Secure Vault scheme in the Cloud Operating Model

Rishiraj Bhattacharyya[1], Avradip Mandal[2], and Meghna Sengupta[3]

[1] University of Birmingham, UK
rishiraj.bhattacharyya@gmail.com
[2] Zfenselabs, USA
avradip@zfenselabs.com
[3] University of Edinburgh, UK
meghna.816@gmail.com

Abstract. The rising demand for data privacy in cloud-based environments has led to the development of advanced mechanisms for securely managing sensitive information. A prominent solution in this domain is the "Data Privacy Vault," a concept that is being provided commercially by companies such as Hashicorp [17], Basis Theory [5], Skyflow Inc. [20], VGS [21], Evervault [15], Protegrity [19], Anonomatic [1], and BoxyHQ [9]. However, no existing work has rigorously defined the security notions required for a Data Privacy Vault or proven them within a formal framework which is the focus of this paper.

Among its other uses, data privacy vaults are increasingly being used as storage for LLM training data which necessitates a scheme that enables users to securely store sensitive information in the cloud while allowing controlled access for performing analytics on specific non-sensitive attributes without exposing sensitive data. Conventional solutions involve users generating encryption keys to safeguard their data, but these solutions are not deterministic and are therefore unsuited for the LLM setting. To address this, we propose a novel framework that is deterministic as well as semantically secure. Our scheme operates in the Cloud Operating model where the server is trusted but stateless, and the storage is outsourced.

We provide a formal definition and a concrete instantiation of this data privacy vault scheme. We introduce a novel tokenization algorithm that serves as the core mechanism for protecting sensitive data within the vault. Our approach not only generates secure, unpredictable tokens for sensitive data but also securely stores sensitive data while enabling controlled data retrieval based on predefined access levels.

Our work fills a significant gap in the existing literature by providing a formalized framework for the data privacy vault, complete with security proofs and a practical construction - not only enhancing the understanding of vault schemes but also offering a viable solution for secure data management in the era of cloud computing.

Keywords: data privacy vault, LLM privacy, cloud storage, secure data storage, tokenization, vault scheme

1 Introduction

In the wake of growing privacy concerns, particularly highlighted by incidents such as the Samsung internal data leak [10], the need for robust safeguards when storing sensitive as well as non-sensitive information pertaining to both clients and businesses, has become more urgent than ever. One promising approach to addressing these concerns is the concept of a "data privacy vault," a tool designed to securely manage sensitive information. Corporations like Hashicorp [17], Basis Theory [5], Skyflow Inc. [20], VGS [21], Evervault [15] etc. have pioneered this approach with their Data Privacy Vault tools, which provide a way to safeguard data while enabling selective access for authorized users.

However, while these commercial solutions offer a strong application oriented approach, there is a noticeable gap in the academic literature when it comes to providing a concrete instantiation of such a vault scheme, along with a formal security analysis. To the best of our knowledge, no existing work has rigorously defined the security notions required for a data privacy vault or proven them within a formal framework.

In this paper, we aim to fill this gap by presenting a concrete instantiation of the data privacy vault scheme. Our approach focuses on securing sensitive information through access-controlled tokenization, ensuring that data stored on the cloud remains confidential even in the event of unauthorized access. We further define meaningful security notions for this scheme, providing a clear framework for understanding its strength and resilience. Finally, we offer formal proofs for these security notions, demonstrating the robustness of our proposed vault scheme.

Additionally, increasingly in many scenarios, the data stored in these data privacy vaults are used for training Large Language Models (LLMs) and therefore privacy in the world of LLMs have a strong connection with securing credentials in the cloud based platforms. At a very high level, we wish to design a scheme which allows a user to securely store their sensitive and non-sensitive data on the cloud. Certain attributes of the non-sensitive data could be collected by a third party for analytics or training. The sensitive data, however, must remain secret. Additionally, the user must also be able to retrieve the data (sensitive as well as non-sensitive attributes of it) when they wish to do so.

Our work not only advances the theoretical understanding of data privacy vaults but also offers practical insights for their implementation in cloud-based environments, where the need for secure, scalable, and cost-effective data management solutions is paramount.

A trivial solution which has been used in practice, for purposes related to this problem [22], is to have the user generate a secret encryption key and then use this key to encrypt all of their sensitive data and store it in the cloud [24,23]. However, standard semantically secure encryption algorithm/functional encryption algorithms will not be deterministic, and hence, will lead to different "token/variable" for the same sensitive data. This leads to incorrect learning output/tracking. To mitigate this risk, we aim to come up with a framework that uses deterministic tokenisation and yet maintains semantic security. All we re-

quire is a method of authentication - some form of identity that does not require storage, for instance, biometric data. This approach enhances the security of sensitive data by separating the authentication and data tokenisation layers.

In addition to this, we want the storage server to also be resilient against key theft. Therefore, we do not want the server to encrypt all the data on the cloud with a single secret key. This would be highly risky since a security breach for the key would then render the entire cloud insecure. Our design actually provides a stronger security guarantee which does not let an adversary who has obtained unauthorised access to the cloud database learn the sensitive data.

1.1 Our Contributions

The framework that we build has three types of parties - the user, the server and the analyst with the cloud as a separate storage unit. The user has some sensitive data and some corresponding non-sensitive data that they wish to store on the cloud. The server manages the access to the cloud as well as the communication with the user to obtain the data that gets stored on the cloud. Due to our requirement that the user should not need to store any key, the server in our scheme needs to be trusted. Once the data is stored, the user can grant the analyst permission to access the non-sensitive data in the database without learning the sensitive data. If the analyst can convince the user, the user can also grant the analyst permission to view all of the data.

The Vault Scheme The proposed vault scheme is described using four algorithms (Param, Store, Access, Retrieve). The parameter generation algorithm Param generates the public (and possibly secret) parameters PP. The *stateful* Store algorithm takes the parameters PP, a user id denoted by id, and a message $M = (M_s, M_{ns})$ along with the state DB as input and produces an access password $psswd$, a data token tok and an updated state DB'. The state may contain a tag such that the application can return the same token and password when a message M is queried again. The Access algorithm takes as input the data token t and returns M_{ns}. The algorithm Retrieve retrieves the message M from the token t and the state DB.

The password acts as the master access token, which allows to retrieve the entire record (as well as generate the analyst token) Analyst to retrieve non-sensitive data submits data token and access token. The owner has additionally a master token, using which they can generate the access token.

We study the privacy of our scheme in the *cloud operating model* where the protocol has limited private storage and the database is stored in a outsourced hybrid database. Instances of this problem has recently been addressed via some commercial entities like Hashicorp [17], Basis Theory [5], Skyflow Inc. [20], VGS [21], Evervault [15], Protegrity [19], Anonomatic [1], BoxyHQ [9], Delinea's Delegated Machine Credentials [11], Entrust [14], AWS serverless services [4] etc. However, any theoretical treatment of the problem is absent from the literature. These tools are widely used by both industrial and individual clients for a wide range of applications. We discuss one such application to provide a concrete motivation for our work.

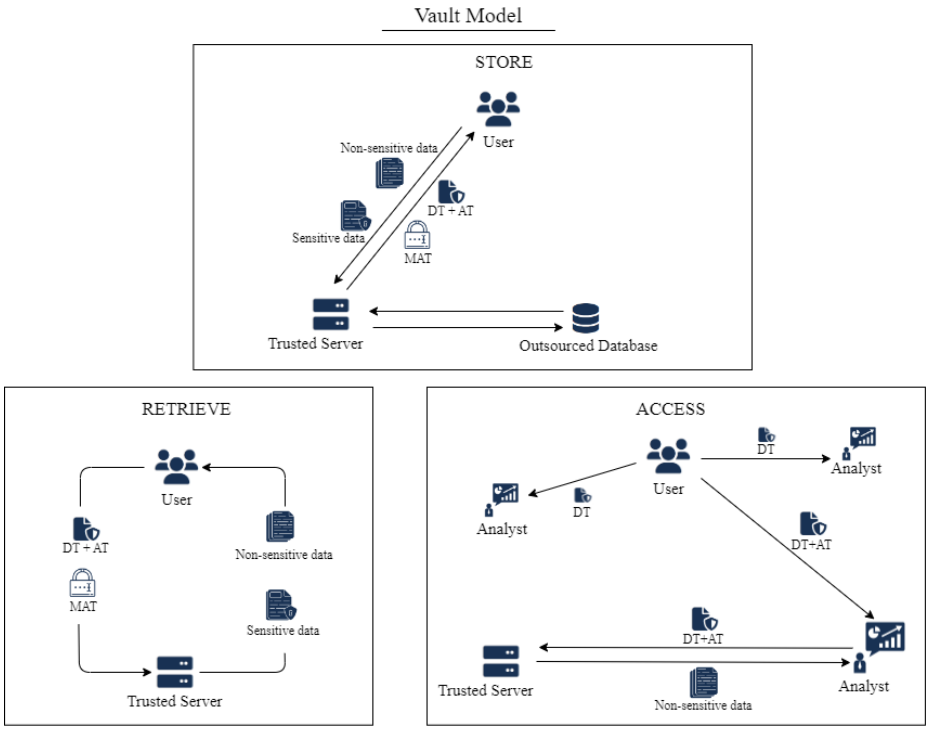

Fig. 1: Functionality of the Vault Scheme
MAT: Master Access Token (Password), DT: Data Token, AT: Access Token

Use-Case. As an example, we consider the following use-case of a government holding the travel data of its citizens which also includes their passport numbers. The government in this case is the user in our framework. The passport data is evidently sensitive data that needs to be stored securely. The government also wants to employ certain contractors or data analysts who need to analyse this travel data but do not really care about the exact passport number. But they do care about travel patterns, e.g. whether any citizen has travelled to a certain place a large number of times in a short period of time. In case somebody's travel pattern looks suspicious, then vetted analysts with high clearance should be able to access the exact passport number corresponding to those persons.

To this end, we design a scheme that lets the Government have a way to grant two levels of access to the data. This is achieved by generating a data token that utilizes tokenization and an access password. Both of these are given to the government (user) when they first store the data on the cloud. Each passport number gets one data token and one access password. The data analysts hold the data tokens which can be used to access the corresponding non-sensitive travel data. The access password is only provided if the travel data is proven to be confidential - in other words, if the non-sensitive data meets some condition determined by the user. The access password, as the name suggests, can be used to learn the sensitive data. If stolen, both the data token and the access password

can be reissued; however, lost authentication passwords which have been revoked can not be used to access data.

In the following subsection, we discuss the concept of tokenization and how it can be used in our problem setting.

TOKENIZATION. Tokenization is an extremely useful method that protects sensitive data by mapping it to non-sensitive random strings of characters [12]. These random strings are called *tokens*. Instead of having the applications store users' sensitive data on their devices, tokenization helps by having them store the corresponding token. The transactions are carried out with the tokens as stand-ins for the actual sensitive data. For this to be secure, an important requirement of the tokenization scheme is that the token generated for any sensitive message must be completely unrelated to the message and *unpredictable*. In many of the applications, the tokens are required to be reversible. A detokenization algorithm retrieves the referenced sensitive data from the token. Tokenization has found extensive usage especially in storing and protecting credit card information, telephone numbers, social security numbers etc.

The standard and widespread technique of tokenization is to use a dedicated database, called token vault [4]. The database contains two entries: the (encrypted) sensitive data and the token. As the database is private, no adversary could match the token with the stored data. Moreover, a *secret* key could be used to encrypt the sensitive data.

Aragona et al. in [3] provide a Reversible Hybrid type tokenization algorithm where they also provide a security proof that complies with the PCI Data Security Standard (PCIDSS) [18]. However, their tokenization algorithm is very specifically meant to be used in the setting of generating tokens for private credit card numbers and not for any other scenarios or use-cases. As such, this algorithm and the PCI-DSS recommendations of security notions for tokenization schemes are not suited to our needs for the Vault scheme.

In this work, we provide a new algorithm for tokenization and further use it in the vault scheme to generate both the data tokens and the access password (discussed in the use-case). Additionally, we also define notions of security which provide a better framework to understand the strength and properties of tokenization schemes.

CLOUD OPERATING MODEL. Traditional data centers usually are known to have dedicated infrastructures. The applications have access to one or more local databases. Security protocols supporting the applications are designed and analyzed in such a local storage setting; servers could store keys, states, or databases that are *inaccessible* by the adversaries.

The recent popularity of the cloud operating model [16] calls for a transition from the dedicated private storage servers to a pool of outsourced storage solutions. Having a stateless server is highly desirable in many use-cases as it provides significant advantages like cost effectiveness and scalability. In particular, the transport layer security might be handled by a separate web service and the tokenization layer might not have any access to any server certificates.

A stateless server implies any database it uses must be outsourced. There is always a possibility, that the remote storage server could mount an insider attack. The idea of encrypting the database does not work either as being stateless, the server does not have access to any persistent secret key. Thus the simple tasks of storing a relational database requires a new privacy treatment.

We build foundations of secure tokenization in this delegated public storage setting. Starting with the definitions of privacy of tokenization schemes we create a clear target for the schemes to achieve. We follow up with new designs of both practical and theoretical interest. Our practical construction is analyzed in the Random Oracle Model (ROM). Finally, we show a standard model construction as well.

In summary, our paper contributes the following -

- We introduce and describe a formal framework which represents a Data Privacy Vault. This Vault scheme features a stateless server in a cloud operating model with an outsourced storage and reflects the features of industrial Vault tools [20,11,4,17] by providing multiple access levels to its users.
- We formally define meaningful security notions for our defined Data Privacy Vault. This represents a significant step toward the standardization and rigorous analysis of the security guarantees offered by both current and future Vault tools.
- In order to give our construction for the Vault scheme we also define the concept of Tokenization which has not been formally defined as yet in literature. We give definitions for the two security notions that we use for tokenization - one for the private database setting and the other for the cloud operating model where the database does not have any security.
- Finally, we provide a concrete construction for the Vault Scheme that we have designed using our Tokenization scheme and further provide its proof of security in the Random Oracle model. Additionally, keeping practicality in mind, we ensure that our solution is compatible with the use of the Vault for storing training secure data for LLM models.

1.2 Previous Works

ANSI X9.119-2 [2] outlines three different approaches for tokenization: (1) On Demand Random Assignment (ODRA) which uses random tokens which are generated on demand. The association with the plaintext value is stored in a dynamic mapping table. When tokens are generated for new plaintexts, entries get added to the table. (2) Static table-driven tokenization which generates tokens using a tokenization mapping process which operates using small pre-generated static random substitution tables used as the tokenization secret. This is also called vault-less tokenization. (3) Encryption-based tokenization which generates its tokens using a suitable Format Preserving Encryption scheme or a symmetric encryption scheme. Here, the encryption key itself is the tokenization secret key.

Durak et al. in [13] discuss a format-preserving encryption scheme that can be used as the base for either static table-driven tokenization or for encryption-

based tokenization. However, they do not provide a security analysis of the tokenization scheme that is being built on top of the FPE scheme. Indeed, security notions for tokenization schemes have not been formally addressed previously in literature. While we do address this in our paper by formalizing a security notion for tokenization schemes in the classic setting, but, our main contribution is introducing and giving a formal security notion for the problem of having the database on a non-secure cloud.

To the best of our knowledge, there has not been any previous work on the security of tokenization schemes in the zero-trust cloud operating model - which is widely used in various applications. Our work introduces notions of security in tokenization schemes in both the traditional private database setting as well as privacy against a honest-but-curious storage server.

1.3 Organization of the paper

- Section 2 provides a brief description of the notations used throughout the paper along with the definition of the primitives that are utilized in our definitions and proofs.
- In Section 3, we define a Vault Scheme along with the introduction and definition of two meaningful security notions - (a) for the privacy of non-sensitive information stored in the Vault and (b) for the privacy of the sensitive information in the Vault.
- In Section 4, we introduce a building block that we use for our construction of a secure Vault Scheme which is Tokenization. We formally define a secure Tokenization Scheme which has been conspicuously missing in literature. We also introduce two security notions that we use for tokenization - one for the private database setting and the other for the cloud operating model where the database does not have any security.
- Section 5 gives the construction that we have designed and its proof of security in the Random Oracle model.
- Section 6 concludes the paper by providing a summary.

2 Notations and Preliminaries

NOTATIONS. If x is a string, $|x|$ denotes the length of the string. $x[i]$ denotes its i-th bit. If S is a set, $|S|$ denotes the size of S, and $s \xleftarrow{\$} S$ denotes sampling an element uniformly at random from S and assign it to s. $\mathbb{N}$ denotes the set of positive integers $\{1, 2, \ldots\}$. For $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, 2, \cdots, n\}$. Composition of two functions is denoted by $\circ$. If $\hat{F} = F \circ \phi$, then $\hat{F}(x) = F(\phi(x))$. $\{0,1\}^n$ denotes the set of all binary strings of length n. The set of all functions with domain $\{0,1\}^m$ and co-domain $\{0,1\}^n$ is denoted by $\mathcal{F}_{m,n}$.

The guessing probability of a random variable X is defined as $GP(X) = \max_x \Pr[X = x]$. The min-entropy of a random variable X is defined by $H_\infty(X) \stackrel{def}{=} -\log \max_x \Pr[X = x]$. The conditional min-entropy of X conditioned on

another random variable Y is defined by $H_\infty(X \mid Y) \stackrel{def}{=} -\log \sum_y \Pr[Y = y] \max_x \Pr[X = x \mid Y = y]$.

Definition 1 (Extractor). *Let $n(\lambda), \ell(\lambda), m(\lambda)$ be polynomial in λ. A family of functions $\mathsf{Ext} = \{\mathsf{Ext}_\lambda \colon \{0,1\}^{s(\lambda)} \times \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{\kappa(\lambda)}\}$ is called (k, ε)-extractor if for any random variable X following a distribution with support $\{0,1\}^{\ell(\lambda)}$ and min entropy k, it holds that*

$$\mathcal{SD}\left(\left(U_{s(\lambda)}, \mathsf{Ext}_\lambda\left(U_{s(\lambda)}, X\right)\right) \ ; \ \left(U_{s(\lambda)}, U_{\kappa(\lambda)}\right)\right) \leq \varepsilon.$$

Algorithms The algorithms considered in this paper are randomized, unless otherwise specified. If $\mathcal{A}$ is an algorithm, we let $y \leftarrow \mathcal{A}(x_1, \ldots; r)$ to denote that running $\mathcal{A}$ with input $x_1, \ldots$, random coin r and assigning the output to y. We let $y \stackrel{\$}{\leftarrow} \mathcal{A}(x_1, \ldots)$ be the result of choosing r uniformly at random and letting $y \leftarrow \mathcal{A}(x_1, \ldots; r)$.

Security Games The results are proven in the framework of code based games of [8]. A game G consists of a `main` oracle and zero or more stateful oracles $O_1, O_2, \cdots, O_n$. If a game G is implemented using a function f, we write $G[f]$ to denote the game. An algorithm $\mathcal{A}$ is said to participate in game G if the `main` oracle invokes algorithm $\mathcal{A}$ who can (optionally) make queries to the oracles $O_1, O_2, \cdots, O_n$. We denote by $G^{\mathcal{A}} = 1$ that an execution of G with $\mathcal{A}$ outputs 1.

Success Probability. The success probability of an algorithm $\mathcal{A}$ in game G is defined by $\mathbf{Succ}_{\mathcal{A},G} \stackrel{def}{=} \Pr[G^{\mathcal{A}} = 1]$.

In all the descriptions, uninitialized integers are assumed to be set to 0, booleans are set to *false*, the strings, sets and lists are set to be empty.

Random Oracles. An (idealized) function $\mathcal{H} : \{0,1\}^\delta \to \{0,1\}^\rho$ is said to be a *Random Oracle*, if for all $x \in \{0,1\}^\delta$, the output $\mathcal{H}(x)$ is independently and uniformly distributed over $\{0,1\}^\rho$.

Semantic Security of Symmetric Encryption: Real-or-Random game The game proceeds as follows - a random bit b is first chosen, and then the adversary $\mathcal{A}$ is invoked. The adversary $\mathcal{A}$ is allowed to make multiple queries to the ROR challenger ROR_C, where each query is a plaintext $m \in \{0,1\}$. When $b = 0$, the challenger first chooses a random key $k \leftarrow \mathcal{K}$ and returns $c \leftarrow \mathsf{Enc}(k, m)$. Note that the challenger chooses a fresh key for each encryption query. If $b = 1$, then the challenger will just return a random bit string of the ciphertext length of SE. The adversary guesses a b' and wins the game if $b' = b$. Advantage is defined as

$$\mathsf{Adv}^{\mathrm{ror}}_{SE,\mathcal{A}}(\lambda) = |\Pr[\mathrm{ROR1}_{\mathcal{A}} \Rightarrow 1] - \Pr[\mathrm{ROR0}_{\mathcal{A}} \Rightarrow 1]|$$

and we say that SE is ROR-secure if $\mathsf{Adv}^{\mathrm{ror}}_{SE,\mathcal{A}}(\lambda)$ is a negligible function for any adversary $\mathcal{A}$.

3 Vault Scheme and its Security

We begin by defining a basic Vault Scheme that can be used in the use-cases that we have discussed so far. Subsequently, we provide a formal notion for the security of such a scheme.

3.1 Vault Scheme

Our vault scheme features three parties, a user, a storage server and an analyst. Figure 1 presents the full functionality of the proposed vault scheme.

Definition 2. *A vault scheme consists of the following probabilistic polynomial time algorithms:*

- $\mathsf{Param}(1^\lambda) \to PP$: is the algorithm that sets up the public parameters P. It takes as input the security parameter λ.
- $\mathsf{Store}(PP, ID, M, DB) \to (psswd, tok, DB')$: is the algorithm used by the user in the scheme to store its sensitive and non-sensitive data in the vault. It takes as input the parameters P, the user id ID, a message M that gets parsed as $M = (M_s, M_{ns})$, and the current state of the database DB and returns an access password $psswd$, a data token tok, and the updated state of the database DB'.
- $\mathsf{Access}_{DB}(PP, tok) \to (M_{ns})$: is the algorithm used by the analyst to access the non-sensitive data M_{ns} in the database. It takes in input λ, and the data token tok and outputs the corresponding non-sensitive data M_{ns}. This algorithm also has read access to the database DB.
- $\mathsf{Retrieve}_{DB}(PP, ID, psswd) \to (M_s, M_{ns})$: is the algorithm used by any entity that possesses the id and the password to retrieve the whole data M from the database. It takes as input the security parameter λ, and the access password $psswd$ and outputs the corresponding tuple of sensitive and non-sensitive data $M = (M_s, M_{ns})$. This algorithm also has read access to the database DB.

Correctness The correctness condition for the vault scheme requires that multiple invocation of $\mathsf{Store}(PP, M)$ is identical for identical PP and M. Formally for all $PP \leftarrow \mathsf{Param}$, all id $ID \in \{0,1\}^*$ and for all $(M_s, M_{ns}) = M \in \mathcal{M}$, if $(psswd, tok, DB') \leftarrow \mathsf{Store}(PP, ID, M)$, it holds that

- *(access correctness):* $\mathsf{Access}\,(PP, t)) = M_{ns}$
- *(retrieve correctness):* $\mathsf{Retrieve}\,(PP, ID, psswd) = M$

Vault scheme in a zero-trust cloud operating model In a zero-trust cloud operating model[16], none of the components are trusted by default. The applications have only limited private storage. The server/device which caries out the data storage operations for users is assumed to be trusted. However, apart from the simple verification information required to prove to the user its authenticity, we should try to minimize any requirement for private storage.

One option is to reuse the server certificate that gets used for proving authenticity of the server or a new secret key to encrypt any token data stored in the outsourced database. However both the options are sub-optimal. Reusing server certificate for encrypting token data would violate the best practices in terms of security. Also using any secret key during the storage of information requires added overhead of key management at this layer including decryption and re-encryption of the database as per the key rotation schedule. Another pitfall is if only a single key gets compromised all tokens become vulnerable.

To address these issues, we consider a trusted but stateless vault server that uses an outsourced database. For storing any large list or database, the applications will need to use a public third party server. We model the framework in which the entity storing the database is honest but curious. This fairly reflects the practical contingency of a successful attack on the database upon which the adversary gains a view of the database but the basic operations on the database are still controlled by the main server of the system.

Security Goals.In order to set a meaningful security notion, we must first identify our target security objective. The fundamental objective is two-fold - (1) the privacy of the sensitive data when the adversary holds the data token but does not have access to the password and (2) the privacy of both the sensitive and the non-sensitive data when the adversary has neither the data token nor the access password. In cryptographic nomenclature, the bare minimum requirement is that the Store() function must be *one-way*. Like the standard security notion in case of encryption schemes, however, we suggest the notion of *indistinguishability* as the correct target. To that end, we propose the following two security notions for the privacy of the vault scheme in terms of non-sensitive and sensitive information respectively.

Privacy for non-sensitive information. One fundamental privacy requirement from the tokenization scheme is the privacy of the of the non-sensitive part of the message without the possession of the access token. To reflect this requirement, we define a security notion which follows the spirit of classical indistinguishability. We denote this notion as the IND-CATA security notion. We present this using a security game in which the adversary, possessing the data token for a particular message, chooses two messages for which the non-sensitive part are different and sends them to the challenger. The challenger provides the adversary with the access token of one of the given messages and the challenge for the adversary is to guess correctly which non-sensitive message the token corresponds to. We formalize the IND-CATA security game in Figure 2.

Privacy for sensitive information The other, in some sense more essential, aspect of the privacy requirement for a vault scheme is the privacy of the sensitive data from any entity that does not possess the master access token. Any adversary $\mathcal{A}$ that possesses the data token, access token and the non-sensitive message, should not be able to extract any extra information about the sensitive part of the message, even when provided with access to the full database.

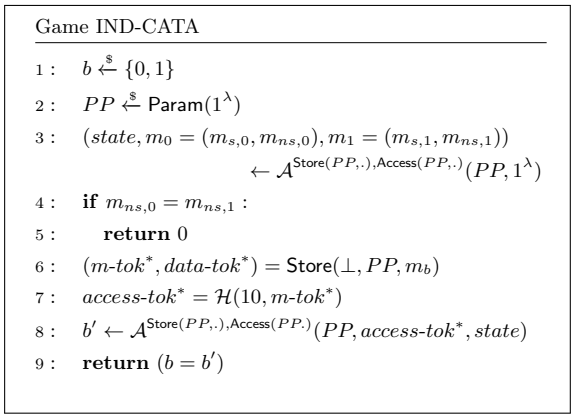

Fig. 2: IND-CATA security game: The adversary is not allowed to query Store with m_0, m_1, and Access with $access\text{-}tok^*, data\text{-}tok^*$

We propose an indistinguishability based security notion for retrieval of sensitive information game (IND-SIA) where we formalize the security of a vault scheme against an adversary that possesses the non-sensitive data of a message along with its data and access tokens. In this setting, the adversary also has complete access to the contents of the database. A negligible success probability in the IND-SIA game implies that the adversary cannot distinguish between the database records of two chosen sensitive messages.

The security game is formalized in Figure 3. In the security game, the challenger, at the beginning, uniformly and randomly chooses a b from $\{0,1\}$ and stores the value. The adversary is provided access to the state DB. Adversary then sends two messages m_0 and m_1 with distinct sensitive bits and the same non-sensitive bits. The challenger runs the Store() routine on message m_b and the adversary can observe the changed database state DB'.

The adversary, when ready, can send a bit b' to the challenger. The adversary wins the game if $b' = b$.

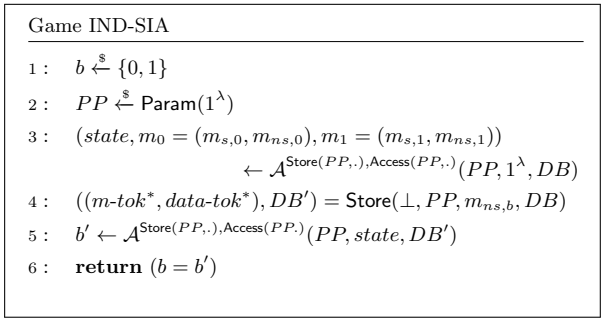

Fig. 3: IND-SIA security game: The adversary is not allowed to query Store with m_0, m_1, and Access with $access\text{-}tok^*, data\text{-}tok^*$

It is important to note that the security notion hinges on the fact that the adversary should not possess the master token.

4 Tokenization Scheme

Having established the definition of the vault scheme that we are aiming to construct, in this section, we introduce and formally define an ingredient that we shall use to design our vault scheme. This is a Tokenization scheme. We then formalize the exact security notions of such a Tokenization scheme.

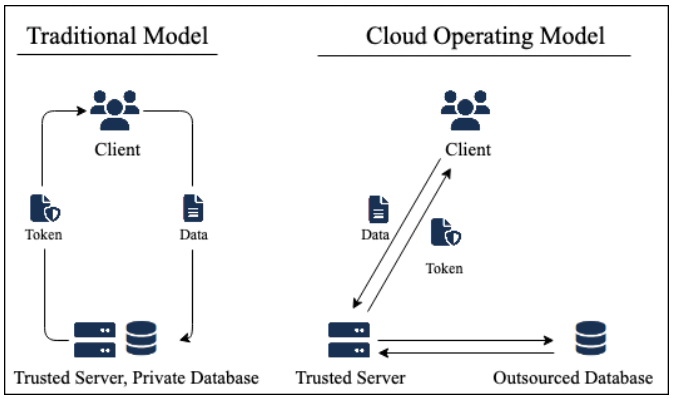

Fig. 4: Tokenization in the Cloud Operating Model

4.1 Definition of Tokenization Scheme

A Tokenization scheme $\mathsf{TOK} = (\mathsf{Param}, \mathsf{Tok}, \mathsf{DeTok})$ is a tuple of three polynomial time algorithms. On input 1^λ, the parameter generation algorithm Param returns public parameters P. The algorithms Tok,DeTok have a common state/-database DB. On inputs P, and a message M, the tokenization algorithm Tok returns a *token* $t \xleftarrow{\$} \mathsf{Tok}(P, M)$. On inputs, P and a token t, the detokenization algorithm DeTok returns $\mathsf{DeTok}(P, t) \in \{0,1\}^* \cup \{\perp\}$. A message space $\mathcal{M}$ and a token space $\mathcal{T}$ is associated with the scheme.
The *tokenization correctness* condition requires multiple invocations of $\mathsf{Tok}(P, M)$ to be identical for identical P and M. The *detokenization correctness* requires that for all $P \leftarrow \mathsf{Param}$, all $M \in \mathcal{M}$, it holds that $\mathsf{DeTok}\,(P, \mathsf{Tok}(P, M)) = M$.

Security Goals. Following the intuition of the security notions that we have defined for the overall vault scheme, we now try to identify our target security objective for the basic tokenization scheme that we use as the building block for the vault scheme. As discussed before, we again propose the notion of *indistinguishability* as the correct target just as the standard security notion in case of encryption schemes.

Indeed, the schemes should achieve security against an adversary who may possess some data-token and sensitive data pairs. Extending the analogy with encryption schemes, we propose the notion of *Indistinguishability* against *Chosen-Token-Attack* (IND-CTA). In this setting, the adversary has oracle access to the tokenization and detokenization routines. The adversary submits two messages m_0 and m_1, and receives a data token t^* which is generated by choosing a random $b \xleftarrow{\$} \{0,1\}$ and executing $\mathsf{Store}(PP, m_b)$. The adversary wins the game if they can correctly guess the value of b. The adversary is not allowed to call the store routine on m_0 and m_1 and the retrieve routine on t^* to make sure the game is not won trivially.

A straightforward scheme that samples tokens randomly and stores them in a table for lookup achieves this IND-CTA security.

In addition to the above, any vault scheme must also ensure the data tokens and the corresponding sensitive data cannot be linked without making a query to the tokenization service. However, such adversaries won't have access to the vault database. For an adversary having access to the vault database, knowledge of either the sensitive data or the data token reveals the other as well, because in a zero trust environment we can not use any secret key. Thus when the parameters PP as well as the states DB are all public, we can not achieve the IND-CTA notion here. The adversary could download the database, locally simulate the retrieve algorithm with the challenge token t^* and find the corresponding message.

As with deterministic public key encryption [6] and message-locked encryption [7], we need semantic security of the database when the messages are unpredictable (sampled from high min-entropy distribution). We formalize an indistinguishability against Chosen-Distribution-Database-Access attack (IND-CDDA) security notion where the database entries resulting from an unpredictable message looks random. Thus, we can argue that making the database public does not provide any additional advantage to the adversary. Our goal is to find a scheme that is secure against an IND-CTA adversary in the classical setting as well as against IND-CDDA attackers in the cloud-operating setting separately.

4.2 Security Notion for Tokenization Scheme

The fundamental privacy requirement from the tokenization scheme is the privacy of the messages given the corresponding token. In the spirit of classical indistinguishability, the natural notion of privacy would be *Indistinguishability* against *Chosen-Data-Attack* (IND-CDA) or *Indistinguishability* against *Chosen-Token-Attack* (IND-CTA). In case of centralized tokenization scheme, the above security notions could be achieved without much complications. In Figure 5, we formalize the IND-CTA security game.

Tokenization with Remote Storage In case of tokenization with remote storage, the functionalities remain the same. The difference however is in the view of the adversary. As the state DB is now public, the adversary $\mathcal{A}$ has (at least) read access of DB.

Game IND-CTA

$b \xleftarrow{\$} \{0,1\}$

$P \xleftarrow{\$} \mathsf{Param}(1^\lambda)$

$(state, m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{Tok}(\mathsf{PP},.),\mathsf{DeTok}(PP,.)}(PP, 1^\lambda)$

$t^* = \mathsf{Tok}(PP, m_b)$

$b' \leftarrow \mathcal{A}^{\mathsf{Tok}(\mathsf{PP},.),\mathsf{DeTok}(PP.)}(PP, t^*, state)$

return $(b = b')$

Fig. 5: IND-CTA security game: The adversary is not allowed to query Tok with m_0, m_1, and DeTok with t^*

Procedure Main(1^λ)

$b \xleftarrow{\$} \{0,1\}$

$P \xleftarrow{\$} \mathsf{Param}(1^\lambda)$

$b' \leftarrow \mathcal{A}^{DB,\mathsf{Pplt},\mathsf{Tok}(\mathsf{PP},.),\mathsf{DeTok}(PP,.)}(1^\lambda)$

return $(b = b')$

Procedure Pplt

if $b = 0$

  $M \xleftarrow{\mathcal{D}} \mathcal{M}$

  $t \leftarrow \mathsf{Tok}(P, M)$

if $b = 1$

  $d \xleftarrow{\$} \{0,1\}^\rho$

  Add d to DB

endif

Fig. 6: IND-CDDA security game: $\mathcal{D}$ is the distribution of an unpredictable source.

In this paper, we consider privacy of tokens in the distributed setting with minimal secret. Naturally, we can not hope to provide indistinguishability of database entries for predictable messages. Thus we formalize the security notion for unpredictable messages. As in the literature, we formalize the notion of a source with arity 1 as a polynomial-time algorithm that on input 1^λ outputs $\mathbf{M} = (m_1, m_2, \ldots, m_\ell)$ where each $m_i \in \{0,1\}^\mu$ and for distinct i, j it holds that $m_i \neq m_j$[4]. A source is called *unpredictable* if $\min_i H_\infty(m_i) \geq \mathcal{O}(\lambda)$. In other words, for an unpredictable source, it holds that $\max_i \max_{x \in \{0,1\}^\mu} \Pr[m_i = x]$ is negligible.

Privacy in presence of remote Storage We propose the notion of indistinguishability against Chosen-Distribution-Database-Access attack game (IND-CDDA) where we formalize the security of a tokenization scheme against a database that is honest-but-curious. In this setting, the adversary has access to the database and can contribute certain records to the database while other records are added independently. A negligible success probability in the IND-CDDA game means that the adversary cannot distinguish the independently generated queries from random.

[4] it is possible to cosider the defition of a source where the $|m_i|$ varies over i. However, for our program it suffices consider the simpler framework without losing any generality.

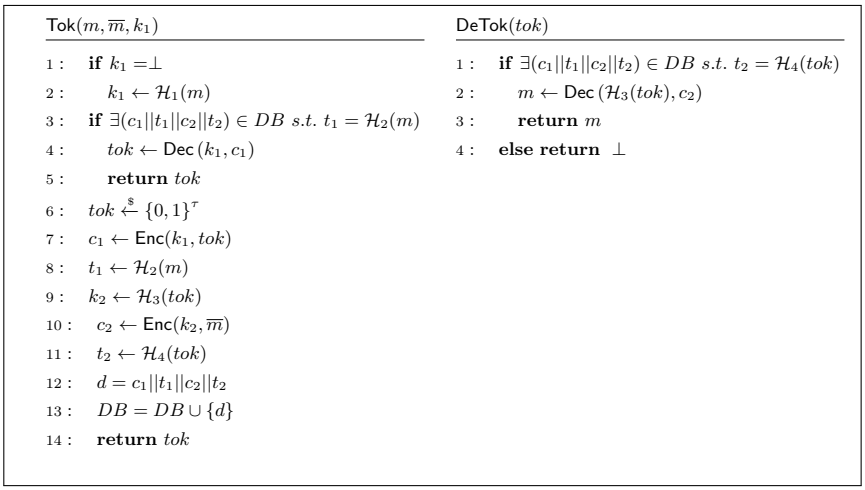

Fig. 7: Our Random Oracle Model Construction. The token space and message space are identical, $\mathcal{M} = \{0,1\}^\mu$. $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4$ denote the functions $\mathcal{H}(00,\cdot)$,$\mathcal{H}(01,\cdot)$, $\mathcal{H}(10,\cdot)$, $\mathcal{H}(11,\cdot)$ respectively. For basic construction $k_1 = \perp, m = \overline{m}$.

The security game is formalized in Figure 6. In the security game, the challenger, at the beginning, uniformly and randomly chooses a b from $\{0,1\}$ and stores the value. The adversary is provided access to the state DB, procedures Tok, DeTok oracles and an additional interface "Populate Database", which as described below -

- The Tokenization procedure, Tok: The procedure takes as input a message m, generates a token and adds an entry to the database corresponding to the message and the token. The procedure returns the token as output.
- The DeTokenization procedure, DeTok: The procedure takes as input a token t and returns the corresponding message as output, if it exists. Otherwise it returns $\perp$.
- The Populate Database Routine: The adversary can call this routine in which the challenger does one of the following - (a) if $b = 0$, challenger samples a message m from the message space $\mathcal{M}$ following the input distribution $\mathcal{D}$. Tokenization algorithm is invoked on m. Note the generated token is not returned to the adversary. (b) If $b = 1$, the challenger augments the database with a randomly chosen entry.

The adversary, when ready, can send a bit b' to the challenger. The adversary wins the game if $b' = b$.

Note that although the adversary does have access to the database, he does not possess any tokens corresponding to messages that he did not tokenize himself (by calling the Tok routine). He can, therefore, only invoke the DeTokenization routine on tokens that he generated himself.

4.3 Our Construction

We propose the following construction for a secure tokenization scheme. The pseudocode is given in the Figure 7.
Ingredients. The scheme is built from a hash function familty $H = (\mathcal{HK}, \mathcal{H})$, and an encryption scheme $SE = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. The key generation algorithm Gen, on input 1^λ, produces a key $k \xleftarrow{\$} \mathcal{K}$. Then Enc algorithm takes the key k and a message $m \in \mathcal{M}$ as input and produces a ciphertext c as output. Then Dec algorithm produces a message m, on input the key k and a ciphertext c . We require the encryption scheme to be perfectly correct; for all $m \in \mathcal{M}, k \in \mathcal{K}$ it holds that $\Pr[\,\mathsf{Dec}\,(k, \mathsf{Enc}(k, m)) = m\,]$. We assume that hash function produces an element from $\mathcal{K}$; the keyspace of the encryption scheme. We require IND-CPA security from SE.
Our Construction. The tokens are element of $\{0,1\}^\tau$. The scheme $V = (\mathsf{Tok}, \mathsf{DeTok})$ is described in Figure 7.

Theorem 1. *Let $\mathcal{H}\colon \{0,1\}^{2+\mu} \to \{0,1\}^k$ be a Random Oracle. Let $SE = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an* IND-CPA *secure encryption scheme with key length k. Then the tokenization scheme $V = (\mathsf{Tok}, \mathsf{DeTok})$ is* IND-CTA.

Proof. In the IND-CTA game, the database is private to the adversary. Thus the database contents are independent from the adversary's view. As the tokens are sampled independently from the messages following the uniform distribution, the adversary can not distinguish a token from a random string. Thus the scheme satisfies IND-CTA security.

Theorem 2. *Let $\mathcal{H}\colon \{0,1\}^{2+\mu} \to \{0,1\}^k$ be a Random Oracle. Let $SE = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an* IND-CPA *secure encryption scheme with key length k. Then the tokenization scheme $V = (\mathsf{Tok}, \mathsf{DeTok})$ is* IND-CDDA *secure.*

Proof. Let IND-CDDA1 be the IND-CDDA game with $b = 1$ and IND-CDDA0 be the IND-CDDA game with $b = 0$. From the standard definition of advantage, we have,

$$\mathsf{Adv}^{\text{IND-CDDA}}_{V,\mathcal{M},A}(\lambda) = \\ \left|\Pr[\text{IND-CDDA1}^A_{V,\mathcal{M}}(1^\lambda)] - \Pr[\text{IND-CDDA0}^A_{V,\mathcal{M}}(1^\lambda)]\right| \tag{1}$$

We proceed via the notion of game playing proof. The first hybrid game G_1 is completely identical to the IND-CDDA0 game (Figure 8). We assume, without loss of generality, adversary queries $\mathcal{H}_1$ and $\mathcal{H}_2$ together (interface $\mathcal{H}_{1,2}(m)$). Similarly $\mathcal{H}_3$ and $\mathcal{H}_4$ are assumed to be queried together (interface $\mathcal{H}_{3,4}(m)$).

Subsequently, in the next hybrid game, G_2, we replace all 4 instances of the Random Oracle function (in steps 6, 8, 9 and 11) with each of the outputs instead being sampled uniformly and randomly from the corresponding range spaces of the four hash functions – $\{0,1\}^k$ for the functions $\mathcal{H}_1$ and $\mathcal{H}_3$ (which is the key length for the encryption scheme SE), $\{0,1\}^{\rho_1}$ for $\mathcal{H}_2$ and $\{0,1\}^{\rho_2}$ for $\mathcal{H}_4$. In order to bound the probability of bad and bad' in game $G2$, we observe that any

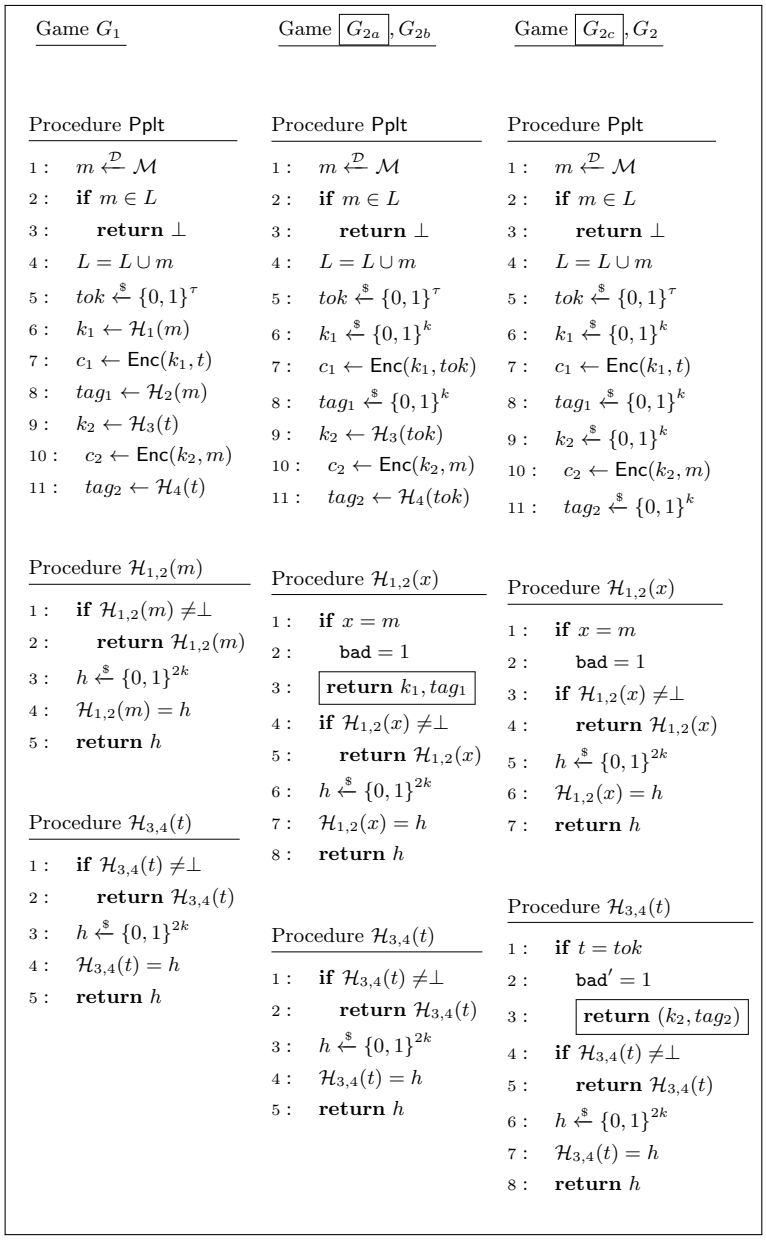

Fig. 8: Game G_1, G_2

adversary $\mathcal{B}$ setting bad and bad' implies that they could get tok or m, and thus breaking the one-way property of SE.

$$\begin{aligned} \Pr[G_1^A] &\leq \Pr[G_2^A] + 2\mathsf{Adv}^{\text{one-way}}_{SE,\mathcal{B}}(\lambda) + \text{negl}(\lambda) \\ &\leq \Pr[G_2^A] + 2\mathsf{Adv}^{\text{ROR}}_{SE,\mathcal{B}}(\lambda) + \text{negl}(\lambda) \end{aligned} \tag{2}$$

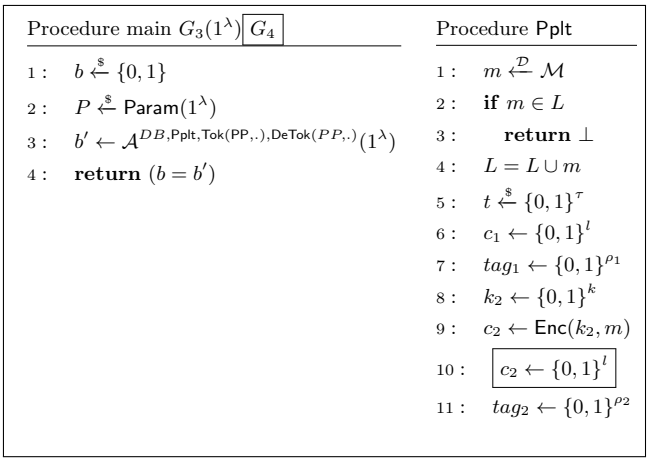

Fig. 9: Game G_3 and G_4

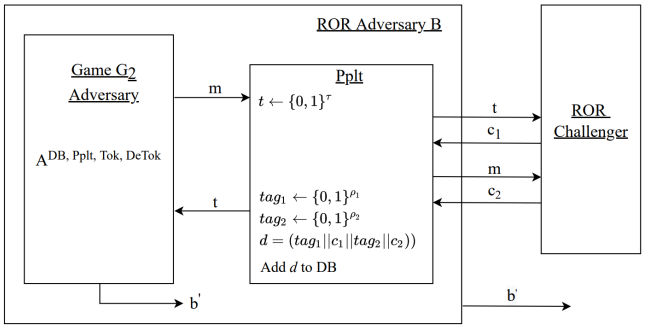

Fig. 10: Real-or-Random Adversary B

The last inequality follows as the one-way advantage of any adversary is at most the ROR advantage against a symmeteric encryption SE.

We now upper bound the transition from G_2 to G_3 (and from G_3 to G_4 subsequently) by reducing to an ROR_{SE} adversary B. Adversary B operates in the exact same way as the challenger in Game G_2 apart from the generation of the ciphertext c_1. Instead, B first queries its ROR Game challenger with the message t and sets c_1 to be the query output from the ROR challenger. For the first transition from G_2 to G_3, k_2 is chosen uniformly and randomly and c_2 is then generated as the encryption of m.

For the transition from Game G_3 to G_4, adversary B generates both c_2 as well as c_1 by querying the ROR challenger. c_2 is generated by querying m. In both of the cases, the adversary B gets output b' from the CDDA adversary and outputs the same b'. The two transitions are consolidated as one adversary in Figure 7.

From the definition of the Real-or-Random security game, we can see that when the ROR challenger sets $b = 0$, it executes exactly Game G_2^A – choosing

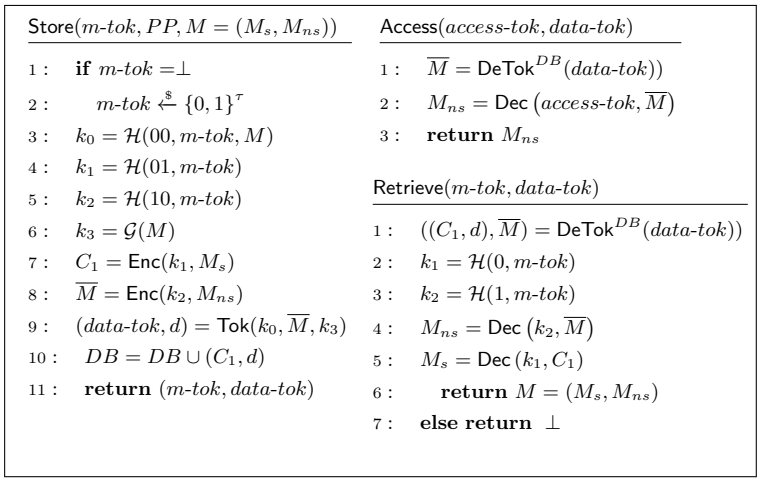

Fig. 11: Our Random Oracle Model Construction for the full strength scheme. The token space and message space are identical, $\mathcal{M} = \{0,1\}^{\mu}$. The access token for any message is the key k_2 generated in the Store() routine. This is referred to as *access-tok* in the Access() routine.

keys k_1 and k_2 uniformly and randomly and then calculating the ciphertexts c_1 and c_2 from the corresponding queries t and m. As such, adversary B outputs 1 only when adversary A outputs 1 when $b = 0$.

On the other hand, when the ROR challenger sets $b = 1$, it executes exactly Game G_4^A – choosing the ciphertexts c_1 and c_2 uniformly and randomly from the ciphertext space which is $\{0,1\}^l$ where l is the length of a ciphertext in the encryption scheme SE. As such, adversary B outputs 1 only when adversary A outputs 1 when $b = 1$. Thus, we have,

$$\begin{aligned}\mathsf{Adv}^{\mathrm{ror}}_{SE,\,B}(\lambda) &= \Pr[\mathrm{ROR}^B_{SE}] = \Pr[\mathrm{ROR1}^B_{SE} \Rightarrow 1] - \Pr[\mathrm{ROR0}^B_{SE} \Rightarrow 1] \\ &= \Pr[G_2^A] - \Pr[G_4^A] \end{aligned} \tag{3}$$

Game G_4 (Figure 6) is exactly identical to the IND-CDDA1 game. From Equation (1), we then have -

$$\mathsf{Adv}^{\text{IND-CDDA}}_{V,\mathcal{M},A}(\lambda) = \Pr[G_1^A] - \Pr[G_4^A] \tag{4}$$

Combining Equations (2), (3) and (4), we have -

$$\mathsf{Adv}^{\text{IND-CDDA}}_{V,\mathcal{M},A}(\lambda) \leq 3\mathsf{Adv}^{\mathrm{ror}}_{SE,\,B}(\lambda) + \mathrm{negl}(\lambda)$$

5 Towards Full-strength Vault Scheme

We propose the following construction for the vault scheme. The pseudocode for the algorithms are given in the Figure.

INGREDIENTS. The scheme is built from a hash function family $H = (\mathcal{HK}, \mathcal{H})$, and an encryption scheme $SE = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. The key generation algorithm

Gen, on input 1^λ, produces a key $k \xleftarrow{\$} \mathcal{K}$. Then Enc algorithm takes the key k and a message $m \in \mathcal{M}$ as input and produces a ciphertext c as output. Then Dec algorithm produces a message m, on input the key k and a ciphertext c . We require the encryption scheme to be perfectly correct; for all $m \in \mathcal{M}, k \in \mathcal{K}$ it holds that $\Pr[\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m] = 1$. We assume that hash function produces an element from $\mathcal{K}$; the keyspace of the encryption scheme. We require IND-CPA security from SE.

Our Construction. The tokens are element of $\{0,1\}^\tau$. The scheme $V = (\mathsf{Param}, \mathsf{Store}, \mathsf{Access}, \mathsf{Retrieve})$ is described in Figure 11.

Theorem 3. *Let $\mathcal{H}\colon \{0,1\}^{2+\mu} \to \{0,1\}^k$ be a Random Oracle. Let $SE = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an* IND-CPA *secure encryption scheme with key length k. Then the tokenization scheme $V = (\mathsf{Tok}, \mathsf{DeTok})$ is* IND-CATA *and* IND-SIA *secure.*

Proof (Proof Sketch). IND-CATA *Security:* The IND-CATA security of the full vault scheme follows from the randomness of the hash function $\mathcal{H}(10, .)$ and the IND-CTA and the IND-CDDA security of the underlying tokenization scheme. In this setting, the database is private to the adversary. Thus the database contents are independent from the adversary's view.

Since we have made the assumption that the hash function $\mathcal{H}()$ is a Random Oracle, the generated access token k_2 is indistinguishable from a random string of the same length. Further, since k_2 is random, the IND-CPA security of the encryption scheme $SE = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ ensures that the plaintext, which is M_{ns} in this case, remains private.

Similarly, k_0 and k_3 also end up getting sampled from a uniformly random distribution, again owing to the random oracle property of the hash functions $\mathcal{H}()$ and $\mathcal{G}()$. This, in turn, implies the IND-CTA and IND-CDDA security of the Tokenization scheme $V = (\mathsf{Tok}, \mathsf{DeTok})$.

Therefore, given an adversary $\mathcal{A}$ that breaks the IND-CATA security of the Vault scheme, we can construct an adversary $\mathcal{B}$ that breaks either the Random Oracle property of the hash function $\mathcal{H}()$ or the IND-CTA and IND-CDDA properties of $V = (\mathsf{Tok}, \mathsf{DeTok})$.

An important note is that even though the adversary has access to the $\mathsf{Store}()$ routine in this security game, they do not possess the master token and therefore, cannot reconstruct the access token by themselves. Thus, the vault scheme is not vulnerable to such an attack.

IND-SIA *Security:* The IND-SIA security notion deals with the ability of the adversary to correctly distinguish between two sensitive messages from the complete database records corresponding to the two messages.

Given our assumption that the hash function $\mathcal{H}()$ is a Random Oracle hash function, the key k_1, generated by applying $\mathcal{H}(01, \cdot)$ on the master token *m-tok*, is going to be uniformly random since *m-tok* is generated uniformly at random as well. This also holds true for the access token which is denoted by k_2 and is generated by applying $\mathcal{H}(10, \cdot)$ on the master token *m-tok*. This further implies that the keys k_1 and k_2 are independent of each other. Therefore, knowledge of the access code, that is, k_2, does not let the adversary have a greater advantage

in guessing k_1. In other words, k_1 would still be indistinguishable from random for the IND-SIA adversary.

Once we have established this, the IND-SIA property trivially reduces to the IND-CPA property of the encryption scheme $SE = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. Given an adversary $\mathcal{A}$ that breaks the IND-SIA game for the vault scheme (by correctly guessing the M_{ns} with a non-negligible advantage), the same adversary can be used to break the IND-CPA security of the encryption scheme since the key k_1 is completely random.

6 Conclusion

In this paper, we introduce and define a Vault Scheme and provide a formalization of the security of the same. To design a secure vault scheme, we formalize the more basic building block of a Tokenization scheme. In order to use it, we define a security notion for the privacy of a tokenization scheme in the traditional setting of having a secure database storage. We then formalize a security notion for tokenization in the security model of a non-secure database. This corresponds to the setting of privacy against a honest-but-curious storage server and is the security setting in many real-life applications where tokenization is being used. We then design a practical tokenization scheme and prove it to be secure in the Random Oracle model.

Having provided a secure base for our construction, we proceed to the design of a full fledged deterministic Vault scheme. Finally, we provide a security proof for the privacy properties of our proposed Vault scheme.

Acknowledgement. RB is supported by EPSRC via EP/Y001680/1. We thank Manish Ahluwalia (Skyfow Inc.) for helpful discussions and suggesting the initial idea that inspired this project.

References

1. Anonomatic. PII Compliance Made Easy with PII as a Service. `https://anonomatic.com/`.
2. ANSI. ANSI X9.119-2-2017, Retail Financial Services - Requirements For Protection Of Sensitive Payment Card Data - Part 2: Implementing Post-Authorization Tokenization Systems. `https://webstore.ansi.org/Standards/ASCX9/ansix91192017`.
3. Riccardo Aragona, Riccardo Longo, and Massimiliano Sala. Several proofs of security for a tokenization algorithm. *Appl. Algebra Eng., Commun. Comput.*, 28(5):425–436, nov 2017.
4. AWS. AWS serverless tokenization. `https://github.com/aws-samples/aws-serverless-tokenization/`.
5. Basis Theory. What are Tokens? `https://developers.basistheory.com/docs/concepts/what-are-tokens`.

6. Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, pages 535–552, 2007.
7. Mihir Bellare, Sriram Keelveedhi, and Thomas Ristenpart. Message-locked encryption and secure deduplication. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, pages 296–312, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
8. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006.
9. BoxyHQ. LLM Vault. The secrets layer for AI. `https://boxyhq.com/`.
10. CyberNews. ChatGPT tied to Samsung's alleged data leak. `https://cybernews.com/news/chatgpt-samsung-data-leak/`.
11. Delinea. Delegated Machine Credentials. `https://delinea.com/`.
12. Sandra Díaz-Santiago, Lil María Rodríguez-Henríquez, and Debrup Chakraborty. A cryptographic study of tokenization systems. *Int. J. Inf. Secur.*, 15(4):413–432, aug 2016.
13. F. Betül Durak, Henning Horst, Michael Horst, and Serge Vaudenay. Fast: Secure and high performance format-preserving encryption and tokenization. Cryptology ePrint Archive, Paper 2021/1171, 2021. `https://eprint.iacr.org/2021/1171`.
14. Entrust. Entrust, Token Manager Solutions. `https://www.entrust.com/`.
15. Evervault. How Evervault Secures Data with Encryption and Privacy Controls. `https://evervault.com/`.
16. Hashicorp. Clound Operating Model. `https://www.hashicorp.com/cloud-operating-model`.
17. Hashicorp. Hashicorp, Vault. `https://www.hashicorp.com/products/vault`.
18. PCI Security Standards. PCI DSS Requirements and Security Assessment Procedures, Version 3.2. Technical Report (2016). `https://listings.pcisecuritystandards.org/documents/PCI_DSS-QRG-v3_2_1.pdf`.
19. Protegrity. AI-Powered dagta security designed for data consumption. `https://protegrity.com/`.
20. Skyflow. Data Privacy with Skyflow LLM Privacy Vault. `https://skyflow.com/post/generative-ai-data-privacy-skyflow-llm-privacy-vault`.
21. Very Good Security (VGS). What is VGS? Protecting Your Data with Very Good Security. `https://www.verygoodsecurity.com/`.
22. Jia Xu, Ee-Chien Chang, and Jianying Zhou. Weak leakage-resilient client-side deduplication of encrypted data in cloud storage. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ASIA CCS '13, page 195–206, New York, NY, USA, 2013. Association for Computing Machinery.
23. Pan Yang, Naixue Xiong, and Jingli Ren. Data security and privacy protection for cloud storage: A survey. *IEEE Access*, 8:131723–131740, 2020.
24. Yinghui Zhang, Robert H. Deng, Shengmin Xu, Jianfei Sun, Qi Li, and Dong Zheng. Attribute-based encryption for cloud computing access control: A survey. *ACM Comput. Surv.*, 53(4), aug 2020.