UNIVERSITY OF CALIFORNIA
UC RIVERSIDE

Unveiling your keystrokes: A Cache-based Side-channel Attack on Graphics Libraries

Daimeng Wang, Ajaya Neupane, Zhiyun Qian,
Nael Abu—Ghazaleh, Srikanth V. Krishnamurthy,
Edward J. M. Colbert†, Paul Yu‡

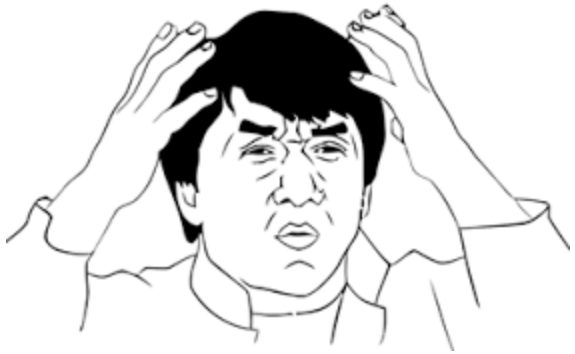
University of California Riverside, Virginia Tech†,
U. S. Army Research Lab‡

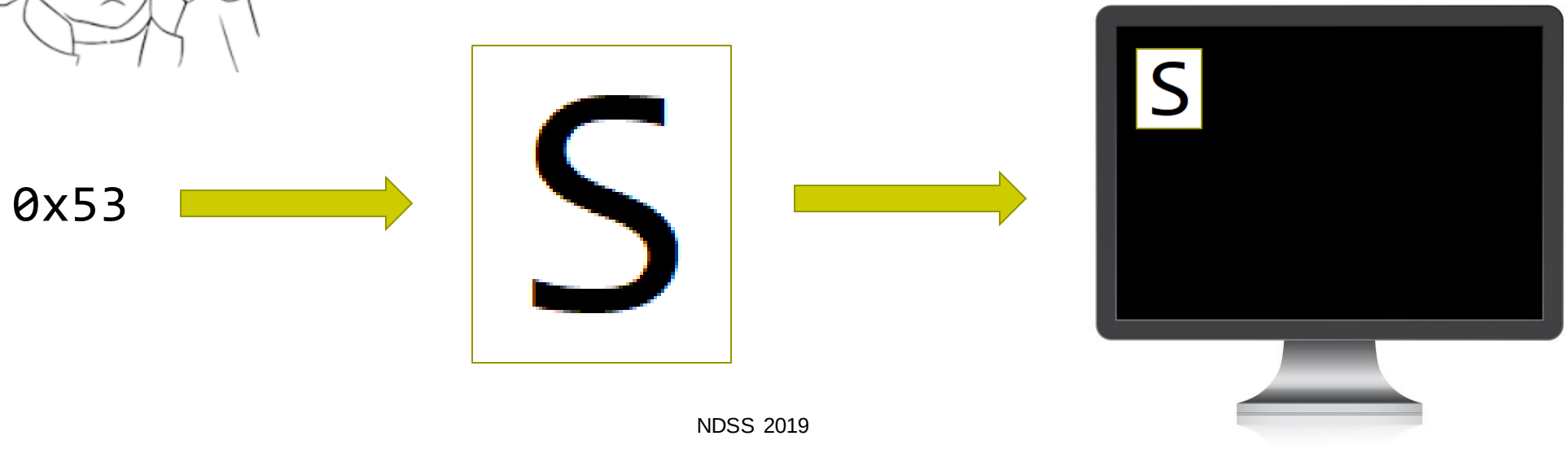
Introduction

- Graphics is essential
- Graphic rendering is complex
  - gdk, gtk, pixman, freetype, cairo, skia, hwui, …

Introduction

- Graphics is essential
- Graphic rendering is complex
  - gdk, gtk, pixman, freetype, cairo, skia, hwui, …





Example

```
static void D32_LCD32_Opaque(...) {
    ...
    do {
        blit_lcd32_opaque_row(dstRow, srcRow, color, width);
        dstRow = (SkPMColor*)((char*)dstRow + dstRB);
        srcRow = (const SkPMColor*)((const char*)srcRow + maskRB);
    } while (--height != 0);
}

static void blit_lcd32_opaque_row(dst, src, color, width) {
    ...
    for (int i = 0; i < width; i++) {
        if (0 == src[i]) {
            continue;
        }
        ...
    }
}
```

Example

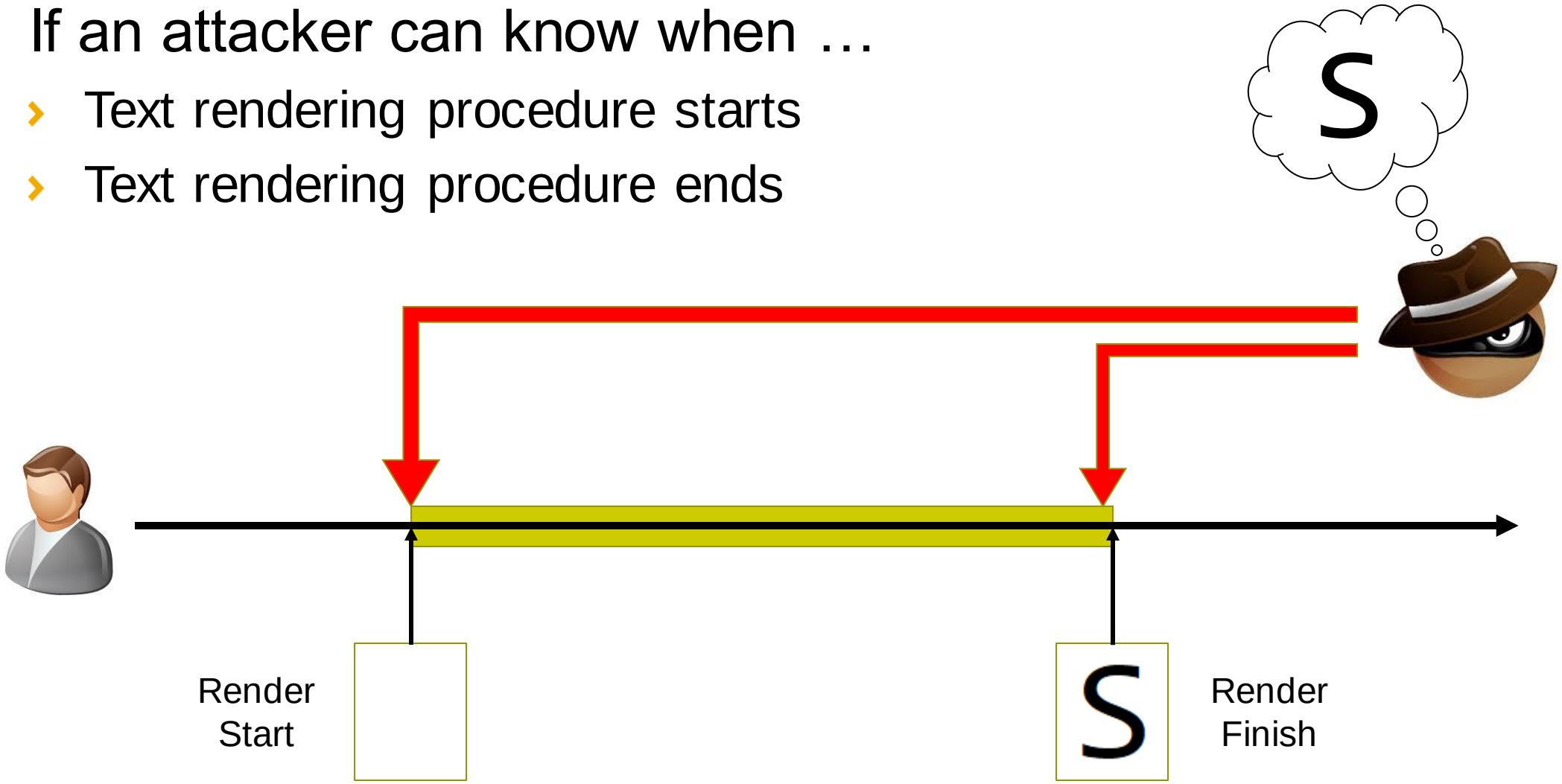
```
static void D32_LCD32_Opaque(...) {
    ...
    do {
        blit_lcd32_opaque_row(dstRow, srcRow, color, width);
        dstRow = (SkPMColor*)((char*)dstRow + dstRB);
        srcRow = (const SkPMColor*)((const char*)srcRow + maskRB);
    } while (--height != 0);
}

static void blit_lcd32_opaque_row(dst, src, color, width) {
    ...
    for (int i = 0; i < width; i++) {
        if (0 == src[i]) {
            continue;
        }
        ...
    }
}
```

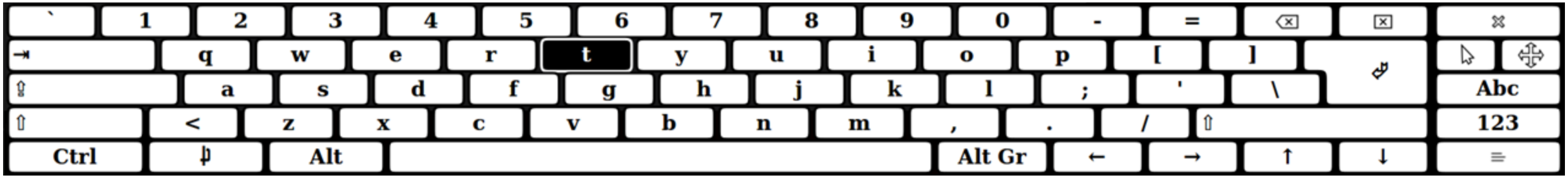
NOT CONSTANT TIME!

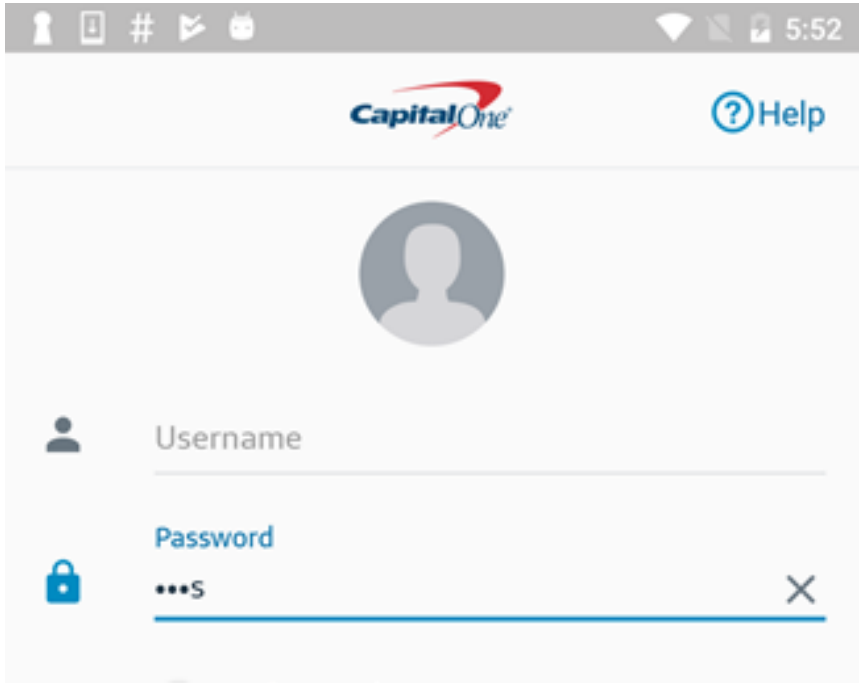
Exploiting the Side-channel

- If an attacker can know when …
  - Text rendering procedure starts
  - Text rendering procedure ends



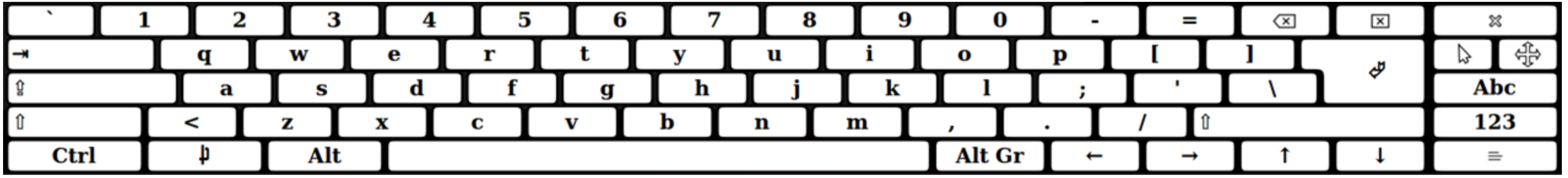
Potentially Vulnerable Apps





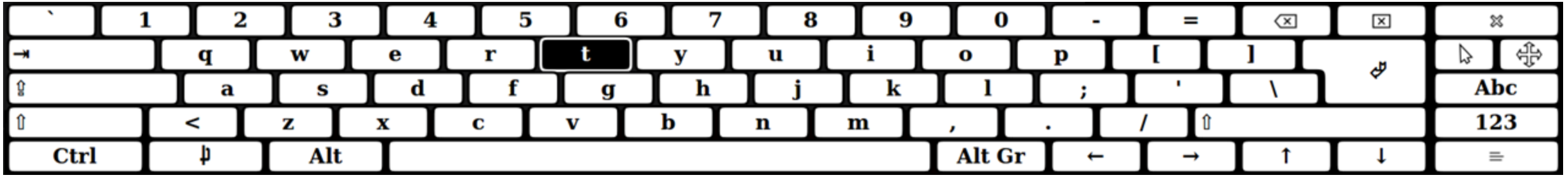
Example: Onboard

- Onscreen keyboard
  - Ubuntu 16.04

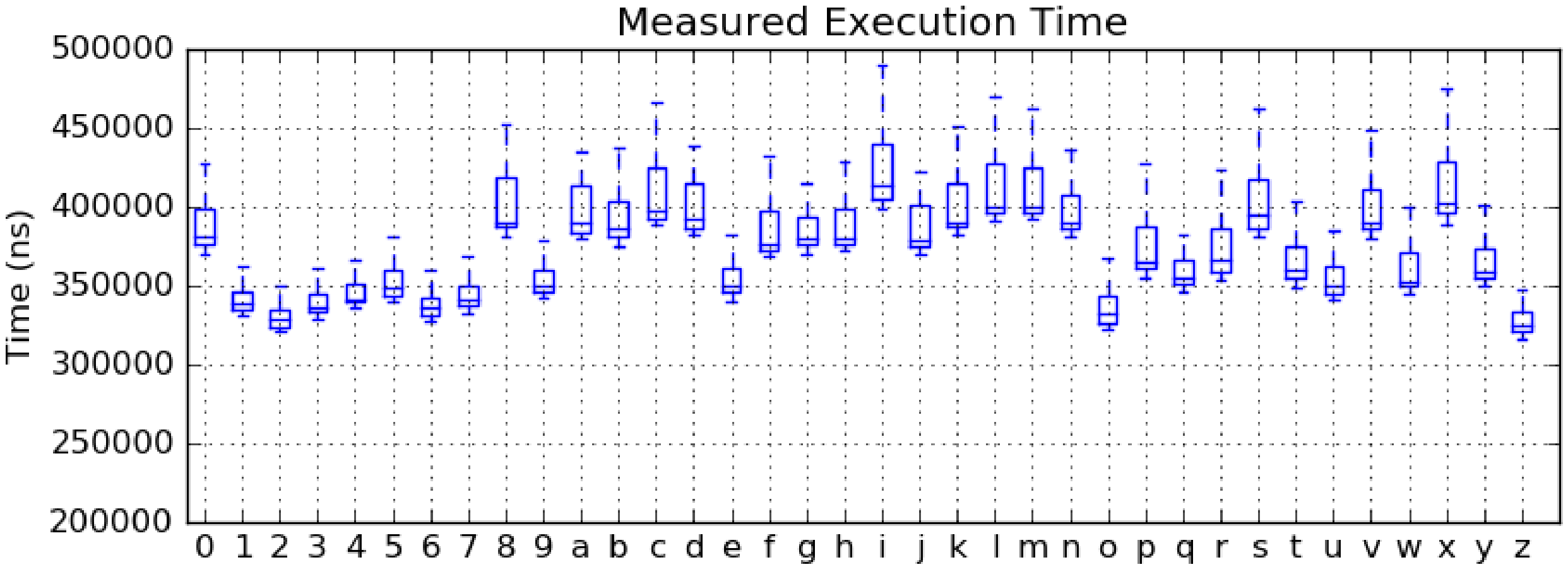


Example: Onboard

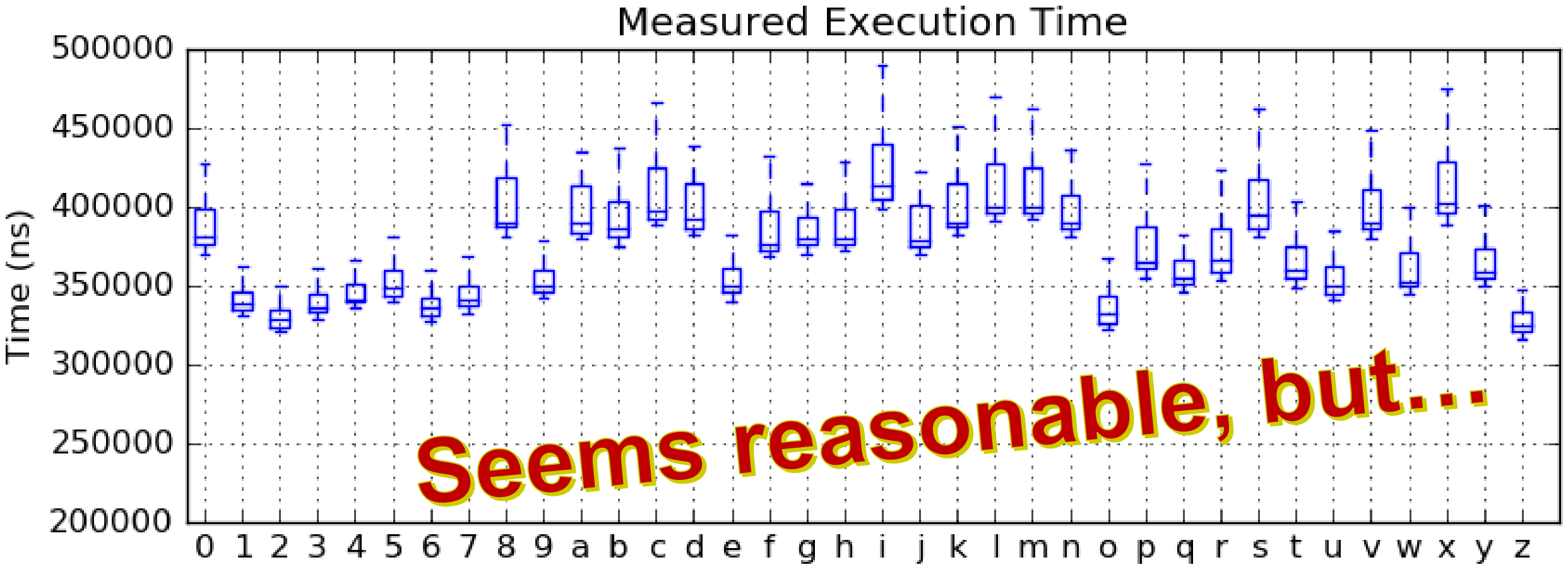
- Onscreen keyboard
  - Ubuntu 16.04



Attack: Onboard

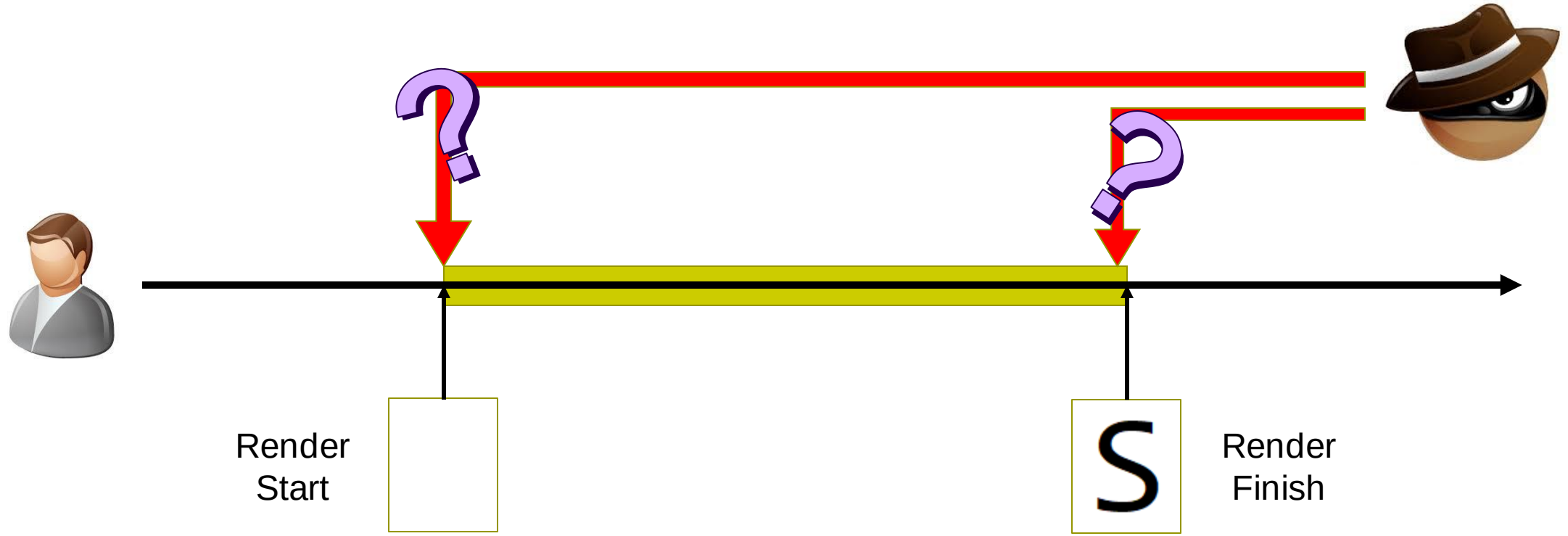


Attack: Onboard



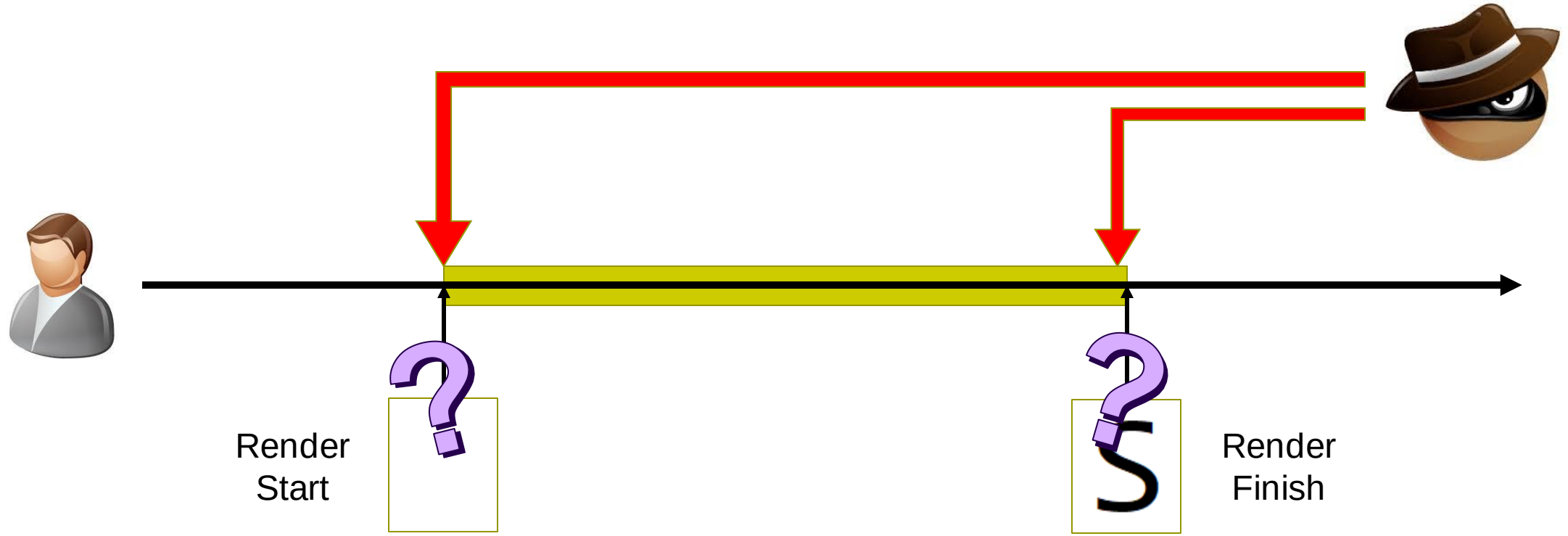
Challenges

- How to perform measurement?
  - Unprivileged attacker



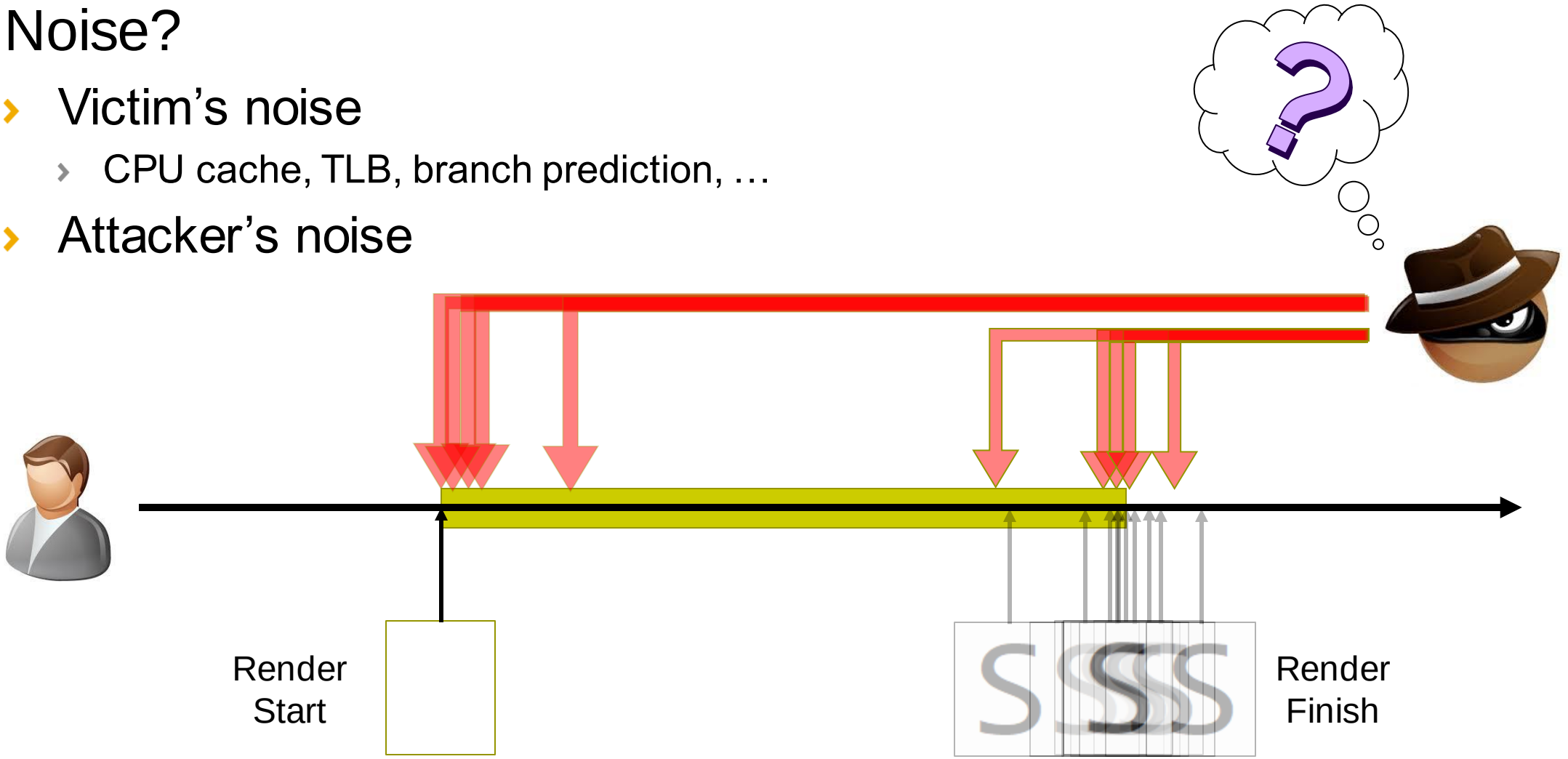
Challenges

- How to find start/end of rendering?
  - Millions of code. Multiple libraries.
  - Varies from application to application.



Challenges

- Noise?
  - Victim's noise
    - CPU cache, TLB, branch prediction, …
  - Attacker's noise



Challenges

- Perform measurement without privilege
- Discover side-channel in graphic libraries
- Noise-resistant key prediction

Threat Model

- Attacker's goal
  - Eavesdrop sensitive text input that will be rendered on screen
    - PIN, passwd, etc

- Attacker's capabilities
  - Access to same model/version of victim's hardware and graphic libraries
    - Offline profiling
  - Launch unprivileged process alongside victim process
    - Online attack

Performing Measurement

…

libA.so

renderStart()

renderEnd()

…

…

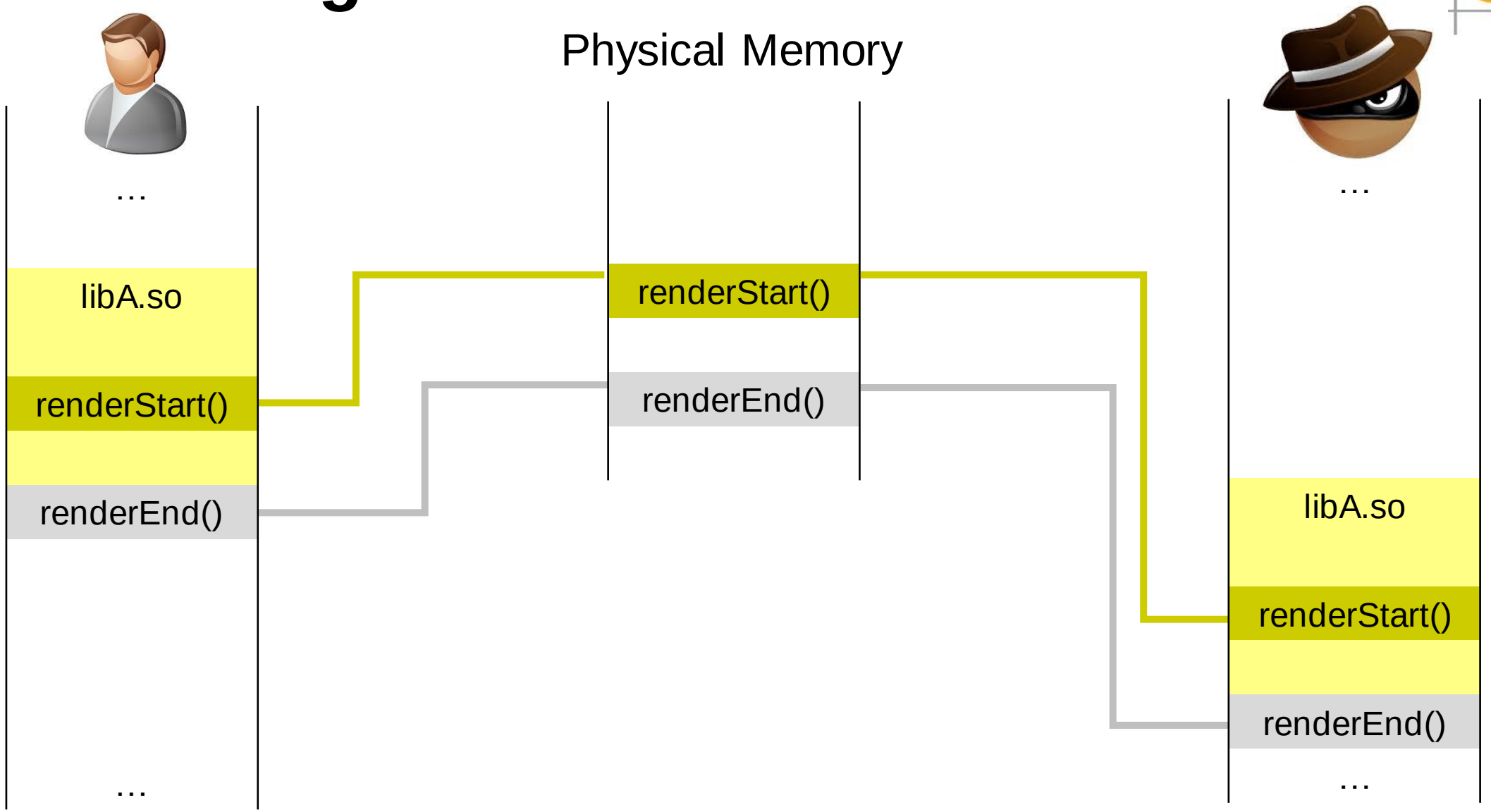
libA.so

renderStart()

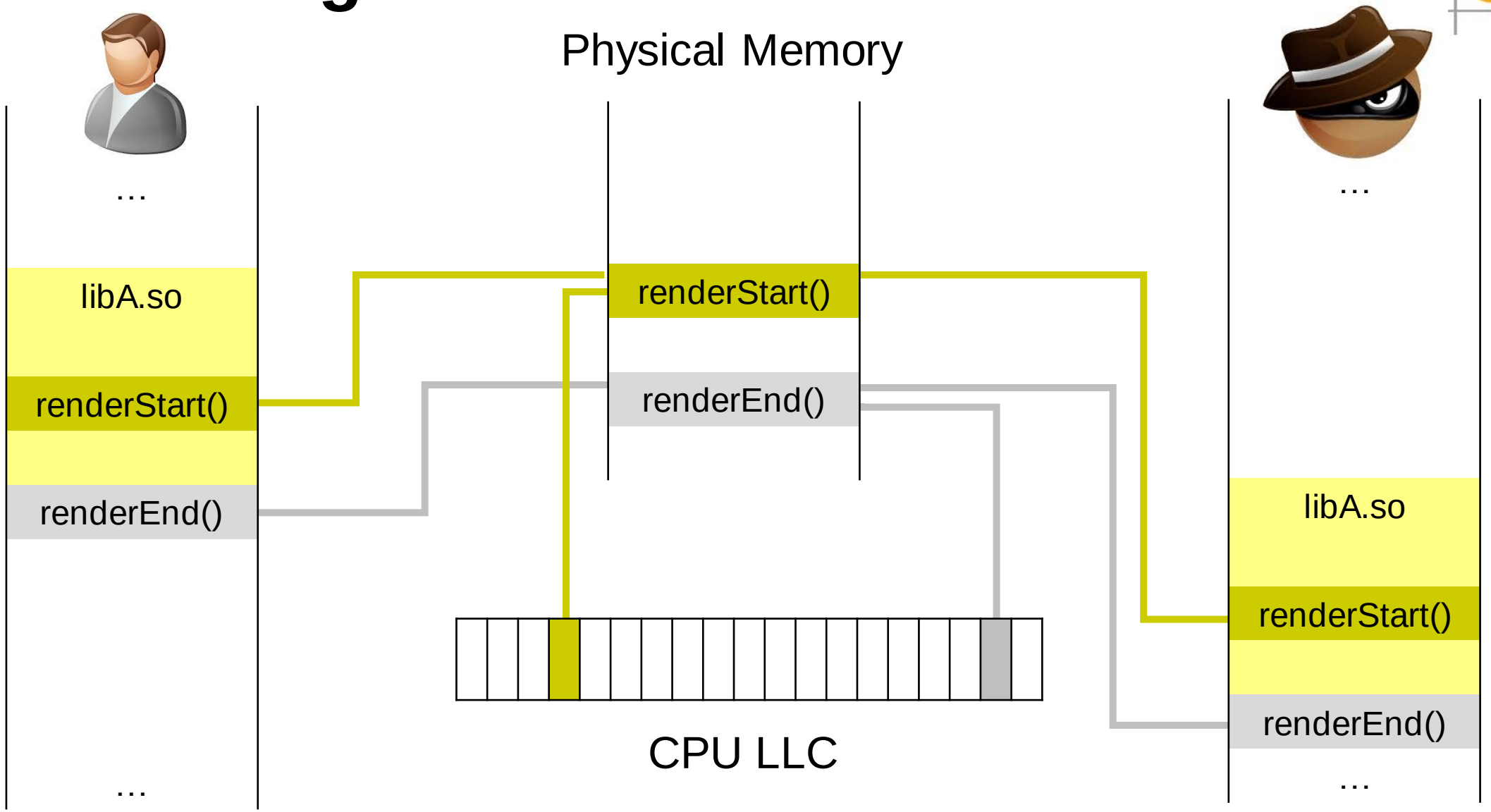
renderEnd()

…

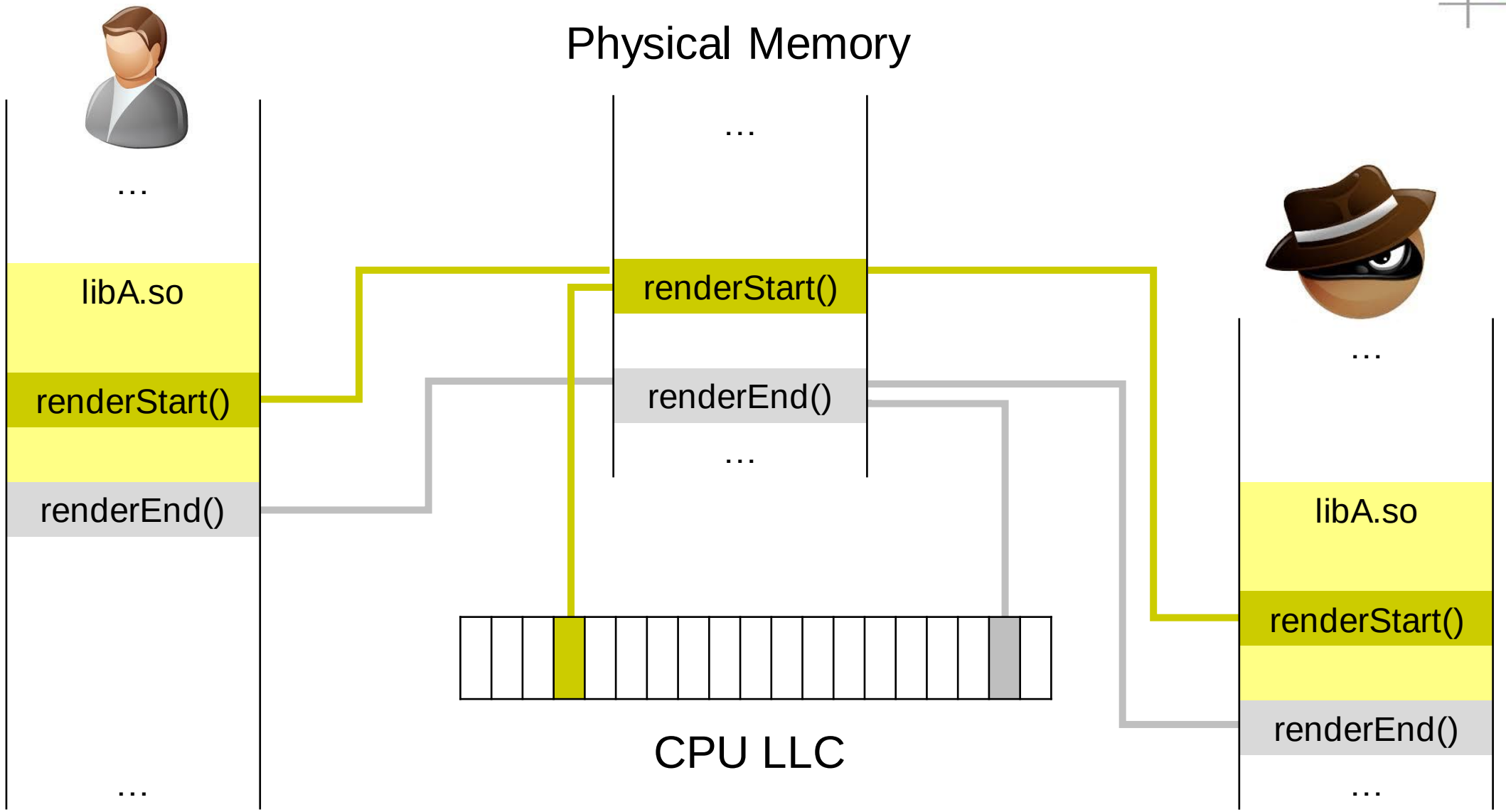
Performing Measurement



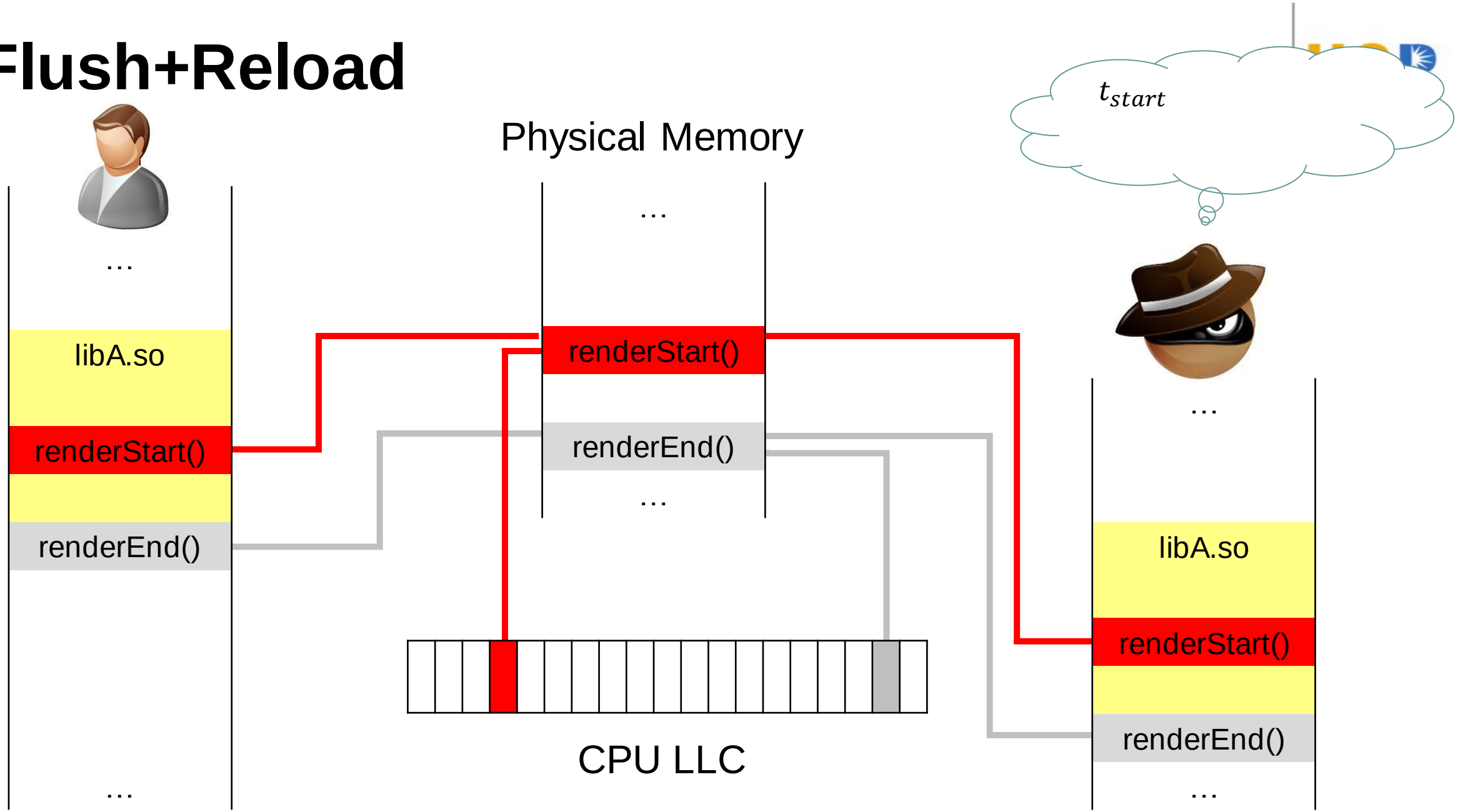
Performing Measurement



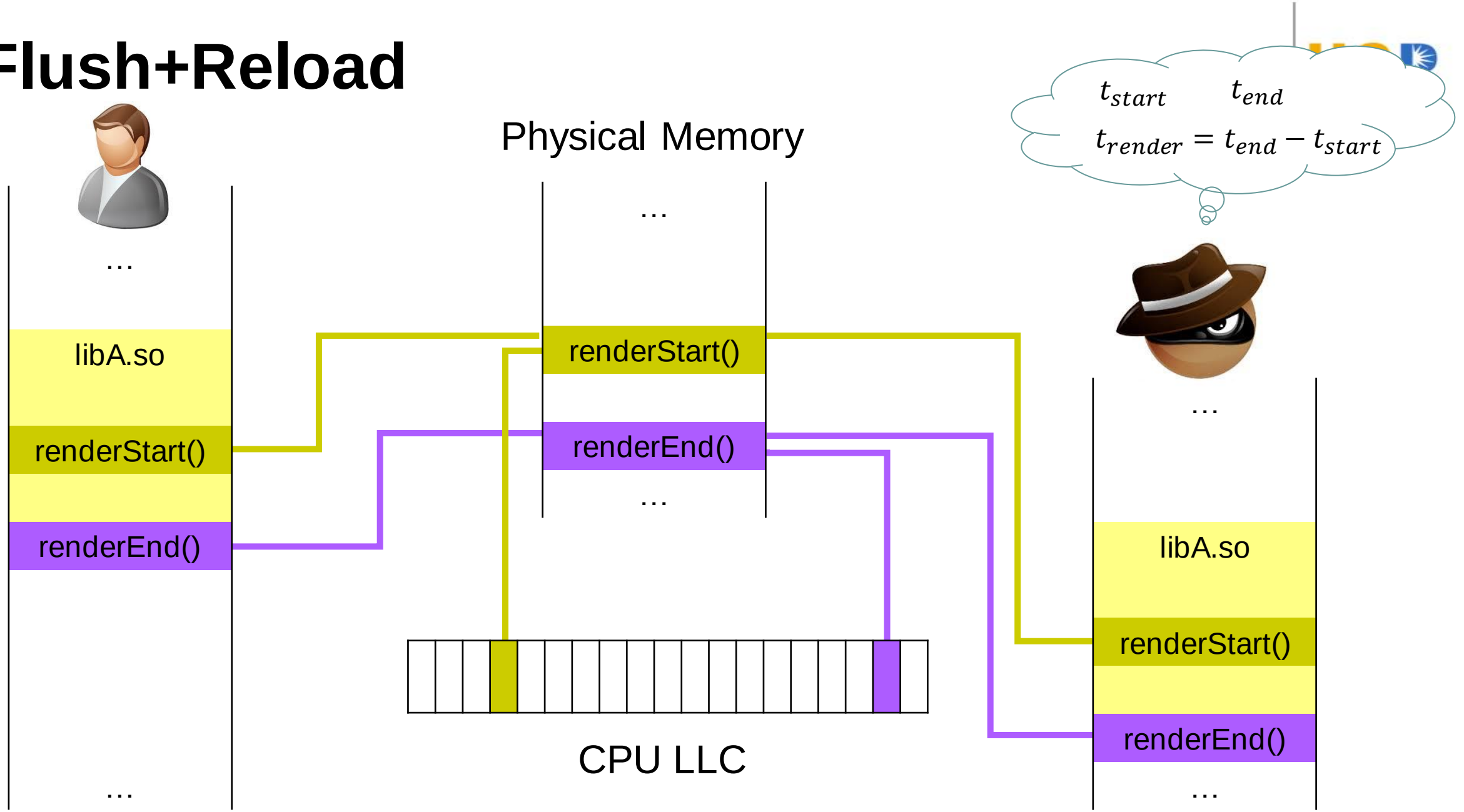
Flush+Reload



Flush+Reload

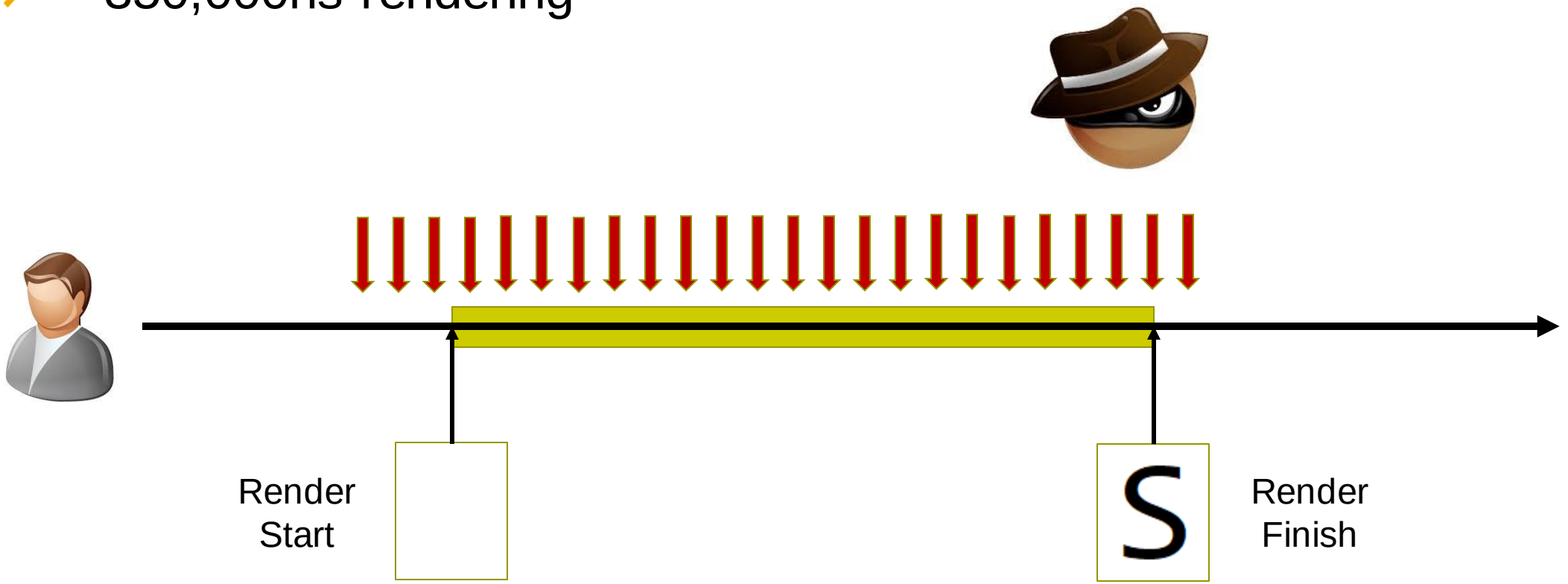


Flush+Reload



Measurement Resolution

- Onboard
  - ~ 600ns per round
  - ~ 350,000ns rendering

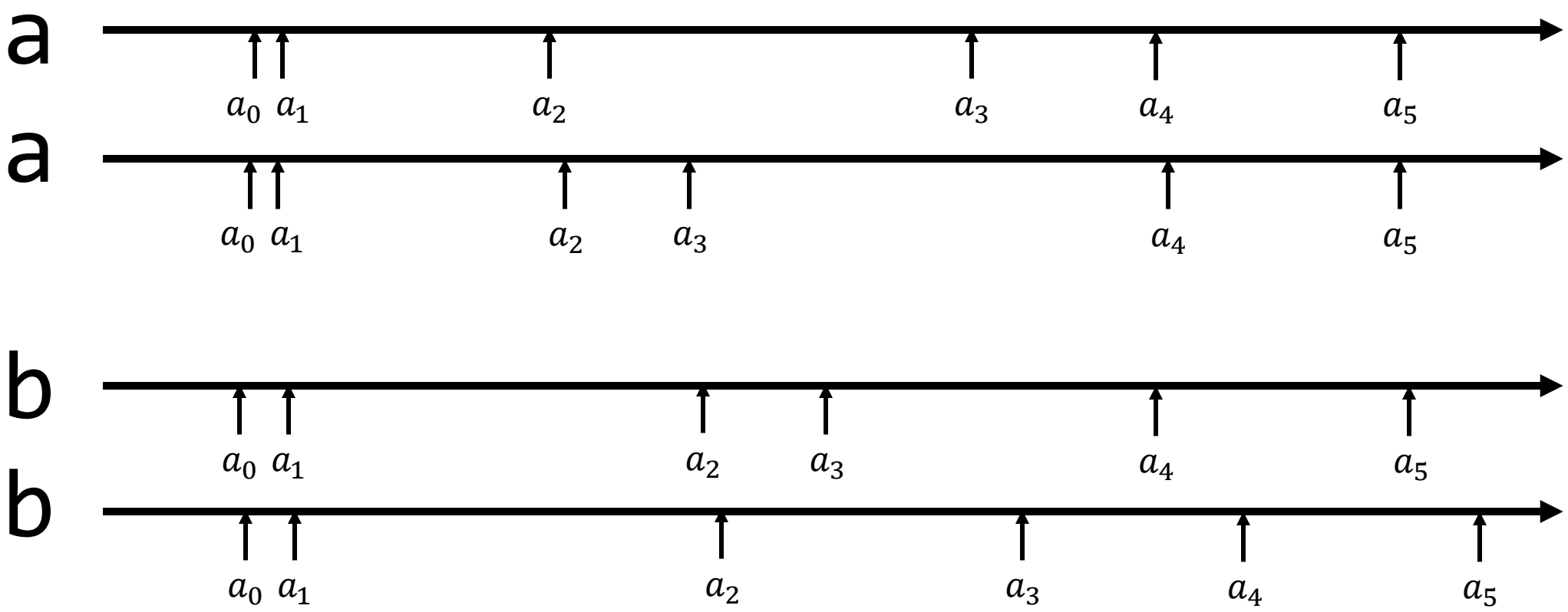


Side-channel Discovery

- Instrument graphic libraries & collect victim trace

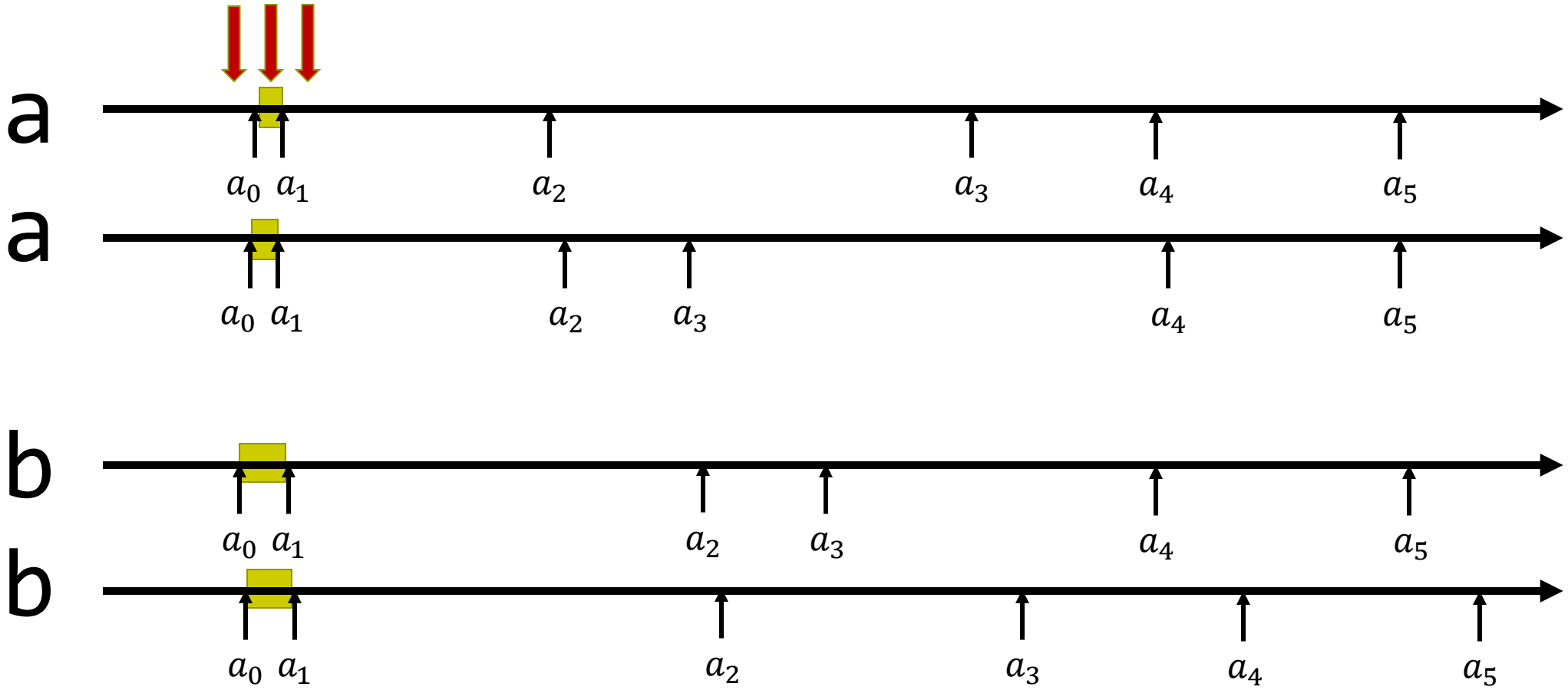
Side-channel Discovery

- Instrument graphic libraries & collect victim trace



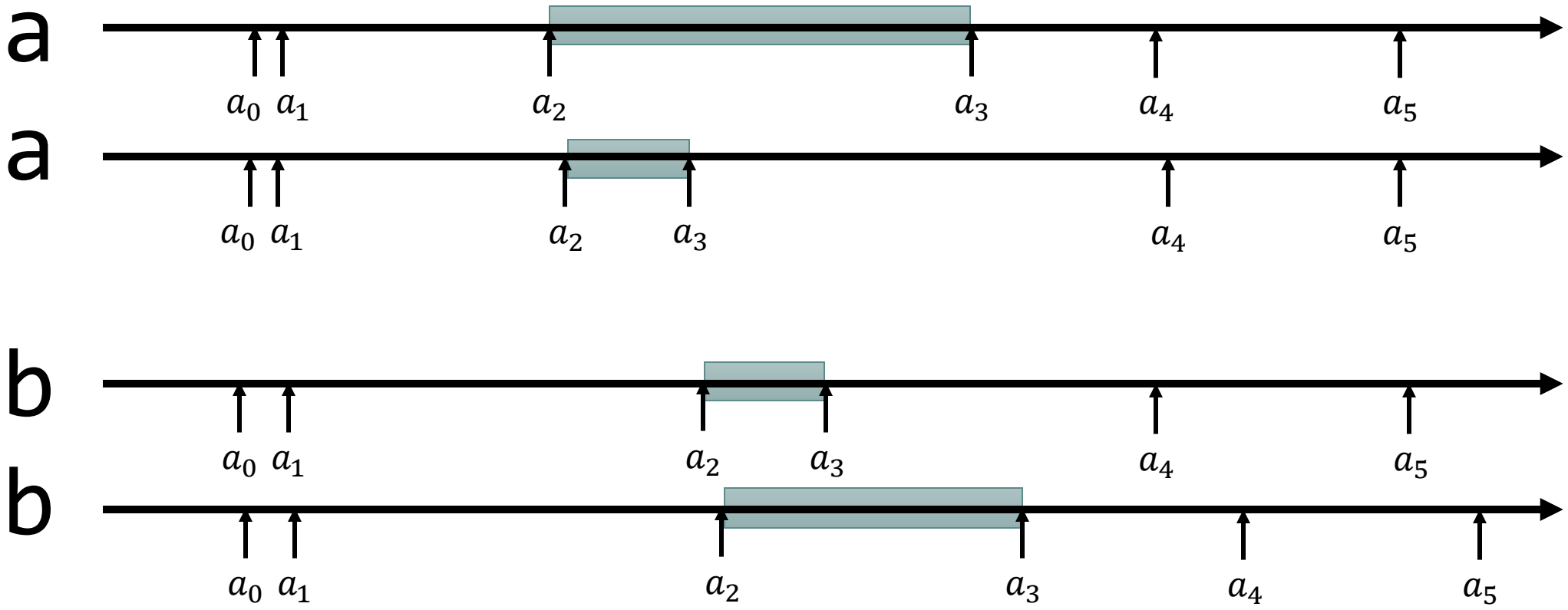
Side-channel Discovery

- Instrument graphic libraries & collect victim trace



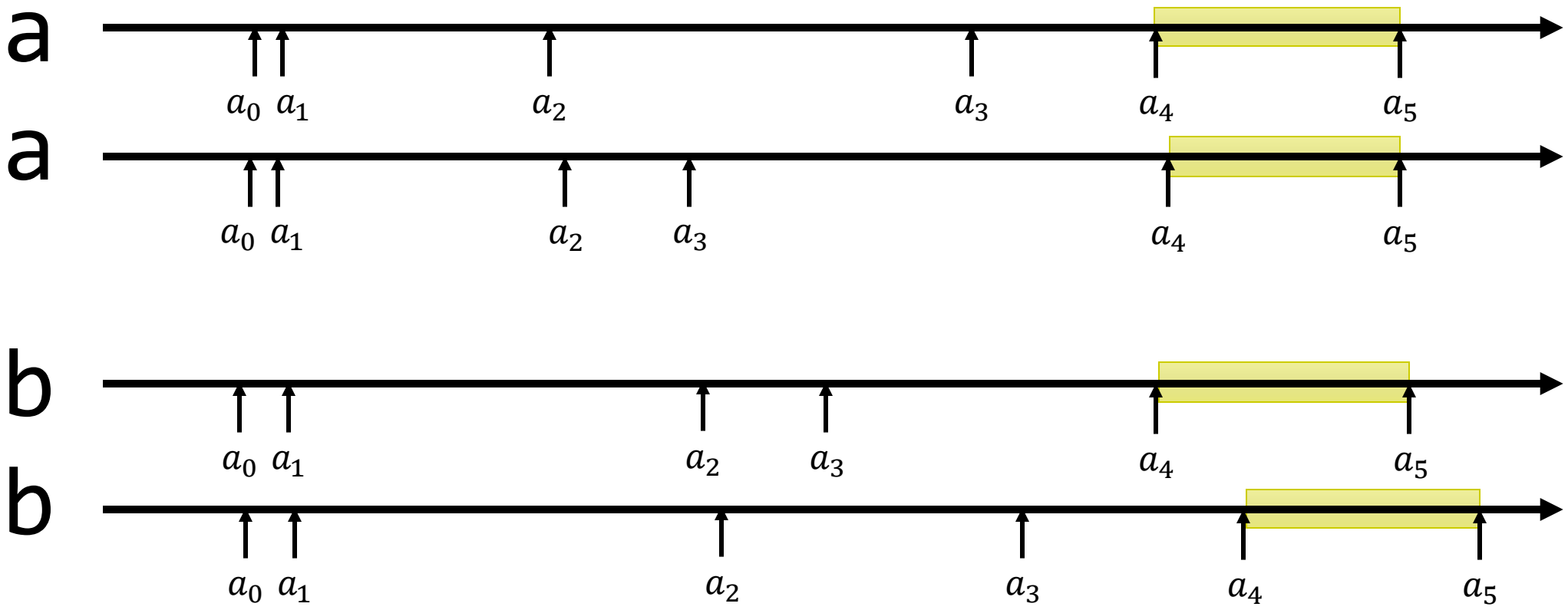
Side-channel Discovery

- Instrument graphic libraries & collect victim trace



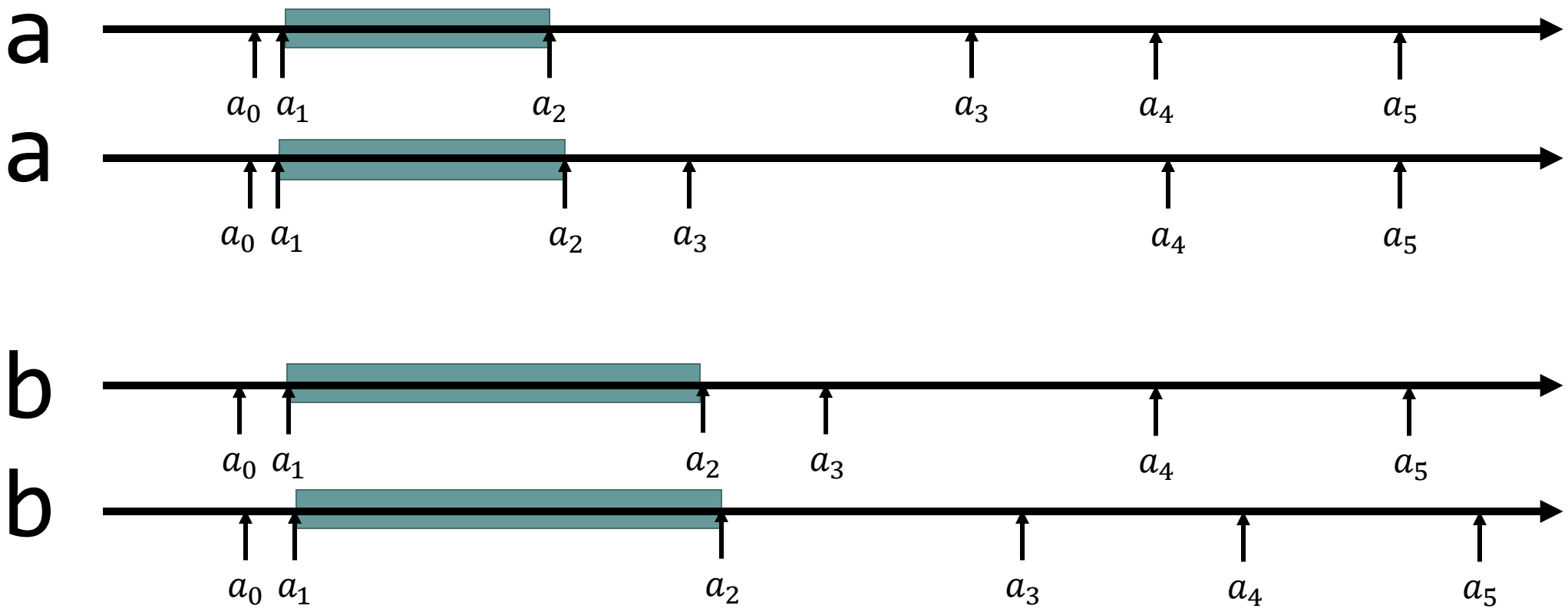
Side-channel Discovery

- Instrument graphic libraries & collect victim trace



Side-channel Discovery

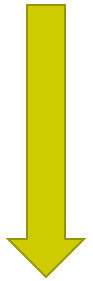
- Instrument graphic libraries & collect victim trace



Side-channel Discovery

- Select pairs of addresses (a_x, a_y)
  - $dist(a_x, a_y)$ is long enough for flush+reload to measure
  - $dist(a_x, a_y)$ is stable across the same input
  - $dist(a_x, a_y)$ has high information gain
  - a_x, a_y not affected by cache architecture

> 6 million pairs

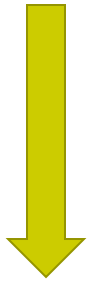


~1000 pairs

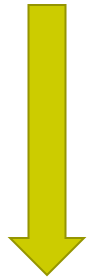
Side-channel Discovery

- Select pairs of addresses (a_x, a_y)
  - $dist(a_x, a_y)$ is long enough for flush+reload to measure
  - $dist(a_x, a_y)$ is stable across the same input
  - $dist(a_x, a_y)$ has high information gain
  - a_x, a_y not affected by cache architecture
- Sanity Check: Run attack offline
  - Filter out addressed affected system noise
  - Filter out addressed affected by instrumentation

> 6 million pairs



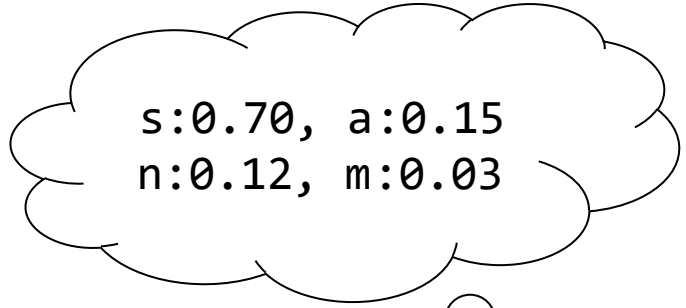
~1000 pairs



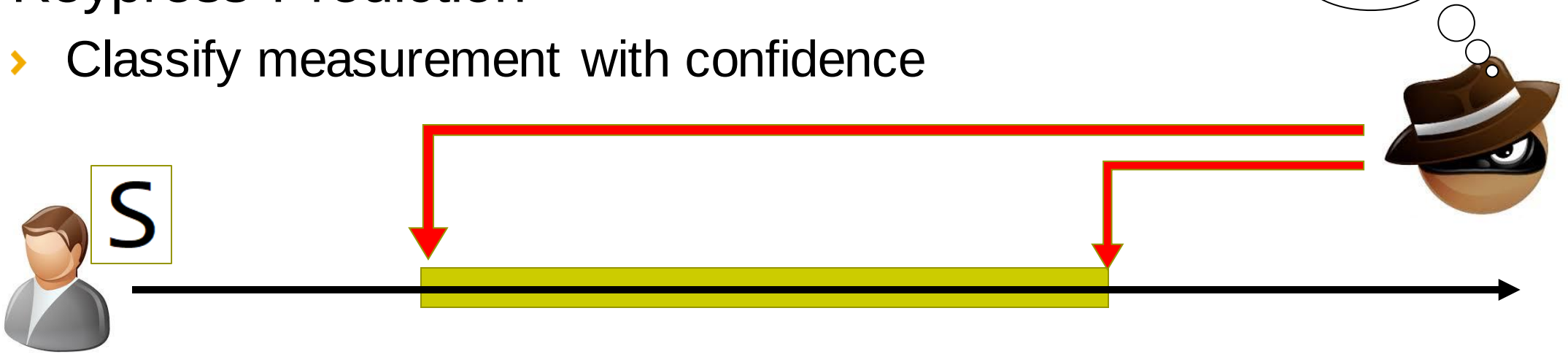
<100 pairs

Keypress Prediction

- Machine Learning Model Construction
  - Collect measurements for all different user inputs
  - Random Forest with 100 estimators



- Keypress Prediction
  - Classify measurement with confidence



Challenges

- ✓ Perform measurement without privilege
- ✓ Discover side-channel in graphic libraries
- ✓ Noise-resistant key prediction

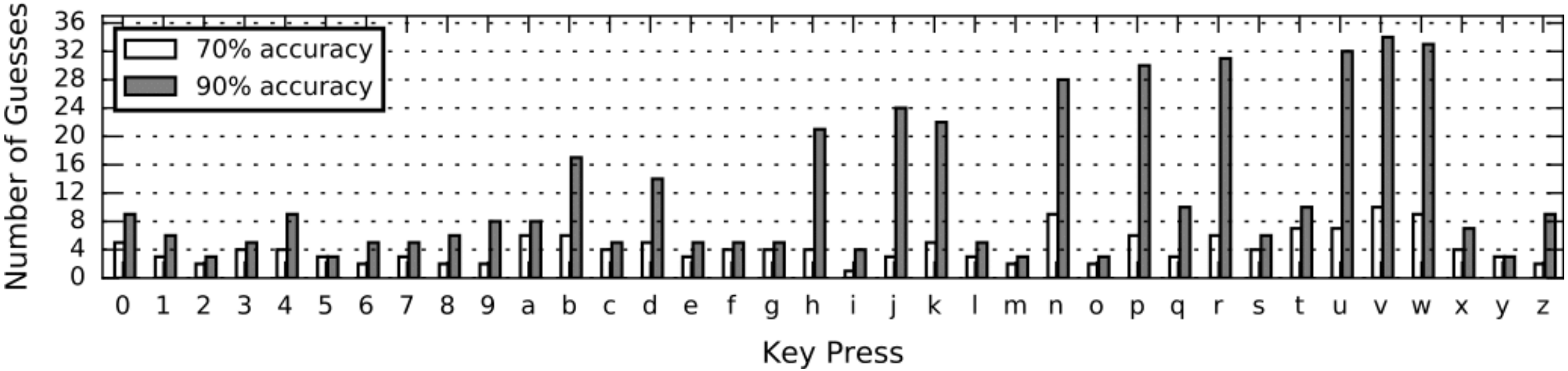
Attack 1: Onboard

- Ubuntu 16.04, Intel Core i7-4770

#	Address	Library	Function
1	0x75a40	libcairo.so	_cairo_surface_create_scratch
	0x69e40	libcairo.so	_cairo_scaled_font_map_lock
2	0x69e40	libcairo.so	_cairo_scaled_font_map_lock
	0x41f40	libcairo.so	_cairo_intern_string
3	0x24440	libcairo.so	_cairo_clip_copy_with_translation
	0xbe000	libcairo.so	_cairo_ft_unscaled_font_lock_face
4	0x6b900	libcairo.so	_cairo_path_fixed_approximate_stroke_extents
	0x41700	libcairo.so	_intern_string_pluc
5	0x6a5c0	libcairo.so	_cairo_scaled_font_thaw_cache
	0x41700	libcairo.so	_intern_string_pluc

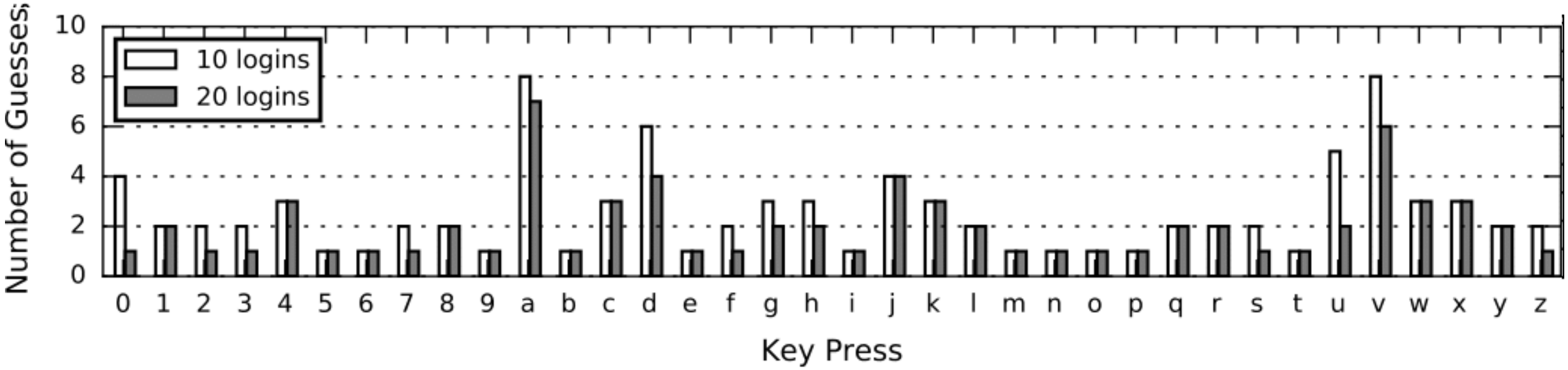
Attack 1: Onboard

- Single keypress prediction



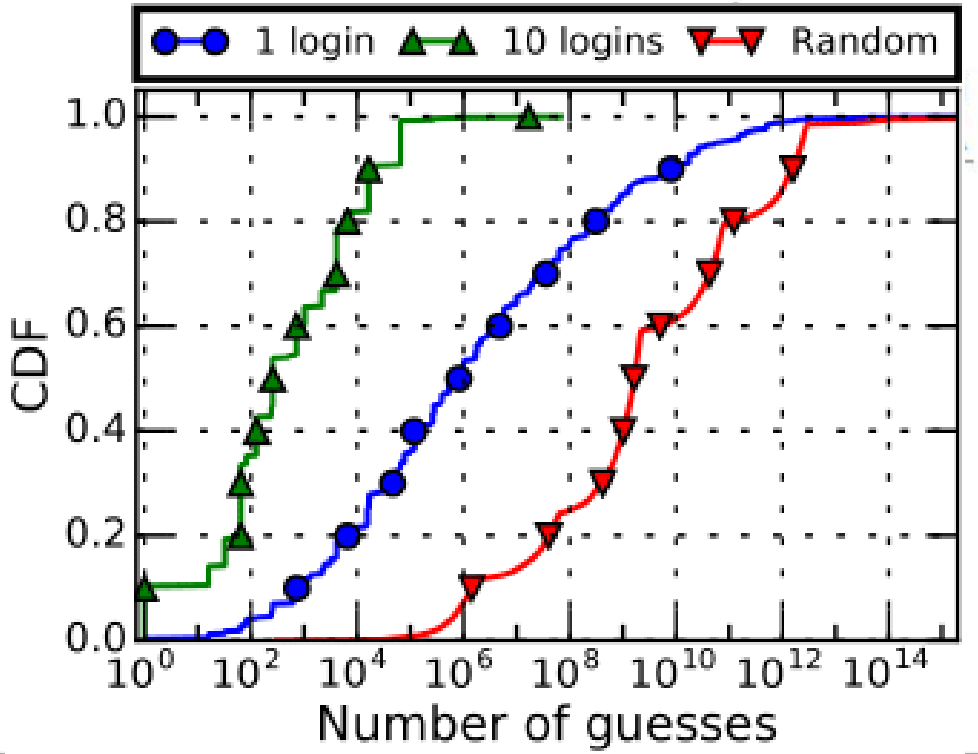
Attack 1: Onboard

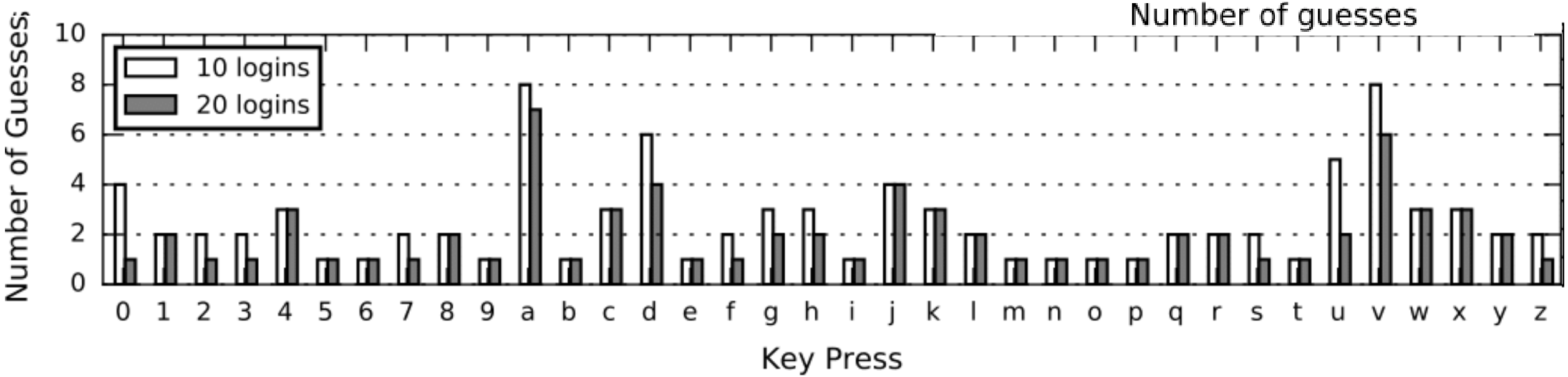
- Single keypress prediction
- Augment 1: Multiple logins



Attack 1: Onboard

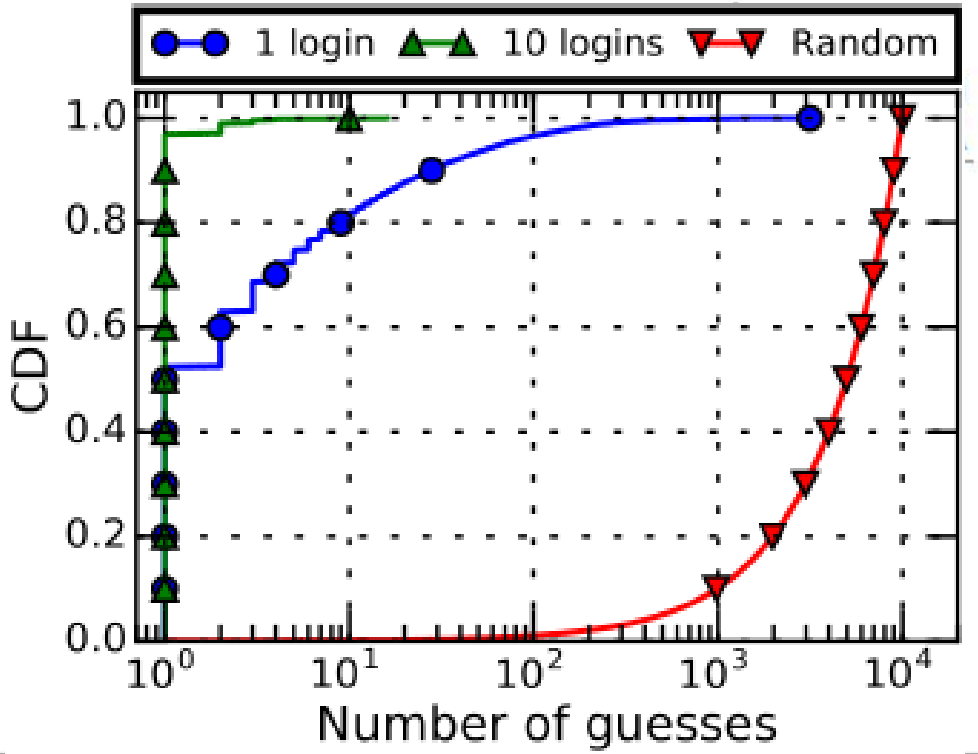
- Single keypress prediction
- Augment 1: Multiple logins

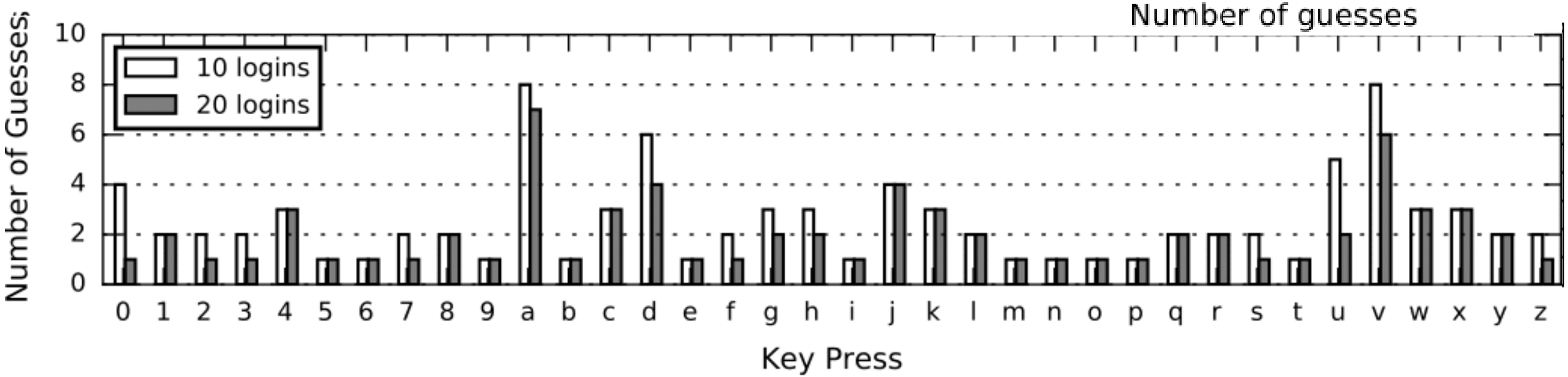




Attack 1: Onboard

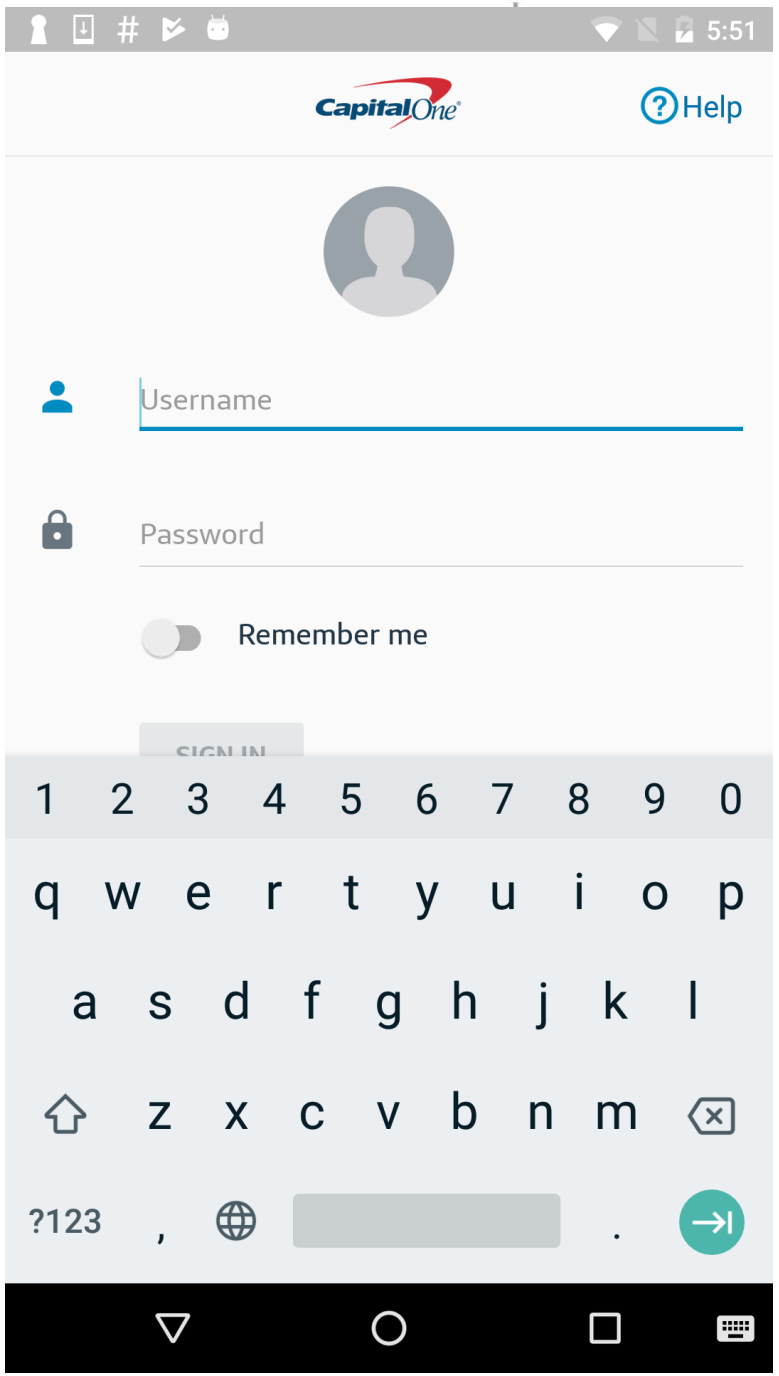
- Single keypress prediction
- Augment 1: Multiple logins
- Augment 2: Dictionary





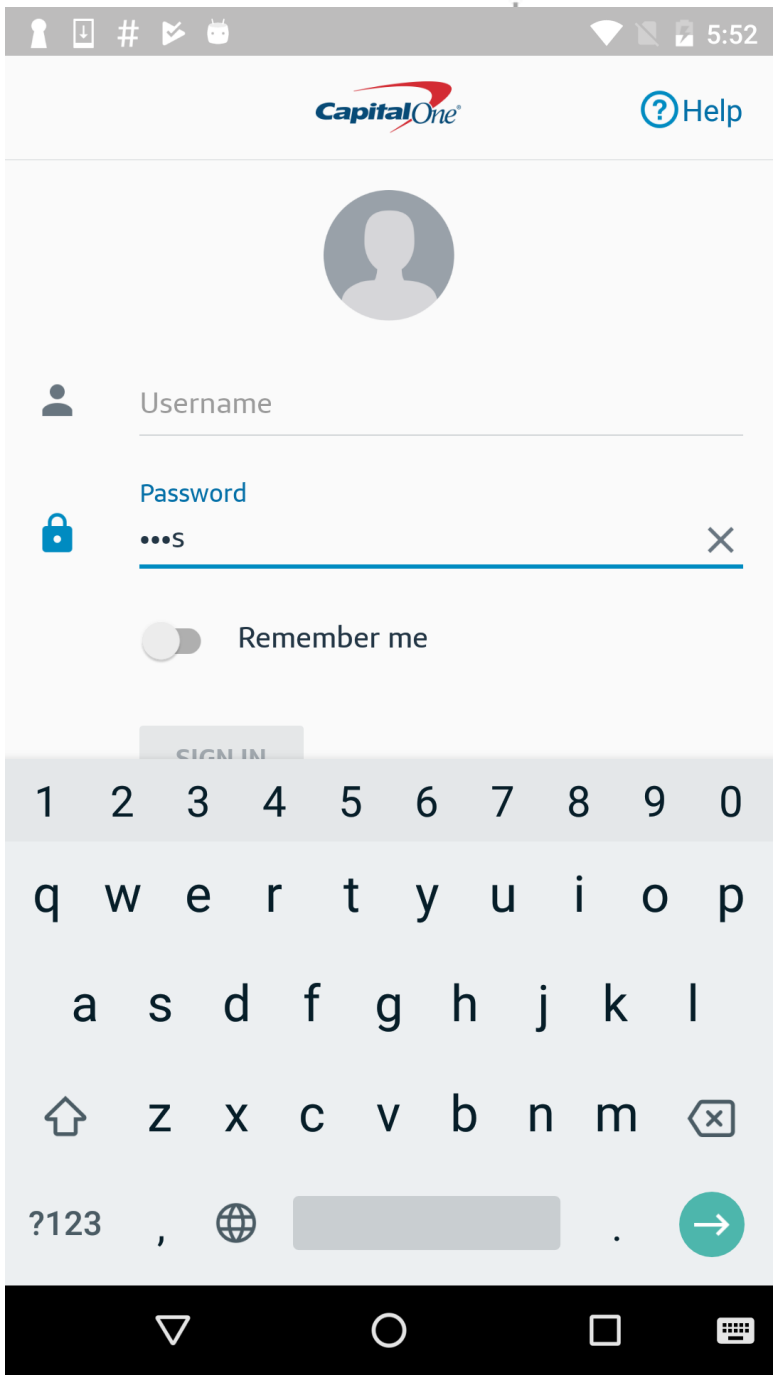
Attack 2: Capital One

- Huawei Nexus 6P. Android 8.0
- Flush+reload => Evict+reload
  - Resolution 20x lower than Intel



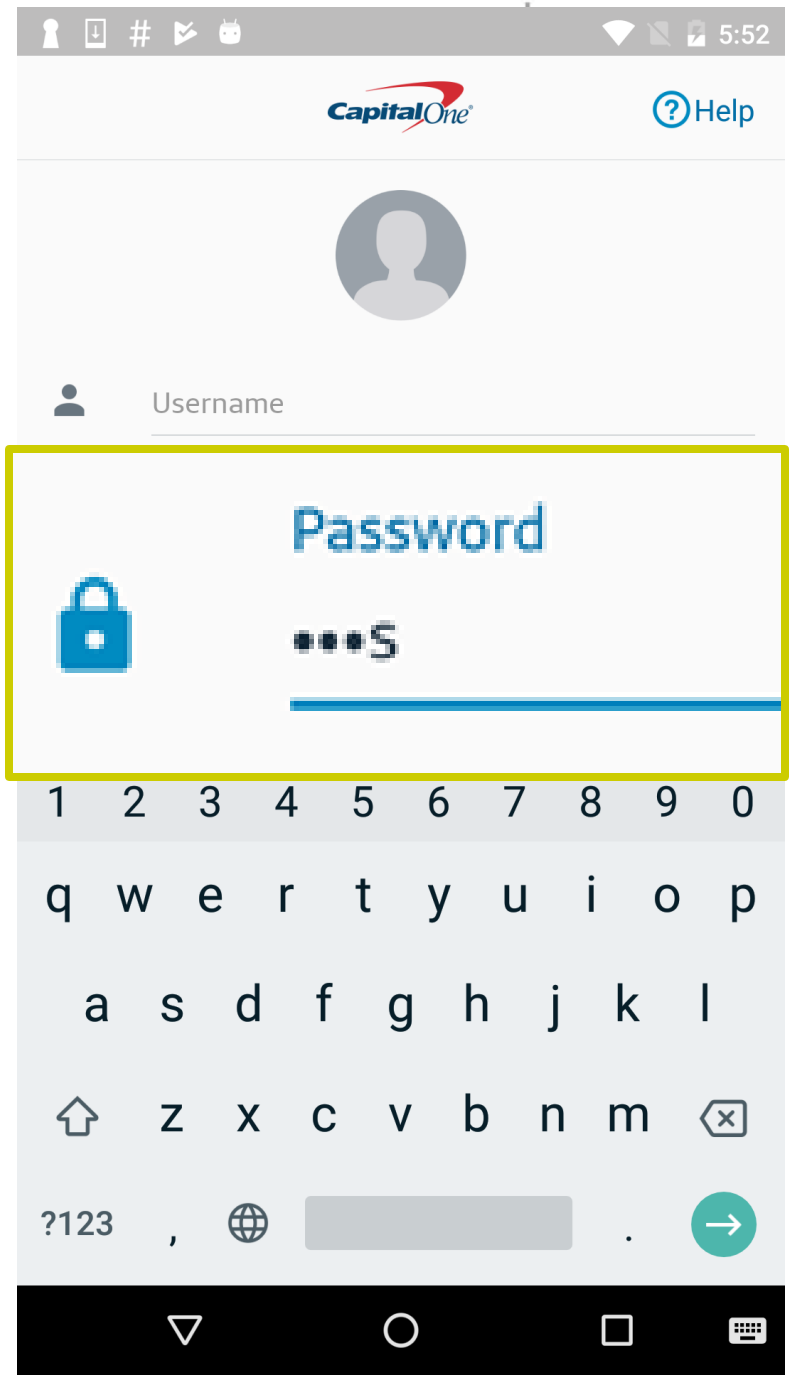
Attack 2: Capital One

- Huawei Nexus 6P. Android 8.0
- Flush+reload => Evict+reload
  - Resolution 20x lower than Intel



Attack 2: Capital One

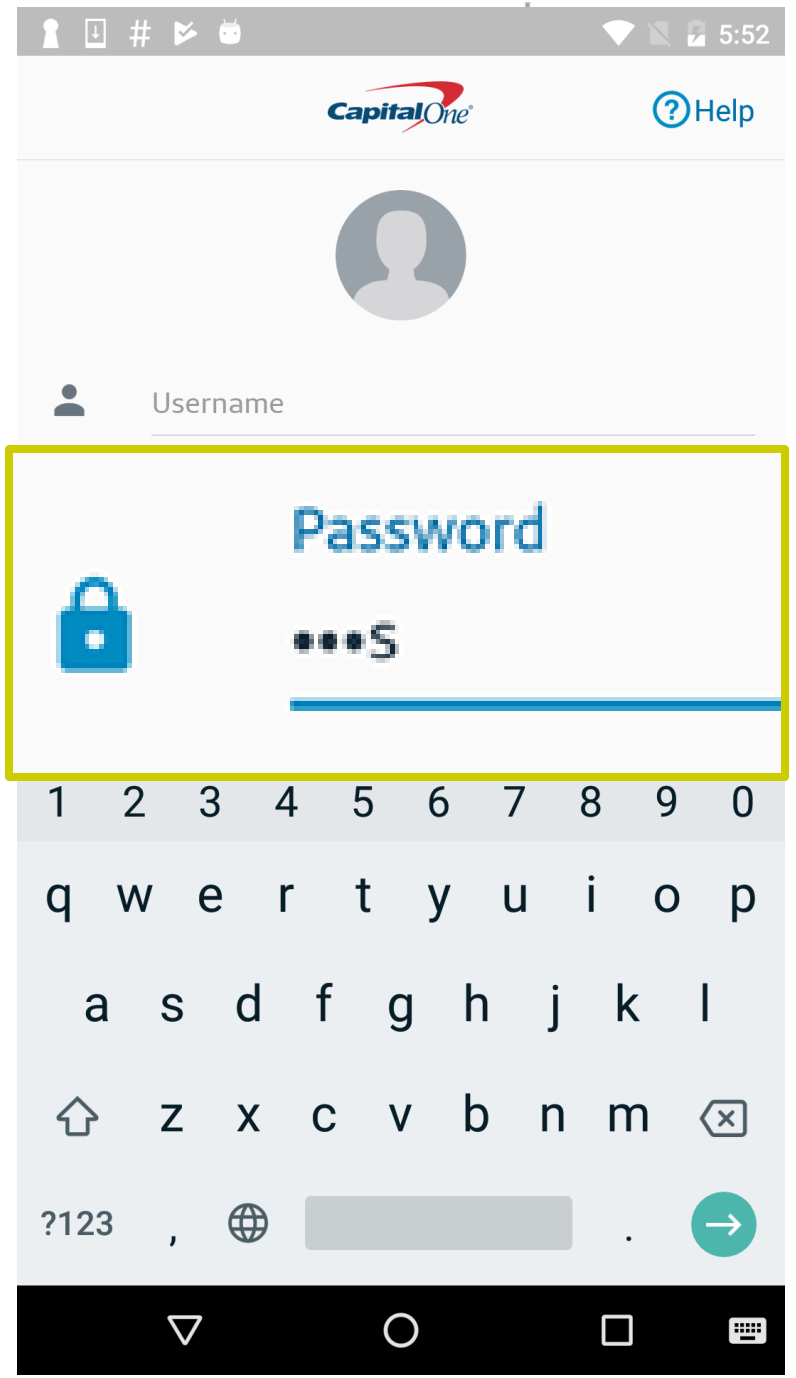
- Huawei Nexus 6P. Android 8.0
- Flush+reload => Evict+reload
  - Resolution 20x lower than Intel



Attack 2: Capital One

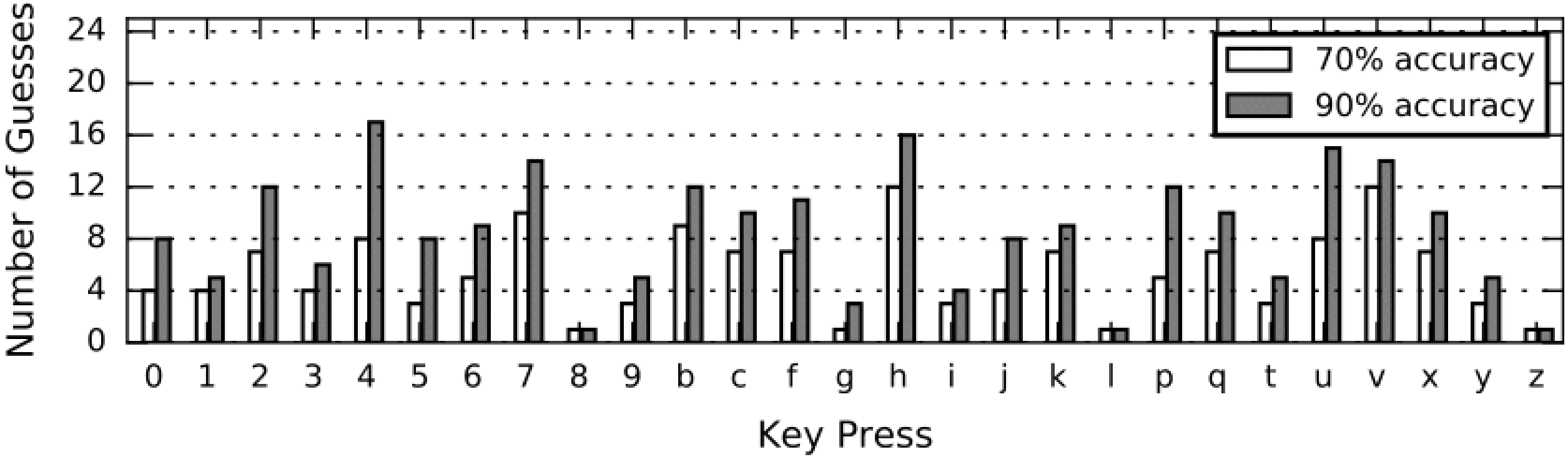
- Huawei Nexus 6P. Android 8.0
- Flush+reload => Evict+reload
  - Resolution 20x lower than Intel
- Pre-render triggered only once

#	Library	Function
1	libskia.so	SkScalerContext_FreeType_Base ::generateGlyphImage
	libskia.so	SkMask::getAddr
2	libskia.so	SkGlyph::computeImageSize
	libskia.so	SkAAClipBlitter::~SkAAClipBlitter



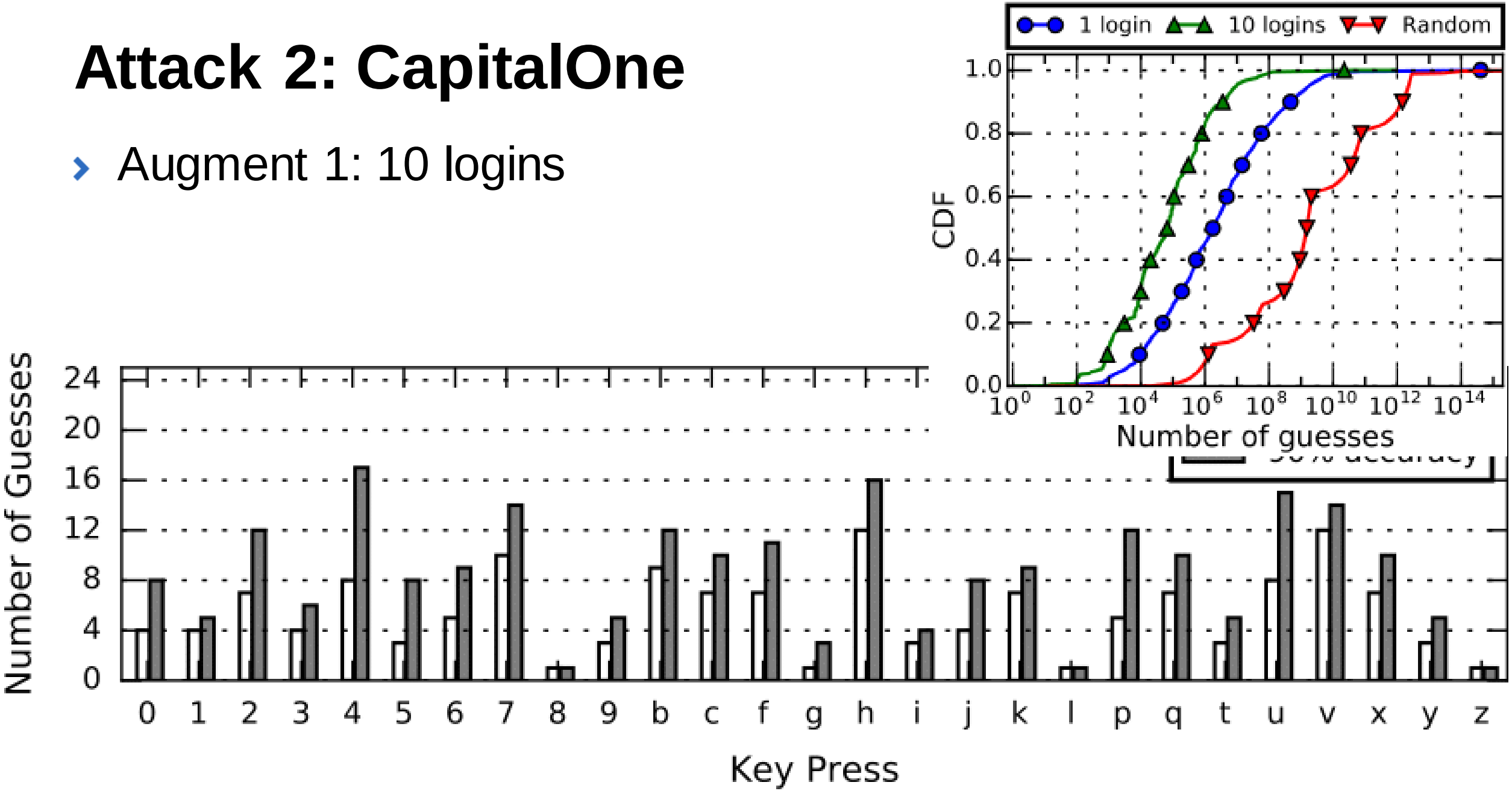
Attack 2: CapitalOne

- Augment 1: 10 logins



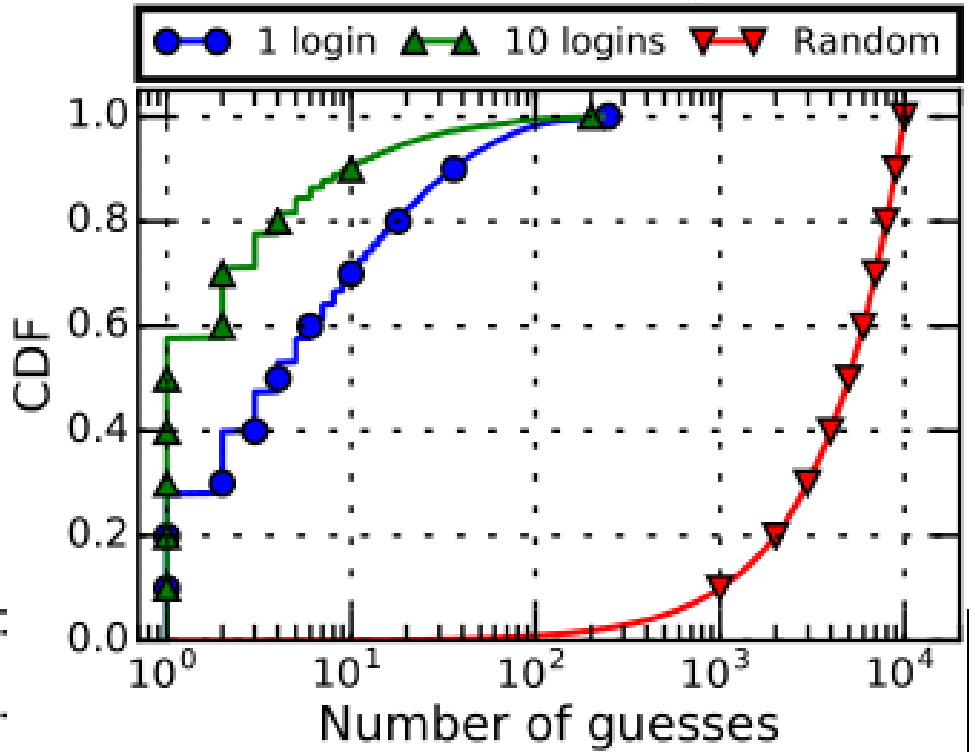
Attack 2: CapitalOne

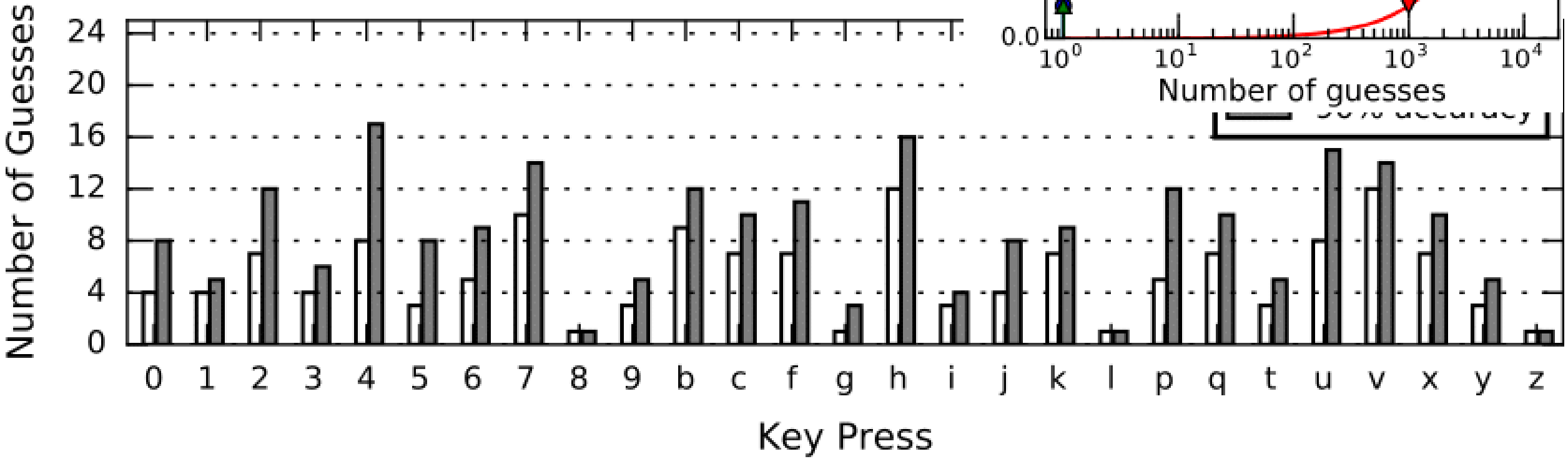
- Augment 1: 10 logins



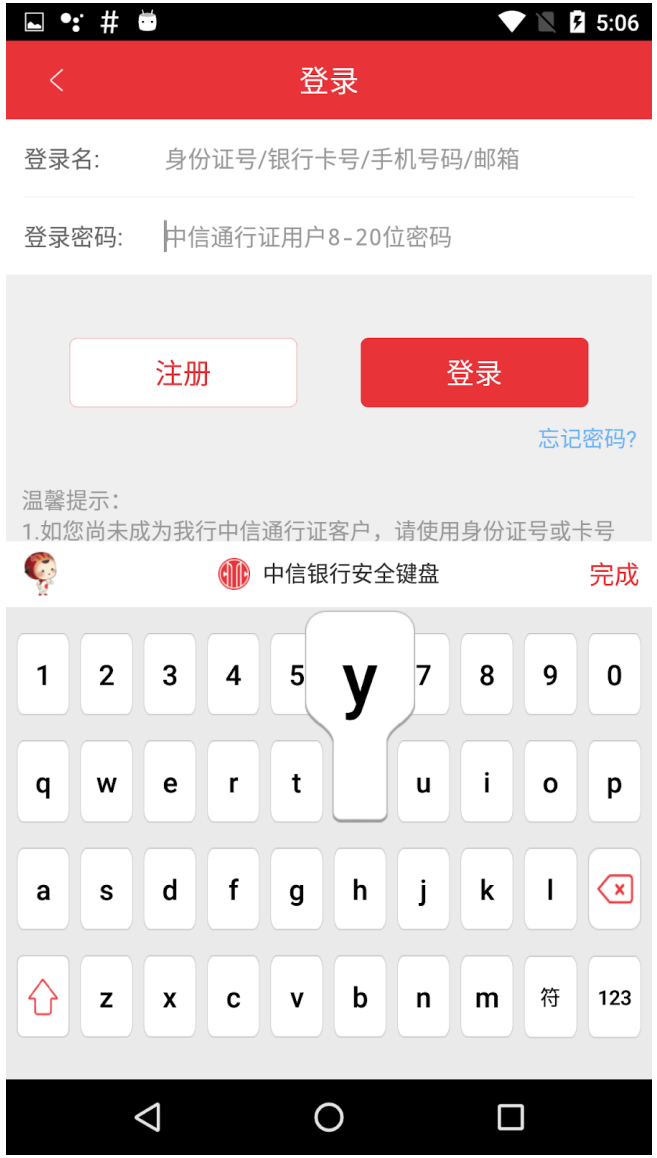
Attack 2: CapitalOne

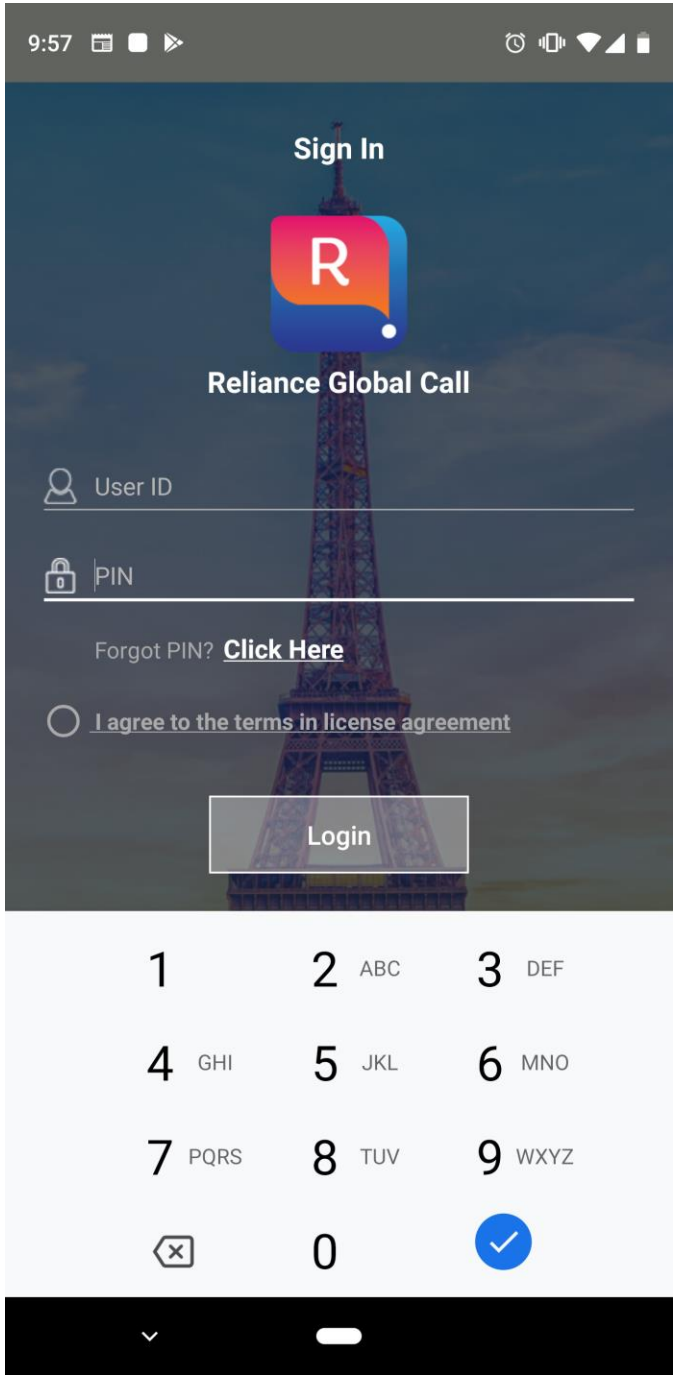
- Augment 1: 10 logins
- Augment 2: Dictionary





Other Apps


5:06
登录
登录名:
身份证号/银行卡号/手机号码/邮箱
登录密码:
中信通行证用户8-20位密码
注册
登录
忘记密码?
温馨提示:
1.如您尚未成为我行中信通行证客户，请使用身份证号或卡号
中信银行安全键盘
完成


9:57
Sign In
Reliance Global Call
User ID
PIN
Forgot PIN? Click Here
I agree to the terms in license agreement
Login

Discussion

- Measurement challenge
  - Measurement resolution
- Extensions
- Mitigations

Discussion

- Measurement challenge
- Extensions
  - Inter-keystroke timing
  - Combining with other side-channels
  - Examine other libraries
- Mitigations

Discussion

- Measurement challenge
- Extensions
- Mitigations
  - Prevent flush+reload
  - Constant-time rendering

Conclusion

- Effective execution time measurement
- Exposing side-channels in graphics libraries
- Evaluations on real-world applications
- Acknowledgement
  - U.S. Army Research Laboratory Cyber Security Collaborative Research Alliance

UNIVERSITY OF CALIFORNIA
UC RIVERSIDE

Thanks!

Q & A