Side-Channel Vulnerabilities in Networking and AI Systems

Sihang Liu

University of Waterloo

Oct 28, 2025

UNIVERSITY OF WATERLOO

Explosion of Cloud and AI

- Cloud is foundational in today’s computing
  - Scalable compute and storage
  - Elastic resource allocation
- Networking is especially critical to the cloud
  - Achieve high-speed interconnect
- AI workloads are usually deployed in the cloud, e.g.,
  - Large language models for chatbot and code generation
  - Diffusion models for creative image/video generation

New Security Concerns

- Sharing is common in cloud environments
- Resource sharing introduces new security risks, e.g.,
  - Network interconnect
  - Generative AI models
  - Compute and storage systems
- Virtualization and isolation can mitigate leakages but not always

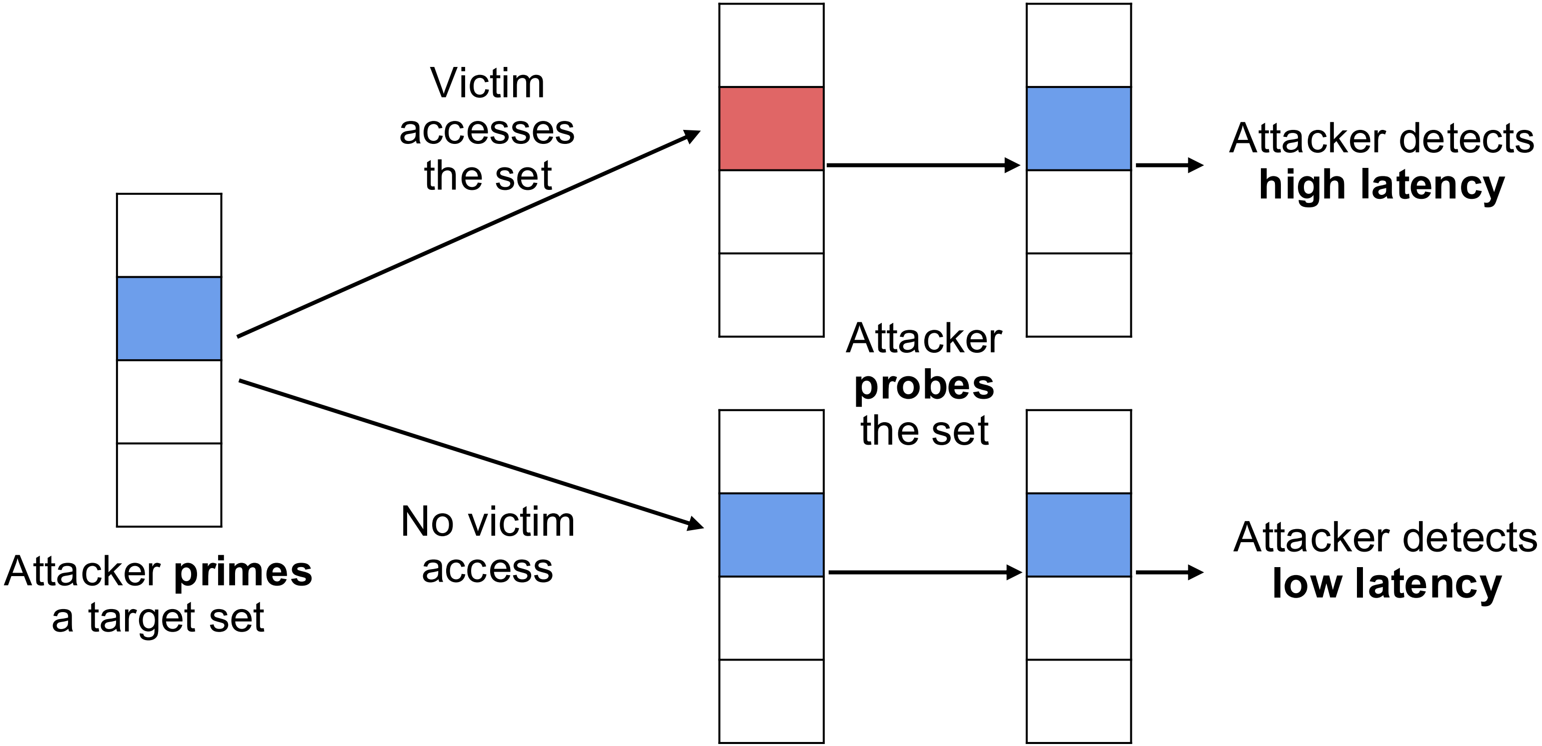
What Are Side-Channel Attacks?

- Side channels are based on indirect, unintended behaviors or features
- Secretly leak information about the target system
- Examples of side channels:
  - Chip power
  - Thermal signal
  - Electromagnetic signal
  - Timing difference

Cache is one of the most typical examples that lead to timing differences

Example of Cache Side Channel

A Prime+Probe attack can infer bits of a memory address through cache



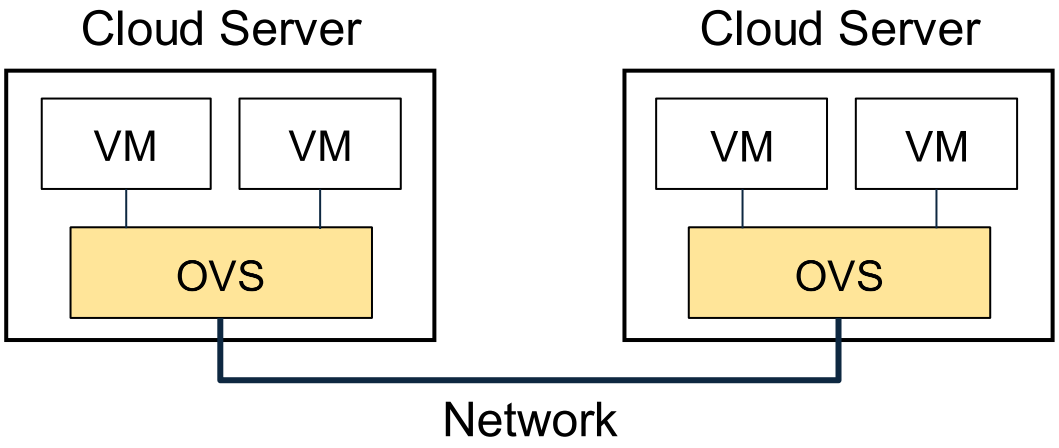
Outline

- **Background**
- Side-channel Study I: Open vSwitch
  - Remote packet header recovery attack
  - Remote packet rate monitoring attack
- Side-channel Study II: Prompt caching of image generation
  - Remote covert channel
  - Prompt stealing attack

Case Study I
Side Channels in Open vSwitch

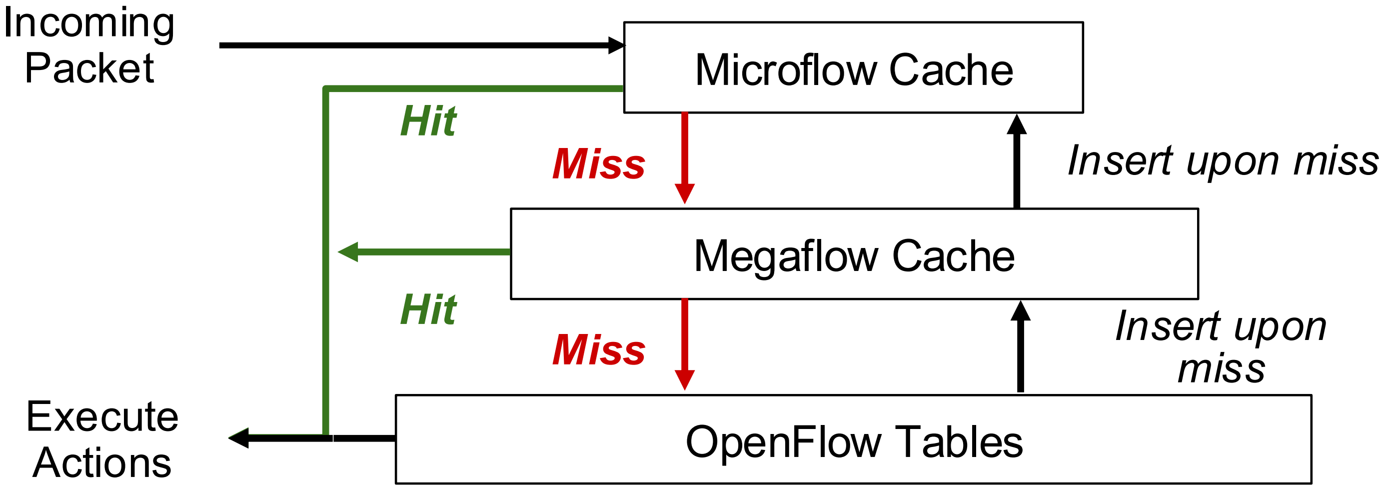
Open vSwitch (OVS)

- Cloud environments usually share resources via VMs
- OVS is one way to accelerate the network in cloud environment
  - It is a virtual switch that connects VMs or containers
  - Works with SDN and enables efficient and flexible network management



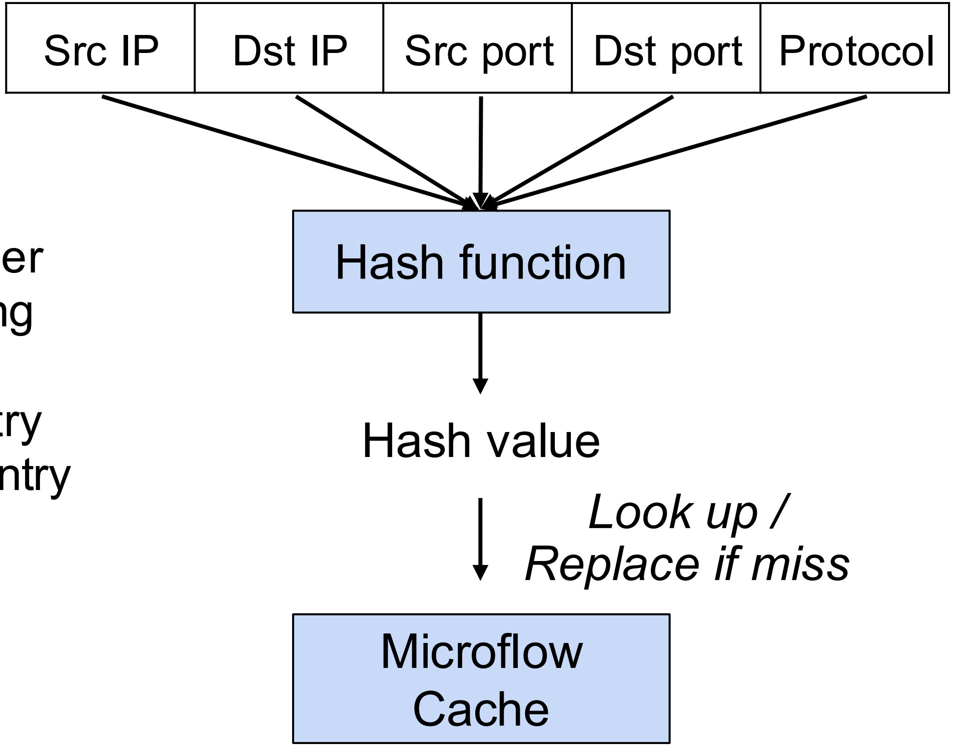
Flow Caching in OVS

- OVS caches network flows to accelerate flow actions
- It has a two-level cache: **microflow cache** and **megaflow cache**
  - The incoming packet first looks up microflow cache
  - If **hit**, execute actions
  - If **miss**, look up megaflow cache and insert flow back to microflow

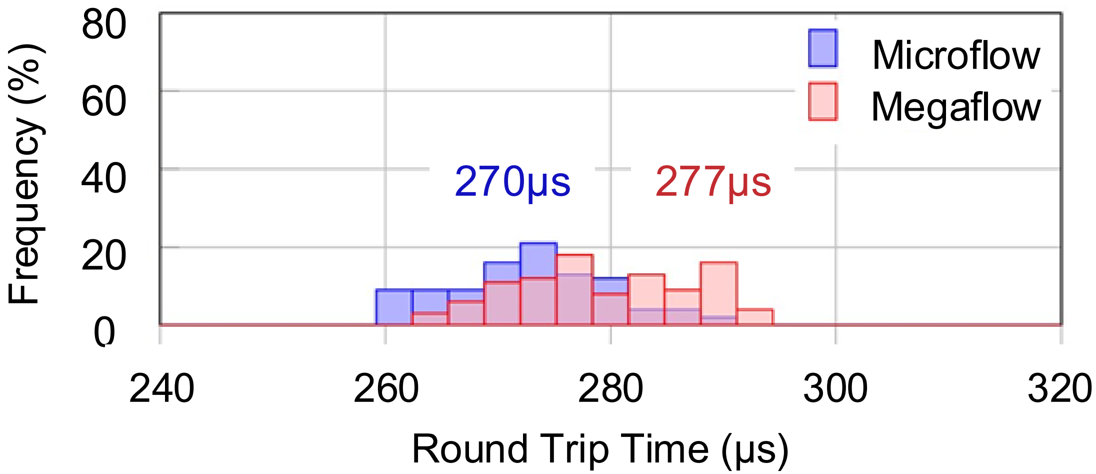


Microflow Cache

- Microflow cache is based on a hash table
- Lookup process:
  1. Generate hash from packet header
  2. Look up the microflow cache using the hash value
     - **Hit**: Use the cached flow entry
     - **Miss**: Replace the conflict entry



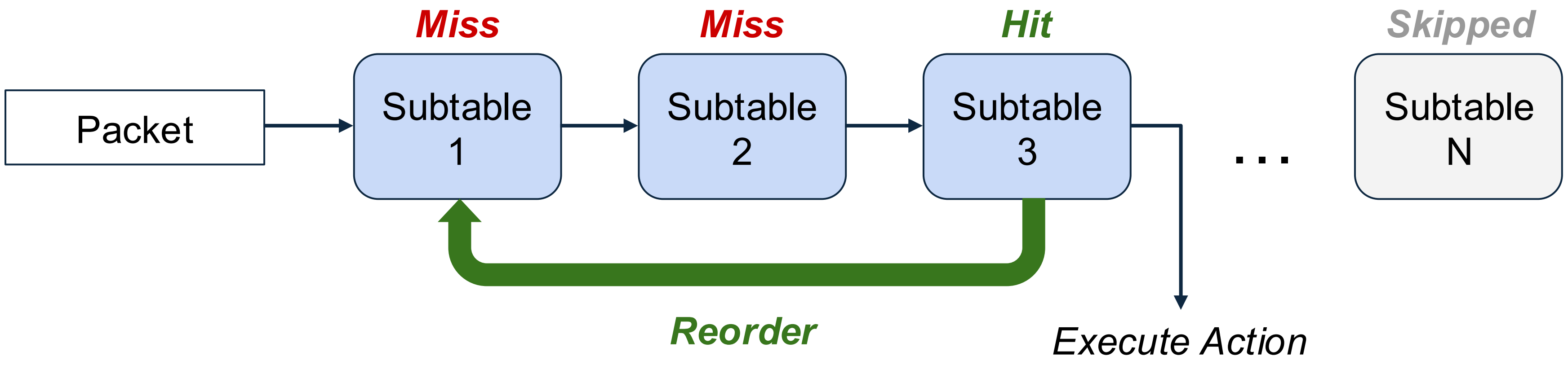
Microflow Cache Latency



OVS microflow and megaflow have distinguishable latencies

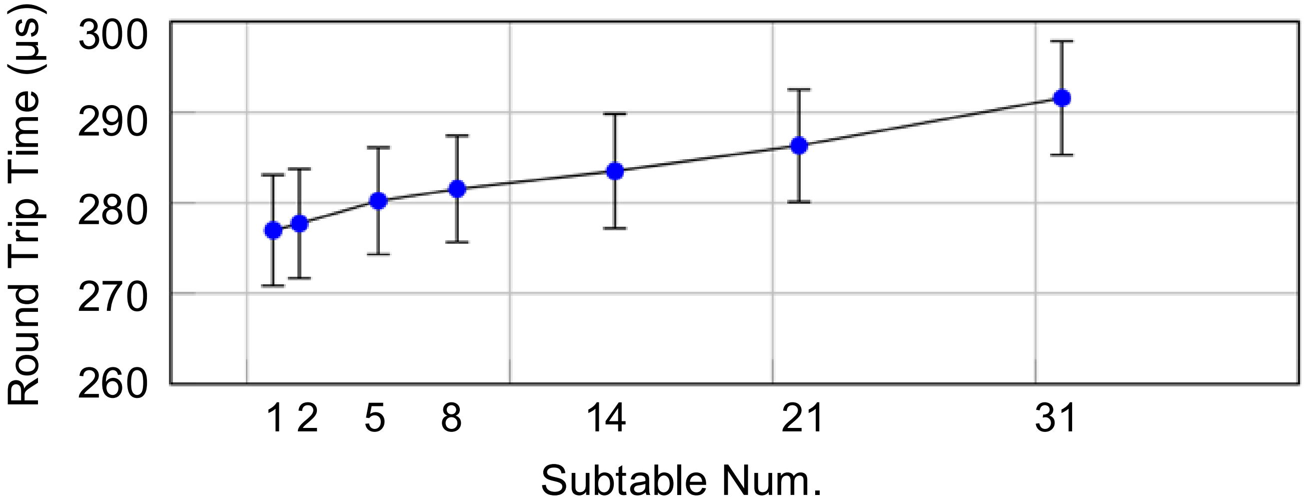
Megaflow Cache

- Megaflow cache consists of a number of subtables
  - Look up subtables sequentially until hit
  - Latency depends on num. subtable lookups
  - Subtables are reordered to the front if accessed frequently



Megaflow Cache Latency

- Evaluate access latency vs. subtable location



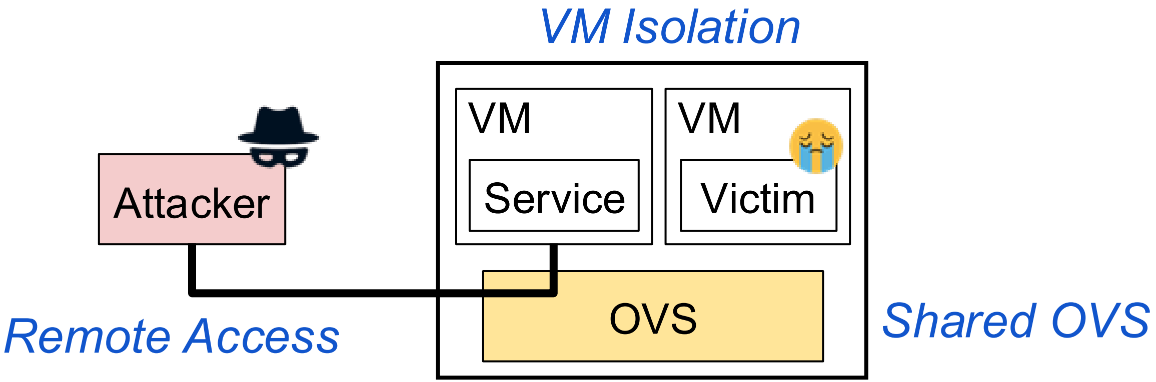
Megaflow has distinguishable levels of latencies, corresponding to subtables order

Summary of OVS Attack Primitives

1. **Timing difference in OVS**
   - Microflow, megaflow, and cache miss have distinguishable latencies
2. **Microflow cache hash collisions**
   - The hash for the microflow cache is generated from packet header fields
   - Collisions can leak information about packet header fields
3. **Megaflow subtable ordering**
   - The megaflow subtable ordering is based on access frequency
   - Subtable latency can leak traffic rate

Attack Model

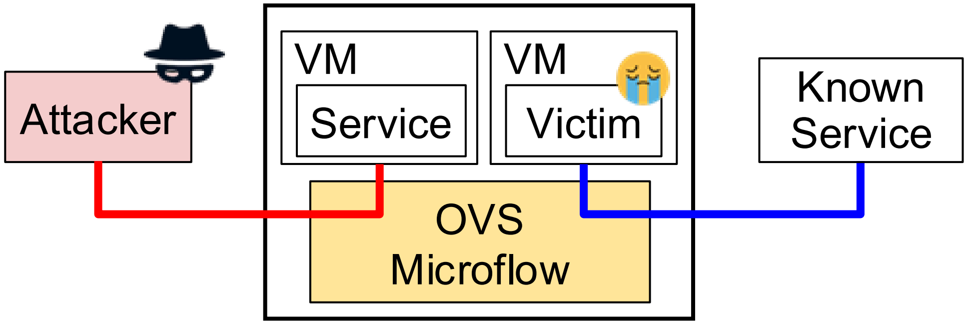
- Assume a cloud environment
  - Attacker and victim have no direct co-location
  - Attacker may access services co-located with the victim's server, but are isolated by VMs
- Only OVS is shared



Remote Packet Header Recovery Attack

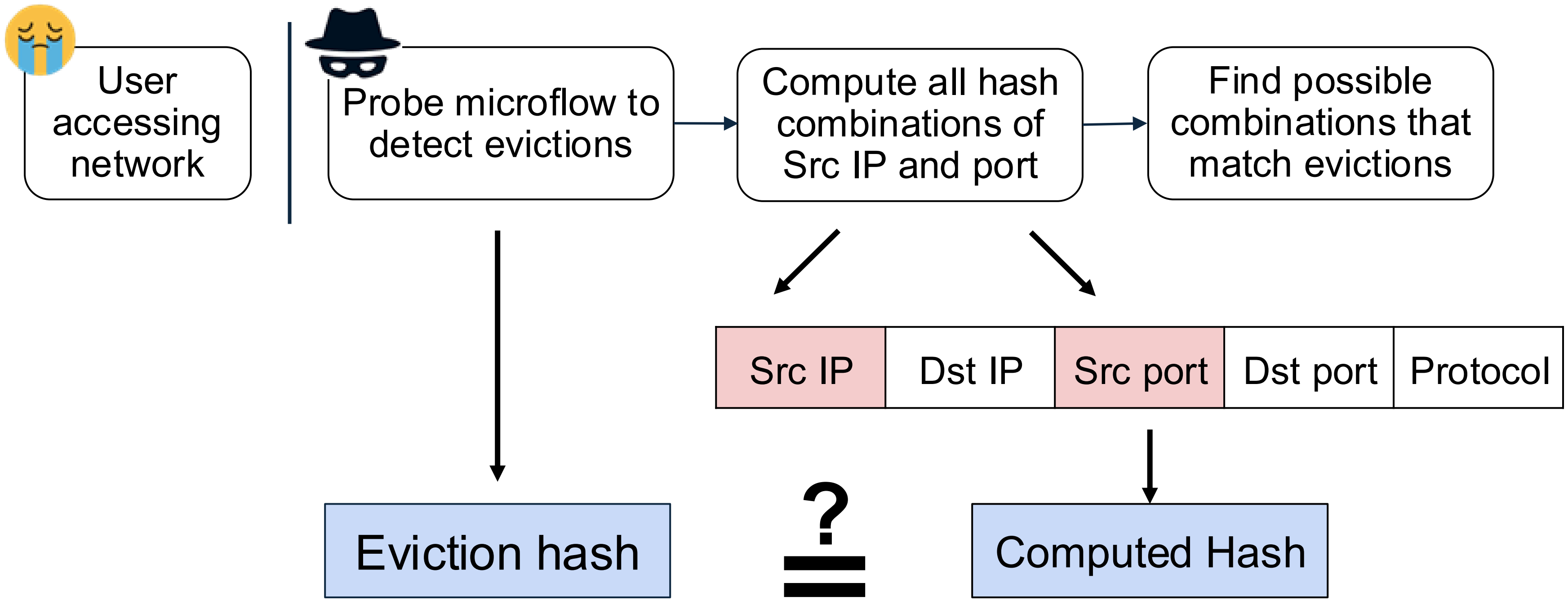
- Attacker knows victim is accessing some well-known service
  - Victim's Dst IP, port, and protocol are known
- Infer victim's remaining header fields
  - Victim's Src IP and port are targets
- Use Attack Primitives 1 & 2 to detect microflow collisions

Infer src IP and port



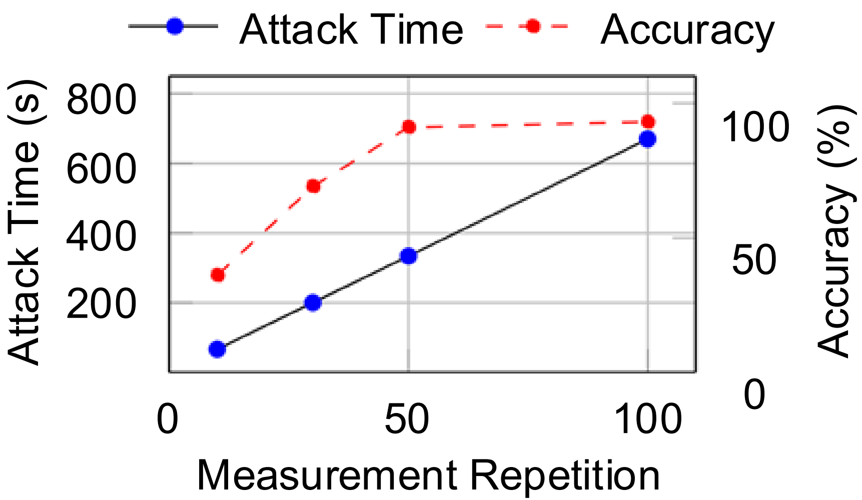
Known dst IP, port, and protocol

Remote Packet Header Recovery Attack



Recovery Accuracy and Time

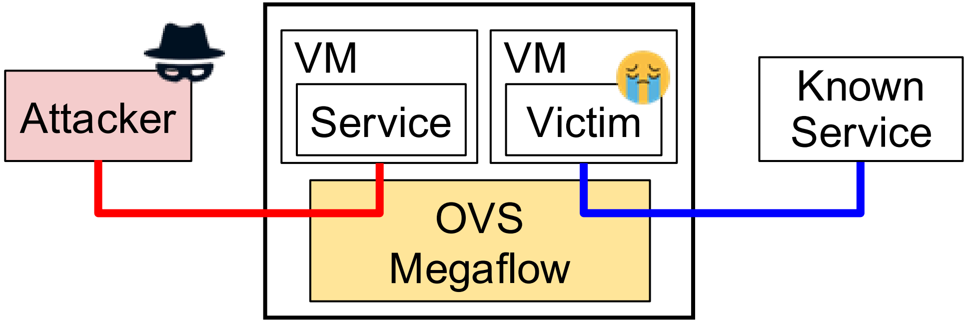
- Measurement repetition increases recovery accuracy
- Probing time increases as attacker repeats probing



Packet header fields can be recovered by probing active entries in microflow cache

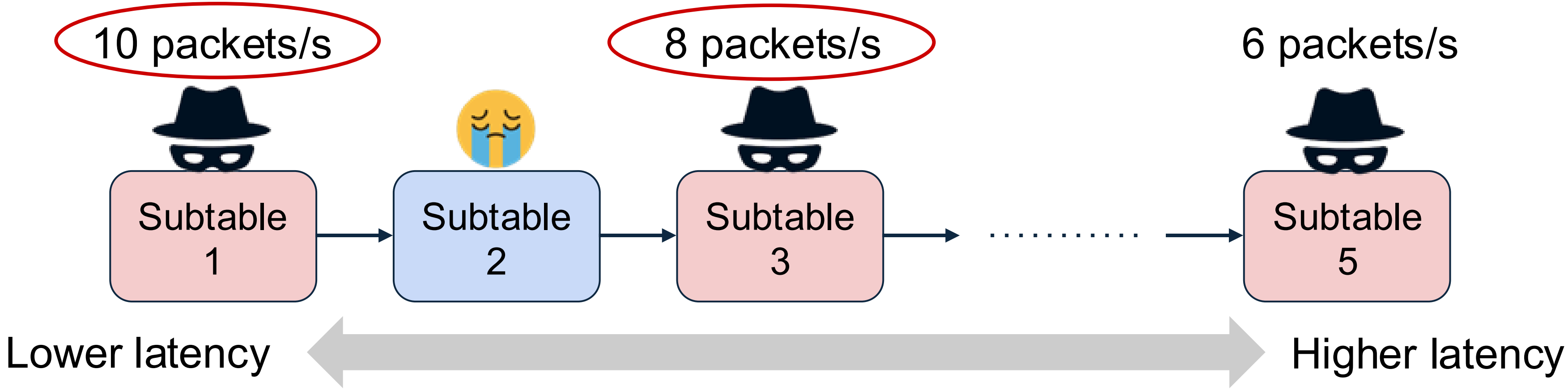
Remote Packet Rate Monitoring Attack

- Attacker has knowledge about victim's packet header
  - Using packet header recovery attack
- Attacker can locate victim flow's subtable and probe it
- Use Attack Primitive 3 to monitor victim's packet rate



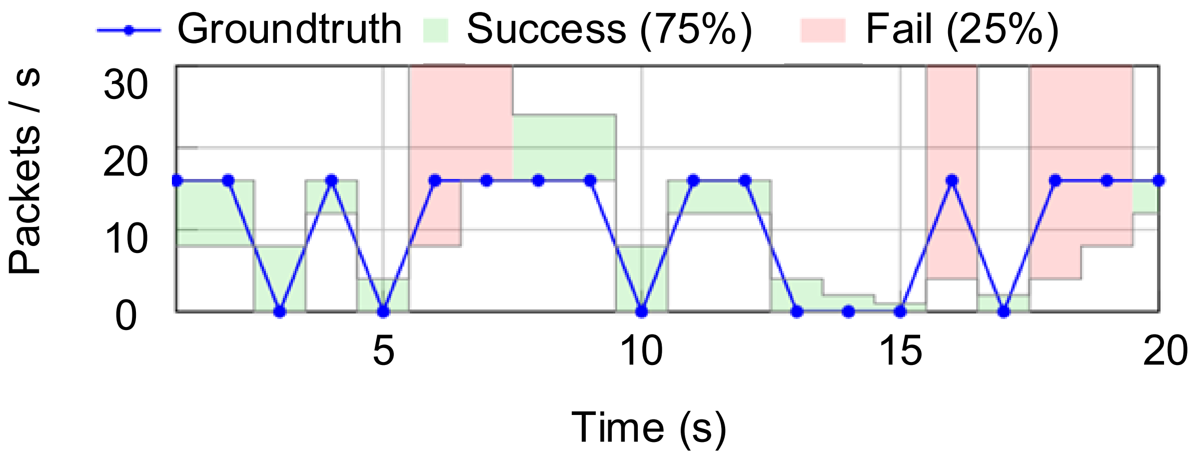
Remote Packet Rate Monitoring Attack

- Attacker measures latency to victim's co-located subtable
- Attacker accesses certain subtables that work as "thresholds"
  - Send packets at fixed rates
  - Compare the victim's subtable latency and determine the relative ordering
- Use neighboring thresholds to identify victim's packet rate



Recovery Accuracy

- Replay packets based on timestamps in UNSW-NB15
- Recovery of an example flow:



Attacker can monitor packet rate at 71.9% accuracy on average

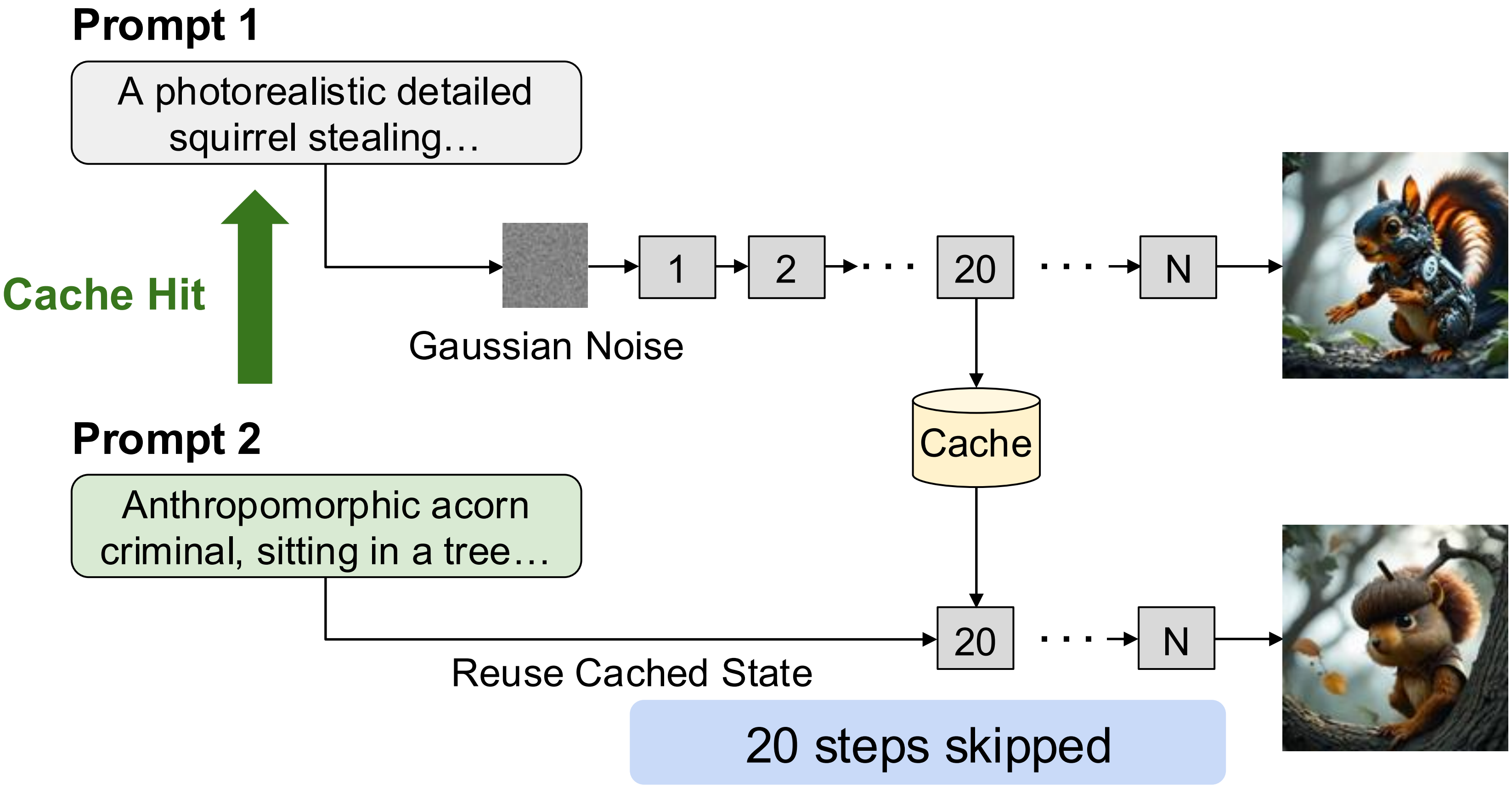
Case Study II
Side Channels in Prompt Cache

Prompt Caching in Gen AI

- Gen AI models cache prompts for faster generation
- Examples of prompt caching:
  - **Large Language Models** can skip computation of identical components in the prompt by reusing the cache
  - **Text-to-Image Diffusion Models** reuse the intermediate states of cached prompts for similar prompts

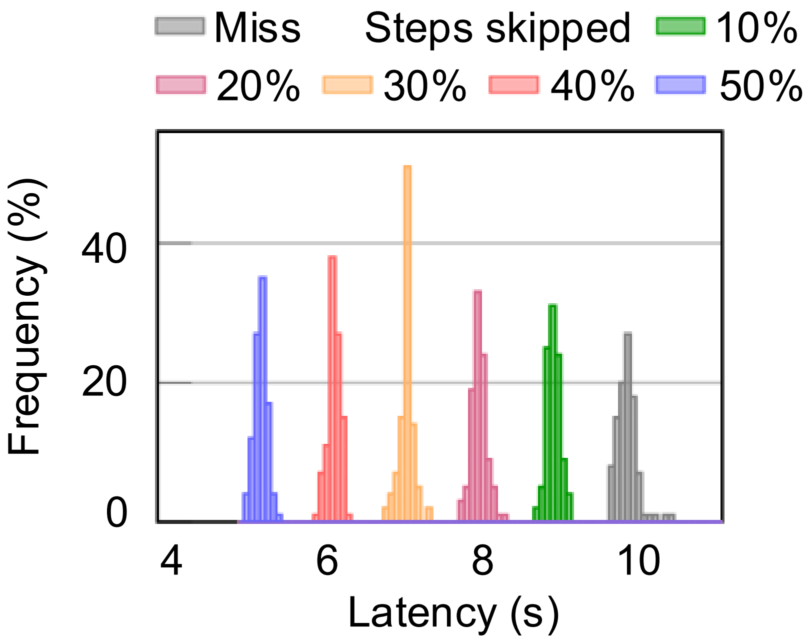
This study focuses on caching in Text-to-Image Diffusion Models

Caching for Text-to-Image Diffusion Models



Timing of Cached Generation

- Model: FLUX
- Platform: H100 GPU
- Evaluate generation time with different numbers of skipped steps



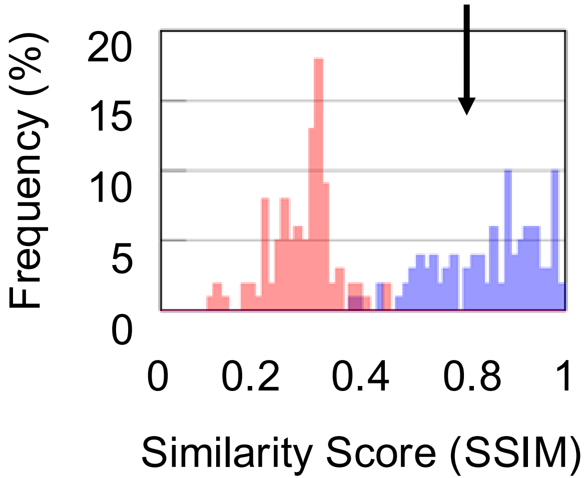
Caching reduces generation time, varying by the number of skipped steps

Similarity of Cached Generations

Original Cache

Cached Generation

Generated from Same Cache



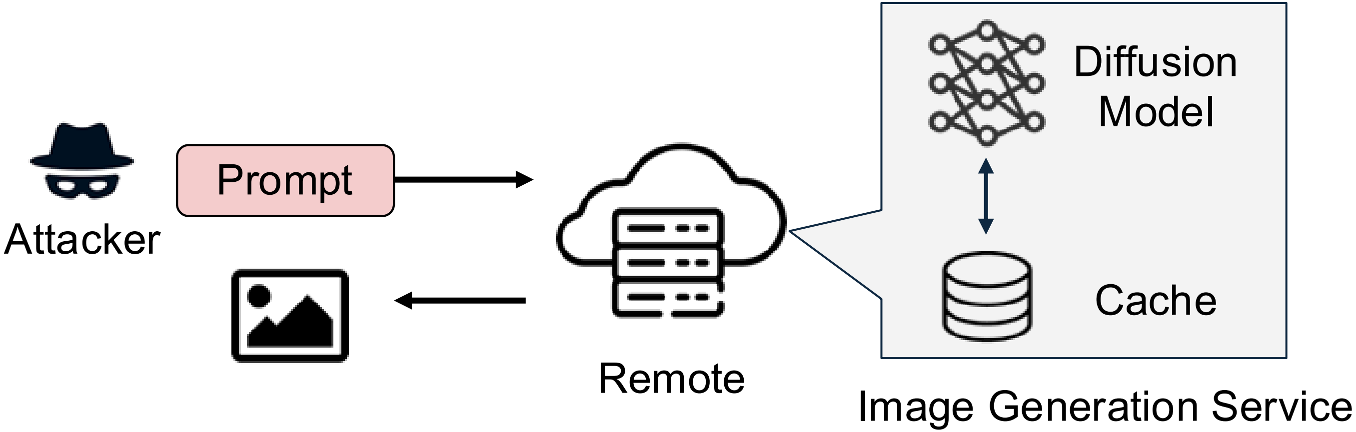
Images generated from the same cache are similar

Summary of Prompt Caching Attack Primitives

1. **Timing Differences**
   - A cache hit results in a significantly lower generation latency
   - An attacker can remotely determine if their prompt hit the cache
2. **Generation Similarity**
   - Images generated from the same cache share high similarities
   - An attacker can analyze the output image to determine if their prompts hit the same cache

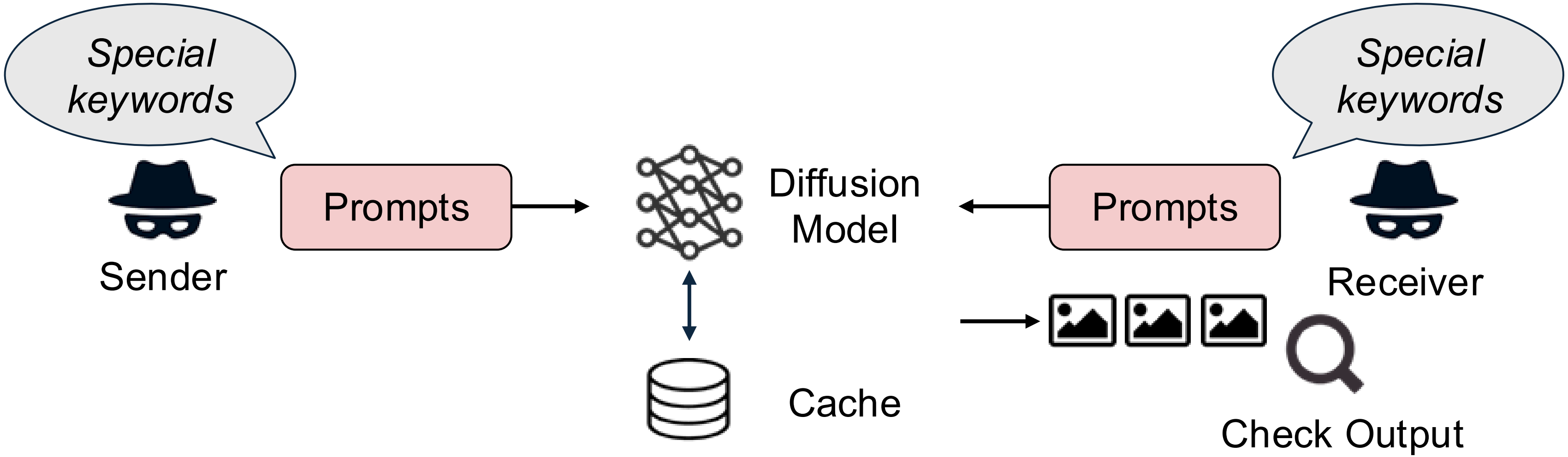
Attack Model

- The image generation service is hosted remotely in the cloud
  - It uses caching for acceleration
- Attackers can only access it remotely through generation prompts



Remote Covert Channel

- A sender and a receiver communicate stealthily through the cache
- Approach
  - Sender: Inserts prompts with special words into the cache
  - Receiver: Probes the cache and checks if special words exist



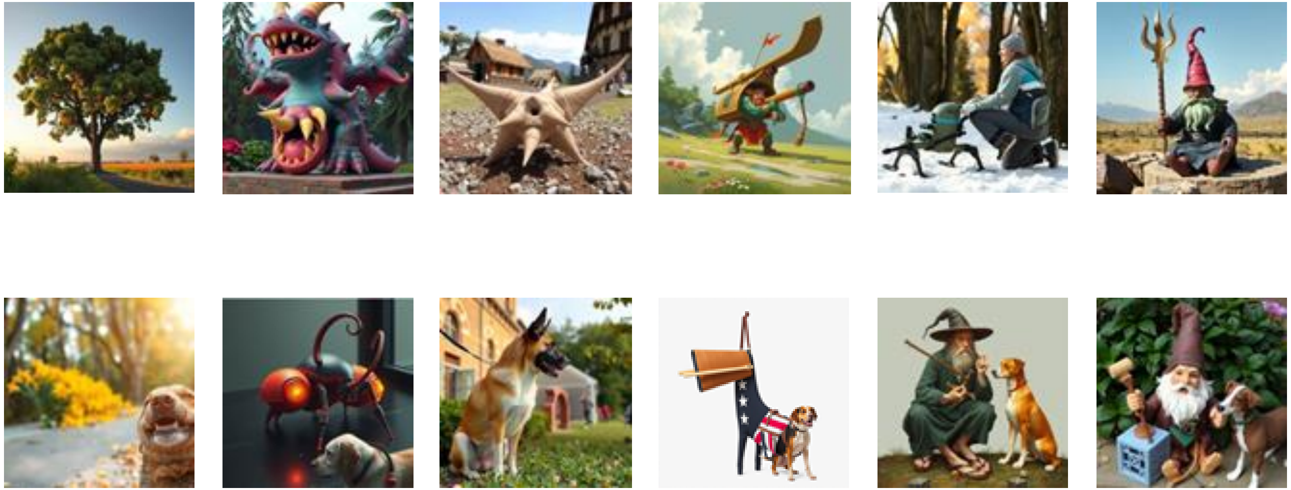
Remote Covert Channel Examples

Sender’s prompts remain cached for ~2 days

Keywords	Apricity	Cacodemon	Caltrop	Crwth	Fleam	Gnomon

Initial Keyword

Keyword Hit

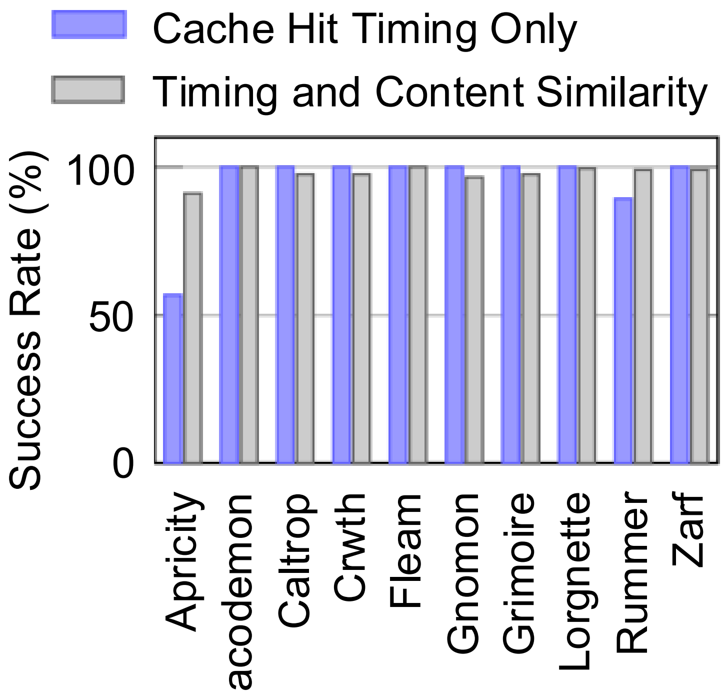


Receiver uses an objective detection model to check the output

Covert Channel Accuracy

- Model: Flux
- Dataset: DiffusionDB
- Baseline: Using cache timing to detect if the sender has injected a keyword

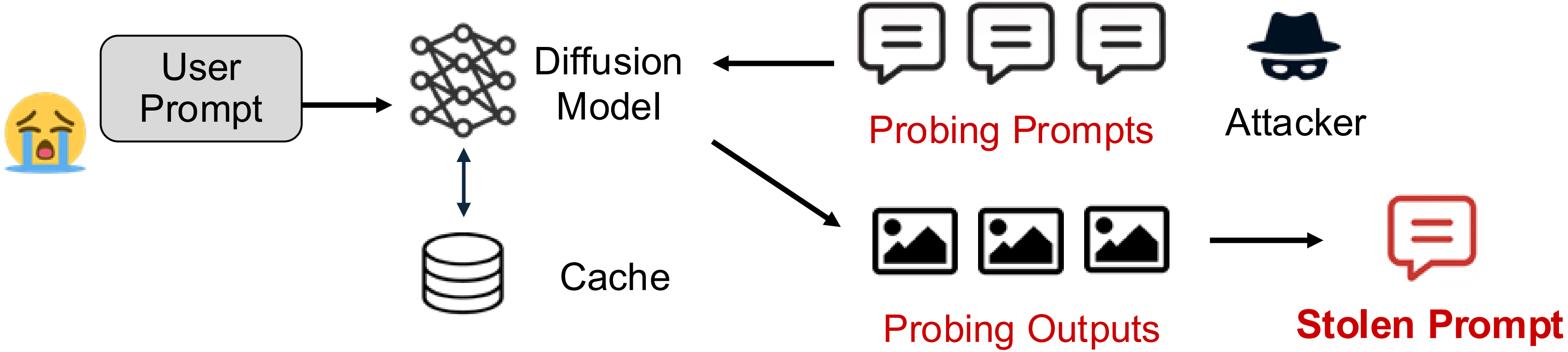
- Accuracy:
  - **Cache timing only: 95%**
  - **Cache timing + content similarity: 98%**



Using both timing and content similarity attack primitives achieves high covert channel accuracy

Prompt Stealing

- Infer user prompts in the cache through cache hits
- Attacker's approach
  - Craft and probe the cache
  - Classify prompts that hit the same cache
  - Use a language model to recover the cached prompt



Prompt Stealing Example

User's Prompt:

Conceptual art of a medieval knight with angel wings in a forest at night, realistic painting, classical painting, high definition, digital art, matte painting, very detailed, realistic …

Stolen Prompt:

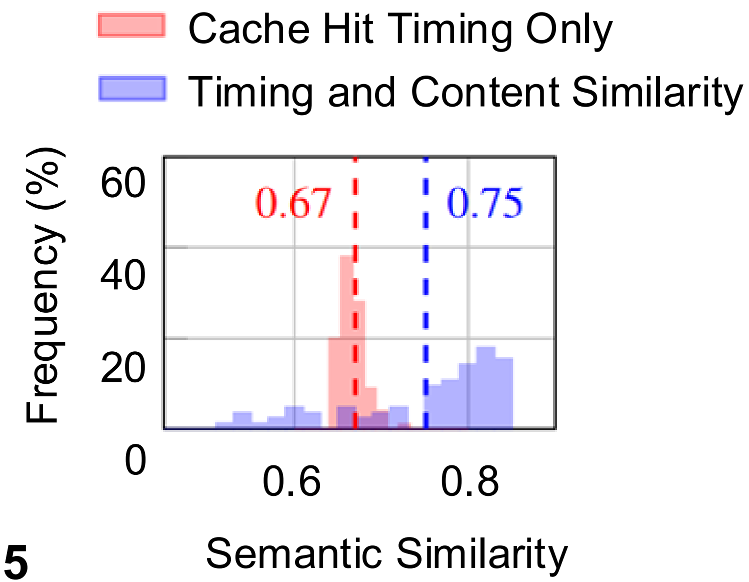
Medieval knight in the forest, highly detailed body, knight in armor made of wood, elden ring inspired, photo-realistic painting, digital art, matte painting, from a classical oil painting …

Prompt semantic similarity: 0.85

Prompt Stealing Accuracy

- Model: Flux
- Dataset: DiffusionDB
- Baseline: Using timing to determine hits, without classifying if prompts hit the same cache

- Accuracy (semantic similarity):
  - **Cache timing only: 0.67**
  - **Cache timing + content similarity: 0.75**



Attacker recovers prompt with high similarity

Summary

- The wide use of cloud and AI introduces a wider attack surface
- We demonstrate two cases of side-channel attacks
  - Open vSwitch can leak user data due to its caching mechanism
  - Prompt caching can allow attackers to steal user prompts and transmit secret messages
- There is an urgent need for mitigating such leakages to ensure trustworthy AI and cloud systems
- Our future direction aims to mitigate side-channel vulnerabilities in cloud and AI applications

Side-Channel Vulnerabilities in Networking and AI Systems

Sihang Liu

University of Waterloo

Oct 28, 2025

UNIVERSITY OF WATERLOO