}CPA}})
\end{aligned}$$

So the ENC-CPA and UPD-CPA notions are sufficient to securely construct the **cUSMR** in the unbounded leakage model, where the age of each database entry is not hidden. □

## Appendix C IND-ENC+UPD-CPA security is necessary for constructing cUSMR$^+$

In order to show that the ENC-CPA and UPD-CPA security notions are both necessary to securely construct the **cUSMR**$^+$ from a **USMR**$^+$ using a UE scheme, we are going to use a technique introduced by Maurer *et al.* in [5]. Keeping our notations, we will start from the real system $\mathbf{R}$ and the ideal one $\mathbf{I}$, where this time we use an arbitrary simulator $\sigma$, and use reductions to construct the resources $\mathbf{G}_b^{\mathsf{ENC\text{-}CPA}}$ (respectively $\mathbf{G}_b^{\mathsf{UPD\text{-}CPA}}$) implementing the ENC-CPA games

(respectively the UPD-CPA games). Formally, we will describe two reductions $\mathbf{C}_0$ and $\mathbf{C}_1$ such that

$$\mathbf{C}_0\mathbf{R} \equiv \mathbf{G}_0^{\mathsf{ENC\text{-}CPA}},\ \mathbf{C}_1\mathbf{R} \equiv \mathbf{G}_1^{\mathsf{ENC\text{-}CPA}} \text{ and } \mathbf{C}_0\mathbf{I} \equiv \mathbf{C}_1\mathbf{I}$$

Let $\mathbf{A}$ be an adversary against the ENC-CPA security notion. Using the triangular inequality, we have

$$\begin{aligned}\Delta^{\mathbf{A}}(\mathbf{G}_0^{\mathsf{ENC\text{-}CPA}}, \mathbf{G}_1^{\mathsf{ENC\text{-}CPA}}) &= \Delta^{\mathbf{A}}(\mathbf{C}_0\mathbf{R}, \mathbf{C}_1\mathbf{R}) \\ &\leq \Delta^{\mathbf{A}}(\mathbf{C}_0\mathbf{R}, \mathbf{C}_0\mathbf{I}) + \underbrace{\Delta^{\mathbf{A}}(\mathbf{C}_0\mathbf{I}, \mathbf{C}_1\mathbf{I})}_{=0} + \Delta^{\mathbf{A}}(\mathbf{C}_1\mathbf{I}, \mathbf{C}_1\mathbf{R}) \\ &= \Delta^{\mathbf{A}\mathbf{C}_0}(\mathbf{R}, \mathbf{I}) + \Delta^{\mathbf{A}\mathbf{C}_1}(\mathbf{R}, \mathbf{I})\end{aligned}$$

Thus, if an UE scheme securely construct (through a protocol) $\mathbf{I}$ from $\mathbf{R}$, then for any distinguisher $\mathbf{D}$, the distinguishing advantage $\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{I})$ is "small" and using the above inequality, the advantage of any adversary $\mathbf{A}$ in distinguishing the ENC-CPA games will also be "small". We conclude that, under the above hypothesis, the UE scheme will be ENC-CPA secure. We do the same thing with different reductions for the UPD-CPA notion to prove our claim.

• We start with the ENC-CPA notion. We are going to describe two above reductions $\mathbf{C}_0$ and $\mathbf{C}_1$. Let $n$ be an upper bound on the number of encryption queries issued to the games by the adversary. In the following, we consider resources of memory size $n+1$. This is done to have enough space to store the returned ciphertext (and the challenge ciphertext) in case the challenger ask for an update later on. Here is how the reductions implement the game oracles :

- On the $i$-th $\mathcal{O}.\mathsf{Enc}(x)$ query : $\mathbf{C}_b$ sends $(\texttt{write}, i, x)$ at interface C, then sends $(\texttt{leak}, i)$ at interface S and outputs the result.
- On $\mathcal{O}.\mathsf{Next}()$ : $\mathbf{C}_b$ sends `askUpdate` at interface C and `update` at interface S.
- On $\mathcal{O}.\mathsf{Upd}(c)$ : $\mathbf{C}_b$ checks if $c$ is the challenge or an updated version of it and if $c$ was present in memory in the previous epoch. If not, it returns $\perp$. If it is, let $i$ be the index where $c$ was written, $\mathbf{C}_b$ sends $(\texttt{leak}, i)$ at interface S and outputs the result.
- On $\mathcal{O}.\mathsf{Corr}(\mathsf{elt}, \hat{e})$ where $\mathsf{elt} \in \{\mathsf{Key}, \mathsf{Token}\}$ : $\mathbf{C}_b$ triggers the event $\mathcal{E}^{\texttt{leaked}}_{\mathsf{elt},\hat{e}}$, sends $(\texttt{fetchelt}, \hat{e})$ at interface S and outputs the result.
- On $\mathcal{O}.\mathsf{Chall}(m_0, m_1)$ : If $m_0 \neq m_1$ and $|m_0| = |m_1|$, the reduction $\mathbf{C}_b$ sends $(\texttt{write}, n+1, m_b)$ at interface C then it sends $(\texttt{leak}, n+1)$ at interface S and outputs the result.
- On $\mathcal{O}.\mathsf{UpdC}()$ : If a challenge has already been issued, $\mathbf{C}_b$ sends $(\texttt{leak}, n+1)$ at interface S and returns the result. If not, it returns $\perp$.

Moreover, all the oracles variables and lists can be emulated faithfully by the reductions via internal calculations and memory. When connected to the real system $\mathbf{R}$, the reduction $\mathbf{C}_b$ exactly implements the game $\mathbf{G}_b^{\mathsf{ENC\text{-}CPA}}$ since

the system leaks the same encryptions and updates as the ones produce by the game. Their observable behaviors being the same, we have $\mathbf{C}_b\mathbf{R} \equiv \mathbf{G}_b^{\mathsf{ENC\text{-}CPA}}$.

Now, if we connect the reductions to the ideal system $\mathbf{I}$, where an arbitrary simulator $\sigma$ is used, their behaviors can only differ on the challenge call. When asked for the challenge, reduction $\mathbf{C}_b$ writes $m_b$ at position $n+1$ then it outputs the result of $(\texttt{leak}, n+1)$. In the ideal system $\mathbf{I}$, the result of a `leak` request on a freshly written entry (which is the case here since we don't change epoch yet) is always the length of the entry. Since the reductions checked that $|m_0| = |m_1|$, the inputs of the simulator $\sigma$ are the same and the behaviors are thus identical. This remains true in the following epochs, when the adversary calls the oracle $\mathcal{O}.\,\mathsf{UpdC}$ to get an updated version of the challenge ciphertext, since the challenge's plaintext is never rewritten. We conclude that the systems $\mathbf{C}_b\mathbf{I}$ are indistinguishable from one another. Written differently, we have $\mathbf{C}_0\mathbf{I} \equiv \mathbf{C}_1\mathbf{I}$ and the theorem is proven in the ENC-CPA case.

• Since all the oracles, except the $\mathcal{O}.\,\mathsf{Chall}$ one, of the UPD-CPA games are the same as the ones present in the ENC-CPA we only need to redefine the reductions for this oracle. Let $\mathbf{C}'_b$ be a reduction that behaves exactly like $\mathbf{C}_b$ except :

– On $\mathcal{O}.\,\mathsf{Chall}(c_0, c_1)$ : If $c_0 \neq c_1$ and $|c_0| = |c_1|$, the reduction $\mathbf{C}'_b$ checks that both $c_0$ and $c_1$ were present in the previous epoch. If they were, let $i_0$ and $i_1$ be the respective positions where these ciphertexts were written, $\mathbf{C}'_b$ sends $(\texttt{leak}, i_b)$ at interface $\mathsf{S}$, returns the result and remembers $i_b$ by writing it in its own internal memory. If not, it returns $\perp$.

Since we don't need space to write a new ciphertext when the challenge is issued in these games, we can use a $\mathbf{USMR}^+$ and $\mathbf{cUSMR}^+$ resources of size $n$ instead of $n+1$ like before (remember that $n$ is an upper bound on the number of calls to the encryption oracle). Since the challenge is written at position $i_b$, we need to slightly modify the behavior of $\mathbf{C}'_b$ when emulating the oracle $\mathcal{O}.\,\mathsf{UpdC}$ like so :

– On $\mathcal{O}.\,\mathsf{UpdC}()$ : If a challenge has already been issued, $\mathbf{C}_b$ sends $(\texttt{leak}, i_b)$ at interface $\mathsf{S}$ and returns the result. If not, it returns $\perp$.

The arguments to show that $\mathbf{C}'_b\mathbf{R} \equiv \mathbf{G}_b^{\mathsf{UPD\text{-}CPA}}$ and $\mathbf{C}'_0\mathbf{I} \equiv \mathbf{C}'_1\mathbf{I}$ remain the same as the ones used in the ENC-CPA case presented above. The conclusion is also the same, mainly that

$$\begin{aligned}
\Delta^{\mathbf{A}}(\mathbf{G}_0^{\mathsf{UPD\text{-}CPA}}, \mathbf{G}_1^{\mathsf{UPD\text{-}CPA}}) &= \Delta^{\mathbf{A}}(\mathbf{C}'_0\mathbf{R}, \mathbf{C}'_1\mathbf{R}) \\
&\leq \Delta^{\mathbf{A}}(\mathbf{C}'_0\mathbf{R}, \mathbf{C}'_0\mathbf{I}) + \underbrace{\Delta^{\mathbf{A}}(\mathbf{C}'_0\mathbf{I}, \mathbf{C}'_1\mathbf{I})}_{=0} + \Delta^{\mathbf{A}}(\mathbf{C}'_1\mathbf{I}, \mathbf{C}'_1\mathbf{R}) \\
&= \Delta^{\mathbf{A}\mathbf{C}'_0}(\mathbf{R}, \mathbf{I}) + \Delta^{\mathbf{A}\mathbf{C}'_1}(\mathbf{R}, \mathbf{I})
\end{aligned}$$

And the UPD-CPA security notion is also necessary to herd this secure construction.

## Appendix D The CCA security notions are unnecessarily strong for constructing ciUSMR

We show that the CCA security notion is in fact too strong for constructing the **ciUSMR** from the **iUSMR** using a UE scheme. To see this, suppose that we securely construct a **ciUSMR** from an **iUSMR** with an IND-UE-CCA secure UE scheme $\Pi$. We construct a UE scheme $\Pi'$ that works just like $\Pi$ except it appends a 0 bit at the end of every ciphertext, this bit being ignored when updating and decrypting ciphertexts. The scheme $\Pi'$ is no longer IND-UE-CCA secure since the adversary can switch the last bit of the challenge to 1 and send it to the decryption oracle to recover the underlying plaintext. However, the scheme $\Pi'$ can still be used for the secure construction described above. Indeed, we can use a simulator which does the following :

On $(\texttt{inject}, i, c)$ the simulator replaces the last bit of $c$ with a 0 and injects it at position $i$. If the last bit was a 1, the simulator can remember it in case it is asked to leak the ciphertext.

With this simulator, the observable behaviors of both the real and the ideal systems are the same as if the scheme $\Pi$ was used. The construction is thus secure even though the scheme $\Pi'$ was not IND-UE-CCA secure. This proves that IND-UE-CCA security is too strong for this construction.