

# Substituting Cuckoo Hashing with Common Variants in OT-Based PSI Protocol

---

Jan. 5, 2022

Huiyang He, Chenglin Li

# Overview

---

1. About PSI
2. Basic Primitives
3. Our Works
4. Challenge & Future Works

# About PSI

## What is Private Set Intersection ?

---

- Private Set Intersection (PSI) allows two parties  $P_1$  and  $P_2$  holding sets  $X$  and  $Y$ , respectively, to identify the intersection  $X \cap Y$  without revealing any information about elements that are not in the intersection.

## What is Private Set Intersection ?

---

### *PARAMETERS:*

- *Two parties:  $P_1$  and  $P_2$ .*
- *$P_1$  has set  $X$ .*
- *$P_2$  has set  $Y$ .*

### *FUNCTIONALITY:*

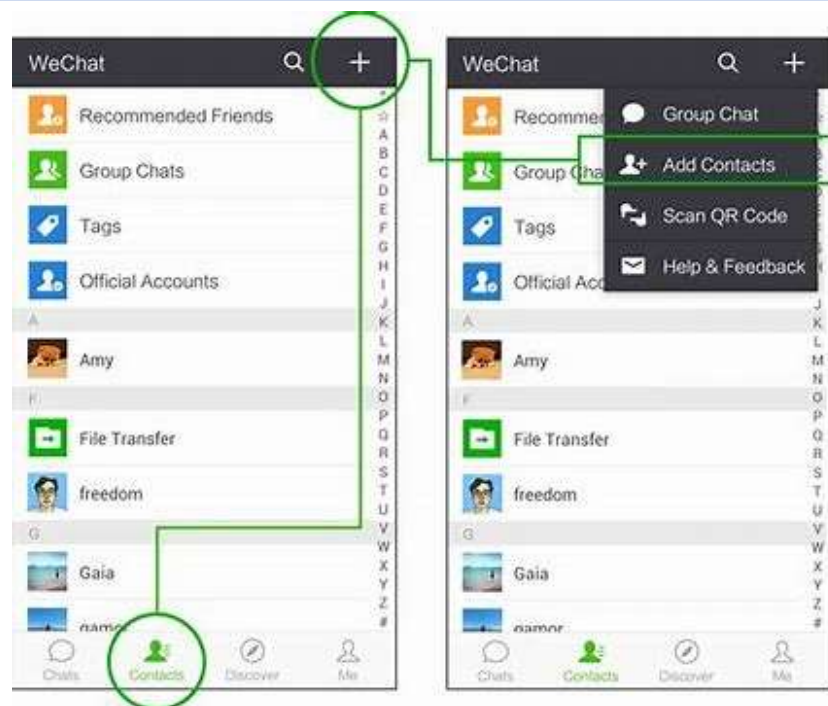
- *$P_1$  outputs  $\perp$ .*
- *$P_2$  outputs  $X \cap Y$ .*

## What is Private Set Intersection ?

---

- An active research field.
- One of the best studied applications of SMPC.
- Many PSI protocols has been proposed.

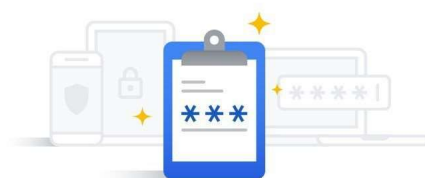
## Contact Discovery






$\{\text{my phone contacts}\} \cap \{\text{users of your service}\}$

# Password Checkup

---



We analyzed your saved passwords and found the following issues

	<b>1 compromised password</b> Change these passwords now	▼
	<b>59 reused passwords</b> Create unique passwords	▼
	<b>28 accounts using a weak password</b> Create strong passwords	▼

$$\{my\ passwords\} \cap \{passwords\ found\ in\ breaches\}$$



# Basic Primitives

## Basic Primitives

---

1. Oblivious Transfer (OT) & OT Extension
2. Private Equality Test (PEQT) & PSI Prototype
3. Cuckoo Hashing

# Oblivious Transfer (OT)

---

## *PARAMETERS:*

- *Two parties: Sender  $S$  and Receiver  $R$ .*
- *$S$  has two secrets,  $x_0, x_1 \in \{0, 1\}^n$ ,*
- *$R$  has a selection bit  $r \in \{0, 1\}$*

## *FUNCTIONALITY:*

- *$R$  outputs  $x_r$*
- *$S$  outputs  $\perp$*

Functionality of  $\binom{2}{1}$  OT

## OT Extension

---

- OT can only be built using public encryption operation, thus very expensive.
- Thus, We need OT extension.
- OT extension uses a small amount of OTs to realize a large number of OT instances.

## Private Equality Test (PEQT)

- 
- Use OT once to compare a single bit

$$x_s = x_r ?$$

$$\Rightarrow O(1)$$

- PEQT: Compare a pair of strings of length  $l$

$$m_s = x_{r_s}^0 \oplus x_{r_s}^1 \oplus \cdots \oplus x_{r_s}^l, m_r = x_{r_r}^0 \oplus x_{r_r}^1 \oplus \cdots \oplus x_{r_r}^l, m_s = m_r ?$$

$$\Rightarrow O(l)$$

- Whether a string  $x$  is in the set  $Y$  with  $n$  elements

$$\Rightarrow \text{PEQT for } x \text{ and each } y_i \text{ in } Y$$

$$\Rightarrow O(nl)$$

## PSI Prototype from PEQT

---

- Intersection of set  $X$  with  $n$  elements and set  $Y$  with  $n$  elements
  - $\Rightarrow$  PEQT for each  $x_i$  in  $X$  and each  $y_i$  in  $Y$
  - $\Rightarrow O(n^2l)$
- Too much cost even if we have OT extension
- Any way to reduce the comparison?

**Hashing can help!**

# Cuckoo Hashing

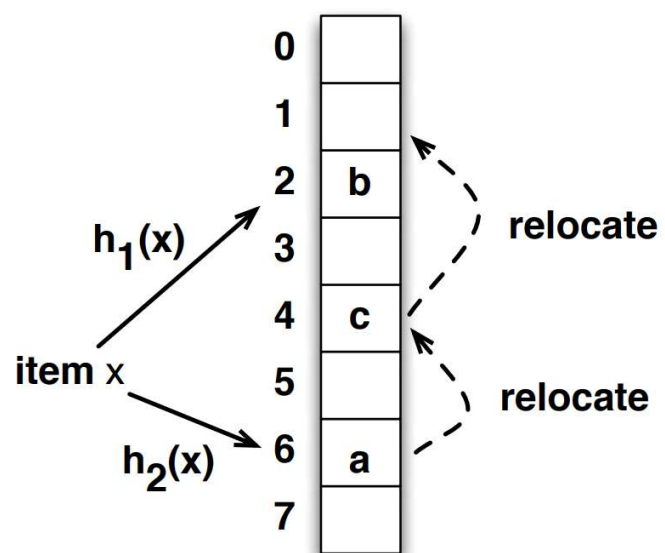
---

- First described by Rasmus Pagh and Flemming Friche Rodler in 2001.
- “Faster Private Set Intersection Based on OT Extension”

PSI based on cuckoo hashing has the same asymptotic overhead as the protocol that uses simple hashing, and the balanced allocations is much higher.



## Two-Way Cuckoo Hashing



0	
1	c
2	b
3	
4	a
5	
6	x
7	



## PSI with Cuckoo Hashing

---

- Alice uses cuckoo hashing
- Bob uses simple hash with each hash functions
- PSIs are only used in corresponding buckets.
- $O(n^2 l) \Rightarrow O(nl)$
- Greatly reduce the number OTs needed.

## Further works

---

- “*Efficient Batched Oblivious PRF with Applications to Private Set Intersection*”
- Batched, related-key OPRF(BaRK-OPRF)
- Each comparison only consumes one OT
- only  $4.3\times$  slower than the *insecure* naïve hashing approach for PSI
- $O(nl) \Rightarrow O(n)$

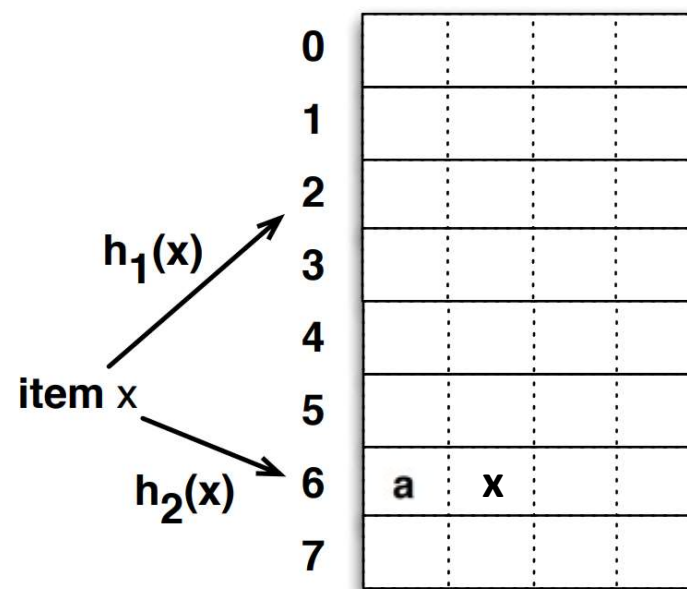
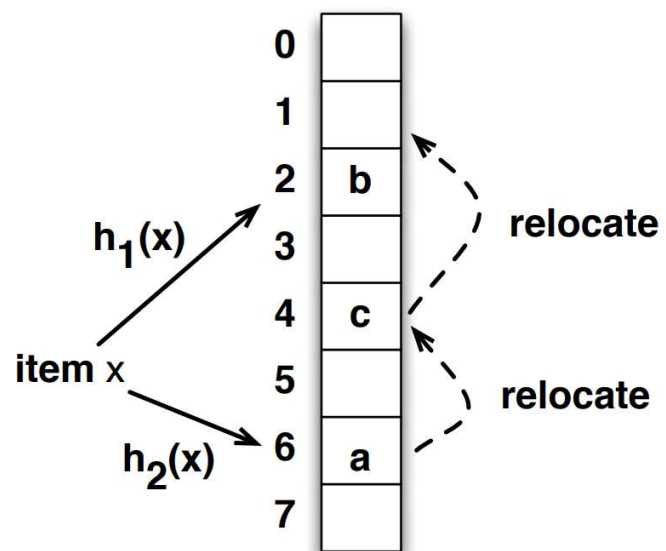
# Our Works

## Our Works

---

1. Cuckoo Hashing Variants
2. Load Factor Test
3. PSI with Cuckoo Hashing Variants

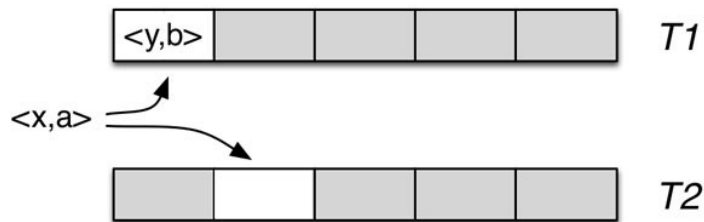
# Cuckoo Hashing with Buckets



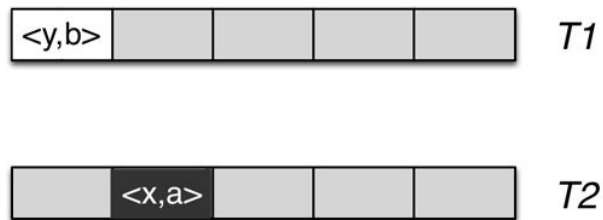
# Cuckoo Hashing Using Separate Tables

## Insertion when one of the two buckets is empty

**Step 1:** Both buckets for  $\langle x, a \rangle$  are tested, the one in T2 is empty.

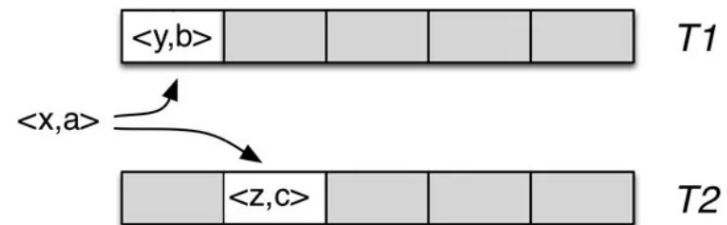


**Step 2:**  $\langle x, a \rangle$  is stored in the empty bucket in T2.

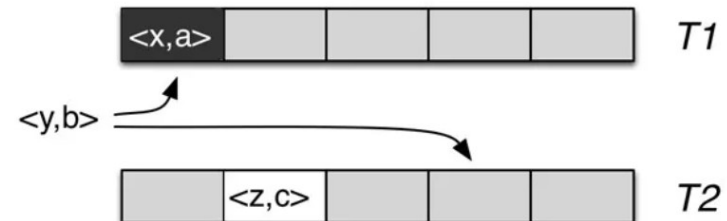


## Insertion when the two buckets already contain entries

**Step 1:** Here  $\langle y, b \rangle$  will be withdrawn from T1 so that  $\langle x, a \rangle$  can be stored.



**Step 2:** After  $\langle x, a \rangle$  has been stored in T1,  $\langle y, b \rangle$  needs to be moved to T2. The bucket in T2 may already contain an entry, if so this entry will need to be moved.



## Load Factor Test

---

- A higher load factor means that the same number of elements can be loaded with a smaller hash table, so that less OTs are needed in PSI protocol.
- We extract the hashing part in the project and test the occupancy of different hashing variants.

---

# Results



## Benchmark Environment

---

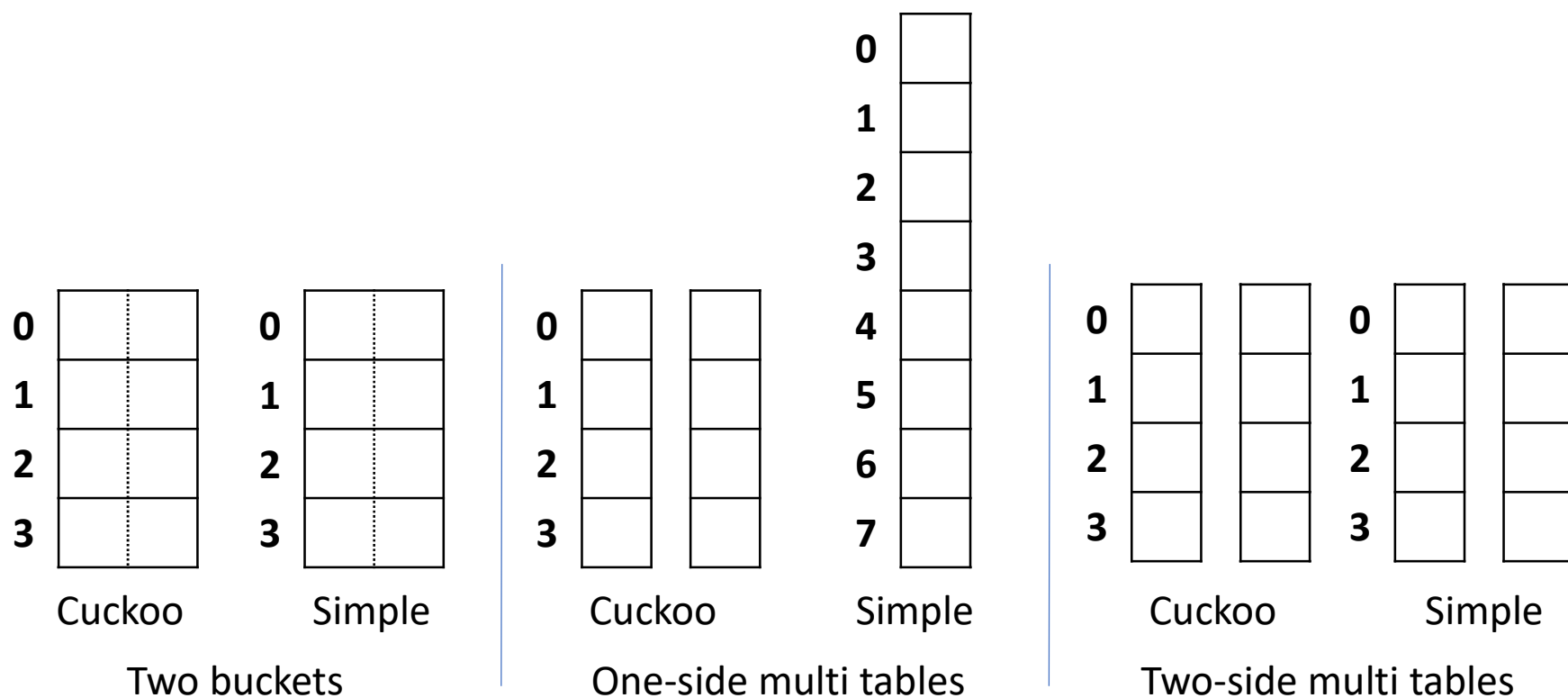
- Machine 1:
  - i9-10900K
  - 32 GB 3000MHz RAM
  - Ubuntu 18.04LTS on portable SSD
- Machine 2:
  - i7-8750H
  - 32 GB 2400MHz RAM
  - Ubuntu 18.04LTS on WSL for WINDOWS 10
- Network: simulated Gigabytes LAN with 2ms latency using Linux tc command

## Load Factor Test

---

hashing technique	occupancy rate [%]	
simple hashing	63.21	-
three-way cuckoo hashing	93.47	91.8 [2]
two-way cuckoo hashing with two buckets	88.05	84 [10]
three-way cuckoo hashing using separate tables	93.46	-

# Attempts on Cuckoo Hashing Variants



## Attempts on Cuckoo Hashing Variants

Protocol	Phase	set size n			
		$2^8$	$2^{12}$	$2^{16}$	$2^{20}$
BaRK-OPRF	Online	4	10	100	2459
	Offline	55	55	67	326
Two buckets	Online	5	11	179	4052
	Offline	56	56	67	278
One-side multi tables	Online	4	14	118	2829
	Offline	56	55	67	281
Two-side multi tables	Online	4	10	102	<b>2437</b>
	Offline	59	54	67	<b>263</b>

Running time in ms with n elements in online and offline phases on machine 1

## Attempts on Cuckoo Hashing Variants

Protocol	Phase	set size n			
		$2^8$	$2^{12}$	$2^{16}$	$2^{20}$
BaRK-OPRF	Online	18	26	169	3341
	Offline	95	96	118	941
Two buckets	Online	16	26	189	4185
	Offline	92	95	108	635
One-side multi tables	Online	17	25	176	3343
	Offline	86	88	125	912
Two-side multi tables	Online	17	26	156	<b>3295</b>
	Offline	86	89	107	<b>678</b>

Running time in ms with n elements in online and offline phases on machine 2

---

# Challenges & Future Works

## Challenges

---

- Compilation – poor dependencies management
- Understand the code: how ROT built into the protocol
- Not suitable hardware: cannot finish the test of size  $2^{24}$

## Challenges

- Unexplainable result on machine 2

Protocol	Phase	set size n			
		$2^8$	$2^{12}$	$2^{16}$	$2^{20}$
BaRK-OPRF	Online	18	26	199	4611
	Offline	98	102	118	984
Two buckets	Online	19	24	188	4258
	Offline	96	95	113	<b>735</b>
One-side multi tables	Online	18	26	196	<b>3533</b>
	Offline	92	88	115	942
Two-side multi tables	Online	18	27	164	4742
	Offline	92	89	107	947



## Next Steps...

---

- Better program organization
- Better dependencies management
- Parallelization based on the separate tables branch
- Test under WAN setting

## Reference

---

- [1] Kolesnikov V, Kumaresan R, Rosulek M, et al. Efficient batched oblivious PRF with applications to private set intersection[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 2016: 818-829.
- [2] Pinkas B, Schneider T, Segev G, et al. Phasing: Private set intersection using permutation-based hashing[C]//24th USENIX Security Symposium (USENIX Security 15). 2015: 515-530.
- [3] Pinkas B, Schneider T, Zohner M. Faster private set intersection based on {OT} extension[C]//23rd USENIX Security Symposium (USENIX Security 14). 2014: 797-812.
- [4] Rabin M O. How To Exchange Secrets with Oblivious Transfer[J]. IACR Cryptol. ePrint Arch., 2005, 2005(187).
- [5] Naor M, Pinkas B. Efficient oblivious transfer protocols[C]//SODA. 2001, 1: 448-457.
- [6] Ishai Y, Kilian J, Nissim K, et al. Extending oblivious transfers efficiently[C]//Annual International Cryptology Conference. Springer, Berlin, Heidelberg, 2003: 145-161.
- [7] Pagh R , Rodler F F . Cuckoo Hashing[J]. Springer, Berlin, Heidelberg, 2001.
- [8] Kirsch, Adam, Mitzenmacher, et al. MORE ROBUST HASHING: CUCKOO HASHING WITH A STASH.[J]. SIAM Journal on Computing, 2009.
- [9] Pinkas B, Schneider T, Zohner M. Faster private set intersection based on {OT} extension[C]//23rd USENIX Security Symposium (USENIX Security 14). 2014: 797-812.
- [10] Fan B, Andersen D G, Kaminsky M, et al. Cuckoo filter: Practically better than bloom[C]//Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies. 2014: 75-88.

**Thank You !**