CLUE: Certificateless Updatable Encryption

Jodie Knapp[1], Elizabeth A. Quaglia[2]

Royal Holloway, University of London

Abstract

We formalise *certificateless public-key updatable encryption* (CLUE), a primitive that has yet to be defined in the public-key updatable encryption (PKUE) literature. Traditionally, PKUE allows outsourcing ciphertext key rotation to an untrusted host using a special token such that the ciphertext is updated to a distinct period known as an epoch. Key to security, the host does not learn anything about the underlying plaintext. In practice, applying PKUE in a public key infrastructure (PKI) requires trust in a third party producing the epoch public and secret keys, which is a clear violation of privacy if the key generator behaves maliciously or is corrupted. In this paper, we are concerned with reducing the trust in the PKI key generator and our chosen solution is to formalise our novel CLUE primitive, from PKUE and certificateless public key encryption (CL-PKE) primitives, as well as a security framework for CLUE. Moreover, we modify the certificateless encryption scheme proposed by Libert et al. (PKC 2006) and demonstrate the provable security of our CLUE scheme. To do so, we follow the modular approach given by Klooß et al. (EUROCRYPT'19) to reduce the security analysis to the standard setting.

Keywords

Public-Key Cryptography, Updatable Encryption, Certificateless Encryption

1. Introduction

Introduced by [1], public-key updatable encryption (PKUE) is a primitive used by a data owner for the long-term storage of encrypted data. For security purposes, the primitive is designed with timely updates of ciphertexts using a key rotation element (token) such that the update process is outsourced to an untrusted server. Crucial to security, the server learns no information regarding the underlying data when equipped with these tokens.

In practice, the public-key infrastructure (PKI) in which PKUE will be used as a building block is a lot more involved than simply considering a data owner and a server. Traditionally, digital certificates are associated with the public and secret key pairs to authenticate the data owner (individual/organisation) in a public-key encryption scheme. An approach to simplifying the public key and certificate management in a PKI is identity-based encryption (IBE), a primitive introduced by [2], in which a unique identifier is attached to the cryptographic key(s). Realistically, in the IBE setting a data owner cannot generate the secret key associated with their unique identifier. Instead, they need to place trust in a key generation centre (KGC) to compute the secret key. Herein lies a problem if we were to use IBE to support a PKUE scheme, namely, security is no longer guaranteed if the KGC becomes corrupted or behaves dishonestly.

ITASEC 2023: The Italian Conference on CyberSecurity

jodie.knapp.2018@rhul.ac.uk (J. Knapp); elizabeth.quaglia@rhul.ac.uk (E. A. Quaglia)

0000-0002-5929-2015 (J. Knapp); 0000-0002-4010-773X (E. A. Quaglia)

CEUR Workshop Proceedings (CEUR-WS.org)

In more words, traditional IBE schemes rely on an arrangement in which the key needed to decrypt a ciphertext is held in escrow so that under certain circumstances, an authorised third party (KGC), can gain access to the secret keys of all users. Thus, if the KGC is corrupt, they can forge signatures on any message and decrypt the ciphertext without the consent of the users, which is a clear privacy issue. We observe that a corrupt KGC has even more power in a PKUE scheme since the epoch secret keys are incorporated into update tokens, meaning the KGC would be able to maliciously update ciphertexts as well as learn the underlying information. Therefore, we must consider a solution to the key escrow problem regarding PKUE, such that an *identifier is associated with an epoch* instead of a user. This is especially important given the sensitive nature of information encrypted in applications of a PKUE scheme.

Our chosen solution is to formalise a novel *certificateless* PKUE primitive that we dub CLUE. Intuitively, our new definition is a PKUE scheme such that the underlying standard encryption scheme is the certificateless public-key encryption (CL-PKE) primitive. First introduced by [3], CL-PKE is an alternative primitive to PKI-supported IBE used to remove the need for certificate management and tackle the *key escrow* problem inherent in traditional identity-based encryption (IBE) schemes [4, 2, 5], whilst continuing to benefit from the advantages of identity-based cryptography. In more detail, the KGC in a CL-PKE scheme generates a *partial secret key* that is distributed to the corresponding data owner who combines this cryptographic element with their own, randomly chosen secret value to generate the secret and public keys associated with their identity. In this way, the KGC does not learn the actual value of the secret key, which resolves the key escrow problem and is crucial to the security of a CL-PKE scheme. We defer to a discussion on related work in Appendix A.

To summarise, we deem CL-PKE to be a suitable candidate for a revised version of the PKUE primitive formally called certificateless public-key updatable encryption (CLUE). We do so with care considering both the security requirements of traditional CL-PKE (including inside and outside adversaries) and the intricacies of security modelling in PKUE arising from information inferred from corrupted tokens and epoch keys. We stress the CLUE primitive applies to any setting in which KGC generating cryptographic keys and the server performing updates are separate entities that cannot be trusted or instances where individuals want to reduce trust in the KGC. Therefore, our main motivation in defining CLUE is to support long-term outsourced storage in an environment with *reduced trust*, with the intent to preserve privacy on behalf of the data owner.

Contributions Our contributions are threefold: first, we introduce and formalise CLUE, a certificateless public-key updatable encryption primitive, in Section 2. Secondly, we define and model a new security notion (CLUE-IND-RCCA security) in Section 3 which captures the indistinguishability of freshly generated and updated ciphertexts. Next, we propose a concrete CLUE scheme in Section 4 which is an adaptation of the pairing-based CL-PKE scheme given in [6] to the updatable setting. Note, in Appendix C we present a sketch analysis that our construction provably satisfies CLUE-IND-RCCA security. We highlight that the long version of this work contains greater detail, including an efficiency analysis of our construction and full security proofs which we have omitted due to lack of space.

2. Certificateless Updatable Encryption

Notation In the following, we define a certificateless public-key *updatable* encryption primitive. The scheme is defined by epochs of time in which the keys, token and ciphertext are associated with a given epoch in time and the ciphertext update algorithm rotates the ciphertext to encryption under a new epoch key using the token. In line with the literature, we denote the current epoch as e, and use the subscript notation e_i if we define multiple epochs at once with the range of time $i = \{0, \ldots, \mathsf{max}\}$ such that e_{max} is the last epoch in the scheme. Further, (e_i, e_{i+1}) are two consecutive epochs for any $i \in \mathbb{N}$ and $\tilde{e}$ represents the challenge epoch in security games.

Definition 1 (CLUE). *Given n epochs identified by the space $\mathcal{IDSP}$, plus the message space $\mathcal{MSP}$, and ciphertext space $\mathcal{CSP}$, let a certificateless public-key updatable encryption scheme be a tuple of nine algorithms $\Pi_{CLUE} =$ (Setup, Partial-SK-Extract, Set-Secret-Value, Set-SK, Set-PK, Set-Token, Enc, Dec, Upd) defined as follows,*

- $\mathsf{Setup}(1^\lambda) \xrightarrow{\$} (pp, msk)$: The key generation centre (KGC) takes security parameter 1^λ as input and outputs public parameters pp and master secret key msk.
- $\mathsf{Partial\text{-}SK\text{-}Extract}(pp, msk, \mathrm{ID}_e) \rightarrow \mathrm{D}_e$: the KGC takes the public parameters pp, the master secret key msk and identity $\mathrm{ID}_e \in \mathcal{IDSP}$ for epoch e as input and outputs partial secret key D_e.[1]
- $\mathsf{Set\text{-}Secret\text{-}Value}(pp, e) \xrightarrow{\$} x_e$: the data owner takes the public parameters pp and the current epoch e that they are running the algorithm for as inputs and randomly chooses secret value x_e.
- $\mathsf{Set\text{-}SK}(pp, \mathrm{D}_e, x_e) \rightarrow sk_e$: the data owner takes the public parameters pp, partial secret key D_e and secret value x_e as inputs and computes their secret key sk_e.
- $\mathsf{Set\text{-}PK}(pp, x_e) \rightarrow pk_e$: the data owner takes the public parameters pp and secret value x_e as inputs and computes their public key pk_e.
- $\mathsf{Set\text{-}Token}(pp, (pk_e, sk_e)) \rightarrow \Delta_{e+1}$: the data owner takes the public parameters pp plus the current epoch public key and secret keys (pk_e, sk_e) as inputs and computes (for epoch identifier ID_e) the update token Δ_{e+1} to epoch $(e+1)$ which is sent to the server.
- $\mathsf{Enc}(pp, M, pk_e, \mathrm{ID}_e) \xrightarrow{\$} \{C_e, \perp\}$: the data owner takes the public parameters pp, message $M \in \mathcal{MSP}$, public key pk_e and identity ID_e as inputs and outputs the ciphertext $C \in \mathcal{CSP}$ for epoch e or failure symbol $\perp$ if public key pk_e does not have the correct form.
- $\mathsf{Dec}(pp, C_e, sk_e) \rightarrow \{M, \perp\}$: the data owner takes the public parameters pp, ciphertext C and secret key sk_e as inputs and outputs the message M or failure symbol $\perp$.
- $\mathsf{Upd}(pp, C_e, \Delta_{e+1}) \rightarrow \{C_{e+1}, \perp\}$: the server takes the public parameters pp, ciphertext C_e and update token Δ_{e+1} as inputs and outputs the updates ciphertext C_{e+1} for epoch $(e+1)$ or failure symbol $\perp$.

[1] This algorithm is run once for each epoch and the KGC distributes the partial secret keys to the data owner in a secure manner [7].

Informally, for a CLUE scheme to satisfy the property of correctness, we require that fresh and updated ciphertexts decrypt to the corresponding plaintext given the appropriate epoch key. The formal definition of CLUE correctness follows.

Definition 2 (Correctness). *Given security parameter $\lambda \in \mathbb{N}$, a certificateless updatable encryption scheme (Π_{CLUE}) formalised in Definition 1 is correct if, for any message $M \in \mathcal{MSP}$ and for any $j \in \{1, \ldots, max\}$, $i \in \{0, \ldots, max\}$ with $max > i$, there exists a negligible function negl such that the following holds with overwhelming probability.*

$$\Pr\left[\begin{array}{l}(pp, msk) \xleftarrow{\$} \mathsf{Setup}(1^\lambda);\\ \mathrm{D}_e \leftarrow \mathsf{Partial\text{-}SK\text{-}Extract}(pp, msk, \mathrm{ID}_e);\\ x_e \xleftarrow{\$} \mathsf{Set\text{-}Secret\text{-}Value}(pp, e); sk_e \leftarrow \mathsf{Set\text{-}SK}(pp, \mathrm{D}_e, x_e);\\ pk_e \leftarrow \mathsf{Set\text{-}PK}(pp, x_e); \Delta_{e_j} \leftarrow \mathsf{Set\text{-}Token}(pp, sk_e, x_{e+1});\\ C_{e_i} \xleftarrow{\$} \mathsf{Enc}(pp, M, pk_{e_i}, \mathrm{ID}_e);\\ \{C_{e_j} \leftarrow \mathsf{Upd}(pp, C_{e_{j-1}}, \Delta_{e_j}) : j \in \{i+1, \ldots, \mathsf{max}\}\};\\ \mathsf{Dec}(pp, C_{e_{\mathsf{max}}}, sk_{\mathsf{max}}) = M\end{array}\right] \geq 1 - \mathsf{negl}(1^\lambda).$$

3. Security Modelling for CLUE

Defining the security of a cryptographic primitive is often a complex process. For CLUE we want to combine the approach to security taken in CL-PKE with the intricacies of UE security modelling to capture the *indistinguishability* of ciphertexts deriving from fresh encryption and updates. The notion of security we settle on is CLUE-IND-RCCA (Definition 4) and we give an intuition of this notion in the full version of our work. Next, we provide an overview of the security experiment in which the adversary has access to oracles and the challenger records essential lists, both of which are key to capturing security given the challenging nuances of the update functionality in CLUE.

High-Level Idea We define ciphertext indistinguishability against replayable chosen ciphertext attacks for the CLUE primitive. This notion is formalised in Definition 1 through the security experiment $\mathsf{Exp}^{\mathsf{CLUE\text{-}IND\text{-}RCCA}}_{\Pi_{\mathsf{CLUE}},\mathcal{A}}(1^\lambda)$ given in Figure 2. Informally, the game is between a challenger and an adversary $\mathcal{A}$ such that the latter can query the oracles detailed in Figures 1. To win the experiment, $\mathcal{A}$ must distinguish the underlying message of the challenge ciphertext without possession of the corresponding epoch secret key, given only access to the relevant oracles and a challenge ciphertext. Security is satisfied if the adversary's advantage in succeeding is negligible, as detailed in Definition 4.

Lists To initialise the CLUE-IND-RCCA security experiment, the challenger runs $\mathsf{Init}(1^\lambda)$ which outputs the *global state* (GS) oracles have access to throughout. At the start, GS $:=$ $(pp, sk_0, pk_0, \Delta_0, \mathbf{L}, 0)$ contains the public parameters pp generated by the CLUE setup algorithm; epoch secret and public keys (sk_0, pk_0) respectively; initial update token $\perp \rightarrow \Delta_0$; set $\mathbf{L} := \{\mathcal{L}, \mathcal{M}^*, \mathcal{T}, \mathcal{K}, \mathcal{C}^*\}$ containing initially empty lists that the challenger is required to

maintain throughout the experiment in order to prevent $\mathcal{A}$ from trivially winning and setting the current epoch $0 \rightarrow e$. List $\mathcal{L}$ is maintained to keep a log of *updated versions* of *honestly-generated* ciphertexts, and the corresponding epoch, that the adversary learns through calls to the relevant oracle. List $\mathcal{M}^*$ tracks the challenge messages the adversary sends to the challenger. Further, list $\mathcal{T}$ records the epoch(s) in which the adversary has obtained an update token and $\mathcal{K}$ tracks the epoch(s) in which the adversary has obtained an epoch secret key or epoch *partial* secret key. List $\mathcal{C}$ tracks the epochs in which an adversary obtains an *updated version* of the challenge-ciphertext through querying the ciphertext update oracle. We must extend this list to capture additional information necessary to prevent an adversary from trivially winning in the security experiment for Definition 4. Following the approach taken in [8] to satisfy RCCA-security, this extension is recorded in $\mathcal{C}^*$ which is a list encapsulating all of the *challenge-equal epochs* in which the adversary knows a *version* of the challenge ciphertext since there are epochs in which the adversary can infer information independently including epochs belonging to lists $\mathcal{C}, \mathcal{T}$. Challenge-equal ciphertexts are defined by a recursive predicate challenge-equal as follows:

$$\mathcal{C}^* \leftarrow \{e \in \{0, \ldots, e_{\max}\} | \mathsf{challenge\text{-}equal}(e) = \mathsf{true}\} \text{ and } \mathsf{true} \leftarrow \mathsf{challenge\text{-}equal}(e) \text{ iff :}$$
$$(e \in \mathcal{C}) \vee (\mathsf{challenge\text{-}equal}(e-1) \wedge e \in \mathcal{T}) \vee (\mathsf{challenge\text{-}equal}(e+1) \wedge (e+1) \in \mathcal{T}).$$

To illustrate, if an adversary knows a ciphertext $\tilde{C}_e$ from challenge epoch e and update token Δ_{e+1}, then the adversary can manually update the ciphertext to the epoch $(e+1)$ and therefore infer $\tilde{C}_{e+1}$ [8]. To re-emphasise the importance of lists, winning conditions in the experiment from Figure 2 state that the intersection of epochs contained within lists $\mathcal{K}$ and $\mathcal{C}^*$ must be empty which is crucial in preventing the adversary from *winning trivially*. That is, the challenge epoch of the experiment cannot belong to the set of epochs in which an update token has been learned or inferred, nor can there exist a single epoch where the adversary knows both the epoch key pair and a version of the challenge-ciphertext.

Oracles Figure 1 provides formal descriptions of the initialisation phase a challenger runs and the oracles an adversary has access to during the security experiment for Definition 4. For clarity, we provide intuition and a definition of an important predicate utilised in UE security modelling to prevent trivial wins. Informally, to prevent the decryption of an updated challenge ciphertext, irrespective of whether the UE scheme is probabilistic or deterministic, a useful predicate defined in [8] can be utilised in the running of decryption and update oracles. Informally, the $\mathsf{isChallenge}(k_{e_i}, C)$ predicate detects any queries to the decryption and update oracles on challenge ciphertexts ($\tilde{C}$), or versions (i.e updated) of the challenge ciphertext.

Definition 3 (isChallenge Predicate [8]). *Given challenge epoch $\tilde{e}$ and challenge ciphertext $\tilde{C}$, the isChallenge predicate, on inputs of the current epoch key k_{e_i} and queried ciphertext C_{e_i}, responds in one of three ways:*

1. If $(e_i = \tilde{e}) \wedge (C_{e_i} = \tilde{C})$, return true;
2. If $(e_i > \tilde{e}) \wedge (\tilde{C} \neq \perp)$, return true if $\tilde{C}_{e_i} = C_{e_i}$ in which $\tilde{C}_{e_i}$ is computed iteratively by running $\mathsf{Upd}(pp, \Delta_{e_{l+1}}, \tilde{C}_{e_l})$ for $e_l = \{\tilde{e}, \ldots, e_i\}$;
3. Otherwise, return false.

$\underline{\mathsf{Init}(1^\lambda)}$
$(pp, msk) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
$\mathsf{ID}_0 \leftarrow \mathsf{Partial\text{-}SK\text{-}Extract}(pp, msk, \mathsf{ID}_0)$
`for a valid` $\mathsf{ID}_0 \in \mathcal{IDSP}$
$x_0 \xleftarrow{\$} \mathsf{Set\text{-}Secret\text{-}Value}(pp, 0)$
$sk_0 \leftarrow \mathsf{Set\text{-}SK}(pp, \mathsf{ID}_0, x_0)$
$pk_0 \leftarrow \mathsf{Set\text{-}PK}(pp, x_0)$
$\Delta_0 \leftarrow \perp$
$e \leftarrow 0$
$\mathbf{L} \in \emptyset$ `for the set of lists`
$\mathbf{L} := \{\mathcal{L}, \mathcal{M}^*, \mathcal{T}, \mathcal{K}, \mathcal{C}^*\}$
return GS
$\mathsf{GS} := (pp, sk_0, pk_0, \Delta_0, \mathbf{L}, 0)$
$\underline{\mathcal{O}_{\mathsf{Dec}}(C_e)}$
$M \leftarrow \mathsf{Dec}(pp, C_e, sk_e)$
if $(M \in \mathcal{M}^*) \vee (\mathsf{isChallenge}(k_e, C_e) = \mathsf{true})$ **then**
  return test
else
  return M
$\underline{\mathcal{O}_{\mathsf{Upd}}(C_{e_i})}$
for $e_j = \{e_{i+1}, \ldots, e\}$ **do**
  $C_{e_j} \leftarrow \mathsf{Upd}(pp, C_{e_i}, \Delta_{e_j})$
  $C_e \leftarrow C_{e_j}$
return C_e
$\mathcal{L} \leftarrow \mathcal{L} \cup \{(e, C_e)\}$
if $(\mathsf{Dec}(pp, C_e, sk_e) = M \in \mathcal{M}^*) \vee (\mathsf{isChallenge}(k_e, C_e) = \mathsf{true})$ **then**
  $\mathcal{C}^* \leftarrow \mathcal{C}^* \cup \{e\}$

$\underline{\mathcal{O}_{\mathsf{Next}}(e)}$
$x_{e+1} \xleftarrow{\$} \mathsf{Set\text{-}Secret\text{-}Value}(pp, e+1)$
$sk_{e+1} \leftarrow \mathsf{Set\text{-}SK}(pp, \mathsf{ID}_e, x_{e+1})$
$pk_{e+1} \leftarrow \mathsf{Set\text{-}PK}(pp, x_{e+1})$
$\Delta_{e+1} \leftarrow \mathsf{Set\text{-}Token}(pp, (pk_e, sk_e))$
`Update GS`
$(pp, sk_{e+1}, pk_{e+1}, \Delta_{e+1}, \mathbf{L}, e+1)$
if $(e \in \mathcal{K}) \vee ((e, C) \in \mathcal{L})$ **then**
  $(C', e+1) \xleftarrow{\$} \mathsf{Upd}(pp, \Delta_{e+1}, C)$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{(e+1, C')\}$
$\underline{\mathcal{O}_{\mathsf{Corrupt\text{-}Token}}(e^*)}$
if $e^* \geq e$ **then**
  return $\perp$
else
  return Δ_{e^*}
$\mathcal{T} \leftarrow \mathcal{T} \cup \{e^*\}$
$\underline{\mathcal{O}_{\mathsf{Corrupt\text{-}key}}(e^*)}$
if $e^* \geq e$ **then**
  return $\perp$
else
  return sk_{e^*}
$\mathcal{K} \leftarrow \mathcal{K} \cup \{e^*\}$
$\underline{\mathcal{O}_{\mathsf{PSKE}}(e^*)}$
if $((e^* \geq e) \vee (e^* \in \mathcal{K}))$ **then**
  return $\perp$
else
  return D_e
$\mathcal{K} \leftarrow \mathcal{K} \cup \{e^*\}$

Figure 1: Details of the initialisation phase run by the challenger and the oracles adversary $\mathcal{A}$ has access to during the security experiment of Definition 4.

Recall that the CL-PKE *adversarial model* focuses on two types of adversaries, namely, an outside and inside (honest but curious KGC) attacker. We explicitly define the oracles in the set $\mathcal{O}$ that these distinct adversaries possess during our security game. Explicitly, adversary $\mathcal{A}_I$ has *no access to the master secret key*, however, they have access to all of the oracles described above. Conversely, adversary $\mathcal{A}_{II}$ has *implicit access to a master secret key*, which means they can compute partial secret keys for their own use given the master secret key and therefore do not need access to oracle $\mathcal{O}_{\mathsf{PSKE}}$. Thus, $\mathcal{A}_{II}$ has access to the set $\mathcal{O} = \{\mathcal{O}_{\mathsf{Dec}}, \mathcal{O}_{\mathsf{Next}}, \mathcal{O}_{\mathsf{Upd}}, \mathcal{O}_{\mathsf{Corrupt\text{-}Token}}, \mathcal{O}_{\mathsf{Corrupt\text{-}Key}}\}$.

Definition 4 (CLUE-IND-RCCA Security). *A CLUE scheme following Definition 1 is*

$\underline{\mathsf{Exp}^{\text{CLUE-IND-RCCA},b}_{\Pi_{\text{CLUE}},\mathcal{A}}(1^\lambda)}$
`Initialise Global State`
$\mathsf{GS} \xleftarrow{\$} \mathsf{Init}(1^\lambda)$; $\mathsf{GS} = (pp, sk_0, pk_0, \Delta_0, \mathbf{L}, 0)$;
$\mathsf{ID}_e \leftarrow \mathsf{Partial\text{-}SK\text{-}Extract}(pp, msk, \mathsf{ID}_e)$ `for a valid epoch identity` $\mathsf{ID}_e \in \mathcal{IDSP}$
$x_e \xleftarrow{\$} \mathsf{Set\text{-}Secret\text{-}Value}(pp, e)$
$sk_e \leftarrow \mathsf{Set\text{-}SK}(pp, \mathsf{ID}_e, x_e)$
$pk_e \leftarrow \mathsf{Set\text{-}PK}(pp, x_e)$
$(M_0, M_1, s) \leftarrow \mathcal{A}^{\mathcal{O}}(pp, pk_e)$
`Some state information s`
if $|M_0| \neq |M_1| \vee \{M_0, M_1\} \notin \mathcal{MSP} \vee (M_0 = M_1)$ **then**
  return $\perp$
else
  $b \xleftarrow{\$} \{0, 1\}$,
  $C \xleftarrow{\$} \mathsf{Enc}(pp, M_b, pk_e, \mathsf{ID}_e)$,
  $\mathcal{M}^* \leftarrow \mathcal{M}^* \cup (M_0, M_1)$; $\mathcal{C} \leftarrow \mathcal{C} \cup \{e\}$;$\tilde{e} \leftarrow \{e\}$
$b' \leftarrow \mathcal{A}^{\mathcal{O}}(pp, C, s)$,
if $(\mathcal{K} \cap \mathcal{C}^* = \emptyset)$ **then**
  return b'
`Else abort.`

Figure 2: The security experiment for CLUE-IND-RCCA security of a CLUE scheme, where the set of lists is $\mathbf{L} := \{\mathcal{L}, \mathcal{M}^*, \mathcal{T}, \mathcal{K}, \mathcal{C}^*\}$ is initially empty, s defines some state information output by the adversary and $\mathcal{O}$ denotes the oracles an adversary has access to, depending on whether they are a type I or type II adversary.

CLUE-IND-RCCA secure if an adversary $\mathcal{A}$ participating in the security game of Figure 2 has a negligible advantage in 1^λ, defined as follows:

$$\begin{aligned}\mathsf{Adv}^{\text{CLUE-IND-RCCA}}_{\Pi_{\text{CLUE}},\mathcal{A}}(1^\lambda) =& |\Pr[\mathsf{Exp}^{\text{CLUE-IND-RCCA},1}_{\Pi_{\text{CLUE}},\mathcal{A}}(1^\lambda) = 1] - \\ & \Pr[\mathsf{Exp}^{\text{CLUE-IND-RCCA},0}_{\Pi_{\text{CLUE}},\mathcal{A}}(1^\lambda) = 1]| \leq \mathsf{negl}(1^\lambda).\end{aligned}$$

4. Construction

In this Section, we present a concrete pairing-based CLUE scheme (Π_{CLUE}). Primarily, we chose to present a concrete CLUE scheme to demonstrate that Π_{CLUE} is comparably efficient to other certificateless updatable PKE schemes such as the CL-PRE scheme from [9]. Informally, our choice of the underlying certificateless PKE scheme is a modified version of the pairing-based NewFullCLE scheme proposed by [6]. Firstly, we deemed the construction from [6] to be a worthy candidate for the underlying CL-PKE scheme used in our construction due to the level of security satisfied. Secondly, we chose a pairing-based CL-PKE scheme for the same reasons as [6]. Namely, regarding CL-PKE literature all concrete schemes generated without pairings are supported by weaker security assumptions in the random oracle model. Whilst schemes without pairings are typically more efficient computationally speaking, the authors of [6] demonstrated

that their NewFullCLE scheme attained comparable efficiency to some non-pairing schemes. We discuss efficiency in greater detail in the full version of this work.

Our choice for the update mechanism is a *key-homomorphic pseudorandom function* (KH-PRF) $\mathsf{F}_{\mathrm{DDH}}$. We chose this KH-PRF, not only for its desired homomorphic properties but also for its use in previous UE schemes [10, 11, 12]. To be clear, we necessitate the use of a KH-PRF building block ($\mathsf{F}_{\mathrm{DDH}}$) to support the update functionality in our CLUE construction and we note that the use of this mechanism is a key differentiator of our construction concerning that of [6]. Necessary to security, we require that the KH-PRF is proven secure in the random oracle model, assuming the hardness of the decisional Diffie-Hellman problem in some finite cyclic group. We defer the reader to the formal definition of a KH-PRF and security of $\mathsf{F}_{\mathrm{DDH}}$ in Appendix B. Concretely, we denote the KH-PRF as $\mathsf{F}_{\mathrm{DDH}} : \mathbb{Z}_q \times \mathbb{G}_2 \to \mathbb{G}_1$ whereby $\mathcal{K} = (\mathbb{Z}_q, \oplus)$ and $\mathcal{X} = (\mathbb{G}_2, \otimes)$ are additive and multiplicative groups respectively. Note that $(\mathbb{G}_1, \mathbb{G}_2)$ are cyclic (multiplicative) groups of prime order q. Evaluation of the KH-PRF is $\mathsf{F}_{\mathrm{DDH}}(k, x) = \mathcal{H}_2(x)^k$ (see Definition 8) for cryptographic hash function $\mathcal{H}_2 : \mathbb{G}_2 \to \mathbb{G}_1$, and $\mathsf{F}_{\mathrm{DDH}}(k_1 + k_2, x) = \mathsf{F}_{\mathrm{DDH}}(k_1, x) \cdot \mathsf{F}_{\mathrm{DDH}}(k_1, x)$ holds. Now we present the formal definition of our concrete CLUE scheme.

Definition 5 (CLUE Construction). *Given security parameter $\lambda \in \mathbb{N}$, n epochs, identity space $\mathcal{IDSP} = \{0,1\}^*$, message space $\mathcal{MSP} = \mathbb{G}_1$ and ciphertext space $\mathcal{CSP} = \mathbb{G}_1 \times \mathbb{G}_1$, let groups $(\mathbb{G}_1, \mathbb{G}_2)$ be cyclic (multiplicative) groups of prime order q (a 1^λ-bit prime). We define the CLUE scheme $\Pi_{CLUE} =$ (Setup, Partial-SK-Extract, Set-Secret-Value, Set-SK, Set-PK, Set-Token, Enc, Dec, Upd) as follows,*

- $\mathsf{Setup}(1^\lambda) \xrightarrow{\$} (pp, msk)$: Given the security parameter λ as input, the setup algorithm defines a symmetric bilinear map $\hat{e} : (\mathbb{G}_1 \times \mathbb{G}_1) \to \mathbb{G}_2$ which is a Type I pairing in Definition 6, Appendix B. The following choices are made.
  1. Choose an arbitrary value $P \in \mathbb{G}_1$ to be the generator of $\mathbb{G}_1$ such that we have the element $g = \hat{e}(P, P) \in \mathbb{G}_2$.
  2. Given $s \xleftarrow{\$} \mathbb{Z}_q^*$ chosen uniformly at random, set the master secret key $msk = s$ and set $P' = sP \in \mathbb{G}_1$.
  3. Choose three cryptographic hash functions used as follows[2]: $\mathcal{H}_1 : \{0,1\}^* \to \mathbb{Z}_q^*$; $\mathcal{H}_2 : \mathbb{G}_2 \to \mathbb{G}_1$; $\mathcal{H}_3 : \{0,1\}^* \to \mathbb{Z}_q^*$.

  Set $pp = (q, 1^\lambda, \mathbb{G}_1, \mathbb{G}_2, P, P', \hat{e}, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, n, \mathcal{MSP}, \mathcal{CSP})$ to be the public parameters and master secret key $msk = s \in \mathbb{Z}_q^*$.
- $\mathsf{Partial\text{-}SK\text{-}Extract}(pp, msk, \mathrm{ID}_e) \to \mathrm{ID}_e$: Given $\mathrm{ID}_e \in \{0,1\}^*$ input as the identifier for epoch e, set the partial secret key as $\mathrm{D}_e = ((s + \mathcal{H}_1(\mathrm{ID}_e))^{-1} \cdot P) \in \mathbb{G}_1$. Secretly send ID_e to the server over a secure broadcast channel.[3]

[2]Importantly, hash function $\mathcal{H}_2$ differs from the CL-PKE scheme in [6] to suit the needs of our construction. That is, we require the homomorphic property from the KH-PRF to satisfy updatability, and $\mathcal{H}_2$ is used in the definition of $\mathsf{F}_{\mathrm{DDH}}$.

[3]Note that a server possessing partial secret key ID_e and update token Δ_{e+1} is incapable of decrypting the ciphertext without corrupting either of the secret keys (sk_{e+1},sk_e), which we assume impossible in our security model.

- Set-Secret-Value$(pp, e) \xrightarrow{\$} x_e \in \mathbb{Z}_q$: the data owner randomly selects set secret value x_e for epoch e.
- Set-SK$(pp, \mathrm{D}_e, x_e) \rightarrow \mathrm{sk}_e$: for epoch e the data owner sets secret key $\mathrm{sk}_e := (x_e, \mathrm{D}_e) \in (\mathbb{Z}_q \times \mathbb{G}_1)$.
- Set-PK$(pp, x_e) \rightarrow \mathrm{pk}_e$: for epoch e the data owner computes the public key $\mathrm{pk}_e := y_e = g^{x_e} \in \mathbb{G}_2$.
- Set-Token$(pp, sk_e, x_{e+1}) \rightarrow \Delta_{e+1}$: Using $sk_e := (x_e, \mathrm{ID}_e)$ and new epoch secret value x_{e+1}, we set the token $\Delta'_{e+1} := (-x_e + x_{e+1}) \in \mathbb{Z}_q$; secret key $sk_{e+1} = (x_{e+1}, \mathrm{ID}_e)$ and compute $pk_{e+1} = g^{x_{e+1}}$. Set $\Delta_{e+1} := (\Delta'_{e+1}, pk_{e+1}) \in (\mathbb{Z}_q \times \mathbb{G}_2)$.
- Enc$(pp, M, pk_e, \mathrm{ID}_e) \xrightarrow{\$} \{C_e, \perp\}$: the data owner performs the following three steps.
  1. Select uniform randomness $\sigma \xleftarrow{\$} \mathbb{Z}_q^*$.
  2. Set $r = \mathcal{H}_3(< M^\sigma || pk_e || \mathrm{ID}_e >) \in \mathbb{Z}_q^*$.[4]
  3. Set $C_e = (c_e^1, c_e^2) = (r\mathcal{H}_1(\mathrm{ID}_e)P + rP', M^\sigma \cdot \mathsf{F}_{\mathrm{DDH}}(x_e, g^r))$.
- Dec$(pp, C_e, \mathrm{sk}_e) \rightarrow \{M, \perp\}$: parse ciphertext $C_e = (c_e^1, c_e^2)$ and secret key $sk_e = (x_e, \mathrm{ID}_e)$ and go through the following steps,
  1. Compute $\omega = \hat{e}(c_e^1, \mathrm{ID}_e)$ such that $\omega = \hat{e}(c_e^1, \mathrm{ID}_e) = \hat{e}(r\mathcal{H}_1(\mathrm{ID}_e) \cdot P + rs \cdot P, (s + \mathcal{H}_1(\mathrm{ID}_e))^{-1} \cdot P) = \hat{e}(r(\mathcal{H}_1(\mathrm{ID}_e) + s) \cdot P, (s + \mathcal{H}_1(\mathrm{ID}_e))^{-1} \cdot P) \overset{(*)}{=} \hat{e}(P, P)^{r(\mathcal{H}_1(\mathrm{ID}_e)+s)\cdot(\mathcal{H}_1(\mathrm{ID}_e)+s)^{-1}} = g^r$ where equality $(*)$ holds due to the bilinearity property of $\hat{e}$ (Definition 6, Appendix B).
  2. In order for the data owner to compute r in the next step, M^σ needs to be determined. Given step 1 in which it is determined that $\omega = g^r$, the following can be computed $c_e^2 \cdot \mathsf{F}_{\mathrm{DDH}}(-x_e, \omega) = M^\sigma \in \mathbb{G}_1$. Correctness holds as follows: $c_e^2 \cdot \mathsf{F}_{\mathrm{DDH}}(-x_e, \omega) = M^\sigma \cdot \mathsf{F}_{\mathrm{DDH}}(x_e, g^r) \cdot \mathsf{F}_{\mathrm{DDH}}(-x_e, \omega) = M^\sigma \cdot \mathsf{F}_{\mathrm{DDH}}(x_e - x_e, g^r) = M^\sigma$.[5] Note that the *data owner* randomly chose σ during encryption, so knowledge of this enables the computation of the message $(M^\sigma)^{-\sigma} := M$.[6]
  3. Use the epoch secret-key and public parameters (sk_e, pp) in addition to the previous two steps to compute $r = \mathcal{H}_3(< M^\sigma || pk_e || \mathrm{ID}_e >) \in \mathbb{Z}_q^*$. Message M is accepted *iff* $c_e^1 = r(\mathcal{H}_1(\mathrm{ID}_e)P + P')$ from the computed r value , else failure $(\perp)$ is output.
- Upd$(pp, C_e, \Delta_{e+1}) \rightarrow \{C_{e+1}, \perp\}$: recall the update token and ciphertext $\Delta_{e+1} := (\Delta'_{e+1}, pk_{e+1})$, $C_e = (c_e^1, c_e^2)$ respectively. The server must perform the following steps:
  1. Check $pk_{e+1}^q = 1_{\mathbb{G}_2}$. Abort the update and output failure symbol $\perp$ if this does not hold. Note, validity holds with an honestly generated epoch public key: $pk_{e+1}^q = (g^{x_{e+1}})^q = (g^q)^{x_{e+1}} = (1_{\mathbb{G}_2})^{x_{e+1}}$.
  2. Compute $\omega = \hat{e}(c_e^1, \mathrm{ID}_e) = g^r \in \mathbb{G}_2$. See step 1 of the decryption algorithm for correctness. Set $c_{e+1}^1 := c_e^1$.

[4] Let $< \cdot >$ denote an encoding of the bracket contents to a string $\{0,1\}^*$.

[5] To see the penultimate equation differently, given the definition of the KH-PRF: $\mathsf{F}_{\mathrm{DDH}}(x_e - x_e, g^r) = \mathcal{H}_2(g^r)^{x_e - x_e} = \mathcal{H}_2(g^r)^0 = \mathrm{id}_{\mathbb{G}_1}$.

[6] The technique of using the corresponding randomness for a given epoch to decrypt the ciphertext is utilised in various UE schemes including [10, 12].

3. Use step 2 and the given public key pk_{e+1} to compute $c^2_{e+1} := c^2_e \cdot \mathsf{F}_{\mathrm{DDH}}(\Delta'_{e+1}, \omega)$ and output $C_{e+1} = (c^1_{e+1}, c^2_{e+1})$. Consistency is upheld using ω as follows:
$c^2_{e+1} = c^2_e \cdot \mathsf{F}_{\mathrm{DDH}}(\Delta'_{e+1}, \omega) = M^\sigma \cdot \mathsf{F}_{\mathrm{DDH}}(x_e, g^r) \cdot \mathsf{F}_{\mathrm{DDH}}(-x_e + x_{e+1}, \omega)$
$= M^\sigma \cdot \mathsf{F}_{\mathrm{DDH}}(x_e - x_e + x_{e+1}, g^r) = M^\sigma \cdot \mathsf{F}_{\mathrm{DDH}}(x_{e+1}, g^r)$.

We note that only the second component (c^2_e) of the ciphertext gets updated and the first component (c^1_e) remains the same, in line with previous identity-based approaches used in CL-PKE literature [13, 14]. The first component contains a secure signature of an identifier for the epoch in which the ciphertext was created, and is crucial for computing the value ω in the decryption and update process. The fact that c^1_e remains unchanged is the reason why we do not achieve the stronger PKUE notion of full ciphertext unlinkability, and instead, our construction only achieves encrypted and updated ciphertext indistinguishability.

Security Results Recall, our security framework presented in Section 3 modelled the first notion of *ciphertext indistinguishability* in certificateless public key updatable schemes. Intuitively, this notion captures the indistinguishability of fresh and updated encryptions. Specifically, we encapsulate security against replayable chosen-ciphertext attacks from an adaptive adversary (CLUE-IND-RCCA). We defined our concrete CLUE scheme (Π_{CLUE}) in Section 4 to illustrate the existence of a CLUE construction satisfying ciphertext indistinguishability, a sketch of which is provided in Appendix C. Note, to demonstrate provable security we make use of a modular proof technique first defined in [8] in which we reduce security to an isolated epoch of our CLUE scheme.

Conclusion In our first contribution of this paper, we formally defined a novel certificateless public-key updatable encryption primitive CLUE to mitigate the risk of a malicious key generation centre, when considering applications of a PKUE primitive in a public key infrastructure. In our second contribution, we provided a security framework to model the first notion of ciphertext indistinguishability in certificateless public key updatable schemes. In particular, security against replayable chosen-ciphertext attacks from an adaptive adversary. Our third contribution was to propose a concrete CLUE scheme (Π_{CLUE}) derived from a modified pairing-based CL-PKE scheme [6], which we used as the underlying PKE scheme, and KH-PRFs applied to support the necessary update mechanism in CLUE. Moreover, we provide a proof sketch that our construction satisfies ciphertext indistinguishability.

References

[1] J. Knapp, E. A. Quaglia, Epoch confidentiality in updatable encryption, in: Lecture Notes in Computer Science, volume 13600, International Conference on Provable Security - ProvSec 2022, Springer, 2022, pp. 60–67.

[2] D. Boneh, M. Franklin, Identity-based encryption from the weil pairing, in: Lecture Notes in Computer Science, volume 2139, Advance in Cryptology - CRYPTO 2001, Springer, 2001, pp. 213–229.

[3] S. Al-Riyami, K. Paterson, Certificateless public key cryptography, in: Lecture Notes in Computer Science, volume 2894, Advances in Cryptology, ASIACRYPT 2003, Springer, 2003, pp. 452–473.

[4] A. Shamir, Identity-based cryptosystems and signature schemes, in: Lecture Notes in Computer Science, volume 196, Advances in Cryptology - CRYPTO 1984, Springer, 1984, pp. 47–53.
[5] C. Gentry, Certificate-based encryption and the certificate revocation problem, in: Lecture Notes in Computer Science, volume 2656, Advances in Cryptology - EUROCRYPT 2003, Springer, 2003, pp. 272–293.
[6] B. Libert, J. Quisquater, On constructing certificateless cryptosystems from identity based encryption, in: Lecture Notes in Computer Science, volume 3958, International Workshop on Public Key Cryptography - PKC 2006, Springer, 2006, pp. 474–490.
[7] A. Dent, A survey of certificateless encryption schemes and security models, International Journal of Information Security 7 (2008) 349–377.
[8] M. Klooß, A. Lehmann, A. Rupp, (r) cca secure updatable encryption with integrity protection, in: Y. Ishai, V. Rijmen (Eds.), Lecture Notes in Computer Science, volume 11476, Advances in Cryptology, EUROCRYPT 2019, Springer, 2019, pp. 68–99.
[9] Z. Guo, H.and Zhang, J. Zhang, C. Chen, Towards a secure certificateless proxy re-encryption scheme, in: Lecture Notes in Computer Science, volume 8209, International Conference on Provable Security - ProvSec 2013, Springer, 2013, pp. 330–346.
[10] D. Boneh, K. Lewi, H. Montgomery, A. Raghunathan, Key homomorphic prfs and their applications, Cryptology ePrint Archive, Report 2015/220, 2015. https://eprint.iacr.org/2015/220.
[11] A. Everspaugh, K. Paterson, T. Ristenpart, S. Scott, Key rotation for authenticated encryption, in: J. Katz, H. Shacham (Eds.), Lecture Note in Computer Science, volume 10403, Advances in Cryptology- CRYPTO 2017, 2017, pp. 98–129.
[12] A. Lehmann, B. Tackmann, Updatable encryption with post-compromise security, in: J. Nielsen, V. Rijmen (Eds.), Lecture Notes in Computer Science, volume 10822, Advances in Cryptology, EUROCRYPT 2018, Springer, 2018, pp. 685–716.
[13] F. Zhang, R. Safavi-Naini, W. Susilo, An efficient signature scheme from bilinear pairings and its applications, in: Lecture Notes in Computer Science, volume 2947, International Workshop on Public Key Cryptography - PKC 2004, Springer, 2004, pp. 277–290.
[14] D. Boneh, X. Boyen, Short signatures without random oracles, in: Lecture Notes in Computer Science, volume 3027, Advances in Cryptology - EUROCRYPT 2004, Springer, 2004, pp. 56–73.
[15] C. Boyd, D. G.T., K. Gjøsteen, Y. Jiang, Fast and Secure Updatable Encryption†, Technical Report, Cryptology ePrint Archive, Report 2019/1457, 2019. https://eprint.iacr.org/2019/1457.pdf, 2020.
[16] P. Ananth, A. Cohen, A. Jain, Cryptography with updates, in: J. Coron, N. J. (Eds.), Lecture Notes in Computer Science, volume 10211, Advances in Cryptology, EUROCRYPT 2017, Springer, 2017, pp. 445–472.
[17] E. Eaton, D. Jao, C. Komlo, Y. Mokrani, Towards post-quantum key-updatable public-key encryption via supersingular isogenies, in: Lecture Notes in Computer Science, volume 13203, Selected Areas in Cryptography: 28th International Conference - SAC 2022, Springer, 2022, pp. 461–482.
[18] Y. Dodis, H. Karthikeyan, D. Wichs, Updatable public key encryption in the standard model, in: Lecture Notes in Computer Science, volume 13044, Theory of Cryptography

Conference - TCC21, Springer, 2021, pp. 254–285.
[19] D. Boneh, S. Eskandarian, S. Kim, M. Shih, Improving speed and security in updatable encryption schemes., in: Advances in Cryptology – ASIACRYPT 2020, volume 12493, Lecture Notes in Computer Science, Springer, 2020.
[20] L. Chen, Y. Li, Q. Tang, Cca updatable encryption against malicious re-encryption attacks, in: Lecture Notes in Computer Science, volume 12493, Advances in Cryptology - ASIACRYPT 2020, Springer, 2020, pp. 590–620.
[21] Y. Jiang, The direction of updatable encryption does not matter much, in: Lecture Notes in Computer Science, volume 12493, Advances in Cryptology - ASIACRYPT 2020, Springer, 2020, pp. 529–558.
[22] R. Nishimaki, The direction of updatable encryption does matter, Cryptology ePrint Archive (2021).
[23] M. Blaze, M. Bleumer, G.and Strauss, Divertible protocols and atomic proxy cryptography, in: International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 1998, pp. 127–144.
[24] L. Xu, X. Wu, X. Zhang, Cl-pre: A certificateless proxy re-encryption scheme for secure data sharing with public cloud, in: Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, ASIA-CCS 2012, Association for Computing Machinery, 2012, p. 87–88.
[25] A. Srinivasan, C. Rangan, Certificateless proxy re-encryption without pairing: revisited, in: Proceedings of the 3rd International Workshop on Security in Cloud Computing, SCC 2015, Association for Computing Machinery, 2015, pp. 41–52.
[26] K. Yang, J. Xu, Z. Zhang, Certificateless proxy re-encryption without pairings, in: Lecture Notes in Computer Science, volume 8565, International Conference on Information Security and Cryptology - ICISC 2013, Springer, 2013, pp. 67–88.
[27] E. Lee, Improved security notions for proxy re-encryption to enforce access control, in: T. Lange, O. Dunkelman (Eds.), Lecture Notes in Computer Science, volume 11368, International Conference on Cryptology and Information Security in Latin America-LATINCRYPT 2017, Springer, 2017, pp. 66–85.
[28] A. Davidson, A. Deo, E. Lee, K. Martin, Strong post-compromise secure proxy re-encryption, in: Australasian Conference on Information Security and Privacy- ACISP 2019, volume 11547, Lecture Notes in Computer Science, Springer, 2019, pp. 58–77.
[29] S. Galbraith, K. Paterson, N. Smart, Pairings for cryptographers, Discrete Applied Mathematics 156 (2008) 3113–3121.
[30] R. Canetti, H. Krawczyk, J. Nielsen, Relaxing chosen-ciphertext security, in: Lecture Notes in Computer Science, volume 2729, Advances in Cryptology- CRYPTO 2003, Springer, 2003, pp. 565–582.

A. Related Work

Updatable encryption (UE) schemes [12, 15, 8, 16] are traditionally designed in the symmetric setting, but recent focus has turned to formalise various public-key encryption primitives imbued with an update functionality [17, 1, 18]. In this paper, we are interested in the PKUE

primitive defined by the authors of [1]. Both UE and PKUE can be viewed through two lenses in the literature: *Ciphertext-dependent* UE schemes [10, 11, 19, 20] and *ciphertext-independent* UE schemes [12, 8, 15, 21, 22, 1]. The former requires the data owner to produce a token for *each* ciphertext, therefore, it is computationally expensive and inefficient for the data owner. Additionally, generating individual tokens translates to the storage of epoch keys over a long time, ultimately defeating the purpose of the UE primitive. Conversely, the latter strain requires the data owner to generate a *single* update token which enables the server to *sequentially* update ciphertexts using a token derived from the current and new epoch keys alone. Observe that the CLUE primitive we introduce (Definition 1, Section 2) is designed in the ciphertext-independent setting.

Proxy re-encryption (PRE), first introduced by [23], is a primitive used for ciphertext decryption delegation in which a proxy server generates a re-encryption key used to rotate the cryptographic key a ciphertext is encrypted by from one user to another. Specific to this paper, *certificateless-PRE* (CL-PRE) [24, 25, 26, 9] is a primitive introduced following the advent of identity-based PRE [2] to resolve the issues of key escrow and *user revocation* simultaneously. The distinctions between CLUE and CL-PRE directly follow from the fundamental differences of the underlying updateable primitive ((PK)UE and PRE respectively). Comparisons between the two have been made in the works of [27, 28, 12, 8]. We highlight the most prevalent difference is that PRE rotates keys to delegate ciphertext decryption, whereas (PK)UE updates ciphertexts to a new period. Further, the security framework of the two primitives differs. In particular, PRE does not typically capture information an adversary can infer from the corruption of the re-encryption key, nor does it consider the notion of ciphertext unlinkability usually captured in UE security modelling.

B. Definitions and Assumptions

In this Section, we explain the intuition and assumptions required such that *ciphertext indistinguishability* is achieved for construction Π_{CLUE} from Section 4. We state further definitions and assumptions are required for security analysis, given in Section C, and note that proofs of the lemmas are omitted due to lack of space. To start, the first definition presented is used when defining the pairing map used in the construction Π_{CLUE}.

Definition 6 (Bilinear Maps). *Let additive groups $\mathbb{G}_1, \mathbb{G}_2$ have prime order q, such that $\mathbb{G}_1$ is generated by P, $\mathbb{G}_2$ is generated by Q, and multiplicative group $\mathbb{G}_T$ is also of prime order q. A pairing is a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties,*

1. **Bilinearity:** $\forall a, b \in \mathbb{F}_q^*, \forall P \in \mathbb{G}_1, Q \in \mathbb{G}_2 : \hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$;
2. **Non-Degeneracy:** $\hat{e} \neq 1$, that is, the mapping is not the identity map;
3. **Computability:** there exists an efficient algorithm to determine the output of map $\hat{e}$.

Definition 6 can be classified into three types, in line with [29]:

- **I** : If $\mathbb{G}_1 = \mathbb{G}_2$. This is known as a *symmetric* bilinear map.
- **II** : If $\mathbb{G}_1 \neq \mathbb{G}_2$ and there exists an efficiently computable *homomorphism* $\phi : \mathbb{G}_2 \to \mathbb{G}_1$.

- **III** : If $\mathbb{G}_1 \neq \mathbb{G}_2$ and there does not exist an efficiently computable homomorphism like ϕ.

Next we introduce the p-Bilinear Diffie Hellman *Inversion* (p-BDHI) problem, which is used to prove the security of our construction in Section 4. The p-BDHI problem is stated as follows:

Definition 7 (p-BDHI Problem). *Given map $\hat{e}$ defined as in Definition 6 over groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ and given $\{P, \alpha P, \alpha^2 P, \ldots, \alpha^p P\} \in \mathbb{G}_1^{p+1}$, the p-BDHI problem is considered hard if it is computationally intractable to compute $\hat{e}(P, P)^{1/\alpha} \in \mathbb{G}_2$ in polynomial time.*

Key-Homomorphic PRFs The *update* feature in CLUE is attained using a collision-resistant homomorphic hash function in the encryption process, which we model as a random oracle. For our construction, we assume the hash function $\mathcal{H}_2 : \mathbb{G}_2 \to \mathbb{G}_1$ is homomorphic, and we consider building the hash function from a key-homomorphic PRF.

Definition 8 (Key-Homomorphic PRF [10]). *Consider an efficiently computable function F : $\mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ such that $(\mathcal{K}, \oplus)$ and $(\mathcal{Y}, \otimes)$ are groups. Then $(F, \oplus, \otimes)$ is a key-homomorphic PRF if the following properties hold,*

1. F is a secure pseudorandom function.
2. For every $k_1, k_2 \in \mathcal{K}$ and every $x \in \mathcal{X}$: $\mathsf{F}(k_1, x) \otimes \mathsf{F}(k_2, x) = \mathsf{F}((k_1 \oplus k_2), x)$.

Lemma 1. *Given the KH-PRF used in Π_{CLUE} defined as $\mathsf{F}_{DDH} : \mathbb{Z}_q \times \mathbb{G}_2 \to \mathbb{G}_1$ with $\mathcal{K} = (\mathbb{Z}_q, \oplus)$, $\mathcal{X} = (\mathbb{G}_2, \otimes)$ the additive and multiplicative groups of prime order q respectively such that $(\mathbb{G}_1, \mathbb{G}_2)$ are cyclic (multiplicative) groups of prime order q, evaluation of the KH-PRF is $F_{DDH}(k, x) = \mathcal{H}_2(x)^k$. Further, $F_{DDH}(k_1 + k_2, x) = F_{DDH}(k_1, x) \cdot F_{DDH}(k_1, x)$. That is, F_{DDH} satisfies Definition 8. Then F_{DDH} is a secure KH-PRF in the random oracle model assuming the hardness of the decisional Diffie-Hellman problem in $\mathbb{G}_1$.*

Updatable Encryption Assumptions Construction CLUE is designed with *deterministic* ciphertext updates, therefore, the security of Π_{CLUE} assumes the properties of randomness-preserving re-encryption; the underlying CL-PKE scheme Π_{PKE} is tidy and simulatable token generation. The formal definitions of these properties are utilised in the security proof of Theorem 1 to argue that the indistinguishability of fresh and updated ciphertexts is satisfied. We present them below. Due to lack of space, we omit the proofs of Lemmas and defer the reader to the full version of this paper.

Definition 9 (Randomness-Preserving Re-Encryption [8]).

Given the updatable scheme CLUE is designed for *deterministic* updates, an updated ciphertext is *randomness-preserving* assuming CLUE encrypts with uniformly chosen randomness ($\mathsf{Enc}(pp, M, pk_e, \mathrm{ID}_e)$ and $\mathsf{Enc}(pp, M, pk_e, \mathrm{ID}_e; r)$ for uniformly chosen r are identically distributed). If for all $(pp, msk) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$; for all old and new epoch

key pairs $k_e := (pk_e, sk_e)$, $k_{e+1} := (pk_{e+1}, sk_{e+1})$ generated from running the Partial-SK-Extract, Set-Secret-Value, Set-SK, Set-PK algorithms in epoch e and $(e+1)$ respectively; for all valid ciphertexts $C \in \mathcal{CSP}$ and for all tokens $\Delta_{e+1} \leftarrow$ Set-Token$(pp, (pk_e, sk_e))$, we then have the following:

$$\mathsf{Enc}(pp, \mathsf{Dec}(pp, C_e, sk_e), pk_{e+1}, \mathrm{ID}_e) = \mathsf{Upd}(pp, C_e, \Delta_{e+1}).$$

Lemma 2. *The scheme Π_{CLUE} satisfies* randomness preserving re-encryption *given in Definition 9.*

Definition 10 (Randomness-Recoverable Tidy Encryption Scheme). *A public-key encryption scheme is called* randomness-recoverable *if there is an associated efficient deterministic algorithm RDec(pp, C_e, sk_e) for epoch e such that $\forall (pk_e, sk_e), M, r$:RDec$(pp, sk_e,$ Enc$(pp, pk_e, M; r)) = (M, r)$. We call a randomness-recoverable public-key encryption scheme* tidy *if $\forall (pk_e, sk_e, C_e)$:*

$$\textit{RDec}(pp, C_e, sk_e) = (M, r) \implies \textit{Enc}(pp, pk_e, M; r) = C_e.$$

Lemma 3. *The CL-PKE scheme Π_{PKE} implicit in Π_{CLUE} satisfies the* randomness recoverable tidy encryption *property given in Definition 10.*

Assumption 1 (Reversible Update Tokens). *Update token Δ^{-1} is called a reverse token of Δ if for every pair of epoch keys $(k_{e_{old}} = (pk_{e_{old}}, sk_{e_{old}}), k_{e_{new}} = (pk_{e_{new}}, sk_{e_{new}}))$ in key-space $\mathcal{KSP}$ such that $\Delta \in$ supp(Set-Token$(pp, sk_{e_{old}}, x_{e_{new}})$), we have reversible token $\Delta^{-1} \in$ supp(Set-Token$(pp, sk_{e_{new}}, x_{e_{old}})$.*

Definition 11 (Simulatable Token Generation). *The CLUE scheme Π_{CLUE} defined in Section 4 has simulatable token generation if the following properties hold:*

1. There exists a PPT algorithm denoted Sim-Set-Token(pp) which samples a pair of update tokens (Δ, Δ^{-1}) of the token and reverse token respectively.
2. For arbitrary (fixed) $k_{e_{\text{old}}} := (pk_{e_{\text{old}}}, sk_{e_{\text{old}}})$ which is generated from running the Partial-SK-Extract, Set-Secret-Value, Set-SK, Set-PK algorithms, the following token (Δ) distributions are the same:
   - Distribution induced by running $(\Delta, \cdot) \xleftarrow{\$}$ Sim-Set-Token(pp);
   - For epoch key $k_{e_{\text{new}}} := (pk_{e_{\text{new}}}, sk_{e_{\text{new}}})$ the distribution is induced by running $(\Delta, \cdot) \xleftarrow{\$}$ Set-Token$(pp, sk_{e_{\text{old}}}, x_{e_{\text{new}}})$.

Lemma 4. *The CLUE scheme Π_{CLUE} defined in Section 4 satisfies* simulatable *token and reversible token generation given in Definition 11.*

C. Security Analysis

In this Section, we provide a sketch analysis of security for our construction Π_{CLUE}. Due to lack of space, we provide an overview of our proof, deferring the reader to the full version of this paper for a detailed proof of correctness and security. Observe that when proving CLUE-IND-RCCA security of Π_{CLUE} to achieve *ciphertext indistinguishability*, we assume several properties regarding the underlying building blocks. This proof method follows directly from [8] who proposed a generic transformation demonstrating that it is sufficient to consider the underlying encryption and key-rotation capabilities of a scheme (almost) separately and therefore reduce proving to the standard-setting. Now, we present a detailed statement of security.

Theorem 1. *Given Π_{CLUE} is a deterministic updatable encryption scheme satisfying randomness-preserving tidy updates (Lemma 2); simulatable token generations (Lemma 4) and the underlying certificateless encryption scheme Π_{PKE} satisfies CLUE-IND-RCCA in an* isolated *epoch, then the construction Π_{CLUE} satisfies security notion CLUE-IND-RCCA assuming the intractability of the p-BDHI problem formalised in Definition 7 (Appendix B).*

Sketch Proof. We take a two-step modular approach in proving Theorem 1, adapting the techniques of [8] to suit our security model, such that we can reduce the proof of security from the updatable setting (CLUE) to the standard setting. That is, we provide a proof reduction to the security of the underlying CL-PKE scheme $\Pi_{\mathsf{PKE}} := (\mathsf{Setup}, \mathsf{Partial\text{-}SK\text{-}Extract}, \mathsf{Set\text{-}Secret\text{-}Value}, \mathsf{Set\text{-}SK}, \mathsf{Set\text{-}PK}, \mathsf{Enc}, \mathsf{Dec})$ of construction Π_{CLUE}. The first step of the proof is used to prove that Π_{PKE} satisfies a security notion akin to CLUE-IND-RCCA for an *isolated epoch* of Π_{CLUE}, labelled CL-PKE-IND-RCCA (full details are provided in the full version of this paper). Again, security is against the adaptive adversary $\mathcal{A} = (\mathcal{A}_{\mathrm{I}}, \mathcal{A}_{\mathrm{II}})$ defined in Section 3. Briefly, we are able to prove this notion is satisfied by observing that the authors of [6] demonstrated that Π_{PKE} satisfies the *strictly stronger* notion of CL-PKE-IND-CCA security against adversary $\mathcal{A}$ in the random oracle model assuming the hardness of Definition 7. Moreover, we prove this security notion holds for Π_{PKE} following the implication [30, 9] that satisfaction of CCA security implies that the same construction will also satisfy CL-PKE-IND-RCCA security.

In the second step of the proof we look at proving the security of the updatable construction Π_{CLUE} over *multiple epochs*. In more detail, this part of the proof sees a series of *hybrid games* H_l built for epochs $e_l \in \{0, \ldots, \hat{e}+1\}$ of Π_{CLUE} where $\hat{e}$ is the maximum number of epochs in which an adversary $\mathcal{A}$ can query oracles (Figure 1). Suppose we have adversary $\mathcal{A}$ against Π_{CLUE}, defined in Section 3. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}_l$ against the standard CL-PKE construction Π_{PKE} [6] which is proven CL-PKE-IND-RCCA secure in the first part of our proof. Constructing adversaries in this way enables us to demonstrate the indistinguishability of games $\mathsf{H}_{l-1}, \mathsf{H}_l$ for the epochs of the CLUE scheme $e_l \in \{0, \ldots, \hat{e}+1\}$. Thus, updatable security can be *reduced* to the security of Π_{PKE} in an isolated epoch of the CLUE scheme.