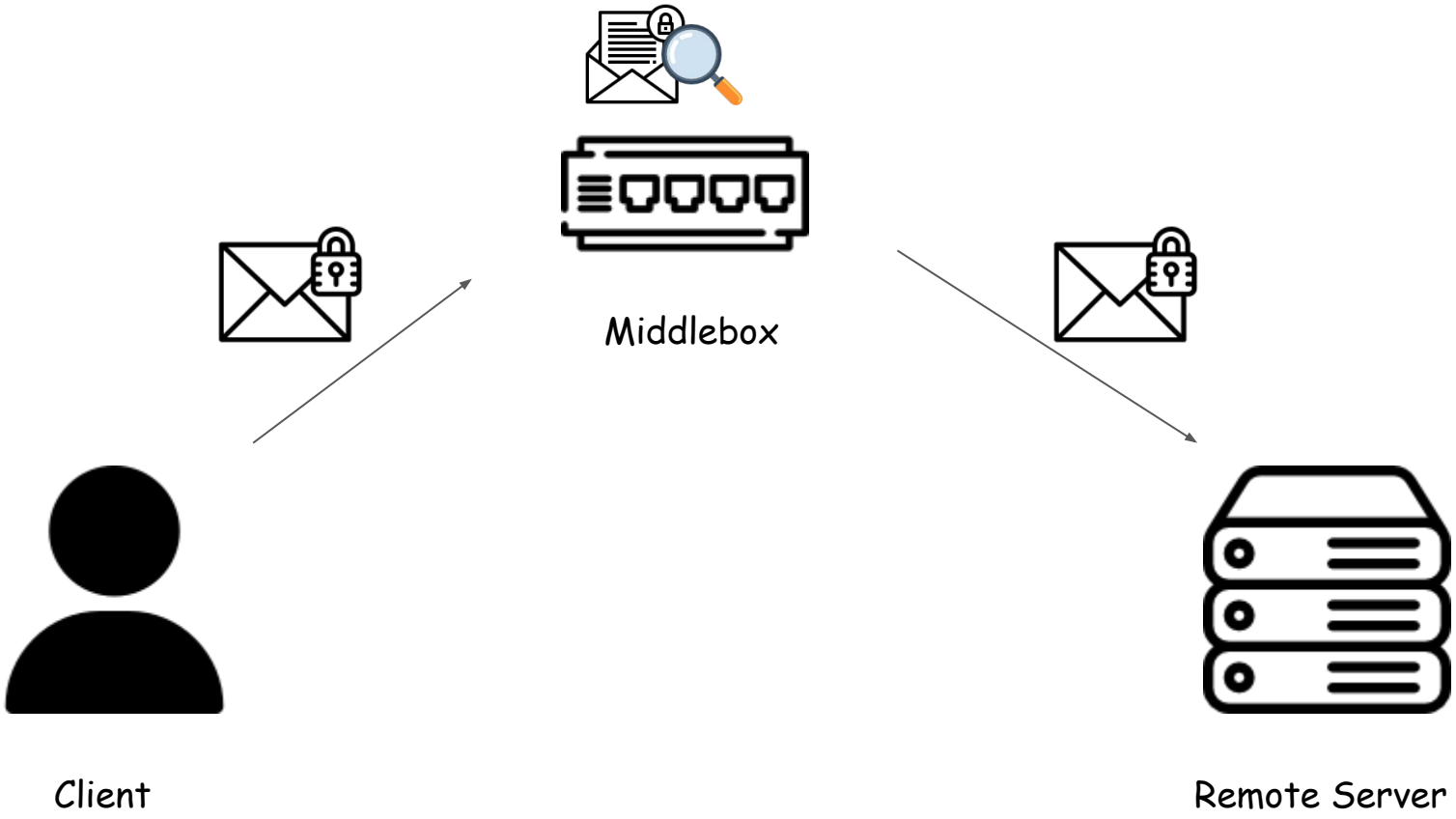
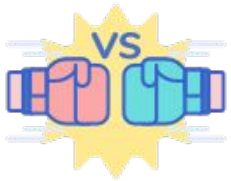
Zombie: Middleboxes that Don’t Snoop

Collin Zhang, Zachary DeStefano, Arasu Arun,
Joseph Bonneau,Paul Grubbs, Michael Walfish

NSDI 2024


Middlebox
Client
Remote Server



Privacy

Policy

Can we get the benefits of both worlds?

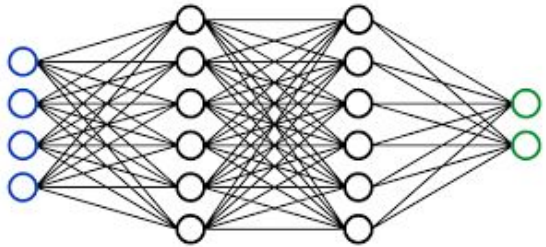
Zero-Knowledge Proof to the rescue

What can we prove in ZK?

All problems in PSPACE

- A program that checks if an assignment of a sudoku is satisfied, and output 1 or 0
  - Prove you know how to solve the puzzle without revealing your solution
- SHA function
  - Prove you know a preimage without revealing it
- Neural Network
  - Prove you know some input produce a certain output without revealing input

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9



Zero-Knowledge Proofs

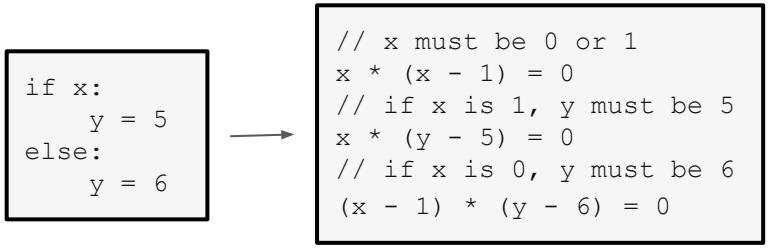
prover

- Soundness: A false statement cannot be proved
- Zero-Knowledge: Verifier learns nothing about the private inputs

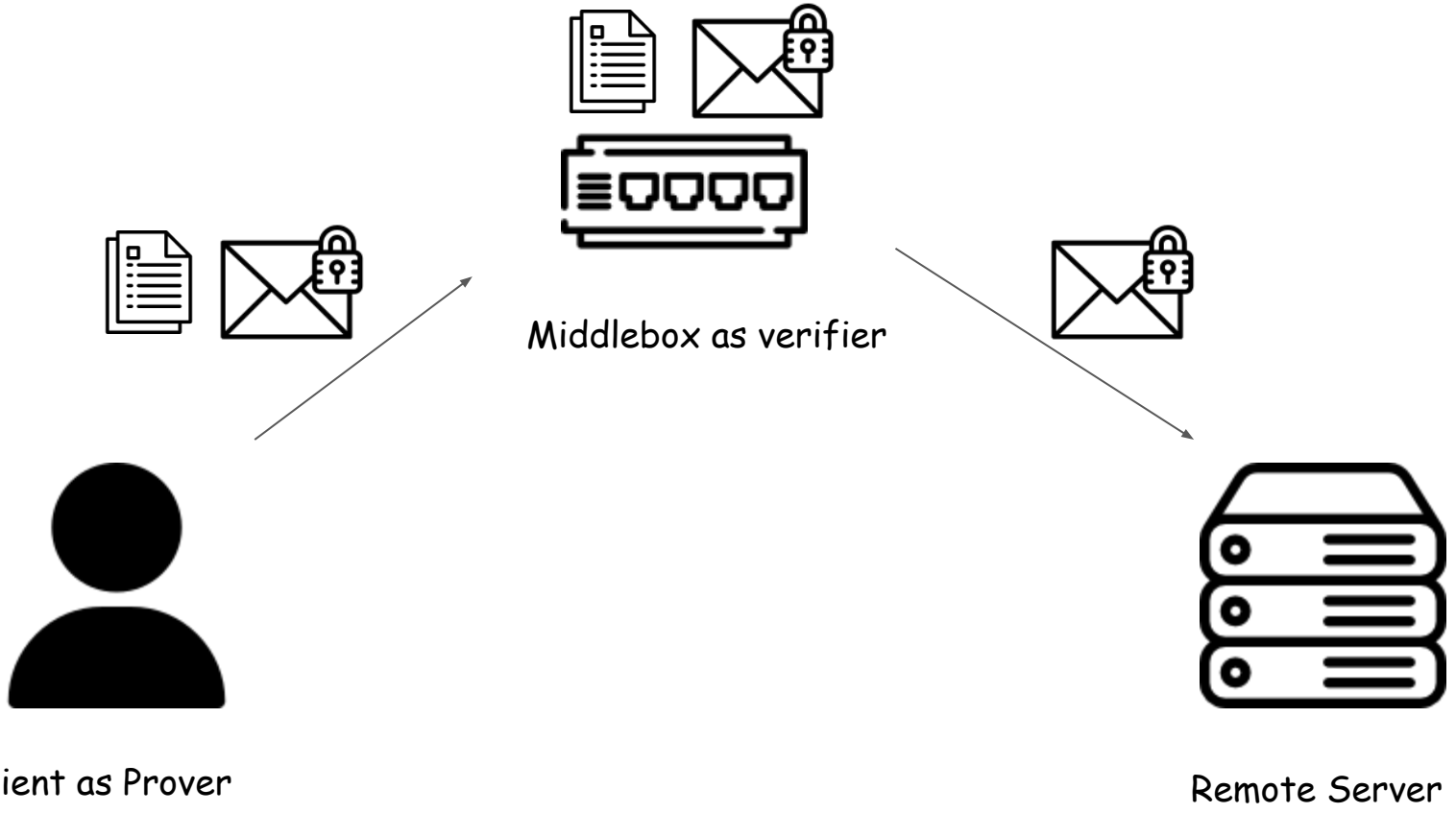
verifier

How does zero-knowledge proof work?

- Write a program in a high-level language
- Frontend
  - A compiler compiles the program to a set of constraints
  - Then compile the set of constraints to a polynomial that always evaluates to 0 if the constraints are satisfied
- Backend
  - Prove: The prover commits to the polynomial with a technique called polynomial commitment
  - Verify: The verifier evaluates the polynomial at a random point without knowing the polynomial



Zero-Knowledge Middleboxes (ZKMB)



DecryptAndCheck

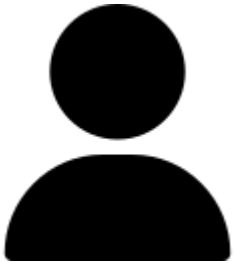
public inputs:
encrypted message,
Private key hash (commit)

private inputs:
Private key

Statement: This Program evaluates to yes

- Check private key commitment
- Derive pseudorandom pad from the private key
- Decrypt the message by XOR it with pseudorandom pad
- Check message for policy compliance

yes/no



client

middlebox

ZKMB latency

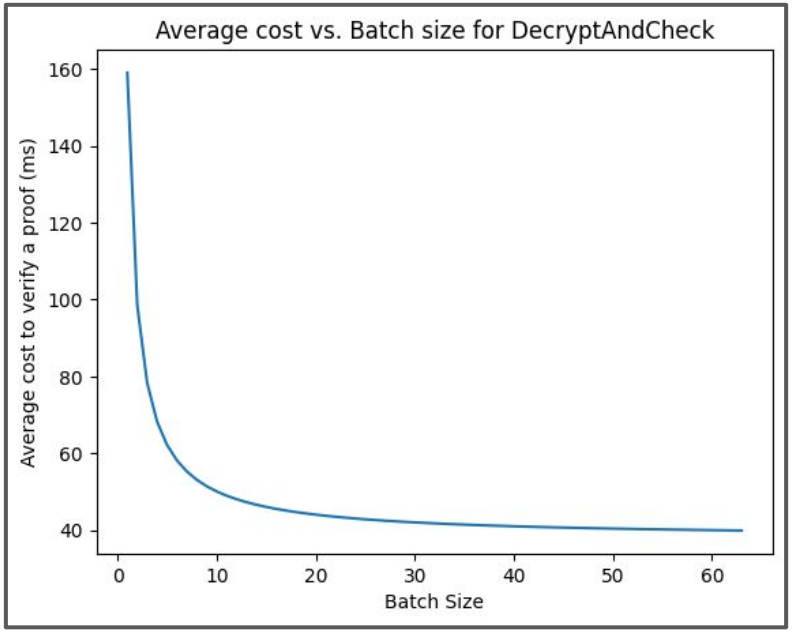
- ZKMB: Groth 16
  - Prover time: 1200 ms
  - Verifier time 1.6 ms
- Zombie: Spartan
  - Prover time: 345 ms
  - Verifier time: 44 ms
- DNS request latency: 20 ms

Zombie

- How can we reduce verifier time while maintain low prover time?
- How can we further reduce the latency?

Batching

- High verification cost => low throughput
- Batch verification
  - Verifier evaluates two polynomials
    - Polynomial encodes the constraints
    - Polynomial encodes the solution of constraints
  - The constraints polynomial is independent of the inputs, so we can reuse that
- Limitation
  - Can only batch proofs from same client
  - Client has to accumulate packets to batch prove them



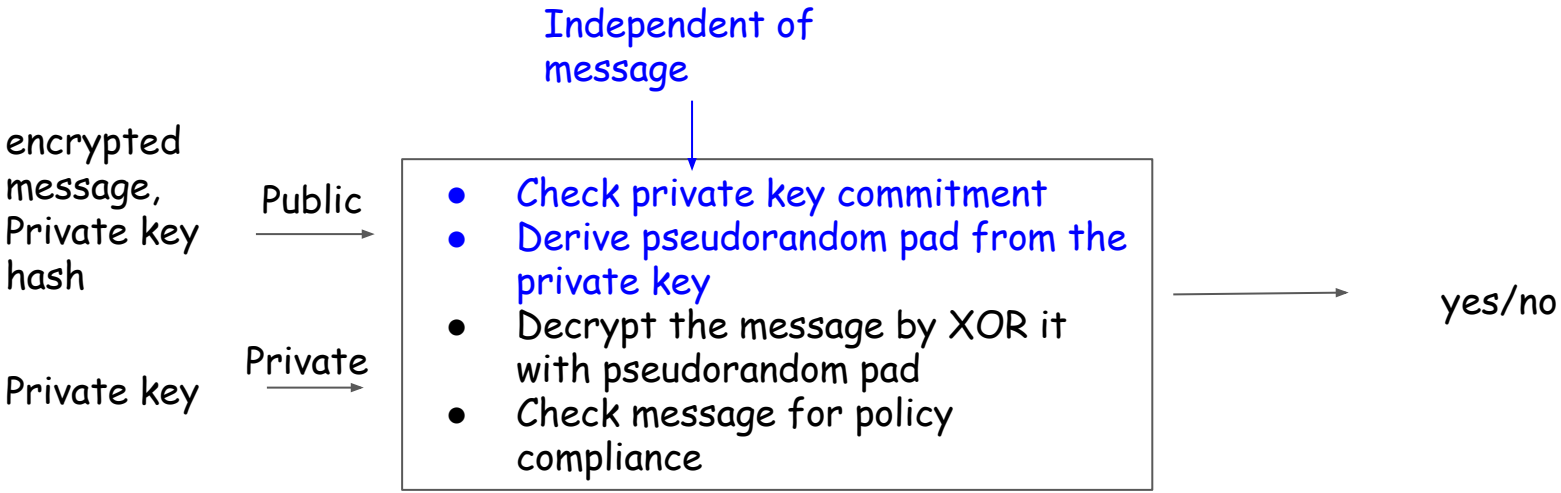
Optimistic approval

- Relaxed security model => Zero Latency
  - Middlebox forward the traffic immediately
  - Middlebox expects a proof from the client within a window of time
  - If proof is invalid or not received, client banned from the public network
- Security sufficient for dns filtering
  - Even if the client knows IP address, it can't browse the website for too long
- Synergy with batching
  - Client can now accumulate proofs and batch proving them!

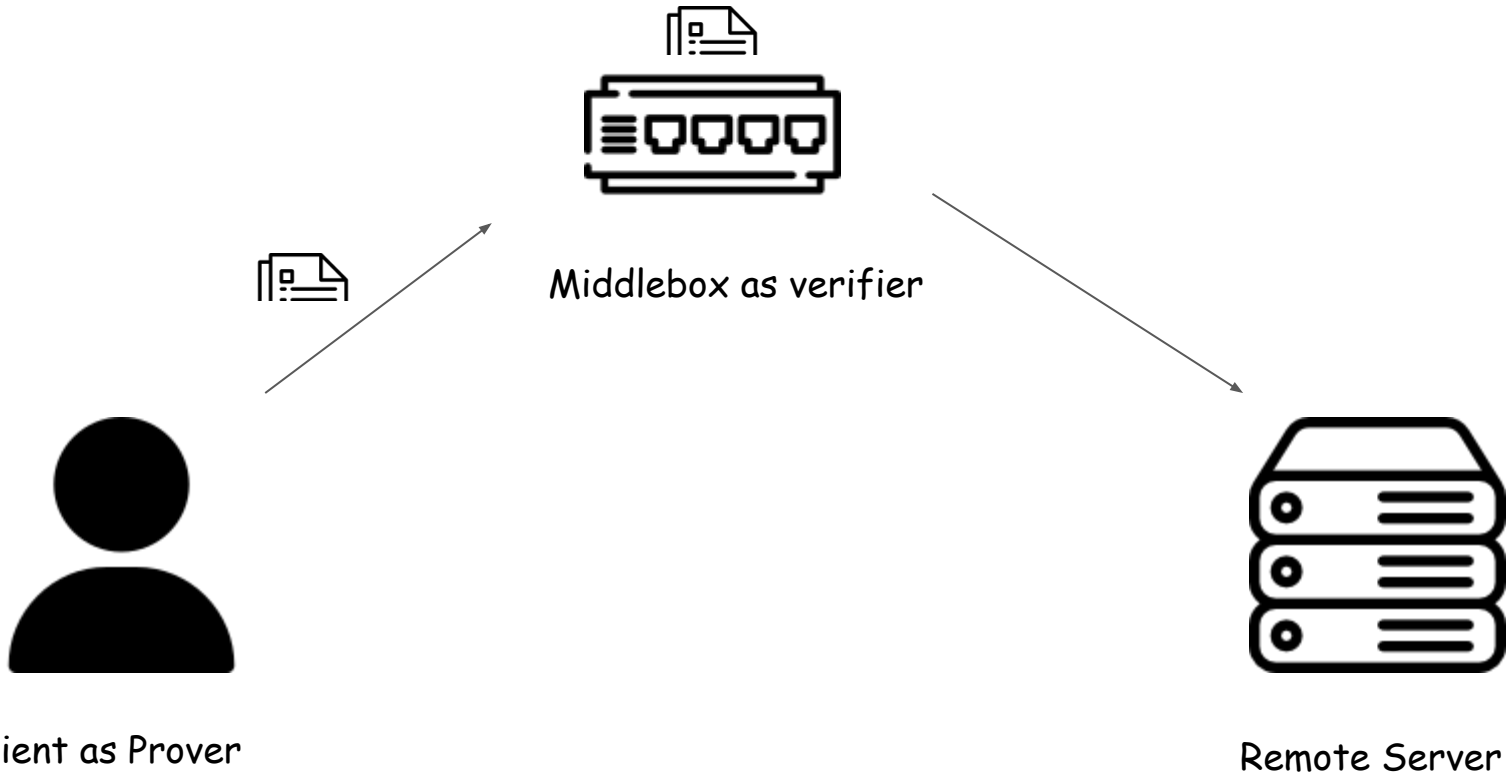
Precompute

- How can we reduce latency without security compromise?

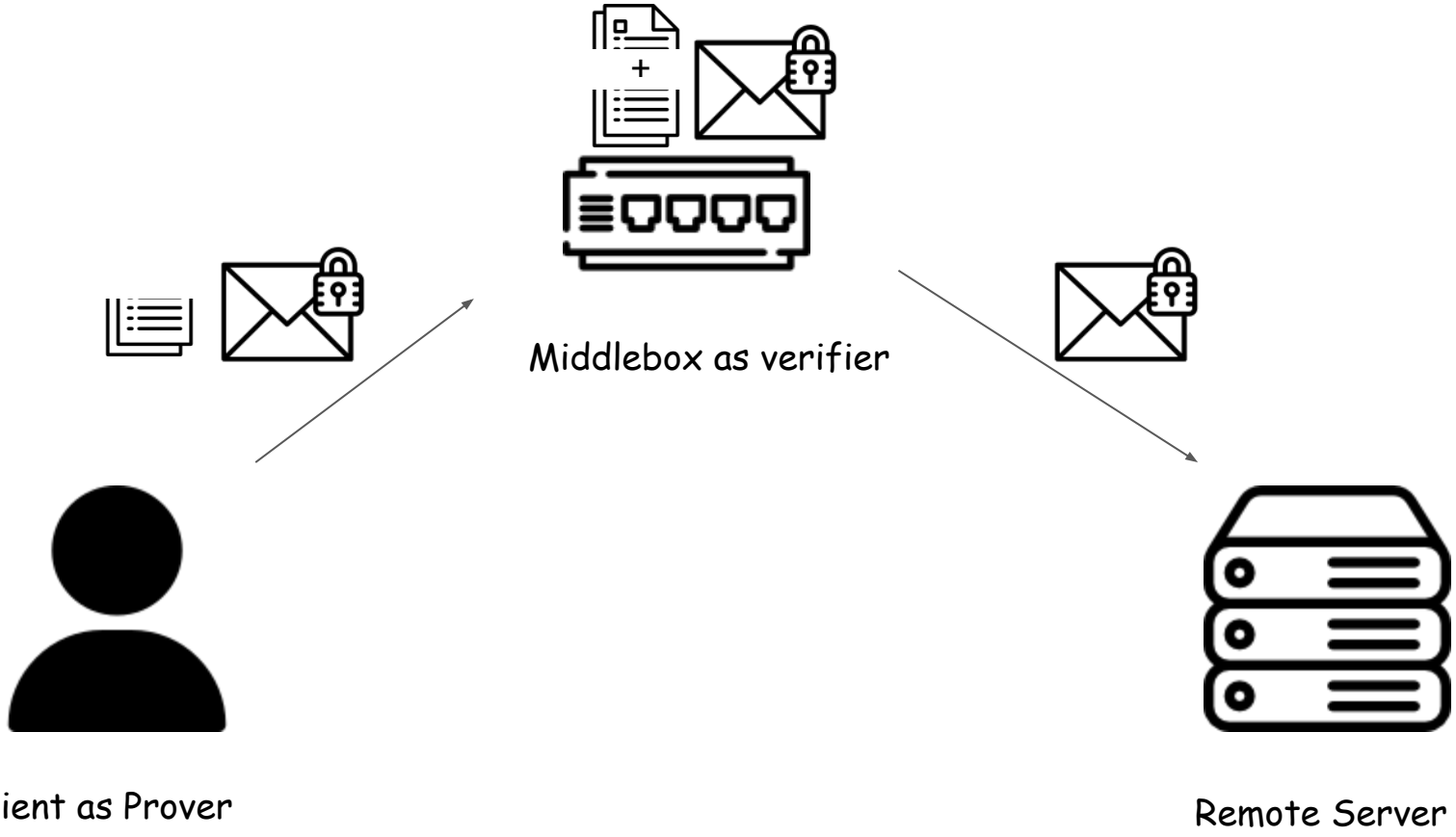
DecryptAndCheck statement



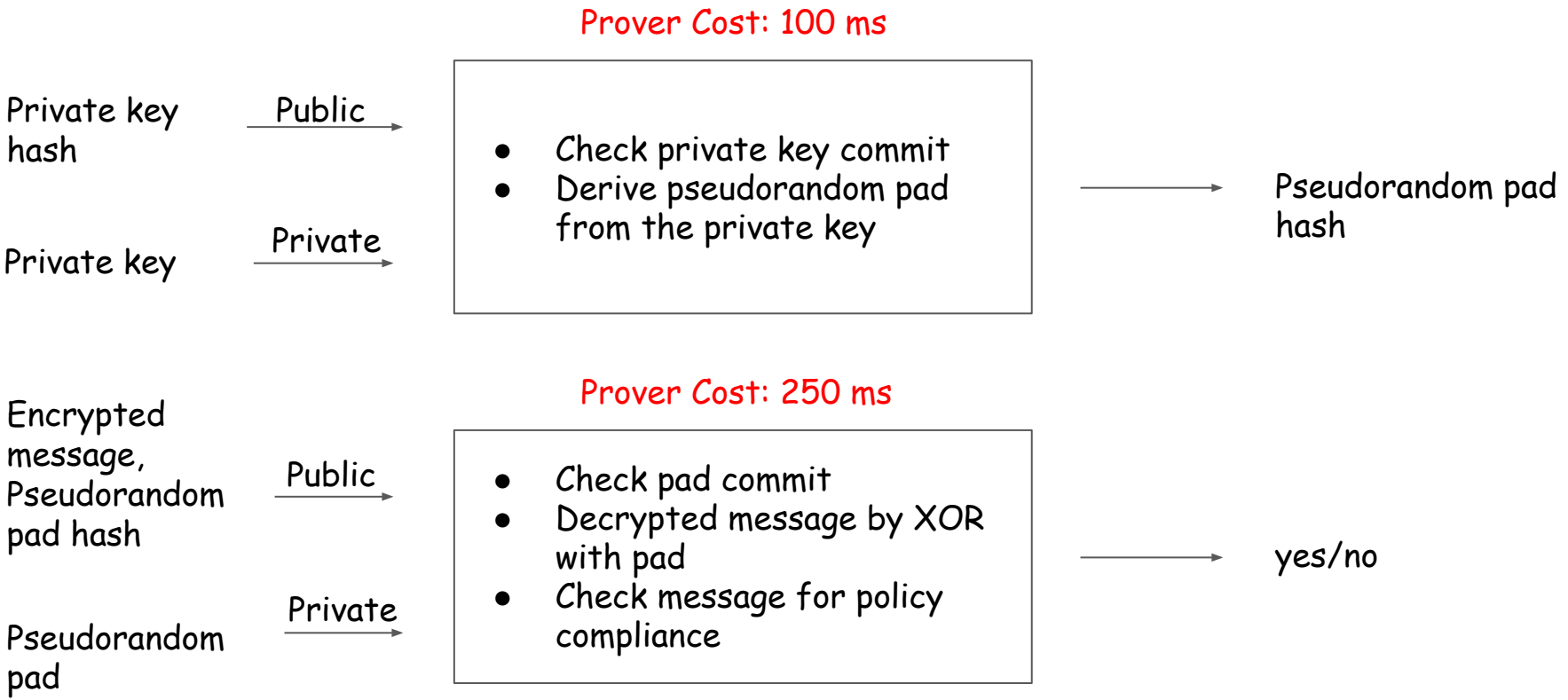
Precompute: When client is idle



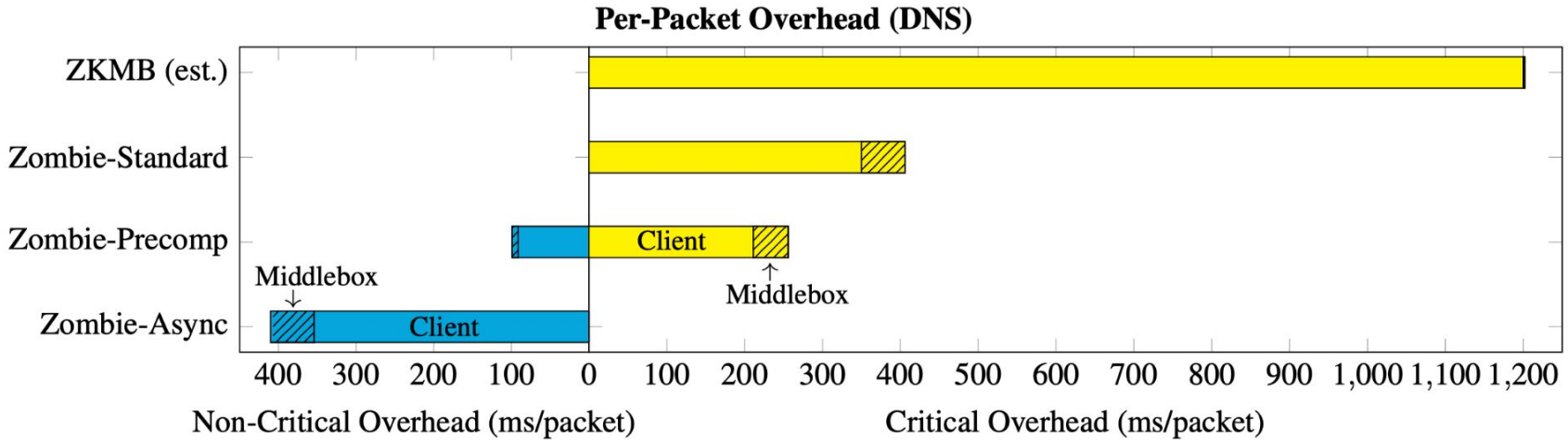
Precompute: When client sends a message



Decompose of DecryptAndCheck statement



Zombie Improvements



Zombie Limitations

- Latency
  - Best sync mode extra latency: 250ms
  - DNS latency: 20ms
- Computation intensive
  - 16 cores CPU run 350ms for each DNS request
- The computation will be more intensive for more complex policies
  - 6 seconds for Microsoft Purview Data Loss Prevention

Thank you!