MoMA LAB

NDSS

Recurrent Private Set Intersection for Unbalanced Databases with Cuckoo Hashing and Leveled FHE

Eduardo Chielle

Center for Cyber Security
New York University Abu Dhabi

Michail Maniatakos

Center for Cyber Security
New York University Abu Dhabi

MoMA LAB

NDSS

Recurrent Private Set Intersection for Unbalanced Databases with Cuckoo Hashing and Leveled FHE

Eduardo Chielle

Center for Cyber Security
New York University Abu Dhabi

Michail Maniatakos

Center for Cyber Security
New York University Abu Dhabi

MoMA LAB

NDSS

Recurrent Private Set Intersection for Unbalanced Databases with Cuckoo Hashing and Leveled FHE

Eduardo Chielle

Center for Cyber Security
New York University Abu Dhabi

Michail Maniatakos

Center for Cyber Security
New York University Abu Dhabi

MoMA LAB

NDSS

Recurrent Private Set Intersection for Unbalanced Databases with Cuckoo Hashing and Leveled FHE

Eduardo Chielle

Center for Cyber Security
New York University Abu Dhabi

Michail Maniatakos

Center for Cyber Security
New York University Abu Dhabi

MoMA LAB

NDSS

Recurrent Private Set Intersection for Unbalanced Databases with Cuckoo Hashing and Leveled FHE

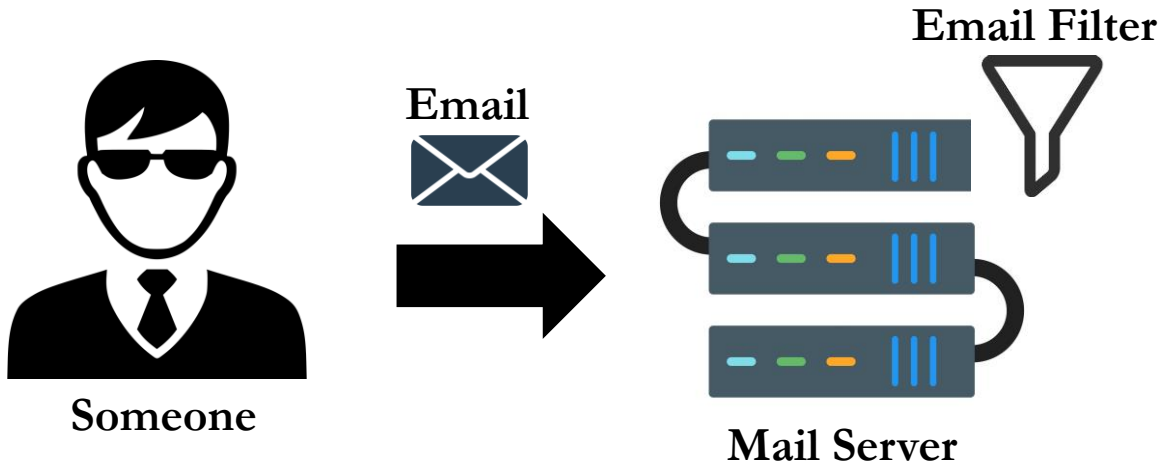
Eduardo Chielle

Center for Cyber Security
New York University Abu Dhabi

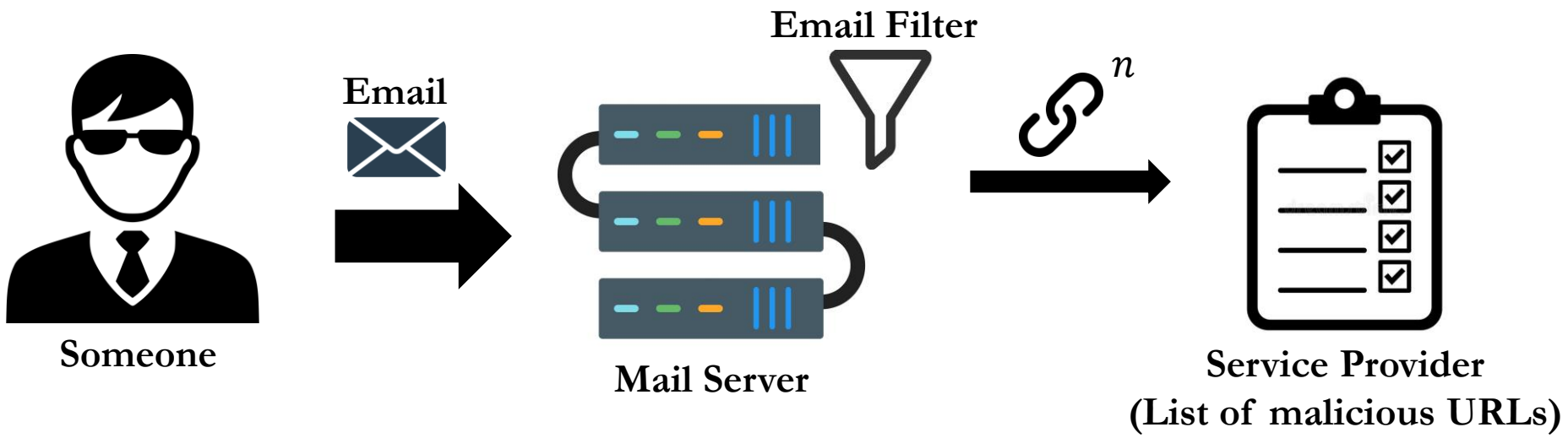
Michail Maniatakos

Center for Cyber Security
New York University Abu Dhabi

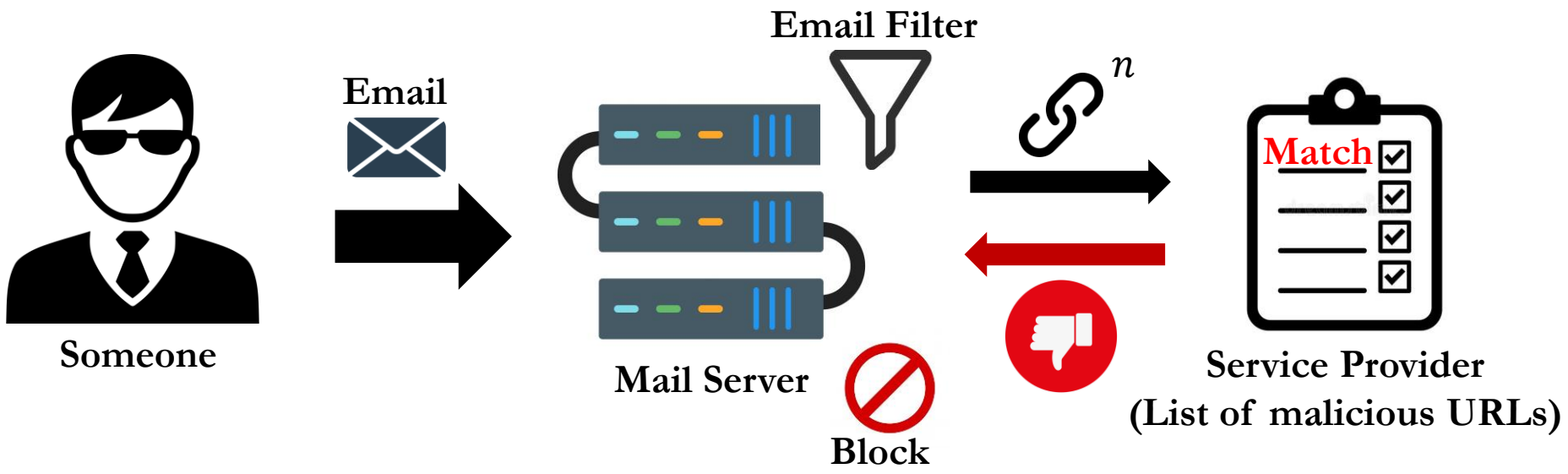
URL Denylisting / Email Filtering



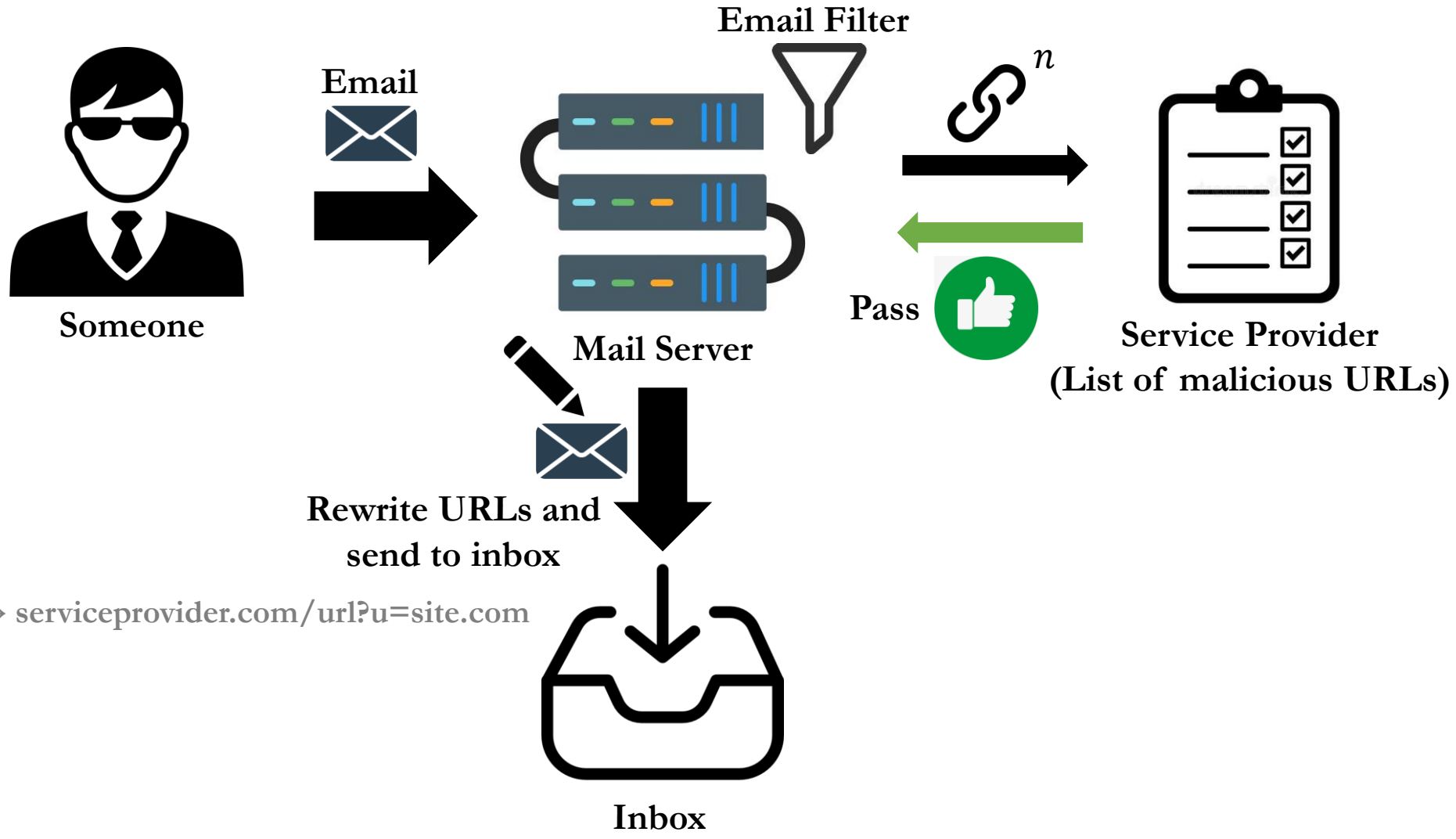
URL Denylisting / Email Filtering



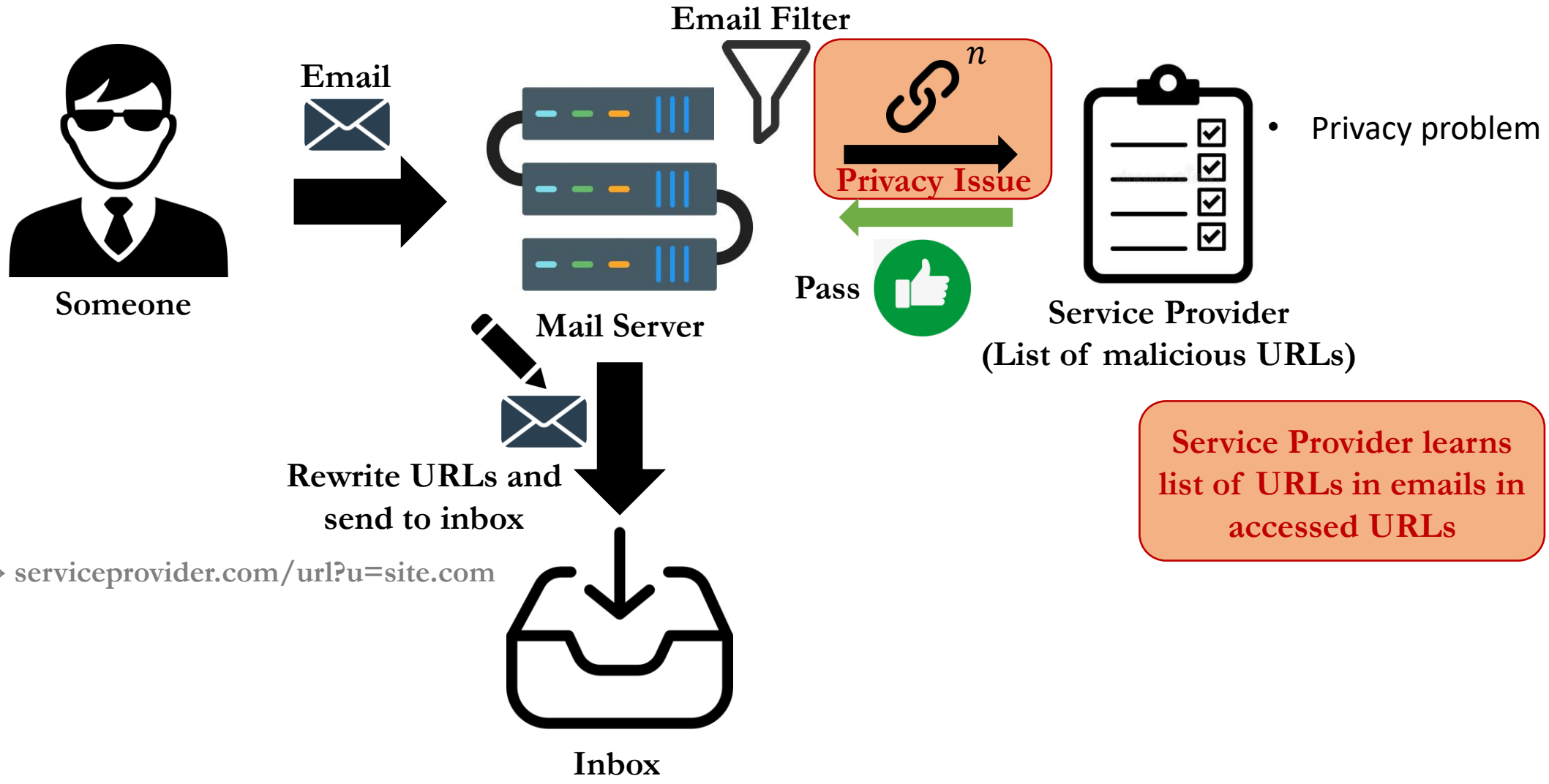
URL Denylisting / Email Filtering



URL Denylisting / Email Filtering



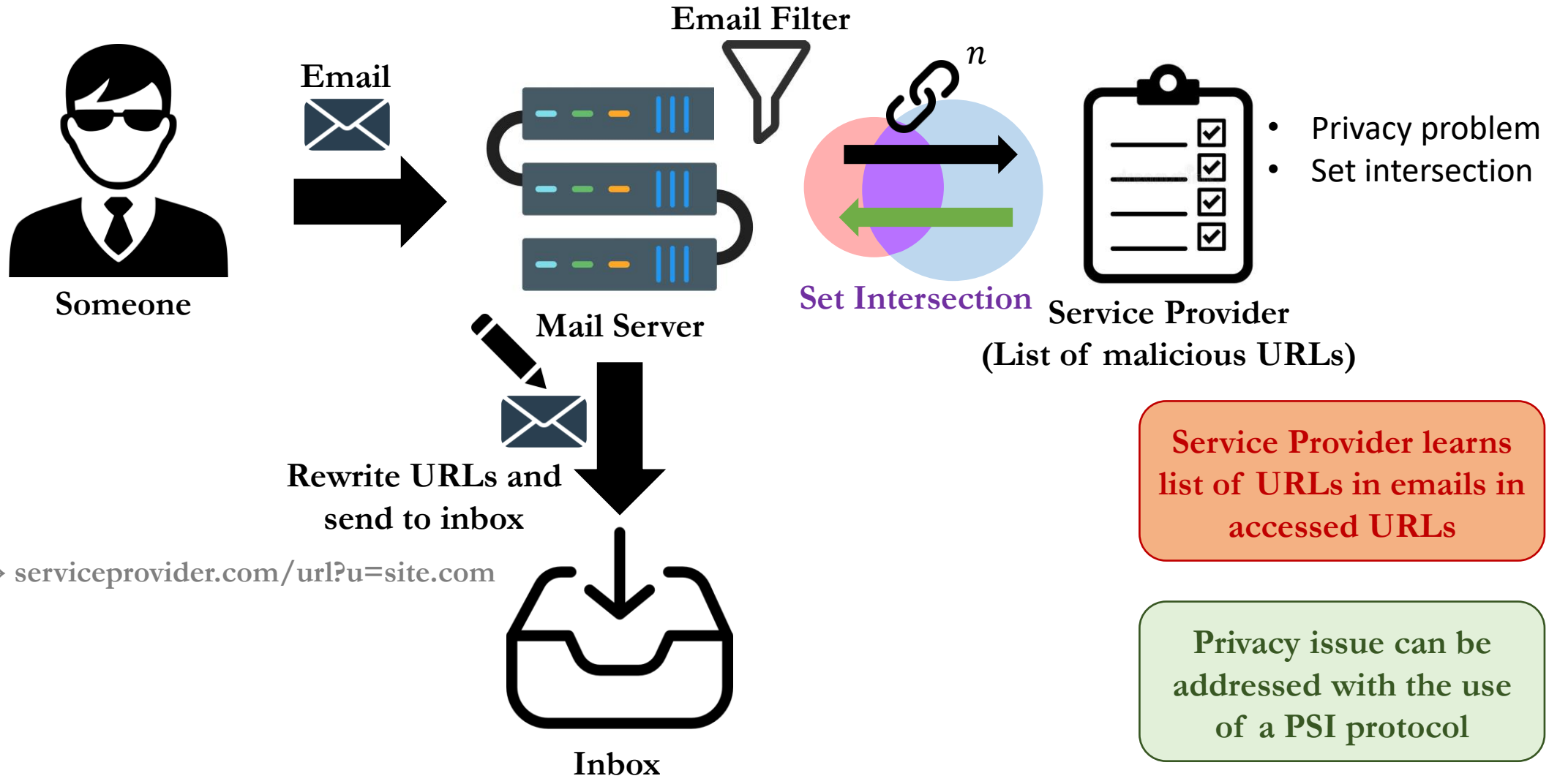
URL Denylisting / Email Filtering



- Privacy problem

Service Provider learns list of URLs in emails in accessed URLs

URL Denylisting / Email Filtering



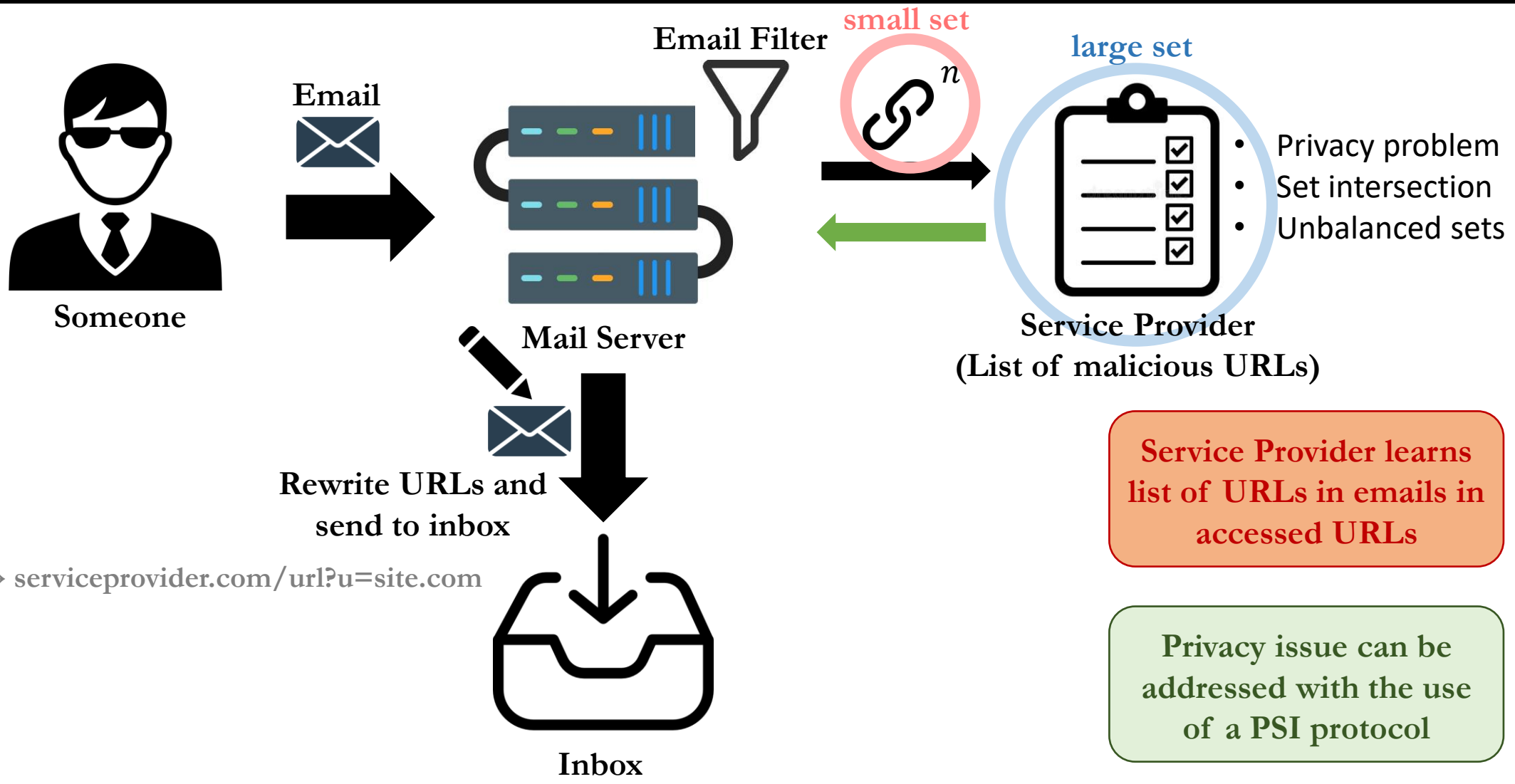
- Privacy problem
- Set intersection

site.com → serviceprovider.com/url?u=site.com

Service Provider learns list of URLs in emails in accessed URLs

Privacy issue can be addressed with the use of a PSI protocol

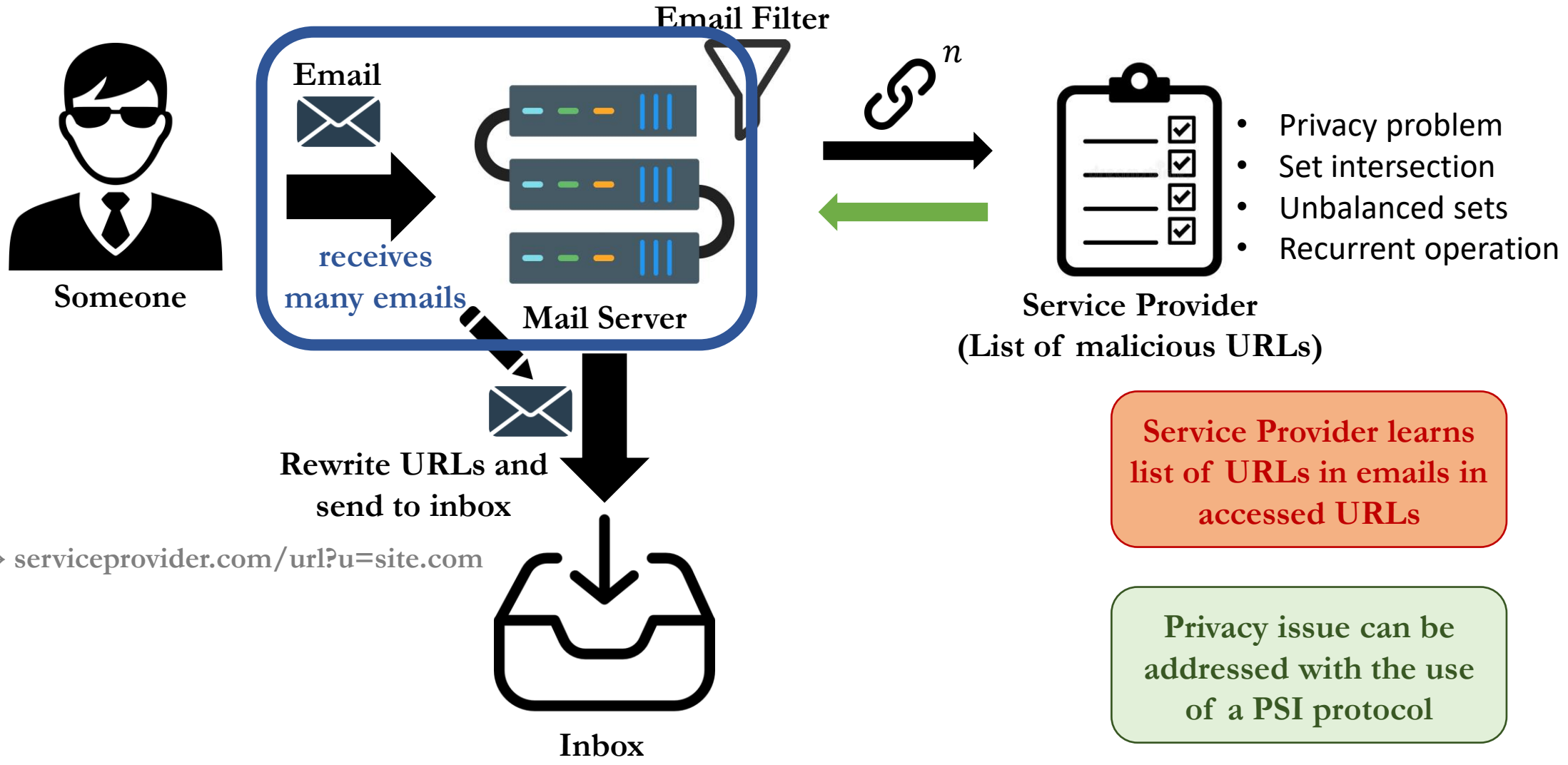
URL Denylisting / Email Filtering



Service Provider learns list of URLs in emails in accessed URLs

Privacy issue can be addressed with the use of a PSI protocol

URL Denylisting / Email Filtering



Contributions

- **Developed a PSI protocol for recurrent set intersection**

Contributions

- **Developed a PSI protocol for recurrent set intersection**
- **One-time cost on the large set and linear recurrent cost on the smaller set**

Contributions

- **Developed a PSI protocol for recurrent set intersection**
- **One-time cost on the large set and linear recurrent cost on the smaller set**
- **Proposed optimizations to reduce computation time and communication**

Contributions

- **Developed a PSI protocol for recurrent set intersection**
- **One-time cost on the large set and linear recurrent cost on the smaller set**
- **Proposed optimizations to reduce computation time and communication**
- **Reduce intersection time by one/two OOM compared to SOTA on slow/fast networks**

Recurrent time (s) for $|X| \approx 2^{20}, |Y| = 4$

Protocol	10 Gbps	100 Mbps
This work	0.03	0.24
CLR [1]	2.21	2.81
KKRT [2]	1.67	6.46
PSSZ [3]	0.88	2.94

[1] Hao Chen, Kim Laine, and Peter Rindal. "Fast private set intersection from homomorphic encryption." *CCS*, 2017.
[2] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. "Efficient batched oblivious PRF with applications to private set intersection." CCS, 2016.
[3] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. "Phasing: Private set intersection using permutation-based hashing." *USENIX Security, 2015.*

Proposed PSI Protocol

(Blueprint) Basic Protocol

Protocol 1 Basic Protocol

Input: $\mathcal{S}$ inputs set X and $\mathcal{R}$ inputs set Y. Both sets consist of bit strings of length σ. Sizes $|X|$, $|Y|$, and σ are public.
Output: $\mathcal{S}$ outputs $\perp$; $\mathcal{R}$ outputs $X \cap Y$.

1) **Setup**: Parties agree on a Fully Homomorphic Encryption scheme and encryption parameters (n, q, t).
2) **Set encryption**: $\mathcal{S}$ encrypts each element $x_i \in X$ under its own key, and sends the $|X|$ ciphertexts $\overline{X} = (\overline{x_1}, ..., \overline{x_{|X|}})$ to $\mathcal{R}$.
3) **Computing intersection**: For each $y_j \in Y$, $\mathcal{R}$:
   a) samples a random plaintext $r_j \in \mathbb{Z}_t$ from a uniform distribution;
   b) homomorphically computes
   $$\overline{d_j} = r_j + \prod_{i=1}^{|X|} (\overline{x_i} - y_j)$$
   c) encrypts $r_j \mapsto \widehat{r_j}$ under its own key;
   d) sends $\overline{d_j}$ and $\widehat{r_j}$ to $\mathcal{S}$.
4) **Decryption**: For each $(\overline{d_j}, \widehat{r_j})$, $\mathcal{S}$:
   a) decrypts $\overline{d_j} \mapsto d_j$;
   b) samples a non-zero random plaintext value $\rho_j \in \mathbb{Z}_t \backslash \{0\}$ from a uniform distribution;
   c) homomorphically computes
   $$\widehat{e_j} = (\widehat{r_j} - d_j) \cdot \rho_j$$
   d) sends $\widehat{e_j}$ to $\mathcal{R}$.
5) **Result**: $\mathcal{R}$ decrypts $\widehat{e_j} \mapsto e_j$ and outputs
   $$X \cap Y = \cup_{j=1}^{|Y|} \{y_j : e_j = 0\}$$

Receiver **Sender**

(Blueprint) Basic Protocol

Protocol 1 Basic Protocol

Input: $\mathcal{S}$ inputs set X and $\mathcal{R}$ inputs set Y. Both sets consist of bit strings of length σ. Sizes $|X|$, $|Y|$, and σ are public.
Output: $\mathcal{S}$ outputs $\perp$; $\mathcal{R}$ outputs $X \cap Y$.

1) **Setup**: Parties agree on a Fully Homomorphic Encryption scheme and encryption parameters (n, q, t).
2) **Set encryption**: $\mathcal{S}$ encrypts each element $x_i \in X$ under its own key, and sends the $|X|$ ciphertexts $\overline{X} = (\overline{x_1}, ..., \overline{x_{|X|}})$ to $\mathcal{R}$.
3) **Computing intersection**: For each $y_j \in Y$, $\mathcal{R}$:
   a) samples a random plaintext $r_j \in \mathbb{Z}_t$ from a uniform distribution;
   b) homomorphically computes

$$\overline{d_j} = r_j + \prod_{i=1}^{|X|} (\overline{x_i} - y_j)$$

   c) encrypts $r_j \mapsto \widehat{r_j}$ under its own key;
   d) sends $\overline{d_j}$ and $\widehat{r_j}$ to $\mathcal{S}$.
4) **Decryption**: For each $(\overline{d_j}, \widehat{r_j})$, $\mathcal{S}$:
   a) decrypts $\overline{d_j} \mapsto d_j$;
   b) samples a non-zero random plaintext value $\rho_j \in \mathbb{Z}_t \backslash \{0\}$ from a uniform distribution;
   c) homomorphically computes

$$\widehat{e_j} = (\widehat{r_j} - d_j) \cdot \rho_j$$

   d) sends $\widehat{e_j}$ to $\mathcal{R}$.
5) **Result**: $\mathcal{R}$ decrypts $\widehat{e_j} \mapsto e_j$ and outputs

$$X \cap Y = \cup_{j=1}^{|Y|} \{y_j : e_j = 0\}$$

Receiver **Sender**

encrypts set with Sender's key

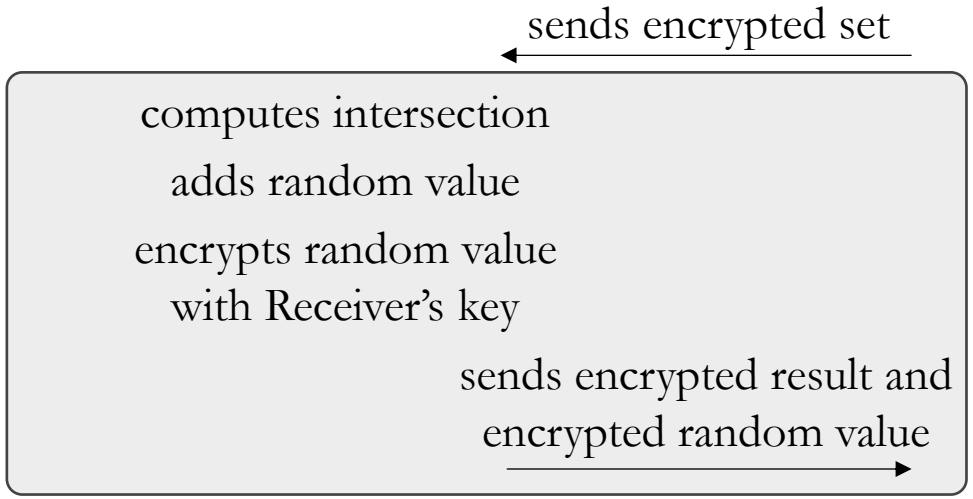
sends encrypted set

(Blueprint) Basic Protocol

Protocol 1 Basic Protocol

Input: $\mathcal{S}$ inputs set X and $\mathcal{R}$ inputs set Y. Both sets consist of bit strings of length σ. Sizes $|X|$, $|Y|$, and σ are public.
Output: $\mathcal{S}$ outputs $\perp$; $\mathcal{R}$ outputs $X \cap Y$.

1) **Setup**: Parties agree on a Fully Homomorphic Encryption scheme and encryption parameters (n, q, t).
2) **Set encryption**: $\mathcal{S}$ encrypts each element $x_i \in X$ under its own key, and sends the $|X|$ ciphertexts $\overline{X} = (\overline{x_1}, ..., \overline{x_{|X|}})$ to $\mathcal{R}$.
3) **Computing intersection**: For each $y_j \in Y$, $\mathcal{R}$:
   a) samples a random plaintext $r_j \in \mathbb{Z}_t$ from a uniform distribution;
   b) homomorphically computes
   $$\overline{d_j} = r_j + \prod_{i=1}^{|X|} (\overline{x_i} - y_j)$$
   c) encrypts $r_j \mapsto \widehat{r_j}$ under its own key;
   d) sends $\overline{d_j}$ and $\widehat{r_j}$ to $\mathcal{S}$.
4) **Decryption**: For each $(\overline{d_j}, \widehat{r_j})$, $\mathcal{S}$:
   a) decrypts $\overline{d_j} \mapsto d_j$;
   b) samples a non-zero random plaintext value $\rho_j \in \mathbb{Z}_t \backslash \{0\}$ from a uniform distribution;
   c) homomorphically computes
   $$\widehat{e_j} = (\widehat{r_j} - d_j) \cdot \rho_j$$
   d) sends $\widehat{e_j}$ to $\mathcal{R}$.
5) **Result**: $\mathcal{R}$ decrypts $\widehat{e_j} \mapsto e_j$ and outputs
   $$X \cap Y = \cup_{j=1}^{|Y|} \{y_j : e_j = 0\}$$



(Blueprint) Basic Protocol

Protocol 1 Basic Protocol

Input: $\mathcal{S}$ inputs set X and $\mathcal{R}$ inputs set Y. Both sets consist of bit strings of length σ. Sizes $|X|$, $|Y|$, and σ are public.
Output: $\mathcal{S}$ outputs $\perp$; $\mathcal{R}$ outputs $X \cap Y$.

1) **Setup**: Parties agree on a Fully Homomorphic Encryption scheme and encryption parameters (n, q, t).
2) **Set encryption**: $\mathcal{S}$ encrypts each element $x_i \in X$ under its own key, and sends the $|X|$ ciphertexts $\overline{X} = (\overline{x_1}, ..., \overline{x_{|X|}})$ to $\mathcal{R}$.
3) **Computing intersection**: For each $y_j \in Y$, $\mathcal{R}$:
   a) samples a random plaintext $r_j \in \mathbb{Z}_t$ from a uniform distribution;
   b) homomorphically computes

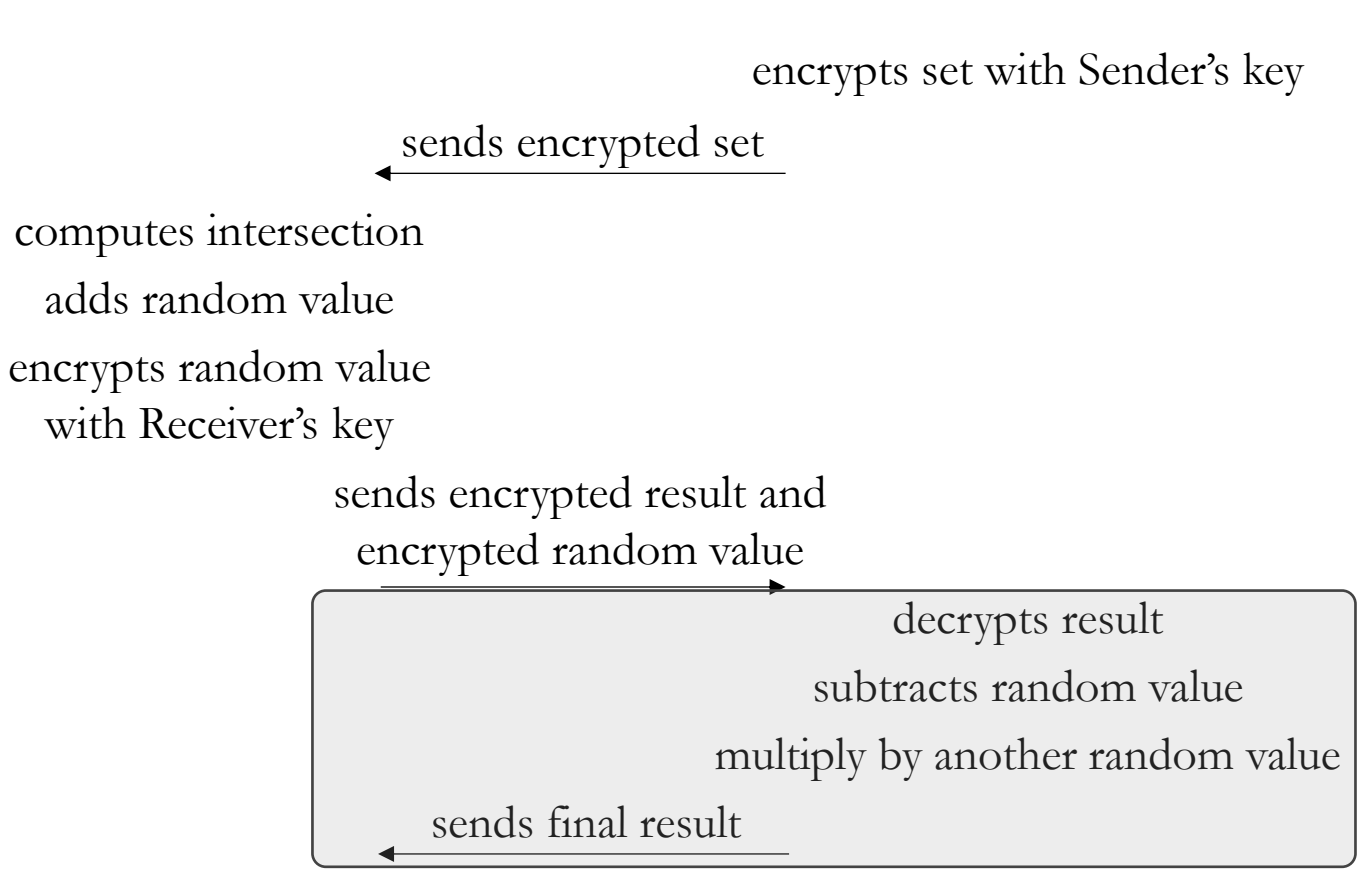
$$\overline{d_j} = r_j + \prod_{i=1}^{|X|} (\overline{x_i} - y_j)$$

   c) encrypts $r_j \mapsto \widehat{r_j}$ under its own key;
   d) sends $\overline{d_j}$ and $\widehat{r_j}$ to $\mathcal{S}$.
4) **Decryption**: For each $(\overline{d_j}, \widehat{r_j})$, $\mathcal{S}$:
   a) decrypts $\overline{d_j} \mapsto d_j$;
   b) samples a non-zero random plaintext value $\rho_j \in \mathbb{Z}_t \backslash \{0\}$ from a uniform distribution;
   c) homomorphically computes

$$\widehat{e_j} = (\widehat{r_j} - d_j) \cdot \rho_j$$

   d) sends $\widehat{e_j}$ to $\mathcal{R}$.
5) **Result**: $\mathcal{R}$ decrypts $\widehat{e_j} \mapsto e_j$ and outputs

$$X \cap Y = \cup_{j=1}^{|Y|} \{y_j : e_j = 0\}$$

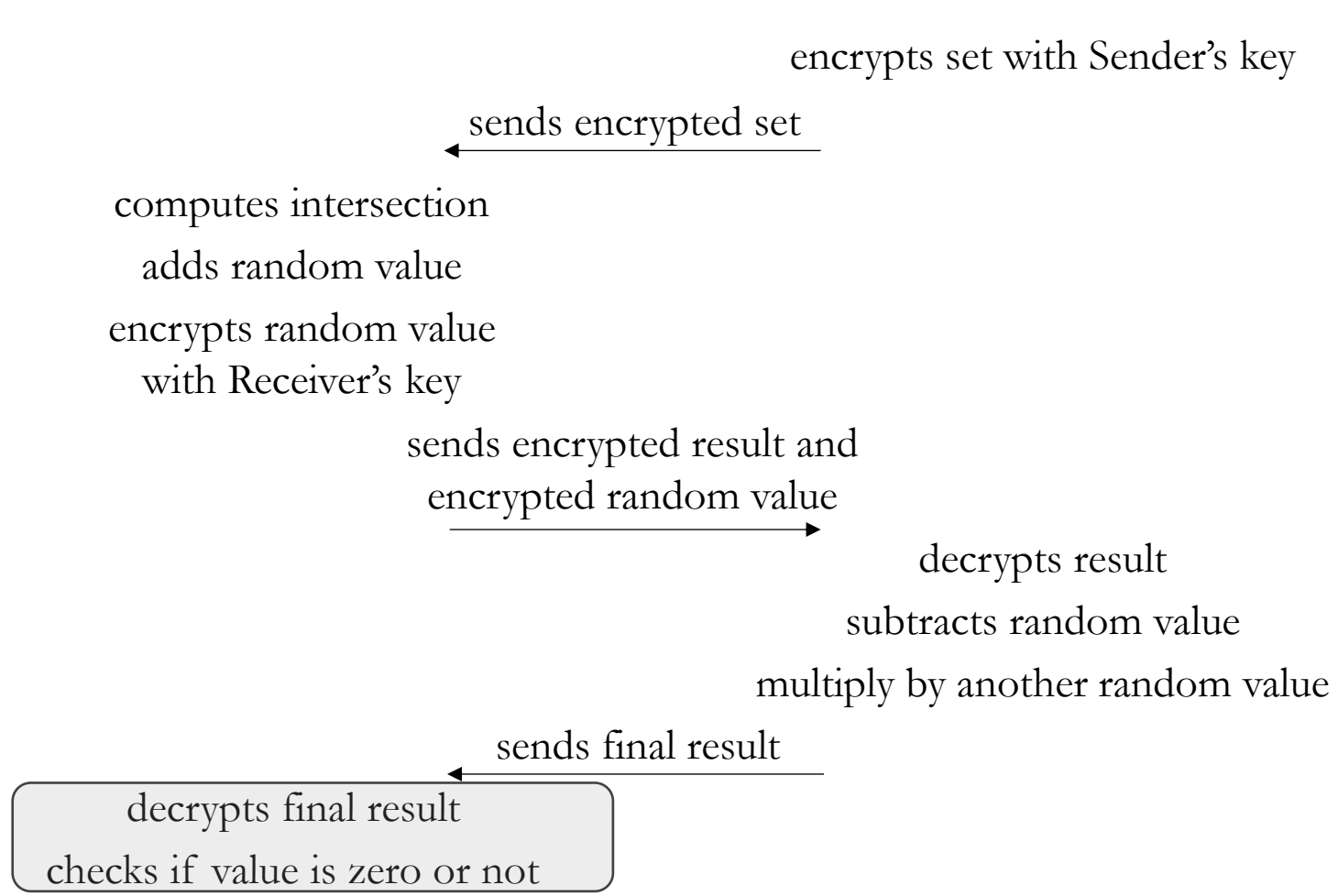


(Blueprint) Basic Protocol

Protocol 1 Basic Protocol

Input: $\mathcal{S}$ inputs set X and $\mathcal{R}$ inputs set Y. Both sets consist of bit strings of length σ. Sizes $|X|$, $|Y|$, and σ are public.
Output: $\mathcal{S}$ outputs $\perp$; $\mathcal{R}$ outputs $X \cap Y$.

1) **Setup**: Parties agree on a Fully Homomorphic Encryption scheme and encryption parameters (n, q, t).
2) **Set encryption**: $\mathcal{S}$ encrypts each element $x_i \in X$ under its own key, and sends the $|X|$ ciphertexts $\overline{X} = (\overline{x_1}, ..., \overline{x_{|X|}})$ to $\mathcal{R}$.
3) **Computing intersection**: For each $y_j \in Y$, $\mathcal{R}$:
   a) samples a random plaintext $r_j \in \mathbb{Z}_t$ from a uniform distribution;
   b) homomorphically computes
   $$\overline{d_j} = r_j + \prod_{i=1}^{|X|} (\overline{x_i} - y_j)$$
   c) encrypts $r_j \mapsto \widehat{r_j}$ under its own key;
   d) sends $\overline{d_j}$ and $\widehat{r_j}$ to $\mathcal{S}$.
4) **Decryption**: For each $(\overline{d_j}, \widehat{r_j})$, $\mathcal{S}$:
   a) decrypts $\overline{d_j} \mapsto d_j$;
   b) samples a non-zero random plaintext value $\rho_j \in \mathbb{Z}_t \backslash \{0\}$ from a uniform distribution;
   c) homomorphically computes
   $$\widehat{e_j} = (\widehat{r_j} - d_j) \cdot \rho_j$$
   d) sends $\widehat{e_j}$ to $\mathcal{R}$.
5) **Result**: $\mathcal{R}$ decrypts $\widehat{e_j} \mapsto e_j$ and outputs
   $$X \cap Y = \cup_{j=1}^{|Y|} \{y_j : e_j = 0\}$$



(Blueprint) Basic Protocol

Protocol 1 Basic Protocol

Input: $\mathcal{S}$ inputs set X and $\mathcal{R}$ inputs set Y. Both sets consist of bit strings of length σ. Sizes $|X|$, $|Y|$, and σ are public.
Output: $\mathcal{S}$ outputs $\perp$; $\mathcal{R}$ outputs $X \cap Y$.

1) **Setup**: Parties agree on a Fully Homomorphic Encryption scheme and encryption parameters (n, q, t).
2) **Set encryption**: $\mathcal{S}$ encrypts each element $x_i \in X$ under its own key, and sends the $|X|$ ciphertexts $\overline{X} = (\overline{x_1}, ..., \overline{x_{|X|}})$ to $\mathcal{R}$.
3) **Computing intersection**: For each $y_j \in Y$, $\mathcal{R}$:
   a) samples a random plaintext $r_j \in \mathbb{Z}_t$ from a uniform distribution;
   b) homomorphically computes
   $$\overline{d_j} = r_j + \prod_{i=1}^{|X|} (\overline{x_i} - y_j)$$
   c) encrypts $r_j \mapsto \widehat{r_j}$ under its own key;
   d) sends $\overline{d_j}$ and $\widehat{r_j}$ to $\mathcal{S}$.
4) **Decryption**: For each $(\overline{d_j}, \widehat{r_j})$, $\mathcal{S}$:
   a) decrypts $\overline{d_j} \mapsto d_j$;
   b) samples a non-zero random plaintext value $\rho_j \in \mathbb{Z}_t \backslash \{0\}$ from a uniform distribution;
   c) homomorphically computes
   $$\widehat{e_j} = (\widehat{r_j} - d_j) \cdot \rho_j$$
   d) sends $\widehat{e_j}$ to $\mathcal{R}$.
5) **Result**: $\mathcal{R}$ decrypts $\widehat{e_j} \mapsto e_j$ and outputs
   $$X \cap Y = \cup_{j=1}^{|Y|} \{y_j : e_j = 0\}$$

Receiver | **Sender**

One-time

encrypts set with Sender's key

sends encrypted set ←

computes intersection

adds random value

encrypts random value with Receiver's key

Recurrent

sends encrypted result and encrypted random value →

decrypts result

subtracts random value

multiply by another random value

← sends final result

decrypts final result

checks if value is zero or not

(Blueprint) Basic Protocol

Protocol 1 Basic Protocol

Input: $\mathcal{S}$ inputs set X and $\mathcal{R}$ inputs set Y. Both sets consist of bit strings of length σ. Sizes $|X|$, $|Y|$, and σ are public.
Output: $\mathcal{S}$ outputs $\perp$; $\mathcal{R}$ outputs $X \cap Y$.

1) **Setup**: Parties agree on a Fully Homomorphic Encryption scheme and encryption parameters (n, q, t).
2) **Set encryption**: $\mathcal{S}$ encrypts each element $x_i \in X$ under its own key, and sends the $|X|$ ciphertexts $\overline{X} = (\overline{x_1}, ..., \overline{x_{|X|}})$ to $\mathcal{R}$.
3) **Computing intersection**: For each $y_j \in Y$, $\mathcal{R}$:
   a) samples a random plaintext $r_j \in \mathbb{Z}_t$ from a uniform distribution;
   b) homomorphically computes
   $$\overline{d_j} = r_j + \prod_{i=1}^{|X|} (\overline{x_i} - y_j)$$
   c) encrypts $r_j \mapsto \widehat{r_j}$ under its own key;
   d) sends $\overline{d_j}$ and $\widehat{r_j}$ to $\mathcal{S}$.
4) **Decryption**: For each $(\overline{d_j}, \widehat{r_j})$, $\mathcal{S}$:
   a) decrypts $\overline{d_j} \mapsto d_j$;
   b) samples a non-zero random plaintext value $\rho_j \in \mathbb{Z}_t \backslash \{0\}$ from a uniform distribution;
   c) homomorphically computes
   $$\widehat{e_j} = (\widehat{r_j} - d_j) \cdot \rho_j$$
   d) sends $\widehat{e_j}$ to $\mathcal{R}$.
5) **Result**: $\mathcal{R}$ decrypts $\widehat{e_j} \mapsto e_j$ and outputs
   $$X \cap Y = \cup_{j=1}^{|Y|} \{y_j : e_j = 0\}$$

Assuming a Sender's set with one million entries

	Receiver	Sender
One-time		encrypts set with Sender's key
	sends encrypted set ≈ 7 Terabytes →	
Recurrent	computes intersection	Linear search (≈ 1 million multiplications)
	adds random value	
	encrypts random value with Receiver's key	
	sends encrypted result and encrypted random value →	
		decrypts result
		subtracts random value
		multiply by another random value
	← sends final result	
	decrypts final result	
	checks if value is zero or not	

Solving The Linear Search Problem

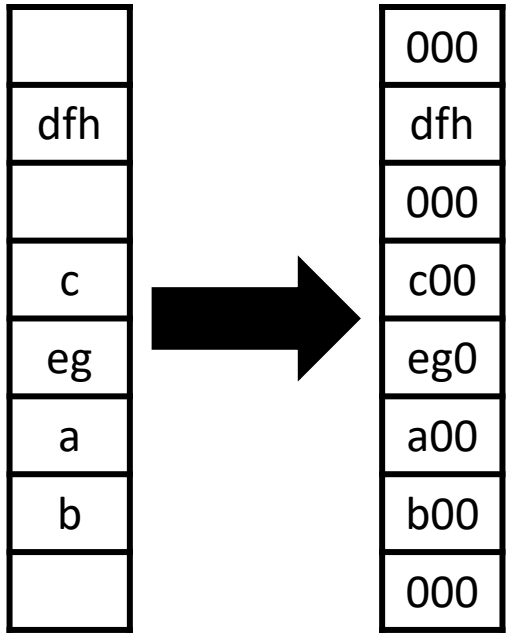
Sender hashes set *X* before sending it to Receiver.

Solving The Linear Search Problem

Sender hashes set X before sending it to Receiver.

Constraint: All bins must have the same number of items.

Simple hashing



Hash functions: h

Maximum load

$$O\left(\frac{\log n}{\log \log n}\right)$$

Number of entries

$$O\left(\frac{n \log n}{\log \log n}\right)$$

Multiplicative depth

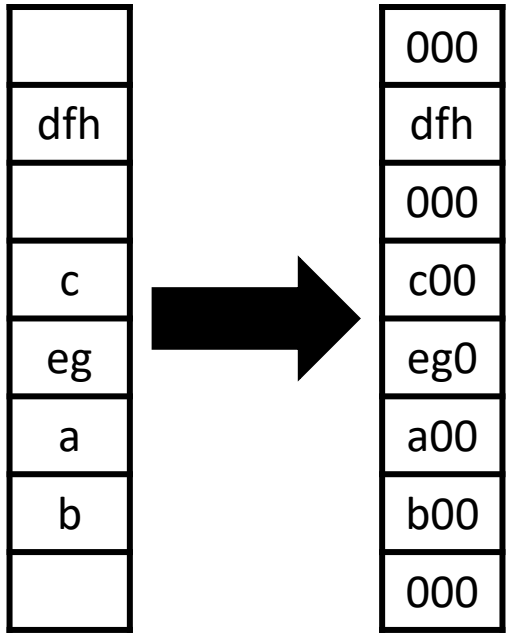
$$O\left(\log \frac{\log n}{\log \log n}\right)$$

Solving The Linear Search Problem

Sender hashes set X before sending it to Receiver.

Constraint: All bins must have the same number of items.

Simple hashing



Hash functions: h

Maximum load

$$O\left(\frac{\log n}{\log \log n}\right)$$

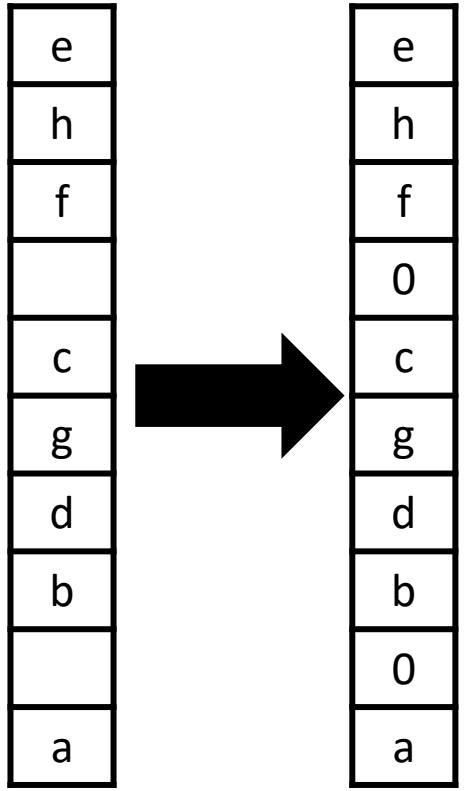
Number of entries

$$O\left(\frac{n \log n}{\log \log n}\right)$$

Multiplicative depth

$$O\left(\log \frac{\log n}{\log \log n}\right)$$

Cuckoo hashing



Hash functions: $h_1, \cdots, h_{|H|}$

Maximum load

1

Number of entries

$$\leq 2n(1+\epsilon)$$

Multiplicative depth

$$O(\log |H|)$$

(Blueprint) Basic Protocol

Protocol 1 Basic Protocol

Input: $\mathcal{S}$ inputs set X and $\mathcal{R}$ inputs set Y. Both sets consist of bit strings of length σ. Sizes $|X|$, $|Y|$, and σ are public.
Output: $\mathcal{S}$ outputs $\perp$; $\mathcal{R}$ outputs $X \cap Y$.

1) **Setup**: Parties agree on a Fully Homomorphic Encryption scheme and encryption parameters (n, q, t).
2) **Set encryption**: $\mathcal{S}$ encrypts each element $x_i \in X$ under its own key, and sends the $|X|$ ciphertexts $\overline{X} = (\overline{x_1}, ..., \overline{x_{|X|}})$ to $\mathcal{R}$.
3) **Computing intersection**: For each $y_j \in Y$, $\mathcal{R}$:
   a) samples a random plaintext $r_j \in \mathbb{Z}_t$ from a uniform distribution;
   b) homomorphically computes
   $$\overline{d_j} = r_j + \prod_{i=1}^{|X|} (\overline{x_i} - y_j)$$
   c) encrypts $r_j \mapsto \widehat{r_j}$ under its own key;
   d) sends $\overline{d_j}$ and $\widehat{r_j}$ to $\mathcal{S}$.
4) **Decryption**: For each $(\overline{d_j}, \widehat{r_j})$, $\mathcal{S}$:
   a) decrypts $\overline{d_j} \mapsto d_j$;
   b) samples a non-zero random plaintext value $\rho_j \in \mathbb{Z}_t \backslash \{0\}$ from a uniform distribution;
   c) homomorphically computes
   $$\widehat{e_j} = (\widehat{r_j} - d_j) \cdot \rho_j$$
   d) sends $\widehat{e_j}$ to $\mathcal{R}$.
5) **Result**: $\mathcal{R}$ decrypts $\widehat{e_j} \mapsto e_j$ and outputs
   $$X \cap Y = \cup_{j=1}^{|Y|} \{y_j : e_j = 0\}$$

Assuming a Sender's set with one million entries

	Receiver	Sender
One-time		encrypts set with Sender's key
	sends encrypted set ≈ 7 Terabytes →	
Recurrent	computes intersection	Linear search (≈ 1 million multiplications)
	adds random value	
	encrypts random value with Receiver's key	
	sends encrypted result and encrypted random value →	
		decrypts result
		subtracts random value
		multiply by another random value
	← sends final result	
	decrypts final result	
	checks if value is zero or not	

(Blueprint) Basic Protocol

Protocol 1 Basic Protocol

Input: $\mathcal{S}$ inputs set X and $\mathcal{R}$ inputs set Y. Both sets consist of bit strings of length σ. Sizes $|X|$, $|Y|$, and σ are public.
Output: $\mathcal{S}$ outputs $\perp$; $\mathcal{R}$ outputs $X \cap Y$.

1) **Setup**: Parties agree on a Fully Homomorphic Encryption scheme and encryption parameters (n, q, t).
2) **Set encryption**: $\mathcal{S}$ encrypts each element $x_i \in X$ under its own key, and sends the $|X|$ ciphertexts $\overline{X} = (\overline{x_1}, ..., \overline{x_{|X|}})$ to $\mathcal{R}$.
3) **Computing intersection**: For each $y_j \in Y$, $\mathcal{R}$:
   a) samples a random plaintext $r_j \in \mathbb{Z}_t$ from a uniform distribution;
   b) homomorphically computes
   $$\overline{d_j} = r_j + \prod_{i=1}^{|X|} (\overline{x_i} - y_j)$$
   c) encrypts $r_j \mapsto \widehat{r_j}$ under its own key;
   d) sends $\overline{d_j}$ and $\widehat{r_j}$ to $\mathcal{S}$.
4) **Decryption**: For each $(\overline{d_j}, \widehat{r_j})$, $\mathcal{S}$:
   a) decrypts $\overline{d_j} \mapsto d_j$;
   b) samples a non-zero random plaintext value $\rho_j \in \mathbb{Z}_t \backslash \{0\}$ from a uniform distribution;
   c) homomorphically computes
   $$\widehat{e_j} = (\widehat{r_j} - d_j) \cdot \rho_j$$
   d) sends $\widehat{e_j}$ to $\mathcal{R}$.
5) **Result**: $\mathcal{R}$ decrypts $\widehat{e_j} \mapsto e_j$ and outputs
   $$X \cap Y = \cup_{j=1}^{|Y|} \{y_j : e_j = 0\}$$

Assuming a Sender's set with one million entries

Receiver		Sender
		encrypts set with Sender's key
One-time	sends encrypted set ≈ **100 Gigabytes** ←	
computes intersection		**Constant-time search (3 multiplications)**
adds random value		
encrypts random value with Receiver's key		
	sends encrypted result and encrypted random value →	
Recurrent		decrypts result
		subtracts random value
		multiply by another random value
	sends final result ←	
decrypts final result		
checks if value is zero or not		

Reducing Communication on Sender's Set

Pack many entries into one ciphertext.

Reducing Communication on Sender's Set

Pack many entries into one ciphertext.

FHE encrypts plaintext polynomials into ciphertext polynomials.

Message space: $\boldsymbol{M = \mathbb{Z}_t}$ **Plaintext space:** $\boldsymbol{P = \mathbb{Z}_t[x]/(x^n + 1)}$ **Ciphertext space:** $\boldsymbol{C = \mathbb{Z}_q[x]/(x^n + 1)}$

Reducing Communication on Sender's Set

Pack many entries into one ciphertext.

FHE encrypts plaintext polynomials into ciphertext polynomials.

Message space: $\boldsymbol{M = \mathbb{Z}_t}$ **Plaintext space:** $\boldsymbol{P = \mathbb{Z}_t[x]/(x^n + 1)}$ **Ciphertext space:** $\boldsymbol{C = \mathbb{Z}_q[x]/(x^n + 1)}$

Integer

$p = \varepsilon(m) : M \mapsto P$

$c = E(p) : P \mapsto C^2$

Reducing Communication on Sender's Set

Pack many entries into one ciphertext.

FHE encrypts plaintext polynomials into ciphertext polynomials.

Message space: $\boldsymbol{M = \mathbb{Z}_t}$ **Plaintext space:** $\boldsymbol{P = \mathbb{Z}_t[x]/(x^n + 1)}$ **Ciphertext space:** $\boldsymbol{C = \mathbb{Z}_q[x]/(x^n + 1)}$

Integer

$$p = \varepsilon(m) : M \mapsto P$$
$$c = E(p): P \mapsto C^2$$

Batching [4]

$$p = \varepsilon(m_1, m_2, \cdots, m_n) : M^n \mapsto P$$
$$c = E(p): P \mapsto C^2$$

[4] Nigel P. Smart and Frederik Vercauteren. "Fully homomorphic SIMD operations." *Designs, codes and cryptography*, 2014.

Reducing Communication on Sender's Set

Pack many entries into one ciphertext.

FHE encrypts plaintext polynomials into ciphertext polynomials.

Message space: $\boldsymbol{M = \mathbb{Z}_t}$ **Plaintext space:** $\boldsymbol{P = \mathbb{Z}_t[x]/(x^n + 1)}$ **Ciphertext space:** $\boldsymbol{C = \mathbb{Z}_q[x]/(x^n + 1)}$

Integer

$$p = \varepsilon(m) : M \mapsto P$$
$$c = E(p): P \mapsto C^2$$

Batching[4]

$$p = \varepsilon(m_1, m_2, \cdots, m_n) : M^n \mapsto P$$
$$c = E(p): P \mapsto C^2$$

Batching + Packing

$$p = \varepsilon(m_1, m_2, \cdots, m_{kn}) : M^{kn} \mapsto P$$
$$c = E(p): P \mapsto C^2$$

[4] Nigel P. Smart and Frederik Vercauteren. "Fully homomorphic SIMD operations." *Designs, codes and cryptography*, 2014.

Reducing Communication on Sender's Set

Pack many entries into one ciphertext.

FHE encrypts plaintext polynomials into ciphertext polynomials.

Message space: $M = \mathbb{Z}_t$ **Plaintext space:** $P = \mathbb{Z}_t[x]/(x^n + 1)$ **Ciphertext space:** $C = \mathbb{Z}_q[x]/(x^n + 1)$

Integer	Batching [4]	Batching + Packing
$p = \varepsilon(m) : M \mapsto P$	$p = \varepsilon(m_1, m_2, \cdots, m_n) : M^n \mapsto P$	$p = \varepsilon(m_1, m_2, \cdots, m_{kn}) : M^{kn} \mapsto P$
$c = E(p) : P \mapsto C^2$	$c = E(p) : P \mapsto C^2$	$c = E(p) : P \mapsto C^2$

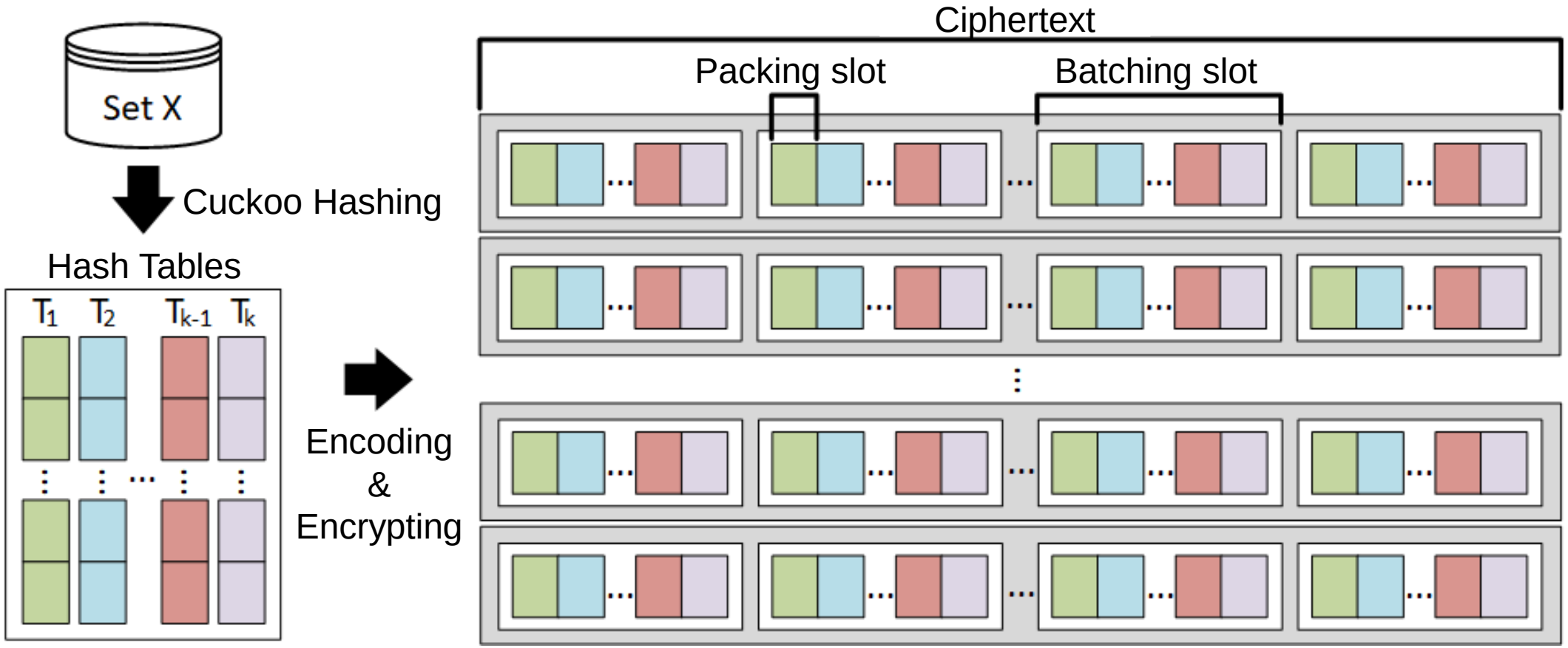
We encode Cuckoo hash table T with proposed "Batching + Packing" method.

≈ 12.5 Megabytes

(Unencrypted is 4 MB)

[4] Nigel P. Smart and Frederik Vercauteren. "Fully homomorphic SIMD operations." *Designs, codes and cryptography*, 2014.

Reducing Communication on Sender's Set



(Blueprint) Basic Protocol

Protocol 1 Basic Protocol

Input: $\mathcal{S}$ inputs set X and $\mathcal{R}$ inputs set Y. Both sets consist of bit strings of length σ. Sizes $|X|$, $|Y|$, and σ are public.
Output: $\mathcal{S}$ outputs $\perp$; $\mathcal{R}$ outputs $X \cap Y$.

1) **Setup**: Parties agree on a Fully Homomorphic Encryption scheme and encryption parameters (n, q, t).
2) **Set encryption**: $\mathcal{S}$ encrypts each element $x_i \in X$ under its own key, and sends the $|X|$ ciphertexts $\overline{X} = (\overline{x_1}, ..., \overline{x_{|X|}})$ to $\mathcal{R}$.
3) **Computing intersection**: For each $y_j \in Y$, $\mathcal{R}$:
   a) samples a random plaintext $r_j \in \mathbb{Z}_t$ from a uniform distribution;
   b) homomorphically computes
   $$\overline{d_j} = r_j + \prod_{i=1}^{|X|} (\overline{x_i} - y_j)$$
   c) encrypts $r_j \mapsto \widehat{r_j}$ under its own key;
   d) sends $\overline{d_j}$ and $\widehat{r_j}$ to $\mathcal{S}$.
4) **Decryption**: For each $(\overline{d_j}, \widehat{r_j})$, $\mathcal{S}$:
   a) decrypts $\overline{d_j} \mapsto d_j$;
   b) samples a non-zero random plaintext value $\rho_j \in \mathbb{Z}_t \backslash \{0\}$ from a uniform distribution;
   c) homomorphically computes
   $$\widehat{e_j} = (\widehat{r_j} - d_j) \cdot \rho_j$$
   d) sends $\widehat{e_j}$ to $\mathcal{R}$.
5) **Result**: $\mathcal{R}$ decrypts $\widehat{e_j} \mapsto e_j$ and outputs
   $$X \cap Y = \cup_{j=1}^{|Y|} \{y_j : e_j = 0\}$$

Assuming a Sender's set with one million entries

Receiver		Sender
		encrypts set with Sender's key
One-time	sends encrypted set ≈ **100 Gigabytes** ←	
computes intersection		**Constant-time search (3 multiplications)**
adds random value		
encrypts random value with Receiver's key		
	sends encrypted result and encrypted random value →	
Recurrent		decrypts result
		subtracts random value
		multiply by another random value
	sends final result ←	
decrypts final result		
checks if value is zero or not		

(Blueprint) Basic Protocol

Protocol 1 Basic Protocol

Input: $\mathcal{S}$ inputs set X and $\mathcal{R}$ inputs set Y. Both sets consist of bit strings of length σ. Sizes $|X|$, $|Y|$, and σ are public.
Output: $\mathcal{S}$ outputs $\perp$; $\mathcal{R}$ outputs $X \cap Y$.

1) **Setup**: Parties agree on a Fully Homomorphic Encryption scheme and encryption parameters (n, q, t).
2) **Set encryption**: $\mathcal{S}$ encrypts each element $x_i \in X$ under its own key, and sends the $|X|$ ciphertexts $\overline{X} = (\overline{x_1}, ..., \overline{x_{|X|}})$ to $\mathcal{R}$.
3) **Computing intersection**: For each $y_j \in Y$, $\mathcal{R}$:
   a) samples a random plaintext $r_j \in \mathbb{Z}_t$ from a uniform distribution;
   b) homomorphically computes
   $$\overline{d_j} = r_j + \prod_{i=1}^{|X|} (\overline{x_i} - y_j)$$
   c) encrypts $r_j \mapsto \widehat{r_j}$ under its own key;
   d) sends $\overline{d_j}$ and $\widehat{r_j}$ to $\mathcal{S}$.
4) **Decryption**: For each $(\overline{d_j}, \widehat{r_j})$, $\mathcal{S}$:
   a) decrypts $\overline{d_j} \mapsto d_j$;
   b) samples a non-zero random plaintext value $\rho_j \in \mathbb{Z}_t \backslash \{0\}$ from a uniform distribution;
   c) homomorphically computes
   $$\widehat{e_j} = (\widehat{r_j} - d_j) \cdot \rho_j$$
   d) sends $\widehat{e_j}$ to $\mathcal{R}$.
5) **Result**: $\mathcal{R}$ decrypts $\widehat{e_j} \mapsto e_j$ and outputs
   $$X \cap Y = \cup_{j=1}^{|Y|} \{y_j : e_j = 0\}$$

Assuming a Sender's set with one million entries

Receiver | **Sender**

One-time

encrypts set with Sender's key

sends encrypted set ≈ **12.5 Megabytes** ←

Recurrent

computes intersection — **Constant-time search (3 multiplications)**

adds random value

encrypts random value with Receiver's key

sends encrypted result and encrypted random value →

decrypts result

subtracts random value

multiply by another random value

← sends final result

decrypts final result

checks if value is zero or not

Results

$|X| \approx 2^{20}, |Y| = \{4, 16, 64\}$

			$\|X\| = \mathscr{L} \cdot 2^{20}$
One-time	Computation	Fast Setup	0.09
		Fast Intersection	0.30
	Communication	Fast Setup	12.63
		Fast Intersection	25.25
	Total time (10 Gbps)	Fast Setup	**0.10**
		Fast Intersection	0.32
	Total time (100 Mbps)	Fast Setup	1.26
		Fast Intersection	2.48

			$\|Y\|$ = 4	16	64
Recurrent	Computation	Fast Setup	0.02	0.11	0.43
		Fast Intersection	0.03	0.10	0.37
	Communication	Fast Setup	1.85	7.39	29.55
		Fast Intersection	0.66	2.65	10.59
	Total time (10 Gbps)	Fast Setup	**0.02**	0.12	0.45
		Fast Intersection	0.03	**0.10**	**0.38**
	Total time (100 Mbps)	Fast Setup	0.33	0.86	2.95
		Fast Intersection	**0.24**	**0.47**	**1.38**

$|X| \approx 2^{20}, |Y| = \{4, 16, 64\}$

			$\lvert X\rvert = \mathscr{L} \cdot 2^{20}$
One-time	Computation	Fast Setup	0.09
		Fast Intersection	0.30
	Communication	Fast Setup	12.63
		Fast Intersection	25.25
	Total time (10 Gbps)	Fast Setup	**0.10**
		Fast Intersection	0.32
	Total time (100 Mbps)	Fast Setup	1.26
		Fast Intersection	2.48

			$\lvert Y\rvert$ = 4	16	64
Recurrent	Computation	Fast Setup	0.02	0.11	0.43
		Fast Intersection	0.03	0.10	0.37
	Communication	Fast Setup	1.85	7.39	29.55
		Fast Intersection	0.66	2.65	10.59
	Total time (10 Gbps)	Fast Setup	**0.02**	0.12	0.45
		Fast Intersection	0.03	**0.10**	**0.38**
	Total time (100 Mbps)	Fast Setup	0.33	0.86	2.95
		Fast Intersection	**0.24**	**0.47**	**1.38**

$|X| \approx 2^{20}, |Y| = \{4, 16, 64\}$

			$\lvert X\rvert = \mathscr{L} \cdot 2^{20}$
One-time	Computation	Fast Setup	0.09
		Fast Intersection	0.30
	Communication	Fast Setup	12.63
		Fast Intersection	25.25
	Total time (10 Gbps)	Fast Setup	**0.10**
		Fast Intersection	0.32
	Total time (100 Mbps)	Fast Setup	1.26
		Fast Intersection	2.48

			$\lvert Y\rvert$ = 4	16	64
Recurrent	Computation	Fast Setup	0.02	0.11	0.43
		Fast Intersection	0.03	0.10	0.37
	Communication	Fast Setup	1.85	7.39	29.55
		Fast Intersection	0.66	2.65	10.59
	Total time (10 Gbps)	Fast Setup	**0.02**	0.12	0.45
		Fast Intersection	0.03	**0.10**	**0.38**
	Total time (100 Mbps)	Fast Setup	0.33	0.86	2.95
		Fast Intersection	**0.24**	**0.47**	**1.38**

$|X| \approx 2^{20}, |Y| = \{4, 16, 64\}$

			$\lvert X\rvert = \mathscr{L} \cdot 2^{20}$
One-time	Computation	Fast Setup	0.09
		Fast Intersection	0.30
		CLR [8]	1.21
		KKRT [31]	NA
		PSSZ [38]	NA
	Communication	Fast Setup	12.63
		Fast Intersection	25.25
		CLR [8]	0.00
		KKRT [31]	NA
		PSSZ [38]	NA
	Total time (10 Gbps)	Fast Setup	**0.10**
		Fast Intersection	0.32
		CLR [8]	1.21
		KKRT [31]	NA
		PSSZ [38]	NA
	Total time (100 Mbps)	Fast Setup	1.26
		Fast Intersection	2.48
		CLR [8]	**1.21**
		KKRT [31]	NA
		PSSZ [38]	NA

			$\lvert Y\rvert$ = 4	16	64
Recurrent	Computation	Fast Setup	0.02	0.11	0.43
		Fast Intersection	0.03	0.10	0.37
	Communication	Fast Setup	1.85	7.39	29.55
		Fast Intersection	0.66	2.65	10.59
	Total time (10 Gbps)	Fast Setup	**0.02**	0.12	0.45
		Fast Intersection	0.03	**0.10**	**0.38**
	Total time (100 Mbps)	Fast Setup	0.33	0.86	2.95
		Fast Intersection	**0.24**	**0.47**	**1.38**

[8] Hao Chen, Kim Laine, and Peter Rindal. "Fast private set intersection from homomorphic encryption." *CCS*, 2017.

[31] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. "Efficient batched oblivious PRF with applications to private set intersection." CCS, 2016.

[38] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. "Phasing: Private set intersection using permutation-based hashing." *USENIX Security, 2015.*

$|X| \approx 2^{20}, |Y| = \{4, 16, 64\}$

			$\|X\| = \mathscr{L} \cdot 2^{20}$
One-time	Computation	Fast Setup	0.09
		Fast Intersection	0.30
		CLR [8]	1.21
		KKRT [31]	NA
		PSSZ [38]	NA
	Communication	Fast Setup	12.63
		Fast Intersection	25.25
		CLR [8]	0.00
		KKRT [31]	NA
		PSSZ [38]	NA
	Total time (10 Gbps)	Fast Setup	**0.10**
		Fast Intersection	0.32
		CLR [8]	1.21
		KKRT [31]	NA
		PSSZ [38]	NA
	Total time (100 Mbps)	Fast Setup	1.26
		Fast Intersection	2.48
		CLR [8]	**1.21**
		KKRT [31]	NA
		PSSZ [38]	NA

			$\|Y\|$ = 4	16	64
Recurrent	Computation	Fast Setup	0.02	0.11	0.43
		Fast Intersection	0.03	0.10	0.37
		CLR [8]	2.21	2.21	2.21
		KKRT[31]	1.62	1.62	1.62
		PSSZ [38]	0.86	0.86	0.71
	Communication	Fast Setup	1.85	7.39	29.55
		Fast Intersection	0.66	2.65	10.59
		CLR [8]	5.60	5.60	5.60
		KKRT [31]	57.17	57.17	57.17
		PSSZ [38]	24.05	24.05	27.11
	Total time (10 Gbps)	Fast Setup	**0.02**	0.12	0.45
		Fast Intersection	0.03	**0.10**	**0.38**
		CLR [8]	2.21	2.21	2.21
		KKRT [31]	1.67	1.67	1.67
		PSSZ [38]	0.88	0.88	0.73
	Total time (100 Mbps)	Fast Setup	0.33	0.86	2.95
		Fast Intersection	**0.24**	**0.47**	**1.38**
		CLR [8]	2.81	2.81	2.81
		KKRT [31]	6.46	6.46	6.46
		PSSZ [38]	2.94	2.94	3.05

[8] Hao Chen, Kim Laine, and Peter Rindal. "Fast private set intersection from homomorphic encryption." *CCS*, 2017.

[31] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. "Efficient batched oblivious PRF with applications to private set intersection." CCS, 2016.

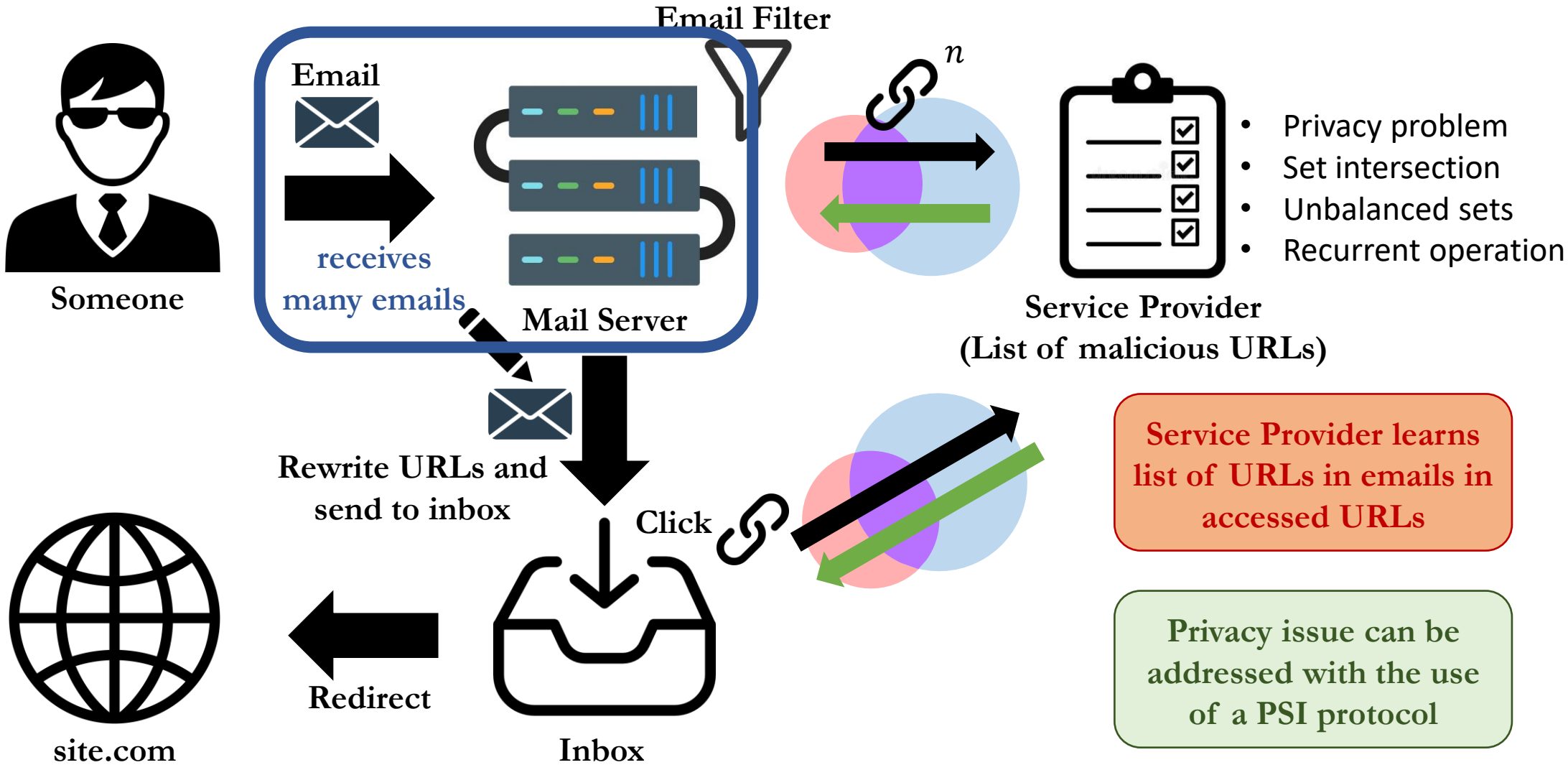
[38] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. "Phasing: Private set intersection using permutation-based hashing." *USENIX Security, 2015.*

Thank You

GitHub

github.com/momalab/psi-ndss2025

URL Denylisting / Email Filtering



Full Protocol

Protocol 2 Full Protocol

Input: $\mathcal{S}$ inputs set X of size $|X|$. $\mathcal{R}$ inputs set Y of size $|Y|$. Both sets consist of bit strings of length σ_d. $|X|$, $|Y|$, and σ_d are public.

Output: $\mathcal{S}$ outputs $\perp$; $\mathcal{R}$ outputs $X \cap Y$.

1) **Setup**: In this step, parties select all parameters to be used in the protocol.
   a) **Security parameters**: Parties agree on computational and statistical security parameters (κ and λ, respectively).
   b) **Data compression**: From Eqs. 4 and 5 (§IV-A5), we have that $\sigma = \min(\sigma_d, \sigma_h)$. Therefore, if $\sigma_h < \sigma_d$, parties agree on a hash function $\mathcal{Z}(\cdot) : \mathbb{Z}_{2^{\sigma_d}} \mapsto \mathbb{Z}_{2^{\sigma_h}}$ for data compression. Otherwise, $\mathcal{Z}(x) = x$.
   c) **Cuckoo hashing and Permutation-based hashing**:
      i) Parties agree on the number of hash functions h and insertion depth limit d for Cuckoo hashing.
      ii) They select the number of Cuckoo hash tables k ($\equiv$ number of packing slots), and a function $g(\cdot)$ that maps a σ-bit entry x to Cuckoo hash table $T^{(g(x))}$ (§IV-B5).
      iii) Following §IV-A3 and §VI-A, $\mathcal{S}$ chooses the Cuckoo hash table size $|T^{(i)}|$ for a failure rate $\leq 2^{-\lambda}$.
      iv) By knowing $|T^{(i)}|$, one can calculate the left $\sigma_L = \lfloor \log_2 |T^{(i)}| \rfloor$ and right $\sigma_R = \sigma - \sigma_L$ sizes for Permutation-based hashing (§IV-A4).
      v) $\mathcal{S}$ selects the h different hash functions $H : \mathbb{Z}_{2^\sigma} \mapsto \mathbb{Z}_{|T^{(i)}|}$, where $H_j(z_L, z_R) = z_L \oplus f_j(z_R)$ and $f_j(z_R) : \mathbb{Z}_{2^{\sigma_R}} \mapsto \mathbb{Z}_{|T^{(i)}|}$ (§IV-A4).
   d) **Encryption parameters**:
      i) Parties agree on partitioning parameters η_s and η_r (§IV-C1) and plaintext modulus $t = \prod_{i=1}^{k} t_i : t_i \geq 2^{\sigma_R} + 2$ (§IV-B4 and §IV-B7).
      ii) Parties select encryption parameters $(n_s, q_s, t), (n_r, q_r, t) : n_s \geq n_r$ following Eq. 3 and §IV-B7. Encryption parameters are public.
      iii) Each party generates decryption and encryption keys, which are kept private.
2) **Cuckoo hashing**: For each entry $x \in X$, $\mathcal{S}$ performs $z = z_L | z_R = \mathcal{Z}(x) : |z_L| = \lfloor \log_2 |T^{(i)}| \rfloor$ and inserts z_R at $T^{(g(z))}_{H_j(z_L, z_R)} : 1 \leq j \leq h$. A dummy value is assigned to empty bins after inserting entries $\in X$ into tables $T^{(i)}$ (§IV-A2).
3) **Set encryption**: $\mathcal{S}$ encodes all Cuckoo hash tables and bins into polynomials following Eq. 8 (§IV-B5), encrypts them into ciphertexts under its own key, and sends the $\lceil \frac{|T^{(i)}|}{n} \rceil$ ciphertexts $\overline{C}$ to $\mathcal{R}$.
4) **Computing intersection**: For each $y \in Y$, $\mathcal{R}$:
   a) Computes $z = z_L | z_R = \mathcal{Z}(y) : |z_L| = \lfloor \log_2 |T^{(i)}| \rfloor$, and for each Cuckoo hash function $H_j(\cdot)$ $\forall j \in \mathbb{Z} : 1 \leq j \leq h$, $\mathcal{R}$:
      i) Creates a polynomial $P \in Z_t[x]/(x^{n_s} + 1)$ s.t. $P_{u,v} = z_R$ for $(u, v) = (H_j(z_L, z_R) \bmod n_s,\ g(z) - 1)$ and a dummy value otherwise.
      ii) Homomorphically computes the difference $\overline{S}^{(j)} = \overline{C}^{\left(\lfloor \frac{H_j(z_L, z_R)}{n_s} \rfloor\right)} - P$.
   b) Performs $h - \eta_s - 1 : \eta_s \in \mathbb{Z}_h$ homomorphic multiplications between $\overline{S}^{(j)} : j \in \mathbb{Z}_{h+1} \backslash \{0\}$, resulting in $\eta_s + 1$ ciphertexts $\overline{M}^{(i)}$ $\forall i \in \mathbb{Z}_{\eta_s+1}$.
   c) Samples $\eta_s + 1$ tuples of random values $(r_1, ..., r_{n_s} \in \mathbb{Z}_t)$ from a uniform distribution, where each tuple is encoded as a polynomial $R^{(i)} \in \mathbb{Z}_t[x]/(x^{n_s} + 1)$.
   d) Computes $\overline{D}^{(i)} = \overline{M}^{(i)} + R^{(i)}$ $\forall i \in \mathbb{Z}_{\eta_s+1}$, modulus switches it, encrypts $R^{(i)} \mapsto \widehat{R}^{(i)}$ under its key, and sends the $(\overline{D}^{(i)}, \widehat{R}^{(i)})$ pairs to $\mathcal{S}$. Note that each $\widehat{R}^{(i)}$ will be several ciphertexts if $n_s > n_r$.
5) **Decryption**: For all pairs $(\overline{D}^{(i)}, \widehat{R}^{(i)})$ $\forall i \in \mathbb{Z}_{\eta_s+1}$, $\mathcal{S}$:
   a) Decrypts each $\overline{D}^{(i)} \mapsto D^{(i)}$, and homomorphically computes $\widehat{M}^{(i)} = \widehat{R}^{(i)} - D^{(i)}$ $\forall i \in \mathbb{Z}_{\eta_s+1}$.
   b) Performs $h - \eta_s - 1 - \eta_r : \eta_r \in \mathbb{Z}_{\eta_s+1}$ homomorphic multiplications between $\widehat{M}^{(i)} : i \in \mathbb{Z}_{\eta_s+1}$, resulting in $\eta_r + 1$ ciphertexts $\widehat{N}^{(j)}$ $\forall j \in \mathbb{Z}_{\eta_r+1}$.
   c) Samples $\eta_r + 1$ tuples of random values $(\rho_1, ..., \rho_{n_r} \in \mathbb{Z}_t \backslash \{\alpha \cdot t_i\}$ $\forall \alpha, i \in \mathbb{Z} : 1 \leq i \leq k)$ such that no random value is a multiple of any packing modulus t_i, and encode each tuple as a polynomial $P^{(j)} \in \mathbb{Z}_t/(x^{n_r} + 1)$.
   d) Homomorphically computes $\widehat{E}^{(j)} = rotate(\widehat{N}^{(j)} \cdot P^{(j)}, \omega_j)$ $\forall j \in \mathbb{Z}_{\eta_r+1}$, where the function $rotate(\cdot)$, rotates the batching slots by a random number of positions $\omega_j \in \mathbb{Z}_{n_r}$, modulus switches and sends each $\widehat{E}^{(j)}$ to $\mathcal{R}$.
6) **Result**: $\mathcal{R}$ decrypts each $\widehat{E}^{(j)} \mapsto E^{(j)}$ and outputs $X \cap Y = \cup_{j=1}^{|Y|} \{y_j \iff 0 \in \cup_{j=0}^{\eta_r} E^{(j)}\}$.

$|X| \approx 2^{24}, |Y| = \{4, 16, 64\}$

			$\|X\| = \mathscr{L} \cdot 2^{24}$
One-time	Computation	Fast Setup	1.79
		Fast Intersection	7.74
		CLR [8]	21.54
		KKRT [31]	NA
		PSSZ [38]	NA
	Communication	Fast Setup	202.04
		Fast Intersection	404.08
		CLR [8]	0.00
		KKRT [31]	NA
		PSSZ [38]	NA
	Total time (10 Gbps)	Fast Setup	1.96
		Fast Intersection	8.08
		CLR [8]	21.54
		KKRT [31]	NA
		PSSZ [38]	NA
	Total time (100 Mbps)	Fast Setup	18.68
		Fast Intersection	41.44
		CLR [8]	21.54
		KKRT [31]	NA
		PSSZ [38]	NA

			$\|Y\|$ = 4	16	64
Recurrent	Computation	Fast Setup	0.02	0.11	0.43
		Fast Intersection	0.03	0.10	0.37
		CLR [8]	23.22	23.22	23.22
		KKRT[31]	30.42	40.42	30.42
		PSSZ [38]	11.11	11.11	8.54
	Communication	Fast Setup	1.85	7.39	29.55
		Fast Intersection	0.66	2.65	10.59
		CLR [8]	11.00	11.00	11.00
		KKRT [31]	946.75	946.75	946.75
		PSSZ [38]	432.05	432.05	432.11
	Total time (10 Gbps)	Fast Setup	0.02	0.12	0.45
		Fast Intersection	0.03	0.10	0.38
		CLR [8]	23.22	23.22	23.22
		KKRT [31]	31.21	31.21	31.21
		PSSZ [38]	11.47	11.47	8.90
	Total time (100 Mbps)	Fast Setup	0.33	0.86	2.95
		Fast Intersection	0.24	0.47	1.38
		CLR [8]	23.59	23.59	23.59
		KKRT [31]	109.27	109.27	109.27
		PSSZ [38]	47.14	47.14	44.57

TABLE VI: Total time (in seconds) for two configurations of our PSI protocol, *Fast Setup* and *Fast Intersection*, and related work, considering a $\mathcal{S}$'s set size $|X| = \mathscr{L} \cdot 2^{24}$ and several sizes of $\mathcal{R}$'s set $|Y|$, where we evaluate the one-time costs and recurrent costs of performing set intersections. KKRT and PSSZ do not have one-time costs.