Bloom Filters

SCS Seminar: March 07, 2025
Florian

Agenda

What are Bloom Filters and Theoretical Analysis
My first contact: Searchable Symmetric Encryption
Application II: Private Record Linkage
Extensions for Bloom Filters
Keyword PIR

Motivation

$\in$?

Motivation

$\in$?

1	1	0	0	0	0	1	0	0	0	0	…

   …   

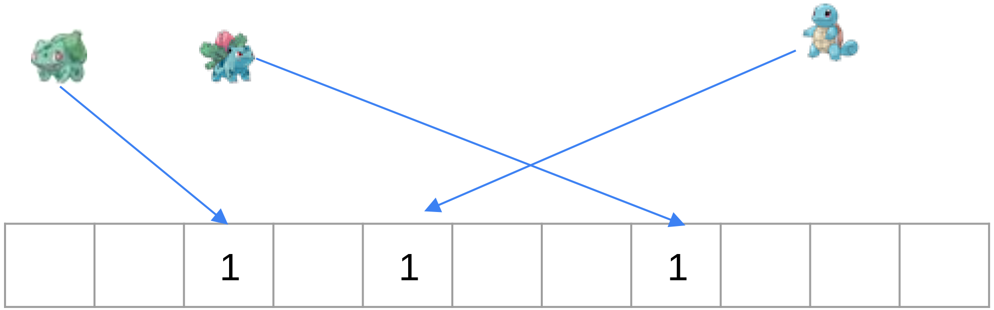
There are 1025 Pokémon nowadays, so we need ~ 1kb to encode this dictionary…

Motivation

Wikipedia: Bloom filter is a space-efficient probabilistic data structure, [...] that is used to test whether an element is a member of a set.

Set dictionary size to m
Use hash function $h(\cdot)$

For each element:
  Set bit at position $h(\cdot) \bmod m$



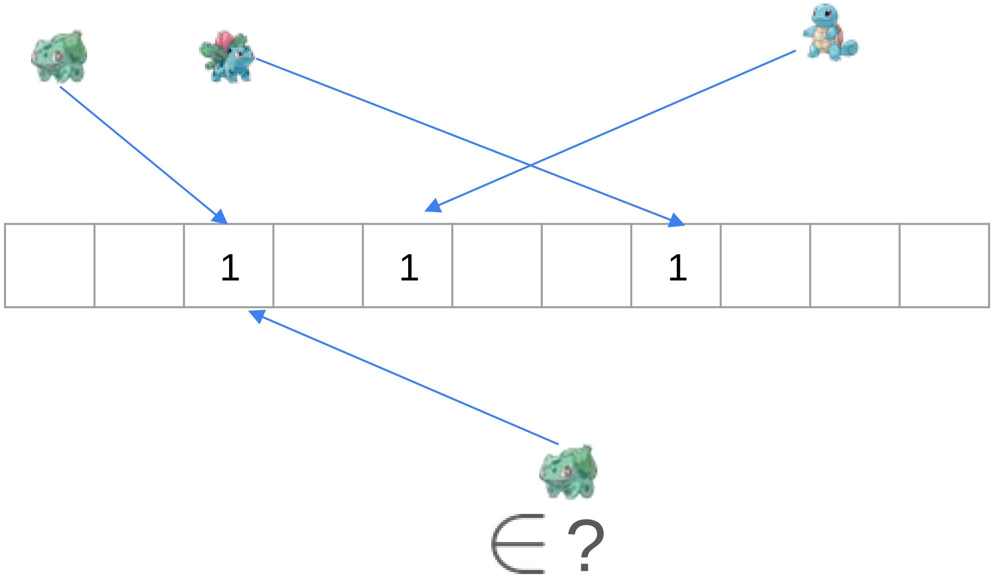
Motivation

Wikipedia: Bloom filter is a space-efficient probabilistic data structure, [...] that is used to test whether an element is a member of a set.

Set dictionary size to m
Use hash function $h(\cdot)$

For each element:
  Set bit at position $h(\cdot) \bmod m$

  Check bit at position $h(\cdot) \bmod m$



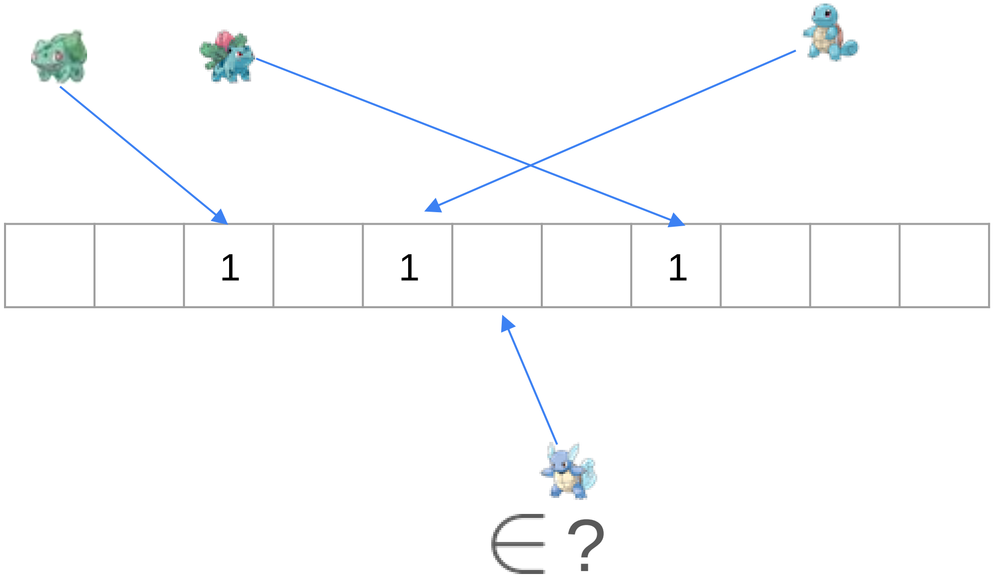
Motivation

Wikipedia: Bloom filter is a space-efficient probabilistic data structure, [...] that is used to test whether an element is a member of a set.

Set dictionary size to m
Use hash function $h(\cdot)$

For each element:
  Set bit at position $h(\cdot) \bmod m$

Check bit at position $h(\cdot) \bmod m$

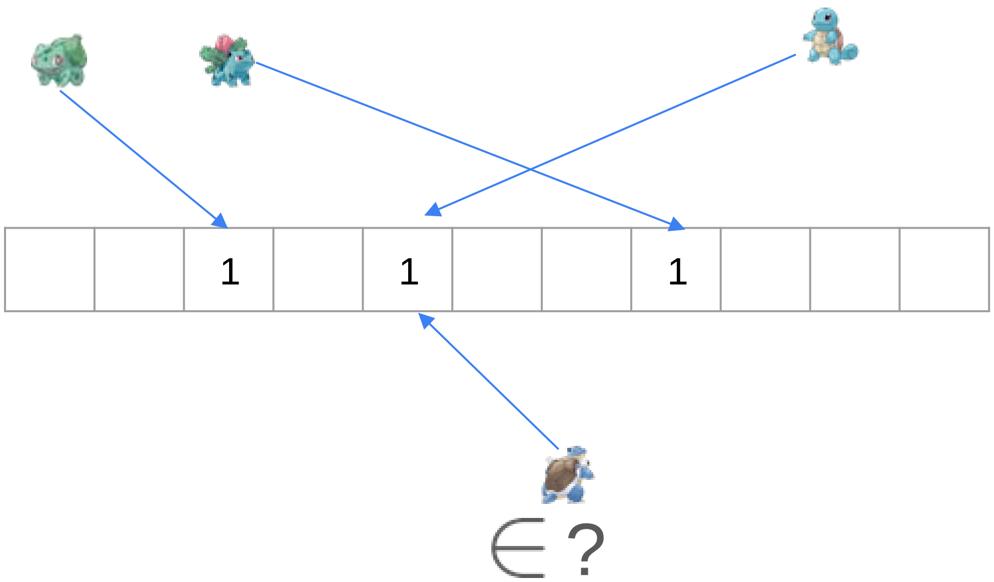


Motivation

Wikipedia: Bloom filter is a space-efficient probabilistic data structure, [...] that is used to test whether an element is a member of a set.

Set dictionary size to m
Use hash function $h(\cdot)$

For each element:
  Set bit at position $h(\cdot) \bmod m$



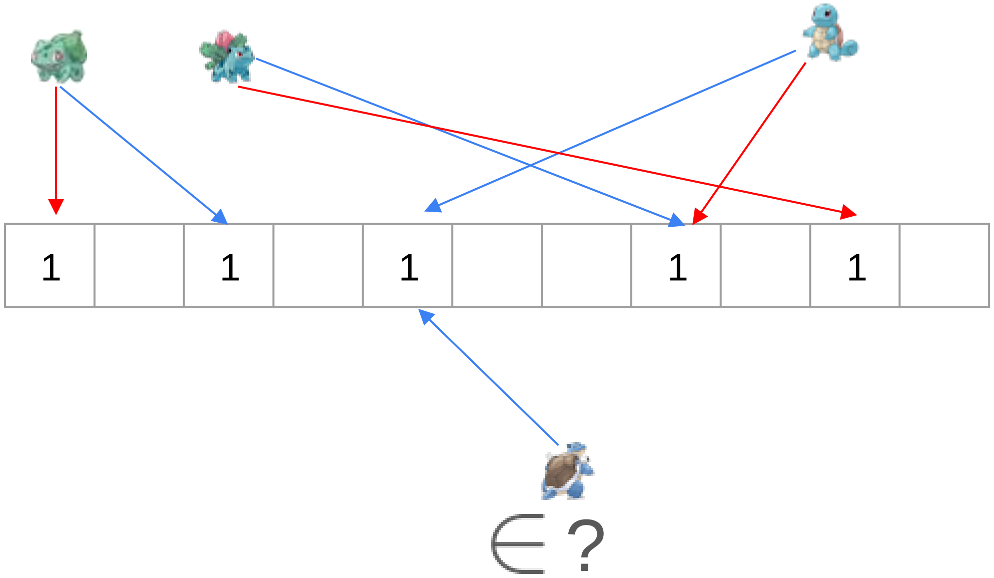
  Check bit at position $h(\cdot) \bmod m$

Finally: A wild Bloom Filter appears!

Wikipedia: Bloom filter is a space-efficient probabilistic data structure, [...] that is used to test whether an element is a member of a set.

Set dictionary size to m
Use k different hash functions $h_i(\cdot)$

For each element:
  For each $1 \leq i \leq k$:
    Set bit at position $h_i(\cdot) \bmod m$



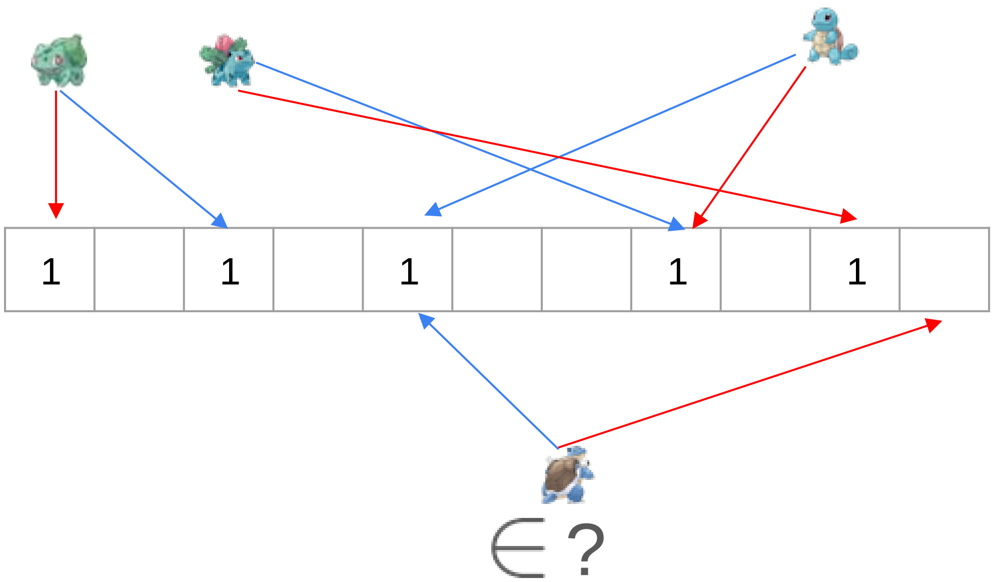
Finally: A wild Bloom Filter appears!

Wikipedia: Bloom filter is a space-efficient probabilistic data structure, [...] that is used to test whether an element is a member of a set.

Set dictionary size to m
Use k different hash functions $h_i(\cdot)$

For each element:
  For each $1 \leq i \leq k$:
    Set bit at position $h_i(\cdot) \bmod m$

  Check all bit positions $h_i(\cdot) \bmod m$
  Return true if all positions are set



False Positive Rate

Assuming a hash function mapping to a random value in [0, m-1]

- Pr[BF[i] = 1] = $\frac{1}{m}$; hence Pr[BF[i] = 0] = $1 - \frac{1}{m}$

For k (independent) hash functions:

- Pr[BF[i] = 0] = $\left(1 - \frac{1}{m}\right)^k = \left(\left(1 - \frac{1}{m}\right)^m\right)^{\frac{k}{m}} \approx e^{-\frac{k}{m}}$

False Positive Rate

Assuming a hash function mapping to a random value in [0, m-1]

- Pr[BF[i] = 1] = $\frac{1}{m}$; hence Pr[BF[i] = 0] = $1 - \frac{1}{m}$

For k (independent) hash functions:

- Pr[BF[i] = 0] = $\left(1 - \frac{1}{m}\right)^k = \left(\left(1 - \frac{1}{m}\right)^m\right)^{\frac{k}{m}} \approx e^{-\frac{k}{m}}$

After we have inserted n elements:

- Pr[BF[i] = 0] = $\left(1 - \frac{1}{m}\right)^{nk} \approx e^{-\frac{nk}{m}}$; hence Pr[BF[i] = 1] $\approx 1 - e^{-\frac{nk}{m}}$

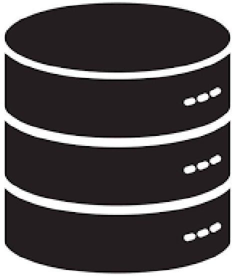
The probability for a false positive for element  requires that all k bits at $h_i(\cdot)$ are set

- $\left(1 - e^{-\frac{nk}{m}}\right)^k$

My first contact

Searchable Symmetric Encryption

Outsource encrypted document collection



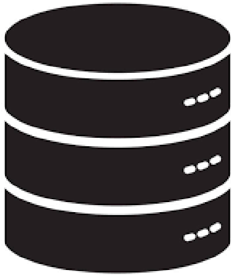
Chang, Yan-Cheng, and Michael Mitzenmacher. "Privacy preserving keyword searches on remote encrypted data." *ACNS* 2005.

My first contact

Searchable Symmetric Encryption

Filter for all documents that contain an encrypted keyword

Identifiers of documents containing the encrypted keyword



Chang, Yan-Cheng, and Michael Mitzenmacher. "Privacy preserving keyword searches on remote encrypted data." *ACNS* 2005.

My first contact

Searchable Symmetric Encryption

Doc_1 = (Squirtle, Bulbasaur)

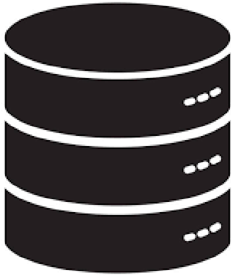
$HMAC_1(k,\cdot), HMAC_2(k,\cdot)$

	1		1			1					1	1				1			1	1	1

Doc_2 = (Bulbasaur, Ivysaur)

$HMAC_1(k,\cdot), HMAC_2(k,\cdot)$

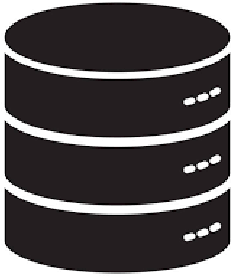
		1		1		1	1		1	1				1		1	1			1	



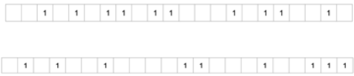
Chang, Yan-Cheng, and Michael Mitzenmacher. "Privacy preserving keyword searches on remote encrypted data." *ACNS* 2005.

My first contact

Searchable Symmetric Encryption



$HMAC_1(k, Squirtle)$,
$HMAC_2(k, Squirtle)$



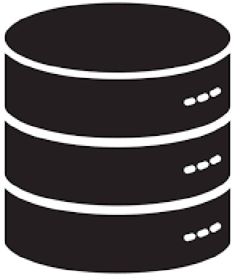
Doc_1

Doc_2

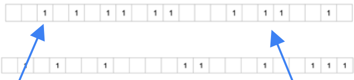
Chang, Yan-Cheng, and Michael Mitzenmacher. "Privacy preserving keyword searches on remote encrypted data." *ACNS* 2005.

My first contact

Searchable Symmetric Encryption



Doc_1



Doc_1

Doc_2

$HMAC_2(k, Squirtle)$

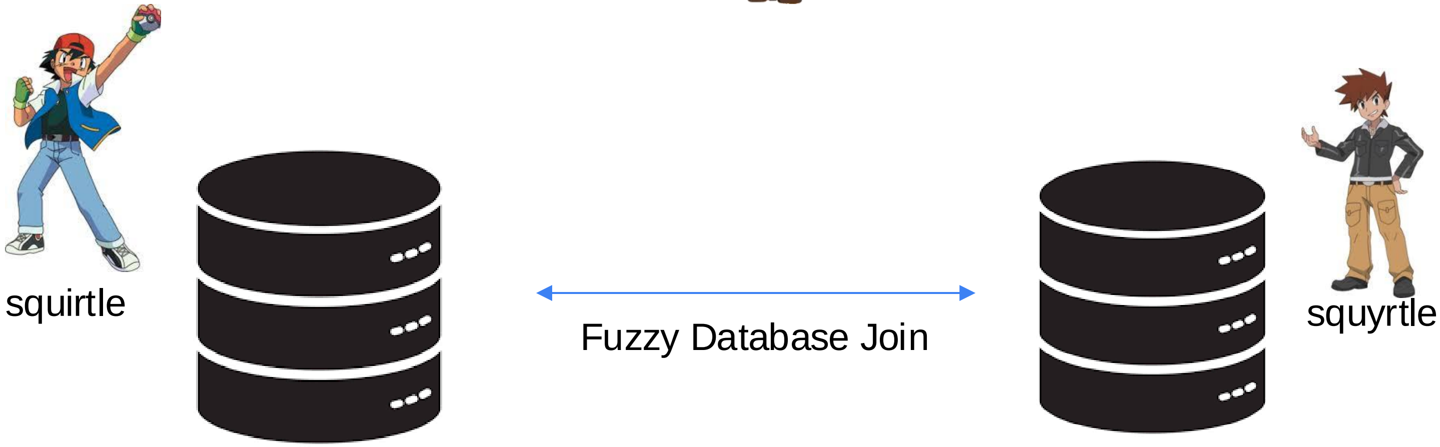
$HMAC_1(k, Squirtle),$

Chang, Yan-Cheng, and Michael Mitzenmacher. "Privacy preserving keyword searches on remote encrypted data." *ACNS* 2005.

Application II

Privacy Preserving Record Linkage

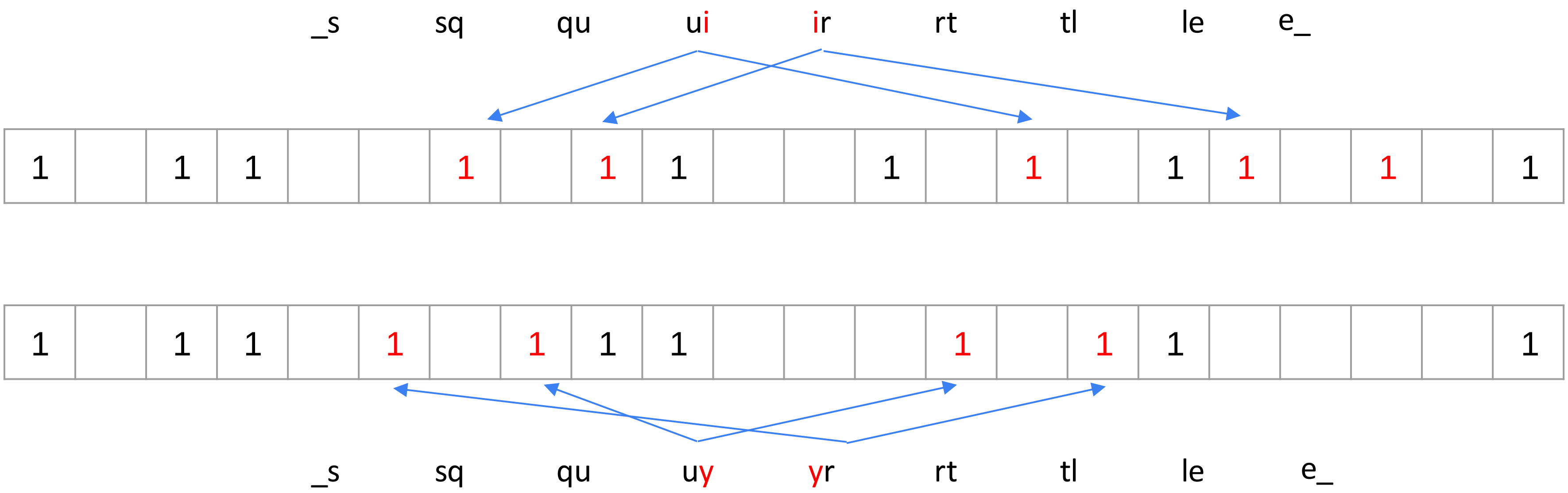
What Pokémons do they have in common?



Schnell, Rainer, Tobias Bachteler, and Jörg Reiher. "Privacy-preserving record linkage using Bloom filters." BMC medical informatics and decision making 2009

Application II

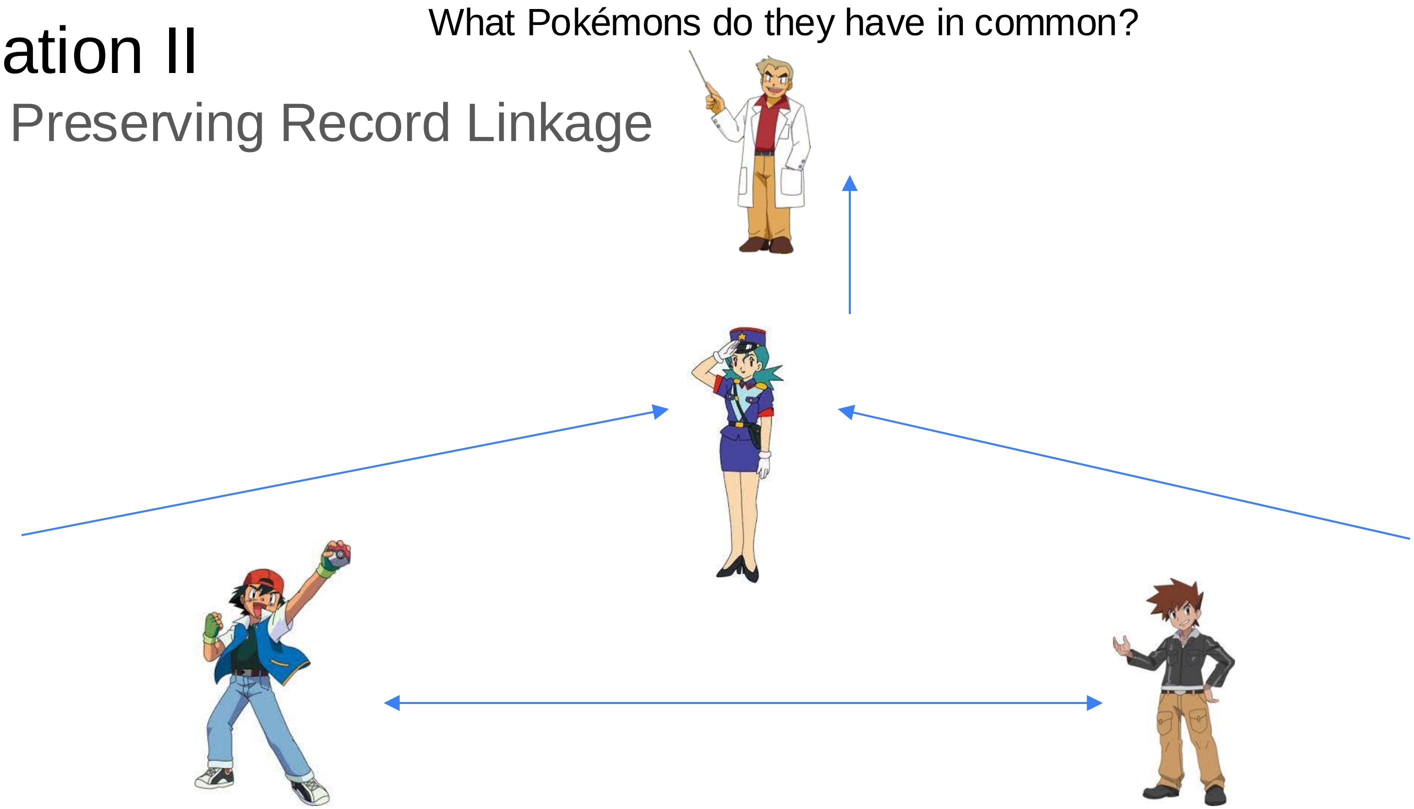
Privacy Preserving Record Linkage



Schnell, Rainer, Tobias Bachteler, and Jörg Reiher. "Privacy-preserving record linkage using Bloom filters." BMC medical informatics and decision making 2009

Application II

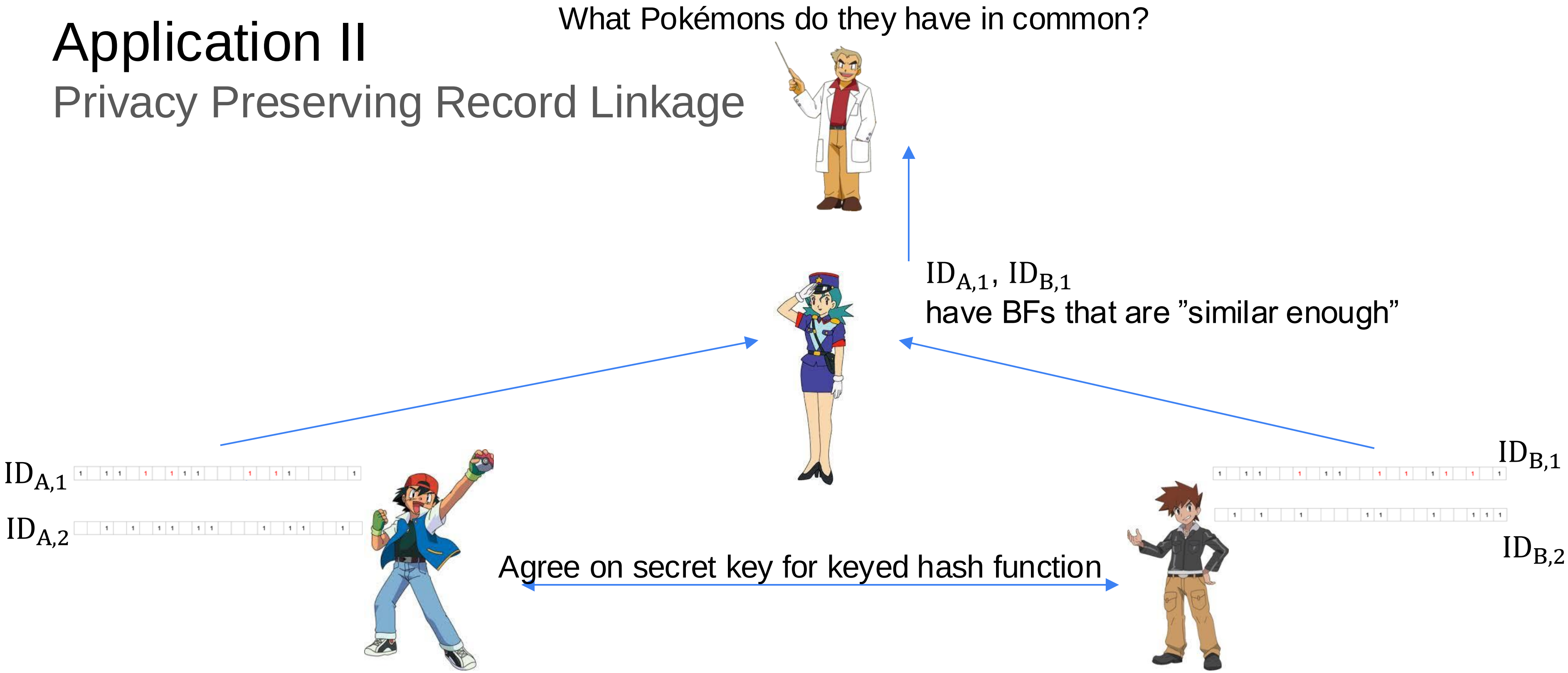
Privacy Preserving Record Linkage



Schnell, Rainer, Tobias Bachteler, and Jörg Reiher. "Privacy-preserving record linkage using Bloom filters." BMC medical informatics and decision making 2009

Application II

Privacy Preserving Record Linkage

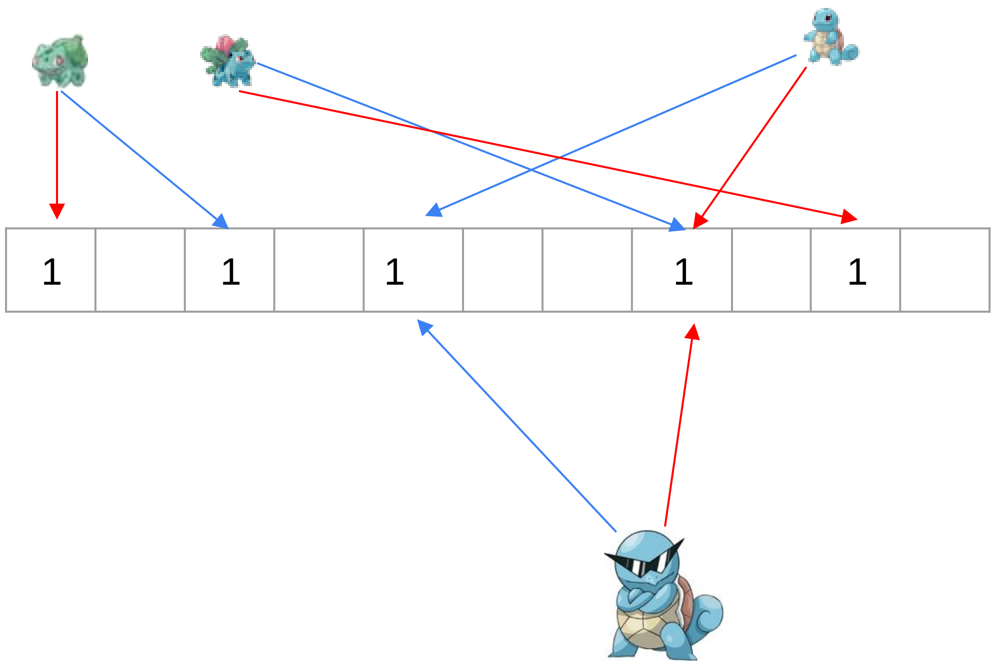


Schnell, Rainer, Tobias Bachteler, and Jörg Reiher. "Privacy-preserving record linkage using Bloom filters." BMC medical informatics and decision making 2009

Extension I: Counting Bloom Filters

Set dictionary size to m
Use k different hash functions $h_i(\cdot)$

For each element:
  For each $1 \leq i \leq k$:
    Set bit at position $h_i(\cdot) \bmod m$

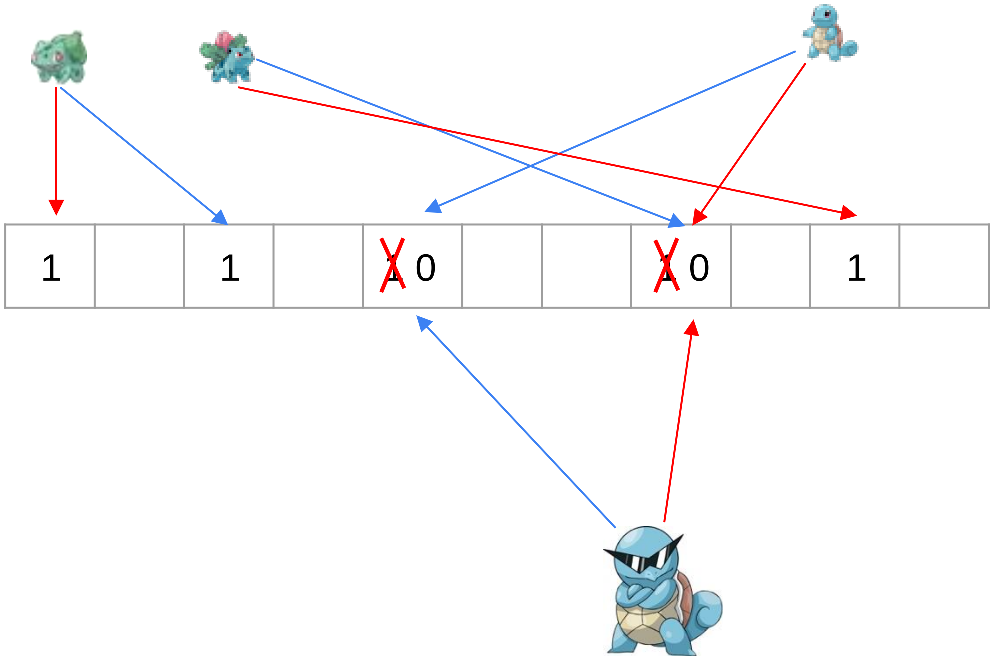


Delete Element:
Unset all bit positions $h_i(\cdot) \bmod m$

Extension I: Counting Bloom Filters

Set dictionary size to m
Use k different hash functions $h_i(\cdot)$

For each element:
  For each $1 \leq i \leq k$:
    Set bit at position $h_i(\cdot) \bmod m$

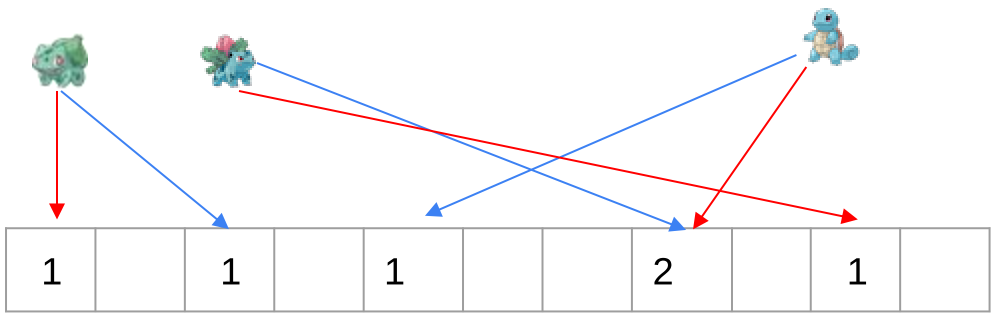


Delete Element:
Unset all bit positions $h_i(\cdot) \bmod m$

Extension I: Counting Bloom Filters

Set dictionary size to m
Use k different hash functions $h_i(\cdot)$

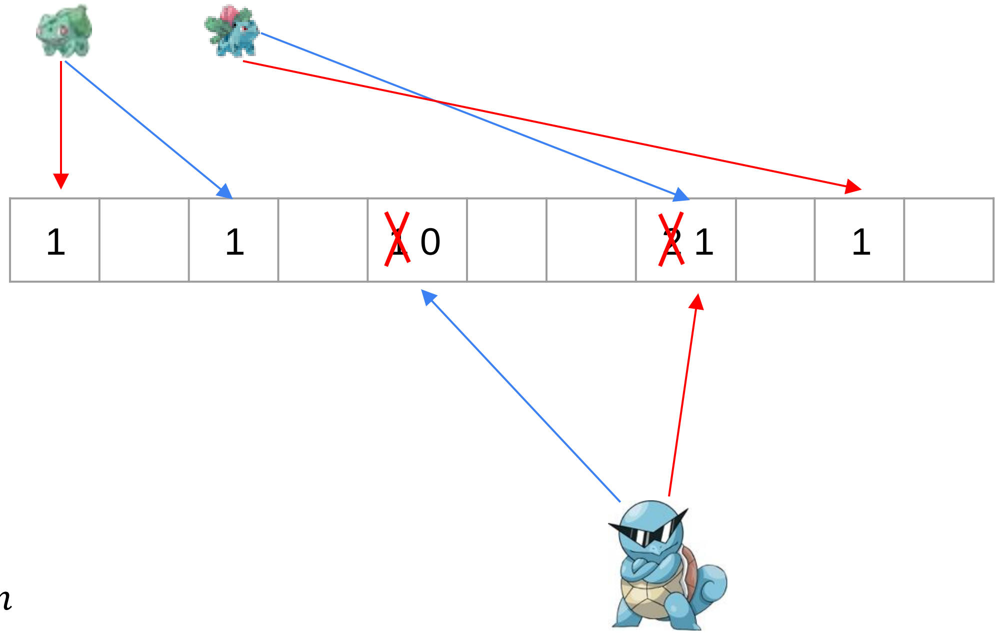
For each element:
  For each $1 \leq i \leq k$:
    Increment at position $h_i(\cdot) \bmod m$



Extension I: Counting Bloom Filters

Set dictionary size to m
Use k different hash functions $h_i(\cdot)$

For each element:
For each $1 \leq i \leq k$:
Increment at position $h_i(\cdot) \bmod m$



Delete Element:
Decrement at bit positions $h_i(\cdot) \bmod m$

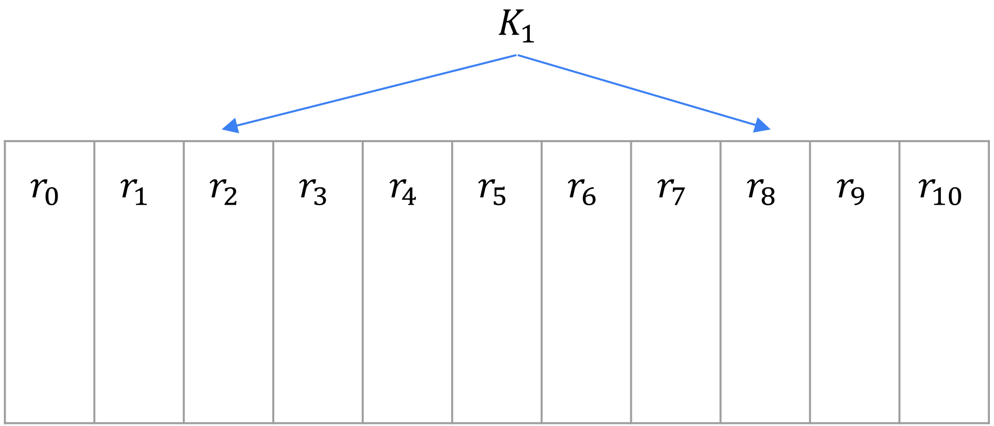
Extension II: Garbled Bloom Filters

(Key, Value) store via Bloom Filters

For (K_1, V_1) with $h_1(K_1) = 2, h_2(k_1) = 7$

Set r_2 and r_7 such that

$r_2 + r_7 = V_1$



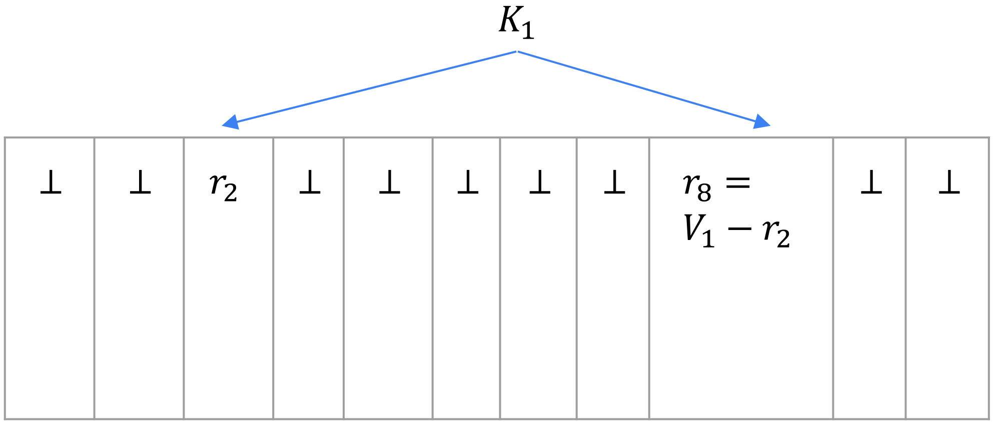
Dong, Changyu, Liqun Chen, and Zikai Wen. "When private set intersection meets big data: an efficient and scalable protocol." CCS 2013.

Extension II: Garbled Bloom Filters

(Key, Value) store via Bloom Filters

Initialize BF F with special element ⊥

For each key K_j:
  $f_j = V_j$
  For each $1 \leq i \leq k-1$:
    $p = h_i(K_i)$
    If p :
      Sample r_p and set $F[p] = r_p$
    $f_j = f_j - F[p]$
  $p = h_k(K_j)$
  I
  A
    $F[p] = f_j$



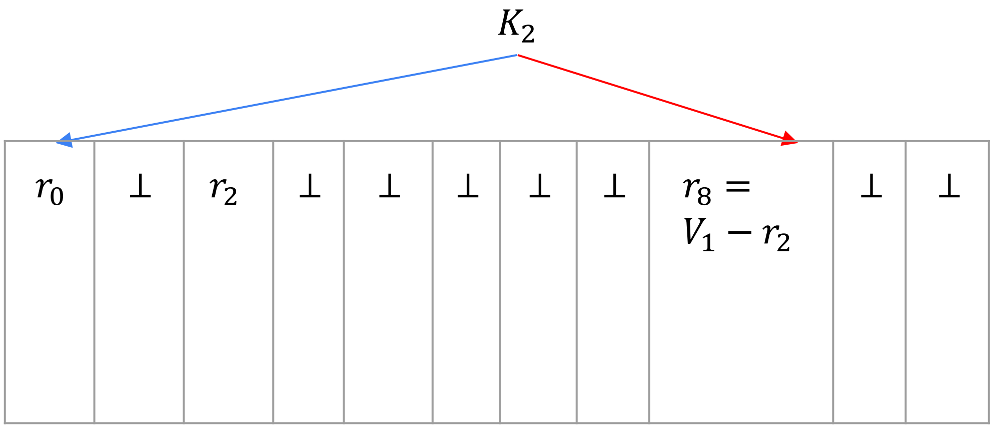
Dong, Changyu, Liqun Chen, and Zikai Wen. "When private set intersection meets big data: an efficient and scalable protocol." CCS 2013.

Extension II: Garbled Bloom Filters

(Key, Value) store via Bloom Filters

Initialize BF F with special element ⊥

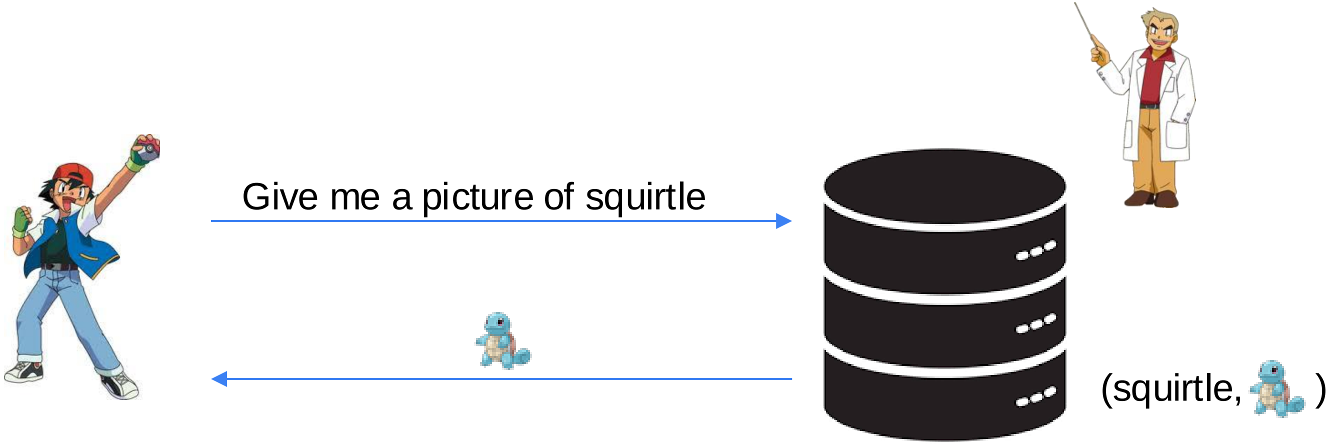
For each key K_j:
  $f_j = V_j$
  For each $1 \leq i \leq k-1$:
    $p = h_i(K_j)$
    If $F[p]$ == ⊥
      Sample r_p and set $F[p] = r_p$
    $f_j = f_j - F[p]$
  $p = h_k(K_j)$
    If $F[p]$!= ⊥
    Abort with Error
      $F[p] = f_j$



Dong, Changyu, Liqun Chen, and Zikai Wen. "When private set intersection meets big data: an efficient and scalable protocol." CCS 2013.

How I stumbled upon a wild Bloom Filters recently... (again!)

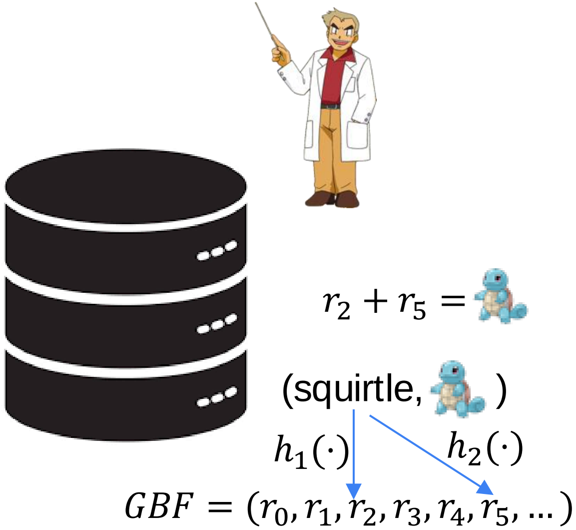
Keyword Private Information Retrieval



Celi, Sofía, and Alex Davidson. "Call me by my name: Simple, practical private information retrieval for keyword queries." CCS 2024.

How I stumbled upon a wild Bloom Filters recently... (again!)

Keyword Private Information Retrieval: Setup



Celi, Sofía, and Alex Davidson. "Call me by my name: Simple, practical private information retrieval for keyword queries." CCS 2024.

How I stumbled upon a wild Bloom Filters recently... (again!)

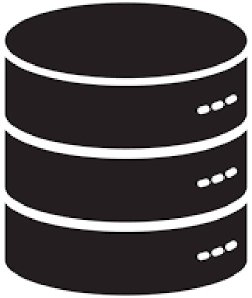
Keyword Private Information Retrieval: Query

$Q = (0, 0, 1, 0, 0, 1, \dots)$

Position $h_1(squirtle)$

Position $h_2(squirtle)$

Q



$r_2 + r_5 =$ 

(squirtle, )

$h_1(\cdot)$ $h_2(\cdot)$

$GBF = (r_0, r_1, r_2, r_3, r_4, r_5, \dots)$

Celi, Sofía, and Alex Davidson. "Call me by my name: Simple, practical private information retrieval for keyword queries." CCS 2024.

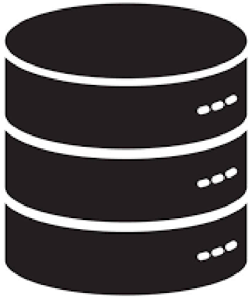
How I stumbled upon a wild Bloom Filters recently... (again!)

Keyword Private Information Retrieval: Response

$\mathrm{Q} = (0, 0, 1, 0, 0, 1, \ldots)$

Q →

← $\langle Q, GBF^{\top} \rangle$



$r_2 + r_5 =$ 

(squirtle, )

$h_1(\cdot)$ $h_2(\cdot)$

$GBF = (r_0, r_1, r_2, r_3, r_4, r_5, \ldots)$

$0 \cdot r_0 + 0 \cdot r_1 + 1 \cdot r_2 + 0 \cdot r_3 + 0 \cdot r_4 + 1 \cdot r_5 \ldots$

$r_2 + r_5 =$ 

Celi, Sofía, and Alex Davidson. "Call me by my name: Simple, practical private information retrieval for keyword queries." CCS 2024.

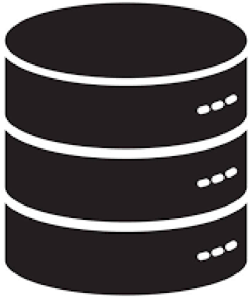
How I stumbled upon a wild Bloom Filters recently... (again!)

Keyword Private Information Retrieval: Now with encryption!

$Q = (0, 0, 1, 0, 0, 1, \ldots)$

$HomEnc(Q)$ →

← $HomEnc(\langle Q, GBF^{\top} \rangle)$



$r_2 + r_5 =$ 

(squirtle, )

$h_1(\cdot)$ $h_2(\cdot)$

$GBF = (r_0, r_1, r_2, r_3, r_4, r_5, \ldots)$

$0 \cdot r_0 + 0 \cdot r_1 + 1 \cdot r_2 + 0 \cdot r_3 + 0 \cdot r_4 + 1 \cdot r_5 \ldots$

$Dec(HomEnc(\langle Q, GBF^{\top} \rangle)) = r_2 + r_5 =$ 

Celi, Sofía, and Alex Davidson. "Call me by my name: Simple, practical private information retrieval for keyword queries." CCS 2024.

Thanks!