USING ALGORITHMS TO PROTECT AGAINST PHYSICAL ATTACKS

THANKS TO PRATYAY MUKHERJEE FOR THE SLIDES
Mathematical Model: Black box

\[ F_k(.) \]

Implement

Tampering

Tampered output
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: $C := \text{Enc}(K)$

Execution of $F'[C](x)$:
1. $K = \text{Dec}(C)$
2. Output $F[K](x)$

Build compiler for any functionality - first proposed in GLMMR04

Problem:

$\text{Encode}(\text{Tamper}(\text{Decode}(C)))$
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 to tolerate
- Hope: easy to implement

Goal: Non-Malleability [DPW10]: Guarantees \( s^* = s \) or unrelated to \( s \) for “interesting” class of functions \( F \)
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.

\[ C := \text{NMEnc}(K) \]

\[ \text{Memory } C^* = f(C) \]
OUR STRONGER TAMPERING MODEL

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

\[ C \coloneqq \text{NMEnc}(K) \]

\[ C' \]

Memory \( M^* = f(M) \)

Adversay can tamper continuously with the same code word.
UNIQUENESS: A NECESSARY PROPERTY

Def: For any Adversary 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 different messages

Why necessary?

Otherwise:

If \(T_2[i] = 0\)
- If \(T_2[i] = 1\)
RESULTS

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

\[ C_i := \text{NMEnc}(s_i) \]