# 鍵更新機能付き検索可能暗号：効率化に向けた一工夫

松崎　なつめ[1]　穴田 啓晃[1]　金岡 晃[2]　渡邉 洋平[3,†1]

**概要**：鍵更新機能付き検索可能暗号は，クライアントの鍵を更新可能とする公開鍵系検索可能暗号である．モバイル機器や IoT 機器からの，機器紛失や盗難に伴うクライアントの鍵の漏洩対策の 1 つとして，ACISP 2018 において，鍵更新機能付き検索可能暗号のモデルと一般的構成が提案されている．本稿では，提案方式のラズベリーパイでの実装結果を報告するとともに，実用化を考慮した一工夫について提案する．

**キーワード**：検索可能暗号，鍵更新，効率化

# Key-updatable Public-key Encryption with Keyword Search : An Efficient Construction

Natsume MATSUZAKI[1]　Hiroaki ANADA[1]　Akira KANAOKA[2]　Yohei WATANABE[3,†1]

***Abstract:*** Key-updatable public-key encryption with keyword search (KU-PEKS) is PEKS that enables a client to update his/her secret key. At ACISP 2018, Anada et al. proposed models and generic constructions of KU-PEKS as one of possible countermeasure against secret key leakage by loss or theft of mobile and IoT devices. In this paper, we report implementation results of the proposed constructions using Raspberry Pie, which is a famous single-board computer, and mention some ideas for practical implementation.

***Keywords:*** Searchable encryption, Public-key encryption with keyword search, Key-updating functionality, Implementation.

## 1. はじめに

### 1.1 背景

検索可能暗号は，クラウドの安全な利用を目的とした高機能暗号の一種であり，クラウドに預託した暗号文から，暗号化したクエリを用いて所望の情報を検索可能とする．

筆者らは，文献 [2][7][8][9][10] において，鍵更新が可能な公開鍵系の検索可能暗号（Key-updatable Public-key Encryption with Keyword Search: KU-PEKS）を検討してきた．その理由は以下のとおりである．検索可能暗号の UseCase として，電子メールの例がよく挙げられる．例えば，「緊急」の検索キーワードを用いて，すぐに返事を返すべき電子メールをクラウドから検索する場合である．この検索を，据え置きの端末のみならず，携帯端末でも可能とする場合，端末の紛失や盗難による秘密鍵の漏洩対策が必要となる．鍵の更新は，鍵漏洩対策の一アプローチであり，万が一鍵が漏洩しても，鍵更新により古い鍵が無効化される．また，電子メールの UseCase 以外にも，小型の IoT 機器を用いて，例えば位置情報を検索キーワードとして様々な情報を検索し，対応したサービスを提供する場合などが考えられる．小型の IoT 機器では，秘密鍵の安全な実装のために，コストがかけにくいことが予測される．そのため，鍵漏洩対策として，KU-PEKS の検討が重要と考える．

[1] 長崎県立大学 情報システム学部 情報セキュリティ学科
Department of information Security, University of Nagasaki
[2] 東邦大学 大学院 理学部 情報科学科
Graduate School of Science, Toho University
[3] 電気通信大学 大学院情報理工学研究科 日本学術振興会特別研究員 (PD)
Graduate School of Informatics and Engineering, The University of Electro-Communications, JSPS Research Fellow (PD)
[†1] 現在，国立研究開発法人 情報通信研究機構
Presently with National Institute of Information and Communications Technology

### 1.2 今までの研究成果と，本稿での貢献

筆者らは，文献 [7][8][9][10] での検討を経て，2018 年，23rd Australasian Conference on Information Security and Privacy (ACISP 2018) において，鍵更新機能付き検索可能暗号（KU-PEKS) の概念と，2 種の一般的構成モデルを提案した [2].

第 1 の構成モデルは，クライアントの秘密鍵の更新に伴い，対応する公開鍵を更新する公開鍵更新モデル（Key-evolution Model）である．公開鍵更新モデルの具体的な方式として，2 方式（$\Pi_{\mathrm{KE}}^{\mathsf{rom}}$，および $\Pi_{\mathrm{KE}}^{\mathsf{std}}$）を提案している．$\Pi_{\mathrm{KE}}^{\mathsf{rom}}$ は，ランダムオラクルモデルの検索可能暗号（例えば，Boneh-Franklin 提案の ID ベース暗号 [3] を応用した検索可能暗号）と，IND-CPA を満たす公開鍵暗号（例えば，ElGamal 暗号）を組み合わせることで，ランダムオラクルモデルの KU-PEKS を実現している．また，$\Pi_{\mathrm{KE}}^{\mathsf{std}}$ は，スタンダードモデルの検索可能暗号（例えば，Jutla-Roy ら提案の ID ベース暗号 [4]）と，IND-CPA を満たす公開鍵暗号を組み合わせることで，スタンダードモデルの KU-PEKS を実現している．

第 2 の構成モデルは，クライアントの公開鍵の更新は不要となる鍵隔離モデル（Key-insulation Model) である．公開鍵の定期的な配布と管理が不要となるため，実用的である一方，公開鍵更新モデルよりも計算が複雑である．鍵隔離モデルの具体的な方式として，1 方式（$\Pi_{\mathrm{KI}}^{\mathsf{std}}$）を提案している．$\Pi_{\mathrm{KI}}^{\mathsf{std}}$ は，スタンダードモデルの鍵隔離暗号（例えば，渡邉-四方提案の鍵隔離暗号 [6]）と，ID ベース暗号を応用した検索可能暗号を組み合わせている．ここで用いる ID ベース暗号は，鍵隔離型 ID ベース暗号を，マスター鍵漏洩に対策するように改良した方式である．詳細は，[2] に記載している．

本稿では，これらのうち，公開鍵更新モデル（2 方式）の，ラズベリーパイでの実装結果を報告するとともに，実用化を考慮した工夫について述べる．

工夫の 1 つは，ハッシュ関数の導入である．検索キーワードの代わりに，そのハッシュ値を用いる．今までの方法では，古い暗号文を再暗号化する際に，検索キーワード自身がクラウドに求められていた．今回，ここにハッシュ関数を導入することにより，替わりに検索キーワードのハッシュ値が導出される．検索キーワードそのものが求められないため，利用者にとり安心感が向上する．安全性は，[2] と同様に証明できる．第 2 章に述べる．

もう 1 つの工夫は，クラウドでの検索の高速化である．検索時に，トラップドア値に加え，検索キーワードのハッシュ値をクラウドに示す．古い暗号文の検索には検索関数を用いず，ハッシュ値の一致のみを用いることで，検索関数の回数を削減する．安全性証明はこれからの課題であるが，実装上の工夫として，第 4 章に示す．

本稿では，これらの工夫を公開鍵更新モデルに適用するが，鍵隔離モデルにも適用可能である．

## 2. 鍵更新機能付き検索可能暗号

鍵更新機能付き検索可能暗号（KU-PEKS) は，クライアント鍵を更新可能とした検索可能暗号である．KU-PEKS は，クライアントの秘密鍵の更新に伴い，対応する公開鍵を更新する公開鍵更新モデルと，公開鍵の更新は不要となる鍵隔離モデルからなる．本章では，前者モデルで説明を行う．

### 2.1 鍵更新機能付き検索可能暗号とは

本節では，KU-PEKS の概念モデルと要件を概説する．KU-PEKS は送信クライアント，受信クライアント，クラウドの 3 つのエンティティからなる．また，通常の検索可能暗号における (1) 預託フェーズと，(3) 検索フェーズに加え，(2) 鍵更新フェーズを備える．鍵更新フェーズでは，受信クライアント側で鍵ペアを更新し，再暗号化鍵を生成する．クラウド側では，再暗号化鍵を用いて，暗号文を更新する．なお，ここでは検索キーワードを含むインデックスの暗号化と検索キーワードの暗号化について焦点をあて，ドキュメント自身の暗号化と復号については省略するものとする．

KU-PEKS $\Pi_{\mathrm{KE}} = (\mathsf{KE.Setup}, \mathsf{KE.Upd}, \mathsf{KE.Enc}, \mathsf{KE.ReEnc}, \mathsf{KE.Trapdoor}, \mathsf{KE.Test})$ は，以下の 6 個のアルゴリズムからなる．更新可能期間を $\mathcal{T}$ とし $|\mathcal{T}| = \mathrm{poly}(\lambda)$ とする．

- $\mathsf{KE.Setup}(1^\lambda) \to (\mathsf{pk}_1, \mathsf{sk}_1)$: セキュリティパラメータ $1^\lambda$ を入力として，初期の鍵ペア $(\mathsf{pk}_1, \mathsf{sk}_1)$ を出力する．
- $\mathsf{KE.Upd}(\mathsf{pk}_i, \mathsf{sk}_i) \to (\mathsf{pk}_{i+1}, \mathsf{sk}_{i+1}, \mathsf{rk}_{i\to i+1})$: 期間 $i \in \mathcal{T}$ の鍵ペアを入力として，次の期間 $i+1 \in \mathcal{T}$ の鍵ペアと，再暗号化鍵 $\mathsf{rk}_{i\to i+1}$ を出力する．
- $\mathsf{KE.Enc}(\mathsf{pk}_i, w) \to \mathsf{c}_{w,i}^{(0)}$: 期間 $i \in \mathcal{T}$ の公開鍵 $\mathsf{pk}_i$ とキーワード $w \in \mathcal{W}$ を入力として，暗号文 $\mathsf{c}_{w,i}^{(0)}$ を出力する．表記の上付き添え字は，更新の回数を示す．$\mathsf{c}_{w,i}^{(0)}$ の更新回数は 0 である．
- $\mathsf{KE.ReEnc}(\mathsf{pk}_{i+1}, \mathsf{rk}_{i\to i+1}, \mathsf{c}_{w,j}^{(k)}) \to \mathsf{c}_{w,j}^{(k+1)}$ or $\perp$: 期間 $i+1 \in \mathcal{T}$ の公開鍵 $\mathsf{pk}_{i+1}$，再暗号化鍵 $\mathsf{rk}_{i\to i+1}$ と暗号文 $\mathsf{c}_{w,j}^{(k)}$ を入力として，$j+k=i$ が満たされる場合には更新した暗号文 $\mathsf{c}_{w,j}^{(k+1)}$ を出力し，それ以外では $\perp$ を出力する．
- $\mathsf{KE.Trapdoor}(\mathsf{pk}_i, \mathsf{sk}_i, w') \to \mathsf{t}_{w',i}$: 期間 $i \in \mathcal{T}$ の鍵ペア $(\mathsf{pk}_i, \mathsf{sk}_i)$ と検索キーワード $w' \in \mathcal{W}$ を入力として，期間 $i \in \mathcal{T}$ のトラップドア $\mathsf{t}_{w',i}$ を出力する．
- $\mathsf{KE.Test}(\mathsf{pk}_i, \mathsf{t}_{w',i}, \mathsf{c}_{w,j}^{(k)}) \to 1$ or 0: 期間 $i \in \mathcal{T}$ の公開鍵 $\mathsf{pk}_i$，トラップドア $\mathsf{t}_{w',i}$ と暗号文 $\mathsf{c}_{w,j}^{(k)}$ を入力とし，$w = w'$ かつ，$j+k=i$ が満たされる場合には 1 を出力し，それ以外では 0 を出力する．

KU-PEKS の要件は以下の 4 点である．

**要件 1**：受信クライアントの新しい秘密鍵は，古い鍵，および公開の情報から導出困難であること．

**要件 2**：古い鍵は受信クライアントから削除すること．

**要件 3**：クラウドでは，更新前の古い公開鍵で暗号化された暗号文を対象に，更新後の新しい鍵で生成したトラップドアで検索できること．この要件は，可用性を目的としたものであり，受信クライアントが，保持する新しい鍵だけで，古い暗号文も対象として検索可能とする．

**要件 4**：クラウドでは，更新後の新しい公開鍵で暗号化された暗号文を対象に，古い鍵で生成したトラップドアで検索できないこと．この要件は，漏洩の可能性のある古い鍵を無効化する意味を持つ．更新前の古い公開鍵で暗号化された暗号文を対象として古い鍵を無効化することは難しいが，本要件により，更新後の新しい公開鍵で暗号化された暗号文を対象とした攻撃を防ぐ．

### 2.2 安全性定義

**正当性.** KU-PEKS $\Pi_{\mathrm{KE}}$ は以下の復号の正当性を要求する．全ての $\lambda \in \mathbb{N}$ と，全ての $i \in \mathcal{T}$，全ての $j \in \{1,\ldots,i-1\}$ と，全ての $(\mathsf{pk}_1,\mathsf{sk}_1) \leftarrow \mathsf{KE.Setup}(1^\lambda)$，全ての $(\mathsf{pk}_\ell,\mathsf{sk}_\ell,\mathsf{rk}_{\ell-1\to\ell}) \leftarrow \mathsf{KE.Upd}(\mathsf{pk}_{\ell-1},\ \mathsf{sk}_{\ell-1})$（ただし，$2 \le \ell \le i$），全ての $w \in \mathcal{W}$ について，$\mathsf{KE.Test}(\mathsf{pk}_i,\mathsf{KE.Trapdoor}(\mathsf{pk}_i,\mathsf{sk}_i,w),\mathsf{c}_{w,j}^{(i-j)}) \to 1$ が成り立つ．ただし $\mathsf{c}_{w,j}^{(i-j)} \leftarrow \mathsf{KE.ReEnc}(\mathsf{pk}_i,\mathsf{rk}_{i-1\to i},\mathsf{KE.ReEnc}(\cdots\mathsf{KE.ReEnc}(\mathsf{pk}_{j+1},\mathsf{rk}_{j\to j+1},\mathsf{KE.Enc}(\mathsf{pk}_j,w))\cdots))$ とする．

**安全性.** 以下の試行 $\mathsf{Exp}^{\mathsf{KE\text{-}CKA}}_{\Pi_{\mathrm{KE}},\mathcal{A}}(1^\lambda)$ を考える．

$\mathsf{Exp}^{\mathsf{KE\text{-}CKA}}_{\Pi_{\mathrm{KE}},\mathcal{A}}(1^\lambda)$

$\mathsf{ctr} := 1,\ \ (\mathsf{pk}_1,\mathsf{sk}_1) \leftarrow \mathsf{KE.Setup}(1^\lambda)$

$(w_0^*, w_1^*, state) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{KG}},\mathcal{O}_{\mathrm{KL}},\mathcal{O}_{\mathrm{TD}}}(\mathsf{pk}_1)$

$b \xleftarrow{\$} \{0,1\},\ \ \mathsf{c}^{(0)}_{w_b^*,\mathsf{ctr}} \leftarrow \mathsf{KE.Enc}(\mathsf{pk}_{\mathsf{ctr}}, w_b^*)$

$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{KL}},\mathcal{O}_{\mathrm{TD}}}(state, \mathsf{c}^{(0)}_{w_b^*,\mathsf{ctr}})$

If $b' = b$ return 1 else return 0

$\mathcal{A}$ は，表記の右上に示されるオラクルに，アクセスできる．各オラクルの説明は次の通りである．

$\mathcal{O}_{\mathrm{KG}}$:最初に $\mathcal{SK} := \emptyset$ とおく．$\mathcal{A}$ からクエリを受け取り，$(\mathsf{pk}_{\mathsf{ctr}+1},\mathsf{sk}_{\mathsf{ctr}+1},\mathsf{rk}_{\mathsf{ctr}\to\mathsf{ctr}+1}) \leftarrow \mathsf{KE.Upd}(\mathsf{pk}_{\mathsf{ctr}},\mathsf{sk}_{\mathsf{ctr}})$ を実行して，$(\mathsf{pk}_{\mathsf{ctr}+1},\mathsf{rk}_{\mathsf{ctr}\to\mathsf{ctr}+1})$ を $\mathcal{A}$ に返すオラクル．そして，$\mathsf{sk}_{\mathsf{ctr}+1}$ を $\mathcal{SK}$ に追加して，$\mathsf{ctr} := \mathsf{ctr}+1$ とする．

$\mathcal{O}_{\mathrm{KL}}$:期間 $i \in \mathcal{T}$ のクエリに対して，もし $i < \mathsf{ctr}$ ならば，$\mathsf{sk}_i \in \mathcal{SK}$ を返し，それ以外では $\perp$ を返すオラクル．このオラクルは鍵漏洩を許していることを意味する．

$\mathcal{O}_{\mathrm{TD}}$:期間 $i \in \mathcal{T}$ のクエリ $(w,i) \in \mathcal{W}\times\mathcal{T}$ に対して，もし $i \le \mathsf{ctr}$ であり $(w,i) \notin \{(w_0^*,\mathsf{ctr}),(w_1^*,\mathsf{ctr})\}$ を満たすなら，トラップドア $\mathsf{KE.Trapdoor}(\mathsf{sk}_i,w)$ を返し，それ以外では $\perp$ を返すオラクル．

$\mathcal{A}$ がチャレンジ後は $\mathcal{O}_{\mathrm{KG}}$ にアクセスできないことに留意する．また，$\mathcal{A}$ は以下の制限を除き，自由にオラクルにアクセスできる．

- $\mathcal{O}_{\mathrm{KL}}$ にクエリする $i$ は $i < \mathsf{ctr}$ でなくてはならない．これは，現在の期間以外であれば，任意の過去の期間の秘密鍵漏洩を許していることを意味する．
- $\mathcal{O}_{\mathrm{TD}}$ にクエリする $(i,w)$ は $i \le \mathsf{ctr}$, かつ $w \notin \{w_0^*, w_1^*\}$ でなくてはならない．これは，$\mathcal{A}$ が可能な限りトラップドアを手に入れられること，すなわち攻撃者とクラウドの結託も想定していることを意味する．

**定義 2.1** (IND-KE-CKA)． 任意の確率的多項式時間攻撃者 $\mathcal{A}$ に対して，*KU-PEKS* $\Pi_{\mathrm{KE}}$ が $\mathsf{Adv}^{\mathsf{KE\text{-}CKA}}_{\Pi,\mathcal{A}}(1^\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{KE\text{-}CKA}}_{\Pi,\mathcal{A}}(1^\lambda)=1] - 1/2| < \epsilon(\lambda)$ を満たすとき，$\Pi_{\mathrm{KE}}$ は IND-KE-CKA を満たすという．

**計算量的一貫性.** 以下の試行 $\mathsf{Exp}^{\mathsf{KE\text{-}Cons}}_{\Pi_{\mathrm{KE}},\mathcal{A}}(1^\lambda)$ を考える．

$\mathsf{Exp}^{\mathsf{KE\text{-}Cons}}_{\Pi_{\mathrm{KE}},\mathcal{A}}(1^\lambda)$

$\mathsf{ctr} := 1,\ \ (\mathsf{pk}_1,\mathsf{sk}_1) \leftarrow \mathsf{KE.Setup}(1^\lambda)$

$(w_0^*, w_1^*, i^*state) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{KG}},\mathcal{O}_{\mathrm{KL}}}(\mathsf{pk}_1)$

$\mathsf{c}^{(0)}_{w_0^*,\mathsf{ctr}} \leftarrow \mathsf{KE.Enc}(\mathsf{pk}_{\mathsf{ctr}}, w_0^*)$

$\mathsf{td}_{w_1^*,i^*} \leftarrow \mathsf{KE.Trapdoor}(\mathsf{pk}_{i^*},\mathsf{sk}_{i^*},w_1^*)$

If $\left\{ \begin{array}{l} \mathsf{KE.Test}(\mathsf{td}_{w_1^*,i^*},\mathsf{c}_{w_0^*,\mathsf{ctr}}) = 1 \\ w_0^* \neq w_1^* \end{array} \right\}$

return 1 else return 0

**定義 2.2** (KE-Computational Consistency)． 任意の確率的多項式時間攻撃者 $\mathcal{A}$ に対して，*KU-PEKS* $\Pi_{\mathrm{KE}}$ が $\mathsf{Adv}^{\mathsf{KE\text{-}Cons}}_{\Pi_{\mathrm{KE}},\mathcal{A}}(1^\lambda) := \Pr[\mathsf{Exp}^{\mathsf{KE\text{-}Cons}}_{\Pi_{\mathrm{KE}},\mathcal{A}}(1^\lambda)=1] < \epsilon(\lambda)$ を満たすとき，$\Pi$ は計算量的一貫性 *(*KE-Computational Consistency*)* を満たすという．

### 2.3 構成方法

本節では，PKE と PEKS を用いた KU-PEKS の鍵更新モデルの一般的構成を説明する．基本的には [2] における構成と同じだが，検索キーワードをそのまま暗号化するのではなく，暗号学的ハッシュ関数 hash に入力したハッシュ値を暗号化する点が異なる．これは，この構成が再暗号化時に一度復号を行うため，キーワード自体がクラウドに知られてしまうことから，ハッシュ値に置き換えることで，ハッシュ関数の原像計算困難性から，キーワードに関する情報漏洩の軽減を図るものである．ただし，これはあくまで直感的な対策であることに留意されたい．この変更点に

ついては，2.4 節でも議論する．3 章での実装はこの構成法に基づいて行われている．

具体的には，公開鍵暗号 $\mathcal{PKE} = (\mathsf{PG}, \mathsf{G}, \mathsf{E}, \mathsf{D})$，検索可能暗号 $\mathcal{PEKS} = (\mathsf{Setup}_{\mathrm{PEKS}}, \mathsf{KeyGen}_{\mathrm{PEKS}}, \mathsf{Enc}_{\mathrm{PEKS}}, \mathsf{Trapdoor}_{\mathrm{PEKS}}, \mathsf{Test}_{\mathrm{PEKS}})$，及びハッシュ関数族 $\mathcal{H}_\lambda$ を用いて，KU-PEKS $\Pi_{\mathrm{KE}} = (\mathsf{KE.Setup}, \mathsf{KE.Upd}, \mathsf{KE.Enc}, \mathsf{KE.ReEnc}, \mathsf{KE.Trapdoor}, \mathsf{KE.Test})$ を以下のように構成する．以下では，キーワード集合 $\mathcal{W}$ と $\mathcal{PKE}$ の平文集合 $\mathcal{M}$ を同一視する．なお，公開鍵暗号，検索可能暗号，および暗号学的ハッシュ関数の定義は，付録に説明する．

- $\underline{\mathsf{KE.Setup}(1^\lambda)}$:

  $\mathsf{par}_{\mathrm{PKE}} \leftarrow \mathsf{PG}(1^\lambda)$
  $\mathsf{par}_{\mathrm{PEKS}} \leftarrow \mathsf{Setup}_{\mathrm{PEKS}}(1^\lambda)$
  $\mathsf{hash} \xleftarrow{\$} \mathcal{H}_\lambda$
  $(\mathsf{ek}_1, \mathsf{dk}_1) \leftarrow \mathsf{G}(\mathsf{par}_{\mathrm{PKE}})$
  $(\mathsf{mpk}_1, \mathsf{msk}_1) \leftarrow \mathsf{KeyGen}_{\mathrm{PEKS}}(\mathsf{par}_{\mathrm{PEKS}})$
  $\mathsf{pk}_1 := (\mathsf{par}_{\mathrm{PKE}}, \mathsf{par}_{\mathrm{PEKS}}, \mathsf{hash}, \mathsf{ek}_1, \mathsf{mpk}_1)$
  $\mathsf{sk}_1 := (\mathsf{dk}_1, \mathsf{msk}_1)$
  **return** $(\mathsf{pk}_1, \mathsf{sk}_1)$

- $\underline{\mathsf{KE.Upd}(\mathsf{pk}_i, \mathsf{sk}_i)}$:

  **parse** $\mathsf{pk}_i = (\mathsf{par}_{\mathrm{PKE}}, \mathsf{par}_{\mathrm{PEKS}}, \mathsf{hash}, \mathsf{ek}_i, \mathsf{mpk}_i)$
  **parse** $\mathsf{sk}_i = (\mathsf{dk}_i, \mathsf{msk}_i)$
  $(\mathsf{ek}_{i+1}, \mathsf{dk}_{i+1}) \leftarrow \mathsf{G}(\mathsf{par}_{\mathrm{PKE}})$
  $(\mathsf{mpk}_{i+1}, \mathsf{msk}_{i+1}) \leftarrow \mathsf{KeyGen}_{\mathrm{PEKS}}(\mathsf{par}_{\mathrm{PEKS}})$
  $\mathsf{pk}_{i+1} := (\mathsf{par}_{\mathrm{PKE}}, \mathsf{par}_{\mathrm{PEKS}}, \mathsf{hash}, \mathsf{ek}_{i+1}, \mathsf{mpk}_{i+1})$
  $\mathsf{sk}_{i+1} := (\mathsf{dk}_{i+1}, \mathsf{msk}_{i+1})$
  $\mathsf{rk}_{i \to i+1} := \mathsf{dk}_i$
  **return** $(\mathsf{pk}_{i+1}, \mathsf{sk}_{i+1}, \mathsf{rk}_{i \to i+1})$

- $\underline{\mathsf{KE.Enc}(\mathsf{pk}_i, w)}$:

  **parse** $\mathsf{pk}_i = (\mathsf{par}_{\mathrm{PKE}}, \mathsf{par}_{\mathrm{PEKS}}, \mathsf{hash}, \mathsf{ek}_i, \mathsf{mpk}_i)$
  $\mathsf{ct}_i \leftarrow \mathsf{E}(\mathsf{ek}_i, \mathsf{hash}(w))$
  $\mathsf{ct}_{w,i} \leftarrow \mathsf{Enc}_{\mathrm{PEKS}}(\mathsf{mpk}_i, \mathsf{hash}(w))$
  $\mathsf{c}_{w,i}^{(0)} := (\mathsf{ct}_i, \mathsf{ct}_{w,i})$
  **return** $\mathsf{c}_{w,i}^{(0)}$

- $\underline{\mathsf{KE.ReEnc}(\mathsf{pk}_{i+1}, \mathsf{rk}_{i \to i+1}, \mathsf{c}_{w,j}^{(k)})}$:

  **parse** $\mathsf{pk}_{i+1} = (\mathsf{par}_{\mathrm{PKE}}, \mathsf{par}_{\mathrm{PEKS}}, \mathsf{hash}, \mathsf{ek}_{i+1}, \mathsf{mpk}_{i+1})$
  **parse** $\mathsf{rk}_{i \to i+1} = (\mathsf{ek}_{i+1}, \mathsf{mpk}_{i+1}, \mathsf{dk}_i)$
  **parse** $\mathsf{c}_{w,i}^{(0)} = (\mathsf{ct}_i, \mathsf{ct}_{w,i})$
  **if** $i \neq j + k$
    **return** $\perp$
  **else**
    $\mathsf{hash}(w) \leftarrow \mathsf{D}(\mathsf{dk}_i, \mathsf{ct}_i)$
    $\mathsf{c}_{w,j}^{(k+1)} \leftarrow \mathsf{KE.Enc}(\mathsf{pk}_{i+1}, \mathsf{hash}(w))$
    **return** $\mathsf{c}_{w,j}^{(k+1)}$

- $\underline{\mathsf{KE.Trapdoor}(\mathsf{pk}_i, \mathsf{sk}_i, w')}$:

  **parse** $\mathsf{pk}_i = (\mathsf{par}_{\mathrm{PKE}}, \mathsf{par}_{\mathrm{PEKS}}, \mathsf{hash}, \mathsf{ek}_i, \mathsf{mpk}_i)$
  **parse** $\mathsf{sk}_i = (\mathsf{dk}_i, \mathsf{msk}_i)$
  $\mathsf{t}_{w',i} \leftarrow \mathsf{Trapdoor}_{\mathrm{PEKS}}(\mathsf{mpk}_i, \mathsf{msk}_i, \mathsf{hash}(w'))$
  **return** $\mathsf{t}_{w',i}$

- $\underline{\mathsf{KE.Test}(\mathsf{pk}_i, \mathsf{t}_{w',i}, \mathsf{c}_{w,j}^{(k)})}$:

  **parse** $\mathsf{pk}_i = (\mathsf{par}_{\mathrm{PKE}}, \mathsf{par}_{\mathrm{PEKS}}, \mathsf{hash}, \mathsf{ek}_i, \mathsf{mpk}_i)$
  **parse** $\mathsf{c}_{w,j}^{(k)} = (\mathsf{ct}_{j+k}, \mathsf{ct}_{w,j+k})$
  **if** $i \neq j + k$
    **return** 0
  **else if** $1 \leftarrow \mathsf{Test}_{\mathrm{PEKS}}(\mathsf{mpk}_i, \mathsf{t}_{w',i}, \mathsf{ct}_{w,i})$
    **return** 1
  **else if** $0 \leftarrow \mathsf{Test}_{\mathrm{PEKS}}(\mathsf{mpk}_i, \mathsf{t}_{w',i}, \mathsf{ct}_{w,i})$
    **return** 0

上記構成法の正当性は，$\mathcal{PKE}$, $\mathcal{PEKS}$, 及び $\mathcal{H}$ の衝突困難性から容易に確認できる．

**定理 2.1.** $\mathcal{PKE}$ が IND-CPA を満たし，$\mathcal{PEKS}$ が IND-CKA を満たし，ハッシュ関数族 $\mathcal{H}$ が衝突困難性を満たすならば，上記構成法による *KU-PEKS* $\Pi$ は IND-KE-CKA を満たす．

証明の概要． [2] の Theorem 1 とほとんど同様に証明可能であるため，ここでは相違点だけ説明する．IND-KE-CKA ゲームにおいて，攻撃者 $\mathcal{A}$ が $\mathsf{hash}(w_0^*) = \mathsf{hash}(w_1^*)$ となるような $(w_0^*, w_1^*)$ をチャレンジとして用いるイベントを考えると，そのイベントが起きる時の攻撃成功確率は，ハッシュ関数族の衝突困難性に帰着することができる．イベントが起きない場合の攻撃成功確率は [2] の Theorem 1 と同様に証明可能である． □

### 2.4 考察

ハッシュ関数を導入することにより，次の効果が期待できる．また，基とした構成法にハッシュ関数を導入しても，検索キーワードがそのハッシュ値に替わるだけであり，衝突困難性を有することから，同様に IND-KE-CKA 安全性が証明できる．なお，以下の効果はあくまで直感的なものであり，理論的に証明ができているわけではないことに留意されたい．

(1) クラウドに対しハッシュ値だけが開示
従来の構成では，クラウドでの再暗号化の KE.ReEnc において，検索キーワード $w$ が直接導出されていた．安全性定義は過去の秘密鍵が全て漏洩した場合までカバーするものとなっており，結果として「再暗号化される前の暗号文（最新の暗号文）」に対しての安全性を保証するものであるため，この事実は定義と矛盾しな

い．しかしながら，本来秘密鍵が漏洩した時に，初めてクラウドに過去の暗号文内のキーワードが知られてしまうべきところを，再暗号化の際に検索キーワードが漏洩してしまう嫌悪感があった．今回の構成では，検索キーワードそのものではなく，そのハッシュ値 $\mathsf{hash}(w)$ が導出される．一般的にハッシュ関数が有する原像計算困難性から，クラウドが元のキーワードを導出することは難しいと考えられ，このことにより，利用者の嫌悪感が緩和されるものと考える．

(2) Keyword Guessing 攻撃への対応

Keyword Guessing 攻撃は，トラップドア値と，攻撃者が試行生成した暗号文を KE.Test に入力することで，検索キーワード $w'$ を求める攻撃である．この攻撃はクラウドによる攻撃を考慮すると，原理的に回避することはできないことが知られている．従来の KU-PEKS の構成では，再暗号化した暗号文内のキーワードがクラウドに知られているため，KE.Test の出力を確認することによって，クラウドが暗号文を試行生成することなく，検索キーワード $w'$ を求めることができる．従って，通常の PEKS と比べて，Keyword guessing 攻撃が容易と考えられる．今回の構成では，検索キーワード $w'$ のハッシュ値 $\mathsf{hash}(w')$ までしかわからず，そこから $w'$ を求めるための試行が必要となる．ハッシュ関数の原像計算困難性から，$w'$ を求めることは難しいため，Keyword Guessing 攻撃にかかるコストが，通常の PEKS と同程度になると考えられる．

## 3. 実装結果

本章では，文献 [2] で提案した KU-PEKS の 3 つの具体的方式（(1)$\Pi^{\mathsf{rom}}_{\mathrm{KE}}$，(2)$\Pi^{\mathsf{std}}_{\mathrm{KE}}$，(3)$\Pi^{\mathsf{std}}_{\mathrm{KI}}$）に対して，今回の hash を導入した実装について報告する．なお，(3) については，机上の評価のみとする．

表 1 は，各具体的方式の特性と，各パラメータのデータ量比較を示す．(1)$\Pi^{\mathsf{rom}}_{\mathrm{KE}}$ は，Boneh-Franklin の ID ベース暗号を応用した検索可能暗号と，ElGamal 公開鍵暗号を組み合わせて実現している．hash 関数としては，SHA を用いて，その出力を $\mathbb{G}_1$ 上の楕円曲線上の点にマッピングしている．また，(2)$\Pi^{\mathsf{std}}_{\mathrm{KE}}$ は，Jutla-Roy らの ID ベース暗号を応用した検索可能暗号と，ElGamal 公開鍵暗号を組み合わせて実現している．hash 関数としては，SHA を用いて，その出力を $\mathbb{G}_T$（Tepla では，12 次拡大体）上の楕円曲線上の点にマッピングしている．さらに，(3)$\Pi^{\mathsf{std}}_{\mathrm{KI}}$ は，渡邉-四方 [6] の鍵隔離暗号と，ID ベース暗号を応用した鍵隔離型の検索可能暗号を組み合わせて実現している．hash 関数として，SHA を用いて，その出力を $\mathbb{G}_T$（Tepla では，12 次拡大体）上の楕円曲線上の点にマッピングしている．

(1) は，ランダムオラクルモデル上での安全性を確保し，最もパラメータのデータ量が少なく効率のよい方法である．一方，(3) は，(1)(2) に比べるとパラメータのデータ量が多くなるものの，公開鍵の配布や管理などが不要であり，鍵更新も時刻で管理するため，運用が容易で実用に近い方式といえる．

表 2 は，IoT での実装を想定し，各関数の，ラズベリーパイ上での処理時間を示す．なお，ここでの報告は (1)(2) のみとし，(3) については，後日報告する．計測の環境は，Raspberry Pi 3 model B, CPU: Broadcom BCM2837 (1.2GHz), RAM: 1GB, OS: Linux (Raspbian 9.1), Libraries: TEPLA 2.0, GMP 5.1.3 である．なお，TEPLA[5] は筑波大学が開発したオープンソースの C 言語暗号ライブラリである．楕円曲線暗号の演算や，ペアリング演算などを含む．

図 1 では，(1)(2) の各関数の処理時間を比較する．(2) は，(1) に比べると，受信クライアントの処理である KE.Trapdoor に時間がかかっているものの，絶対値としては，300msec 弱であり，十分実用的な数字であると考える．

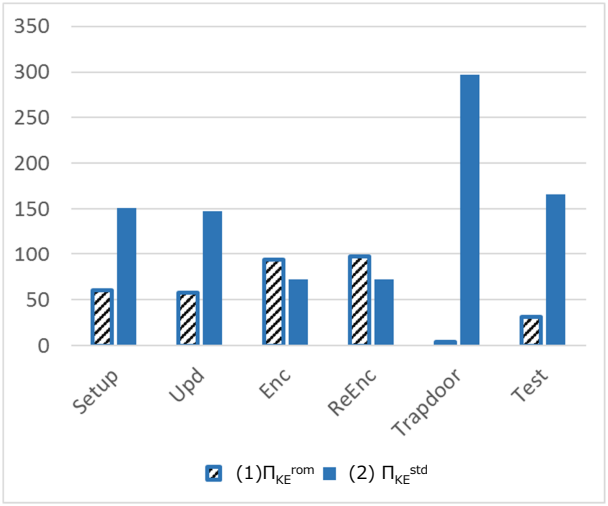


図 1 各関数の処理時間の比較（msec/1 回）

## 4. さらなる効率化に向けての工夫

前の章では，IoT 機器側における各関数の処理時間を報告した．一方，クライアントから検索を行い，クラウドからの結果が戻ってくる一連の処理の中，処理時間がかかるのは，検索対象の暗号文が多い場合，クラウド側の処理である．ここでは，クラウド側の KE.Test 関数の処理回数削減を目的として，検索時に，クライアント側から，KE.Trapdoor とともに，$\mathsf{hash}(w')$ の値をクラウドに開示する方法を提示する．クラウド側では，古い暗号文を，KE.ReEnc 時に導出される $\mathsf{hash}(w)$ を用いてタグ付けして保存しておく．そして，古い暗号文は，検索時に開示される $\mathsf{hash}(w')$ の値を用いて照合する．KE.Test 関数を用いて，一致を確認するのは，最新の公開鍵で暗号化した暗号文のみとなるため，結果としてクラウド側の検索処理を高速化できる．

クライアントが KE.Trapdoor 値に加えて，$\mathsf{hash}(w')$ を開

表 1　具体方式の特性と，各パラメータのデータ量比較
#pk, #sk, #rk, #td, #c は，それぞれ公開鍵，秘密鍵，再暗号化鍵， トラップドア，暗号文のサイズを示す．サイズ $[a, b, c, d]$ は，$\mathbb{Z}_p$ 上の要素が $a$ 個，$\mathbb{G}_1$ 上の要素が $b$ 個，$\mathbb{G}_2$ の要素が $c$ 個，$\mathbb{G}_T$ の要素が $d$ 個あることを示す．また，Asmp は安全性の仮定を示す

| | pk is fixed? | #pk | #sk | #rk | #td | #c | Asmp. |
|---|---|---|---|---|---|---|---|
| (1)$\Pi_{\mathrm{KE}}^{\mathsf{rom}}$ | No | $[0,4,0,0]$ | $[2,0,0,0]$ | $[1,0,0,0]$ | $[0,1,0,0]$ | $[2,3,0,0]$ | DDH1, DBDH |
| (2)$\Pi_{\mathrm{KE}}^{\mathsf{std}}$ | No | $[0,7,0,1]$ | $[9,0,1,0]$ | $[1,0,0,0]$ | $[0,0,5,0]$ | $[1,5,0,1]$ | SXDH |
| (3)$\Pi_{\mathrm{KI}}^{\mathsf{std}}$ | Yes | $[0,13,7,1]$ | $[8,0,17,0]$ | $[0,0,O(t),0]$ | $[0,0,5,0]$ | $[2t+1,3t+3,0,t+1]$ | SXDH |

表 2　具体方式の性能比較（msec ／ 1 回）

| | Setup | Upd | Enc | ReEnc | Trapdoor | Test |
|---|---|---|---|---|---|---|
| (1)$\Pi_{\mathrm{KE}}^{\mathsf{rom}}$ | 60.30 | 57.21 | 93.40 | 97.25 | 3.68 | 31.15 |
| (2)$\Pi_{\mathrm{KE}}^{\mathsf{std}}$ | 151.45 | 147.83 | 72.32 | 72.16 | 297.16 | 165.76 |

示することに伴う安全性の劣化については，今後検討する．

## 5. まとめ

本稿では，かねて提案の鍵更新機能付きの検索可能暗号に，hash 関数を追加導入した．実装上の工夫を行い，実装結果を報告した．

**謝辞**　本研究の一部は，JSPS 科研費（第一，第二著者，第三著者においては JP17K00189，第四著者においては JP16J10532 と JP17K12697）の助成を受けています．

### 参考文献

[1] Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P. and Shi, H.: Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions, *J. Cryptology*, Vol. 21, No. 3, pp. 350–391 (2008).

[2] Anada, H., Kanaoka, A., Matsuzaki, N. and Watanabe, Y.: Key-Updatable Public-Key Encryption with Keyword Search: Models and Generic Constructions, *Information Security and Privacy* (Susilo, W. and Yang, G., eds.), Cham, Springer International Publishing, pp. 341–359 (2018).

[3] Boneh, D. and Franklin, M. K.: Identity-Based Encryption from the Weil Pairing, *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '01, London, UK., Springer-Verlag, pp. 213–229 (online), available from ⟨http://dl.acm.org/citation.cfm?id=646766.704155⟩ (2001).

[4] Jutla, C. S. and Roy, A.: Shorter Quasi-Adaptive NIZK Proofs for Linear Subspaces, *Advances in Cryptology – ASIACRYPT 2013* (Sako, K. and Sarkar, P., eds.), Lecture Notes in Computer Science, Vol. 8269, Springer Berlin Heidelberg, pp. 1–20 (2013).

[5] of Tsukuba, U.: TEPLA: University of Tsukuba Elliptic, Curve and Pairing Libraary (Jan.2013 Released TEPLA 1.0, Dec-2015 Released TEPLA 2.0).

[6] Watanabe, Y. and Shikata, J.: Identity-based Hierarchical Key-insulated Encryption without Random Oracles, Cryptology ePrint Archive, Report 2015/1254 (2015).

[7] 松崎なつめ，穴田啓晃：検索可能暗号の鍵更新についての調査，*Symposium on Cryptography and Information Security (SCIS2017)* (2017).

[8] 松崎なつめ，穴田啓晃，渡邉洋平：鍵更新可能な検索可能暗号の一提案：検索可能代理人再暗号化の適用について，*IEICE Technical Report ISEC2017-1* (2017).

[9] 松崎なつめ，穴田啓晃，渡邉洋平：鍵更新機能付き検索可能暗号：公開鍵更新モデルによる実現，*Computer Security Symposium 2017* (2017).

[10] 渡邉洋平，穴田啓晃，松崎なつめ：鍵更新機能付き検索可能暗号：鍵隔離モデルによる実現，*Computer Security Symposium 2017* (2017).

## 付　　録

### A.1 公開鍵暗号

公開鍵暗号 (Public Key Encryption: PKE) $\mathcal{PKE} = (\mathsf{PG}, \mathsf{G}, \mathsf{E}, \mathsf{D})$ は以下のように定義される．

- $\mathsf{PG}(1^\lambda) \to \mathsf{par}_{\mathrm{PKE}}$: セキュリティパラメータ $1^\lambda$ を入力として，公開パラメータ $\mathsf{par}_{\mathrm{PKE}}$ を出力する．
- $\mathsf{G}(\mathsf{par}_{\mathrm{PKE}}) \to (\mathsf{dk}, \mathsf{ek})$: 公開パラメータ $\mathsf{par}_{\mathrm{PKE}}$ を入力として，鍵ペア（復号鍵 dk と暗号化鍵 ek）を出力する．
- $\mathsf{E}(\mathsf{ek}, m) \to \mathsf{ct}$: 暗号化鍵 ek と平文 $m \in \mathcal{M}_{\mathrm{PKE}}$ を入力して，暗号文 ct を出力する．$\mathcal{M}$ はセキュリティパラメータによって決まる平文集合である．
- $\mathsf{D}(\mathsf{dk}, \mathsf{ct}) \to m$ or $\perp$: 復号鍵 dk と暗号文 ct を入力として，$m$ または復号失敗を表す $\perp$ を出力する．

上記モデルは以下の正当性を要求する．全ての $\lambda \in \mathbb{N}$, $\mathsf{par}_{\mathrm{PKE}} \leftarrow \mathsf{PG}(1^\lambda)$, $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{G}(\mathsf{par}_{\mathrm{PKE}})$, 任意の $m \in \mathcal{M}$ に

対して，$\mathsf{D}(\mathsf{dk}, \mathsf{E}(\mathsf{ek}, m)) = m$.

本稿で扱う安全性である選択平文攻撃に対する識別不可能性 (IND-CPA) について，以下の試行 $\mathsf{Exp}^{\mathsf{CPA}}_{\mathcal{PKE},\mathcal{A}}(1^\lambda)$ を考える．

$\mathsf{Exp}^{\mathsf{CPA}}_{\mathcal{PKE},\mathcal{A}}(1^\lambda)$

$\mathsf{par}_{\mathrm{PKE}} \leftarrow \mathsf{PG}(1^\lambda),\ \ (\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{G}(\mathsf{par}_{\mathrm{PKE}})$

$(m_0^*, m_1^*, state) \leftarrow \mathcal{A}(\mathsf{par}_{\mathrm{PKE}}, \mathsf{ek})$ s.t. $|m_0^*| = |m_1^*|$

$b \xleftarrow{\$} \{0,1\},\ \ \mathsf{ct}_b^* \leftarrow \mathsf{E}(\mathsf{ek}, m_b^*)$

$b' \leftarrow \mathcal{A}(state, \mathsf{ct}_b^*)$

If $b' = b$ return 1 else return 0

**定義 A.1.1** (IND-CPA)．任意の確率的多項式時間攻撃者 $\mathcal{A}$ に対して，$\mathcal{PKE}$ が $\mathsf{Adv}^{\mathsf{CPA}}_{\mathcal{PKE},\mathcal{A}}(1^\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{CPA}}_{\mathcal{PKE},\mathcal{A}}(1^\lambda) = 1] - 1/2| < \epsilon(\lambda)$ を満たすとき，$\mathcal{PKE}$ は IND-CPA を満たすという．

## A.2 検索可能暗号

検索可能暗号 (Public Key Encryption with Keyword Search: PEKS) $\mathcal{PEKS} = (\mathsf{Setup}_{\mathrm{PEKS}}, \mathsf{KeyGen}_{\mathrm{PEKS}}, \mathsf{Enc}_{\mathrm{PEKS}}, \mathsf{Trapdoor}_{\mathrm{PEKS}}, \mathsf{Test}_{\mathrm{PEKS}})$ は以下のように定義される．

- $\mathsf{Setup}_{\mathrm{PEKS}}(1^\lambda) \to \mathsf{par}_{\mathrm{PEKS}}$: セキュリティパラメータ $1^\lambda$ を入力として，公開パラメータ $\mathsf{par}_{\mathrm{PEKS}}$ を出力する．
- $\mathsf{KeyGen}_{\mathrm{PEKS}}(\mathsf{par}_{\mathrm{PEKS}}) \to (\mathsf{msk}, \mathsf{mpk})$: 公開パラメータを $\mathsf{par}_{\mathrm{PEKS}}$ を入力として，鍵ペア（秘密鍵 msk と公開鍵 mpk）を出力する．
- $\mathsf{Enc}_{\mathrm{PEKS}}(\mathsf{mpk}, w) \to \mathsf{ct}_w$: 公開鍵 mpk と検索キーワード $w \in \mathcal{W}$ を入力として，検索用の暗号文（暗号 index）$\mathsf{ct}_w$ を出力する．$\mathcal{W}$ はセキュリティパラメータによって決まるキーワード集合である．
- $\mathsf{Trapdoor}_{\mathrm{PEKS}}(\mathsf{msk}, w') \to \mathsf{td}_{w'}$: 秘密鍵 msk と検索キーワード $w' \in \mathcal{W}$ を入力として，トラップドア $\mathsf{td}_{w'}$ を出力する．
- $\mathsf{Test}_{\mathrm{PEKS}}(\mathsf{td}_{w'}, \mathsf{ct}_w) \to 1$ or 0: 暗号文 $\mathsf{ct}_w$ とトラップドア $\mathsf{t}_{w'}$ を入力として，もし $w = w'$ であれば 1 を出力する．そうでなければ 0 を出力する．

上記モデルは以下の正当性を要求する．全ての $\lambda \in \mathbb{N}$, 全ての $\mathsf{par}_{\mathrm{PEKS}} \leftarrow \mathsf{Setup}_{\mathrm{PEKS}}(1^\lambda)$, 全ての $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{KeyGen}_{\mathrm{PEKS}}(\mathsf{par}_{\mathrm{PEKS}})$, 全ての $w \in \mathcal{W}$ に対して，

$\mathsf{Test}_{\mathrm{PEKS}}(\mathsf{Trapdoor}_{\mathrm{PEKS}}(\mathsf{msk}, w), \mathsf{Enc}_{\mathrm{PEKS}}(\mathsf{mpk}, w)) = 1.$

本稿で扱う安全性である選択キーワード攻撃に対する識別不可能性 (IND-CKA) について，次の試行 $\mathsf{Exp}^{\mathsf{CKA}}_{\mathcal{PEKS},\mathcal{A}}(1^\lambda)$ を考える．

$\mathsf{Exp}^{\mathsf{CKA}}_{\mathcal{PEKS},\mathcal{A}}(1^\lambda)$

$\mathsf{par}_{\mathrm{PEKS}} \leftarrow \mathsf{Setup}_{\mathrm{PEKS}}(1^\lambda)$

$(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{KeyGen}_{\mathrm{PEKS}}(\mathsf{par}_{\mathrm{PEKS}})$

$(w_0^*, w_1^*, state) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{TD}}}(\mathsf{par}_{\mathrm{PEKS}}, \mathsf{mpk})$ s.t. $|w_0^*| = |w_1^*|$

$b \xleftarrow{\$} \{0,1\},\ \ \mathsf{ct}_{w_b^*}^* \leftarrow \mathsf{Enc}_{\mathrm{PEKS}}(\mathsf{mpk}, w_b^*)$

$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{TD}}}(state, \mathsf{ct}_{w_b^*}^*)$

If $b' = b$ return 1 else return 0

$\mathcal{A}$ はオラクル $\mathcal{O}_{\mathrm{TD}}$ にアクセスできる．$\mathcal{O}_{\mathrm{TD}}$ は，$\mathcal{A}$ から $w$ を受け取り，$\mathsf{td}_w \leftarrow \mathsf{Trapdoor}_{\mathrm{PEKS}}(\mathsf{msk}, w)$ を返すオラクルであり，$w \notin \{w_0^*, w_1^*\}$ でなくてはならない．これは，$\mathcal{A}$ が可能な限りトラップドアを手に入れられること，すなわち攻撃者とクラウドの結託を想定していることを意味する．

**定義 A.2.1** (IND-CKA [1])． 任意の確率的多項式時間攻撃者 $\mathcal{A}$ に対して，$\mathcal{PEKS}$ が $\mathsf{Adv}^{\mathsf{CKA}}_{\mathcal{PEKS},\mathcal{A}}(1^\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{CKA}}_{\mathcal{PEKS},\mathcal{A}}(1^\lambda) = 1] - 1/2| < \epsilon(\lambda)$ を満たすとき，$\mathcal{PEKS}$ は IND-CKA を満たすという．

また，以下の計算量的一貫性 (Computational Consistency) を考える．以下の試行 $\mathsf{Exp}^{\mathsf{Cons}}_{\mathcal{PEKS},\mathcal{A}}(1^\lambda)$ を考える．

$\mathsf{Exp}^{\mathsf{Cons}}_{\mathcal{PEKS},\mathcal{A}}(1^\lambda)$

$\mathsf{par}_{\mathrm{PEKS}} \leftarrow \mathsf{Setup}_{\mathrm{PEKS}}(1^\lambda)$

$(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{KeyGen}_{\mathrm{PEKS}}(\mathsf{par}_{\mathrm{PEKS}})$

$(w_0^*, w_1^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{TD}}}(\mathsf{par}_{\mathrm{PEKS}}, \mathsf{mpk})$ s.t. $|w_0^*| = |w_1^*|$

$\mathsf{ct}_{w_0^*}^* \leftarrow \mathsf{Enc}_{\mathrm{PEKS}}(\mathsf{mpk}, w_0^*)$

$\mathsf{td}_{w_1^*} \leftarrow \mathsf{Trapdoor}_{\mathrm{PEKS}}(\mathsf{msk}, w_1^*)$

If $\mathsf{Test}_{\mathrm{PEKS}}(\mathsf{td}_{w_1^*}, \mathsf{ct}_{w_0^*}) = 1$ and $w_0^* \neq w_1^*$ return 1

else return 0

IND-CKA 同様，$\mathcal{A}$ はオラクル $\mathcal{O}_{\mathrm{TD}}$ に同様の制限の下でアクセスできる．

**定義 A.2.2** (Computational Consistency[1])．任意の確率的多項式時間攻撃者 $\mathcal{A}$ に対して，$\mathcal{PEKS}$ が $\mathsf{Adv}^{\mathsf{Cons}}_{\mathcal{PEKS},\mathcal{A}}(1^\lambda) := \Pr[\mathsf{Exp}^{\mathsf{Cons}}_{\mathcal{PEKS},\mathcal{A}}(1^\lambda) = 1] < \epsilon(\lambda)$ を満たすとき，$\mathcal{PEKS}$ は Computational Consistency を満たすという．

## A.3 ハッシュ関数

ハッシュ関数族 $\mathcal{H}_\lambda$ は任意長の文字列を短い固定長の文字列に圧縮する関数 $\mathsf{hash} : \mathcal{X} \to \mathcal{Y}$ の族であり，そのサイズはセキュリティパラメータ $\lambda$ によって決まる．以下の試行 $\mathsf{Exp}^{\mathsf{CR}}_{\mathcal{H},\mathcal{A}}(1^\lambda)$ を考える．

$\mathsf{Exp}^{\mathsf{CR}}_{\mathcal{H},\mathcal{A}}(1^\lambda)$

$\mathsf{hash} \xleftarrow{\$} \mathcal{H}_\lambda$

$(x, x^*) \leftarrow \mathcal{A}(\mathsf{hash})$

If $(x, x^*) \in \mathcal{X}^2$ and $x \neq x^*$ and $\mathsf{hash}(x) = \mathsf{hash}(x^*)$

then return 1 else return 0

**定義 A.3.1** (衝突困難性)**.** 任意の確率的多項式時間攻撃者 $\mathcal{A}$ に対して，$\mathcal{H}_\lambda$ が $\mathsf{Adv}^{\mathsf{CR}}_{\mathcal{H},\mathcal{A}}(1^\lambda) := \Pr[\mathsf{Exp}^{\mathsf{CR}}_{\mathcal{H},\mathcal{A}}(1^\lambda) = 1] < \epsilon(\lambda)$ を満たすとき，$\mathcal{H}_\lambda$ は衝突困難性を持つという**.**