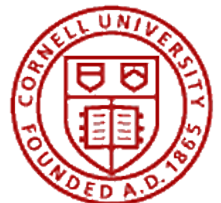


Predictive Mitigation of Timing Channels in Interactive Systems

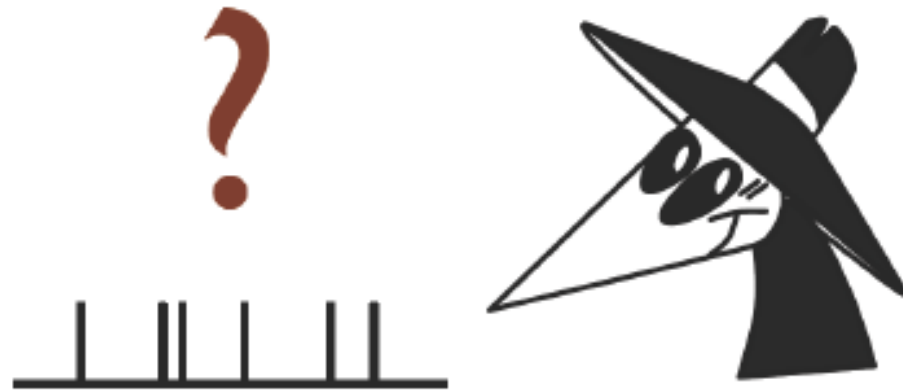
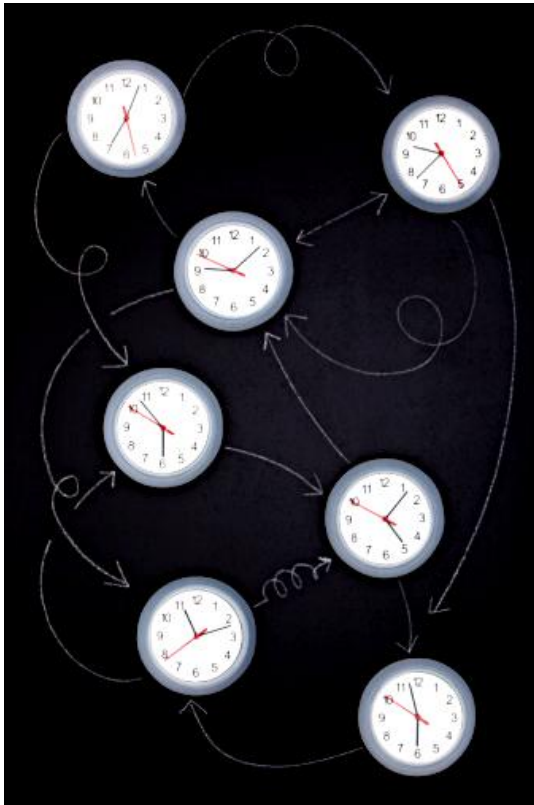
Danfeng Zhang, Aslan Askarov, Andrew C. Myers

Cornell University

CCS 2011



Timing Channels



Hard to detect and prevent

Timing Channels: Examples

- Cryptographic timing attacks [Kocher 96, Brumley&Boneh 05, Osvik et. al. 06]
 - RSA, AES keys are leaked by decryption time
- Cross-site timing attacks [Bortz&Boneh 07]
 - Load time of a web-page reveals login status, as well as the size and contents of shopping cart
- Use as covert channels [Meer&Slaviero 07]
 - Transmit confidential data by controlling response time, e.g., combined with SQL injection
- **Timing channels are big threats to security!**

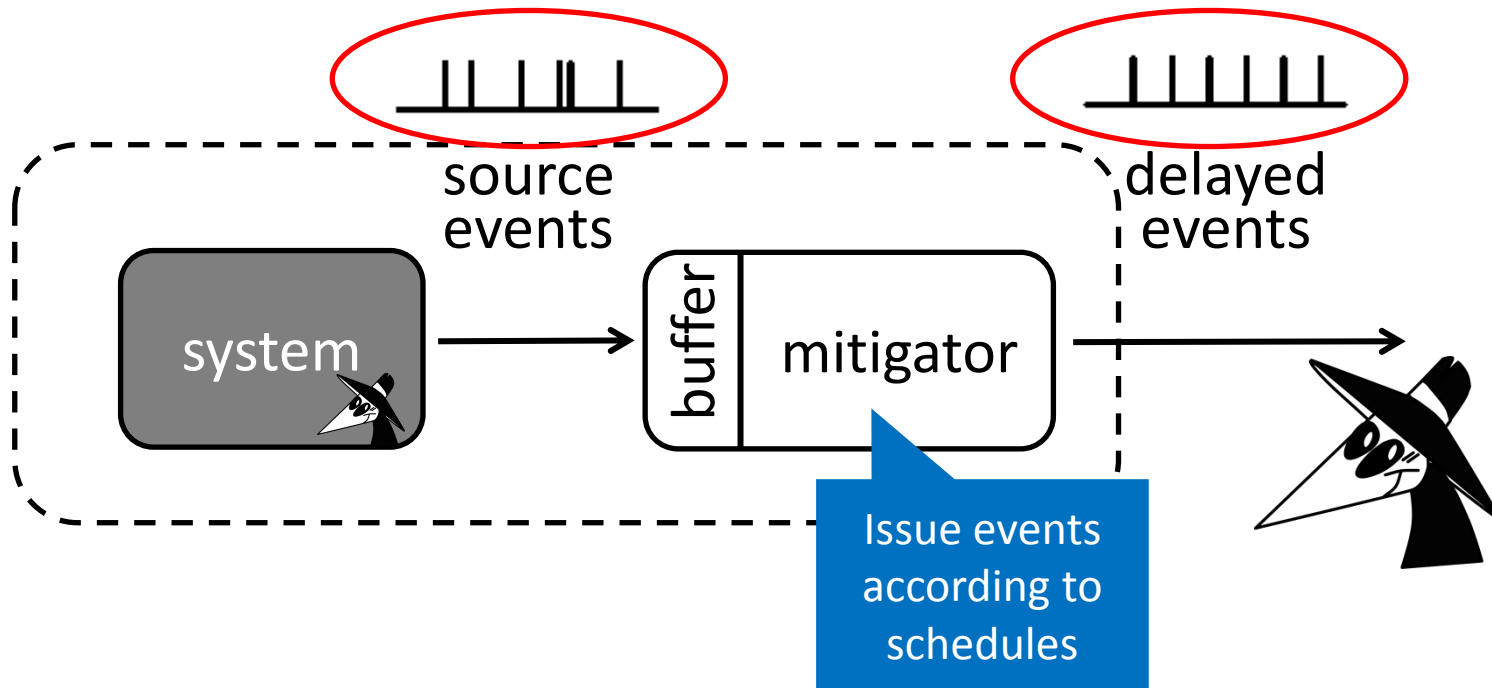
Timing Channel Mitigation

- Limitations of known approaches
 - Delay to the worst-case execution time – bad performance
 - Add random delays – linear leakage
 - Input blinding – specialized to cryptography
- Our solution:
 - Asymptotically logarithmic leakage
 - Effective in practice
 - Applies to general computation

Outline

- Background on predictive black-box mitigation (CCS'10)
- Predictive mitigation for interactive systems (e.g., web services)
 - Prediction with public information
 - Generalized penalty policy & leakage analysis
 - Composition of mitigators
- Evaluation

Background: Predictive Black-Box Mitigation of Timing Channels (CCS'10)

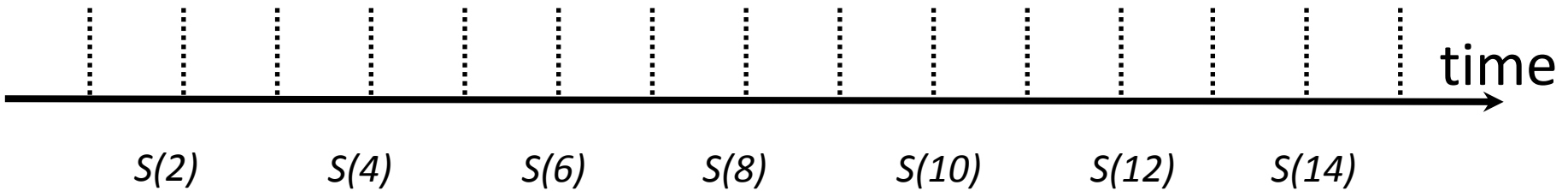


Strong attacker model: timing of source events may be controlled

Example: Doubling

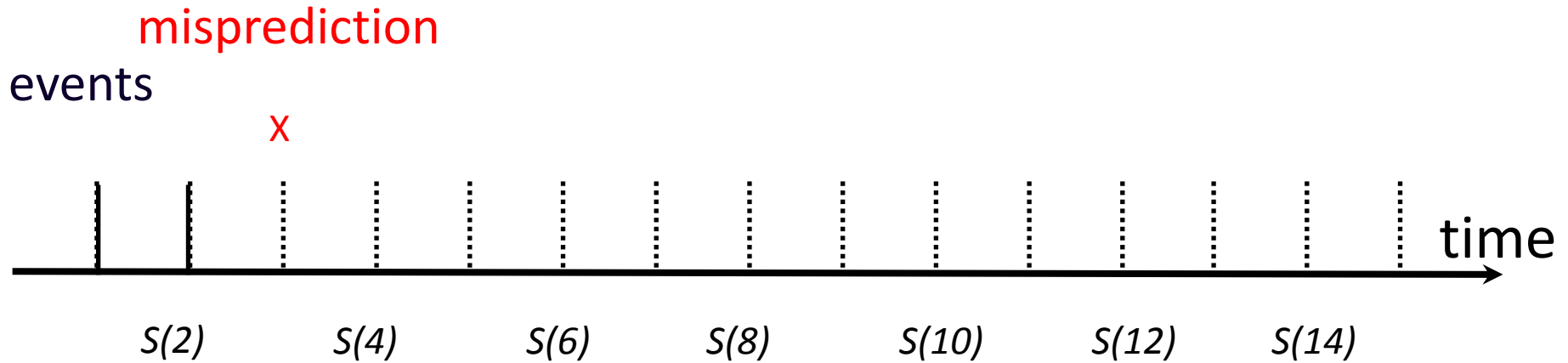
When mitigator expects to deliver events

predictions



Mitigator starts with a fixed schedule S
 $S(i)$ – prediction for i -th event

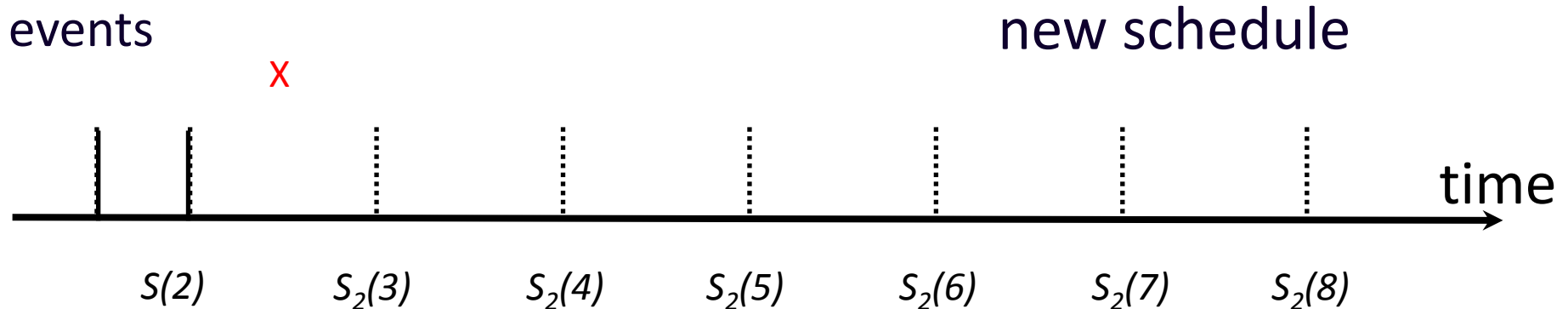
Example: Doubling



When event comes before or at the prediction – delay the event

little information leaked

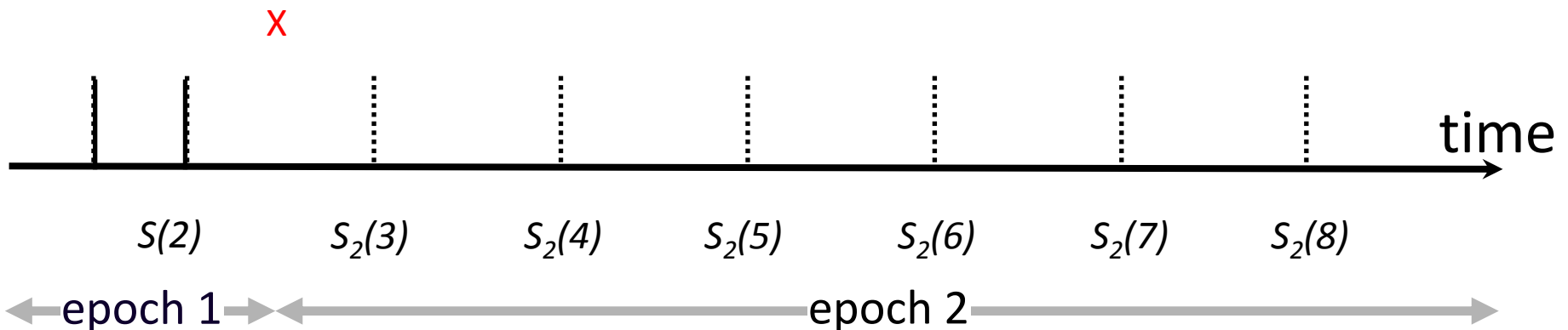
Example: Doubling



Adversary observes mispredictions **information leaked!**
New fixed schedule S_2 **penalizes** the event source

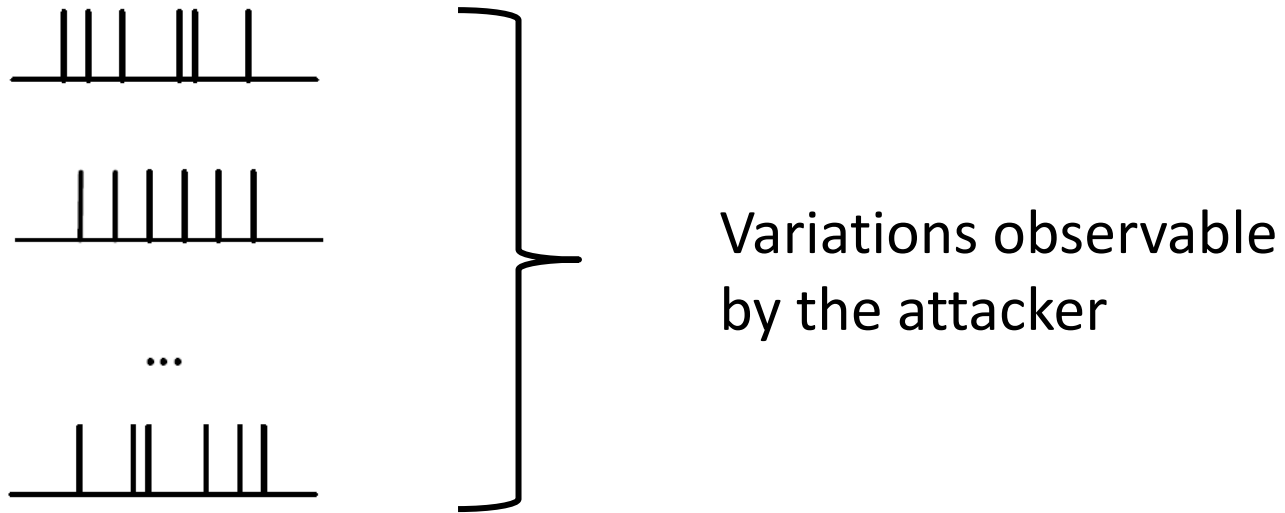
Example: Doubling

Epoch: period of time during which mitigator meets all predictions



Little information
leaked in each epoch

Leakage & Variations



- Leakage measurement (log of # timing variations)
 - Also bounds
 - mutual information (Shannon entropy)
 - min-entropy

Important Features

- Information leaks via *mispredictions*!
- General class of timing mitigators
 - Doubling scheme

$$\text{Leakage} \leq O(\log^2 T) \text{ bits}$$

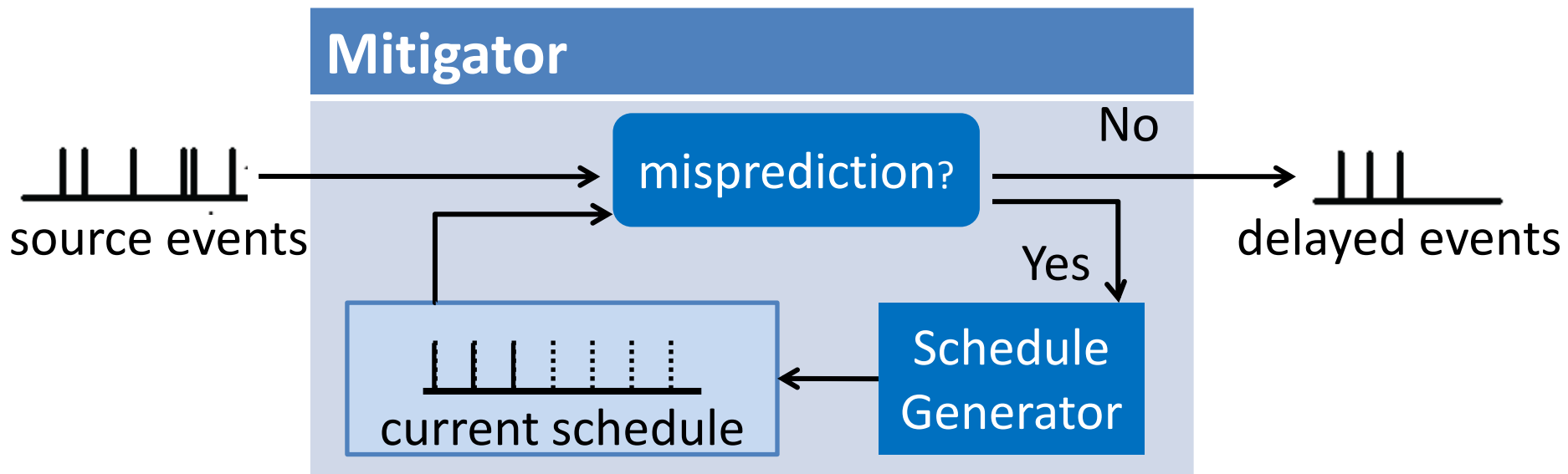
- Adaptive transitions
- ...

Outline

- Background on predictive black-box mitigation (CCS'10)
- Predictive mitigation for interactive systems (e.g., web services)
 - Prediction with public information
 - Generalized penalty policy & leakage analysis
 - Composition of mitigators
- Evaluation

Insight: Use Public Information

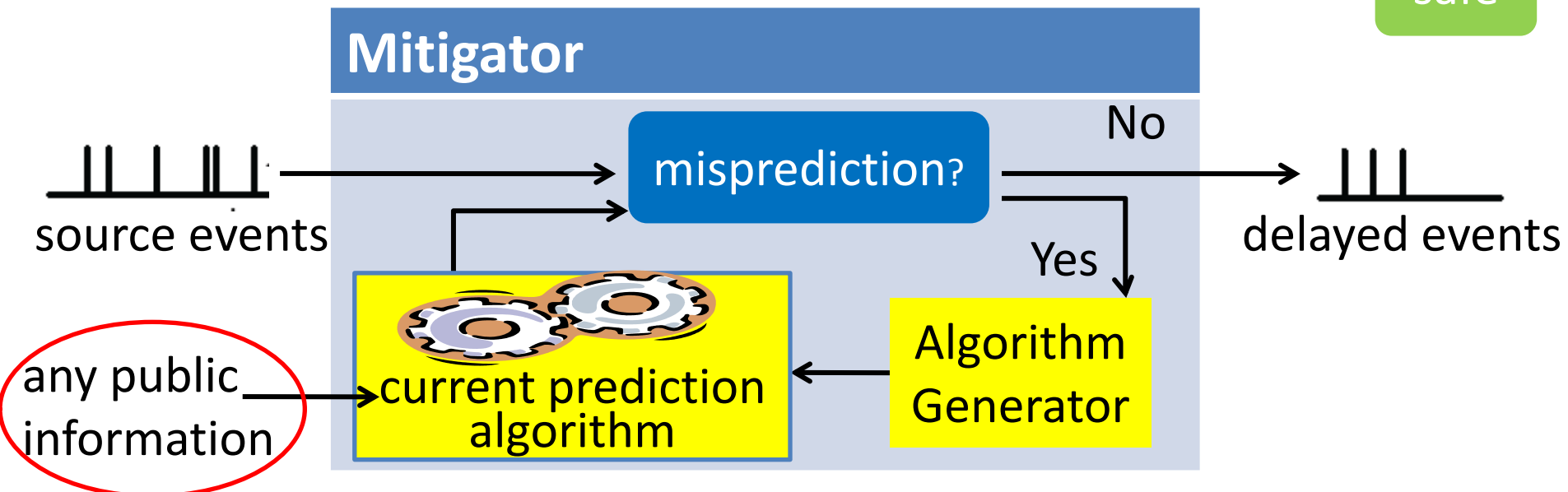
- Previous black-box model safe
 - No misprediction: events delivered according to schedule
 - Misprediction: entire schedule is **statically** determined (difficult for interactive systems)



Insight: Use Public Information

- Previous black-box model
 - Schedule is **dynamically** calculated by prediction algorithm
 - No misprediction: schedule is deterministic given public info.
 - Misprediction: select a new prediction algorithm

safe

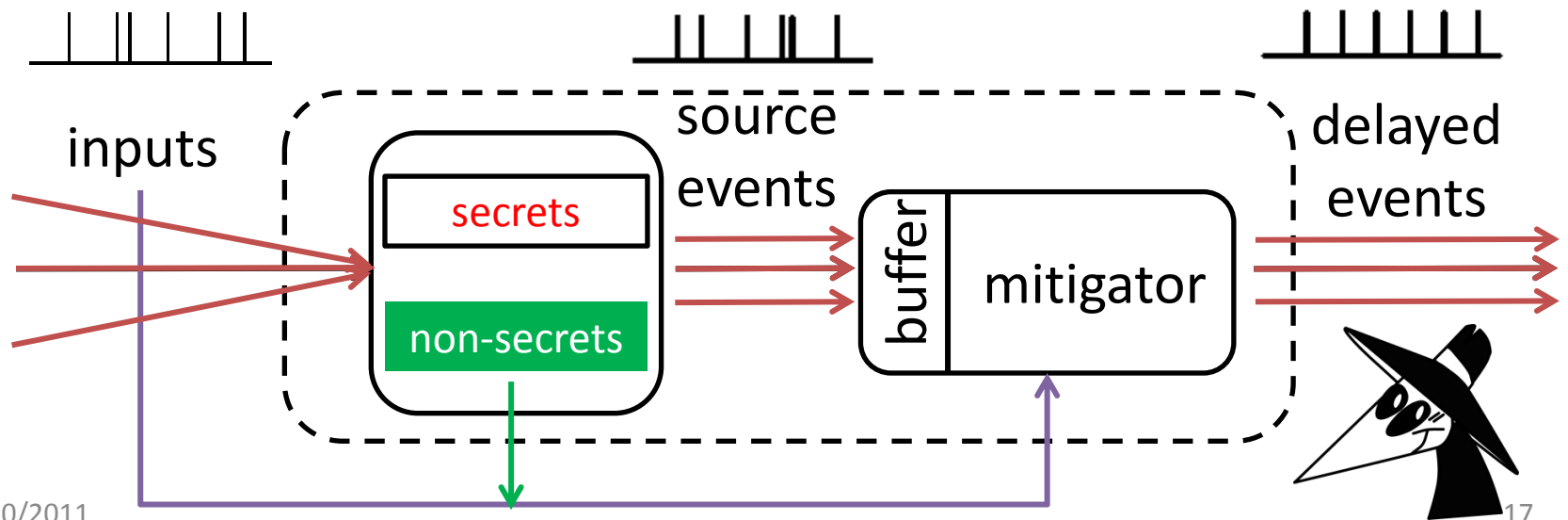


Outline

- Background on predictive black-box mitigation (CCS'10)
- Predictive mitigation for interactive systems (e.g., web services)
 - Prediction with public information
 - Generalized penalty policy & leakage analysis
 - Composition of mitigators
- Evaluation

Public Information

- Public information in interactive systems
 - *Request types*: public payloads in requests, such as URLs
 - www.example.com/index.html vs. www.example.com/background.gif
 - *Public information in system*: such as input times
 - *Concurrency model*



Prediction with Public Information



prediction algorithm

Epoch #

- Prediction for request type r : $p(N, r)$
- Schedule (output time) for i^{th} event in the N^{th} epoch

- Single thread

Start time

Handling time

$$S_N(i) = \max(inp_i, S_N(i-1)) + p(N, r_i)$$

- Multiple, concurrent threads

- Calculated in similar ways

Schedules are computed dynamically within each epoch using only *public* information

Information Leakage

- Information still leaks via *mispredictions*!
- Formal result

of epochs

of messages

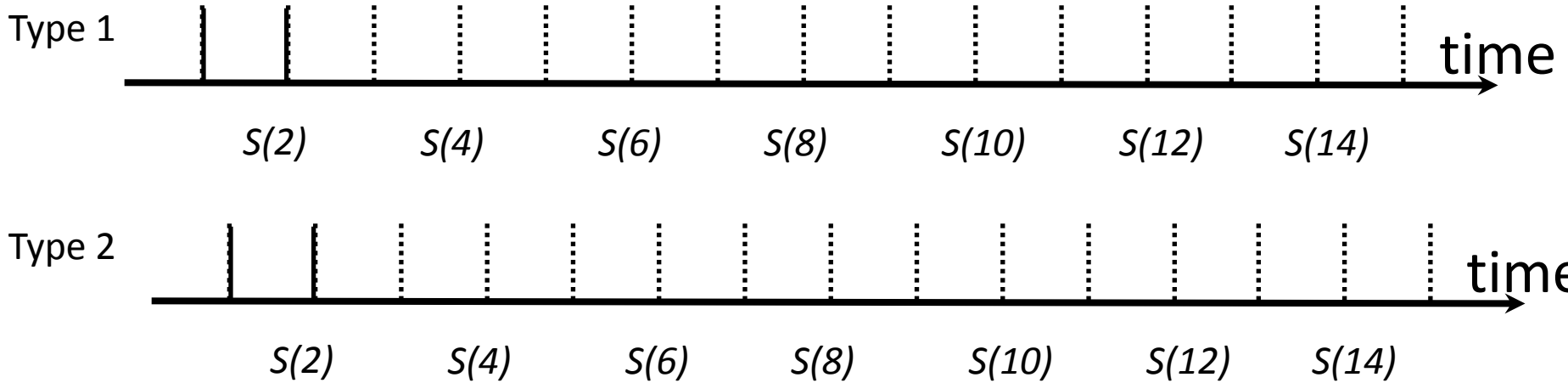
Leakage $\leq N \times \log(M + 1)$ bits

Outline

- Background on predictive black-box mitigation (CCS'10)
- Predictive mitigation for interactive systems (e.g., web services)
 - Prediction with public information
 - Generalized penalty policy & leakage analysis
 - Composition of mitigators
- Evaluation

Penalty Policy – What

x misprediction



Penalty policy

- **Which** type should be penalized?
- **How much** it should be penalized?

Penalty Policy – Why

of epochs

of messages

$$\text{Leakage} \leq N \times \log(M + 1) \text{ bits}$$

- Concurrency & request types also bring ***new threats***
- Request types are penalized separately (local penalty policy)
 - Attacker controls the timing of R request types
 - **N is proportional to R**
- Penalize all request types (global penalty policy)
 - Performance is bad

Grace Period Penalty Policy

- Better trade off?
 - Information leaks via *mispredictions!*
 - “Well-behaved” types
 - Few mispredictions, leak little information
 - Share little penalty from other types
- L -level grace period policy
 - type i is penalized by other types only when it triggers more than L mispredictions

Leakage Analysis

- Difficult for general penalty policies
 - Influences between request types
 - Different predictions
 - Need to consider *all* possible input sequences
- Principled way of bounding total leakage
 - Transform into optimization problem with R constraints
(formal proof & details provided in paper)

request types

Leakage Analysis

Leakage

- Global: $O(\log T \times \log M)$
 - running time
 - # of messages
- Local: $O(R \times \log T \times \log M)$
 - # request types
- Grace period: $O(\log T \times \log M)$
 - *Better trade-off*

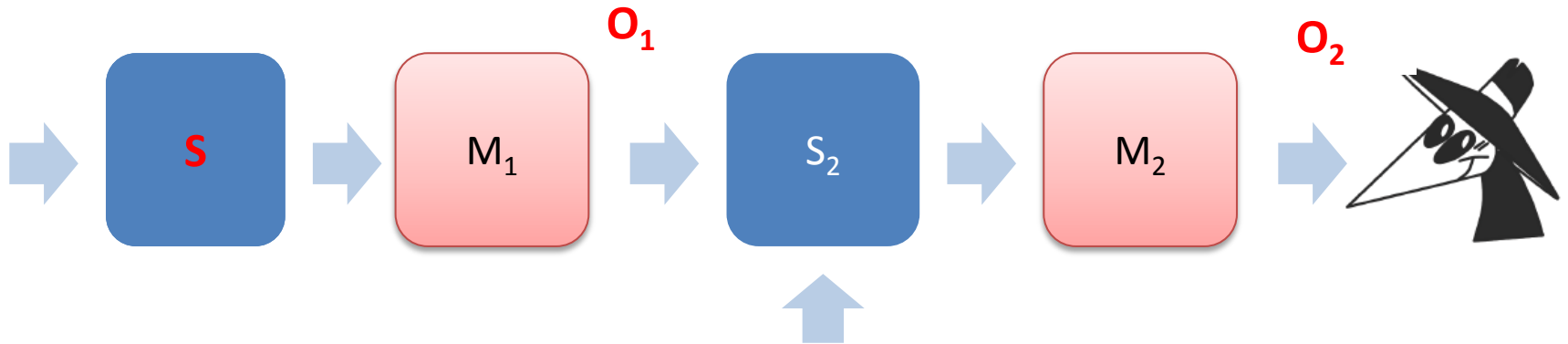
Outline

- Background on predictive black-box mitigation (CCS'10)
- Predictive mitigation for interactive systems (e.g., web services)
 - Prediction with public information
 - Generalized penalty policy & leakage analysis
 - **Composition of mitigators**
- Evaluation

Composition of Mitigators

- Security guarantee on an interactive system that is
 - composed of mitigated subsystems
- Decompose complicated systems into 2 gadgets
 - Sequential
 - Parallel

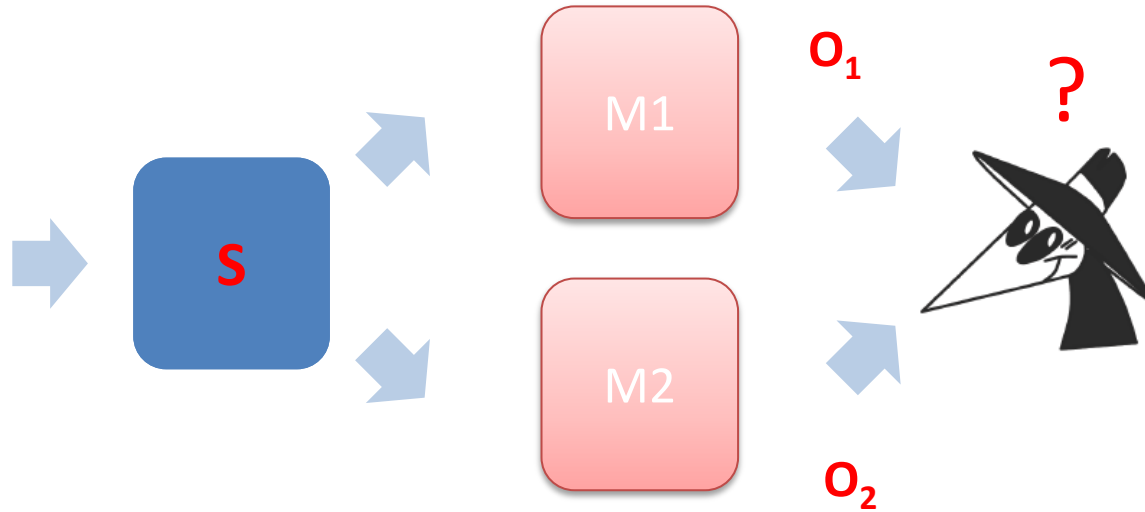
Sequential Case



Theoretical results

- Leakage in $O_2 \leq$ Leakage in O_1
- Valid for
 - mutual information
 - min-entropy

Parallel Case



Theoretical result

- Leakage in O_1 and $O_2 \leq$ Leakage in $O_1 +$ Leakage in O_2

Outline

- Background on predictive black-box mitigation (CCS'10)
- Predictive mitigation for interactive systems (e.g., web services)
 - Prediction with public information
 - Generalized penalty policy & leakage analysis
 - Composition of mitigators
- **Evaluation**

Evaluation

Real-world web applications (with HTTP(S) proxy)



Real-world applications

The screenshot shows the Cornell University Department of Computer Science website. It features a navigation menu with links for Information, Events, Admissions, People, Courses, Undergrad Program, Graduate Program, and Research. A central image shows students at a 2011 Game Design Initiative Showcase. Below the image, there are two research areas: Systems and Networking (operating systems, distributed computing, networking, wireless systems, security and protection) and Computational Biology (sequence analysis, structural classification, gene network dynamics).

Proxy

The screenshot shows the Microsoft Office Outlook Web Access security screen. It includes a security warning with two radio button options: 'This is a public or shared computer' (selected) and 'This is a private computer'. There is a checkbox for 'Use Outlook Web Access Light' which is checked. Below the security options are input fields for 'User name:' and 'Password:', and a 'Log On' button.

Client

Mitigating Proxy

Demo

Evaluation

Measurements

- **Performance**: round-trip latency from the client side
- **Security**: leakage bounds in bits

Experiments with Web Applications

Parameters

- 5-level grace period policy
- Doubling scheme
- Various request types
 - TYPE/HOST
 - HOST+URLTYPE
 - TYPE/URL

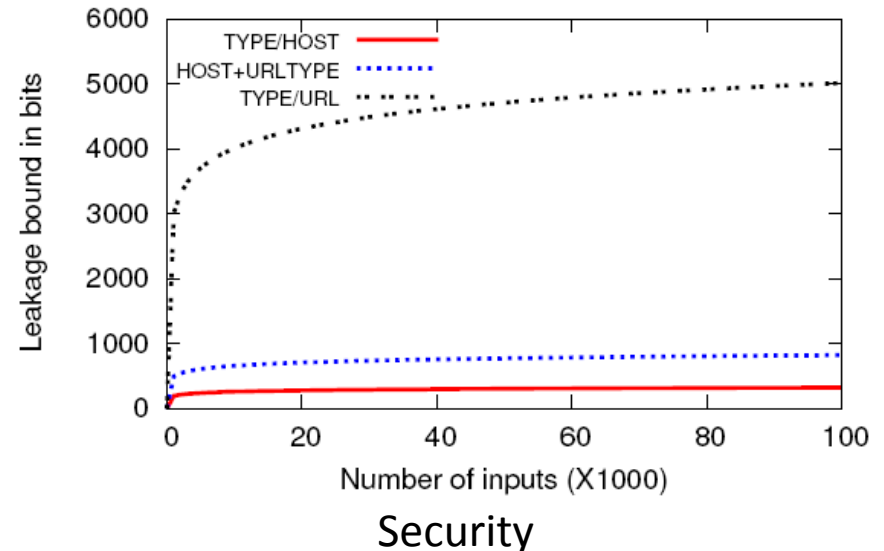
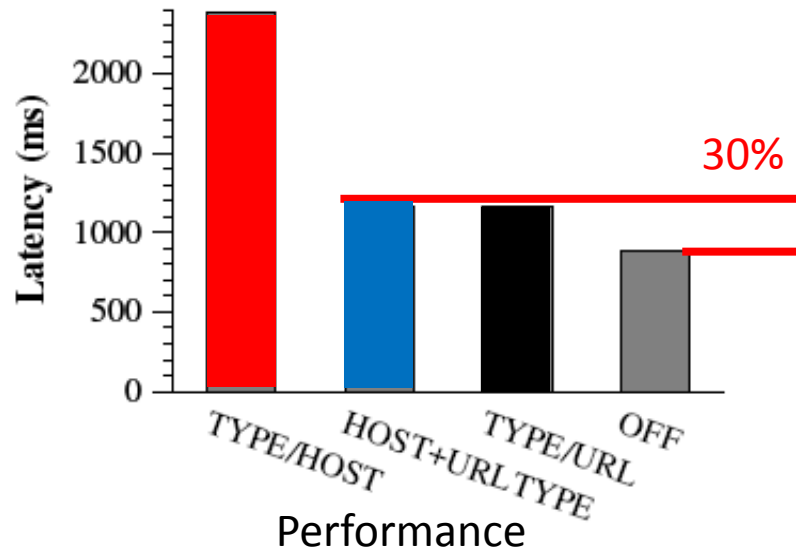
Experiments with Web Applications

Mitigating department homepage via HTTP

(49 different requests)

– Predictive mitigation gains good balance (**HOST+URLTYPE**)

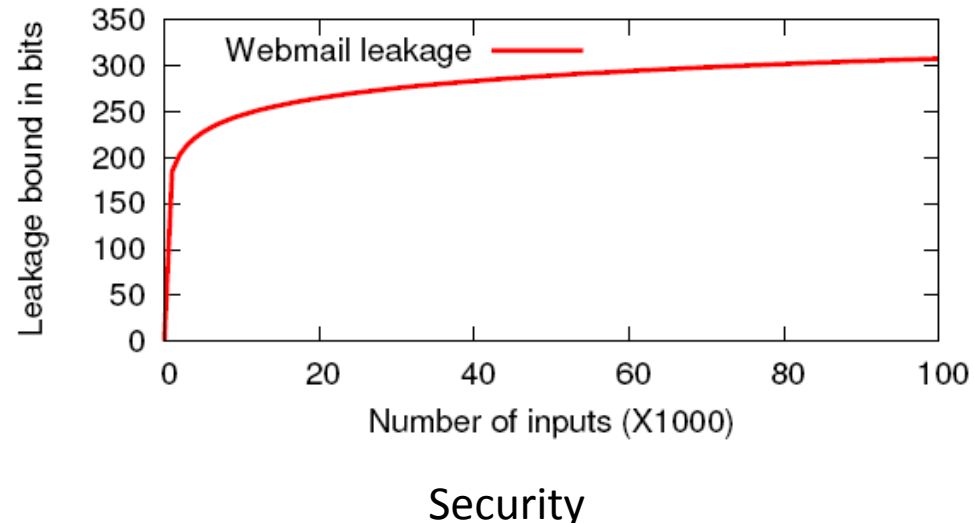
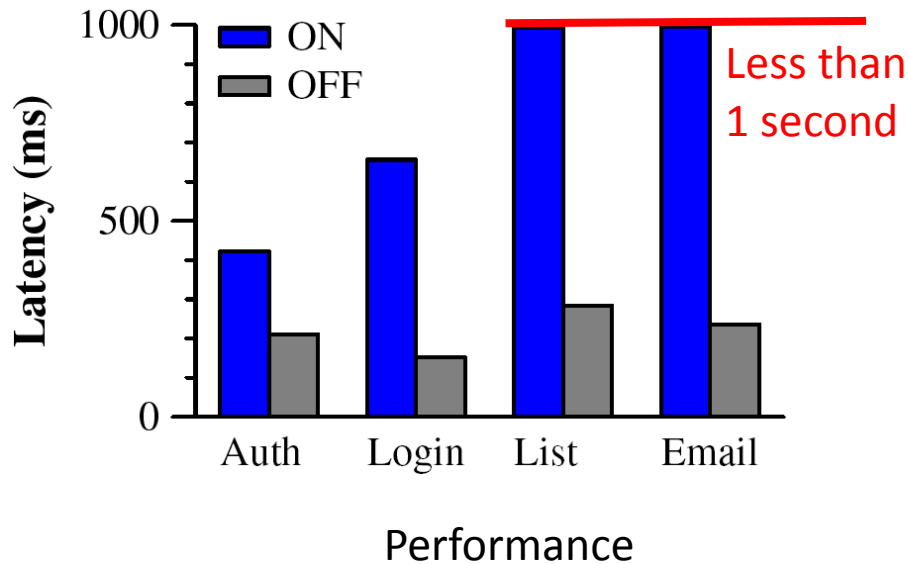
- About **30%** latency overhead
- At most **850bits** for 100,000 inputs



Experiments with Web Applications

Mitigating department webmail server via HTTPS

- Secure-sensitive (URL is encrypted)
- At most 300 bits for 100,000 inputs
- At most 450 bits for 32M inputs (1 input/sec for one year)



Related Work

- **Timing mitigation for cryptographic operations** [Kocher 96, Kopf & Durmuth 09, Kopf & Smith 10]
 - Assumes input blinding
- **NRL Pump/Network Pump** [Kang et. al. 93, 96]
 - Only address covert channels from input acks
 - Linear bound
- **Information theory community** [Hu 91, Giles&Hajek 02]
 - Timing mitigation based on random delays
 - Linear bound

Conclusion



Predictive mitigation of timing channels

- Generalized predictive mitigation model
More public information → better performance
- General penalty policies & leakage analysis
- Composition of mitigated systems
- Evaluation suggests this technique is practical