Verifiable Hierarchical Key Assignment Schemes

Anna Lisa Ferrara, Federica Paci, Chiara Ricciardi

Verifiable Hierarchical Key Assignment Schemes

Anna Lisa Ferrara[1], Federica Paci[2], and Chiara Ricciardi[1]

[1] Università degli Studi del Molise, Italy,
annalisa.ferrara@unimol.it; c.ricciardi1@studenti.unimol.it
[2] Università degli Studi di Verona, Italy
federicamariafrancesca.paci@univr.it

Abstract. A Hierarchical Key Assignment Scheme (HKAS) is a method to assign some private information and secret keys to a set of classes in a partially ordered hierarchy, so that the private information of a higher class together with some public information can be used to derive the keys of all classes lower down in the hierarchy. Historically, HKAS has been introduced to enforce multi-level access control, where it can be safely assumed that the public information is made available in some authenticated form. Subsequently, HKAS has found application in several other contexts where, instead, it would be convenient to certify the trustworthiness of public information. Such application contexts include key management for IoT and for emerging distributed data acquisition systems such as wireless sensor networks. In this paper, motivated by the need of accommodating this additional security requirement, we first introduce a new cryptographic primitive: *Verifiable Hierarchical Key Assignment Scheme* (VHKAS). A VHKAS is a key assignment scheme with a verification procedure that allows honest users to verify whether public information has been maliciously modified to induce an honest user to obtain an incorrect key. Then, we design and analyse VHKASs which are provably secure. Our solutions support key update for compromised secret keys by making a limited number of changes to public and private information.

Keywords: hierachical key assignment, access control, applied cryptography

1 Introduction

Users of a computer system could be organized into a hierarchy consisting of a number of separate classes. These classes, called security classes, are positioned and ordered within the hierarchy according to the fact that some users have more access rights than others. For instance, in a hospital, doctors can access their patients' medical records, while researchers can only consult anonymous clinical information for studies.

A hierarchical key assignment (HKAS) scheme is a method to assign a secret key and some private information to each class in the hierarchy so that keys for descendant classes can be obtained via a key derivation procedure. This

assignment is carried out by a central authority, the Trusted Authority (TA). Following the seminal work by Akl and Taylor [2], many researchers have proposed different HKASs that either have better performances or allow dynamic updates to the hierarchy (e.g., [3, 5, 10, 14, 12, 16, 19]). Crampton et al. in [15] provided a detailed classification for HKASs, according to several parameters, including memory requirements for public and private information and the complexity of handling dynamic updates. In particular, they identified families of schemes where the public information is used to store secret keys in order to reduce the amount of private information users need to manage[3]. The use of HKASs belonging to such families is desirable to prevent or limit the change of private information and its redistribution when handling encryption key updates. Indeed, for these schemes, the key update procedure which is necessary to replace compromised keys, often requires changing only the public information without the need of interacting with the involved users. In the remainder of the paper, we refer to HKAS schemes belonging to such families.

Historically, HKASs have been introduced to enforce multi-level access control in scenarios where it can be safely assumed that the public information is made available to everyone via a publicly accessible repository for which only the TA has write permissions. However, key assignment schemes have recently been employed in different application context where the public information may be exposed to changes by malicious users. These application contexts include key management for IoT and distributed data acquisition systems such as wireless sensor networks [4, 10, 24] as well as sensitive data outsourcing to the cloud [9, 11, 17, 18, 23]. For instance, consider sensitive data outsourcing in cloud; the data owner encrypts the data before outsourcing them at the server and distributes the private information (i.e., secret keys used to encrypt the data and derivation material) to the users by means of an HKAS according to the access policy. Only the data owner and users who know the appropriate encryption and derivation keys will be able to decrypt the data. However, metadata which includes the public information will also be stored at the server and thus may be modified voluntarily or involuntarily by those who have access to it, including the cloud service provider which is not necessarily trusted. Unfortunately, a change in the public information will prevent an honest user to derive a correct decryption key. Similarly, in wireless sensor networks, the cluster head nodes are responsible for forwarding any public information that has been changed as a result of key updates. If a cluster head node is corrupted, this information may be maliciously modified before reaching its destination.

The scenarios outlined above introduce the need for an honest user to verify the trustworthiness of the public information. In order to accommodate this additional security requirement, we introduce a new cryptographic primitive: *Verifiable Hierarchical Key Assignment Scheme* (VHKAS). A VHKAS is a key assignment scheme with a verification procedure that allows honest users to verify whether public information has been maliciously modified. In order to capture a notion of security against an adversary who has the ability to replace or

[3] Such families include IKEKAS, DKEKAS, and TKEKAS[15].

modify the public information, we introduce the notion of *strong key-consistency*. This notion models the fact that even the TA is unable to maliciously modify the public information once distributed the private information.

More in detail, our contributions are as follows:

- We first give formal definitions for VHKAS; and the notion of security *strong key-consistency*;
- subsequently, we present a construction of VHKAS that uses as building block a Message Locked Encryption (MLE) scheme. We show that the construction is provably-secure with respect to strong key-consistency and *key indistinguishability*, which corresponds to the requirement that an adversary is not able to learn any information about a key that it should not have access to;
- afterwards, we show how to handle key replacement for compromised secret keys by making a limited number of changes to public and private information;
- finally, we instantiate our MLE-based construction with the deterministic MLE scheme proposed by Abadi et al.[1].

The paper is organized as follows: in Section 2 we review the definitions of HKAS, and MLE as well as their notions of security. In Section 3 we define VHKAS and introduce the security notion of strong key-consistency. In Section 4 we show our MLE-based construction, prove it to be provably-secure with respect to key-consistency and key indistinguishability and instantiate it with the deterministic MLE scheme proposed by Abadi et al.[1]. In Section 5 we show how to handle key replacements. In Section 6 we evaluate the performance of our construction by comparing it with that of popular HKAS from the literature, while Section 7 concludes the paper.

2 Preliminaries

Notation. We use the standard notation to describe probabilistic algorithm and experiments. If $A(\cdot,\cdot,\ldots)$ is any probabilistic algorithm then $a \leftarrow A(x,y,\ldots)$ denotes the experiment of running A on inputs $x, y, \ldots$ and letting a be the outcome, the probability being over the coins of A. Similarly, if X is a set then $x \leftarrow X$ denotes the experiment of selecting an element uniformly from X and assigning x this value. If w is neither an algorithm nor a set then $x \leftarrow w$ is a simple assignment statement. For two bit-strings x and y we denote by $x||y$ their concatenation. A function $\epsilon : \mathbb{N} \rightarrow \mathbb{R}$ is *negligible* if for every constant $c > 0$ there exists an integer n_c such that $\epsilon(n) < n^{-c}$ for all $n \geq n_c$.

2.1 Hierarchical Key Assignment Schemes

Consider a set of users divided into a number of disjoint classes, called *security classes*. A binary relation $\preceq$ that partially orders the set of classes V is defined in accordance with authority, position, or power of each class in V. The poset $(V, \preceq)$

is called a *partially ordered hierarchy*. For any two classes u and v, the notation $u \preceq v$ is used to indicate that the users in v can access u's data. The partially ordered hierarchy $(V, \preceq)$ can be represented by the directed graph $G = (V, E)$, where each class corresponds to a vertex and there is a path from class v to class u if and only if $u \preceq v$. A hierarchical key assignment scheme is a method to assign a secret key and some private information to each class in the hierarchy. The private information will be used by each class to compute the keys assigned to all classes lower down in the hierarchy. This assignment is carried out by the TA. Formally:

Definition 1 ([20]). *Let Γ be a family of graphs corresponding to partially ordered hierarchies. A* HKAS *for Γ is a pair (Gen, Der) of algorithms satisfying the following conditions:*

1. *The* information generation algorithm *Gen is probabilistic polynomial-time. It takes as input the security parameter 1^τ and a graph $G = (V, E)$ in Γ, and produces as outputs*
   - *a private information s_u, for any class $u \in V$;*
   - *a key k_u, for any class $u \in V$;*
   - *a public information pub.*

   We denote by (s, k, pub) the output of the algorithm Gen, where s and k denote the sequences of private information and of keys, respectively.
2. *The* key derivation algorithm *Der is deterministic polynomial-time. It takes as input the security parameter 1^τ, a graph $G = (V, E)$ in Γ, two classes u, v in V, the private information s_u assigned to class u and the public information pub, and produces as output the key k_v assigned to class v if $v \preceq u$, or a special rejection symbol $\perp$ otherwise.*

   We require that for each class $u \in V$, each class $v \preceq u$, each private information s_u, each key k_v, each public information pub which can be computed by Gen on inputs 1^τ and G, it holds that $Der(1^\tau, G, u, v, s_u, pub) = k_v$.

Security notions. Atallah et al. [5] first introduced two different security goals for HKASs: security with respect to key indistinguishability and security against key recovery. In this paper, we only consider the stronger notion of key indistinguishability [6, 5].

$\texttt{STAT}_u$ is a static adversary who wants to attack a class $u \in V$ and who is able to corrupt *all* users not entitled to compute the key of class u. Algorithm $Corrupt_u$ which, on input the private information s generated by the algorithm Gen, extracts the secret values s_v associated to each class that the adversary is able to corrupt. In the indistinguishability game, the adversary must distinguish the key of class u from a random value.

Definition 2. `[IND-ST]` *Let Γ be a family of graphs corresponding to partially ordered hierarchies, let $G = (V, E)$ be a graph in Γ, let (Gen, Der) be a HKAS for Γ and let $\texttt{STAT}_u$ be a static adversary who attacks a class u. Consider the following two experiments:*

Experiment $\mathbf{Exp}^{\mathtt{IND}-1}_{\mathtt{STAT}_u}(1^\tau, G)$	Experiment $\mathbf{Exp}^{\mathtt{IND}-0}_{\mathtt{STAT}_u}(1^\tau, G)$
$(s, k, pub) \leftarrow Gen(1^\tau, G)$	$(s, k, pub) \leftarrow Gen(1^\tau, G)$
$corr \leftarrow Corrupt_u(s)$	$corr \leftarrow Corrupt_u(s)$
$d \leftarrow \mathtt{STAT}_u(1^\tau, G, pub, corr, k_u)$	$\rho \leftarrow \{0,1\}^{length(k_u)}$
return d	$d \leftarrow \mathtt{STAT}_u(1^\tau, G, pub, corr, \rho)$
	return d

The advantage of $\mathtt{STAT}_u$ *is defined as* $\mathbf{Adv}^{\mathtt{IND}}_{\mathtt{STAT}_u}(1^\tau, G) = |Pr[\mathbf{Exp}^{\mathtt{IND}-1}_{\mathtt{STAT}_u}(1^\tau, G) = 1] - Pr[\mathbf{Exp}^{\mathtt{IND}-0}_{\mathtt{STAT}_u}(1^\tau, G) = 1]|$. *The scheme is* secure in the sense of IND-ST *if, for each graph* $G = (V, E)$ *in* Γ *and each* $u \in V$*, the function* $\mathbf{Adv}^{\mathtt{IND}}_{\mathtt{STAT}_u}(1^\tau, G)$ *is negligible, for each adversary* $\mathtt{STAT}_u$ *with time complexity polynomial in* τ[4]*.*

2.2 Message-Locked Encryption

An Message-Locked Encryption (MLE) is a symmetric encryption scheme in which the key is itself derived from the message. Provably-secure MLE schemes have been first proposed by Bellare et al. in [8].

Definition 3 ([1]). *A* MLE *scheme is a tuple* $(PPGen, KD, Enc, Dec, Valid)$[5] *of algorithms satisfying the following conditions:*

1. *The* parameter generation algorithm $PPGen$ *on input* 1^τ *returns a public parameter pp.*
2. *The* key derivation function KD *takes as input the message* m *in the message space* $\mathcal{M}$ *and pp and produces as output message-derived key* k_m*.*
3. *The* encryption algorithm Enc *takes as input pp, the key* k_m*, and the message* m *in* $\mathcal{M}$*, and produces as output the ciphertext c.*
4. *The* decryption algorithm Dec *takes as input pp, the key* k_m *and the ciphertext c and produces as output the message* m *or* $\perp$*.*
5. *The* validity-test $Valid$ *takes as input public parameters pp and a ciphertext c and outputs 1 if the ciphertext c is a valid ciphertext, and 0 otherwise.*

Security notions. The definitions below make use of some parameters that are functions of the security parameter. Specifically, $k = k(\tau)$ denoting min-entropy requirements over message sources, and $T = T(\tau)$ representing the number of blocks in the message.

Entropy. The min-entropy of a random variable X is defined as $H_\infty(X) = -log\ (max_x\ Pr[X = x])$. In other words, $H_\infty(X) = k$, if $max_x Pr[X = x] = 2^{-k}$. A k-source is a random variable X with $H_\infty(X) \geq k$. A $(k_1, \ldots, k_T)$-source is a random variable $\mathbf{X} = (X_1, \ldots, X_T\ )$ where each X_i is a k_i -source. A (T, k)-source is a random variable $\mathbf{X} = (X_1, \ldots, X_T)$ where, for each $i = 1, \ldots, T$, it holds that X_i is a k-source. Next, we recall the definitions of real-or-random encryption oracle, polynomial-size X-source adversary, and, for schemes that rely on random oracles, q-query X-source adversary.

[4] In [6] it has been proven that security against adaptive adversaries is (polynomially) equivalent to security against static adversaries.

[5] MLE definition also includes an equality algorithm. We omit it since it is not necessary for our goals.

Definition 4 ([1]). Real or Random encryption oracle. *The real-or-random encryption oracle,* RoR*, takes as input triplets of the form* $(\mathtt{mode}, \mathtt{pp}, \mathbf{M})$*, where* $\mathtt{mode} \in \{\mathtt{real}, \mathtt{rand}\}$*, pp denotes public parameters, and* $\mathbf{M}$ *is a polynomial size circuit representing a joint distribution over T messages. If* $\mathtt{mode} = \mathtt{real}$ *then the oracle samples* $(m_1, \ldots, m_T) \leftarrow \mathbf{M}$*, and if* $\mathtt{mode} = \mathtt{rand}$ *then the oracle samples uniform and independent messages* $(m_1, \ldots, m_T) \leftarrow \mathcal{M}$*. Next, for each* $i = 1, \ldots, T$*, it samples* $k_i \leftarrow KD(pp, m_i)$*, computes* $c_i \leftarrow Enc(pp, k_i, m_i)$ *and outputs the ciphertext vector* $(c_1, \ldots, c_T)$*.*

Definition 5 ([1]). Poly-sampling complexity adversary. *Let* $\mathcal{A}$ *be a probabilistic polynomial-time algorithm that is given as input a pair* $(1^\tau, pp)$ *and oracle access to* $(1^\tau, pp)$ *for some* $\mathtt{mode} \in \{\mathtt{real}, \mathtt{rand}\}$*. Then,* $\mathcal{A}$ *is a polynomial-size* (T, k)*-source adversary if for each of* $\mathcal{A}$*'s* RoR*-queries* $\mathbf{M}$ *it holds that* $\mathbf{M}$ *is an* (T, k)*-source that is samplable by a circuit of (an arbitrary) polynomial size in the security parameter.*

Definition 6 ([1]). PRV-CDA2 security[6]**.** *An MLE scheme* $\Pi = (PPGen, KD, Enc, Dec, EQ, Valid)$ *is* (T, k)*-source* PRV-CDA2 *secure, if for any probabilistic polynomial-time polynomial-size* (T, k)*-source adversary* $\mathcal{A}$*, there exists a negligible function* $\epsilon(\tau)$ *such that the advantage of* $\mathcal{A}$ *is defined as*

$$\mathbf{Adv}^{\mathtt{PRV-CDA2}}_{\mathcal{A}}(1^\tau) = |Pr[\mathbf{Exp}^{\mathtt{real}}_{\mathcal{A}}(1^\tau) = 1] - Pr[\mathbf{Exp}^{\mathtt{rand}}_{\mathcal{A}}(1^\tau) = 1]| \leq \epsilon(\tau)$$

where for each $\mathtt{mode} \in \{\mathtt{real}, \mathtt{rand}\}$ *the experiment* $\mathbf{Exp}^{\mathtt{mode}}_{\mathcal{A}}(\tau)$ *is defined in the following game:*

Experiment $\mathbf{Exp}^{\mathtt{PRV-CDA2}}_{\mathcal{A}}$
  $pp \leftarrow PPGen(1^\tau)$
  return $\mathcal{A}^{\mathtt{RoR}(\mathtt{mode},\mathtt{pp},\cdot)}(1^\tau, pp)$

3 Verifiable Hierarchical Key Assignment Schemes

In this section, we introduce a novel cryptographic primitive that we call *Verifiable Hierarchical Key Assignment Scheme* (VHKAS). A VHKAS is a hierarchical key assignment scheme equipped with a verification procedure that allows honest users to check whether the public information has been maliciously changed.

A VHKAS for a family Γ of graphs, corresponding to partially ordered hierarchies, is defined as follows:

Definition 7. *A* VHKAS *is a triple* (Gen, Der, Ver) *of algorithms satisfying the following conditions:*

1. *The* information generation algorithm *Gen is probabilistic polynomial-time defined as in Definition 1.*
2. *The* key derivation algorithm *Der is deterministic polynomial-time defined as in Definition 1.*

[6] Notice that PRV-CDA2 notion enables adversaries to query the oracle with message distributions that depend on the public parameters *pp*.

3. *The* verification algorithm Ver *is deterministic polynomial-time. It takes as input the security parameter* 1^τ*, a graph* $G = (V, E)$ *in* Γ*, a class* u *in* V*, the private information* s_u*, a public information* pub *and it outputs* 1 *if for each class* $v \in V$ *such that* $v \preceq u$*,* $Der(1^\tau, G, u, v, s_u, pub)$ *return a valid key for the class* v*,* 0 *otherwise.*

Security Notions. In order to capture a notion of security against an adversary who has the ability to replace or modify the public information so as to mislead an honest user into deriving an incorrect key, we introduce the notion of *Strong Key-Consistency* (Strong-KC). Specifically, an adversary SSTAT is able to generate the secret information s, the set of keys k, and two different public values pub and pub' in such a way that, given a class u, the key of some class $v \preceq u$ derived by a user in class u according to pub differs from that derived according to pub' while the verification procedure succeeds in both cases. This notion models the fact that even the TA is not able to maliciously modify public information once private information has been distributed.

Now, we formally define the notion of Strong-KC:

Definition 8. [Strong-KC] *Let* Γ *be a family of graphs corresponding to partially ordered hierarchies, let* $G = (V, E)$ *be a graph in* Γ*, let* (Gen, Der, Ver) *be a VHKAS for* Γ *and let* SSTAT *be an adversary. Consider the following experiment:*

Experiment $\mathbf{Exp}^{\texttt{Strong-KC}}_{\texttt{SSTAT}}(1^\tau, G)$
  $(s, k, pub, pub', u) \leftarrow \texttt{SSTAT}(1^\tau, G)$
  if $(Ver(1^\tau, G, u, s_u, pub) = 0 \vee Ver(1^\tau, G, u, s_u, pub') = 0)$ **return** 0
  for each $v \preceq u$
    if $(Der(1^\tau, G, u, v, s_u, pub) \neq Der(1^\tau, G, u, v, s_u, pub'))$
      return 1
  return 0

The advantage of SSTAT *is defined as* $\mathbf{Adv}^{\texttt{Strong-KC}}_{\texttt{SSTAT}}(1^\tau) = |Pr[\mathbf{Exp}^{\texttt{Strong-KC}}_{\texttt{SSTAT}}(1^\tau) = 1]|$*. The scheme is* Strong-KC secure *if, the function* $\mathbf{Adv}^{\texttt{Strong-KC}}_{\texttt{SSTAT}}(1^\tau)$ *is negligible, for each adversary* SSTAT *whose time complexity is polynomial in* τ*.*

Remark 1. To help understand the extra feature provided by a VHKAS over an HKAS, in the following we consider the Encryption-Based Construction (EBC) proposed in [20] and show an example of how an attacker who manages to modify the public values is able to induce honest users to derive an incorrect key. In the EBC every class u is assigned a private information s_u, a secret key k_u, and a public information $\pi(u, u)$, which is the encryption of the key k_u with the private information s_u; furthermore, for each edge (u, v), there is a public value $p(u, v)$, which allows class u to compute the private information s_v held by class v. Indeed, $p(u, v)$ consists of encrypting the private information s_v with the private information s_u. This allows any user of a class u to compute the key k_v held by any class v lower down in the hierarchy.

The public value $\pi(u, u)$ can be considered as being associated with an additional edge connecting the class u to a dummy class u'. The figure 1 illustrates

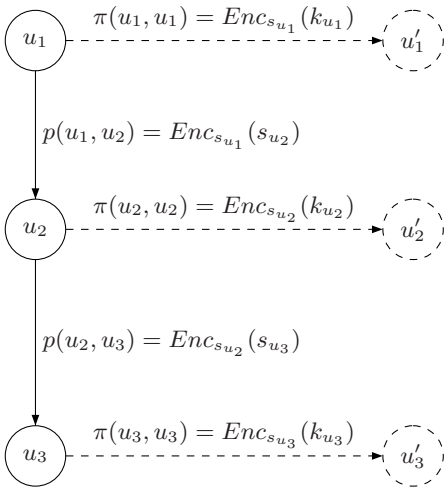

Fig. 1: A chain of length 3 along with the public information generated by the EBC.

a chain of length $t = 3$ together with the public information associated by the EBC with the edges of the chain, as well as that associated with the additional edges, which are represented by dashed lines.

A malicious user belonging to the class u_2 who manages to modify the public value $\pi(u_2, u_2)$ is able to induce honest users of u_2 and u_1 to derive a secret key k of his choice instead of the real key k_{u_2} associated with the class u_2. Indeed, such a malicious user, as part of the class u_2, holds the private information s_{u_2} and can use it to compute the value $Enc_{s_{u_2}}(k)$ which can then be substituted for the public value $\pi(u_2, u_2)$ associated with the edge (u_2, u'_2).

Note that even if the TA digitally signs the public values, the EBC construction does not achieve the notion of Strong-KC. Indeed, the adversary could reuse old public values and induce honest users to derive old keys. Furthermore, digitally signing public values does not prevent scenarios where the TA is involved in maliciously modifying public information.

A VHKAS will be able to withstand such attacks because public and private information will be crafted in such a way that any malicious changes to public values will be identified through the verification procedure.

4 An MLE-based Construction

In this section, we present a VHKAS which uses as a building block an MLE scheme. The scheme assumes that the partially ordered hierarchy has been partitioned into chains (i.e. totally ordered sets) [13, 22]. This gives a method of constructing a HKAS, represented by a directed acyclic graph $G = (V, E)$, from a HKAS for a simple chain by partitioning the poset into chains. This approach

Let Γ be a family of graphs corresponding to chains. Let $G = (V, E) \in \Gamma$ and let $\Pi = (PPGen, KD, Enc, Dec, Valid)$ be an MLE scheme whose key derivation function KD is collision-resistant and let $\mathcal{F} : \{0,1\}^\tau \times \{0,1\}^\tau \to \{0,1\}^\tau$ be a PRF.

Algorithm $Gen(1^\tau, G)$

1. Let $u_1, \ldots, u_t$ be the classes in the chain.
2. Let $pp \leftarrow PPGen(1^\tau)$;
3. Let $\pi_{u_{t+1}} \leftarrow \{0,1\}^\tau$;
4. For each class u_i, for $i = t, \ldots, 1$, let
   (a) $r_{u_i} \leftarrow \{0,1\}^\tau$;
   (b) $\pi_{u_i} \leftarrow KD(pp, \pi_{u_{i+1}} || r_{u_i})$;
   (c) $k_{u_i} \leftarrow \mathcal{F}(\pi_{u_i}, r_{u_i})$;
   (d) $s_{u_i} = (pp, r_{u_i}, \pi_{u_i}, k_{u_i})$;
5. Let s and k be the sequences of private information $s_{u_1}, s_{u_2}, \ldots, s_{u_t}$ and keys $k_{u_1}, k_{u_2}, \ldots, k_{u_t}$, respectively, computed in the previous steps;
6. For each $i = 1, \ldots, t$, compute the public information

$$p_{(u_i, u_{i+1})} \leftarrow Enc(pp, \pi_{u_i}, \pi_{u_{i+1}} || r_{u_i});$$

7. Let pub be the sequence of public information $p_{(u_1,u_2)}, p_{(u_2,u_3)}, \ldots, p_{(u_t,u_{t+1})}$ computed in the previous step;
8. Output(*s, k, pub*).

Algorithm $Der(1^\tau, G, u, v, s_u, pub)$

1. Let $u = u_i$ and $v = u_j$, for some $j \geq i$, $j = 1. \ldots, t$.
2. Parse s_u as $(pp, r_{u_i}, \pi_{u_i}, k_{u_i})$;
3. For any $z = i, \ldots, j$, extract the public value $p_{(u_z, u_{z+1})}$ from pub
   (a) if $Valid(pp, p_{(u_z,u_{z+1})}) = 0$, return $\perp$;
   (b) compute

$$(\pi_{u_{z+1}} || r_{u_z}) \leftarrow Dec(pp, \pi_{u_z}, p_{(u_z,u_{z+1})});$$

4. Output $k_v \leftarrow \mathcal{F}(\pi_{u_j}, r_{u_j})$.

Algorithm $Ver(1^\tau, G, u, s_u, pub)$

1. Let $u = u_i$, for some $i = 1, \ldots, t$.
2. Parse s_u as $(pp, r_{u_i}, \pi_{u_i}, k_{u_i})$;
3. For each $j = i, \ldots, t$,
   (a) extract the public value $p_{(u_j,u_{j+1})}$ from pub;
   (b) if $Valid(pp, p_{(u_j,u_{j+1})}) = 0$, return 0;
   (c) let $\alpha_i = \pi_{u_i}$, compute

$$(\alpha_{j+1} || \beta_j) \leftarrow Dec(pp, \alpha_j, p_{(u_j,u_{j+1})});$$

   (d) if $KD(pp, \alpha_{j+1} || \beta_j)$ is different from α_j, return 0;
4. return 1.

Fig. 2: The MLE-based Construction.

has the nice property that the amount of private storage needed per class, is bounded by the width of the poset[7]. Thus, in the following we will only consider a family Γ of graphs corresponding to chains. Figure 2 shows the MLE-based construction for a chain of t classes $u_1, \ldots, u_t$. In order to simplify the presentation, we consider a dummy class u_{t+1}. This will enable us to consider all public information as values associated to the edges of a chain.

The public information, associated to the edges of the chain, is used to store the secret keys in an encrypted form. Indeed, for each $i = 1, \ldots, t$, $k_{u_i} \leftarrow \mathcal{F}(\pi_{u_i}, r_{u_i})$ can be obtained by retrieving π_{u_i} and r_{u_i}, respectively from $p_{(u_{i-1},u_i)}$ and $p_{(u_i,u_{i+1})}$. Figure 3 illustrates the public information computed by the scheme for a chain of length $t = 3$. The verification procedure allows to check whether a

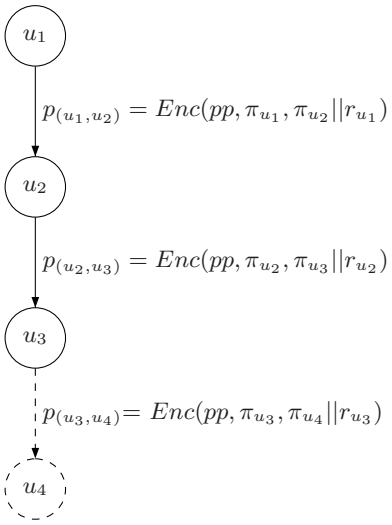

Fig. 3: Public information generated by the MLE-based construction for a chain of length $t = 3$.

honest user is able to derive valid keys. Specifically, in the MLE-based construction a honest user in some class u_i will derive valid keys $k_{u_j} = Der(1^\tau, G, u_i, u_j, s_{u_i}, pub)$, if for each $j = i, \ldots, t$, the public information $p_{(u_j,u_{j+1})}$ stores a value x such that $\pi_{u_j} = KD(pp, x)$. Intuitively, any changes to public values will be detected since KD is collision resistant. For example, if a malicious user belonging to class u_1 (see figure 3) changes $p(u_1, u_2)$ in such a way that honest users in u_1

[7] The width is the cardinality of the largest antichain in V. $A \subseteq V$ is an antichain in V if for all $u, v \in A$, where $u \neq v$, we have $v \preceq u$ and $u \preceq v$.

derive a value $\pi'_{u_2} \neq \pi_{u_2}$, the verification procedure fails since $KD(pp, \pi'_{u_2}||r_{u_1})$ is different from π_{u_1}.

4.1 Analysis of the Scheme

In this section we show that the security of the MLE-based construction depends upon the security properties of the underlying MLE scheme.

We first show that the MLE-based construction is secure in the sense of IND-ST.

Theorem 1. *Let $\Pi = (PPGen, KD, Enc, Dec, Valid)$ be a (T, μ)-source* PRV-CDA2 *secure MLE scheme where $\mu = \omega(\log \tau)$ and let $\mathcal{F} : \{0,1\}^\tau \times \{0,1\}^\tau \rightarrow \{0,1\}^\tau$ be a PRF. The MLE-based VHKAS of Figure 2 is secure in the sense of* IND-ST.

Proof. Let $\mathtt{STAT}_u$ be a static adversary attacking class u. Let $V = \{u_1, \ldots, u_t\}$ and $(u_i, u_{i+1}) \in E$, for $i = 1, \ldots, t-1$, and, w.l.o.g., let $u = u_j$ for some $1 \leq j < t$. In order to prove the theorem, we need to show that the adversary's views in experiments $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_u}$ and $\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT}_u}$ are indistinguishable. Notice that the only difference between $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_u}$ and $\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT}_u}$ is the last input of $\mathtt{STAT}_u$, which corresponds to the key k_u in the former experiment and to a random value in the latter. Thus, while in $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_u}$ the public information is related to the last input of $\mathtt{STAT}_u$, in $\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT}_u}$ it is completely independent on such a value.

We construct a sequence of 4 experiments $\mathbf{Exp}^1_u, \ldots, \mathbf{Exp}^4_u$, all defined over the same probability space, where the first and the last experiments of the sequence correspond to $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_u}$ and $\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT}_u}$, respectively. In each experiment we modify the way the view of $\mathtt{STAT}_u$ is computed, while maintaining the view's distributions indistinguishable among any two consecutive experiments. For any $q \in \{2, 3\}$, experiment $\mathbf{Exp}^q_u$ is defined as follows:

Experiment $\mathbf{Exp}^q_u(1^\tau, G)$
  $(s, k, pub^q) \leftarrow Gen^q(1^\tau, G)$
  $corr \leftarrow Corrupt_u(s)$
  return $\mathtt{STAT}_u(1^\tau, G, pub^q, corr, \alpha_u)$

The algorithm Gen^q used in $\mathbf{Exp}^q_u(1^\tau, G)$ differs from Gen for the way part of the public information pub^q is computed. Indeed, for any $i = 1, \ldots, j$, the public values associated to the edge (u_i, u_{i+1}) is computed as the encryption $Enc(pp, \pi_i, \gamma_{u_{i+1}}||r_{u_i})$ where $\gamma_{u_{i+1}}$ and r_{u_i} are random values in $\{0,1\}^\tau$ and $\pi_i \leftarrow KD(pp, \gamma_{u_{i+1}}||r_{u_i})$. Moreover, $\mathbf{Exp}^2_u(1^\tau, G)$ differs from $\mathbf{Exp}^3_u(1^\tau, G)$ for the way α_{u_j} is constructed. Specifically, $\alpha_{u_j} \leftarrow \mathcal{F}(\gamma_{u_j}, r_{u_j})$ in $\mathbf{Exp}^2_u$ while $\alpha_{u_j} \leftarrow \{0,1\}^\tau$ in $\mathbf{Exp}^3_u$.

Now we show that, the adversary's view in $\mathbf{Exp}^1_u$ is indistinguishable from the adversary's view in $\mathbf{Exp}^2_u$. Assume by contradiction that there exists a polynomial-time adversary A which is able to distinguish between the adversary $\mathtt{STAT}_u$'s views in experiments $\mathbf{Exp}^1_u$ and $\mathbf{Exp}^2_u$ with non-negligible advantage. We show how to construct a polynomial-time distinguisher D which uses A to break the security of the (T, μ)-source PRV-CDA2 MLE scheme.

Let $\Pi = (PPGen, KD, Enc, Dec, Valid)$ be an MLE scheme.

Algorithm for M

On input the public parameter pp output by $PPGen$, and a τ-length value π, produces T messages $m_1, \ldots, m_T$ as follows:

- choose $r_T \in \{0,1\}^\tau$;
- compute $m_T \leftarrow \pi || r_T$;
- compute $\pi_T \leftarrow KD(pp, \pi || r_T)$;
- for $i = 1, \ldots, T-1$, choose $r_i \in \{0,1\}^\tau$;
- for $i = 1, \ldots, T-1$, compute $\pi_i \leftarrow KD(pp, \pi_{i+1} || r_i)$;
- for $i = 1, \ldots, T-1$, compute $m_i \leftarrow \pi_{i+1} || r_i$.

Fig. 4: Polynomial size circuit $\mathbf{M}$ representing a joint distribution over T messages.

In particular, the distinguisher D constructs the public values associated to the edges (u_i, u_{i+1}), for $i = 1, \ldots, j$, calling the $\texttt{RoR}$ oracle where $\mathbf{M}$ implements the circuit of Figure 4 with $\pi = \pi_{u_{j+1}}$ and $T = j$. Notice that $\mathbf{M}$ is (T, μ)-source. Indeed, since r_{u_i} is chosen at random by $\mathbf{M}$, it is easy to see that the min-entropy of m_i is at least τ, for each $i = 1, \ldots, T$. Distinguisher D is defined in Figure 5.

Notice that if $\texttt{mode} = \texttt{real}$, then $\texttt{STAT}_\texttt{u}$'s view is that of $\mathbf{Exp}_u^1(1^\tau, G)$ while when $\texttt{mode} = \texttt{rand}$, $\texttt{STAT}_\texttt{u}$'s view is that of $\mathbf{Exp}_u^2(1^\tau, G)$. Therefore, if the algorithm A is able to distinguish between such views with non negligible advantage, it follows that D is able to break the $\texttt{PRV-CDA2}$ security of the MLE scheme.

Algorithm $D^{\mathcal{ROR}(mode, pp, \cdot)}(1^\tau)$
- $\pi_{u_{t+1}} \leftarrow \{0,1\}^\tau$
- $corr \leftarrow \emptyset$
- **for** each $i = t, \ldots, j+1$
  - $r_{u_i} \leftarrow \{0,1\}^\tau$
  - $\pi_{u_i} \leftarrow KD(pp, \pi_{u_{i+1}} || r_{u_i})$
  - $k_{u_i} \leftarrow \mathcal{F}(\pi_{u_i}, r_{u_i})$
  - $p_{(u_i, u_{i+1})} \leftarrow Enc(pp, \pi_{u_i}, \pi_{u_{i+1}} || r_{u_i})$
  - $corr \leftarrow corr \cup (\pi_{u_i}, k_{u_i}, r_{u_i})$
- Let $c_i = p_{(u_i, u_{i+1})}$, for $i = 1, \ldots, j$
- $(c_1, \ldots, c_j) \leftarrow \texttt{RoR}(\texttt{mode}, \texttt{pp}, \texttt{M}(\pi_{u_{j+1}}, j))$
- $b \leftarrow A(1^\tau, G, pub, corr, k_{u_j})$

Algorithm $D'^{f(\cdot)}(1^\tau)$
- $\pi_{u_{t+1}} \leftarrow \{0,1\}^\tau$
- **for** each $i = t, \ldots, j+1$
  - $r_{u_i} \leftarrow \{0,1\}^\tau$
  - $\pi_{u_i} \leftarrow KD(pp, \pi_{u_{i+1}} || r_{u_i})$
  - $k_{u_i} \leftarrow \mathcal{F}(\pi_{u_i}, r_{u_i})$
  - $p_{(u_i, u_{i+1})} \leftarrow Enc(pp, \pi_{u_i}, \pi_{u_{i+1}} || r_{u_i})$
- $corr \leftarrow (\pi_{u_{j+1}}, k_{u_{j+1}}, r_{u_{j+1}})$
- **for** each $i = j, \ldots, 1$
  - $r_{u_i} \leftarrow \{0,1\}^\tau$
  - $\pi_{u_i} \leftarrow \{0,1\}^\tau$
  - $p_{(u_i, u_{i+1})} \leftarrow Enc(pp, \pi_{u_i}, \pi_{u_{i+1}} || r_{u_i})$
- $\alpha_{u_j} \leftarrow f(r_{u_j})$
- $b \leftarrow B(1^\tau, G, pub, corr, \alpha_{u_j})$

Fig. 5: Distinguishers D and D'.

Now we show that, the adversary's view in $\mathbf{Exp}_u^2$ is indistinguishable from the adversary's view in $\mathbf{Exp}_u^3$. Assume by contradiction that there exists a polynomial-time algorithm B which is able to distinguish between the adversary $\mathtt{STAT}_u$'s views in experiments $\mathbf{Exp}_u^2$ and $\mathbf{Exp}_u^3$ with non-negligible advantage. We show how to construct a polynomial-time distinguisher D' which uses B to distinguish whether its oracle $f(\cdot)$ corresponds to the pseudorandom function $\mathcal{F}(k,\cdot)$ or to a random function $\mathcal{F}(\cdot)$.

Notice that if α_{u_j} corresponds to the evaluation of the pseudorandom function $\mathcal{F}(k,\cdot)$ on r_{u_j} then $\mathtt{STAT}_\mathtt{u}$'s view is that of $\mathbf{Exp}_u^2(1^\tau, G)$ while when it is the output of a random value, $\mathtt{STAT}_\mathtt{u}$'s view is that of $\mathbf{Exp}_u^3(1^\tau, G)$. Therefore, if the algorithm B is able to distinguish between such views with non negligible advantage, it follows that the distinguisher D' is able to break the pseudorandomness of $\mathcal{F}$. Figure 5 defines distinguisher D'.

Algorithm $D'^{\mathcal{ROR}(mode,pp,\cdot)}(1^\tau)$
- $\pi_{u_{t+1}} \leftarrow \{0,1\}^\tau$
- $corr \leftarrow \emptyset$
- **for** each $i = t, \ldots, j+1$
  - $r_{u_i} \leftarrow \{0,1\}^\tau$
  - $\pi_{u_i} \leftarrow KD(pp, \pi_{u_{i+1}} || r_{u_i})$
  - $k_{u_i} \leftarrow \mathcal{F}(\pi_{u_i}, r_{u_i})$
  - $p_{(u_i,u_{i+1})} \leftarrow Enc(pp, \pi_{u_i}, \pi_{u_{i+1}} || r_{u_i})$
  - $corr \leftarrow corr \cup (\pi_{u_i}, k_{u_i}, r_{u_i})$
- Let $c_i = p_{(u_i,u_{i+1})}$, for $i = 1, \ldots, j$
- $(c_1, \ldots, c_j) \leftarrow \mathtt{RoR}(\mathtt{mode}, \mathtt{pp}, \mathtt{M}(\pi_{u_{j+1}}, j))$
- $\rho \leftarrow \{0,1\}^\tau$
- $b \leftarrow C(1^\tau, G, pub, corr, \rho)$

Fig. 6: Distinguisher D''.

We finally show that, the adversary's view in $\mathbf{Exp}_u^4$ is indistinguishable from the adversary's view in $\mathbf{Exp}_u^3$.

Assume by contradiction that there exists a polynomial-time algorithm C which is able to distinguish between the adversary $\mathtt{STAT}_u$'s views in experiments $\mathbf{Exp}_u^4$ and $\mathbf{Exp}_u^3$ with non-negligible advantage.

Notice that such views differ only for the the public values associated to the edges (u_i, u_{i+1}) for $i = 1, \ldots, j$. We show how to construct a polynomial-time distinguisher D'' which uses C to break the PRV-CDA2 security of the MLE scheme. In particular, the algorithm D'', on input 1^τ, constructs the public values associated to the edges (u_i, u_{i+1}), for $i = 1, \ldots, j$, calling the RoR oracle where $\mathbf{M}$ implements the circuit of Figure 4 with $\pi = \pi_{u_{j+1}}$ and $T = j$.

Formally, distinguisher D'' is defined in Figure 6. Notice that if $\mathtt{mode} = \mathtt{real}$, then $\mathtt{STAT}_\mathtt{u}$'s view is that of $\mathbf{Exp}_u^4$ while when $\mathtt{mode} = \mathtt{rand}$, $\mathtt{STAT}_\mathtt{u}$'s view is that of $\mathbf{Exp}_u^3$.

Thus, if C distinguishes such views with non negligible advantage, it follows that algorithm D'' breaks the PRV-CDA2 security of the MLE scheme. □

We now show that the MLE-based construction is secure in the sense of Strong-KC.

Theorem 2. *Let $\Pi = (PPGen, KD, Enc, Dec, Valid)$ be an MLE scheme whose key derivation function KD is collision-resistant. The MLE-based VHKAS of Figure 2 is secure in the sense of* Strong-KC.

Sketch of the Proof. Let $V = \{u_1, \ldots, u_t\}$ and $(u_i, u_{i+1}) \in E$, for $i = 1, \ldots, t-1$. We show by contradiction that if there exists a static adversary SSTAT whose advantage $\mathbf{Adv}^{\text{Strong-KC}}_{\text{SSTAT}}$ is non-negligible, then there exists a PPT adversary $\mathcal{A}$ such that $Pr[(x_0, x_1) \leftarrow \mathcal{A}(1^\tau, KD(pp, \cdot)) : x_0 \neq x_1 \wedge KD(pp, x_0) = KD(pp, x_1)]$ is non-negligible.

The adversary SSTAT first produces two public values pub and pub' along with the private information. Then, it chooses a class $u \in V$ such that $Ver(1^\tau, G, u, s_u, pub) = Ver(1^\tau, G, u, s_u, pub') = 1$ while there exists a class $v \prec u$ where $Der(1^\tau, G, u, v, s_u, pub) = k_v$, $Der(1^\tau, G, u, v, s_u, pub') = k'_v$ and $k_v \neq k'_v$.

W.l.o.g., let $u = u_i$ and $v = u_j$, for some $1 \leq i < j \leq t$ where j is the smallest index such that $k_{u_j} = \mathcal{F}(\pi_{u_j}, r_{u_j})$ is different from $k'_{u_j} = \mathcal{F}(\pi'_{u_j}, r'_{u_j})$.

We distinguish the following two cases:

1. $\pi_{u_j} = \pi'_{u_j}$ and $r_{u_j} \neq r'_{u_j}$;
2. $\pi_{u_j} \neq \pi'_{u_j}$.

In the following, for each class u_s we will denote by $s_{u_z} = (pp, r_{u_s}, \pi_{u_s}, k_{u_s})$ and $s_{u'_z} = (pp, r'_{u_s}, \pi'_{u_s}, k'_{u_s})$, the private information computed by using the derivation procedure with respect to pub and pub'. Consider the first case. In order for the verification procedure to succeed on both pub and pub' it must be $\pi_{u_j} = KD(pp, \pi_{u_{j+1}} || r_{u_j}) = KD(pp, \pi'_{u_{j+1}} || r'_{u_j})$ where $\pi_{u_{j+1}}$ may or may not be equal to $\pi'_{u_{j+1}}$. Thus, the adversary $\mathcal{A}$ wins his game by exhibiting $x_0 = \pi_{u_{j+1}} || r_{u_j}$ and $x_1 = \pi'_{u_{j+1}} || r'_{u_j}$.

Now, consider the second case. In order for the verification procedure to succeed on both pub and pub' it must be

$$\begin{aligned}\pi_{u_i} &= KD(pp, \pi_{u_{i+1}} || r_{u_i}) \\ &= KD(pp, KD(pp, \pi_{u_{i+2}} || r_{u_{i+1}}) || r_{u_i}) \\ &= KD(pp, KD(\ldots (KD(pp, \pi_{u_j} || r_{u_{j-1}}) || r_{u_{j-2}}) \ldots) || r_{u_i})\end{aligned}$$

and

$$\begin{aligned}\pi_{u_i} &= KD(pp, \pi'_{u_{i+1}} || r'_{u_i}) \\ &= KD(pp, KD(pp, \pi'_{u_{i+2}} || r'_{u_{i+1}}) || r'_{u_i}) \\ &= KD(pp, KD(\ldots (KD(pp, \pi'_{u_j} || r'_{u_{j-1}}) || r'_{u_{j-2}}) \ldots) || r'_{u_i}).\end{aligned}$$

Since π_{u_j} is different from π'_{u_j} and $\pi_{u_i} = KD(pp, \pi_{u_{i+1}}||r_{u_i}) = KD(pp, \pi'_{u_{i+1}}||r'_{u_i})$ it holds that KD is not collision resistant, thus if SSTAT wins his game with non-negligible probability also $\mathcal{A}$ succeeds with overwhelming probability. □

4.2 A Concrete Instance

In Figure 7 we instantiate the scheme of Figure 8 with the deterministic MLE scheme $\Pi^{(q)}_{det}$ proposed in [1].

Let $G = (V, E) \in \Gamma$ (family of graphs corresponding to chains.) Let $(PPGen, KD, Enc, Dec, Valid)$ be the $\Pi^{(q)}_{det}$ scheme and let $\mathcal{F} : \{0,1\}^\tau \times \{0,1\}^\tau \to \{0,1\}^\tau$ be a PRF.

Algorithm $Gen(1^\tau, G)$

1. Let $u_1, \ldots, u_t$ be the classes in the chain.
2. Let $pp = (H_1, H_2, q) \leftarrow PPGen(1^\tau)$. Let $\pi_{u_{t+1}} \leftarrow \{0,1\}^\tau$;
3. For each class u_i, for $i = t, \ldots, 1$, let
   (a) $r_{u_i} \leftarrow \{0,1\}^\tau$;
   (b) $\pi_{u_i} \leftarrow KD(pp, \pi_{u_{i+1}}||r_{u_i}) = H_1(\pi_{u_{i+1}}||r_{u_i}||1) \oplus \ldots \oplus H_1(\pi_{u_{i+1}}||r_{u_i}||t+1)$
   (c) $k_{u_i} \leftarrow \mathcal{F}(\pi_{u_i}, r_{u_i})$; $s_{u_i} = (pp, r_{u_i}, \pi_{u_i}, k_{u_i})$;
4. Let s and k be the sequences of private information $s_{u_1}, s_{u_1}, \ldots, s_{u_t}$ and keys $k_{u_1}, k_{u_1}, \ldots, k_{u_t}$, respectively, computed in the previous step;
5. For each $i = 1, \ldots, t$, let $w_{u_{i+1}} = H_2(\pi_{u_{i+1}}||r_{u_i}||1) \oplus \ldots \oplus H_2(\pi_{u_{i+1}}||r_{u_i}||t+1)$, compute the public information $p_{(u_i, u_{i+1})} \leftarrow Enc(pp, \pi_{u_i}, \pi_{u_{i+1}}||r_{u_i}||w_{u_{i+1}})$.

Algorithm $Der(1^\tau, G, u, v, s_u, pub)$

1. Let $u = u_i$ and $v = u_j$, for some $j \geq i$, $j = 1, \ldots, t$;
2. For any $z = i, \ldots, j$, extract the public value $p_{(u_z, u_{z+1})}$ from pub
   (a) if $Valid(pp, p_{(u_z, u_{z+1})}) = 0$, return $\perp$;
   (b) otherwise, compute $(\pi_{u_{z+1}}||r_{u_z}||w_{u_{z+1}}) \leftarrow Dec(pp, \pi_{u_z}, p_{(u_z, u_{z+1})})$
3. Output $k_v \leftarrow \mathcal{F}(\pi_{u_j}, r_{u_j})$.

Algorithm $Ver(1^\tau, G, u, v, s_u, pub)$

1. Let $u = u_i$, for $i = 1, \ldots, t$;
2. For any $j = i, \ldots, t$,
   (a) extract the public value $p_{(u_j, u_{j+1})}$ from pub
   (b) if $Valid(pp, p_{(u_j, u_{j+1})}) = 0$, return 0;
   (c) let $\alpha_i = \pi_{u_i}$, compute $(\alpha_{j+1}||\beta_j||\gamma_{j+1}) \leftarrow Dec(pp, \alpha_j, p_{(u_j, u_{j+1})})$;
   (d) if $KD(pp, \alpha_{j+1}||\beta_j)$ is different from α_j, return 0;
3. return 1.

Fig. 7: An instance of the MLE-based Construction.

$\Pi_{det}^{(q)}$ uses as a building block a symmetric-key encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and two hash functions $H_1 : \{0,1\}^* \to \{0,1\}^\tau$ and $H_2 : \{0,1\}^* \to \{0,1\}^\rho$ with randomness length ρ. If $\mathcal{SE}$ is an IND-CPA secure scheme and H_1 and H_2 are modeled as random oracles, then, for any $T = poly(\tau)$ and any $k = \omega(log\tau)$, $\Pi_{det}^{(q)}$ is q-query (T,k)-source PRV-CDA2-secure.

The scheme $\Pi_{det}^{(q)} = (PPGen, KD, Enc, Dec, EQ, Valid)$ is defined as follows:

- **Parameter-generation algorithm**: On input 1^λ, the algorithm $PPGen$ chooses two hash functions $H_1 : \{0,1\}^* \to \mathcal{K}$ and $H_2 : \{0,1\}^* \to \{0,1\}^\rho$. It outputs the public parameters $pp = (H_1, H_2, q)$.

- **Key-derivation function:** The algorithm KD takes as input public parameters pp, the message m and outputs the message-derived key $k_m = H_1(m||1) \oplus H_1(m||2) \oplus \ldots \oplus H_1(m||q+1) \in \mathcal{K}$.

- **Encryption algorithm**: The algorithm Enc takes as input public parameters pp, a message m, and a message-derived key k_m. It computes $w_m = H_2(m||1) \oplus H_2(m||2) \oplus \ldots \oplus H_2(m||q+1)$ and outputs $\mathrm{E}_{k_m}(m||w_m) \in C$.

- **Validity test**: The algorithm $Valid$ outputs 1 on any input $c \in C$.

- **Decryption algorithm**: Dec takes as input public parameters pp, a ciphertext c, and a message-derived key k_m and outputs $m \leftarrow D_{k_m}(c)$.

- **Equality algorithm**:Algorithm EQ on input public parameters pp and ciphertexts c_1 and c_2 outputs 1 if and only if $c_1 = c_2$.

From the PRV-CDA2 security of $\Pi_{det}^{(q)}$ the instance of Figure 7 is secure in the random oracle model.

5 Handling Key Replacement

Cryptographic keys need to be periodically changed. Thus, a key assignment scheme should feature an efficient procedure for the TA to handle key replacements.

A VHKAS which handles key replacement is a tuple $(Gen, Der, Ver, KReplace)$, where (Gen, Der, Ver) is defined in Definition 7 and algorithm $KReplace$ satisfies the following conditions:

- The *key replacement algorithm* $KReplace$ is probabilistic polynomial-time. It takes as input 1^τ and a graph $G = (V, E)$ in Γ, a class u, the secret information s, the public information pub and produces as output (s, k, pub).

A key replacement procedure may require both public information and private values to be changed. Ideally, such a procedure will only change public information so that private values are not redistributed. In general, it is desirable to design the key replacement algorithm to modify as few private values as possible.

Figure 8 shows how the TA can handle the replacement of a key k_u for a class u in the MLE-based construction of Figure 2. In such a procedure only the classes higher than u in the chain are affected by the change.

Let (Gen, Der, Ver) be the scheme of Figure 2. The MLE-based scheme with key replacement is the tuple $(Gen, Der, Ver, KReplace)$ where $KReplace$ is as follows:
Algorithm $KReplace(1^\tau, G, u, s, pub)$

1. Let $u_1, \ldots, u_t$ be the classes in the chain.
2. Let $u = u_j$, for some $1 \leq j \leq t$;
3. For $i = j, \ldots, 1$, let
   (a) choose a new $r_{u_i} \leftarrow \{0,1\}^\tau$;
   (b) $\pi_{u_i} \leftarrow KD(pp, \pi_{u_{i+1}} || r_{u_i})$;
   (c) $p_{(u_i, u_{i+1})} \leftarrow Enc(pp, \pi_{u_i}, \pi_{u_{i+1}} || r_{u_i})$;
   (d) $k_{u_i} \leftarrow \mathcal{F}(\pi_{u_i}, r_{u_i})$.
4. Let s, k, and pub be the new sequences of private information, keys and public values, respectively;
5. Output(s, k, pub).

Fig. 8: MLE-based construction with key replacement.

In Figure 9 we describe the key replacement procedure when the MLE-based construction is instantiated with the deterministic MLE scheme $\Pi_{det}^{(q)}$ proposed in [1].

Algorithm $KReplace(1^\tau, G, u, s, pub)$

1. Let $u_1, \ldots, u_t$ be the classes in the chain.
2. Let $u = u_j$, for some $1 \leq j \leq t$;
3. Choose a new $r_{u_j} \leftarrow \{0,1\}^\tau$;
4. For $i = j, \ldots, 1$, let $w_{u_{i+1}} = H_2(\pi_{u_{i+1}} || r_{u_i} || 1) \oplus \ldots \oplus H_2(\pi_{u_{i+1}} || r_{u_i} || t+1)$
   (a) $\pi_{u_i} \leftarrow KD(pp, \pi_{u_{i+1}} || r_{u_i})$;
   (b) $p_{(u_i, u_{i+1})} \leftarrow Enc(pp, \pi_{u_i}, \pi_{u_{i+1}} || r_{u_i} || w_{u_{i+1}})$;
   (c) $k_{u_i} \leftarrow \mathcal{F}(\pi_{u_i}, r_{u_i})$.
5. Let s, k, and pub be the new sequences of private information, keys and public values, respectively;
6. Output(s, k, pub).

Fig. 9: The key replacement procedure for the instance of Figure 7.

6 Comparisons with Hierarchical Key Assignment Schemes

We evaluate the performance of our construction by comparing it with that of popular IND-ST HKAS from the literature having the feature that the public information is used to store encryption keys. The comparison shows that the performance of our construction is similar to that of such schemes despite the fact that it also achieves the notion of Strong-KC.

Figure 10 shows the comparison. The summary takes into account several parameters, such as the size of the public and private information, the number and the type of operations required by a class $u \in V$ to compute the key of a class v lower down in the hierarchy. Moreover, it specifies the notions of security achieved. In Figure 10, τ and τ_1 correspond respectively to the size of the secret key in symmetric encryption based constructions and in schemes obtained from factoring[8]. The value c is a constant depending on the underlying encryption scheme. For instance c is equal to 2 for the so called XOR construction in [7]. Notice that, to describe the parameters of the MLE-based construction we refer to the concrete instance described in Section 4.2 where the MLE considered uses as a building block a symmetric encryption scheme. Finally, w represents the width of the poset which corresponds to the number of chains in the partition.

Scheme	Public info.	Private info.	Key derivation	Security Notions
MLE-based §4	$\|E\|c\tau$	$\tau O(w)$	$Path(u,v)$ MLE decryptions +1 PRF eval.	IND-ST Strong-KC.
EBC [20]	$(\|E\|+\|V\|)c\tau$	τ	$Path(u,v)+1$ decryptions	IND-ST.
Freire et al. [21]	τ_1	$\tau_1 O(w)$	$Path(u,v)+1$ Modular squaring operations	IND-ST.
Atallah et al. [5]	$2\|E\|c\tau + \|V\|\tau$	τ	$2 \cdot Path(u,v)+1$ operations: -$Path(u,v)$ decryptions -$Path(u,v)+1$ PRF eval	IND-ST.

Fig. 10: Comparisons with Hierarchical Key Assignment Schemes.

In [18], De Capitani di Vimercati et al. proposed a data-outsourcing architecture which employs the HKAS in [5] for representing an authorization policy trough an equivalent encryption policy. The positive results provided by the experimental analysis performed in [18], are encouraging in order to evaluate the feasibility of our approach. Indeed, as shown above the performance of our MLE-based construction in terms of space required to store public information and key derivation operations are similar to that exhibited by the HKAS in [5].

[8] 1024-bit, 2048-bit, 3072-bit RSA keys are equivalent in strength to respectively, 80-bit, 112-bit, 128-bit symmetric keys.

Only, the users need to manage a bigger number of secrets. Also, the verification procedure should not add significant overhead, indeed, it consists of computing the value of a collision resistant function (step 3(c) of the Ver Algorithm of Figure 2) at each step of derivation beside the decryption operation (step 3(b) of Der Algorithm of Figure 2).

7 Conclusions

In this paper we have introduced Verifiable HKAS and have designed it using an MLE scheme as a building block. The security properties of our construction depends on those of the underlying MLE scheme. The concrete instance described in Section 4.2 achieves the notions `IND-ST` and `Strong-KC` in the random oracle model. Our proposal also manages with the replacement of compromised encryption keys by making a limited number of changes to public and private information. We leave as an interesting open problem that of building VHKAS secure in the standard model. Our solution produces a scheme which belongs to the family of IKEKAS, it would also be interesting to consider other HKAS families identified in [15] to construct VHKAS and study their performance.

Acknowledgements

The work of Ferrara and Ricciardi is partially supported by the project PON-ARS01 00860 titled *Ambient-intelligent Tele-monitoring and Telemetry for Incepting and Catering over hUman Sustainability - ATTICUS* funded by the Italian Ministry of Education and Research - RNA/COR 576347.

References

1. Abadi, M., Boneh, D., Mironov, I., Raghunathan, A., Segev, G.: Message-locked encryption for lock-dependent messages. In: Advances in Cryptology - CRYPTO 2013. Lecture Notes in Computer Science, vol. 8042, pp. 374–391. Springer (2013)
2. Akl, S.G., Taylor, P.D.: Cryptographic solution to a multilevel security problem. In: Advances in Cryptology: Proceedings of CRYPTO '82. pp. 237–249. Plenum Press, New York (1982)
3. Alderman, J., Farley, N., Crampton, J.: Tree-based cryptographic access control. In: ESORICS 2017 - 22nd European Symposium on Research in Computer Security. Lecture Notes in Computer Science, vol. 10492, pp. 47–64. Springer (2017)
4. Altaha, M., Muhajjar, R.: Lightweight key management scheme for hierarchical wireless sensor networks. pp. 139–147 (09 2017)
5. Atallah, M.J., Blanton, M., Fazio, N., Frikken, K.B.: Dynamic and efficient key management for access hierarchies. ACM Trans. Inf. Syst. Secur. **12**(3), 18:1–18:43 (2009)
6. Ateniese, G., De Santis, A., Ferrara, A.L., Masucci, B.: Provably-secure time-bound hierarchical key assignment schemes. J. Cryptology **25**(2), 243–270 (2012)
7. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: FOCS 1997. pp. 394–403 (1997)

8. Bellare, M., Keelveedhi, S., Ristenpart, T.: Message-locked encryption and secure deduplication. In: EUROCRYPT 2013,. Lecture Notes in Computer Science, vol. 7881, pp. 296–312. Springer (2013)
9. Berlato Stefano, Carbone Roberto, L.A.J., Silvio, R.: Exploring architectures for cryptographic access control enforcement in the cloud for fun and optimization. (ASIA CCS '20) (2020)
10. Castiglione, A., De Santis, A., Masucci, B., Palmieri, F., Castiglione, A., Huang, X.: Cryptographic hierarchical access control for dynamic structures. IEEE Trans. Information Forensics and Security **11**(10), 2349–2364 (2016)
11. Castiglione, A., De Santis, A., Masucci, B., Palmieri, F., Huang, X., Castiglione, A.: Supporting dynamic updates in storage clouds with the akl-taylor scheme. Inf. Sci. **387**, 56–74 (2017)
12. Chang, C.C., Hwang, R.J., Wu, T.C.: Cryptographic key assignment scheme for access control in a hierarchy. Information Systems **17**(3), 243 – 247 (1992)
13. Crampton, J., Daud, R., Martin, K.M.: Constructing key assignment schemes from chain partitions. In: Data and Applications Security and Privacy (IFIP). Lecture Notes in Computer Science, vol. 6166, pp. 130–145. Springer (2010)
14. Crampton, J., Farley, N., Gutin, G.Z., Jones, M., Poettering, B.: Cryptographic enforcement of information flow policies without public information. In: ACNS 2015. Lecture Notes in Computer Science, vol. 9092, pp. 389–408. Springer (2015)
15. Crampton, J., Martin, K.M., Wild, P.R.: On key assignment for hierarchical access control. In: 19th IEEE Computer Security Foundations Workshop, (CSFW-19 2006). pp. 98–111. IEEE Computer Society (2006)
16. D'Arco, P., De Santis, A., Ferrara, A.L., Masucci, B.: Variations on a theme by akl and taylor: Security and tradeoffs. Theor. Comput. Sci. **411**(1), 213–227 (2010)
17. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: A data outsourcing architecture combining cryptography and access control. In: CSAW 2007. pp. 63–69. ACM (2007)
18. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Encryption policies for regulating access to outsourced data. ACM Trans. Database Syst. **35**(2), 12:1–12:46 (2010)
19. De Santis, A., Ferrara, A.L., Masucci, B.: New constructions for provably-secure time-bound hierarchical key assignment schemes. Theor. Comput. Sci. **407**(1-3), 213–230 (2008)
20. De Santis, A., Ferrara, A.L., Masucci, B.: Efficient provably-secure hierarchical key assignment schemes. Theor. Comput. Sci. **412**(41), 5684–5699 (2011)
21. Freire, E.S.V., Paterson, K.G.: Provably secure key assignment schemes from factoring. In: Parampalli, U., Hawkes, P. (eds.) ACISP 2011. Lecture Notes in Computer Science, vol. 6812, pp. 292–309. Springer (2011)
22. Freire, E.S.V., Paterson, K.G., Poettering, B.: Simple, efficient and strongly ki-secure hierarchical key assignment schemes. In: Topics in Cryptology - CT-RSA 2013. vol. 7779, pp. 101–114. Springer (2013)
23. Zarandioon, S., Yao, D.D., Ganapathy, V.: K2C: cryptographic cloud storage with lazy revocation and anonymous access. In: 7th International ICST Conference, SecureComm 2011. g, vol. 96, pp. 59–76 (2011)
24. Zhu, W.T., Deng, R.H., Zhou, J., Bao, F.: Applying time-bound hierarchical key assignment in wireless sensor networks. In: 13th International Conference, ICICS 2011. vol. 7043, pp. 306–318 (2011)