# Information Flow Tracking & Checking for OTBN Assembly

Advisor: **Roderick Bloem**

## Motivation

The OpenTitan Big Number coprocessor (OTBN) accelerates cryptographic operations on highly sensitive data. To keep these operations safe against side-channel attacks, the assembly code must obey strict secure coding guidelines: how registers are cleared, which instructions may follow each other, and where transitional leakage can occur. Such rules are derived from hardware leakage analyses with tools like Coco-Alma.

Today, checking that a piece of OTBN assembly actually respects these rules is done by hand. That is slow, tedious, and error-prone, and it does not scale to real implementations such as ECDSA or RSA. We want this check to run automatically, so that a software update cannot silently break the security guarantees of the hardware.

## Goals and Tasks

- Define how secret shares are annotated and tracked in OTBN assembly
- Extend the Python OTBN instruction set simulator with dynamic taint analysis
- Build a module that checks the resulting information flow against the Coco-Alma guidelines
- Evaluate the tool on OpenTitan implementations (e.g., ECDSA, RSA): violations found, false positives

## Literature


- B. Gigerl et al.
  Coco: Co-Design and Co-Verification of Masked Software Implementations on CPUs
  USENIX Security 2021
  https://www.usenix.org/conference/usenixsecurity21/presentation/gigerl
- lowRISC
  OpenTitan Big Number Accelerator (OTBN) Technical Specification
  https://opentitan.org/book/hw/ip/otbn/


## Courses & Deliverables

☑ **Master Project**
Project code
Report
Presentation

– OR –

☑ **Master's Thesis**
Initial presentation
Project code
Thesis (60+ pages)
Final presentation

## Recommended if you're studying

☑ CS ☑ ICE ☑ SEM

## Prerequisites

- Solid Python, some assembly
- Interest in side-channel countermeasures (no prior knowledge needed)

## Advisor Contact