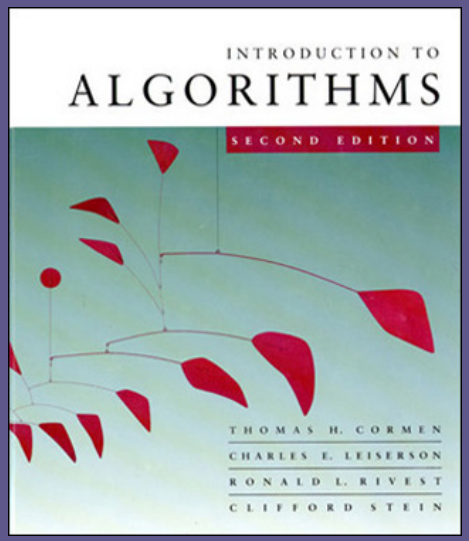
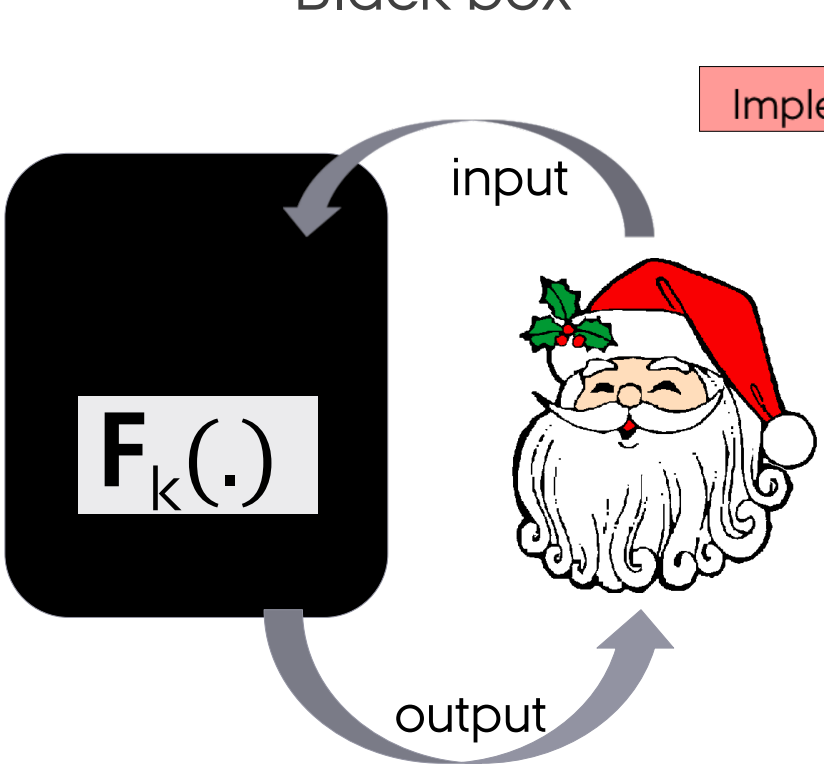
USING ALGORITHMS TO PROTECT AGAINST PHYSICAL ATTACKS

THANKS TO PRATYAY MUKHERJEE FOR THE SLIDES

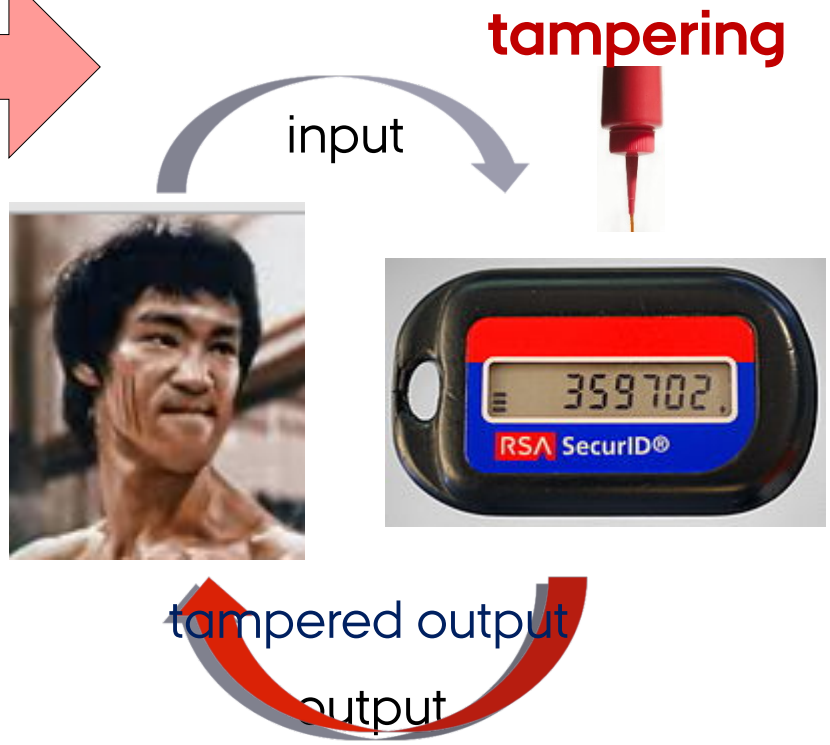


TAMPERING

Mathematical Model:
Black box

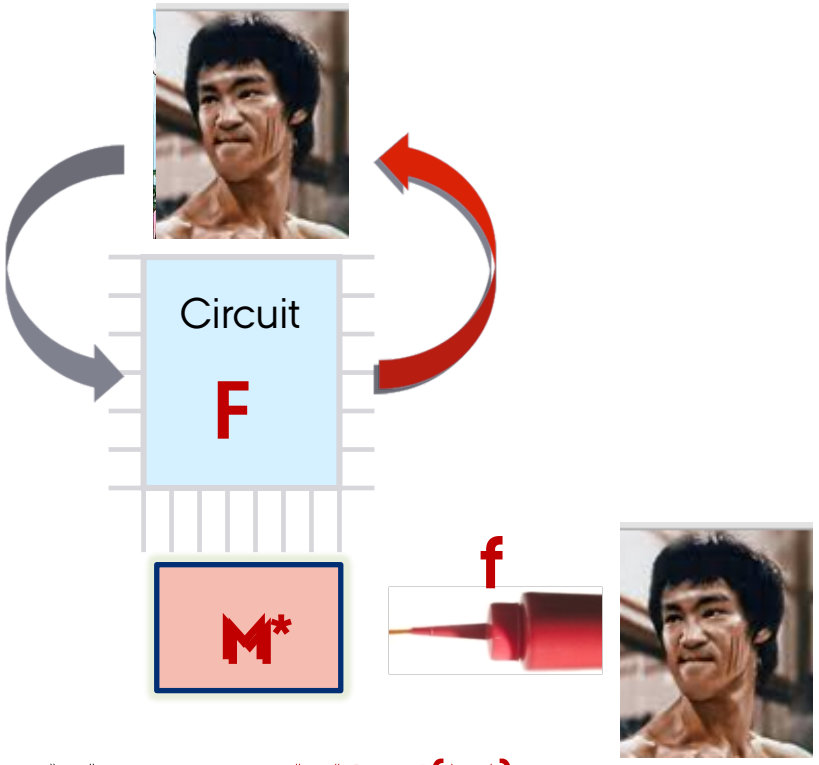


Implement



THE THEORETICAL MODEL

Tamper Resilient := Bruce cannot learn more using tampering than without!



Memory $M^* = f(M)$

A WAY TO PROTECT AGAINST MEMORY TAMPERING

Code based solution:

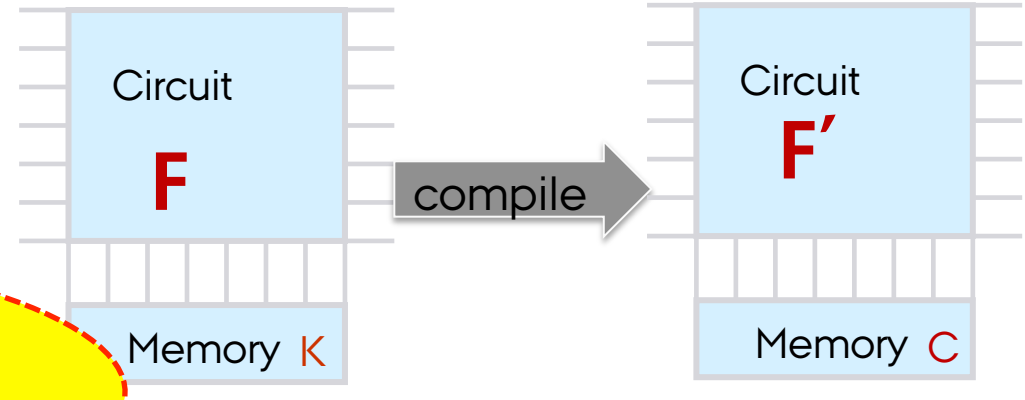
Initialization: C := **Enc**(K)

Execution of F'[C](x):
1. K = **Dec**(C)
2. Output F[K](x)

Build compiler for **any** functionality

-first proposed in

GLMMR04

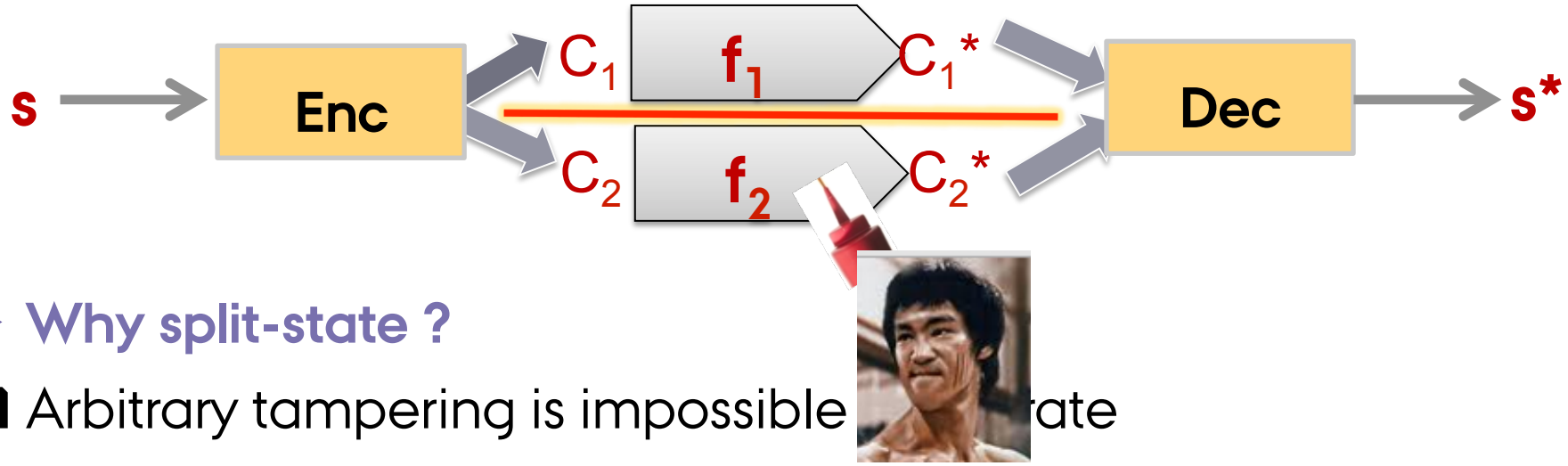


Problem:

Encode(Tamper(Decode(C)))

SPLIT-STATE TAMPERING

In this model, $C = (C_1, C_2)$ and $f = (f_1, f_2)$ for arbitrary f_1, f_2



- **Why split-state ?**
  - ❑ Arbitrary tampering is impossible
  - ❑ Hope: easy to **implement**

Goal: Non-Malleability[DPW10]: Guarantees s* = s or **unrelated** to s for "**interesting**" **class of functions F**

AARHUS
UNIVERSITY

Continuous Non-Malleable Codes and a Leakage and Tamper Resilient RAM

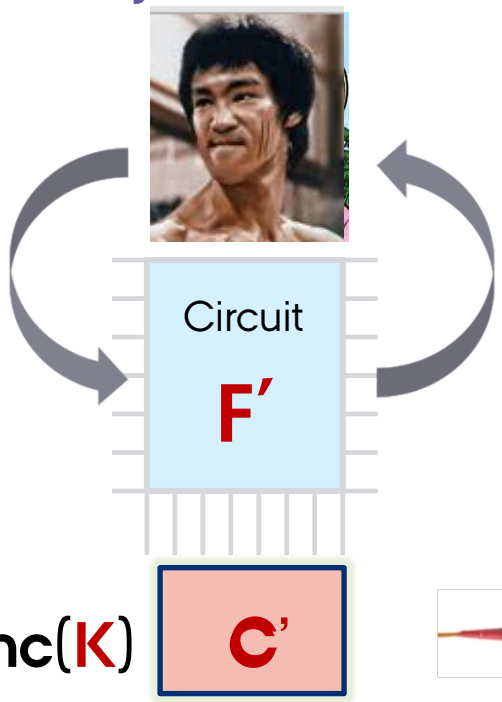
Pratyay Mukherjee, Sebastian Faust,

Jesper Buus Nielsen and Daniele Venturi

TCC 2014 + ...

ASSUMPTION OF PREVIOUS MODELS

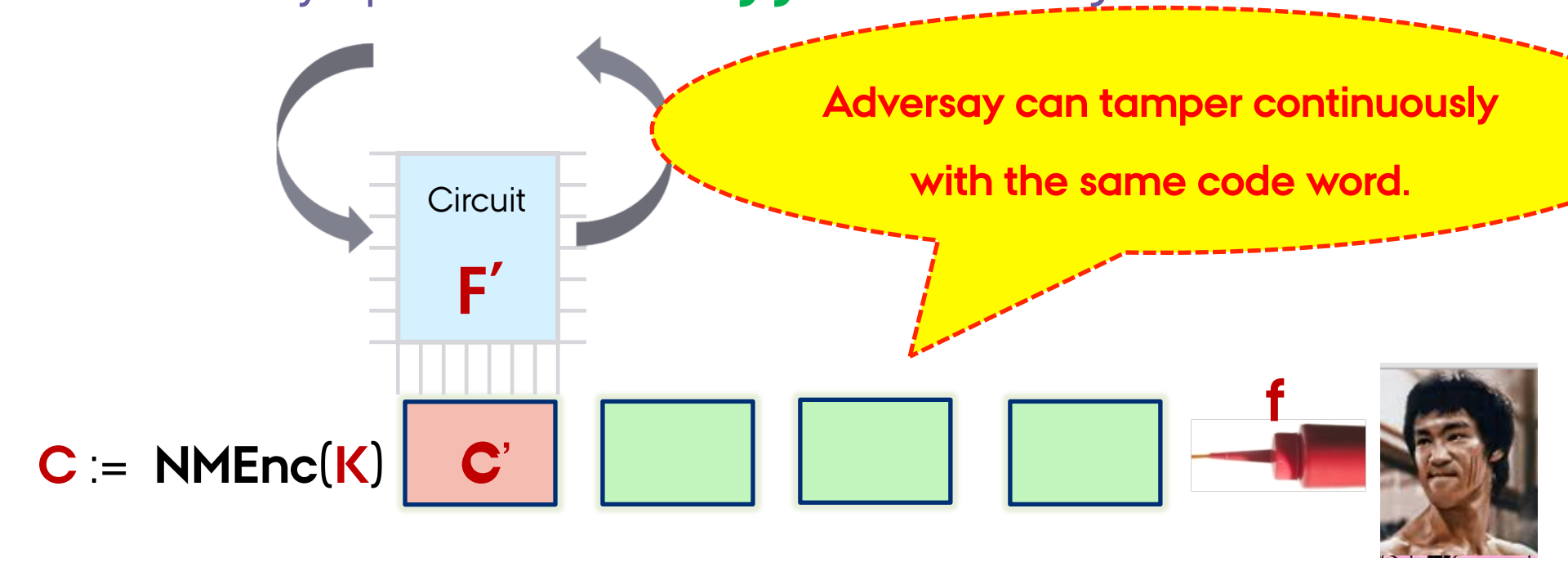
- Tampering irreversibly modifies memory.



Memory C*=f(C)

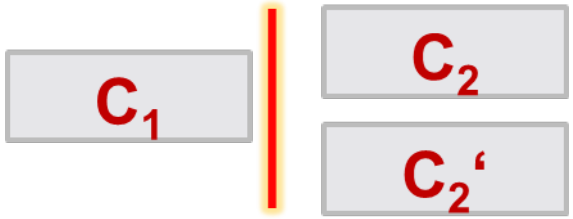
OUR STRONGER TAMPERING MODEL

- Memory space much **bigger** than length of codeword.



UNIQUENESS: A NECESSARY PROPERTY

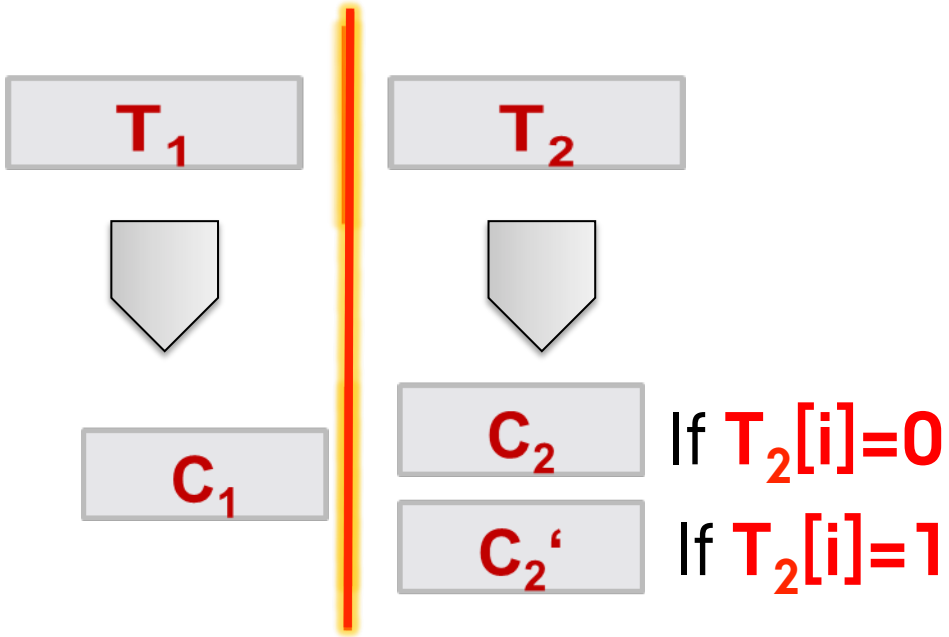
- **Def**: For any **Adv** it's **hard** to find (C_1, C_2, C_2') such that:



Both (C_1, C_2) and (C_1, C_2') are **valid**
(C_1, C_2) and (C_1, C_2') encode diff. msg

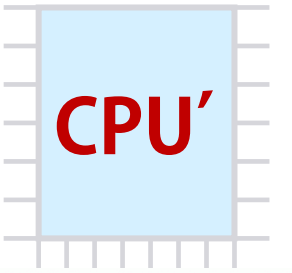
- **Why necessary ?**

Otherwise:



RESULTS

- We build the first continuous non-malleable code
  - We = { Pratyay Mukherjee, Sebastian Faust, Jesper Buus Nielsen, Daniele Venturi }
- We use it to describe the first leakage and tamper resilient random access machine



Memory M

$C_i := \mathbf{NMEnc}(s_i)$

C_1 C_2 C_3 C_4