ファイルの安全な追加・削除・検索が可能な暗号システム

国立研究開発法人情報通信研究機構
サイバーセキュリティ研究所
セキュリティ基盤研究室　研究員

渡邉 洋平

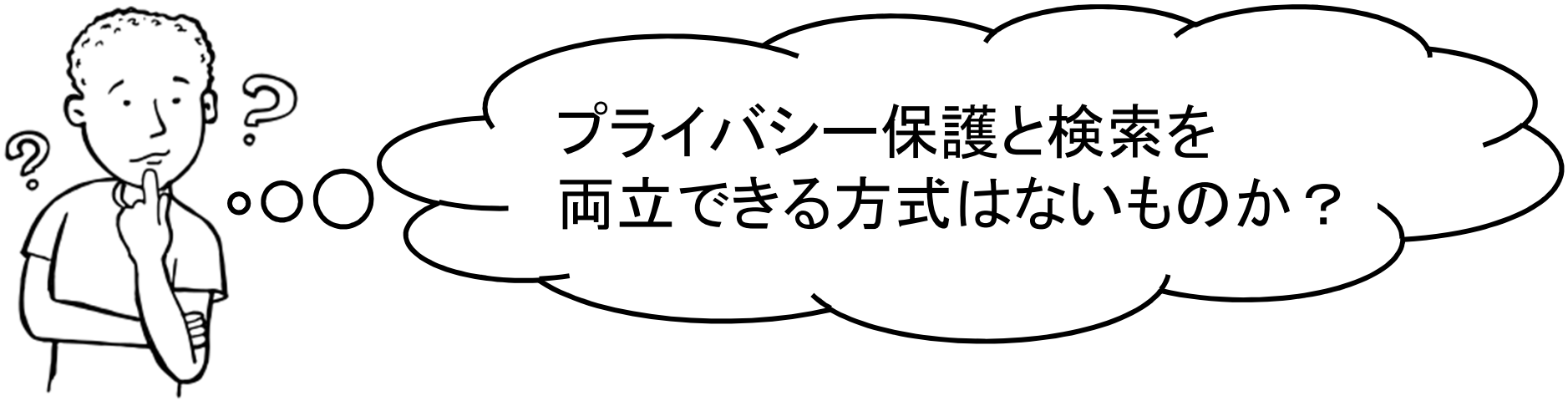
2019年7月18日

先端情報技術による恩恵と脅威

- データベース管理システム (DBMS)
  - データの取込，保存，検索，取出
  - 検索サイト，ショッピングサイト等，幅広く利用
- ビッグデータ技術等の発展
  - データ解析による恩恵
  - データの中央集権化や価値・種類の増大
  - データのプライバシーを考える必要性も増大
- データベースを狙った攻撃も多数報告 (以下NYT記事)
  - 政府：Hacking Linked to China Exposes Millions of U.S. Workers
  - 民間企業：9 Recent Cyberattacks Against Big Businesses

素朴な対策

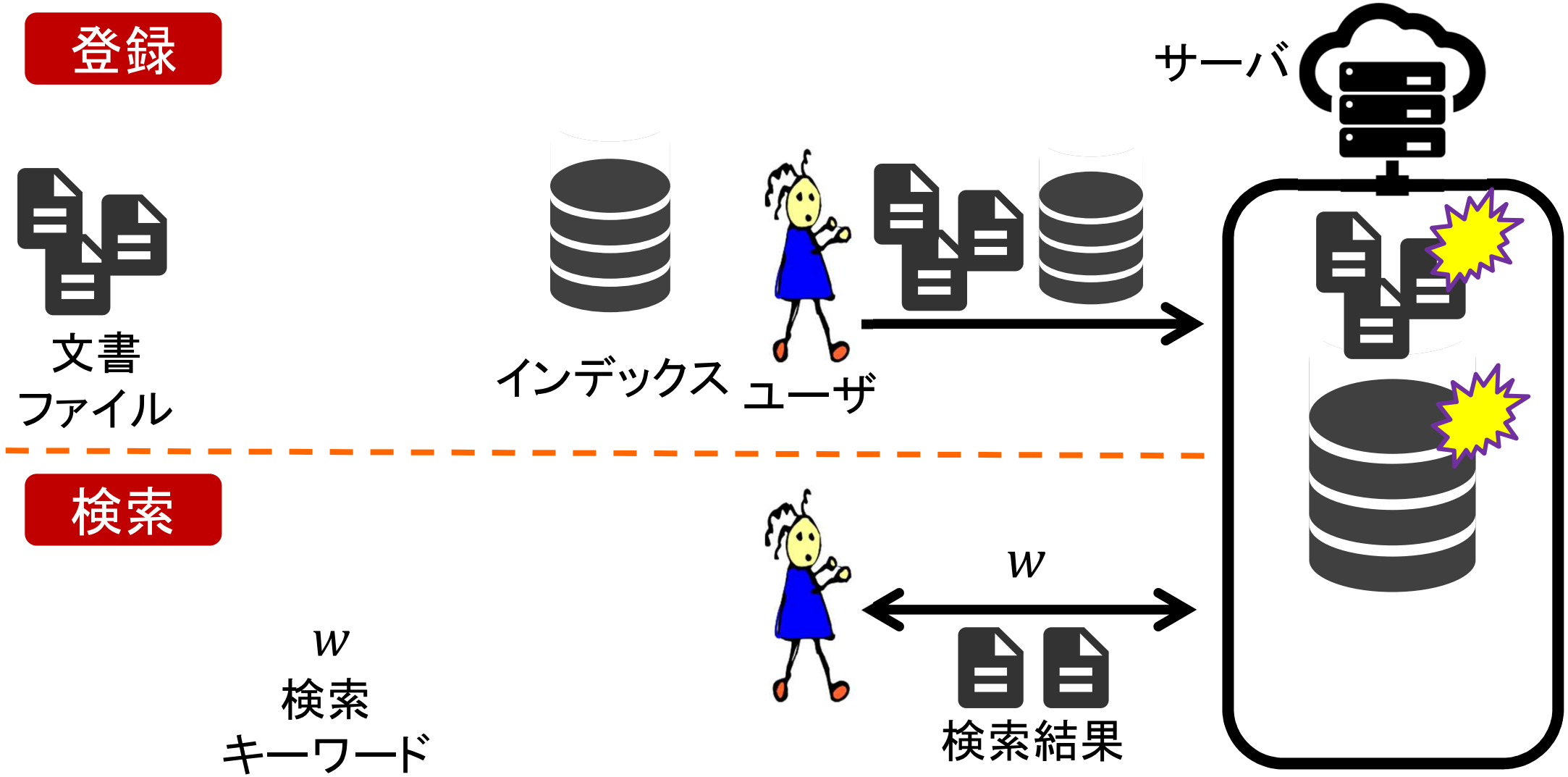
- データベースの暗号化
  - データのプライバシーは守られる ☺
  - 検索等, これまで可能だった操作が不可能になる ☹



検索可能暗号の登場

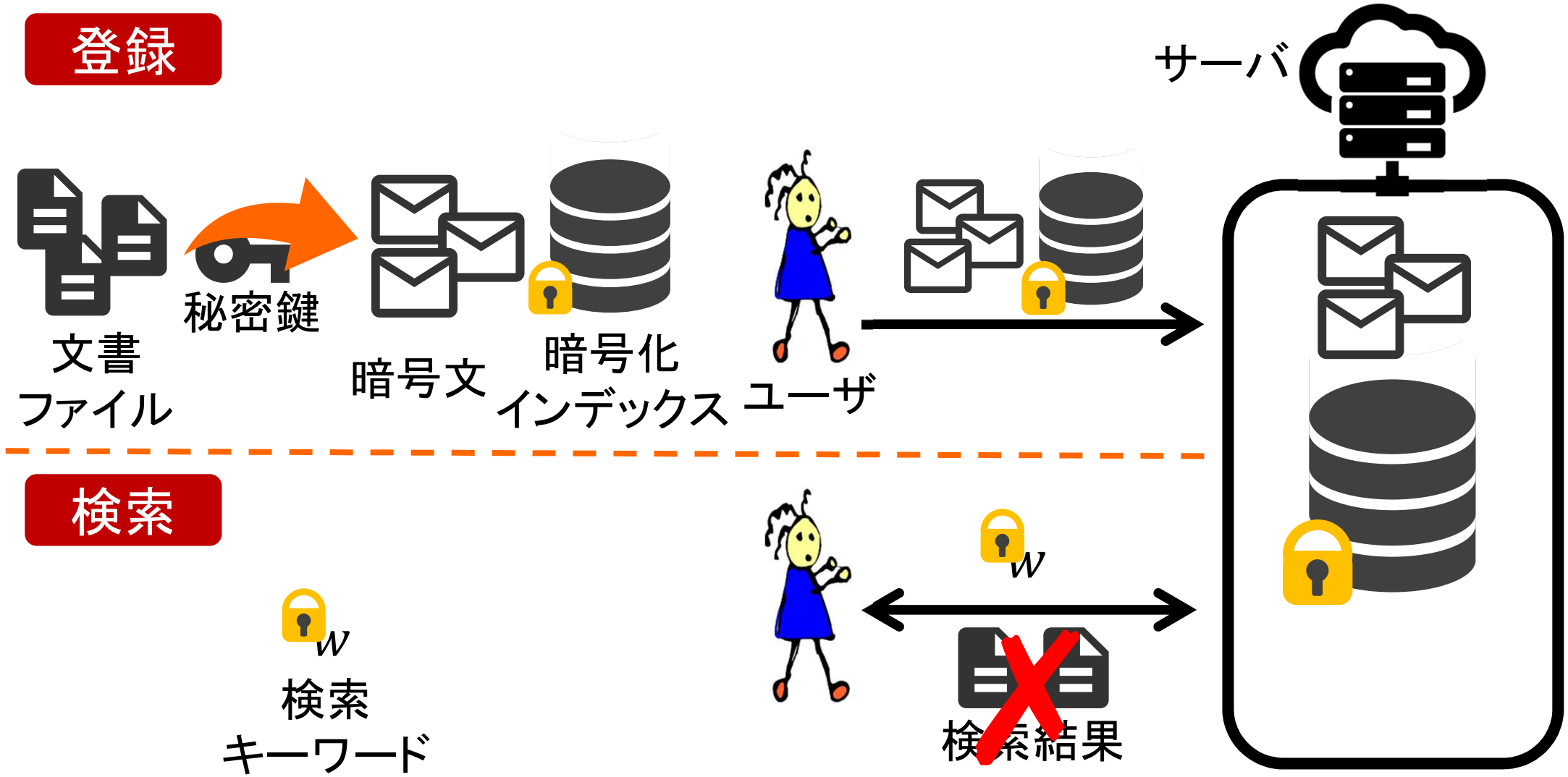
通常のデータ検索

- ファイルやインデックスをそのまま保存
- ファイルの内容や検索キーワードがサーバに漏洩 ☹



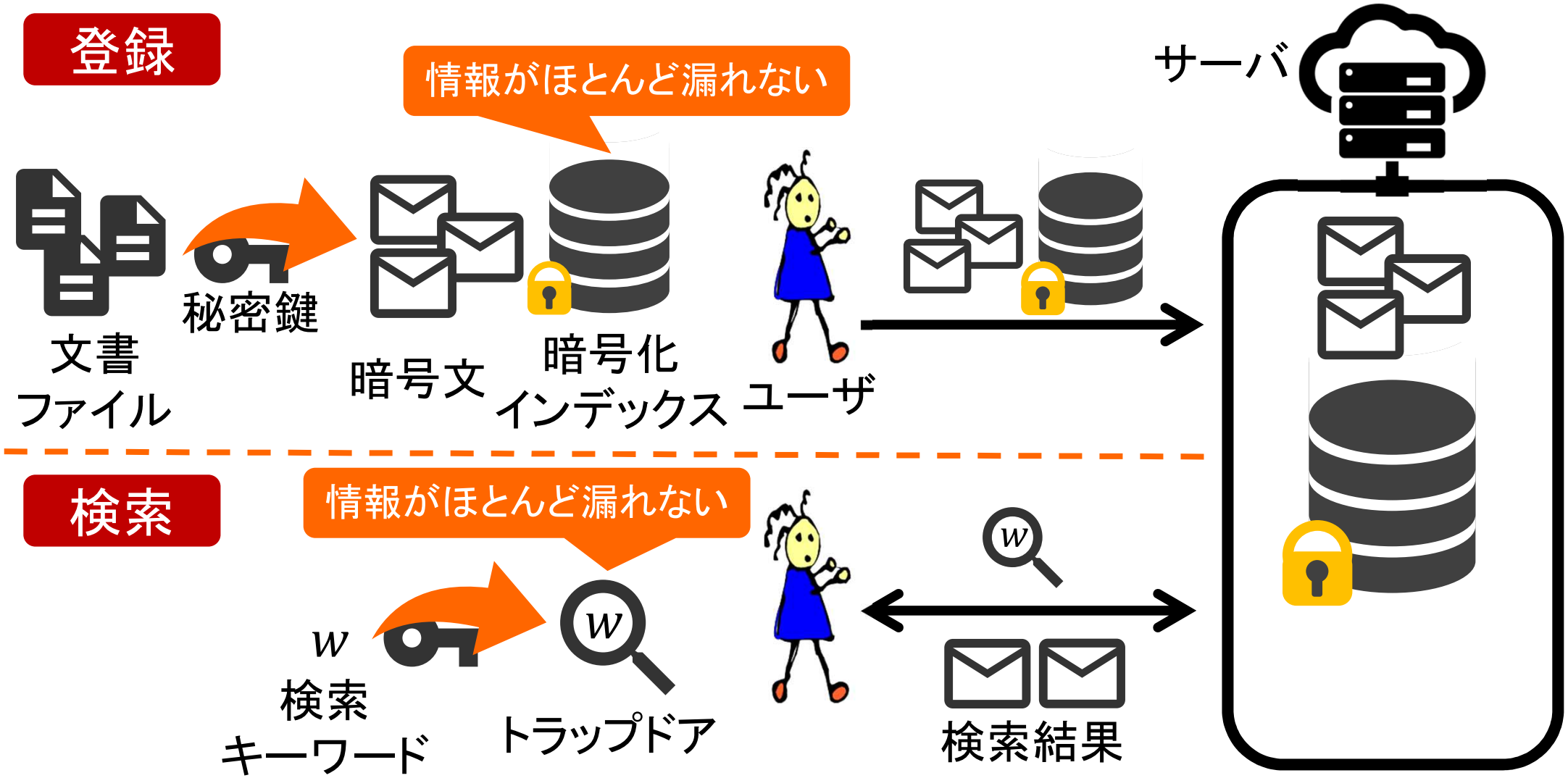
素朴な対策：暗号化

- ファイルやインデックス，検索キーワードを暗号化
- 乱数化されるため，上手く検索できなくなる ☹



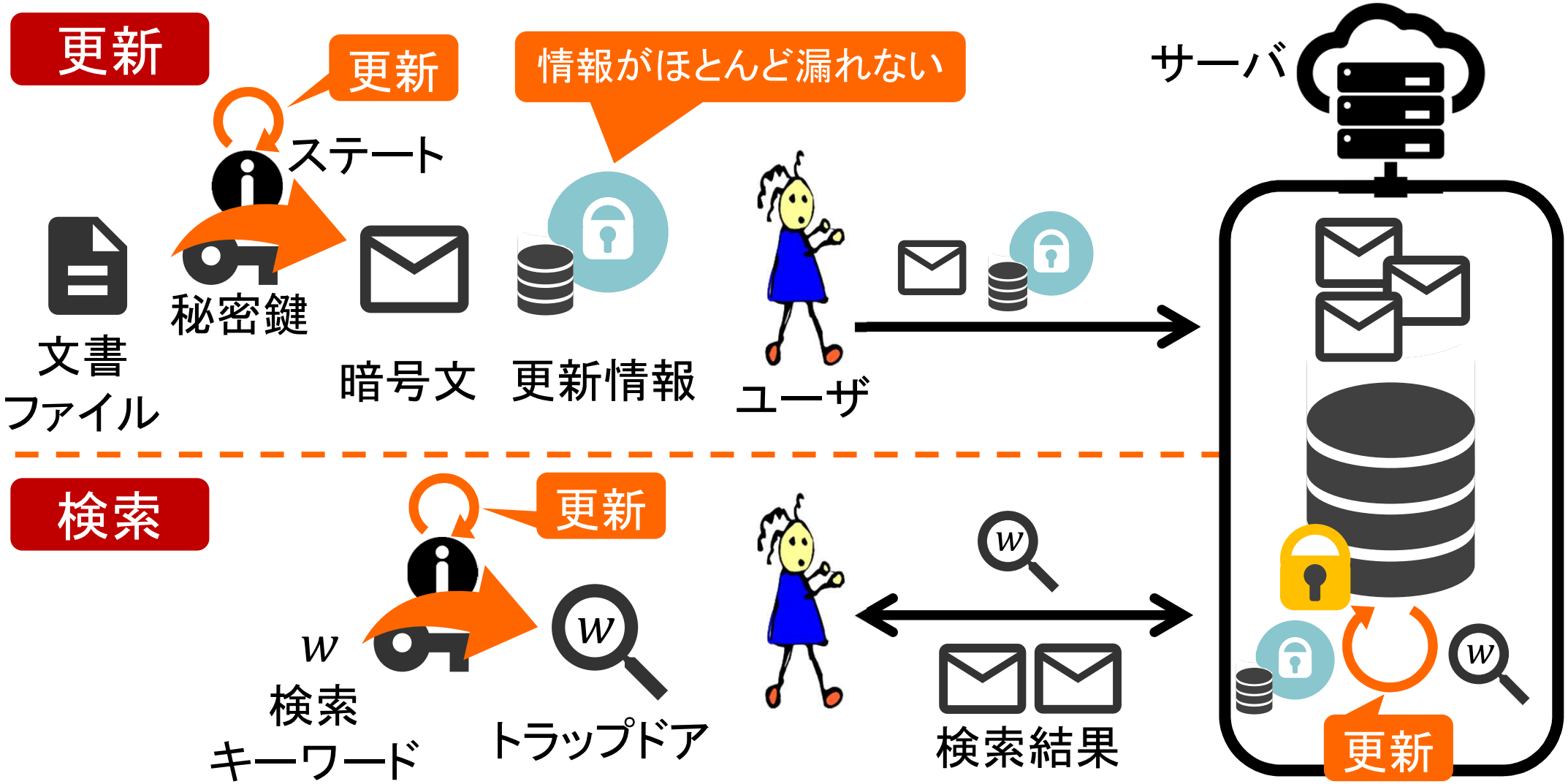
検索可能暗号 (Symmetric Searchable Encryption: SSE) [CGKO06]

- 暗号化したまま検索を可能とする暗号技術
- 可能な限り情報を漏らさずに検索を実現 ☺



動的検索可能暗号 (Dynamic SSE: DSSE) [KPR12]

- 動的にファイルを追加・削除できる検索可能暗号
- ユーザ側もステート情報 (定期更新される情報) を利用



漏洩を許す情報，許さない情報

- **DSSEでは多少の“取るに足らない”情報の漏洩を許す**
  - どの暗号文が検索にヒットしたかがわかる
    - それらが同じキーワードを含んでいることがわかる
    - 検索キーワード自体はわからない
  - 同じキーワードの検索には前回と同じ処理がなされる
    - 処理内容から過去にも検索されたキーワードだとわかる
- **必ず守るべき情報は何か？**
  - ファイルの内容
    - 以降，ファイル中の単語全てをキーワードとみなす
  - 検索キーワード

DSSEが保証する安全性

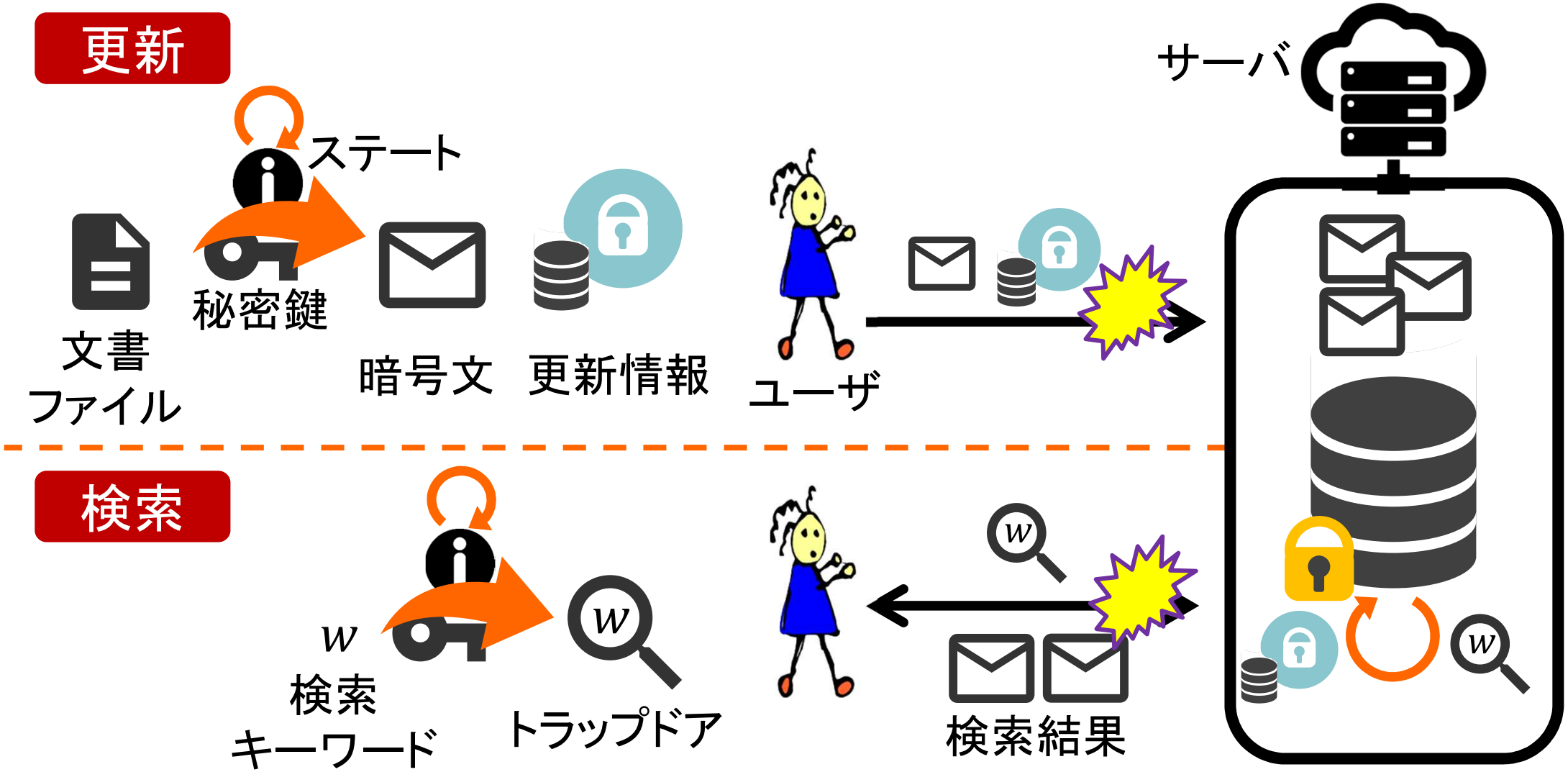
次の“漏洩を許す情報”以外，ファイルやキーワードに関する情報が全く漏れないことを保証する

- **検索時に漏洩する情報**
  - 検索結果 (どの暗号化ファイルが返されたか)
  - 検索履歴 (過去のどの時点で同キーワードが検索されたか)
  - ＋α
- **更新時 (特にファイル追加時) に漏洩する情報**
  - ファイルの長さ
  - ファイルが含む (異なる) キーワード数

  } フォワード安全性
  - 過去に検索されたキーワードの中でファイルが含むもの
  - ＋α

DSSEで漏れる情報の一例

- 追加：サイズが100KBで200種の単語を含むファイルであること
- 検索：キーワード w の検索履歴と検索結果 (w 自体は漏れない)



従来技術とその問題点

1. **効率性：ステート情報が大きい**
   - データベース上の (異なる) キーワード数に依存
   - 一般的に，キーワード数 ＞ 登録ファイル数
2. **実現可能性：次のいずれかの制約が必要**
   - 検索または更新時により多くの情報が漏洩
   - (比較的効率が悪い) 公開鍵暗号技術を利用して構成
   - 検索時のキーワードの再登録等，追加の手順が必要
3. **安全性：漏洩情報の妥当性が不十分**
   - “漏洩を許す情報” は本当に漏洩を許していい情報なのか？

新技術の概要・特長

1. **効率性：ステート情報が小さい**
   - データベース上の登録ファイル数のみに依存
   - 本技術を応用してステート情報不要な方式も実現可能
2. **実現可能性：これまでの制約も必要なし**
   - 必要最小限の情報漏洩のみ
   - 効率の良い共通鍵暗号技術のみで実現
   - 追加の手順が不要なシンプルな構成
3. **安全性：より強い安全性を達成可能**
   - 情報漏洩を更に抑えることが可能

利便性及び安全性の両方が向上

新技術の概要・特長

1. **効率性：ステート情報が小さい**
   - データベース上の登録ファイル数のみに依存
   - 本技術を応用してステート情報不要な方式も実現可能

2. **実現可能性：これまでの制約も必要なし**
   - 必要最小限の情報漏洩のみ
   - 効率の良い共通鍵暗号技術のみで実現
   - 追加の手順が不要なシンプルな構成

3. **安全性：より強い安全性を達成可能**
   - 情報漏洩を更に抑えることが可能

利便性及び安全性の両方が向上

従来技術とその問題点：より詳細に

方式	ステート情報長	暗号化インデックスサイズ	検索コスト	更新コスト	備考
SPS14	$O(\sqrt{N})$	$O(N)$	$O(n_w \log^3 N)$	$O(\mu_{\mathrm{id}} \log^2 N)$	検索時により多くの情報が漏洩
Bos16	$O(d)$	$O(N)$	$O\left(n_w^{(a)} + n_w^{(d)}\right)$	$O(\mu_{\mathrm{id}})$	公開鍵暗号技術を利用
KKL+17	$O(d)$	$O(N)$	$O\left(n_w^{(a)}\right)$	$O(\mu_{\mathrm{id}})$	検索時により多くの情報が漏洩
EKPE18	$O(d+n)$	$O(N)$	$O\left(\left(n_w + n_w^{(s)}\right)/p\right)$	$O(\mu_{\mathrm{id}}/p)$	検索時にキーワードの再登録が必要

1. ステート情報長がキーワード数 d に依存
   - 一般的に，キーワード数 d > ファイル数 n
   - データベースサイズ $N >> d > n$
   - ファイル数に依存するのが望ましい

2. 強い道具・仮定，余計な手順が必要
   - 情報漏洩は極力少なくしたい
   - 公開鍵暗号技術は重いので避けたい
   - シンプルな構成が望ましい

DSSEの代表的な構成アプローチ

- 検索用の暗号化配列を暗号化インデックスとして構成
  - アドレス：検索用トラップドア
  - 格納値：検索結果

w や id に依存

id に依存

アドレス	格納値
$addr_1$	val_1
$addr_2$	val_2
$addr_3$	val_3
$addr_4$	val_4
⋮	⋮

インデックス

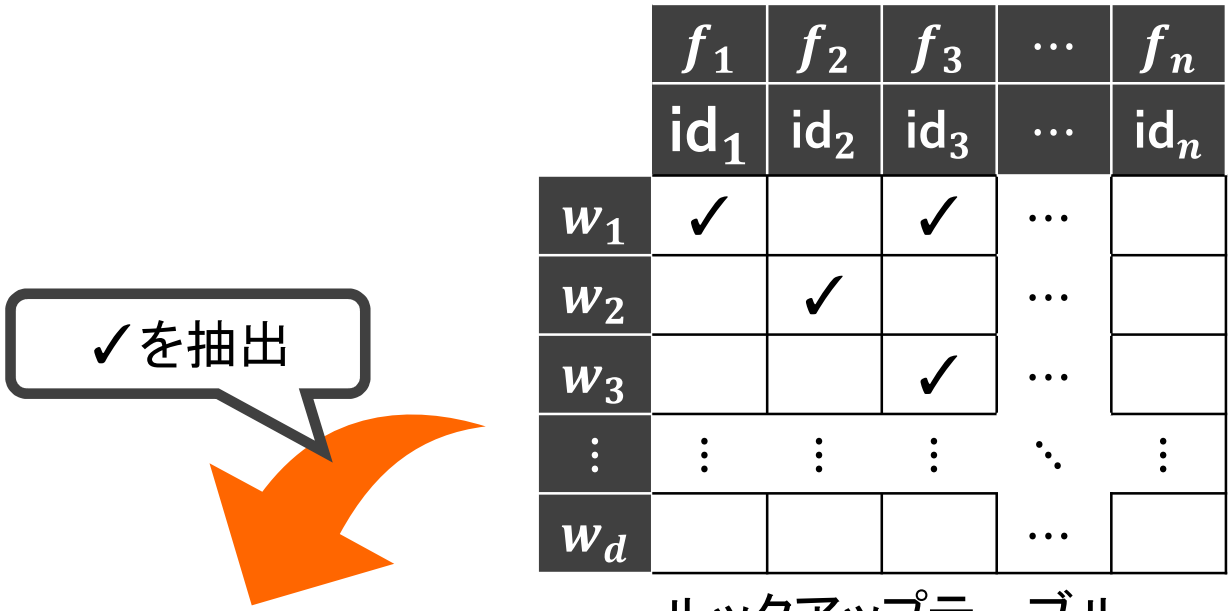
疑似ランダム置換 π で乱数化
- 文字列を乱数に変換
- 入力が異なれば出力される乱数も異なる

アドレス	格納値
$\pi_k(addr_1)$	val_1
$\pi_k(addr_2)$	val_2
$\pi_k(addr_3)$	val_3
$\pi_k(addr_4)$	val_4
⋮	⋮

暗号化インデックス

アドレスをどう作るかがポイント

本技術の構成アプローチ



アドレス	格納値
$w_1 \| \| \mathrm{id}_1$	id_1
$w_1 \| \| \mathrm{id}_3$	id_3
$w_2 \| \| \mathrm{id}_2$	id_2
$w_3 \| \| \mathrm{id}_3$	id_3
⋮	⋮

インデックス

擬似ランダム置換 π で乱数化

アドレス	格納値
$\pi_k(w_1 \| \| \mathrm{id}_1)$	id_1
$\pi_k(w_1 \| \| \mathrm{id}_3)$	id_3
$\pi_k(w_2 \| \| \mathrm{id}_2)$	id_2
$\pi_k(w_3 \| \| \mathrm{id}_3)$	id_3
⋮	⋮

暗号化インデックス

本技術における検索方法

キーワード q の検索用トラップドア T_q は暗号化インデックスのアドレスに対応

例：

ステート情報：登録ファイルの識別子
$\sigma = \{id_1, id_2, \ldots, id_n\}$

トラップドア $T_q := \begin{matrix} \pi_k(q||id_1) \\ \pi_k(q||id_2) \\ \vdots \\ \pi_k(q||id_n) \end{matrix}$

アドレス	識別子
$\pi_k(q\|\|id_1)$	id_1
$\pi_k(q\|\|id_3)$	id_3
$\pi_k(w\|\|id_2)$	id_2
$\pi_k(\omega\|\|id_3)$	id_3
⋮	⋮

暗号化インデックス

検索結果 $X_q := \{id_1, id_3\}$

応用：ステート情報を用いない検索

ステート情報$\sigma = \{\mathrm{id}_1, \mathrm{id}_2, \ldots, \mathrm{id}_n\}$をサーバに預ける (公開する) ことが可能

- ステート情報の同期が不要 (様々なデバイスからのアクセスが容易)
- 既存方式は公開不可能 (安全性が崩れる)

例：

request

$\sigma = \{\mathrm{id}_1, \mathrm{id}_2, \ldots, \mathrm{id}_n\}$

トラップドア $\mathrm{T}_q :=$ $\pi_k(q||\mathrm{id}_1)$, $\pi_k(q||\mathrm{id}_2)$, ⋮, $\pi_k(q||\mathrm{id}_n)$

検索結果 $\mathrm{X}_q := \{\mathrm{id}_1, \mathrm{id}_3\}$

$\sigma = \{\mathrm{id}_1, \mathrm{id}_2, \ldots, \mathrm{id}_n\}$

アドレス	識別子
$\pi_k(q\|\|\mathrm{id}_1)$	id_1
$\pi_k(q\|\|\mathrm{id}_3)$	id_3
$\pi_k(w\|\|\mathrm{id}_2)$	id_2
$\pi_k(\omega\|\|\mathrm{id}_3)$	id_3
⋮	⋮

暗号化インデックス

従来技術と本技術の構成アプローチの違い

	アドレスの作成方法	ステート情報
本技術	$\pi(k, w\|\|\mathrm{id})$	データベース中の全てのファイルのid (ファイル数に依存)
EKPE18	$\pi(f(k, w\|\|\mathrm{sc}_w), \mathrm{fc}_w)$	データベース中の全てのキーワードの $(\mathrm{sc}_w, \mathrm{fc}_w)$ (キーワード数に依存)

πの鍵kはずっと固定

検索後のアドレスの再登録は必要なし

πの鍵$f(k, w||\mathrm{sc}_w)$が(w, sc_w)で変化

sc_wは検索されるたびに増える
→検索後にアドレスの再登録が必要

sc_w: wが今まで検索された回数
fc_w: wを含んでいるファイルの数

一般的に
ファイル数＜キーワード数
→本発明の方が効率的

新技術の概要・特長 (再掲)

1. 効率性:ステート情報が小さい
   - データベース上の登録ファイル数のみに依存
   - 本技術を応用してステート情報不要な方式も実現可能
2. 実現可能性:これまでの制約も必要なし
   - 必要最小限の情報漏洩のみ
   - 効率の良い共通鍵暗号技術のみで実現
   - 追加の手順が不要なシンプルな構成
3. 安全性:より強い安全性を達成可能
   - 情報漏洩を更に抑えることが可能

利便性及び安全性の両方が向上

再考：更新時に漏洩を許す情報

- 既存のDSSE方式は，更新時に以下の情報を漏洩する：
  - ファイル f の長さ $|f|$
  - ファイル f に含まれる異なるキーワードの数

この情報は本当に漏洩して問題ないのだろうか？

情報漏洩が危惧される一例：STR法

- DNAデータベース上におけるDSSEの利用を想定
- STR (Short Tandem Repeat) 法
  - STR: DNA中に繰り返し現れる短い配列
  - STRの反復数は個人によって異なり，両親から遺伝される
  - 親子鑑定や犯罪捜査で利用
- 実際，FBIがDNAデータベースCODISでSTR法を利用

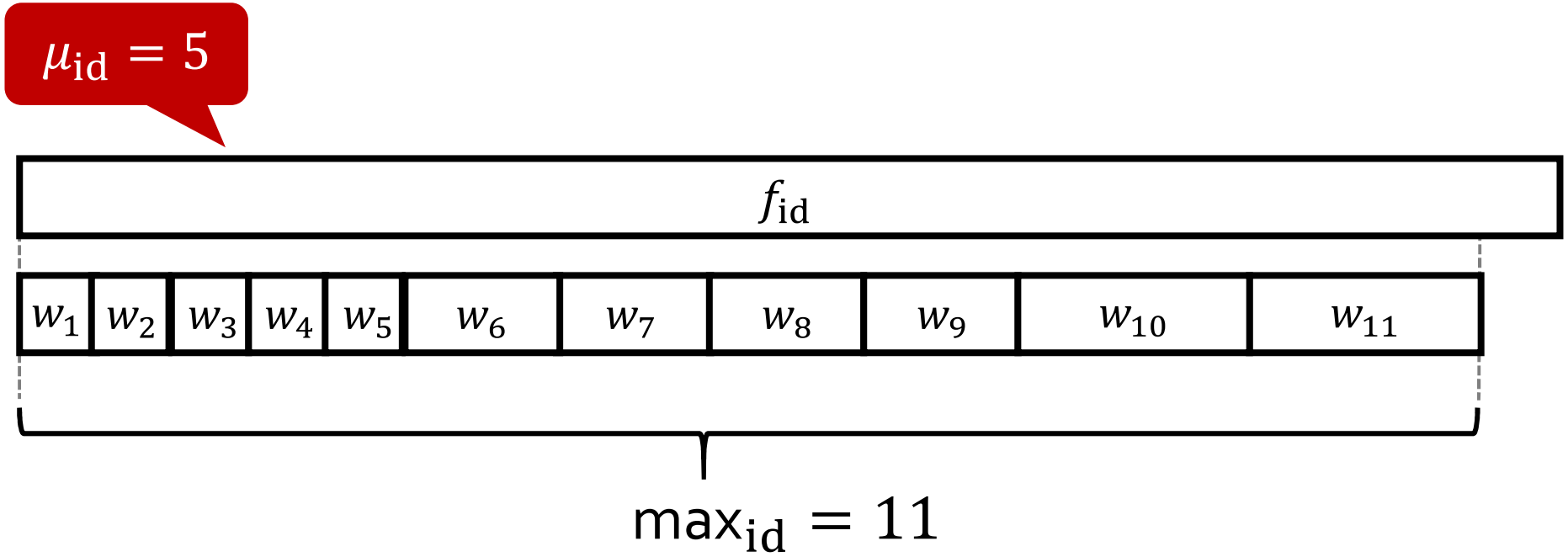
暗号化されたDNAサンプルのキーワード数が少ない

➡ STRが多く繰り返されているかもしれない

➡ 重要な情報が漏洩する可能性

より強い安全性に向けて

- $\max_{\mathrm{id}}$: ファイル f_{id} が含むことのできる最大のキーワードの個数



$\max_{\mathrm{id}}$個になるまでidのダミーエントリを追加することで
ファイル中のキーワード数を漏洩しない，
より強い安全性を満たす方式を実現する

従来技術との比較

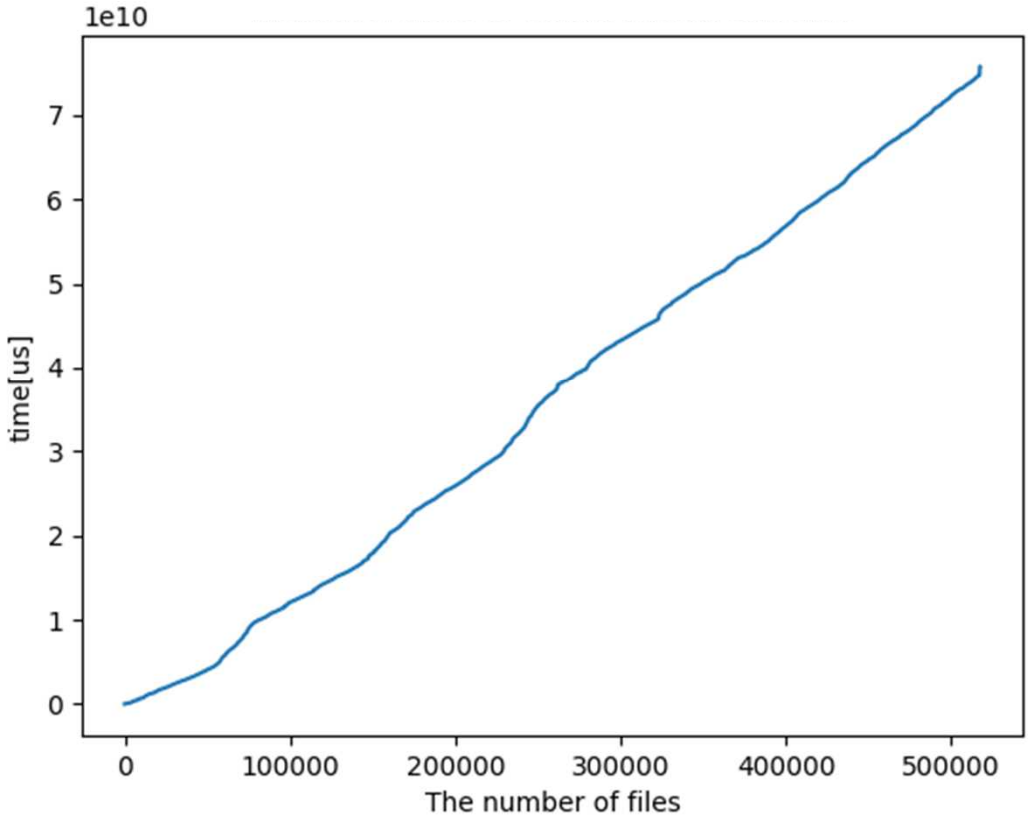
- **ステート情報のサイズを削減**
  - 例：Wikipediaベースのデータセット
    - ファイル数: 2,835,729個，キーワード数: 9,801,551個
    - EKPE18 (両方に依存): 約52MB
    - 本技術 (ファイル数のみに依存): 約14.6MB
  - 通信回数を増やすことでステート情報不要にもできる

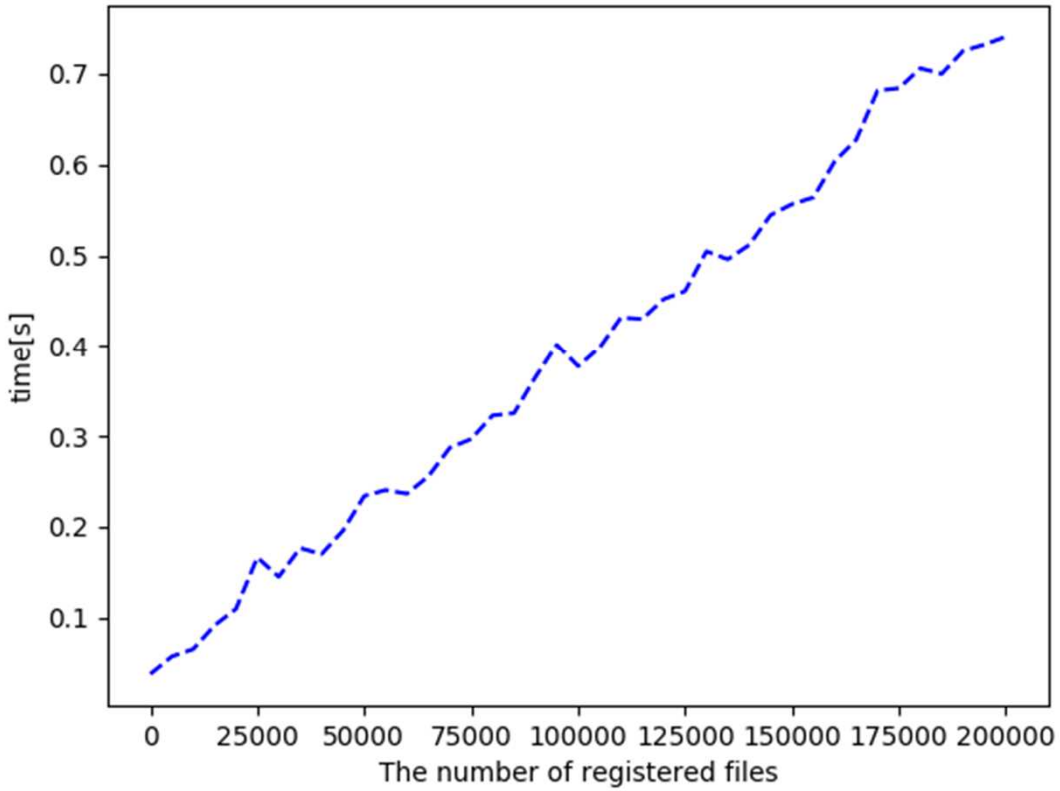
72%削減！

- **漏洩情報，利用技術，実行手順に制約が不要**
  - 構成がシンプルゆえに実装も容易
  - 必要な暗号技術はAESのみ
- **従来よりも強い安全性を達成可能**
  - より幅広いユースケースに対応可能

実装結果

- Enronの提供するメールデータセットを利用した実装実験
- 50万ファイルの登録に約75秒
- 20万ファイルの検索に約0.7秒

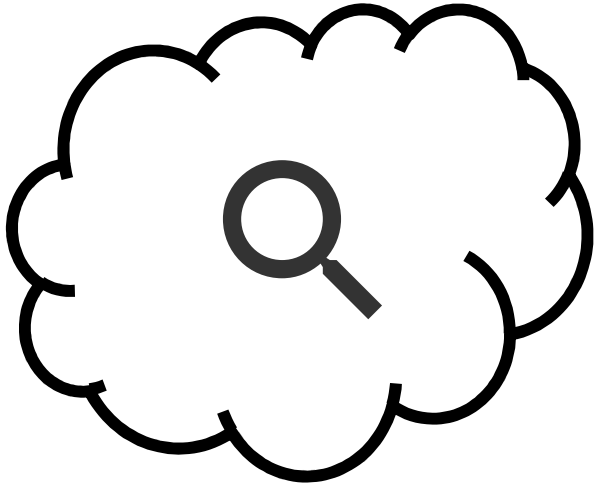




本技術の想定される用途

- 機密性の高い情報を取り扱うデータベース
- 検索内容を保護すべきアプリケーション

メールサービス



クラウドストレージ

DNAデータベース

これ以外に，検索を主としないシステムも応用先として考えられる

本技術による効果

- 利用者側からは何も効果が見えない
  - むしろ不便さや，重くなったと感じるかもしれない
- コストを犠牲にしてでもプライバシーを保護する意味
  - 情報漏洩時のインパクトや利用者の安心を考慮
  - 情報漏洩はなくならず，大部分はヒューマンエラーが原因
    - 2018年の個人情報漏洩インシデントの総被害者数: 561万人
    - インシデント一件当たりの漏洩人数: 1万3千人
- プライバシー保護が前提のシステムには必須
  - 顧客情報データベース
  - DNAデータベース

実用化に向けた課題

- より現実的な実験環境での性能の検討
  - 実際に近い状況で追加・削除が繰り返された後の検索効率
  - データベースの負荷分散を含めた設計
  - AES-NI等の専用命令を持たないデバイスでの性能
  - メールデータ以外での性能
  - 障害耐性の検討
- アプリケーションごとの最適化
  - 何をキーワードとするか
  - どこまでの安全性があればよいのか
  - 想定されるデバイスに応じた設計

企業への期待

- 「実用化に向けた課題」の解決に向けた共同研究・開発
  - 本技術をベースとしたシステムの実装実験
  - 想定するデータ・環境の下での性能評価
  - 有効性を確認した上での実用化

- 本技術の導入が有効だと思われる企業
  - 何かしらのデータベースを提供・管理する企業
  - その他，検索機能が必要なサービスを提供する企業

本技術に関する知的財産権

- 発明の名称
  - 動的検索可能暗号処理システム及び動的検索可能暗号処理方法
- 出願番号
  - 2019-003908
- 出願人
  - 国立研究開発法人情報通信研究機構
  - 国立大学法人電気通信大学
- 発明者
  - 渡邉 洋平 (情報通信研究機構)
  - 岩本 貢 (電気通信大学)
  - 太田 和夫 (電気通信大学)

お問い合わせ先

国立研究開発法人情報通信研究機構
イノベーション推進部門
技術移転コーディネータ　宇梶、橘田

TEL　　:042－327 － 6950
FAX　　:042－327 － 6659
e-mail　　: ippo@ml.nict.go.jp

参考文献 (非特許文献)

[Bos16] R. Bost: Σοφος – Forward Secure Searchable Encryption. In: CCS 2016.

[CGKO06] R. Curtmola, J. Garay, S. Kamara, and R. Osrtovsky: Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In: CCS 2006.

[EKPE18] M. Etemad, A. Küpçü, C. Papamanthou, and D. Evans: Efficient Dynamic Searchable Encryption with Forward Privacy. In: PETS 2018.

[KPR12] S. Kamara, C. Papamanthou, and T. Roeder: Dynamic Searchable Symmetric Encryption. In: CCS 2012.

[KKL+17] K.S. Kim, M. Kim, D. Lee, J.H. Park, and W.-H. Kim: Forward Secure Dynamic Searchable Symmetric Encryption with Efficient Updates. In: CCS 2017.

[SPS14] E. Stefanov, C. Papamanthou, and E. Shi: Practical Dynamic Searchable Encryption with Small Leakage. In: NDSS 2014.

記号の説明

- N: データベースサイズ
- d: キーワード数
- n: ファイル数
- n_w: wを含むファイル数
- $n_w^{(a)}$ ($n_w^{(d)}$): wを含むファイルの累計追加 (削除) 回数
  - 一般的に, $N > d > n \geq n_w = n_w^{(a)} - n_w^{(d)}$
- $n_w^{(s)}$: 前回の検索から今までに削除された, wを含むファイル数
- μ_{id}: ファイルf_{id}が含む異なるキーワード数
- p: プロセッサ数 (pが表中に登場する＝並列処理可能)