Anonymous Router Network

Honors Oral Defense

Committee: Professor Ke Wu
Professor Paul Grubbs

by
Anthony Flath
April 3rd, 2026

Roadmap

- Preliminaries and Overview
- Security Definitions
- Relevant Schemes/Algorithms
- Our Construction
- Results and Benchmarks
- Conclusion

Roadmap

- **Preliminaries and Overview**
- Security Definitions
- Relevant Schemes/Algorithms
- Our Construction
- Results and Benchmarks
- Conclusion

Anonymous Communication

- Popular forms include threshold security, namely Tor
- Not necessarily bad
- Drawbacks
  - Actors with resources can gain significant influence
  - Can often be slow

Non-Interactive Anonymous Router

- “NIAR” by Shi and Wu
- A single untrusted router with similar results
- Key takeaway: oblivious shuffling
- Theoretical, not implemented

(somewhat) Practical Anonymous Router

- “sPAR” by Das and Park
- Similar idea to NIAR, and showed implementation
- However, relatively slow
  - Needs at least 4 seconds for 32 messages transmitted
  - Speed scales with respect to the amount of messages

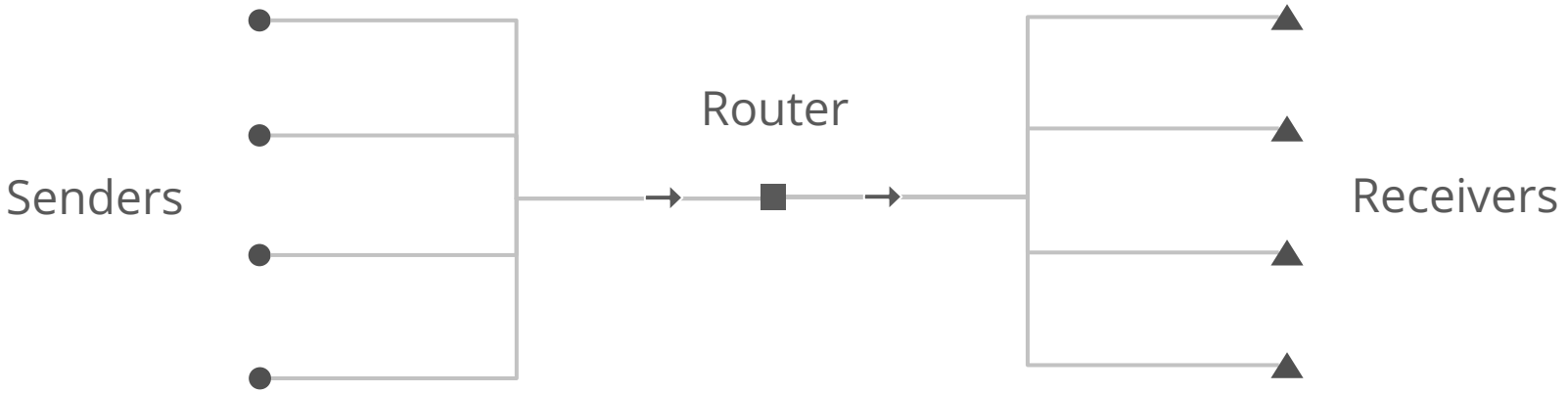
Motivation for our work

- Is it possible to achieve a fast implementation?
- Can we limit scaling performance with respect to number of messages?
- Is it possible to eliminate reliance on a single machine?

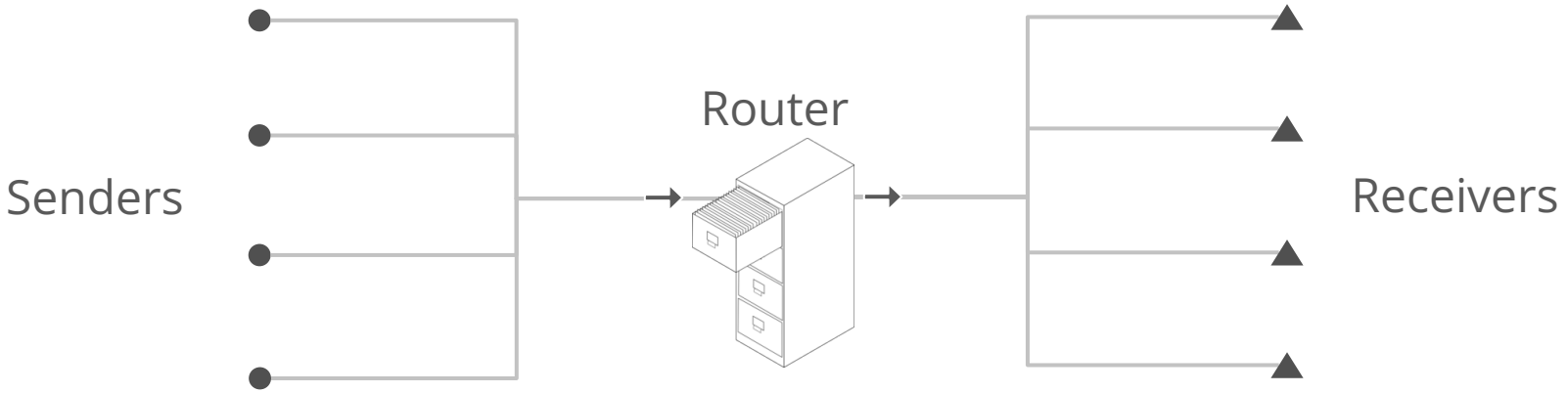
Our results

- Adding more routers to the system improves performance.
- A subset of messages still get sent when some routers fail.
- Opens the door for future performance improvements.

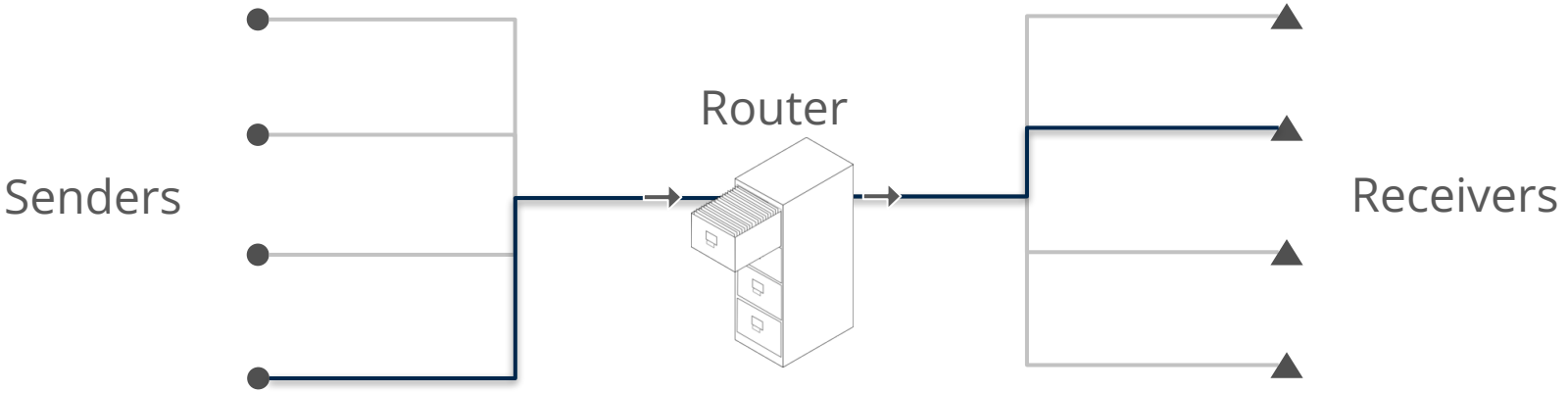
Informal overview of the system



Informal overview of the system

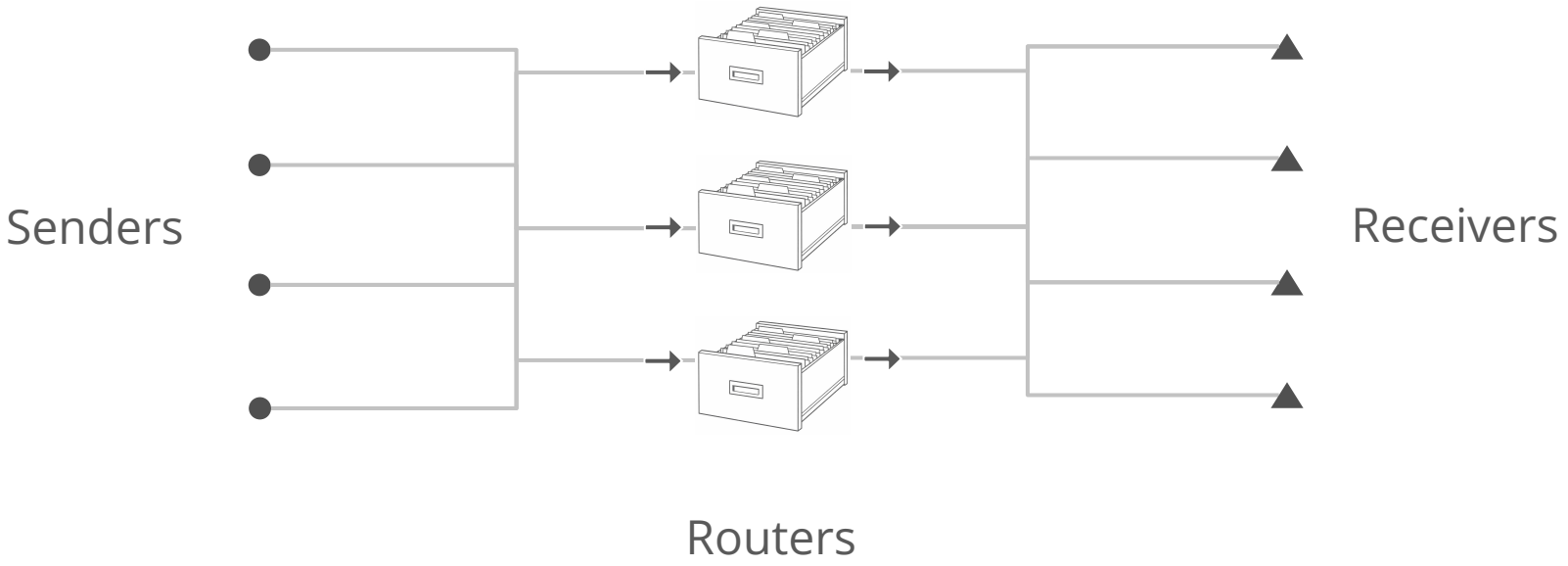


Informal overview of the system



Big filing cabinets take a while to organize!

Informal overview of the system



Less informal overview

n	number of senders/receivers
k	number of routers
p	bin multiplier
π	message destination
C	ciphertext array
$\mathcal{M}$	message multiset

Roadmap

- Preliminaries and Overview
- **Security Definitions**
- Relevant Schemes/Algorithms
- Our Construction
- Results and Benchmarks
- Conclusion

The two notions of security

The ARN experiment

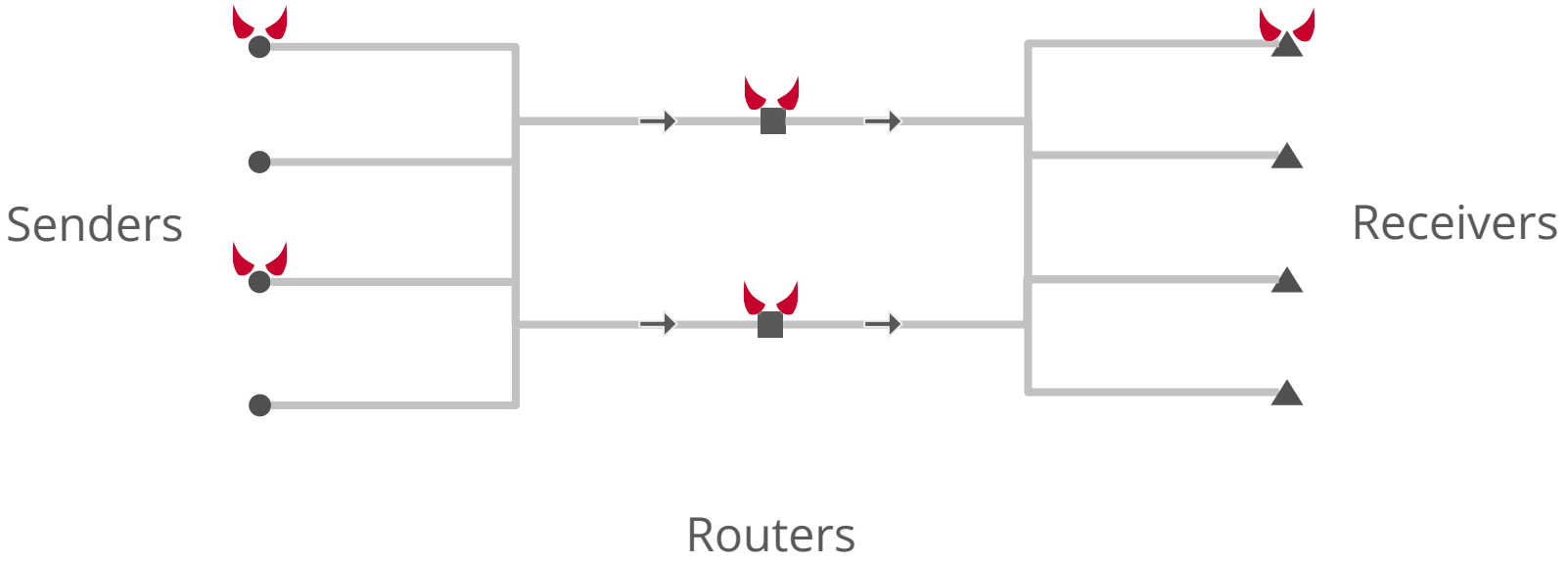
n	number of senders/receivers
k	number of routers
p	bin multiplier
π	message destination
C	ciphertext array
$\mathcal{M}$	message multiset

The ARN experiment correctness

Sender Anonymity

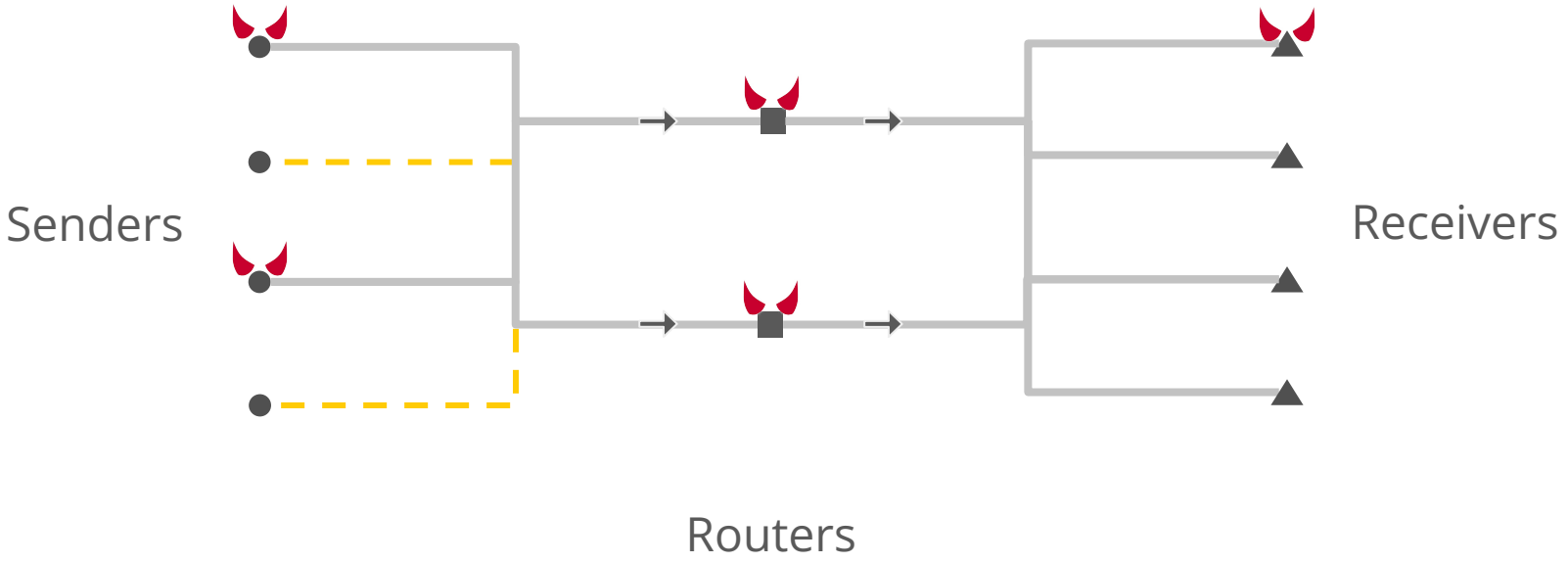
- An adversary **can** know:
  - The destination and message from every corrupt sender
  - The content of every message
- An adversary **cannot** know:
  - The sender of a specific message
  - The destination given by a sender

Sender Anonymity example



Sender Anonymity example

———	Known by adversary
- - -	Not known by adversary

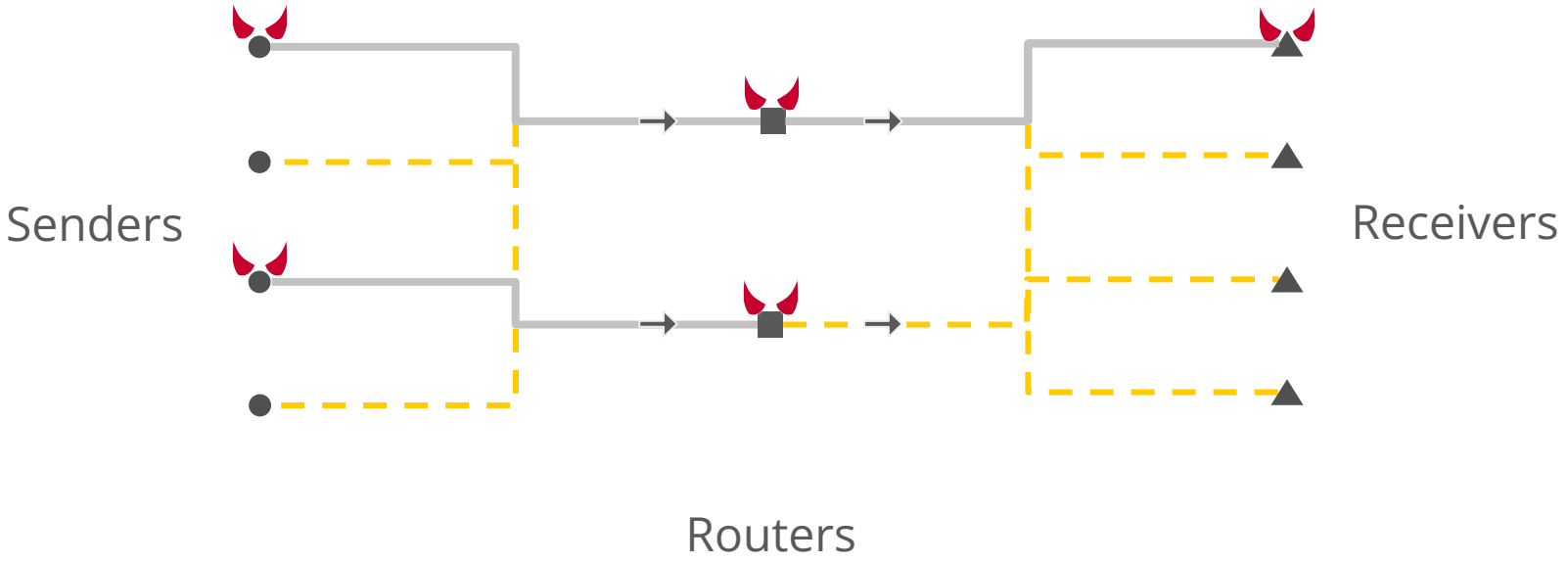


Receiver Anonymity

- An adversary can know:
  - All corrupt-to-corrupt communication
  - The content of every message sent to a corrupt receiver
- An adversary cannot know:
  - Everything else regarding the content and mapping of messages

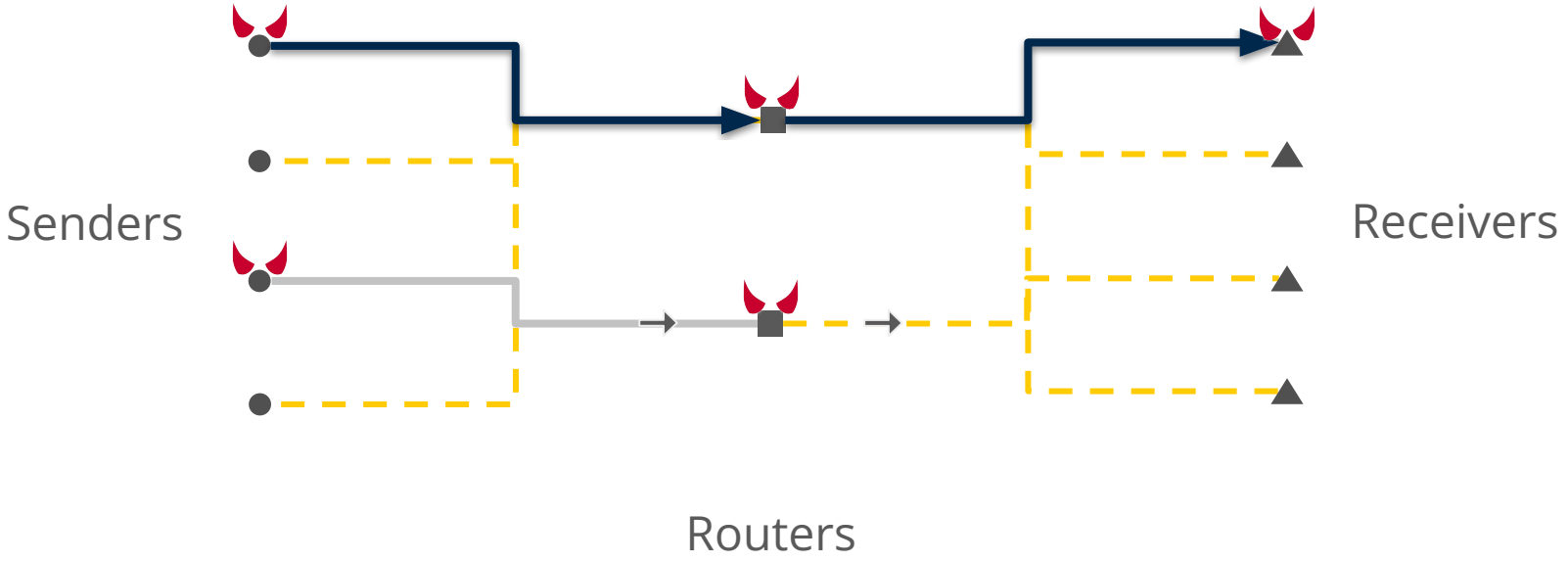
Receiver Anonymity example

———	Known by adversary
- - -	Not known by adversary



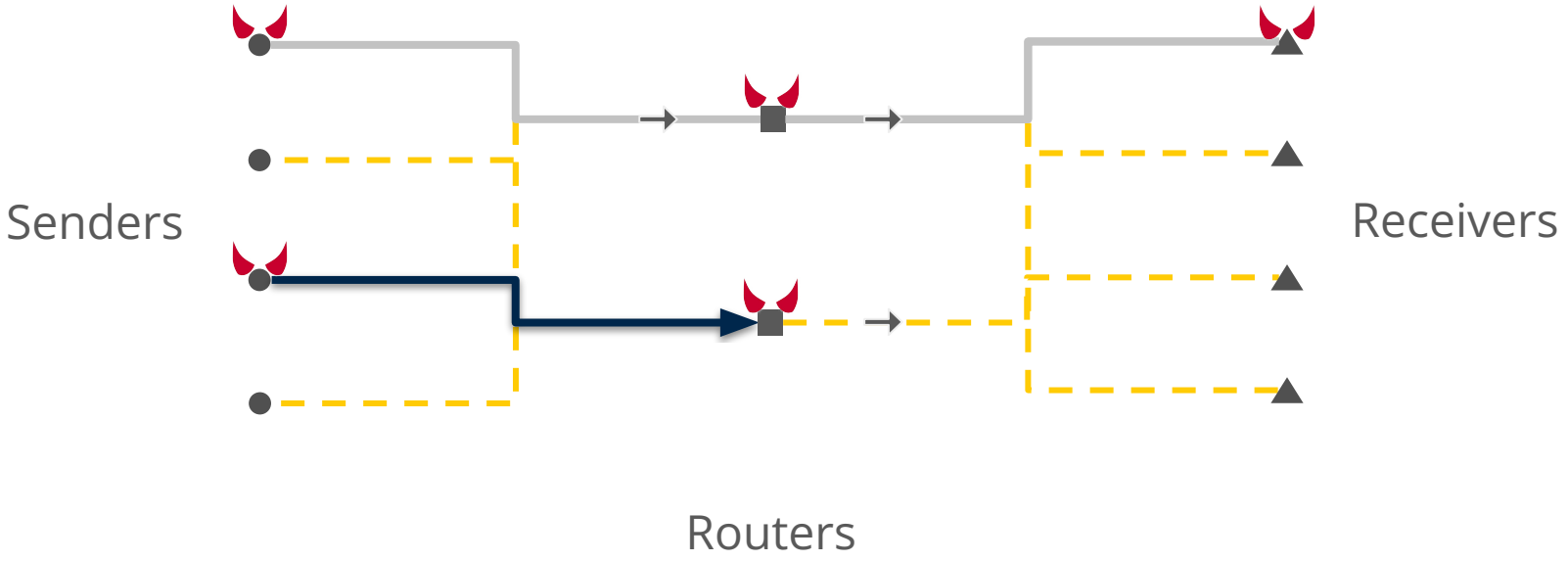
Receiver Anonymity example

———	Known by adversary
- - -	Not known by adversary



Receiver Anonymity example

———	Known by adversary
- - -	Not known by adversary



Roadmap

- Preliminaries and Overview
- Security Definitions
- **Relevant Schemes/Algorithms**
- Our Construction
- Results and Benchmarks
- Conclusion

Fully homomorphic encryption (FHE)

- Operate (addition and multiplication) on ciphertexts
- This is the basis of the oblivious placing mechanism

$$m_1 + m_2 = \mathrm{Dec}(\mathrm{Enc}(m_1) + \mathrm{Enc}(m_2))$$

$$m_1 \cdot m_2 = \mathrm{Dec}(\mathrm{Enc}(m_1) \cdot \mathrm{Enc}(m_2))$$

Why this is useful

- Suppose we wanted to place

$$m_1 \cdot 1 = \mathrm{Dec}(\mathrm{Enc}(m_1) \cdot \mathrm{Enc}(1))$$

- Suppose we did not want to place

$$m_1 \cdot 0 = \mathrm{Dec}(\mathrm{Enc}(m_1) \cdot \mathrm{Enc}(0))$$

FHE with Ring Learning With Errors (RLWE)

- A known polynomial multiplied by a secret polynomial added with some small error is indistinguishable from a randomly selected polynomial.

polynomial **x** *secret* **+** *error* **≡** *random polynomial*

q	ciphertext modulus
$\mathcal{R}_q$	$\mathbb{Z}[x]/(x^N+1) \mod q,\ q, N \in \mathbb{N}$

For a secret $s \in \mathcal{R}_q$ with polynomials $a, u \in \mathcal{R}_q$ and small error e, $(a, a \cdot s + e)$ and (a, u) are computationally indistinguishable.

RLWE Ciphertext

q	ciphertext modulus
t	plaintext modulus $t \ll q$
$\mathcal{R}_q$	$\mathbb{Z}[x]/(x^N + 1) \mod q,\ q, N \in \mathbb{N}$
χ	some small error distribution

$$\Delta = q/t$$
$$m \in \mathcal{R}_t$$
$$a, s \in \mathcal{R}_q$$
$$e \leftarrow \chi$$

$$\mathsf{RLWE}_s(m) = (a, b) \in \mathcal{R}_q^2 \text{ such that } b = a \cdot s + \Delta \cdot m + e$$

$$\Delta m \approx b - a \cdot s$$

RLWE Addition

- RLWE + RLWE makes sense:

$$\mathsf{RLWE}_s(m_1) + \mathsf{RLWE}_s(m_2) = (a = a_1 + a_2,\ b = b_2 + b_2)$$

$$\begin{aligned}
\Delta \cdot (m_1 + m_2) &\approx b - s \cdot a \\
&= b_1 + b_2 - s(a_1 + a_2) \\
&= (a_1 \cdot s + \Delta \cdot m_1 + e_1) + (a_2 \cdot s + \Delta \cdot m_2 + e_2) - s(a_1 + a_2) \\
&= \Delta \cdot m_1 + e_1 + \Delta \cdot m_2 + e_2
\end{aligned}$$

- Multiplication is not so simple.

Ring Gentry-Sahai-Waters (RGSW)

- Vector of decomposed RLWE ciphertexts

$$\mathsf{RGSW}_s(m) = (\mathrm{Decompose}(\mathsf{RLWE}_s(-s \cdot m)),\ \mathrm{Decompose}(\mathsf{RLWE}_s(m)))$$

$$\gamma = \quad q \quad \frac{q}{\beta} \quad \frac{q}{\beta^2} \quad \frac{q}{\beta^3} \quad \cdots \quad \frac{q}{\beta^{\ell-1}} \quad 0$$

Zama. Accessed 3-30-2026. https://www.zama.org/post/tfhe-deep-dive-part-3

- This is a nuanced topic that is largely outside of our scope.

Why RGSW is important

- RGSW allows for multiplication.

$$m_1, m_2 \in \mathcal{R}_t$$
$$s \in \mathcal{R}_q$$

$$\mathsf{RLWE}_s(m_1) \cdot \mathsf{RGSW}_s(m_2) = \mathsf{RLWE}_s(m_1 \cdot m_2)$$

Multi-party FHE

Multi-party FHE

- We want a scheme in which a router cannot simply decrypt.

  - $(\widehat{\mathsf{pk}},\ \text{user secret keys}) \leftarrow \mathbf{Setup}(1^\lambda, n)$
  - $(c) \leftarrow \mathbf{Enc}(\widehat{\mathsf{pk}}, m)$
  - $(m) \leftarrow \mathbf{Dec}(\text{user secret keys},\ c)$

Multi-party FHE with RLWE - I

- Participants agree on some a.
- Each user i calculates $\mathsf{pk} = a \cdot s_i + e$.
- $\widehat{\mathsf{pk}} = \sum_{i=1}^{n} \mathsf{pk}$.

Multi-party FHE with RLWE - II

- $(a', b') \leftarrow \mathbf{Enc}(\widehat{\mathsf{pk}}, m)$:
  - Sample u from $\mathcal{R}_q$
  - $a' = u \cdot a + e$ and $b' = u \cdot \widehat{\mathsf{pk}} + \Delta m + e$
- $(\Delta m) \leftarrow \mathbf{Dec}(\text{user secret keys}, \ (a', b'))$:
  - Each user i calculates $d_i = a' \cdot s_i + e_i$
  - $\Delta m = b' - \sum_{i=1}^{n} d_i$

Homomorphic Placing

Understanding the placing problem

- Reminder: we want the router to be able to obliviously place ciphertexts.
- We could have RLWE encrypted messages, with RGSW encrypted bits.
- Have the user send the encrypted bits!

The placing mechanism

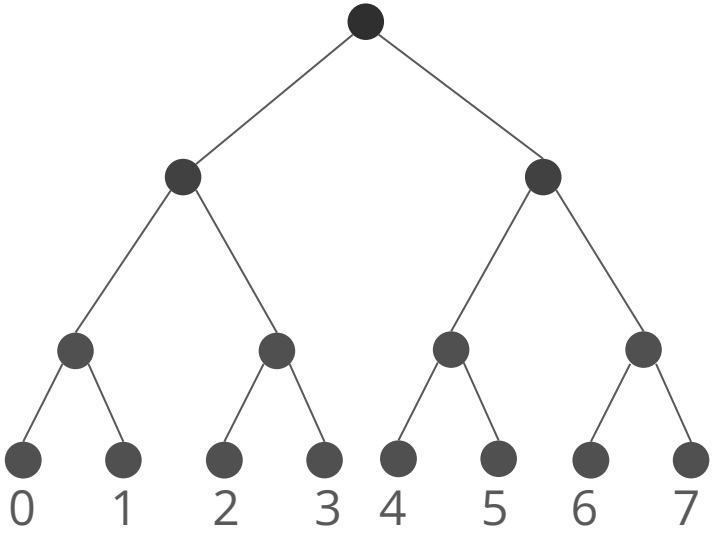
- SortingHat by Cong et al.
- Binary tree structure in which the the root is distributed to the leaves.
- Distribution is done according to the indices.

Quick detour: Plinko

- Plinko is from the game show “The Price is Right”

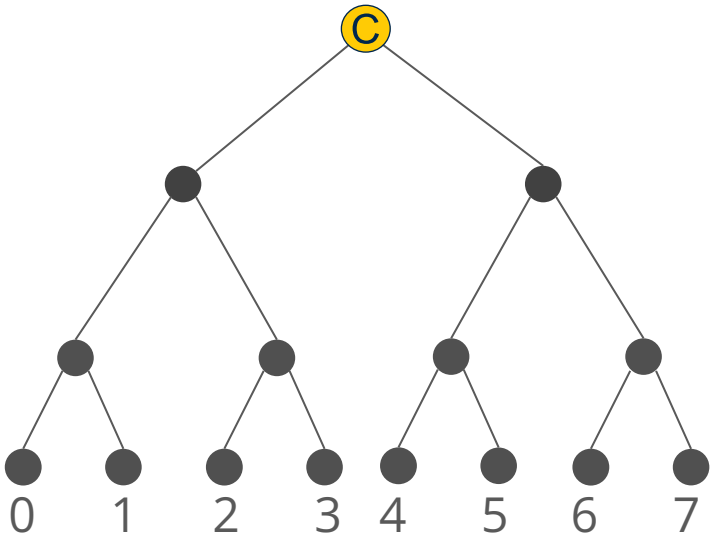
The placing mechanism through Plinko

Index: [1, 0, 1]



The placing mechanism through Plinko

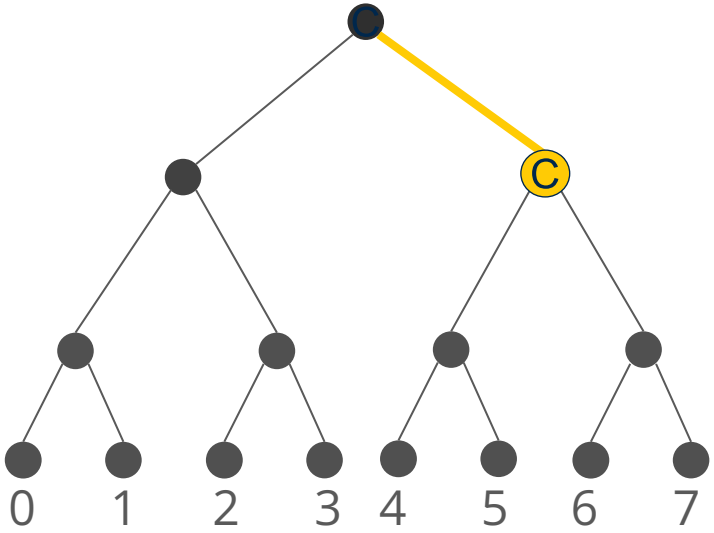
Index: [1, 0, 1]



1

The placing mechanism through Plinko

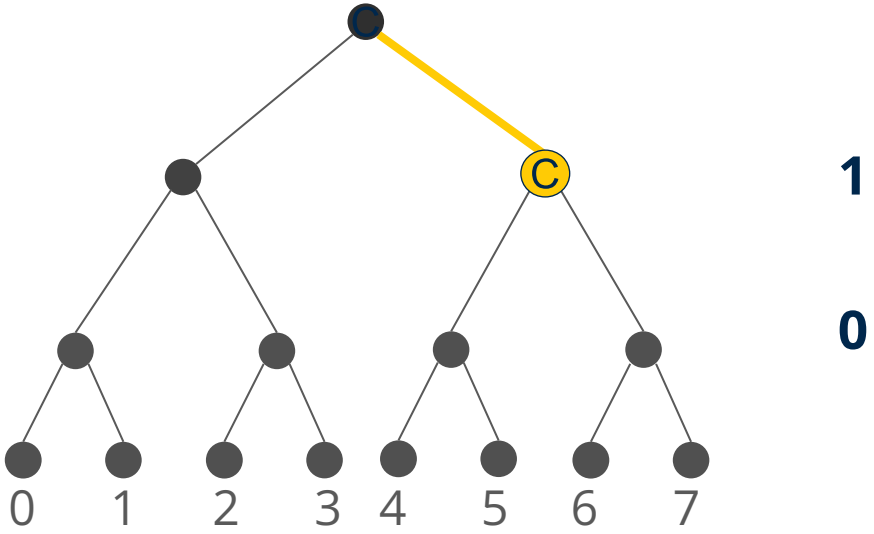
Index: [1, 0, 1]



1

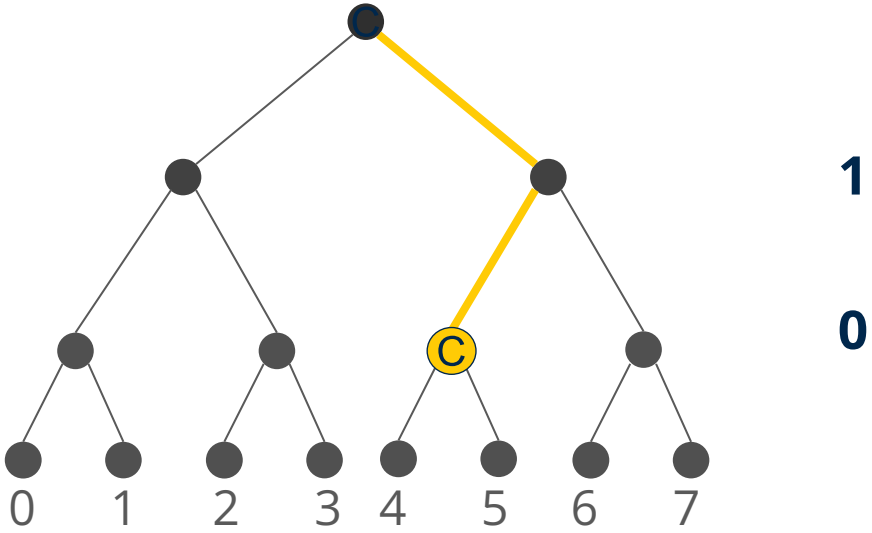
The placing mechanism through Plinko

Index: [1, 0, 1]



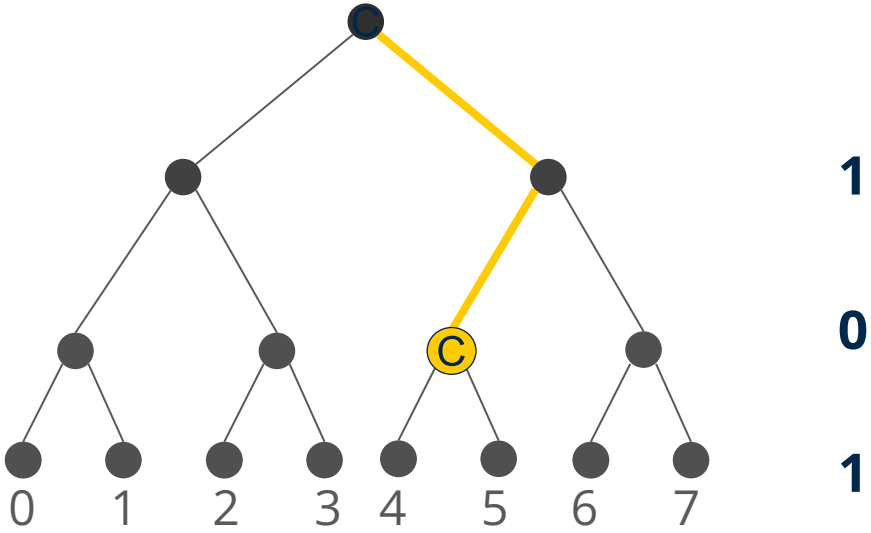
The placing mechanism through Plinko

Index: [1, 0, 1]



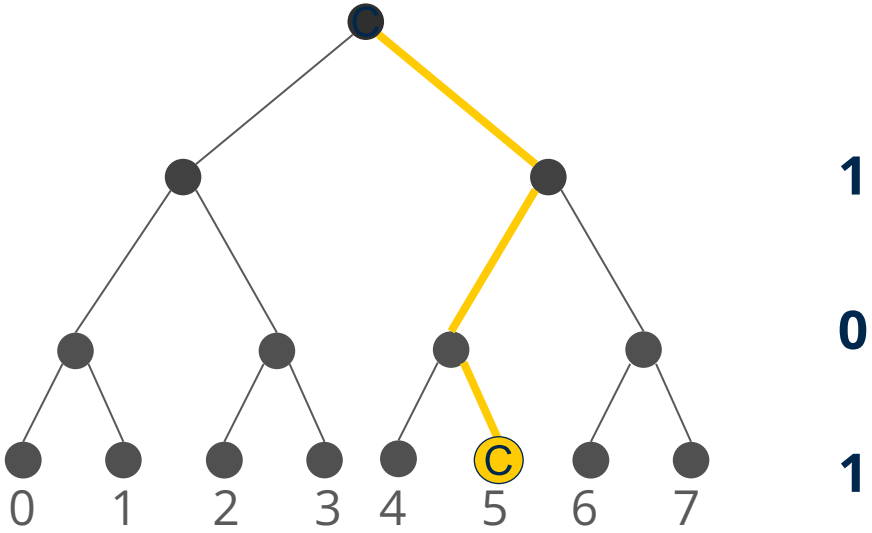
The placing mechanism through Plinko

Index: [1, 0, 1]



The placing mechanism through Plinko

Index: [1, 0, 1]



The actual placing algorithm

Algorithm 1 Homomorphic Placing

Input: $c \coloneqq \mathsf{RLWE}_{\widehat{\mathsf{pk}}}(m)$, $A = \{\mathsf{RGSW}_{\widehat{\mathsf{pk}}}(m_w)\}_{w \in \{0,\ldots,\log(p\cdot n)-1\}}$

Output: RLWE ciphertext array with c placed according to A

function HomPlacing(c, A^*)
  $b_0 \coloneqq c$
  $b_x \coloneqq \mathsf{RLWE}_{\widehat{\mathsf{pk}}}(0)$ for $x \in \{1, \ldots, 2p \cdot n - 2\}$
  for $w \leftarrow \{0, \ldots, \log(p \cdot n) - 1\}$ **do** ▷ Iterate through each layer
    for $z \leftarrow \{0, \ldots, 2^w - 1\}$ **do** ▷ Iterate through each leaf
      $b_{2^{w+1}+2\cdot z-1} \coloneqq b_{2^w+z-1} - (b_{2^w+z} \cdot A^*_w)$
      $b_{2^{w+1}+2\cdot z+1} \coloneqq (b_{2^w+z-1} \cdot A^*_w)$
    end for
  end for
  return $b_{p\cdot n}, \ldots b_{2p\cdot n-1}$
end function

What about the rest of the messages?

- We add the output of all placing algorithm executions.
- This results in an failure rate dependent on the value of the bin multiplier.
- We'll analyze this in the results section.

Roadmap

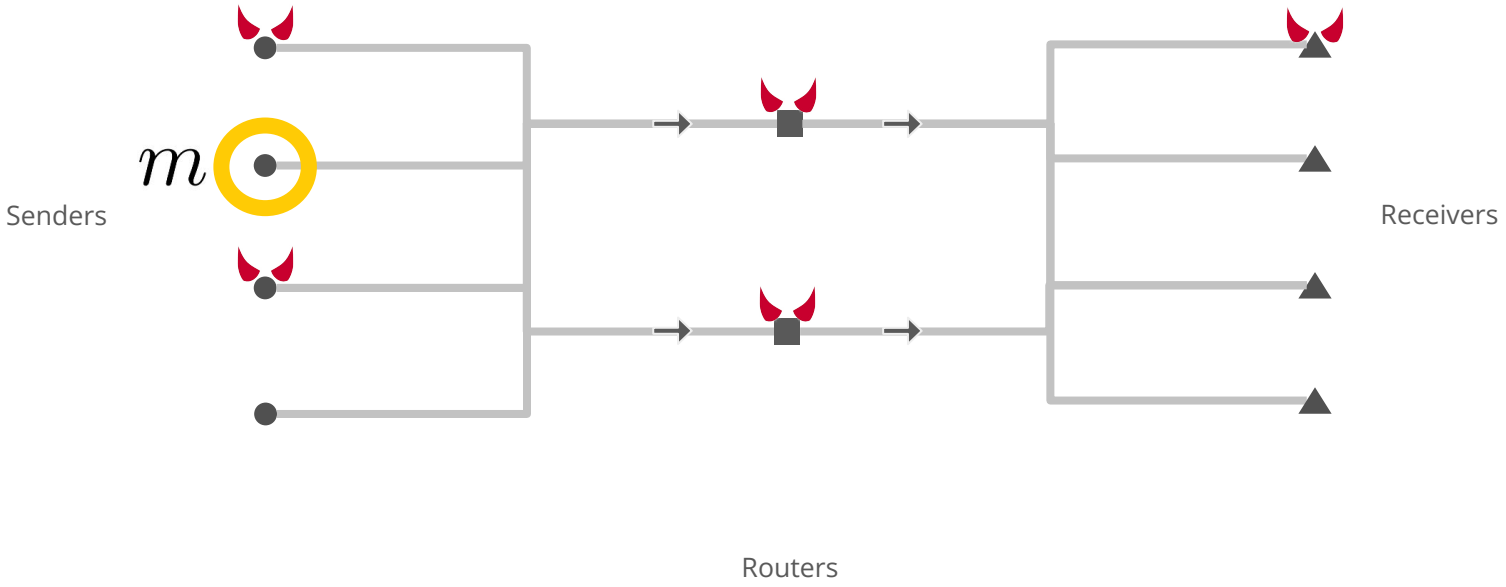
- Preliminaries and Overview
- Security Definitions
- Relevant Schemes/Algorithms
- **Our Construction**
- Results and Benchmarks
- Conclusion

ARN with Sender Anonymity

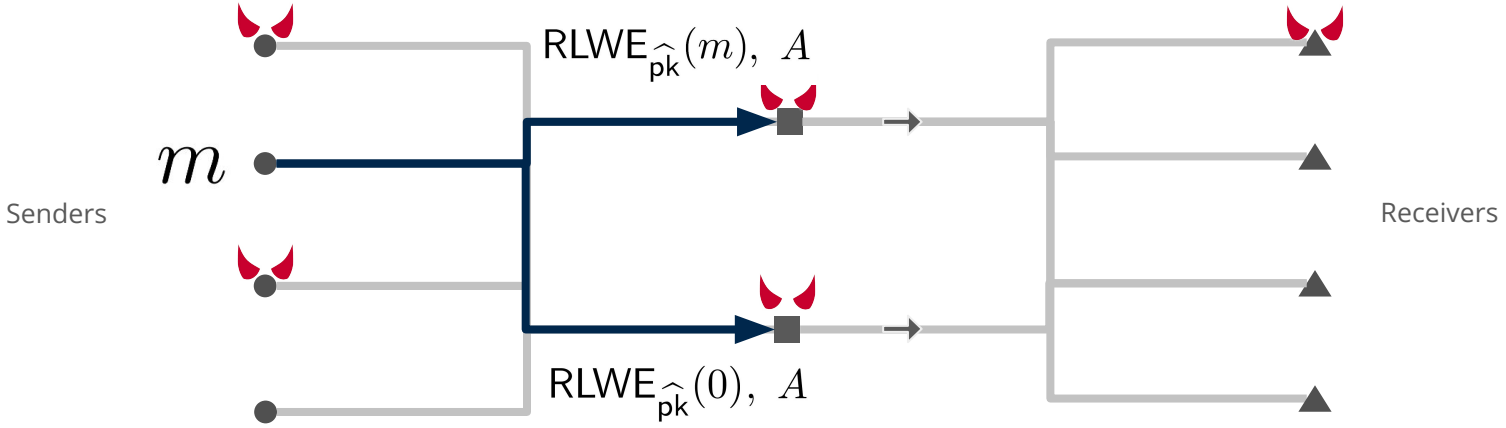
ARN with Sender Anonymity reminder

- We want messages to not be tied to its sender.
- We do not care if the message is seen by anyone.

ARN with Sender Anonymity example



ARN with Sender Anonymity example



$$\forall x \in [\log p \cdot n], A_x = \mathsf{RGSW}_{\widehat{\mathsf{pk}}}(a_x) \text{ where } a \in \{0,1\}$$

ARN with Sender Anonymity construction - I

- **Setup**$(1^\lambda, n, k)$: Output $\widehat{\mathsf{pk}}$ and user secret keys from $\mathsf{MP}.\mathbf{Setup}(1^\lambda, n, k)$.
- **Enc**$(\widehat{\mathsf{pk}}, m, \pi)$:
  - Randomly sample u from $\{1, \ldots, k\}$
  - For each router j: $c'_j := \begin{cases} \mathsf{RLWE}_{\widehat{\mathsf{pk}}}(m) & \text{if } j = u, \\ \mathsf{RLWE}_{\widehat{\mathsf{pk}}}(0) & \text{otherwise.} \end{cases}$
  - Randomly sample a from $\{0, \ldots, p \cdot n - 1\}$
  - Let A be the binarized a such that $A_x = \mathsf{RGSW}_{\widehat{\mathsf{pk}}}(a_x)$
  - Output $\{c_j\}_{j \in [k]}$ such that $c_j := (c'_j, A)$

ARN with Sender Anonymity construction - II

- We only have to use the placing algorithm for routing.

$\mathbf{Rte}(\{c_i\}_{i\in[n]})$:

- $c_i := (c'_i, A_i)$
- Output C, the addition of all $\mathbf{HomPlacing}(c'_i, A_i)$.

- The zero-encrypted messages do not affect the algorithm.

ARN with Sender Anonymity construction - III

- Collectively run the multi-party decryption method.

$\mathbf{Dec}(\text{user secret keys}, C)$:

- $\mathbf{m}^* \leftarrow \mathsf{MP}.\mathbf{Dec}(\text{user secret keys}, c^*)$ where $c^* \in C$.
- Output multiset of all non-zero $\mathbf{m}^*$

- Non-zero messages are forwarded to every user.

ARN with Receiver Anonymity

ARN with Receiver Anonymity

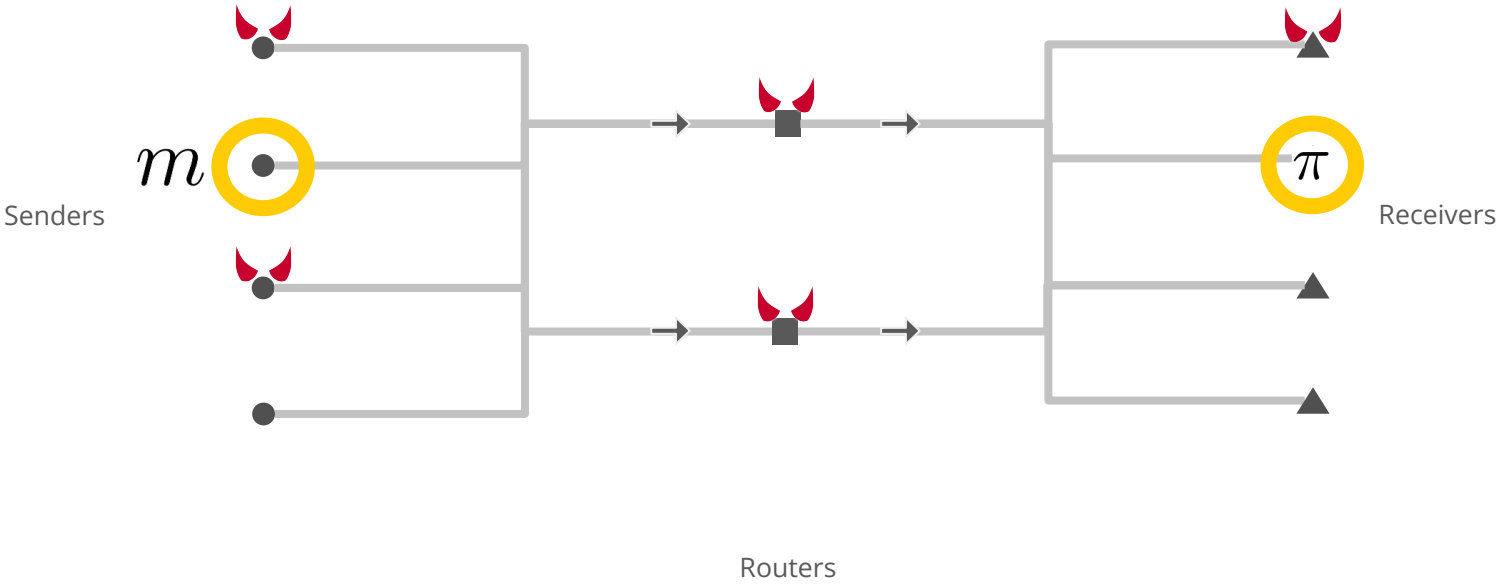
- Sender Anonymity seems very lacking.
- However, it is very useful in creating ARN with Receiver Anonymity.
- Reminder:
  - Senders don’t know the destination.
  - Receivers don’t know the sender.
  - A message is only known by its sender and receiver.
- A standard public key seems to fit this perfectly.

Defining public key encryption

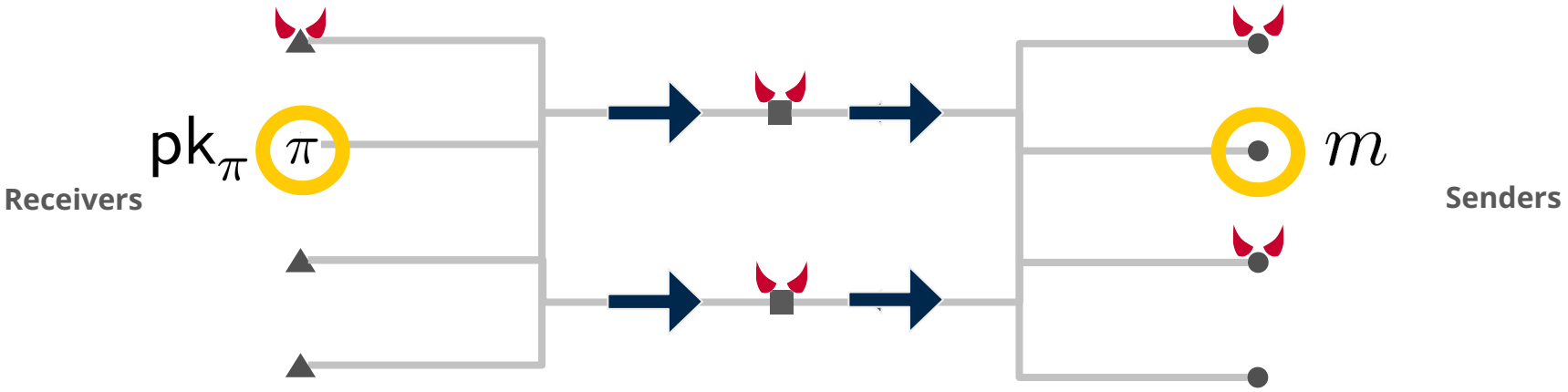
$$\text{Let } \mathsf{PKE} \coloneqq (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec}) \text{ where } \left\{ \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathbf{Gen}(1^\lambda), \\ c \leftarrow \mathbf{Enc}(\mathsf{pk}, m), \\ m \leftarrow \mathbf{Dec}(\mathsf{sk}, c). \end{array} \right\}$$

A user's public key should not have identifiable information!

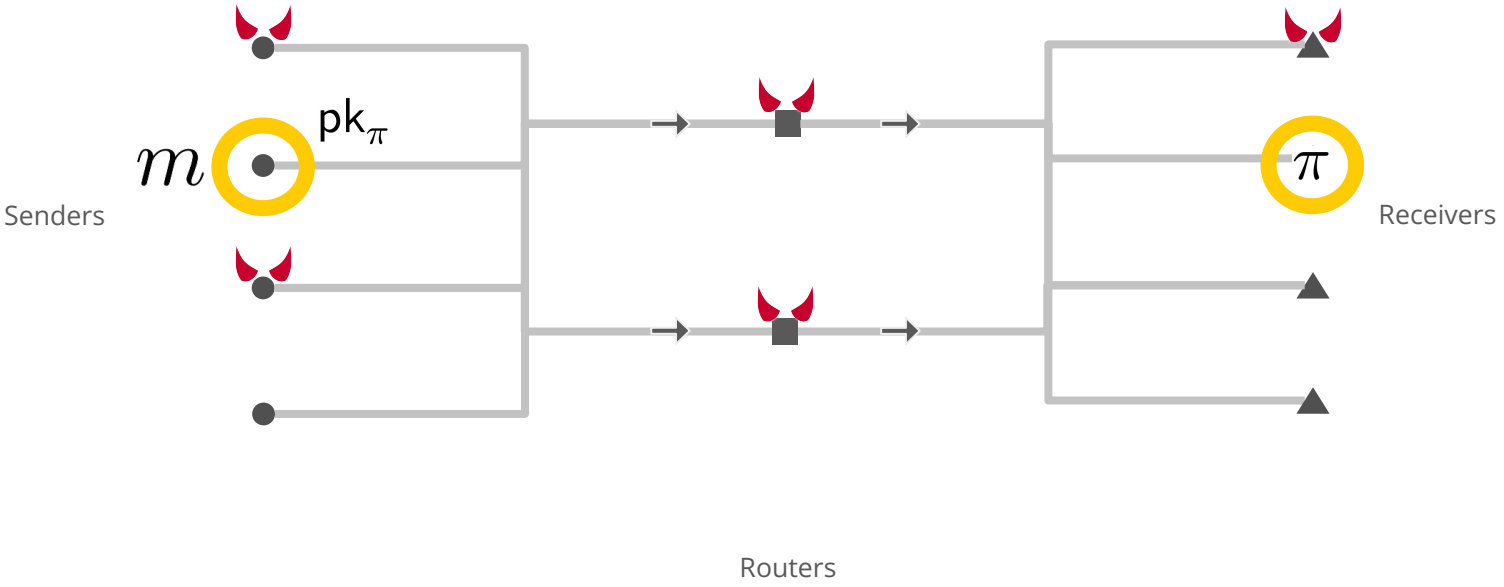
Receiver Anonymity example



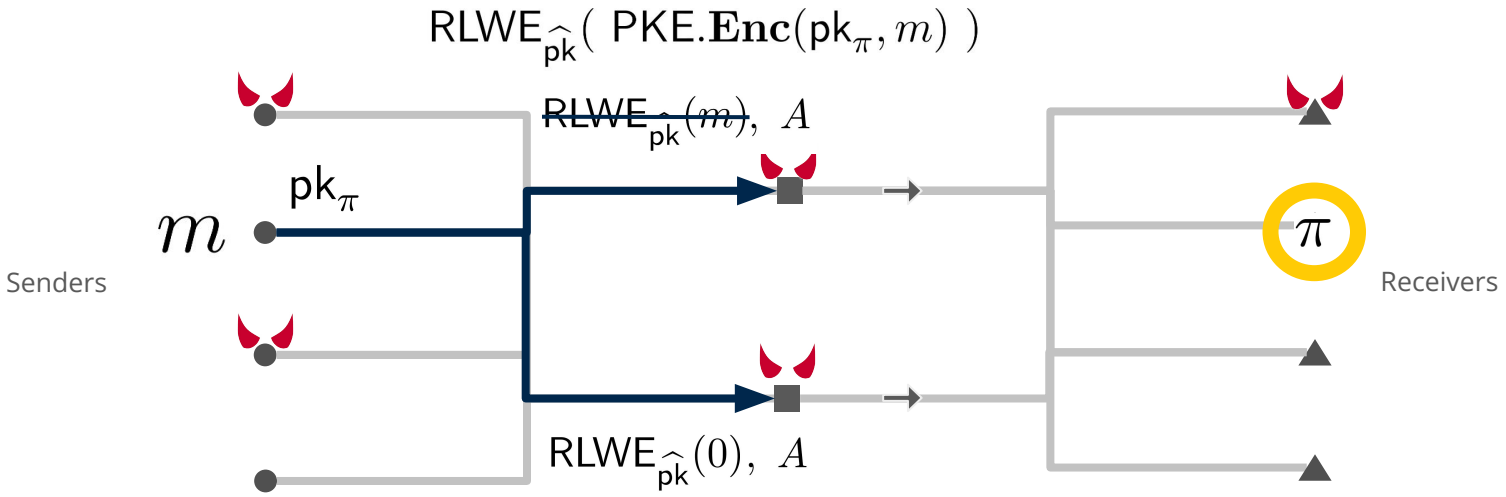
Receiver Anonymity example



Receiver Anonymity example



Receiver Anonymity example



$$\forall x \in [\log p \cdot n], A_x = \mathsf{RGSW}_{\widehat{\mathsf{pk}}}(a_x) \text{ where } a \in \{0,1\}$$

ARN with Receiver Anonymity construction - I

- $\mathbf{Setup}(1^\lambda, n, k)$:
  - Output $\widehat{\mathsf{pk}}^*$ and user secret keys from $\mathsf{MP}.\mathbf{Setup}(1^\lambda, n, k)$.
  - For each receiver ℓ: $\mathsf{pk}_\ell, \mathsf{sk}_\ell \leftarrow \mathsf{PKE}.\mathbf{Gen}(1^\lambda)$

ARN with Receiver Anonymity construction - II

- $\mathbf{Enc}(\widehat{\mathsf{pk}}, m, \pi)$:
  - $\widehat{\mathsf{pk}} := (\widehat{\mathsf{pk}}^*, \mathsf{pk})$ and $c^* := \mathsf{PKE}.\mathbf{Enc}(\mathsf{pk}_\pi, m)$.
  - Output ARN with sender anonymity's $\mathbf{Enc}(\widehat{\mathsf{pk}}^*, c^*, \pi)$

Roadmap

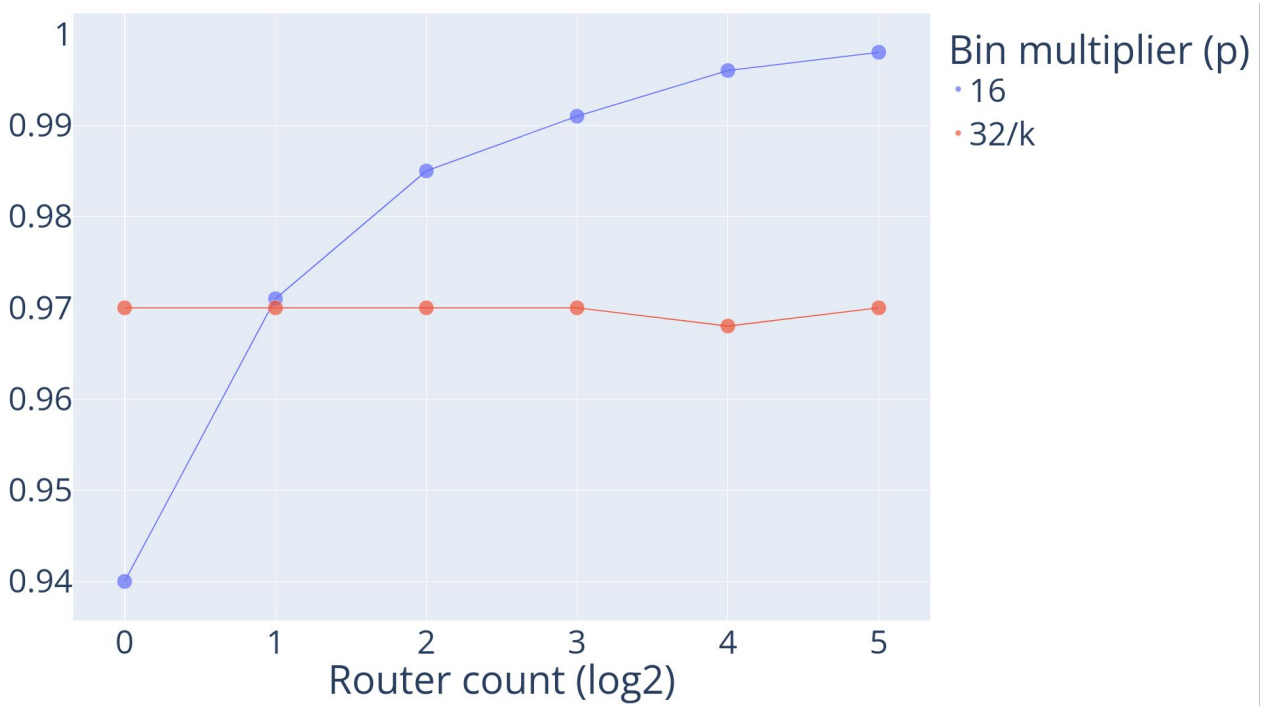
- Preliminaries and Overview
- Security Definitions
- Relevant Schemes/Algorithms
- Our Construction
- **Results and Benchmarks**
- Conclusion

The bin multiplier p

- Failure is largely dependent on it.
- We show the probability of success for two values:

$$p \in \{\ 16,\ 32/k\ \}$$

Success probabilities per the bin multiplier



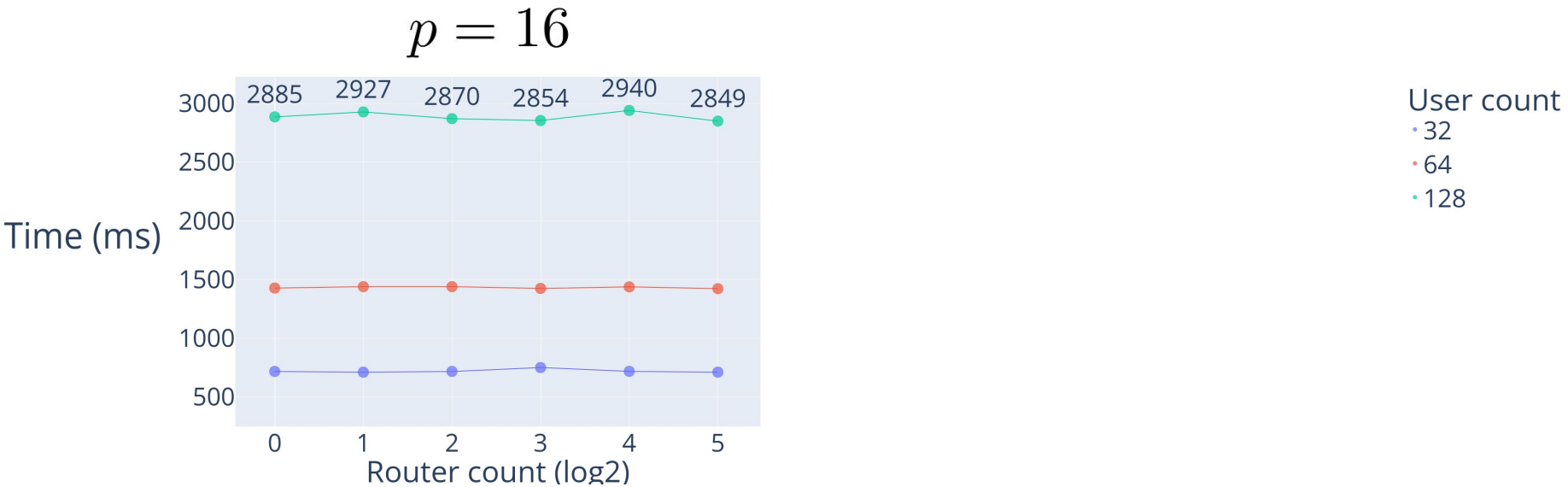
Implementation details

- Using the Lattice-Estimator library, yields **115 bits of security**.
- Built in Rust with the TFHE-rs library.
- Intel i5-14400
- 32GB of RAM

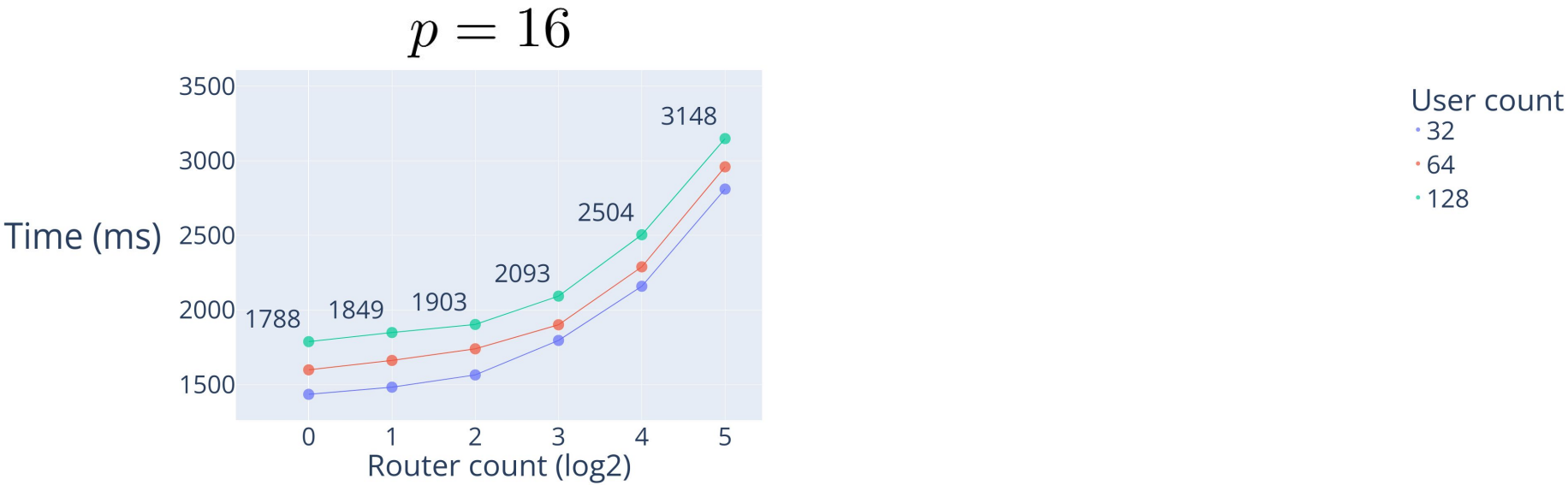
Benchmark details

- We show computation times for
  - $p \in \{\ 16,\ 32/k\ \}$
  - and $n \in \{32, 64, 128\}$.
- We give a non-direct comparison to sPAR

Setup times for $p \in \{ 16, 32/k \}$

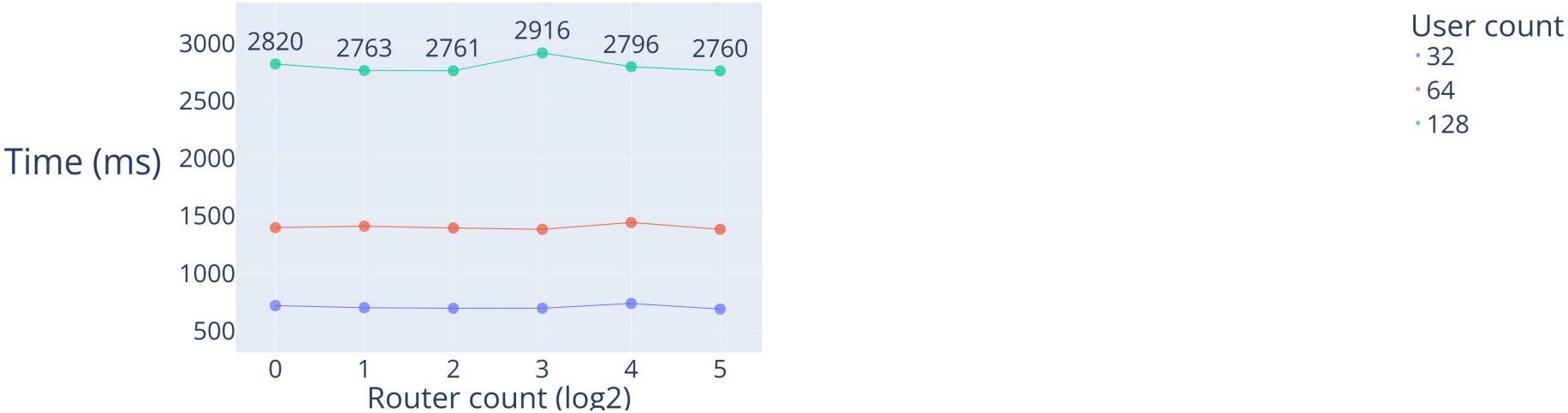


Encryption times for $p \in \{ 16, 32/k \}$

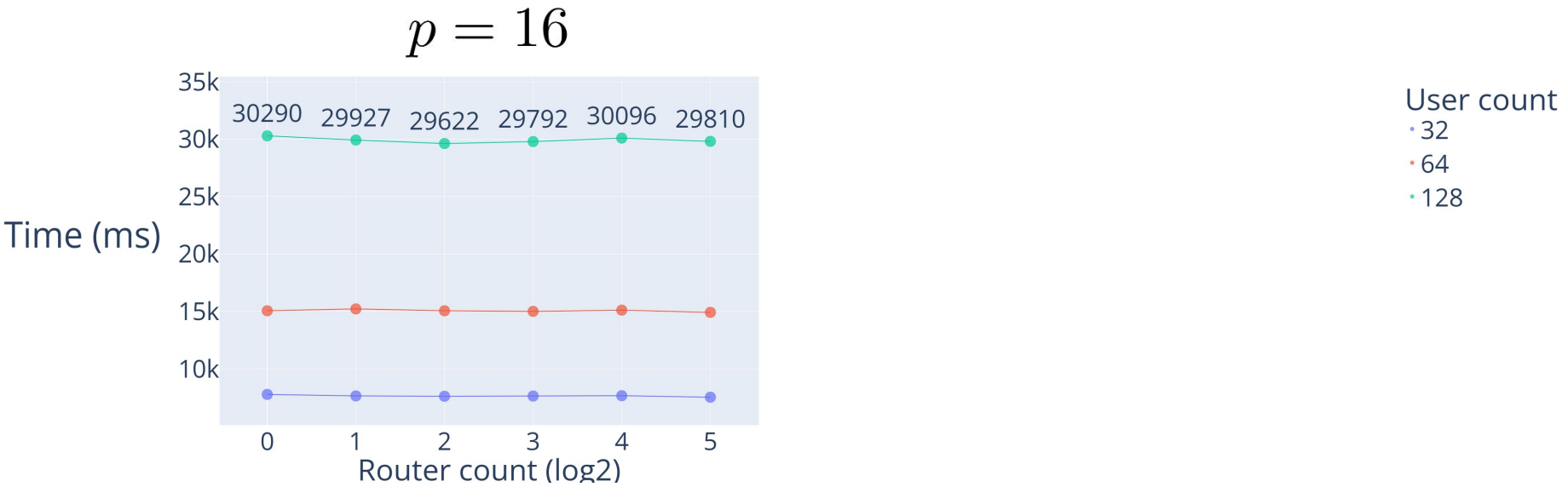


Decryption times for $p \in \{ 16, 32/k \}$

$p = 16$



Router times for $p \in \{ 16, 32/k \}$



Roadmap

- Preliminaries and Overview
- Security Definitions
- Relevant Schemes/Algorithms
- Our Construction
- Results and Benchmarks
- **Conclusion**

Another detour for the bin multiplier p

- This is a significant departure from sPAR.
- Allows a smaller bin multiplier and smaller probability of failure.
- Essentially, it allows for $p = 1/k$
- However, there are some complications.

Final remarks

- The scenario in which some routers fail, a subset of messages still get transmitted.
- This opens the door to users picking a subset of routers
- We achieved an improvement in performance

Thank you

Questions?

References - I

- Craig Gentry, Amit Sahai, and Brent Waters. “Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based”. In: Advances in Cryptology – CRYPTO 2013, pp. 75–92. isbn: 978-3-642-40041-4.
- Vadim Lyubashevsky, Chris Peikert, and Oded Regev. “On ideal lattices and learningwith errors over rings”. In: EUROCRYPT’10. French Riviera, France: Springer-Verlag,2010, pp. 1–23. isbn: 3642131891. doi: 10 . 1007 / 978 - 3 - 642 - 13190 - 5 _ 1. url:https://doi.org/10.1007/978-3-642-13190-5_1.
- Jeongeun Park. “Homomorphic Encryption for Multiple Users With Less Communi-cations”. In: IEEE Access 9 (2021), pp. 135915–135926. doi: 10.1109/ACCESS.2021.3117029.
- Shi, E., Wu, K. (2021). Non-Interactive Anonymous Router. In: Canteaut, A., Standaert, FX. (eds) Advances in Cryptology – EUROCRYPT 2021. EUROCRYPT 2021. Lecture Notes in Computer Science(), vol 12698. Springer, Cham. https://doi.org/10.1007/978-3-030-77883-5_17
- Roger Dingledine, Nick Mathewson, and Paul Syverson. “Tor: The Second-Generation Onion Router”. In: 13th USENIX Security Symposium (USENIX Security 04). San Diego, CA: USENIX Association, Aug. 2004. url: https : / / www.usenix.org /conference/13th-usenix-security-symposium/tor-second-generation-onion-router.

References - II

- Ilaria Chillotti et al. “TFHE: Fast Fully Homomorphic Encryption Over the Torus”. In: 33.1 (Jan. 2020), pp. 34–91. issn: 0933-2790. doi: 10.1007/s00145-019-09319-x. url: https://doi.org/10.1007/s00145-019-09319-x.
- Christian Mouchet et al. “Multiparty Homomorphic Encryption from Ring-Learning-with-Errors”. In: Proceedings on Privacy Enhancing Technologies 2021 (Oct. 2021), pp. 291–311. doi: 10.2478/popets-2021-0071.
- Kelong Cong et al. “SortingHat: Efficient Private Decision Tree Evaluation via Homomorphic Encryption and Transciphering”. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. CCS ’22. Los Angeles, CA, USA: Association for Computing Machinery, 2022, pp. 563–577. isbn: 9781450394505. doi: 10 . 1145 / 3548606 . 3560702. url:https://doi.org/10.1145/3548606.3560702.
- Zama. TFHE-rs: A Pure Rust Implementation of the TFHE Scheme for Boolean and Integer Arithmetics Over Encrypted Data. https://github.com/zama-ai/tfhe-rs. 2022.
- Martin R. Albrecht, Rachel Player, and Sam Scott. “On the concrete hardness of Learning with Errors”. In: Journal of Mathematical Cryptology 9 (2015), pp. 169–203. url: https://api.semanticscholar.org/CorpusID:86408.