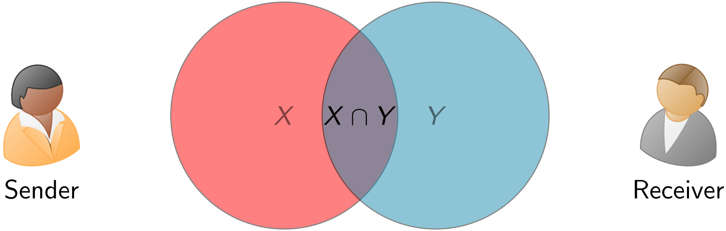
KU LEUVEN

Labeled PSI from Homomorphic Encryption with Reduced Computation and Communication

ACM CCS 2021

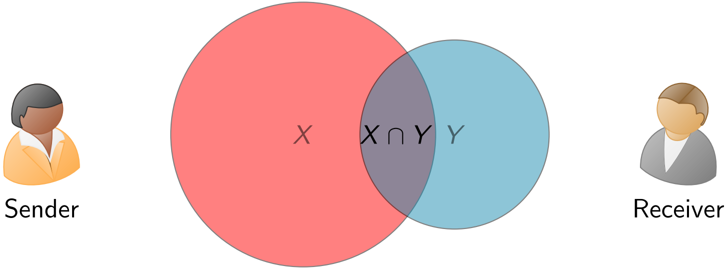
Kelong Cong, imec-COSIC, KU Leuven
Radames Cruz Moreno, Microsoft Research
Mariana Botelho da Gama, imec-COSIC, KU Leuven
Wei Dai, Microsoft Research
Ilia Iliashenko, imec-COSIC, KU Leuven
Kim Laine, Microsoft Research
Michael Rosenberg, University of Maryland

Private Set Intersection



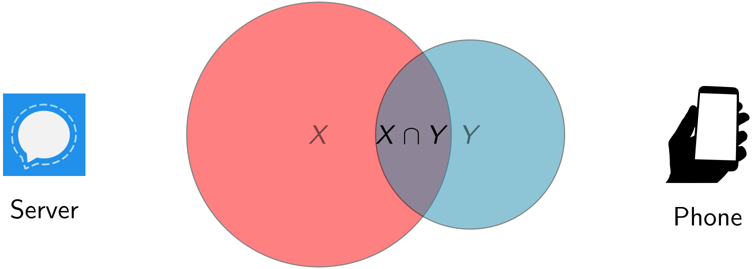
- Receiver learns $X \cap Y$.
- X and Y remain private.

Unbalanced PSI



- Unbalanced PSI - assume $|X| \gg |Y|$.

Private Contact Discovery Application



- X: registered phone numbers
- Y: contacts on the phone

Unbalanced PSI: Related Work

Unbalanced PSI: Related Work

Based on OPRF
Kales et al. USENIX'19

- Sender distributes cuckoo filter created from X
- Communication is $\mathcal{O}(|X|)$
- Very efficient online phase

Unbalanced PSI: Related Work

Based on OPRF
Kales et al. USENIX'19

- Sender distributes cuckoo filter created from X
- Communication is $\mathcal{O}(|X|)$
- Very efficient online phase

Based on HE
Chen et al. CCS'18

- Intersection is computed by the sender
- Communication is $\mathcal{O}(|Y| \log |X|)$
- Computation is $\mathcal{O}(|X|)$
- Starting point of our work

(Somewhat) Homomorphic Encryption

Functionality of HE

- $f(\mathrm{Ctxt}(Y)) = \mathrm{Ctxt}(f'(Y))$
- f' is any arithmetic circuit of bounded depth, e.g., $+, -, \cdot, \mathrm{Aut}$
- e.g., $f'(Y) = X \cap Y$, where X is hardwired

(Somewhat) Homomorphic Encryption

Functionality of HE

- $f(\mathsf{Ctxt}(Y)) = \mathsf{Ctxt}(f'(Y))$
- f' is any arithmetic circuit of bounded depth, e.g., $+, -, \cdot, \mathsf{Aut}$
- e.g., $f'(Y) = X \cap Y$, where X is hardwired

Cost of HE

- Multiplication is the most expensive
- Need to minimize multiplicative width and depth
- Operations can be parallelized (more on this later)

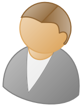
Basic PSI Protocol Using HE

Inputs: Sender inputs set X, receiver inputs set Y, $|X| \gg |Y|$

Setup: Receiver generates a key pair for the HE scheme.

Set encryption: $[\![y_1]\!], \ldots, [\![y_{|Y|}]\!]$

Compute intersection: $[\![d_1]\!], \ldots, [\![d_{|Y|}]\!]$



$$[\![d_i]\!] = r_i \prod_{x \in X} ([\![y_i]\!] - x)$$

Reply extraction: Receiver decrypts the ciphertexts and outputs

$$X \cap Y = \{y_i : \text{HE.Decrypt}([\![d_i]\!]) = 0\}$$

KU LEUVEN

Basic PSI Protocol Using HE

- Intersection polynomial

$$r \prod_{x \in X} (\llbracket y \rrbracket - x) = r\llbracket y \rrbracket^{|X|} + ra_{|X|}\llbracket y \rrbracket^{|X|-1} + \ldots + ra_0$$

- Multiplicative depth is $\mathcal{O}(\log |X|)$ from square and multiply
- Communication cost is $\mathcal{O}(|Y|)$ HE ciphertexts
- Computation cost is $\mathcal{O}(|X| \cdot |Y|)$ homomorphic operations

Previous Work
Windowing

- Instead of sending a single $[\![y]\!]$
- Send powers of $[\![y]\!]$, e.g., $[\![y^{2^0}]\!], [\![y^{2^1}]\!], \ldots, [\![y^{2^{\log|X|}}]\!]$
- New multiplicative depth $\mathcal{O}(\log\log|X|)$
- Communication increased by a factor of $\mathcal{O}(\log|X|)$

Previous Work

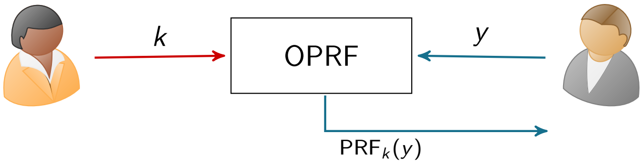
Parallel computation

	pt_0	pt_1	pt_2		ct_0
slot 3	$x_2^{(0)}$	$x_3^{(1)}$	$x_4^{(0)}$	$P_3(x) = (x - x_2^{(0)})(x - x_3^{(1)})(x - x_4^{(0)})$	y_1
slot 2	$x_0^{(1)}$	$x_3^{(0)}$		$P_2(x) = (x - x_0^{(1)})(x - x_3^{(0)})$	y_0
slot 1	$x_1^{(0)}$	$x_2^{(1)}$		$P_1(x) = (x - x_1^{(0)})(x - x_2^{(1)})$	y_2
slot 0	$x_0^{(0)}$	$x_1^{(1)}$	$x_4^{(1)}$	$P_0(x) = (x - x_0^{(0)})(x - x_1^{(1)})(x - x_4^{(1)})$	

- Use cuckoo hashing for Y
- Same for X but without eviction, hash x_i into $x_i^{(0)}$ and $x_i^{(1)}$
- Polynomials are evaluated in parallel!

Previous Work

OPRF preprocessing



- No need padding or randomizing the intersection polynomial
- Security against malicious receiver

Our Improvements

Our Improvements

General optimizations

- Fast OPRF from FourQ (Costello and Longa 2015).
- Polynomial evaluation with Paterson-Stockmeyer algorithm.

Our Improvements

General optimizations

- Fast OPRF from FourQ (Costello and Longa 2015).
- Polynomial evaluation with Paterson-Stockmeyer algorithm.

Improved computation and communication

- Operations over prime fields.
- Extremal postage stamp bases.
- Implemented with SEAL.

Optimizing for communication complexity

- Operations over extension fields.
- Depth-free homomorphic Frobenius automorphisms.
- Implemented with HElib.

General optimizations

Paterson-Stockmeyer algorithm

Compute the degree D intersection polynomial in $\mathcal{O}(\sqrt{D})$ ciphertext-ciphertext multiplications.

The sender computes two sets of powers:

- **Low powers** $[\![y]\!]^2, [\![y]\!]^3, \dots, [\![y]\!]^{L-1}$
- **High powers:** $[\![y]\!]^L, [\![y]\!]^{2L}, [\![y]\!]^{3L}, \dots, [\![y]\!]^{(H-1)\cdot L}$

with $L, H \approx \sqrt{D}$.

General optimizations

Paterson-Stockmeyer algorithm

Then, rewrite the intersection polynomial:

$$\sum_{i=0}^{D} a_i \cdot [\![y]\!]^i$$

$$\Downarrow$$

$$\sum_{i=0}^{H-1} [\![y]\!]^{iL} \left(\sum_{j=0}^{L-1} \left(a_{iL+j} \cdot [\![y]\!]^j \right) \right)$$

KU LEUVEN

General optimizations

Paterson-Stockmeyer algorithm

Then, rewrite the intersection polynomial:

$$\sum_{i=0}^{D} a_i \cdot [\![y]\!]^i$$

$$\downarrow$$

$$\sum_{i=0}^{H-1} [\![y]\!]^{iL} \left(\sum_{j=0}^{L-1} \left(a_{iL+j} \cdot [\![y]\!]^j \right) \right)$$

- **Non-scalar multiplicative complexity:** $\mathcal{O}(\sqrt{D})$

Improved computation and communication

Extremal postage stamp bases

How to minimize the number of powers sent without exceeding the target depth?

How to minimize the number of powers sent without exceeding the target depth?

Global postage-stamp problem

Given positive integers h and k, determine a set of k positive integers $A_k = \{a_1 = 1 < a_2 < \ldots < a_k\}$ such that all integers $1, 2, \ldots, n$ can be written as a sum of h or fewer of the a_j, and n is as large as possible.

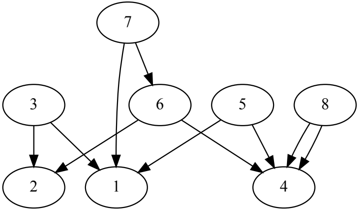
The set A_k is called an extremal postage-stamp basis.

Improved computation and communication

Extremal postage stamp bases

Computing powers of the query...

- when using windowing



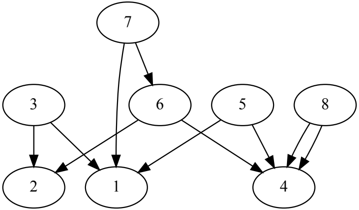
KU LEUVEN

Improved computation and communication

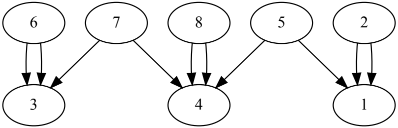
Extremal postage stamp bases

Computing powers of the query...

- when using windowing

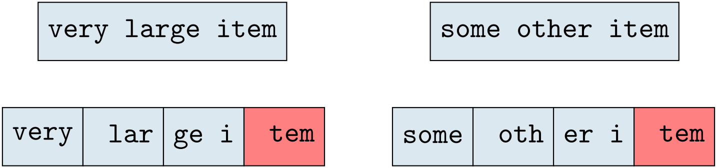


- when using extremal postage stamp bases



Improved computation and communication

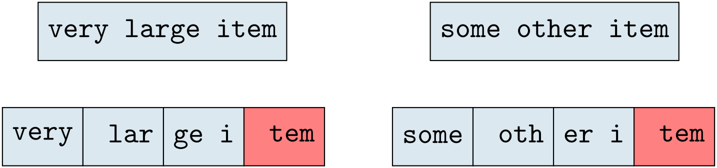
Dealing with large items



- Split items into multiple parts.

Improved computation and communication

Dealing with large items



- Split items into multiple parts.
- Perform OPRF before splitting the items to protect from partial item leakage.

Improved computation and communication

Results

$\lvert X \rvert$	$\lvert Y \rvert$	Protocol	Sender offline (s)	Sender online (s)
2^{28}	1024	This work (T=24)	3,680	7.80
		Chen et al. (T=32)	4,628	12.1
		LowMC-GC-PSI	1,869	0.93
		ECC-NR-PSI	52,332	1.34
2^{20}	5535	This work	28	3.23
		Chen et al.	43	4.23
		LowMC-GC-PSI	7.3	5.63
		ECC-NR-PSI	242	5.93

Improved computation and communication

Results

$\lvert X \rvert$	$\lvert Y \rvert$	Protocol	Offline comm. and receiver storage (MB)	Comm. (MB)
2^{28}	1024	This work (T=24)	0	6.08
		Chen et al. (T=32)	0	18.57
		LowMC-GC-PSI	1,072	24.01
		ECC-NR-PSI	1,072	6.06
2^{20}	5535	This work	0	5.39
		Chen et al.	0	11.50
		LowMC-GC-PSI	4.2	129.73
		ECC-NR-PSI	4.2	32.71

Optimizing for communication complexity

Frobenius automorphism

- The Frobenius automorphism maps any $y \in \mathbb{F}_{t^d}$ to $\mathrm{Frob}(y, r) = y^{t^r}$.
- This operation introduces much less noise than homomorphic multiplication.
- We can get depth $\mathcal{O}(\log \log D)$ sending only $\mathcal{O}(1)$ pre-computed powers instead of $\mathcal{O}(\log D)$.

Optimizing for communication complexity

Frobenius automorphism

Example

Take a plaintext modulus $t = 2$; the Frobenius operation can compute $[\![x]\!] \mapsto [\![x^{2^i}]\!]$.
Suppose the sender has 255 values in its set.
To use Paterson-Stockmeyer, the sender needs:

- **Low powers** $[\![y]\!]^2, [\![y]\!]^3, \ldots, [\![y]\!]^{15}$
- **High powers:** $[\![y]\!]^{16}, [\![y]\!]^{32}, [\![y]\!]^{48}, \ldots, [\![y]\!]^{240}$

Optimizing for communication complexity

Frobenius automorphism

Example

Take a plaintext modulus $t = 2$; the Frobenius operation can compute $[\![x]\!] \mapsto [\![x^{2^i}]\!]$.
Suppose the sender has 255 values in its set.
To use Paterson-Stockmeyer, the sender needs:

- **Low powers** $[\![y]\!]^2, [\![y]\!]^3, \ldots, [\![y]\!]^{15}$
- **High powers:** $[\![y]\!]^{16}, [\![y]\!]^{32}, [\![y]\!]^{48}, \ldots, [\![y]\!]^{240}$

The receiver sends only $[\![y]\!]$. The sender calculates:

- $[\![y]\!], [\![y^2]\!], [\![y^4]\!], [\![y^8]\!]$ with depth 0.

Optimizing for communication complexity

Frobenius automorphism

Example

Take a plaintext modulus $t = 2$; the Frobenius operation can compute $[\![x]\!] \mapsto [\![x^{2^i}]\!]$.
Suppose the sender has 255 values in its set.
To use Paterson-Stockmeyer, the sender needs:

- **Low powers** $[\![y]\!]^2, [\![y]\!]^3, \ldots, [\![y]\!]^{15}$
- **High powers:** $[\![y]\!]^{16}, [\![y]\!]^{32}, [\![y]\!]^{48}, \ldots, [\![y]\!]^{240}$

The receiver sends only $[\![y]\!]$. The sender calculates:

- $[\![y]\!], [\![y^2]\!], [\![y^4]\!], [\![y^8]\!]$ with depth 0.
- $[\![y^3]\!] = [\![y]\!] \cdot [\![y^2]\!]$, $[\![y^5]\!] = [\![y]\!] \cdot [\![y^4]\!]$, $[\![y^7]\!] = [\![y]\!] \cdot [\![y^2]\!] \cdot [\![y^4]\!]$, $[\![y^9]\!] = [\![y]\!] \cdot [\![y^8]\!]$,

KU LEUVEN

Optimizing for communication complexity

Frobenius automorphism

Example

Take a plaintext modulus $t = 2$; the Frobenius operation can compute $[\![x]\!] \mapsto [\![x^{2^i}]\!]$.
Suppose the sender has 255 values in its set.
To use Paterson-Stockmeyer, the sender needs:

- **Low powers** $[\![y]\!]^2, [\![y]\!]^3, \ldots, [\![y]\!]^{15}$
- **High powers:** $[\![y]\!]^{16}, [\![y]\!]^{32}, [\![y]\!]^{48}, \ldots, [\![y]\!]^{240}$

The receiver sends only $[\![y]\!]$. The sender calculates:

- $[\![y]\!], [\![y^2]\!], [\![y^4]\!], [\![y^8]\!]$ with depth 0.
- $[\![y^3]\!] = [\![y]\!] \cdot [\![y^2]\!]$, $[\![y^5]\!] = [\![y]\!] \cdot [\![y^4]\!]$, $[\![y^7]\!] = [\![y]\!] \cdot [\![y^2]\!] \cdot [\![y^4]\!]$, $[\![y^9]\!] = [\![y]\!] \cdot [\![y^8]\!]$,
- $[\![y^{11}]\!] = [\![y]\!] \cdot [\![y^2]\!] \cdot [\![y^8]\!]$, $[\![y^{13}]\!] = [\![y]\!] \cdot [\![y^4]\!] \cdot [\![y^8]\!]$, $[\![y^{15}]\!] = [\![y]\!] \cdot [\![y^2]\!] \cdot [\![y^4]\!] \cdot [\![y^8]\!]$

Optimizing for communication complexity

Results

$\lvert Y \rvert$	Online communication (MB)			
	$\lvert X \rvert = 2^{20}$	2^{22}	2^{24}	2^{26}
1245	2.09	2.28	2.28	2.28
1024 (Chen et al.)	6.45	-	9.02	-
558	1.27	1.27	1.27	1.36
512 (Chen et al.)	5.01	-	10.64	-
341	1.10	1.32	1.32	1.32
256 (Chen et al.)	4.73	-	13.58	-
210	0.72	0.76	0.76	0.76
128 (Chen et al.)	4.69	-	18.32	-
126	0.63	0.63	0.66	-

Optimizing for communication complexity

Results

$\lvert X \rvert$	$\lvert Y \rvert$	**Offline (s)** $T = 24$	**Online (s)** $T = 24$
2^{26}	1245	296	889
	210	1450	1640
2^{24}	1245	64.7	338
	210	305	354
2^{22}	1245	14.1	140
	210	65.2	105
2^{20}	1245	2.88	43.4
	210	14.0	38.7

Conclusion

When intersecting 2^{28} and 2048 item sets:

- Reduced computation by 71%, communication by 63%.

When intersecting 2^{24} and 4096 item sets:

- Reduced computation by 27%, communication by 63%.

PSI with **nearly constant communication** in the larger set size.

➤ Optimizations also apply in the **labeled mode**.

Implementation available at:
`https://github.com/microsoft/APSI/`