2nd Symposium and Bootcamp on the Science of Security (HotSoS)
April 21st, 2015

Integrity Assurance in Resource-Bounded Systems through Stochastic Message Authentication

Aron Laszka, Yevgeniy Vorobeychik, and Xenofon Koutsoukos

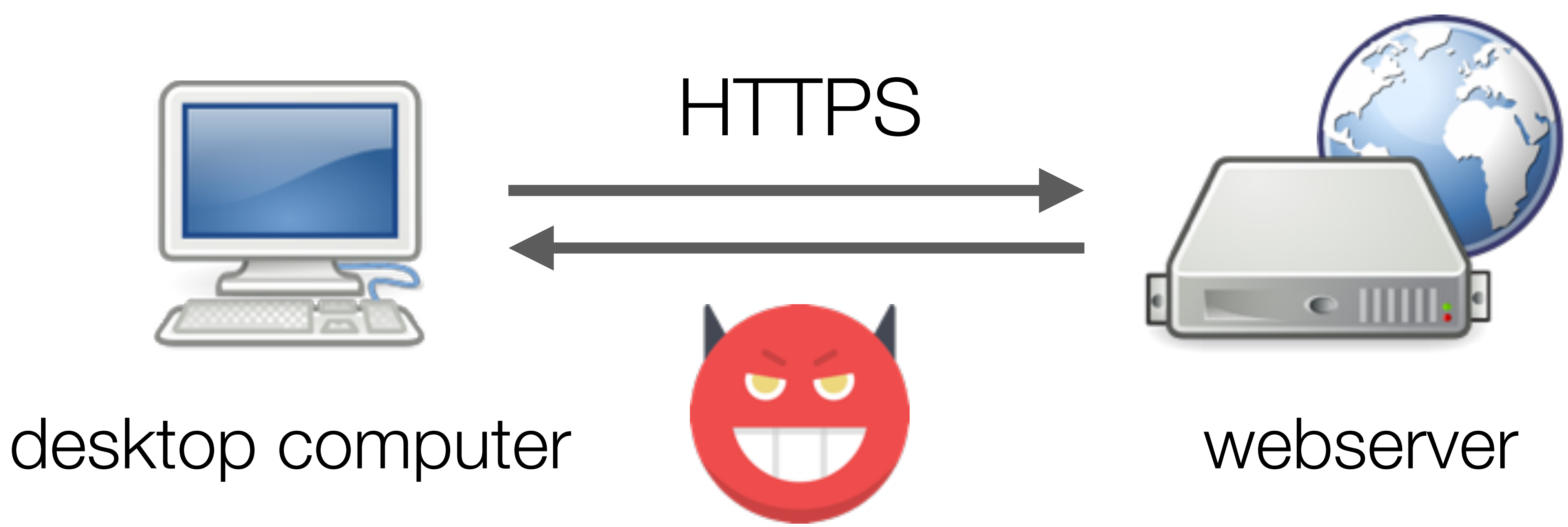
Institute for Software Integrated Systems
Department of Electrical Engineering and Computer Science
VANDERBILT UNIVERSITY

The Science of Security initiative is funded by the National Security Agency http://hot-sos.org/

Data Integrity

- Data integrity:
  assuring that data cannot be modified in an unauthorized and undetected manner

- Classic, non-resource-bounded example:



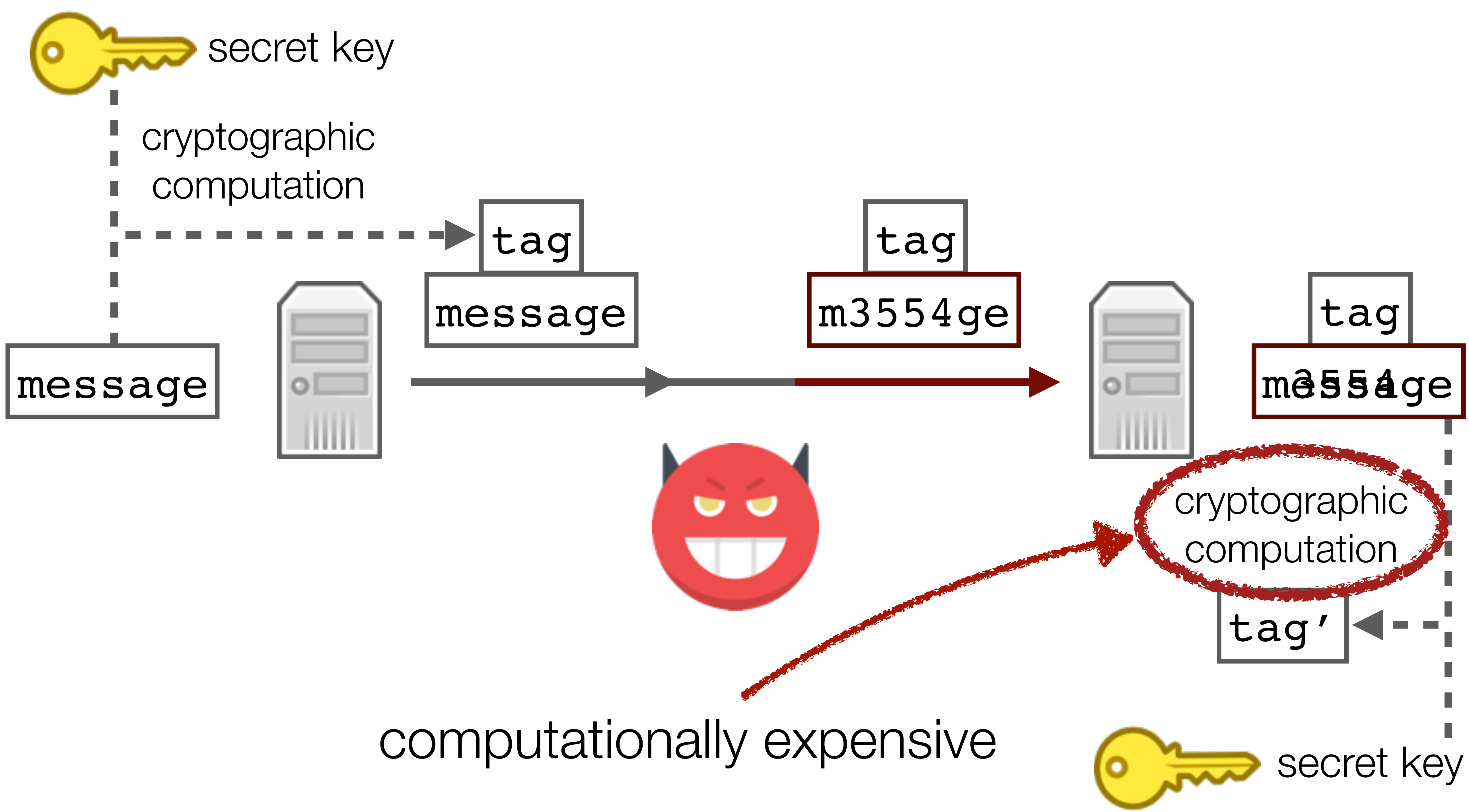
Not really an issue these days, right?

Example of Data-Tampering

Traffic monitoring: Sensys Networks VDS240

- wireless vehicle detection system based on magnetic sensors embedded in roadways
- insecure communication protocol lacks integrity protection
- attacker may cause disastrous traffic congestions

Message Authentication



Insufficient resources

messages are **not** verified

zero security

Limited amount of resources

some messages are verified

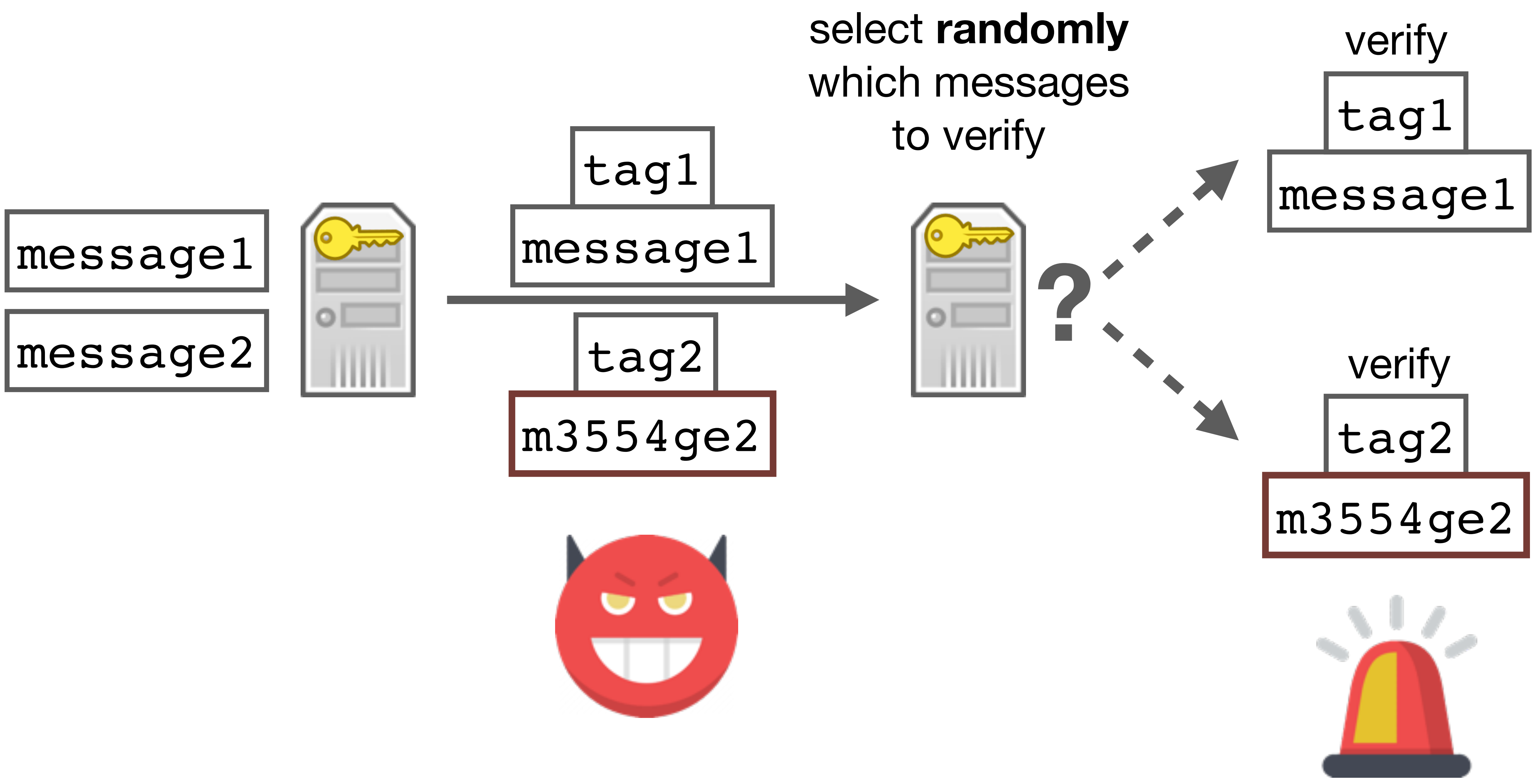
maximal achievable security

Sufficient resources

messages are verified

maximum security

Stochastic Verification



Applications

- In many scenarios, suboptimal data acquisition and control is **costly** but **not disastrous**
  - inefficient traffic control
  - incorrect smart-metering
  - …
- Resource-bounded devices
  - battery-powered devices
  - legacy devices
  - low-performance devices
  - …
- Comparison to lightweight cryptography
  - we build on well-known and widely deployed cryptographic primitives
  - our system adapts to arbitrary resource bounds

Game-Theoretic Model

"Which messages to verify?"

- Stackelberg security game with a defender and an attacker

Messages

- divided into classes
- messages of class i may cause L_i damage

1. Defender

- chooses verification probabilities p_i
- subject to computational budget constraint

$$\sum p_i T_i \leq B$$

where T_i is the cost of verifying all messages of class i

Game-Theoretic Model (contd.)

1. Defender

2. Attacker

- selects the number a_i of modified/forged messages for each class i
- knows the defender's strategy (i.e., p_i for every i)

3. Payoffs

outcome:

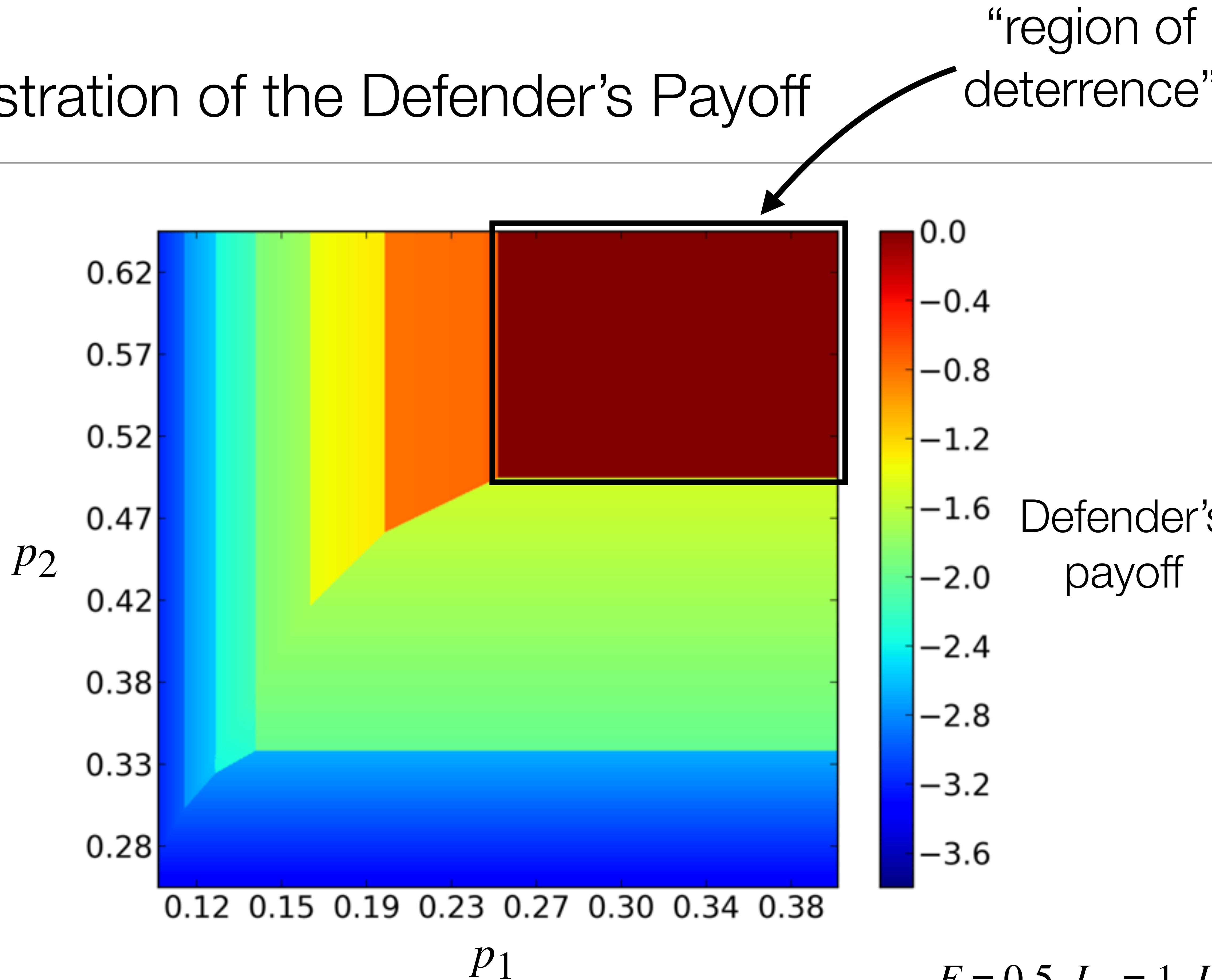
$1 - \prod(1 - p_i)^{a_i}$

attack detected:
attacker receives
punishment F

$\prod(1 - p_i)^{a_i}$

attack not detected:
defender loses /
attacker gains $\sum a_i L_i$

Illustration of the Defender's Payoff



$F = 0.5, L_1 = 1, L_2 = 3$

Deterrence Strategies

- Deterrence strategy:
  attacker's best response is not to modify any messages

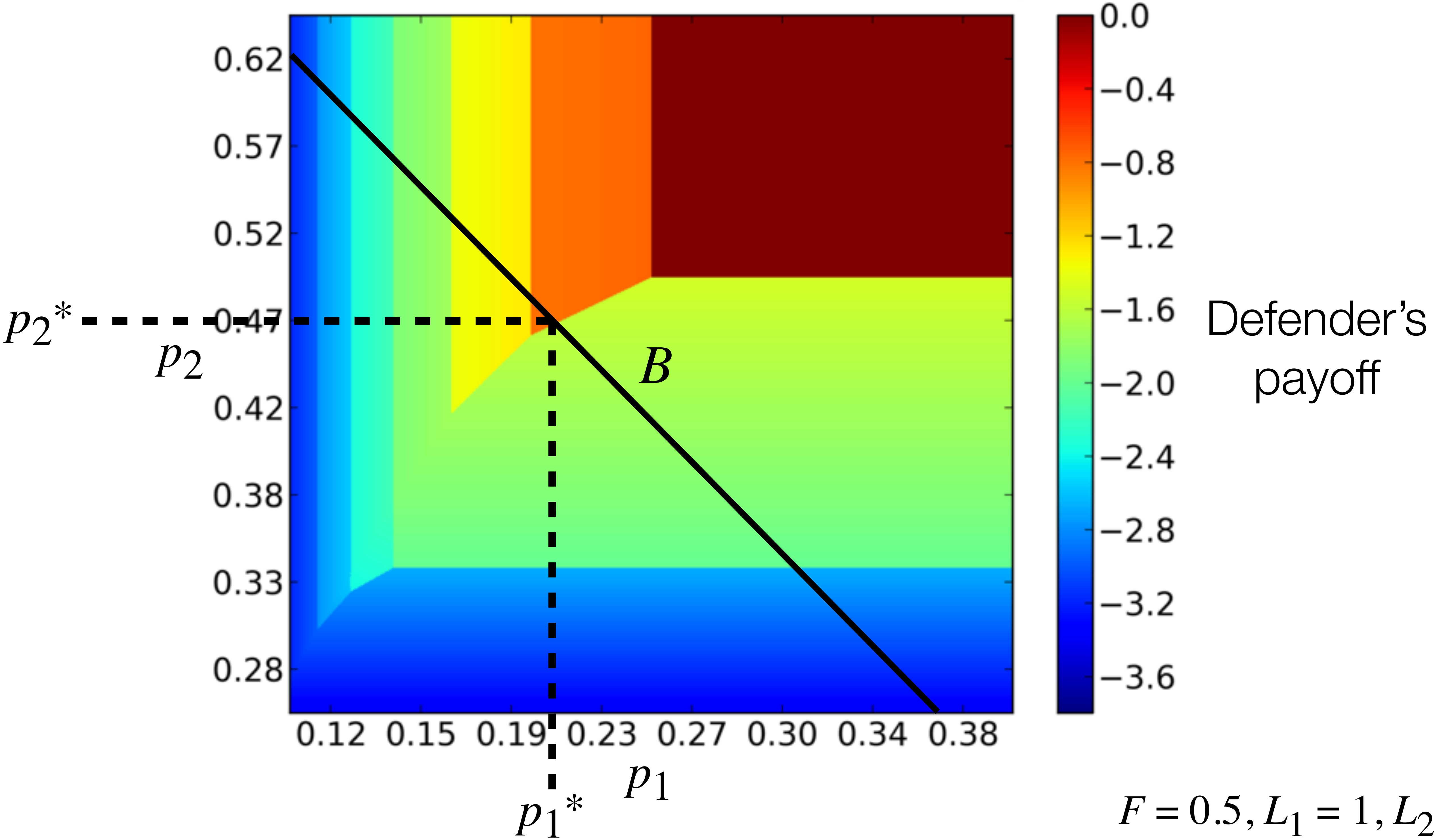
Theorem: The defender has a deterrence strategy if and only if

$$B \geq \sum_i \frac{L_i}{L_i + F} T_i$$

and the minimal deterrence strategy is

$$p_i = \frac{L_i}{L_i + F}$$

Non-Deterrence Strategies



$F = 0.5, L_1 = 1, L_2 = 3$

Continuous Relaxation

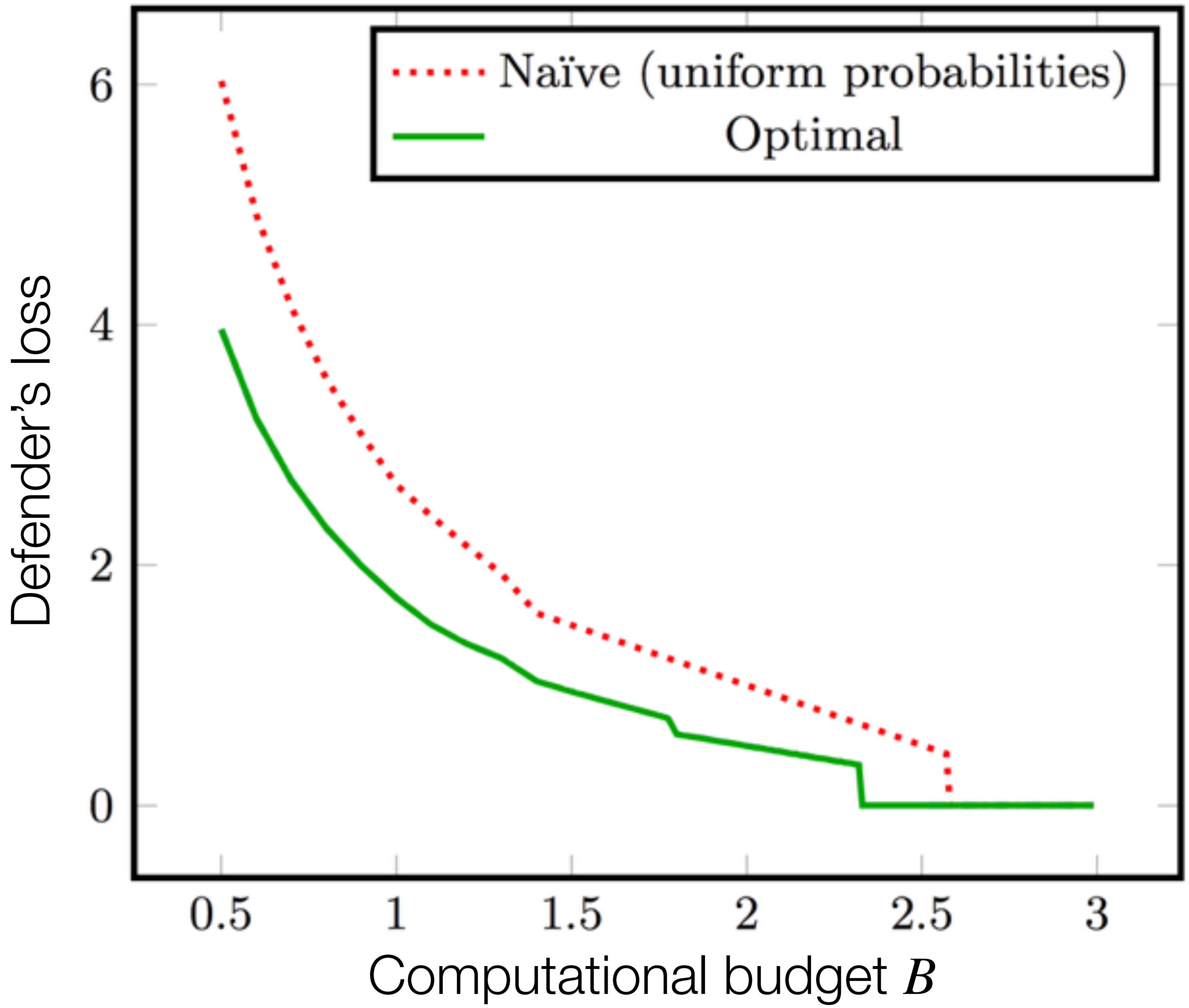
- No closed-form solution for the original model
- Continuous relaxation of the model
  - a_i is continuous (i.e., $a_i = 1.5$ means that the attacker modifies one and a half messages)

Theorem: Optimal strategy in the continuous relaxation is

$$\frac{L_1}{\ln(1-p_1)} = \frac{L_2}{\ln(1-p_2)} = \ldots = \frac{L_C}{\ln(1-p_C)}$$

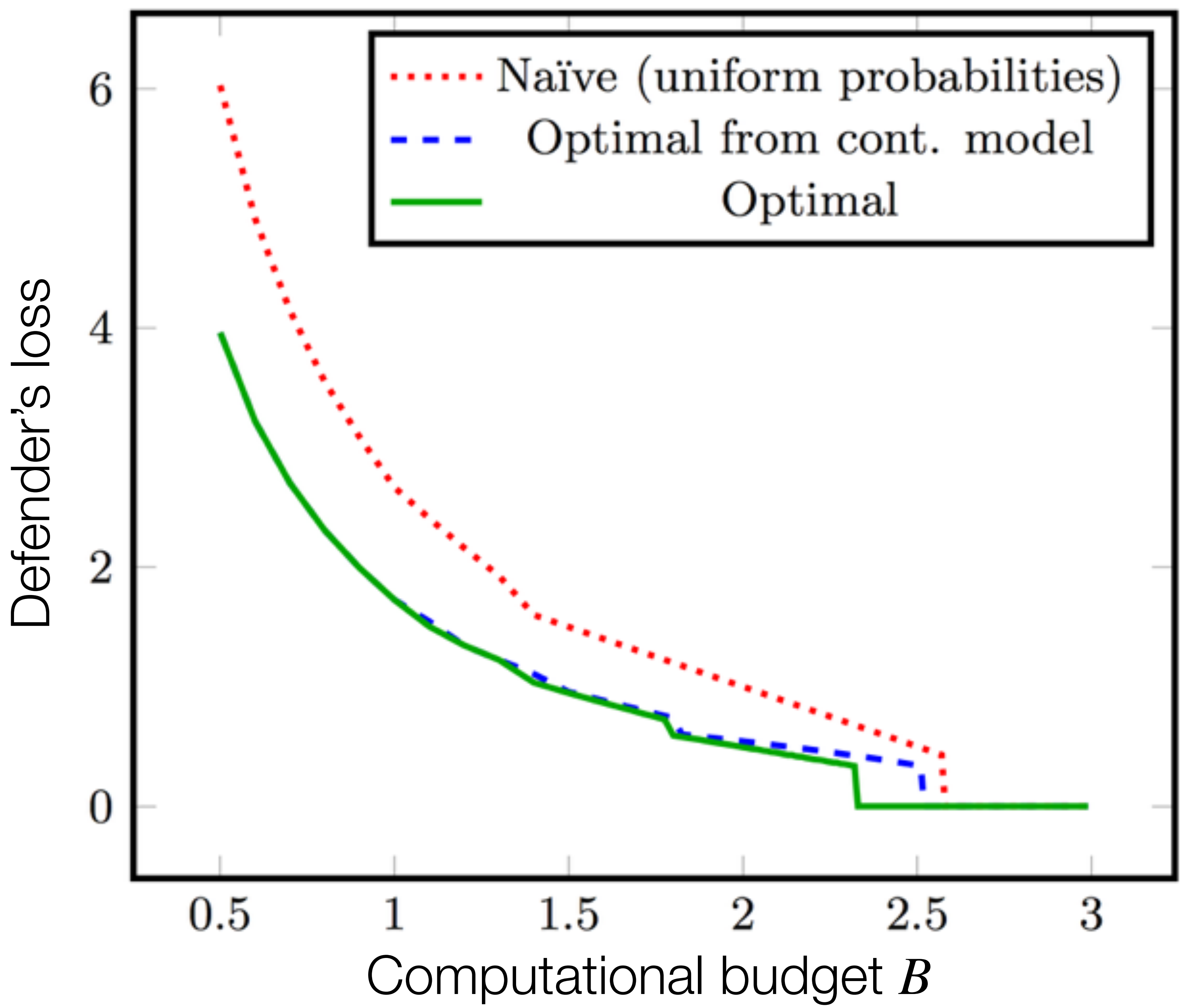
$$\sum p_i T_i = B$$

Numerical Example Comparing Strategies



$F = 0.5, L_1 = 1, L_2 = 2, L_3 = 3, T_1 = T_2 = T_3 = 1$

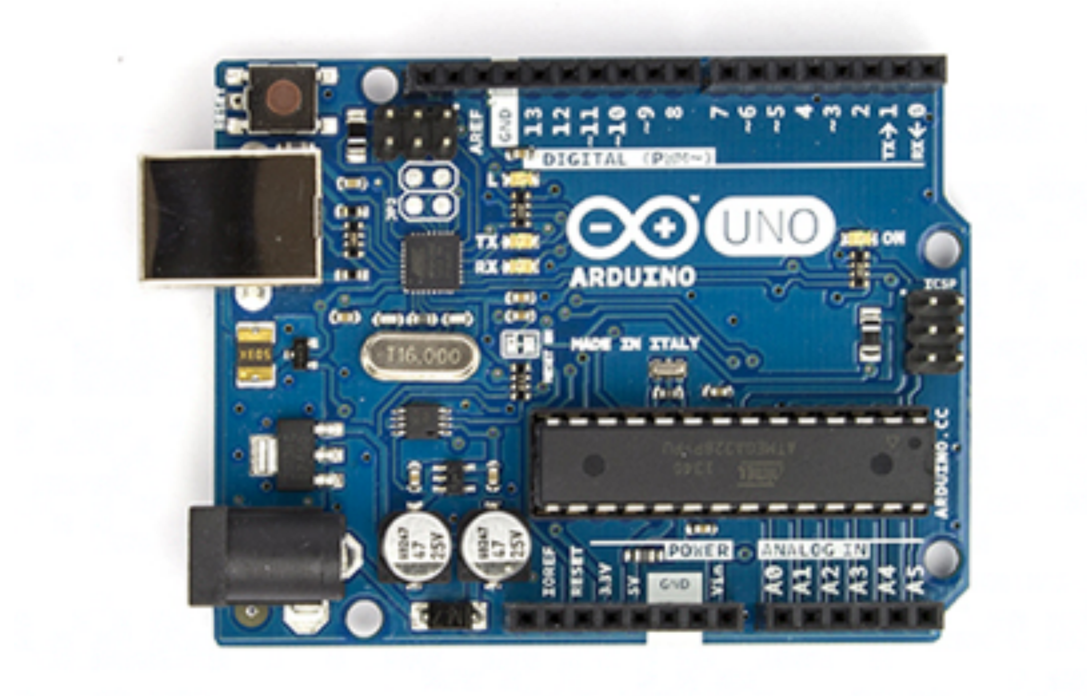
Numerical Example Comparing Strategies



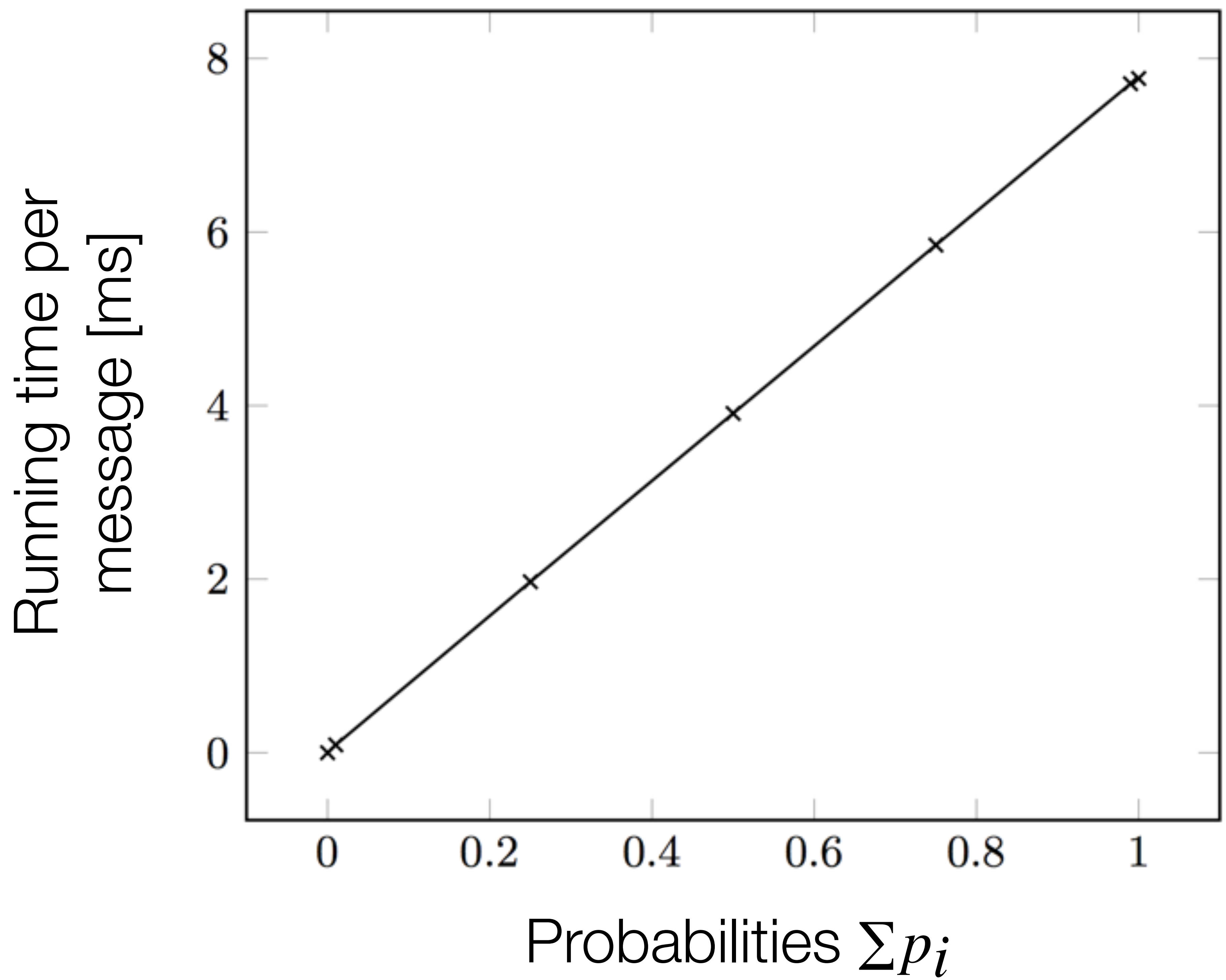
$F = 0.5, L_1 = 1, L_2 = 2, L_3 = 3, T_1 = T_2 = T_3 = 1$

Experiments

- Implementation and testing on an `ATmega328P` microcontroller
- Message authentication tag generation and verification:
  - HMAC (keyed-hash message authentication code)
  - using the SHA-1 hash function
- Random number generation:
  - linear-feedback shift register



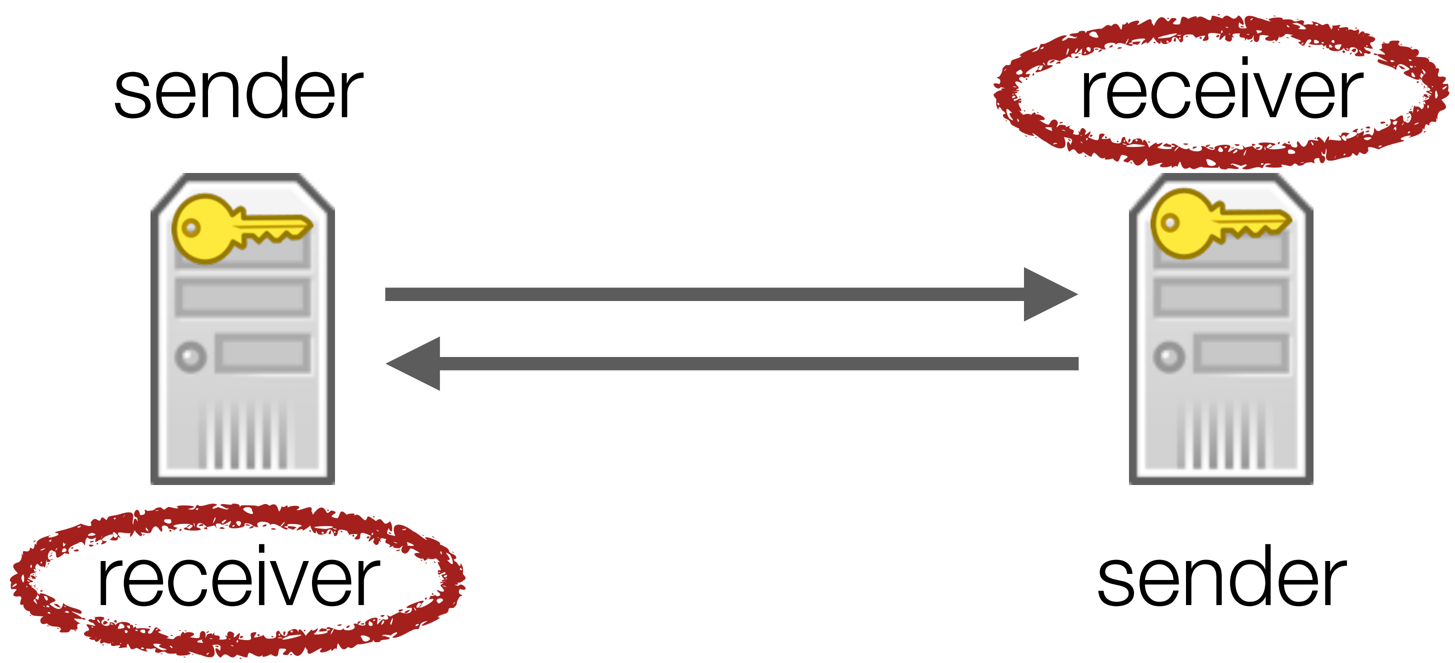
Experimental Results



Resource-Bounded Senders

- So far, we have saved computation **only at the receiver**

- Two-way communication



up to 100% saving when receiving
+ 0% saving when sending

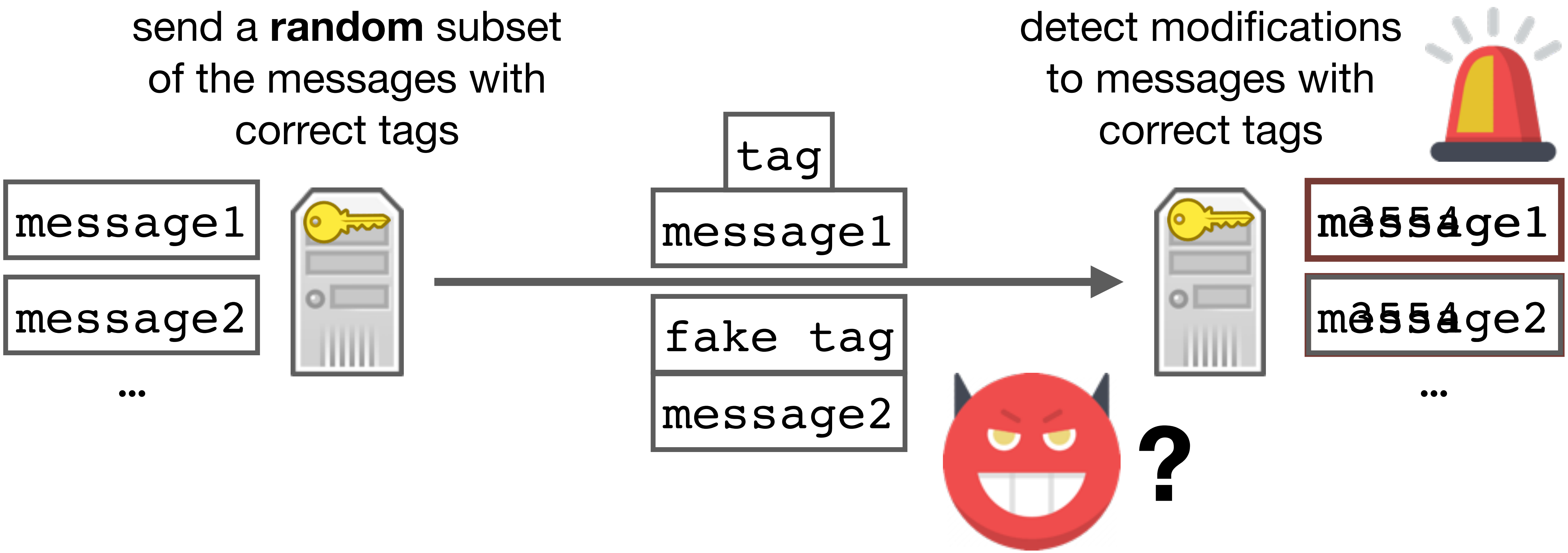
---

up to 50% saving overall

“Could we also save computation when generating tags?”

- **next: stochastic authentication tag generation**

Stochastic Message Authentication



- Fake tags
  - indistinguishable from correct tags for the attacker
  - distinguishable from incorrect tags for the receiver
  - computationally inexpensive to generate and verify

Generating and Verifying Fake Tags

- Proof-of-concept algorithms based on the HMAC construction with a Merkle-Damgard hash function

Algorithm 1 MAC tag generation in partial HMAC

function GENERATETAG(K, $\boldsymbol{m}$)
    $rnd \leftarrow \mathcal{U}(0,1)$
    if $rnd \leq p_{\text{class}(\boldsymbol{m})}$ **then**
        return HMAC($\boldsymbol{m}$)
    else
        return $f(f(IV,\ K \oplus ipad),\ m_1)$
    end if
end function

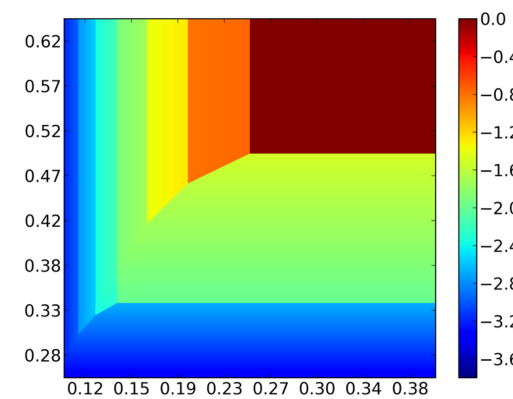
Algorithm 2 MAC tag verification in partial HMAC

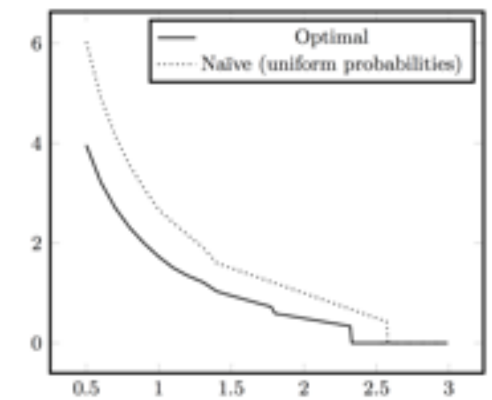
function VERIFYTAG(K, $\boldsymbol{m}$, t)
    $t_f \leftarrow f(f(IV,\ K \oplus ipad),\ m_1)$
    if $t = t_f$ **then**
        return *fake*
    else
        $t_c \leftarrow H\big((K \oplus opad) \mid \underbrace{f(f(\ldots f(t_f, m_2), \ldots, m_n), \text{length padding})}_{=H(K \oplus ipad \mid \boldsymbol{m})}\big)$
        if $t = t_c$ **then**
            return *correct*
        else
            return *incorrect*
        end if
    end if
end function

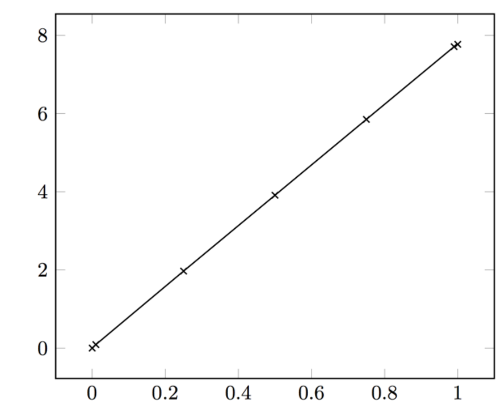
- Implementation and testing show substantial savings for both the receiver and sender on an `ATmega328P` microcontroller

Conclusion

- Stochastic message verification
  - message authentication for resource-bounded devices
  - game-theoretic model for defending against worst-case attackers
  - experimental results confirm computational cost model







- Next: stochastic message authentication tag generation
  - allows saving computation for both sender and receiver

Thank you for your attention!

Questions?