Time Trial

Racing Towards Practical Remote Timing Attacks

Daniel A. Mayer
@DanIAMayer
http://cysec.org

Joel Sandin
jsandin@matasano.com

black hat USA 2014

matasano SECURITY

Who we are...

- Daniel A. Mayer
  - Senior Appsec consultant with Matasano Security.
  - Ph.D. in Computer Science (Security and Privacy).
- Joel Sandin
  - Appsec consultant with Matasano

- Matasano Security
  - Application Security Consultancy.
  - Offices in New York, Chicago, Sunnyvale.
  - Part of nccgroup freedom from doubt

Agenda

1. Timing Side-Channels
2. Remote Timing Attacks
3. Our Tool: Time Trial
4. Timing Attacks in Practice
5. Conclusion

Side-Channels

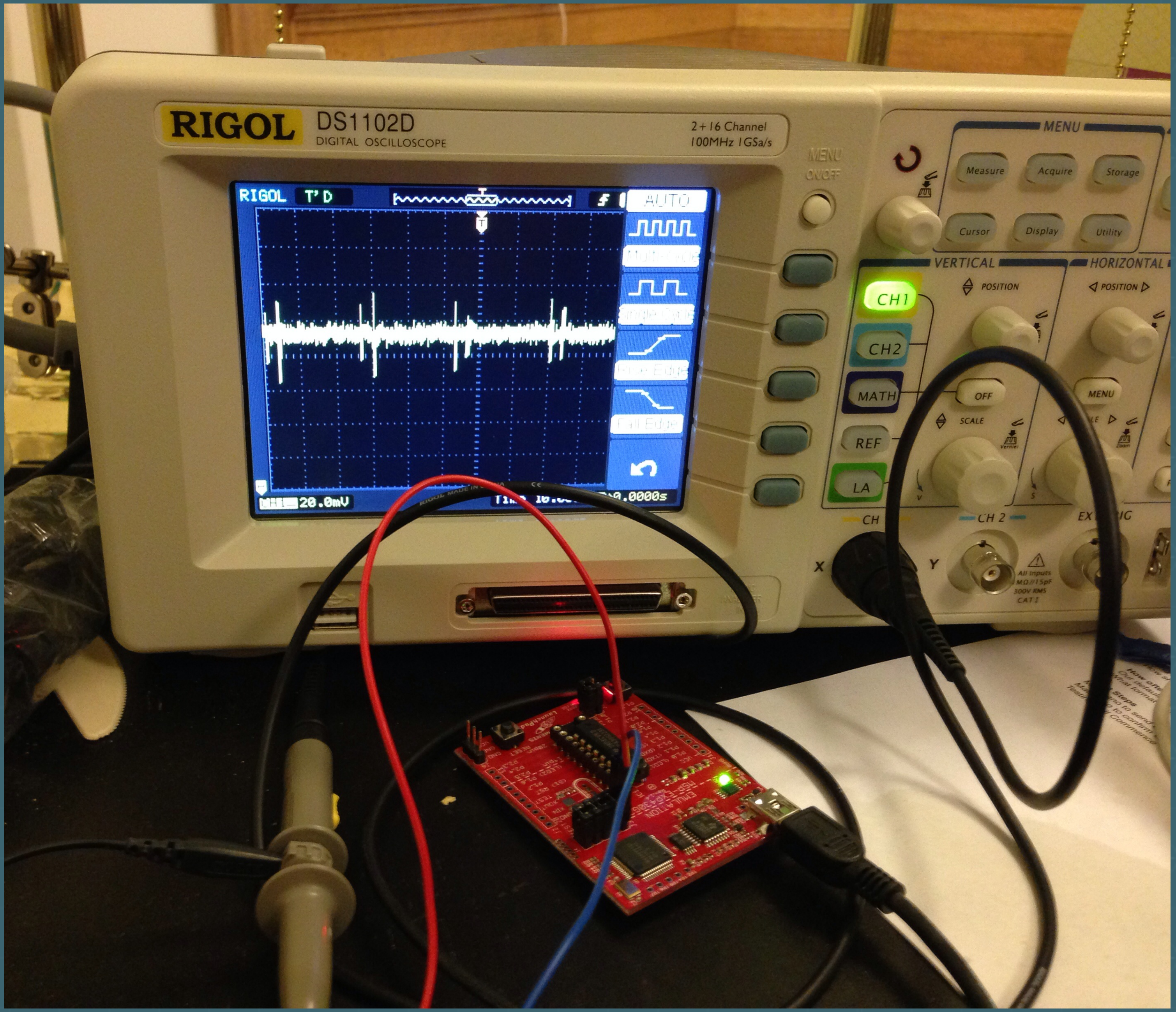
Side-Channel Attacks

matasano SECURITY

Side-Channel Attacks

Examples of Side-Channels

- Power consumption
- RF emissions
- Sound
- Processing Time



- *Really, anything that can be measured and is related to a secret.*

"Regular Vulns" vs. Side-Channels

- Many vulnerabilities well understood
  - XSS, CSRF, SQL injection
  - Developers becoming more aware
  - Frameworks: Harder to introduce bugs

- Side-channels: Less so
  - Easy to introduce using "innocent" operators
  - Hard to observe and test for
  - Have to go out of one's way to prevent them

Timing Side-Channels

- Response time differs depending on computation
- Attacker can learn information about system
  - sensitive credentials
  - internal system state
- Easy to introduce
- Exploitable remotely?

Timing Side-Channels

- Exploitable remotely?

Basic Timing Side-Channel

```
post '/login' do
  if not valid_user?(params[:user])
    "Username or Password incorrect"
  else
    if verify_password(params[:user], params[:password])
      "Access granted"
    else
      "Username or Password incorrect"
    end
  end
end
```

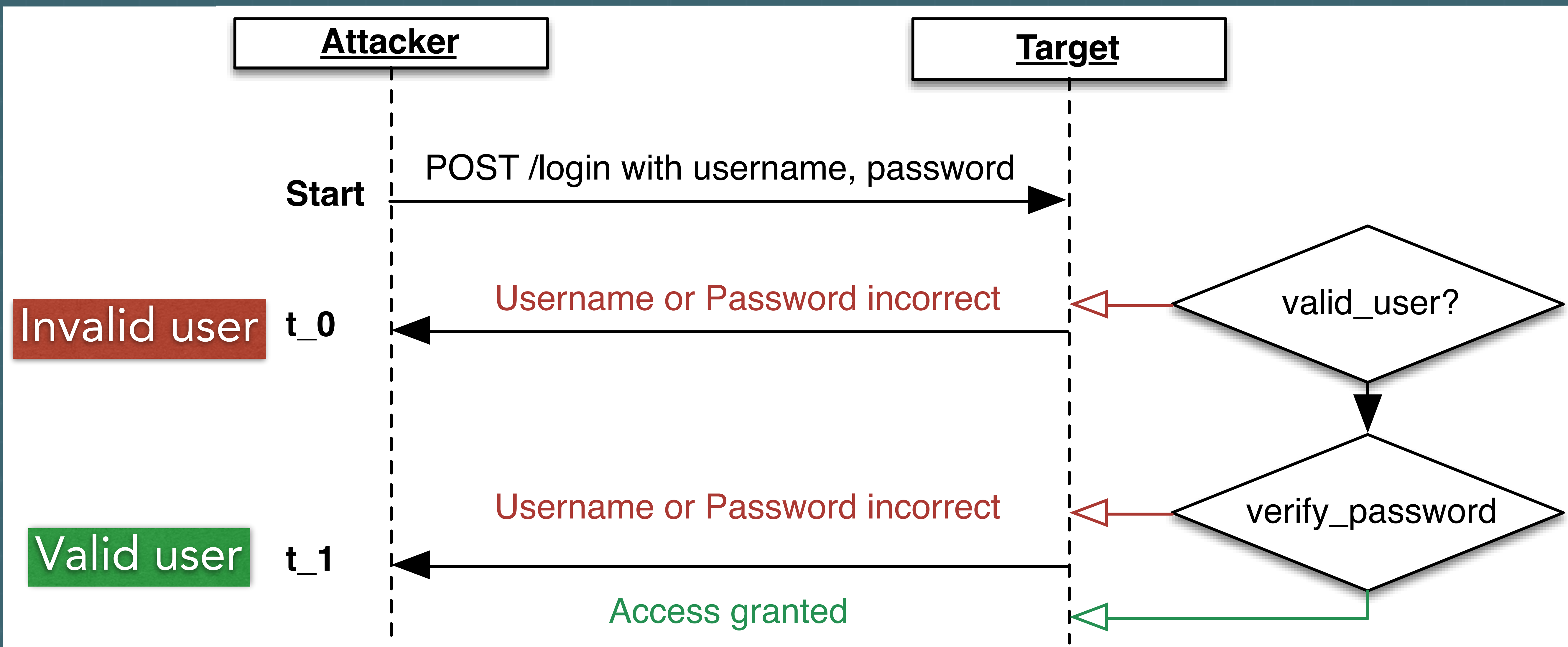
Invalid user

Valid user wrong password

matasano SECURITY

Timing Attacks

- Reason about system based on response time



matasano SECURITY

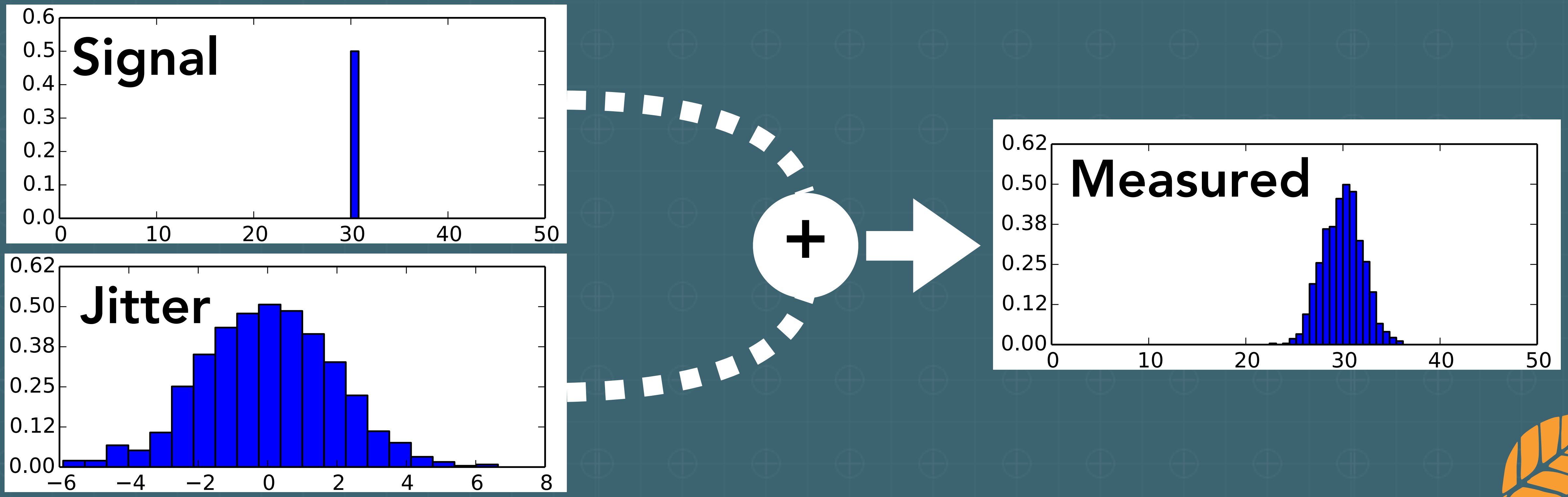
Prior Work!

- Rich history of timing attacks in crypto, e.g.
  - **Kocher**, 1996
    Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems
  - **Brumley and Boneh**, 2005
    Remote Timing Attacks are Practical
- Excellent empirical studies, e.g.
  - **Crosby et al.**, 2009
    Opportunities and Limits of Remote Timing Attacks
  - **Lawson and Nelson**, 2010
    Exploiting Timing Attacks In Widespread Systems

Remote Timing Attacks

Local vs. Remote - Challenges

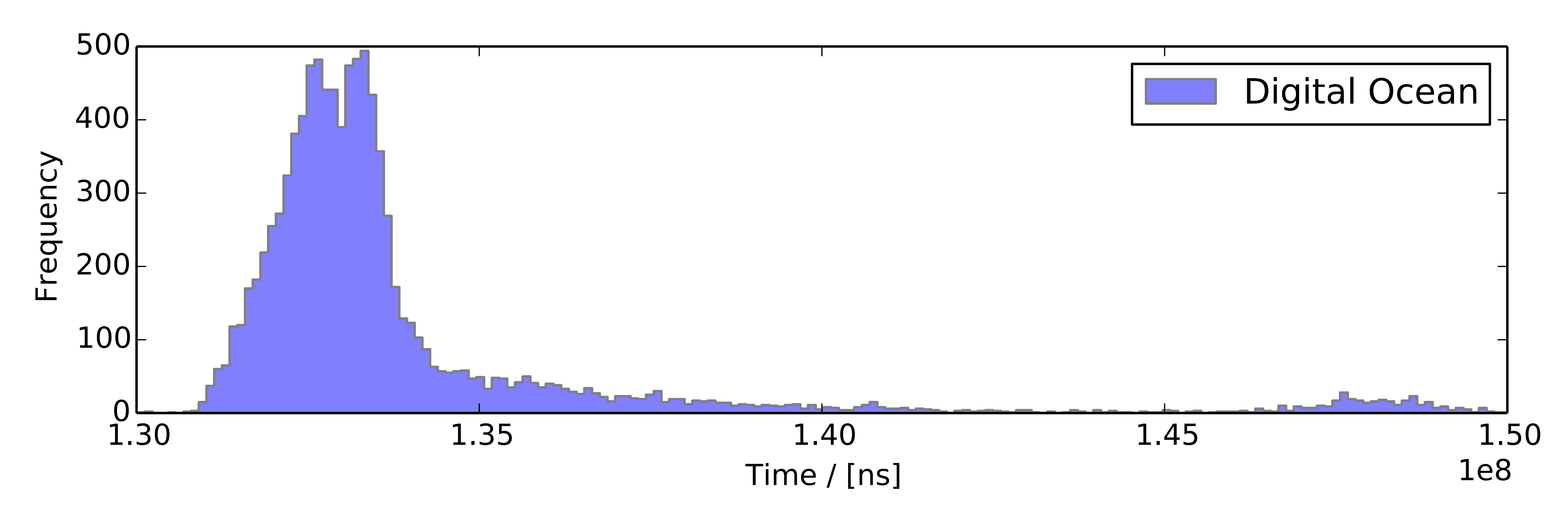
- Local attacks
  - Precise measurement of execution time
  - Can minimize external influences
- Remote attacks
  - Propagation time added to the measurement.
  - Network delays add jitter.



matasano SECURITY

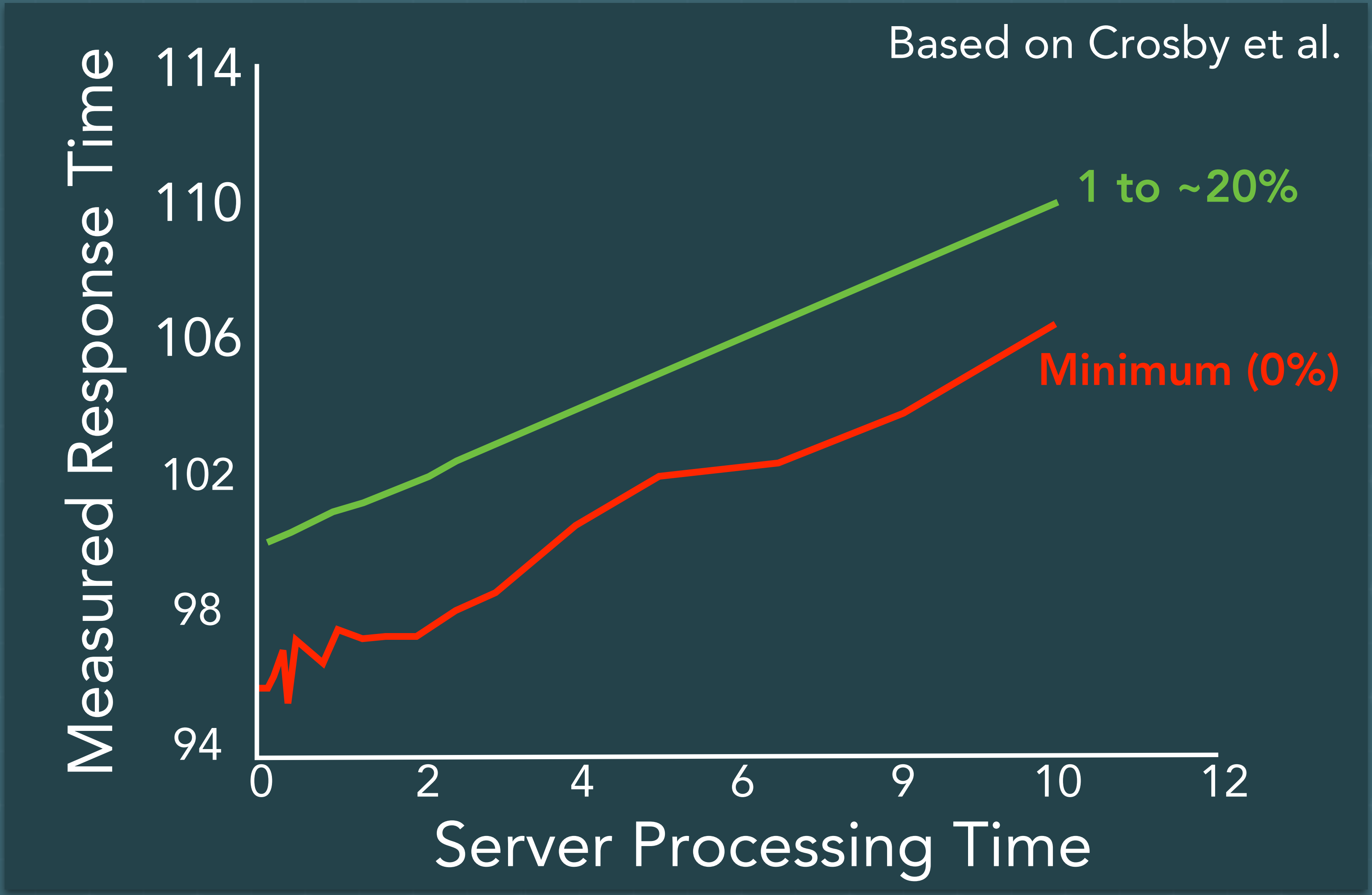
Real Jitter

- Additional Caveat:
  - Distribution isn't Gaussian, hard to model
  - Skewed, multiple modes



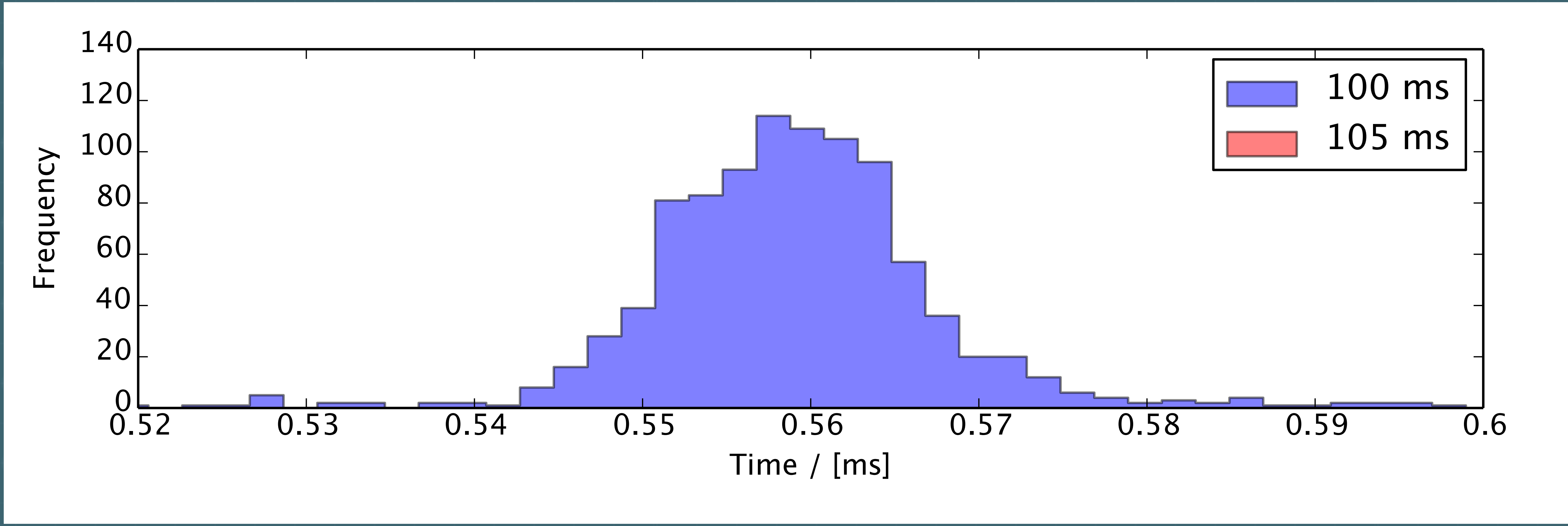
Statistical Methods

- Measure a large number of response times
- Measurement must be related to processing time!
- Median and minimum not good indicators



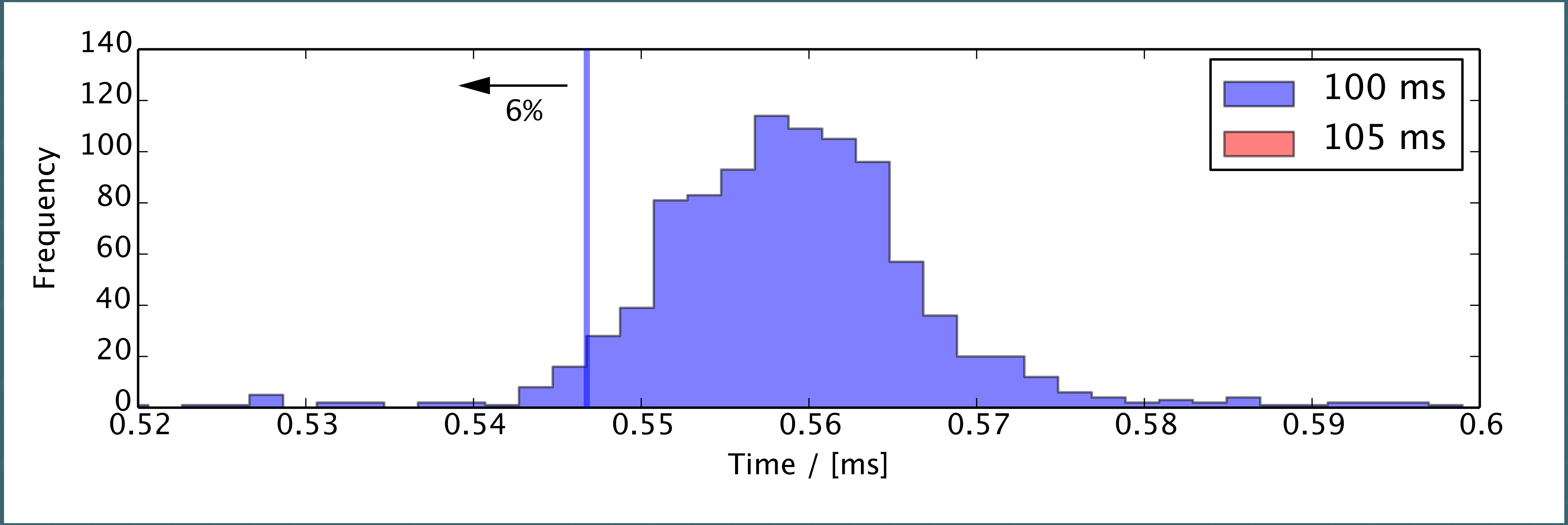
Statistical Methods

- The Box Test
- Compare intervals induced by percentiles
- Percentiles to be determined empirically



Statistical Methods

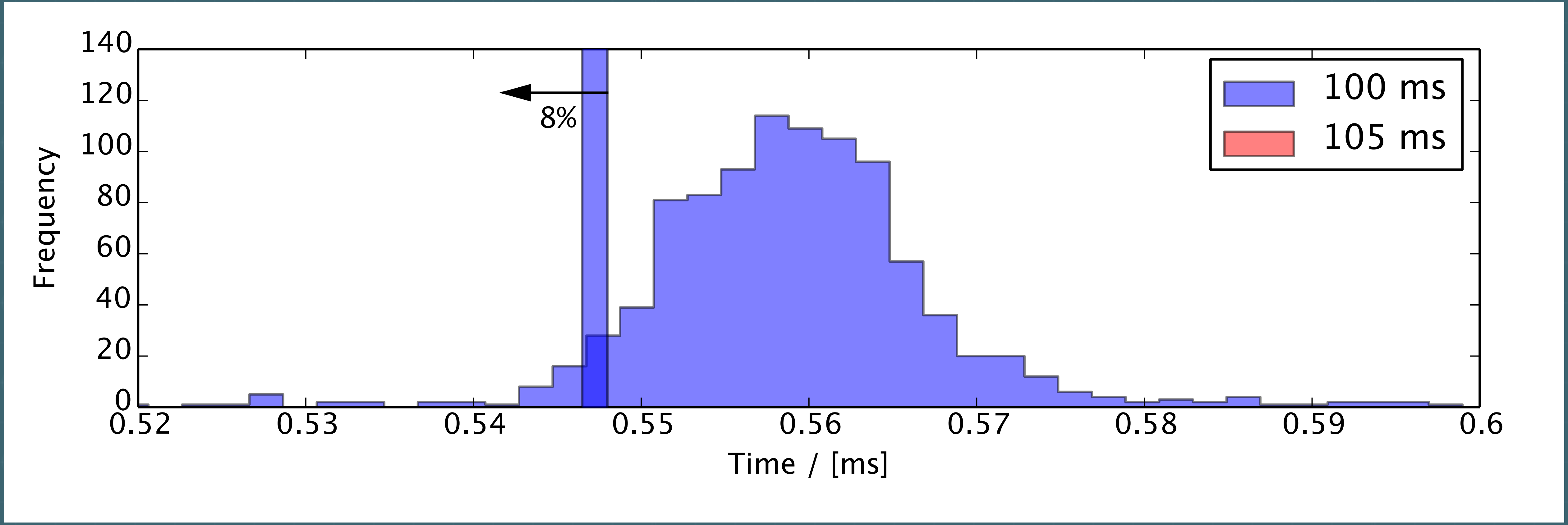
- The Box Test
- Compare intervals induced by percentiles
- Percentiles to be determined empirically



matasano SECURITY

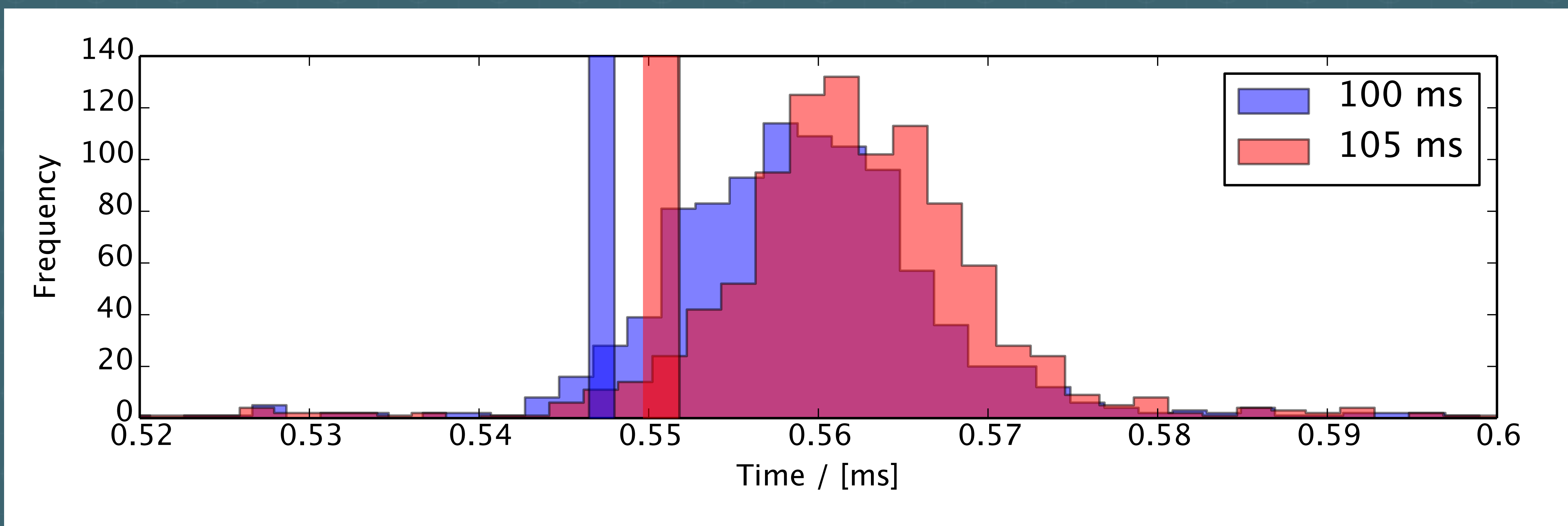
Statistical Methods

- The Box Test
- Compare intervals induced by percentiles
- Percentiles to be determined empirically



Statistical Methods

- The Box Test
- Compare intervals induced by percentiles
- Percentiles to be determined empirically



matasano SECURITY

New Tool: Time Trial

Why a tool for timing attacks?

- No way to demonstrate impact
- Separate theoretical issues from exploitable vulnerabilities
- Reframes the debate about practicality of these attacks

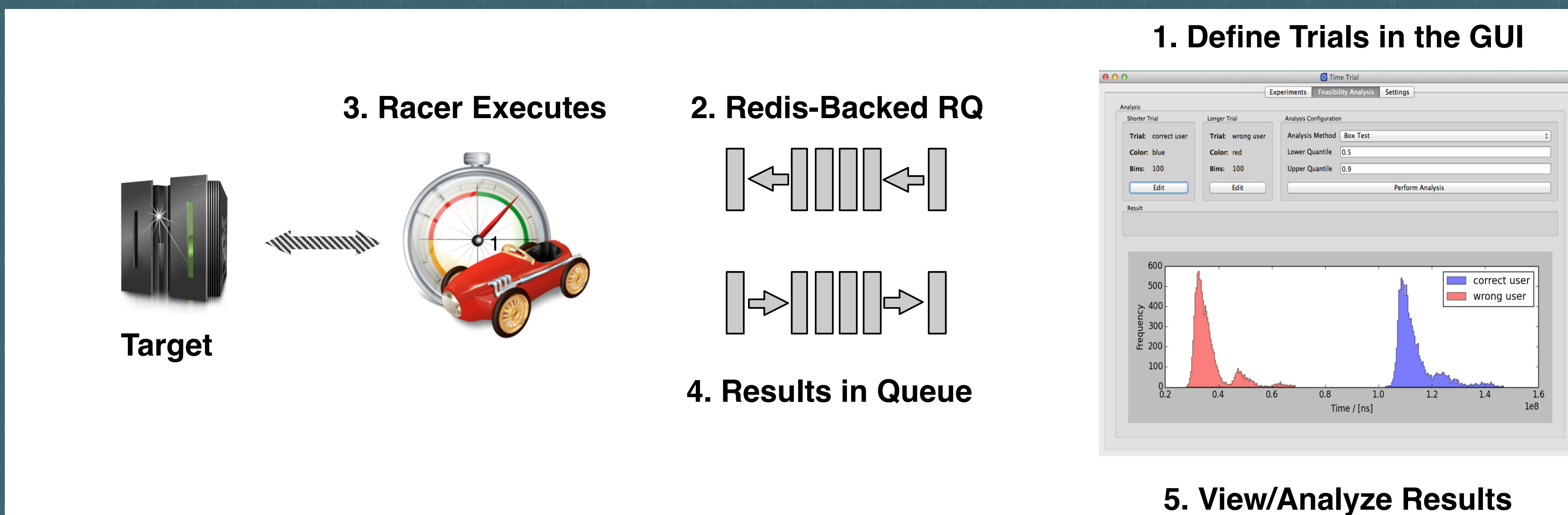
matasano SECURITY

Time Trial

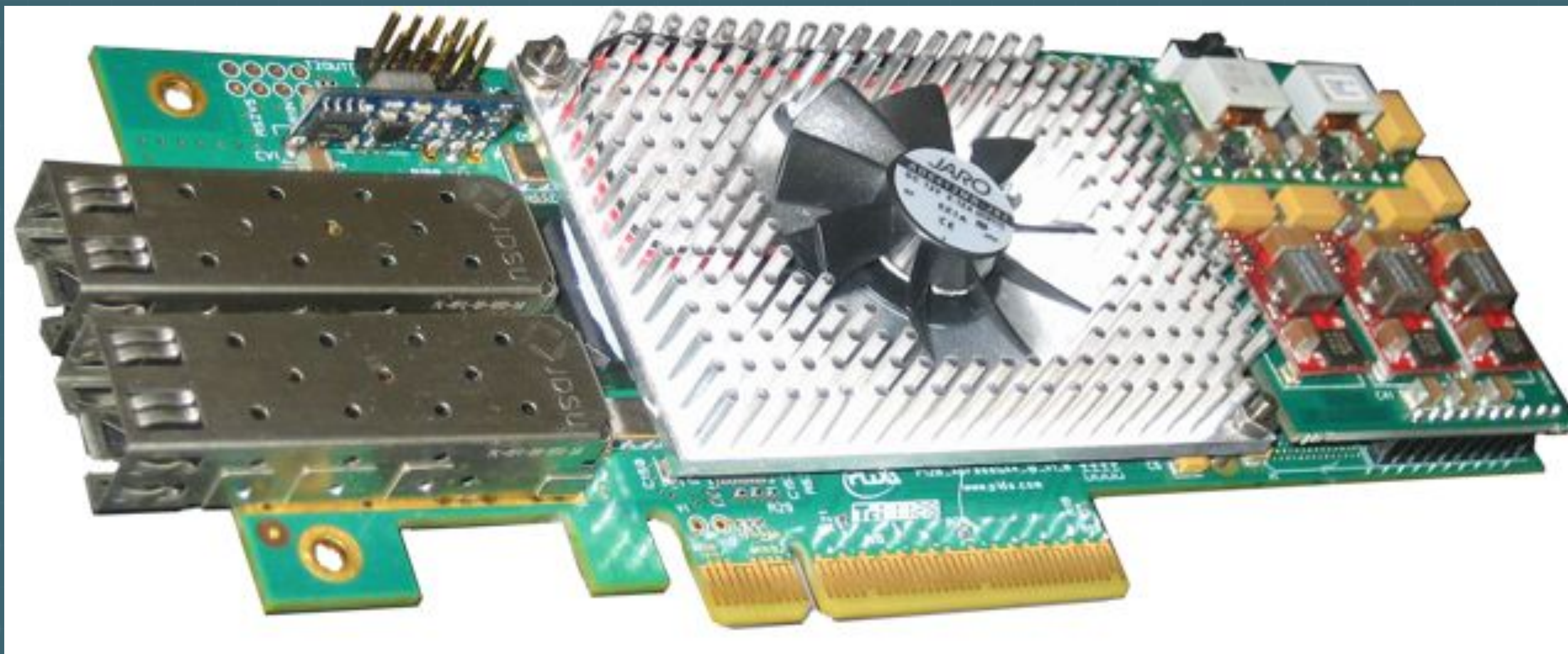
- What Time Trial is:
  - A framework for capturing precise timing
  - A tool for feasibility analysis
  - A generator of visual proof-of-concepts
- What Time Trial is NOT (yet):
  - A read-to-use exploit framework
  - An automated attack tool

Goals and Design

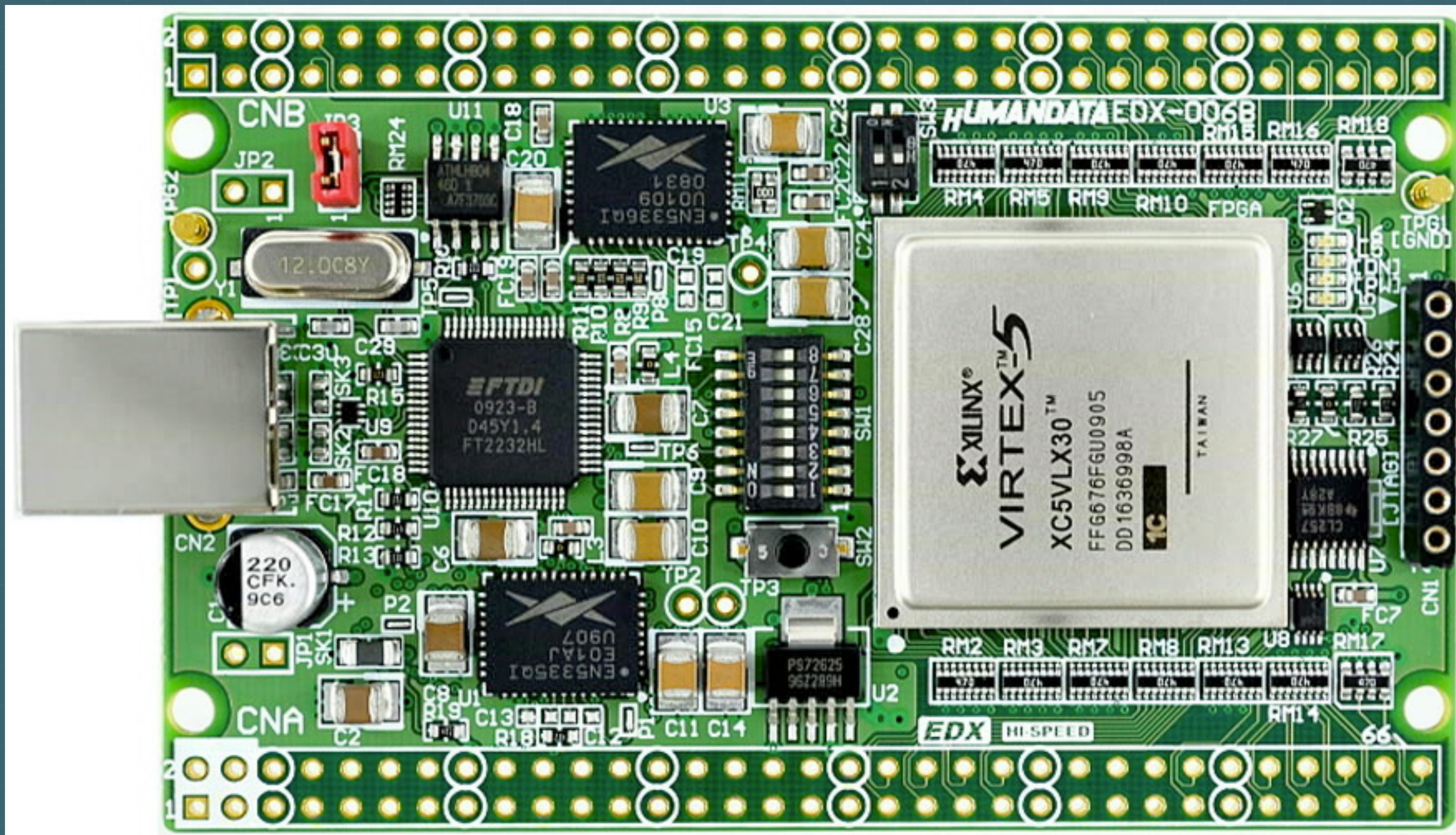
- Separate "racer" sensor from analytic front end.
  - Front end: Python + Qt
  - Racer: C++
- Schedule trials and analyze results



How to do precise time measurements?



How to do precise time measurements?



How to do precise time measurements?

matasano SECURITY

Optimizations

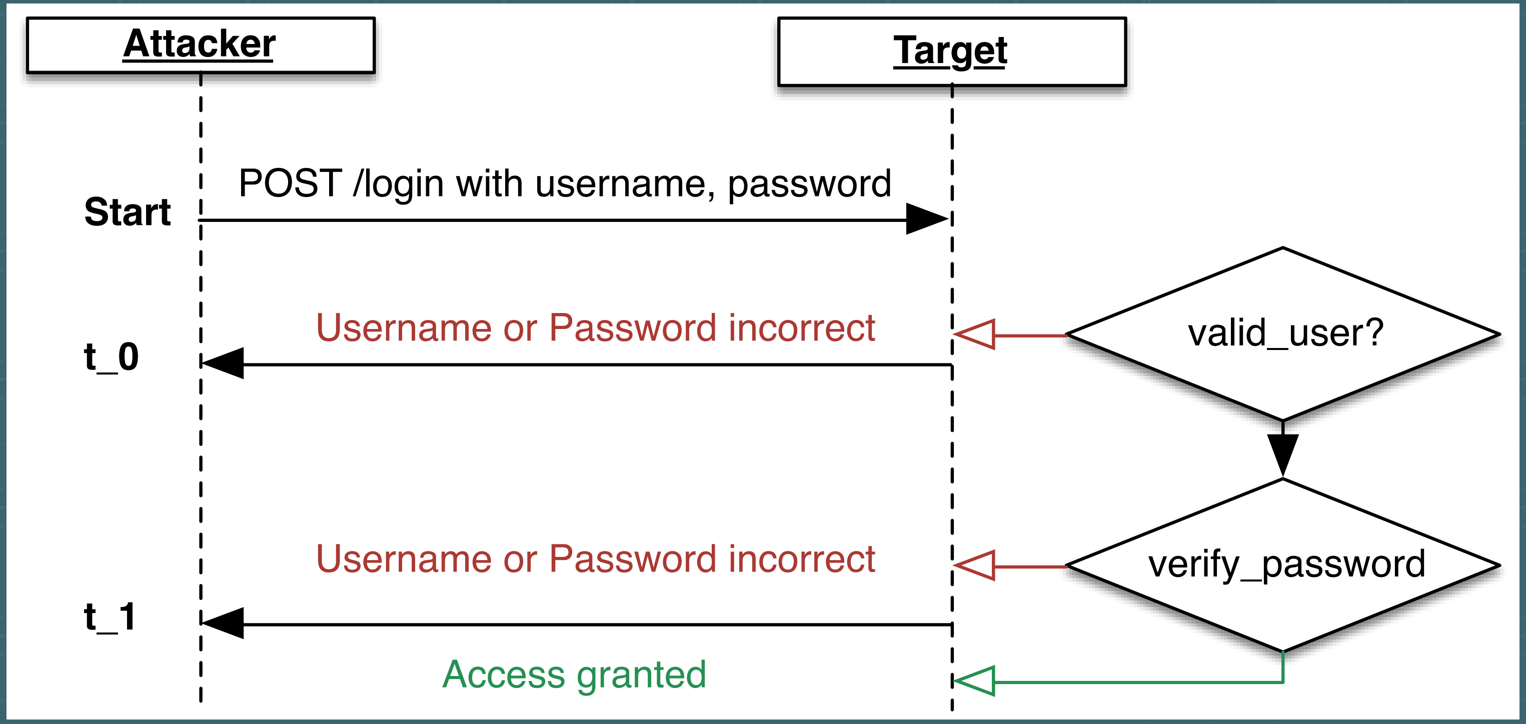
- Use clock_gettime for nanosecond timer
  - Using MONOTONIC clock
- Used fixed, reserved CPU core
  - GRUB_CMDLINE_LINUX_DEFAULT="maxcpus=2 isolcpus=1"
  - CPU affinity
- Run with real-time priority
- Disable frequency scaling

DEMO: Time Trial

Lets get some data!

Data across different networks

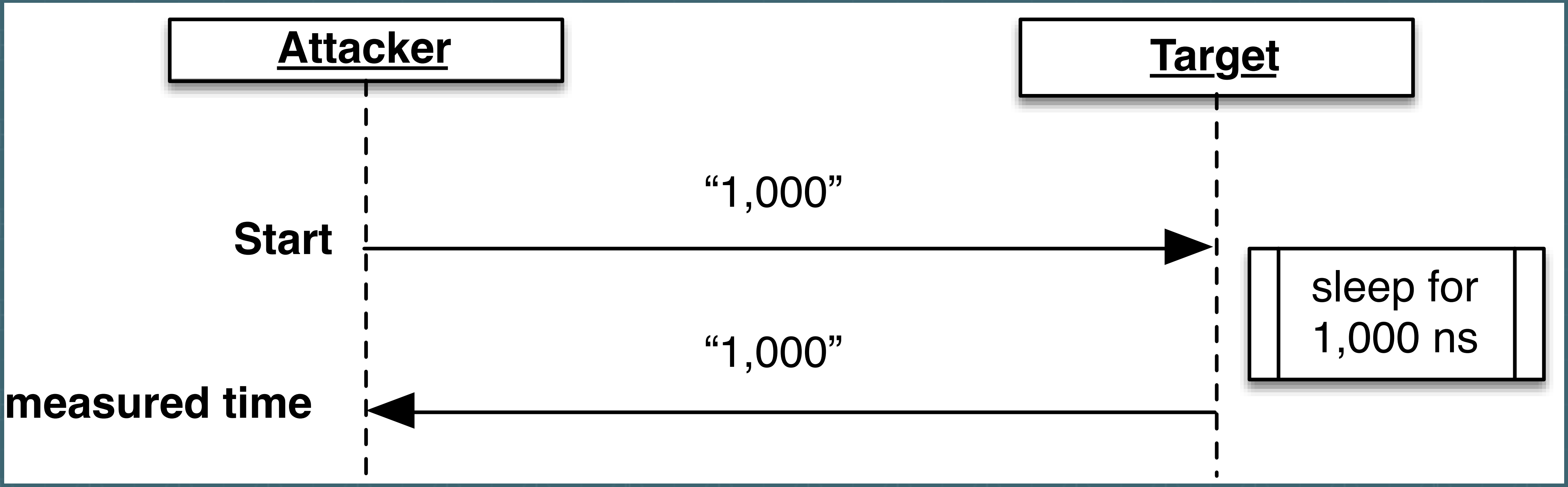
- Analyzed response time distributions for different networks:
  - LAN
  - Internet at large
  - Cloud environments



- In order to exploit: distinguish response times.
  - Was the response t_0 or t_1 for given input?

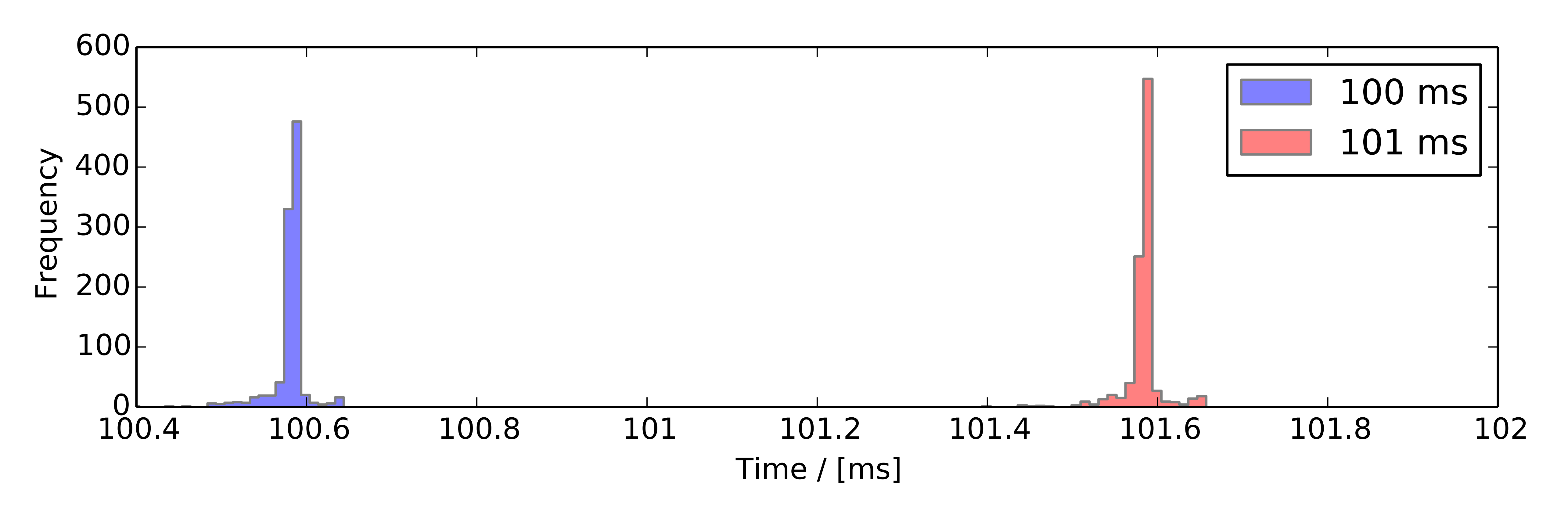
Feasibility Based on Echo Trials

- What timing differences can be distinguished in practice?
  - Similar to the approach by Crosby et al.



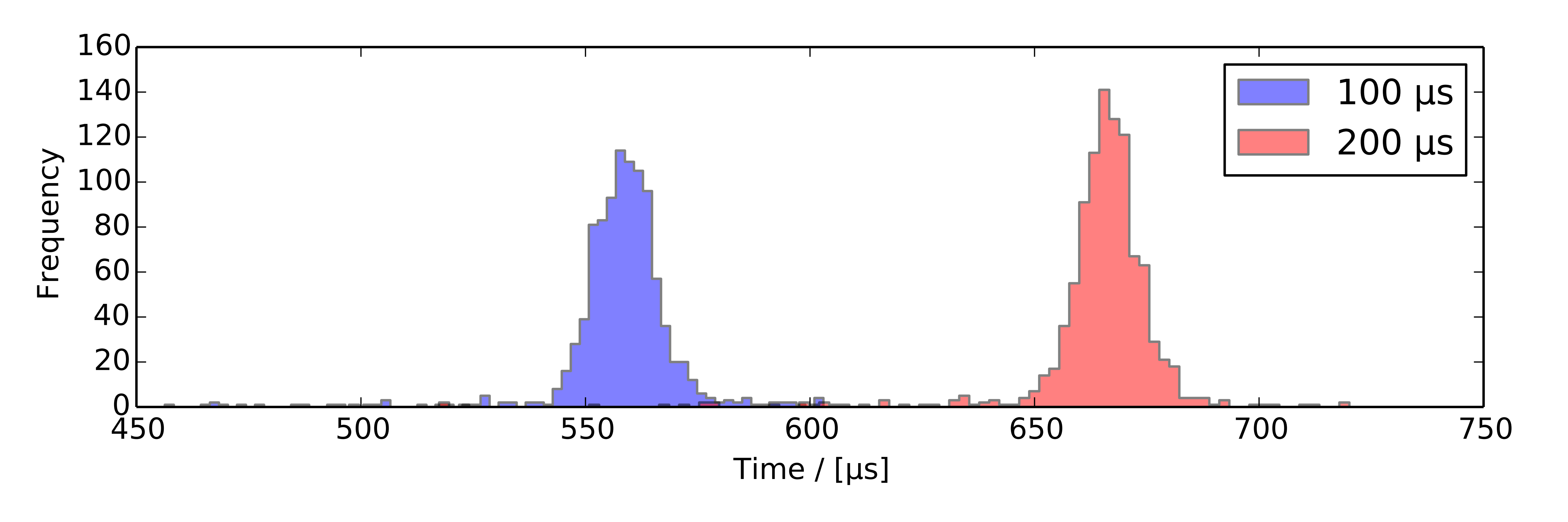
matasano SECURITY

Timing Resolution: LAN



1,000 Repetitions

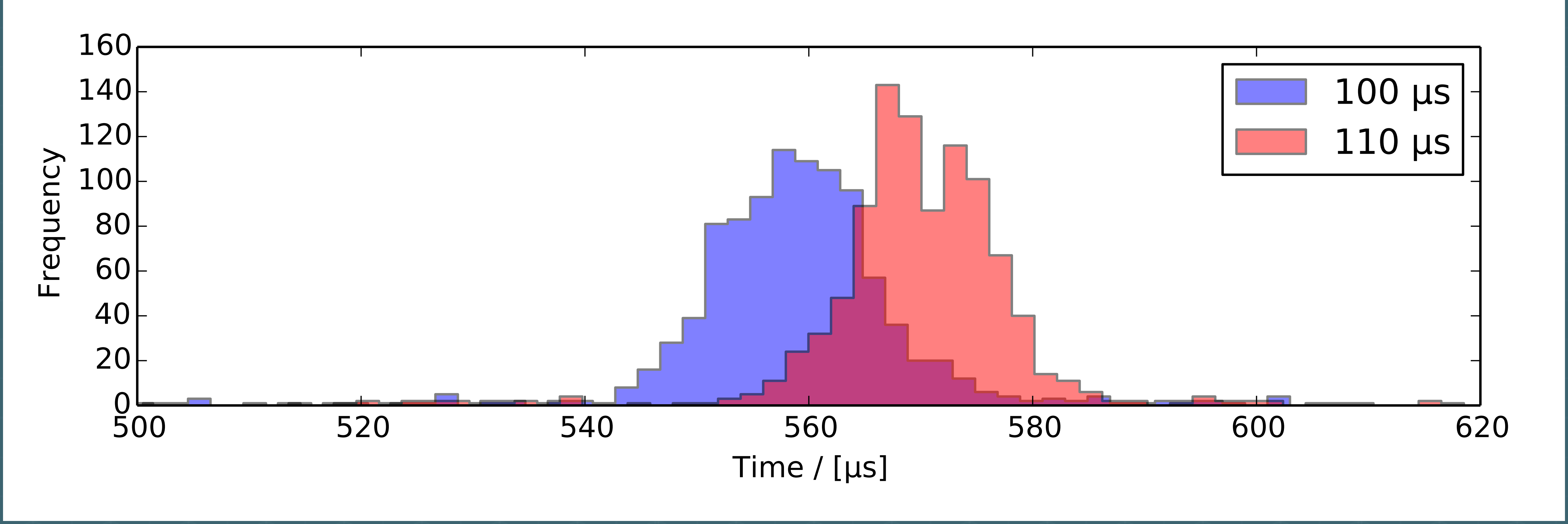
Timing Resolution: LAN



1,000 Repetitions

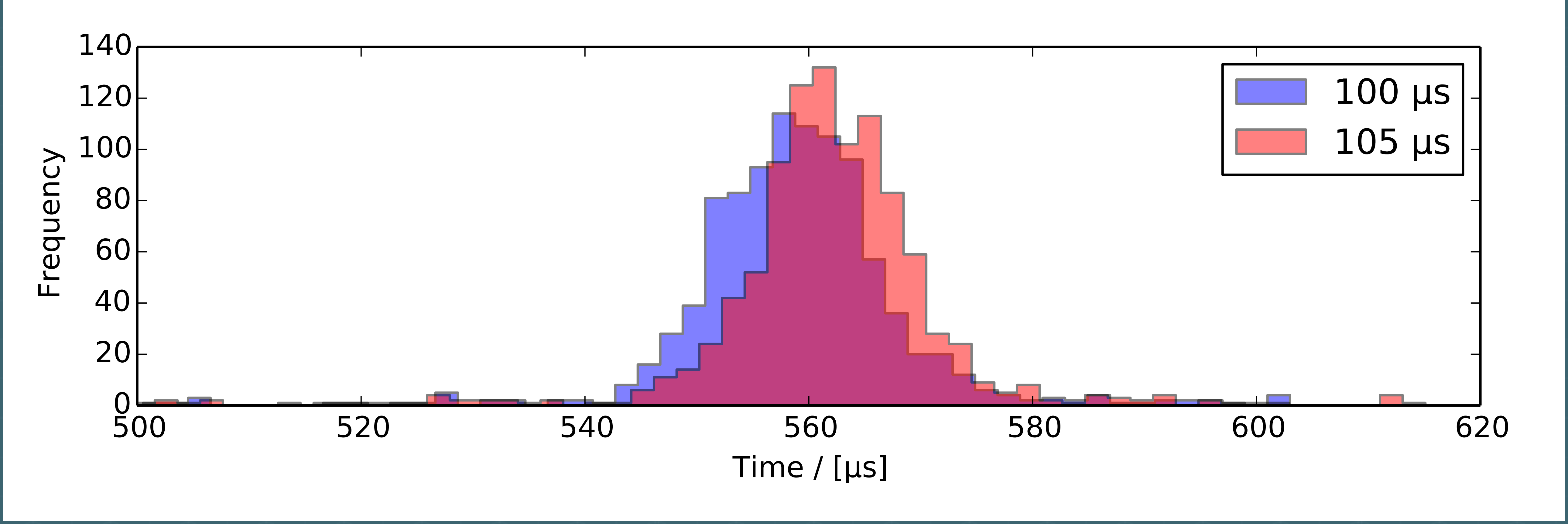
matasano SECURITY

Timing Resolution: LAN



1,000 Repetitions

Timing Resolution: LAN

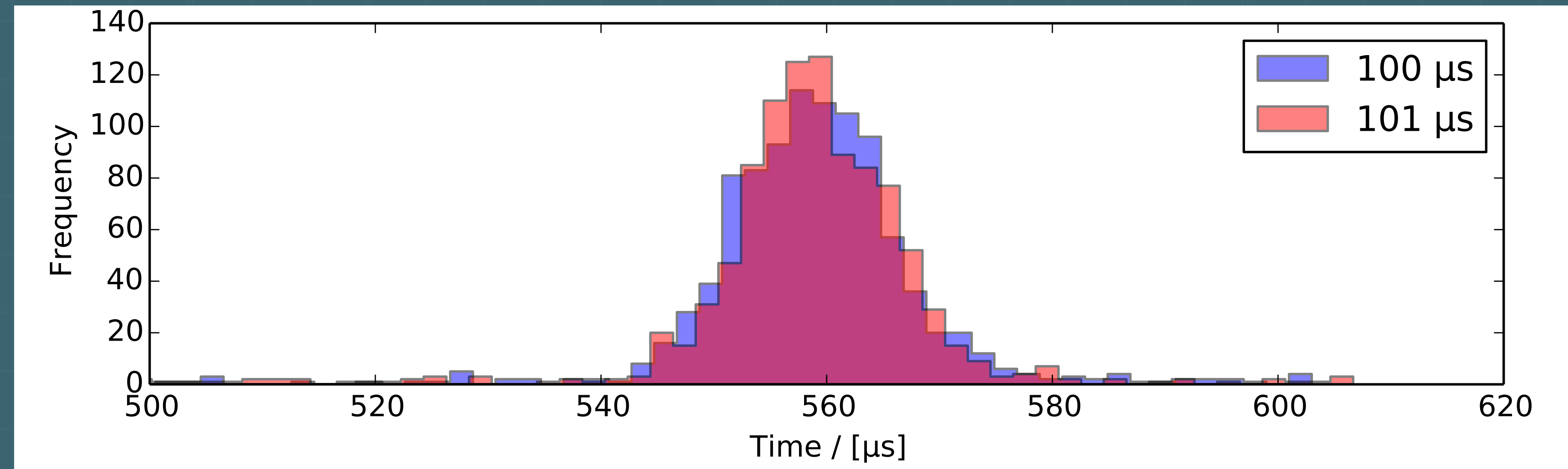


1,000 Repetitions

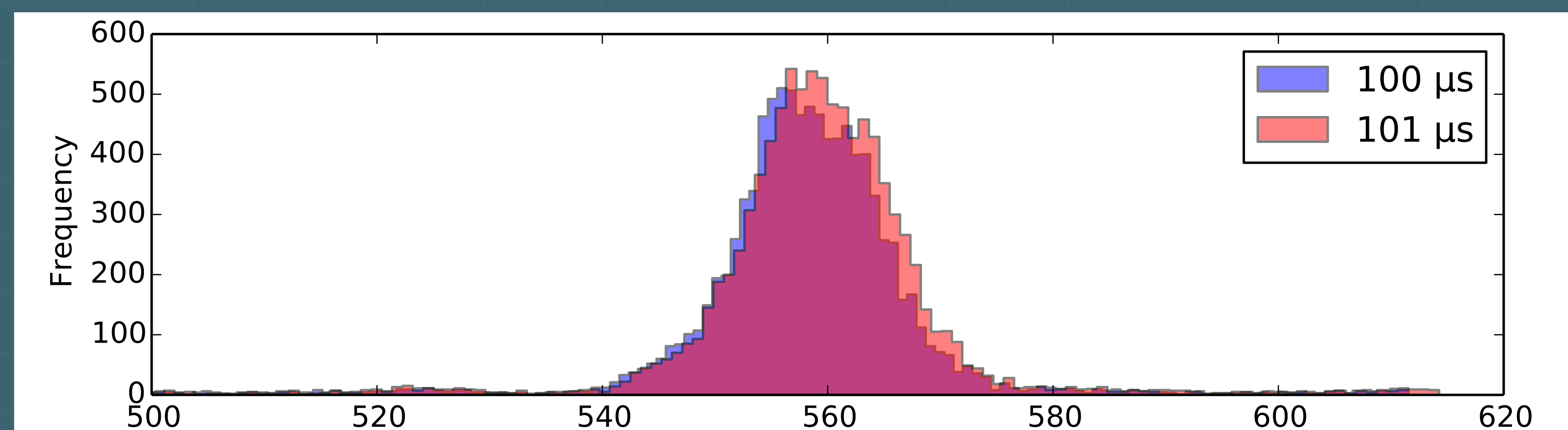
matasano SECURITY

Timing Resolution: LAN

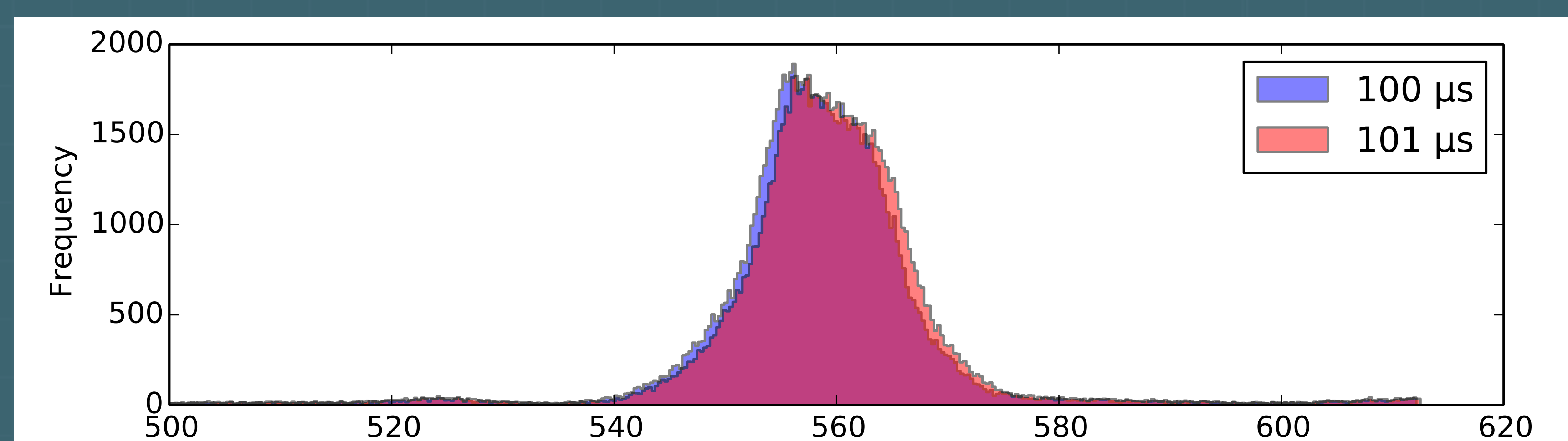
1,000 Repetitions



10,000 Repetitions



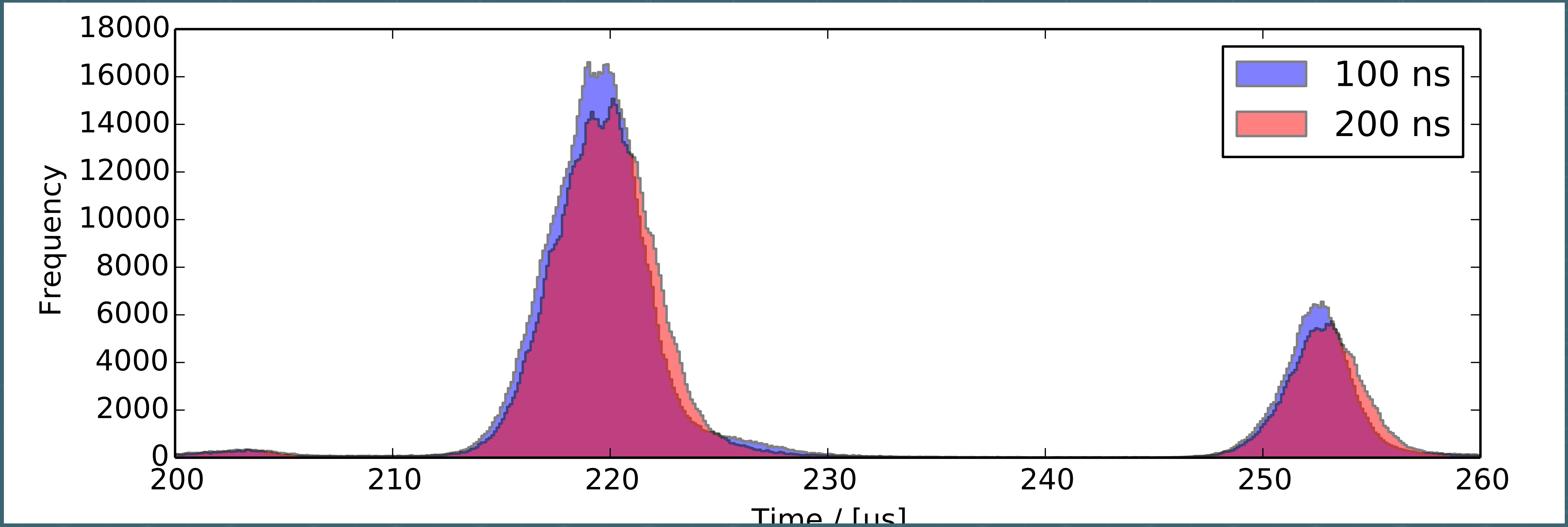
100,000 Repetitions



matasano SECURITY

Timing Resolution: LAN Limit

- 100 ns difference clear
- < 100 ns inconsistent

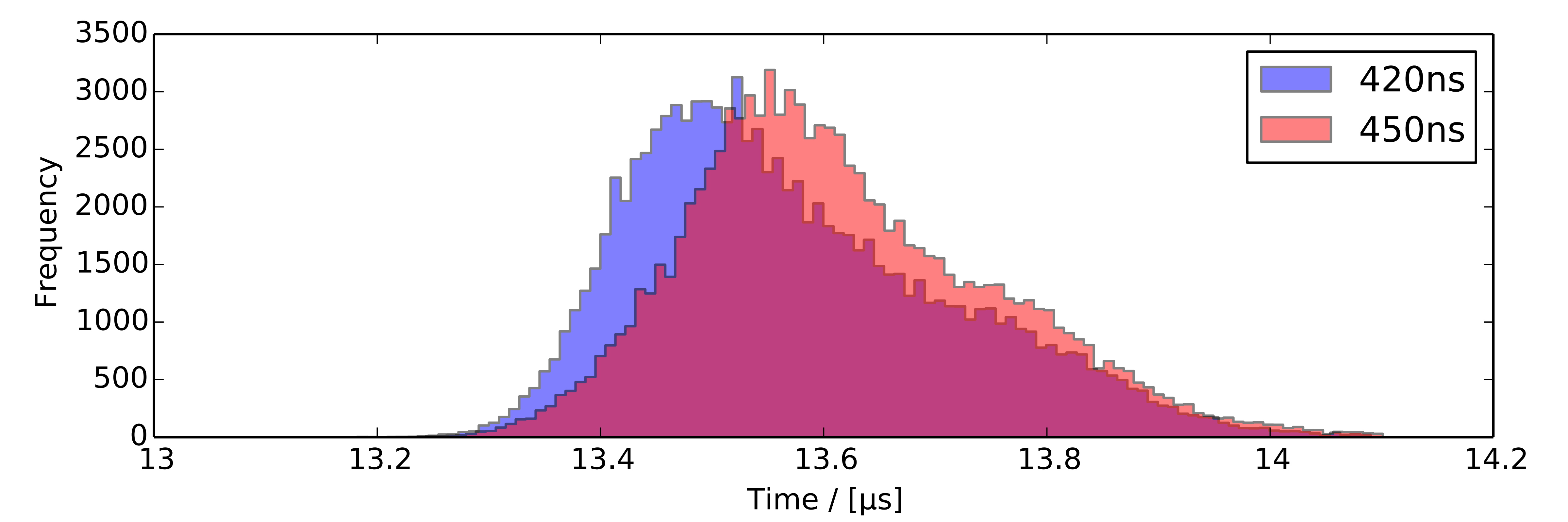


1,000,000

matasano SECURITY

Timing Resolution: Loopback

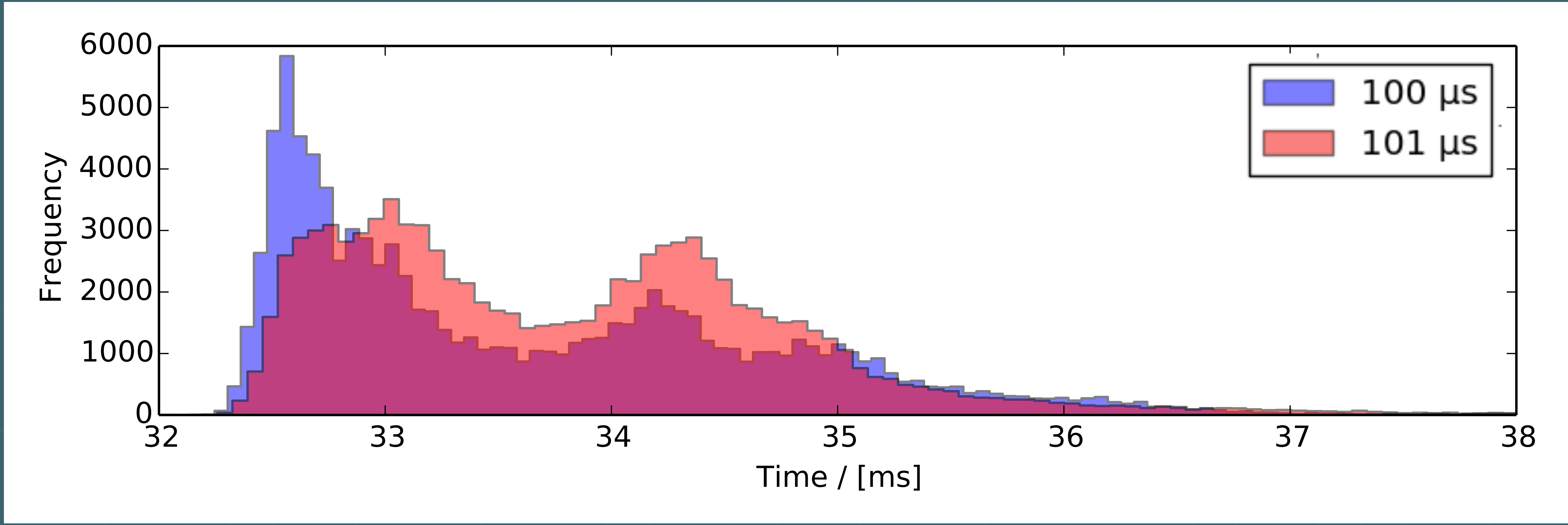
- Better than 30 ns



100,000

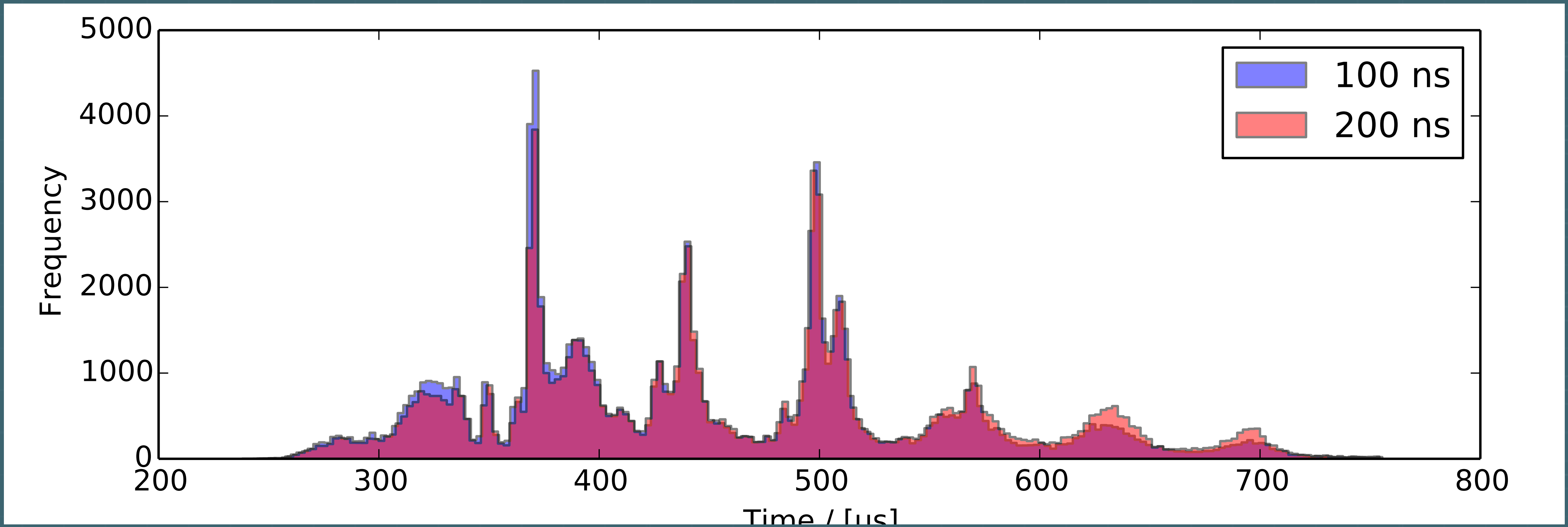
matasano SECURITY

Timing Resolution: WAN Limit



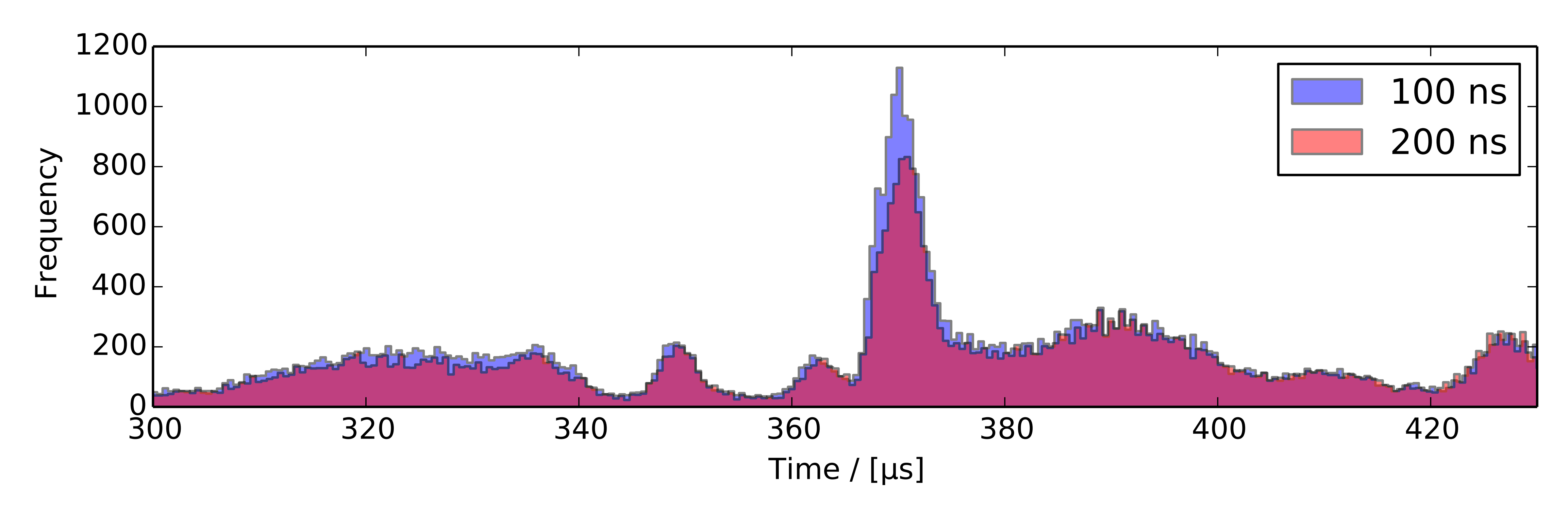
100,000

Timing Resolution: EC2 Limit



100,000

Timing Resolution: EC2 Limit

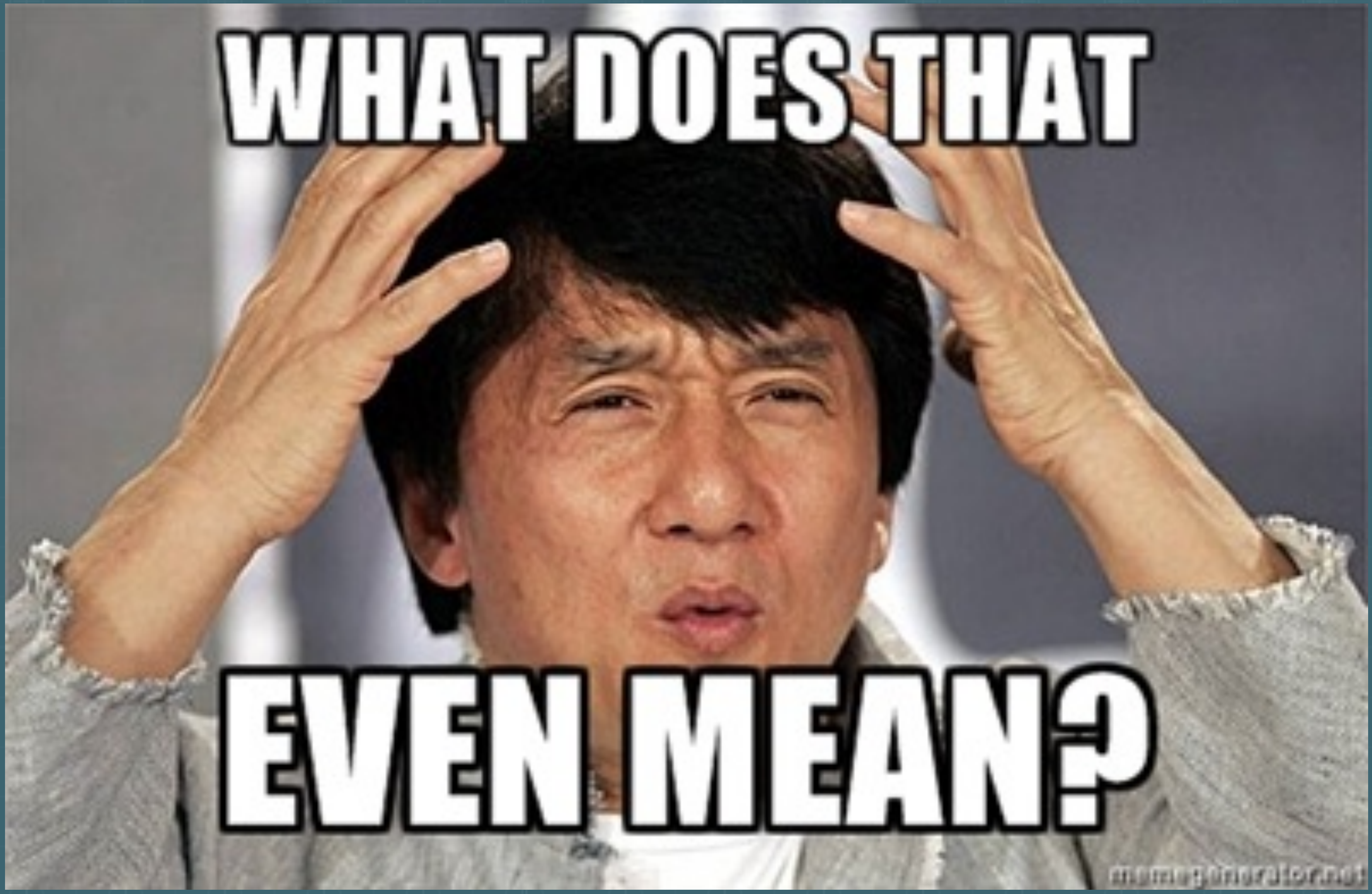


100,000

Overview of Results

	1 ms	1 μs	100 ns	< 100 ns
Loopback	✔	✔	✔	✔
LAN	✔	✔	✔	⊘
EC2	✔	✔	✔	⊘
WAN	✔	✔	? ⊘	⊘

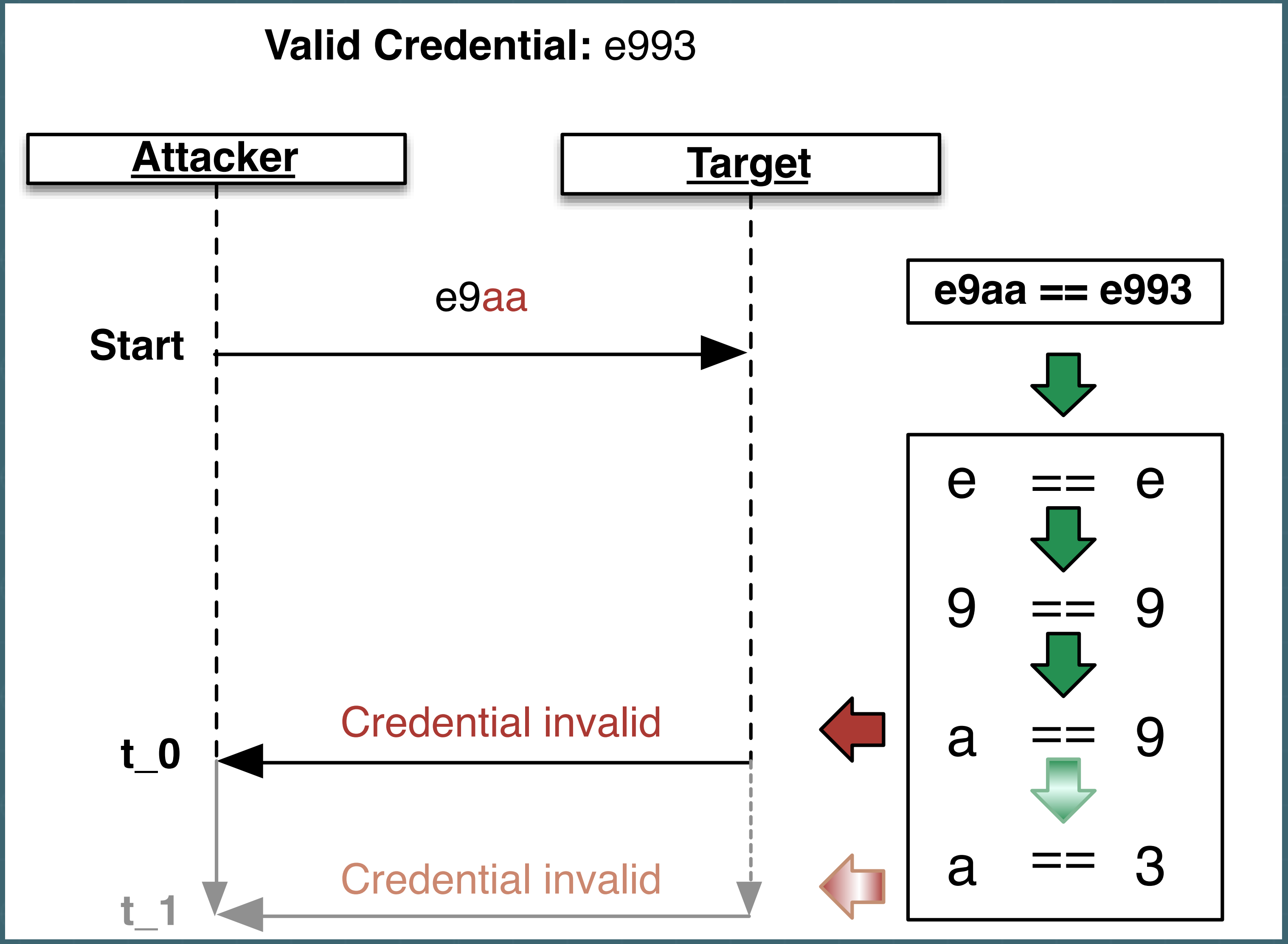
Impact on Real-world Applications



Timing Attacks in Practice

String comparison

- Most string comparison return early
  - Leaks timing information about which byte differed

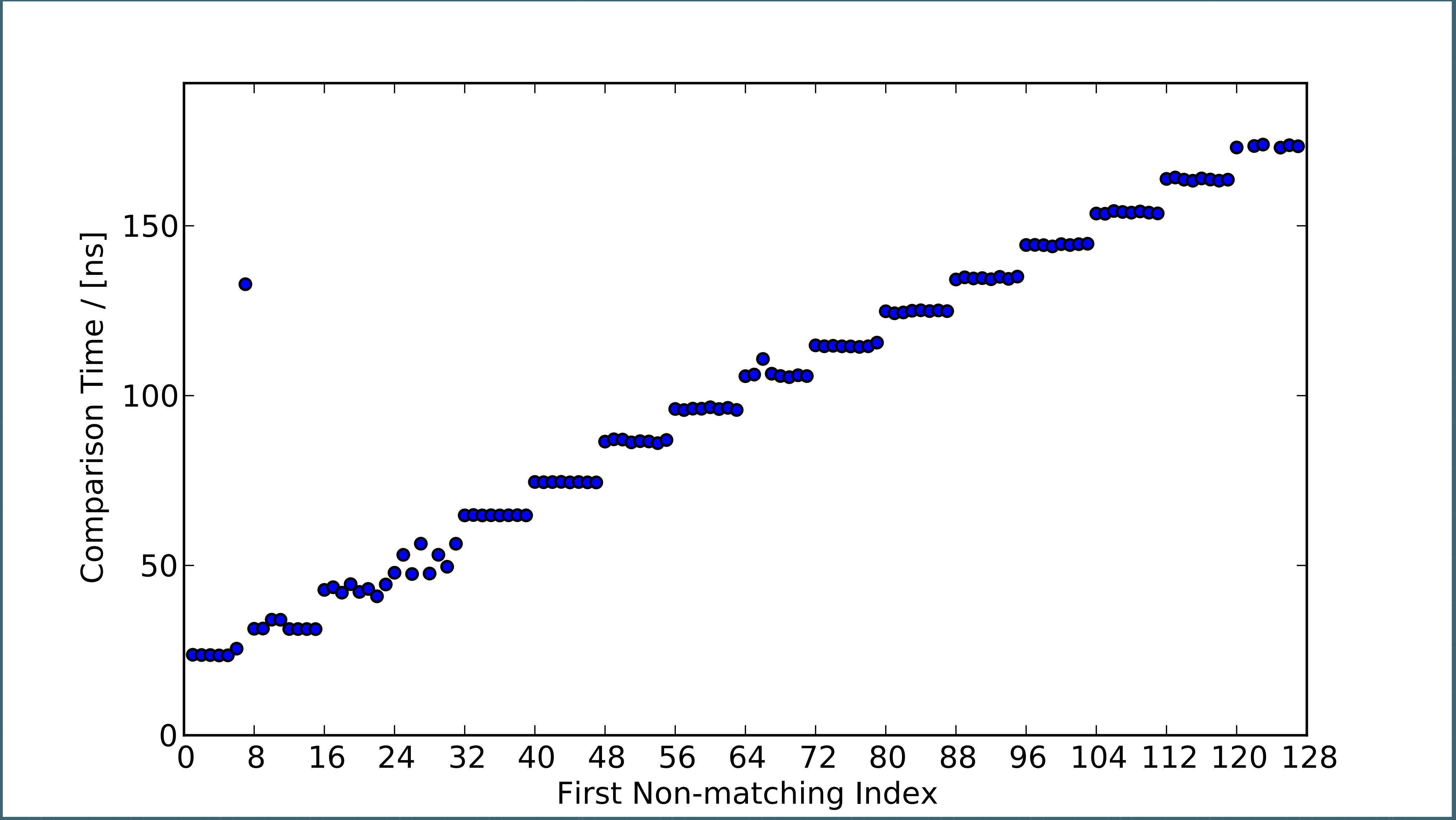


String comparison

- Introduced when attacker-controlled data is compared to a secret
- Commonly prone to timing attacks:
  - HMACs (e.g., session state)
  - Web API keys
  - OAuth token checks
  - Middleware authentication
- Exploitable remotely?

String Comparison: Conclusions

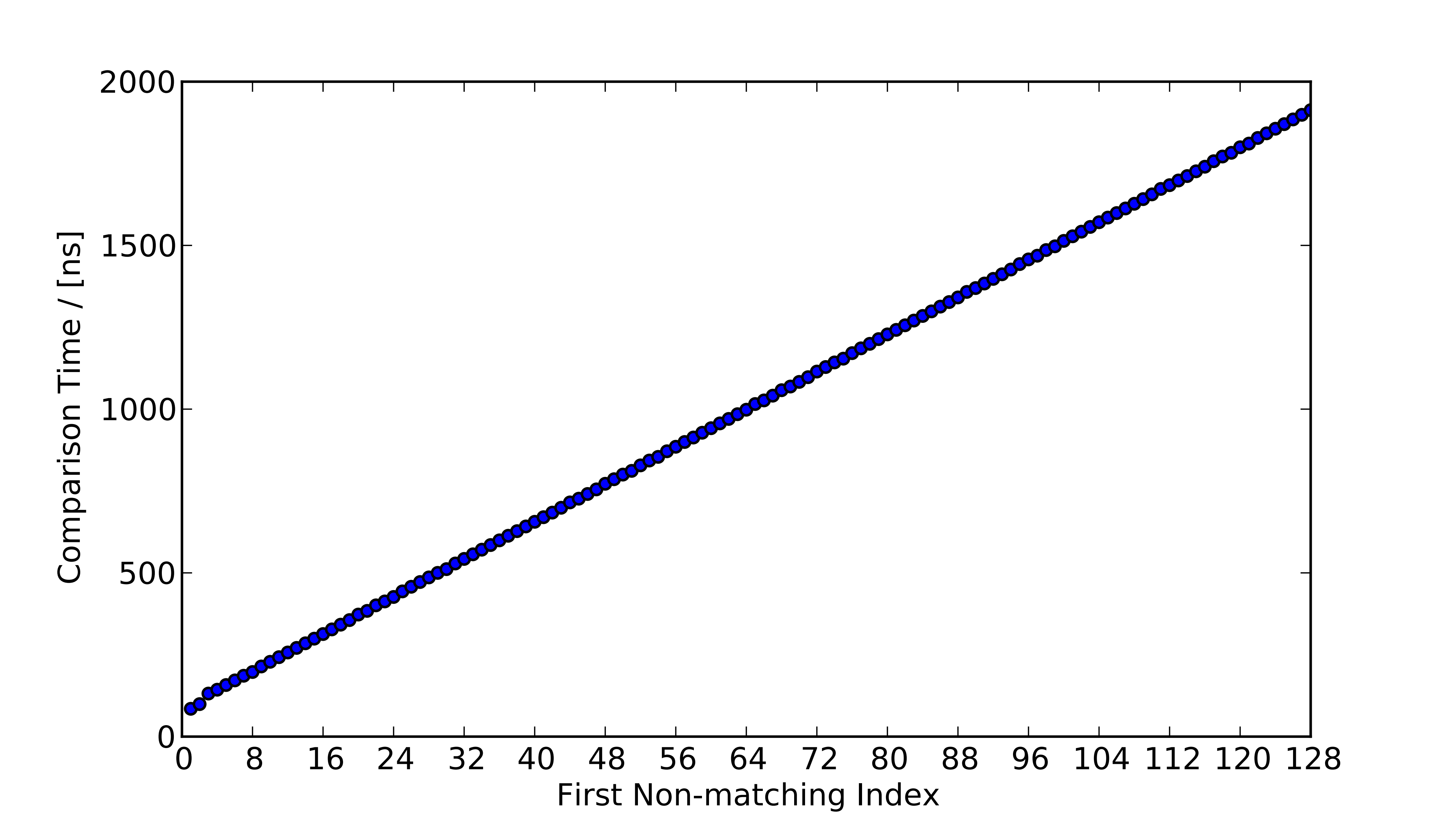
- Most 64-bit OSes compare 8 bytes at a time!
  - http://rdist.root.org/2010/08/05/optimized-memcmp-leaks-useful-timing-differences/



matasano SECURITY

Internet of Things

- BeagleBone Black: 1 GHz ARM Cortex-A8
  - Java benchmarks put it within reach, exit on first byte:



Microbenchmarks (in nanoseconds)

DIFFICULT

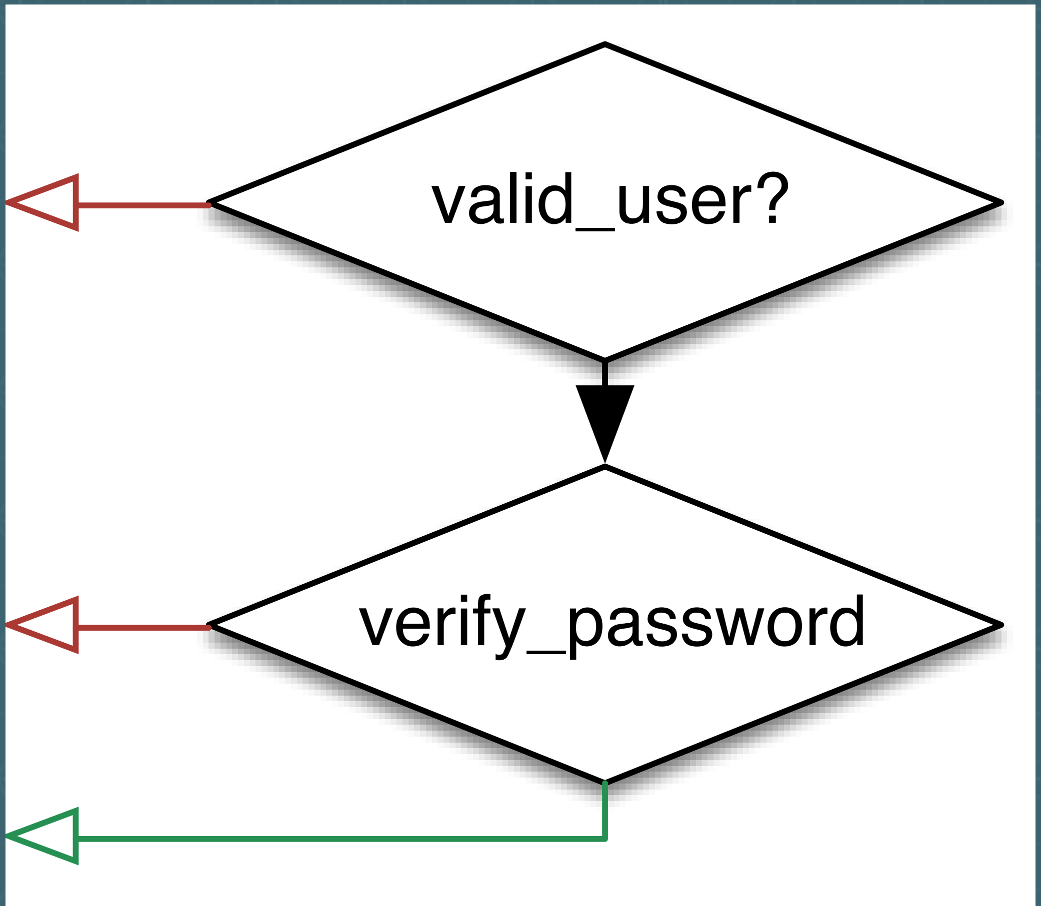
Language	Function	Lawson 2010*	i5-3210M 2.50GHz	Cortex-A8 1GHz
		per byte	*per word*	*per byte*
C	*memcmp*	0.719	0.243	1.37
C	*strcmp*	-	0.41	4.04
Ruby	*str ==*	0.840	0.36	1.75
Python	*str ==*	1.400	0.224	1.48
Java	*String.equals*	40.594	7.65	18.91

- Resolution < differences of multiple bytes
- **Remote exploitation highly unlikely in practice!**

* Hardware: AMD Athlon X2 2.7 GHz

Branching

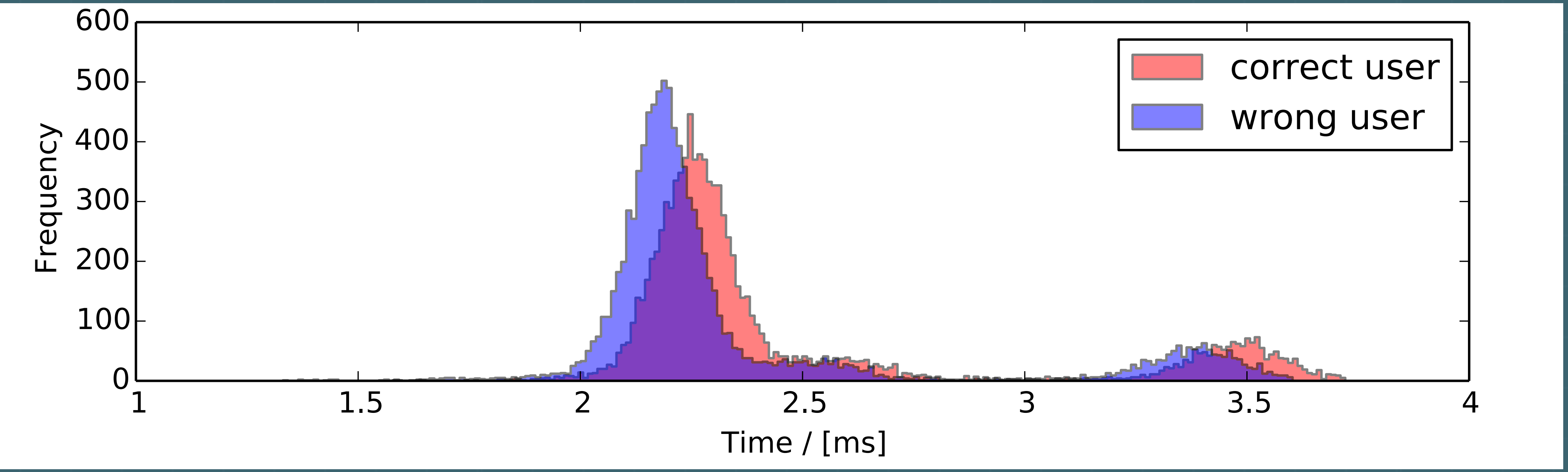
- Different code path based on secret state
- Timing difference depends on application
- Which operation performed in each code path?



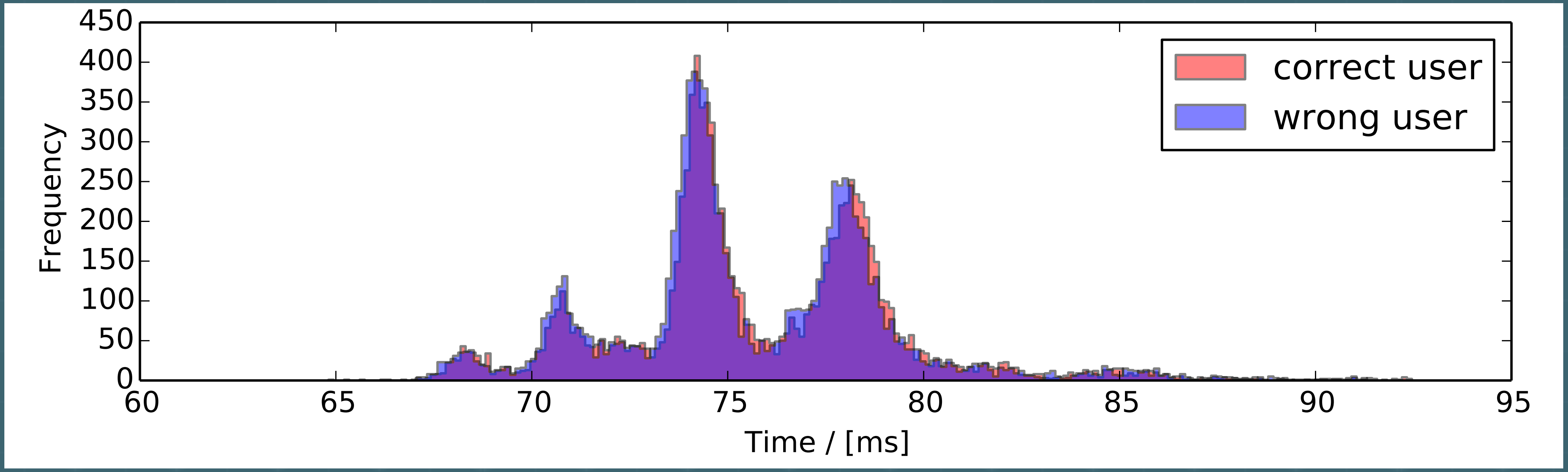
Branching

- User enumeration (SHA-256)
  - (Not a SHA-256 attack!)

LAN

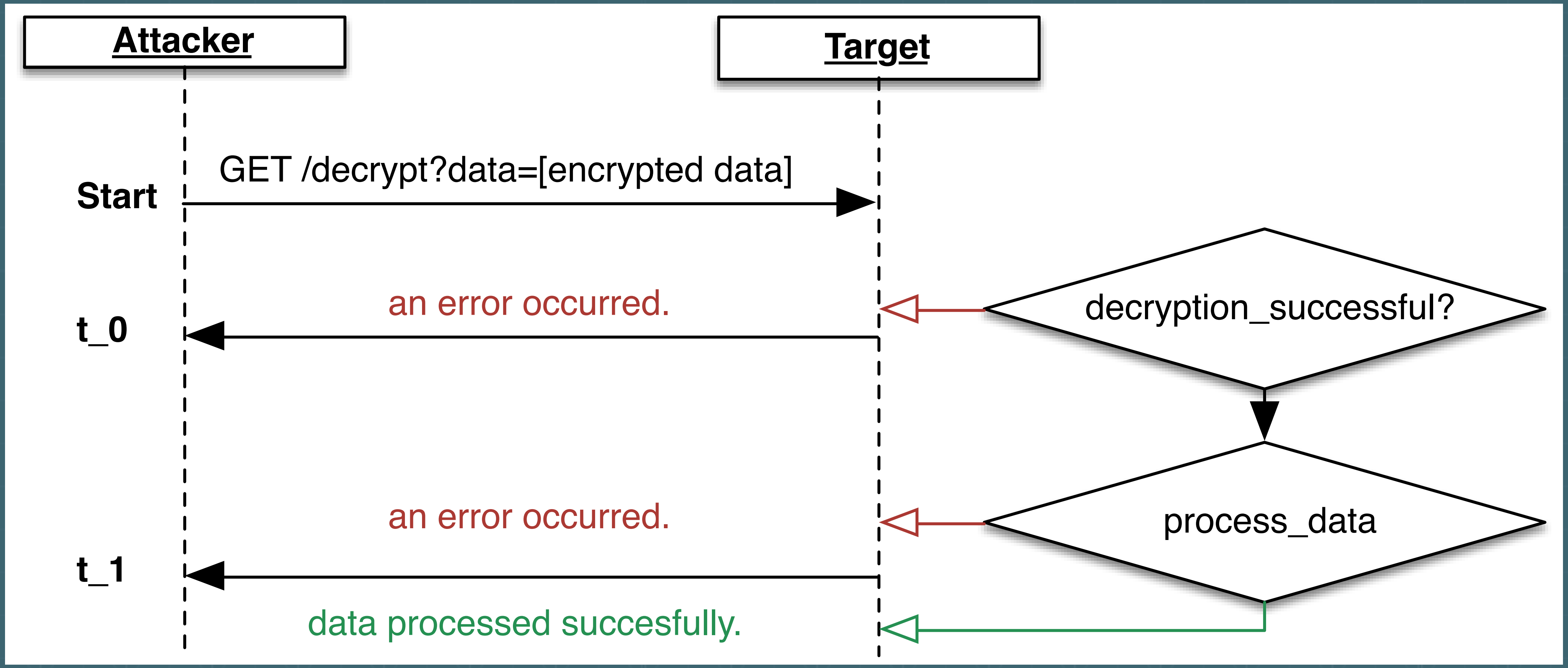


WAN



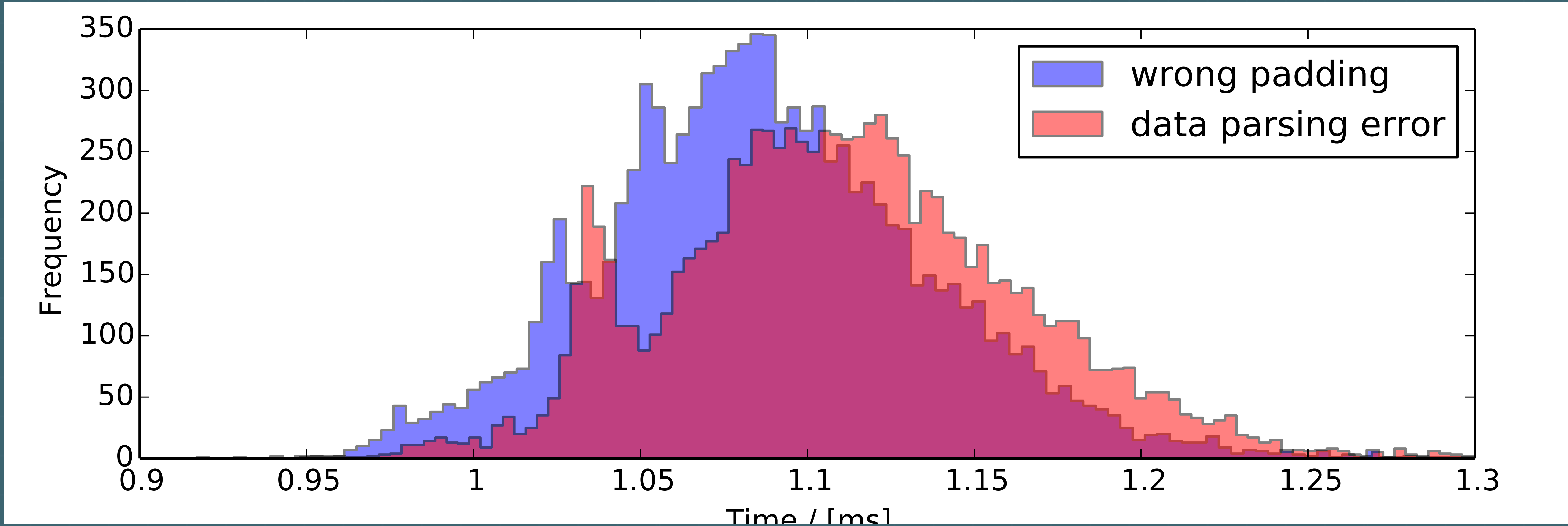
Time-Based Padding Oracle

- AES CBC Padding Oracle
- Distinguish
  - Wrong Padding
  - Other Processing Error



Time-Based Padding Oracle

- Perform SQLite query when decrypt successful
  - Actual difference depends on application!



DEMO: Time-Based CBC Padding Oracle

matasano SECURITY

Take Away: Microbenchmarks

- Computing performance continues to improve
  - Comparison-based vulnerabilities difficult to exploit.
- Branching-based often feasible
- Embedded systems at greater risk
  - Java on ARM a feasible target
  - Attacking string-comparison on Arduino realistic.

matasano SECURITY

Preventing timing attacks

- Ensure sensitive operations take constant time
  - Analyze for branching side-channels
  - This is hard!
- Use constant time comparison functions
  - See our white paper
- Best practices
  - Throttle or lock out misbehaving clients
  - Monitor for failed requests

Future Plans

- More empirical studies
- Implement (feasible!) attacks
- Jitter changes over time
  - Alternate long and short measurements

Send bug reports, feature / pull requests!

Thanks!

Questions?

https://github.com/dmayer/time_trial

http://matasano.com/research/

- **Daniel A. Mayer**
  mayer@cysec.org
  @DanlAMayer

- **Joel Sandin**
  jsandin@matasano.com