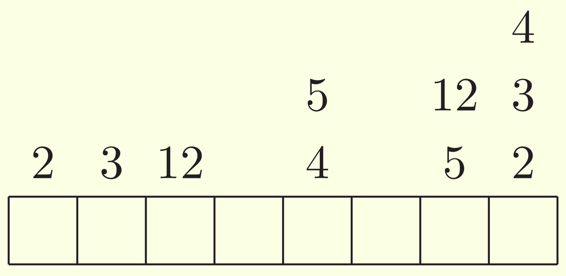
Jelle Vos
Jorrit van Assen
Tjitske Koster
Evangelia Anna Markatou
Zekeriya Erkin

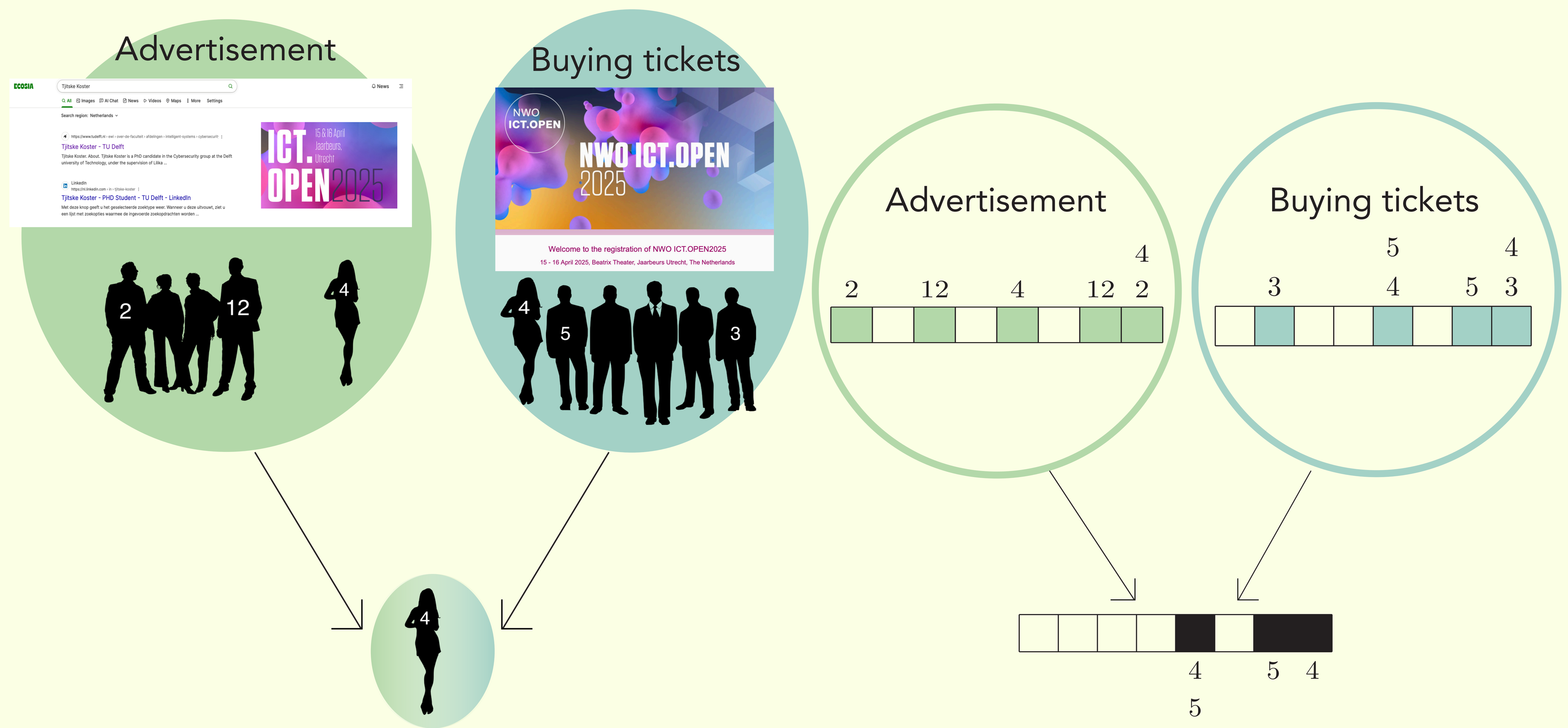
On the Insecurity of Bloom Filter-Based Private Set Intersections

Introduction

Bloom filter

Hash each element twice into the filter.





Why PSI?

We want to learn the intersection without revealing private data.

- Ads Conversion Measurement.
- Financial transactions.
- Comparison of no-fly lists.
- Private Contact Discovery.
- Password Breach Monitoring.

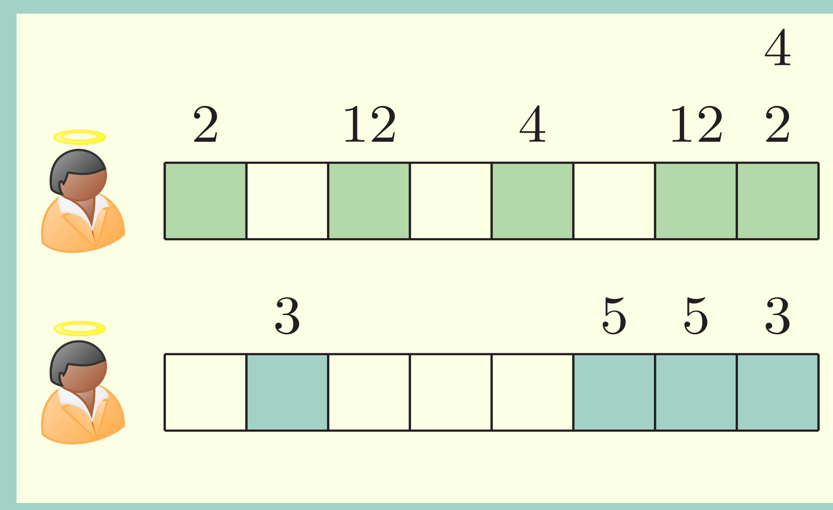
Why Bloom filter-based PSI?

- Bloom filters are **smal** compared to the input.
- Hash functions are **easy** to compute.
- The intersection is a **fast** logical AND operation.
- There might be false positives.
- No false negatives.

Flaws in existing proofs

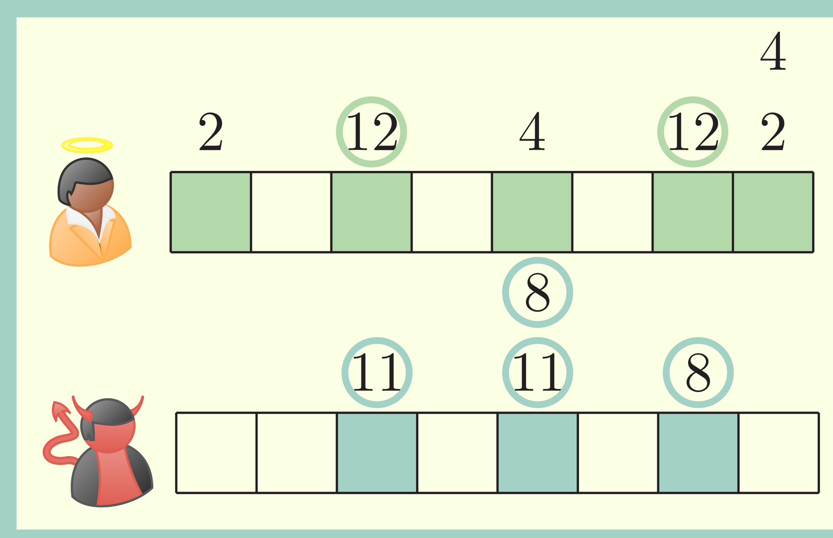
Honest Bloom filter intersection

We expect no false positives, because the false positive rate is low.



Input malicious Bloom filter intersection

Given a particular bloom filter we might evoke false positives.

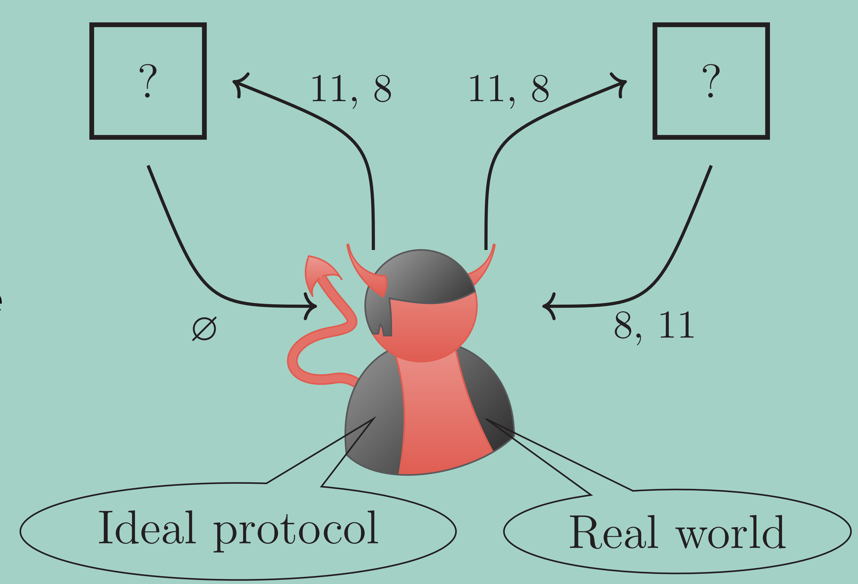


Distinghuisher real and ideal

Given a particular bloom filter we might evoke false postives.

In the **ideal functionality** the false postive rate is **low**. So, we expect no false positives.

In the **real world** we can **trigger** false positives, so we expect false positives.



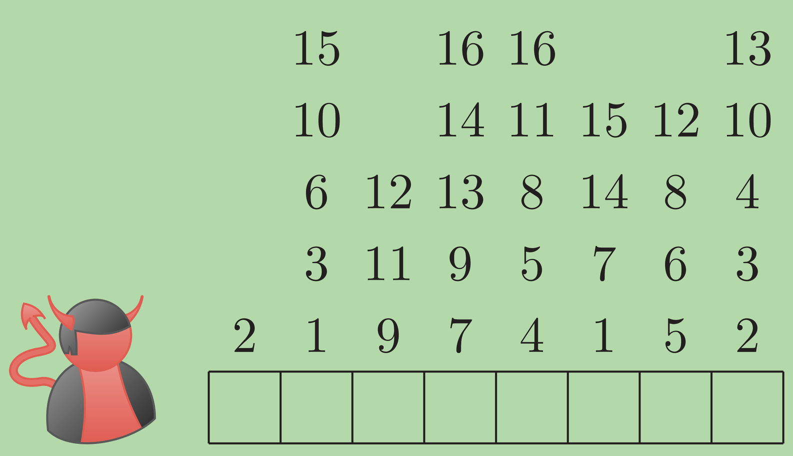
Practical attack

1. Guess the input set of the other

We expect that Ecosia showed our add to 16 people.

2. Determine the Bloom filter

The Hash functions are public



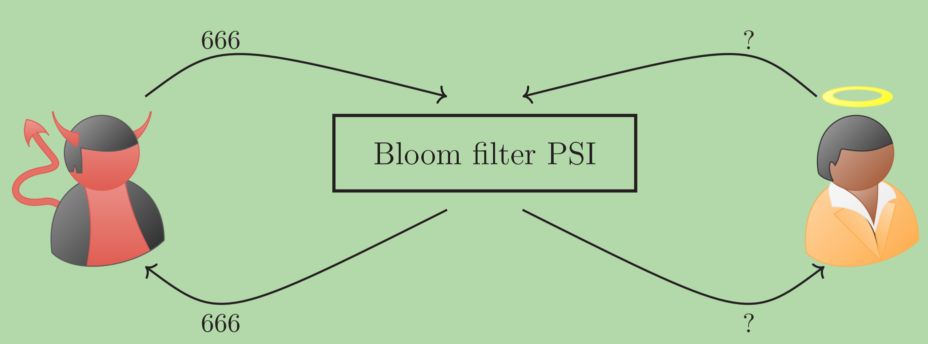
3. Find target element T

2 is an easy target, because it is alone in its bin

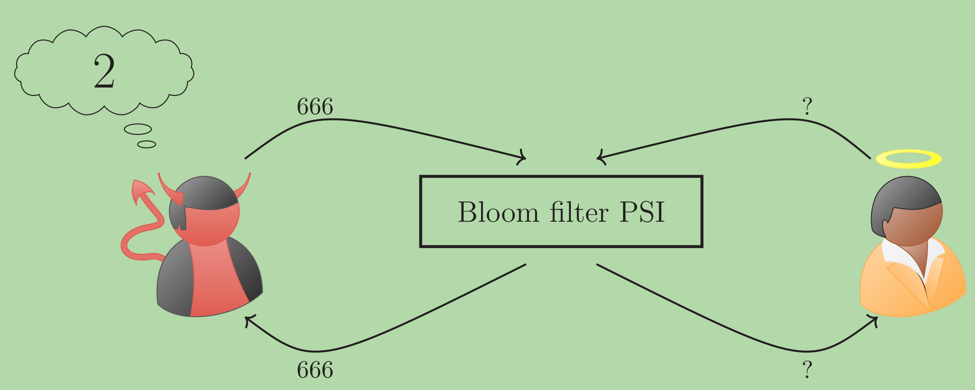
4. Find input revealing target T

666 reveals 2 if the honest party has at least one of 4,5,8,11,16.

5. Execute the protocol



6. Conclude from intersection



Attack in practice

Success of the attack

- The attack succeeds for previously used parameters.
- We limit the search to 2 hours.

Setting		Parameters		Success percentage		
	set size			Universe size		
false pos.	k	hash	filter size	2k	3k	4k
2^{-5}	256	5	1,852	100%	100%	100%
2^{-10}	256	10	3,702	100%	82%	61%
	4,096	10	59,102	100%	-	-
	65,536	10	945,493	100%	-	-
2^{-20}	256	20	7,403	15%	1%	0.2%
	4,096	20	118,202	98%	-	-
	65,536	20	1,890,985	68%	-	-
2^{-30}	256	30	11,103	3%	2^{-14}%	-
	4,096	30	177,302	2%	-	-
	65,536	30	2,836,477	2^{-10}%	-	-

Proposed mitigations

How can we prevent this attack?

- Using larger parameters
- Using OPRFs
- Using PBKDFs

- Protocol takes longer
- More communication

Time in seconds

Setting		State of the art			Mitigation 1: Large Bloom filters			Mitigation 2: OPRFs			Mitigation 3: PBKDFs		
Parties	Set size	Time	Rounds	Comm.	Time	Rounds	Comm.	Time	Rounds	Comm.	Time	Rounds	Comm.
2	256	0.25	5	0.27 MB	3.26	5	3.66 MB	0.31	7	0.30 MB	21.63	5	0.27 MB
	4096	3.87	5	4.36 MB	51.38	5	58.46 MB	4.89	7	4.86 MB	345.58	5	4.36 MB
	65536	60.74	5	69.71 MB	815.37	5	935.33 MB	78.21	7	77.71 MB	> 1h	5	69.71 MB
3	256	0.38	5	0.55 MB	5.29	5	7.32 MB	0.53	7	0.64 MB	21.91	5	0.55 MB
	4096	6.06	5	8.71 MB	82.59	5	116.93 MB	8.44	7	10.21 MB	351.11	5	8.71 MB
	65536	97.37	5	139.42 MB	1338.55	5	1.83 GB	138.80	7	163.42 MB	> 1h	5	139.42 MB

Key takeaways

Theoretical attack

- Previous proofs were flawed.
- Bloom filters cannot be used to approximate the intersection.

Practical attack

- Security parameters must be order of magnitudes larger;
- Or mitigations should be used.

Mitigations

- Either take longer;
- Or more communication.