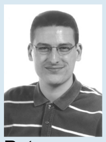
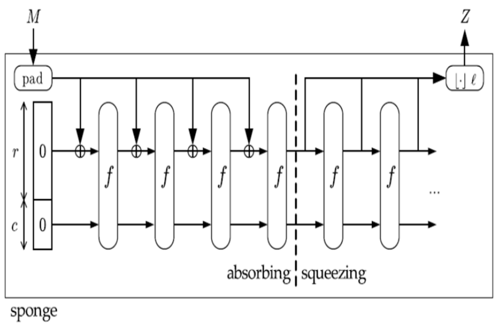

Reto Guadagnini

Graduate Candidate	Reto Guadagnini
Examiner	Prof. Dr. Andreas Steffen
Co-Examiner	Dr. Edgar Lederer, FHNW
Subject Area	Software and Systems

Towards a Formally Verified Implementation of SHA-3 in SPARK

A Practical Application of Formal Methods for Program Verification



The sponge construction of the SHA-3 hash algorithm (Source: http://keccak.noekeon.org/)

```
spark_vc procedure_pi_6
proof -
  from
    `∀j. 0 ≤ j ∧ j ≤ loop__2__y - 1 →
      a_new j ((2 * loop__1__x + 3 * j) mod 5) = a_init loop__1__x j`
  show ?C1 by simp

  from
    `∀i. 0 ≤ i ∧ i ≤ loop__1__x - 1 →
      (∀j. 0 ≤ j ∧ j ≤ 4 → a_new j ((2 * i + 3 * j) mod 5) = a_init i j)
    `loop__1__x ≤ 4` `loop__2__y ≤ 4`
  show ?C2
    apply auto
    apply (drule mod_add_cong [OF _ refl, of _ _ _ "- (3 * loop__2__y)"])
    apply simp
    apply (drule mod_mult_cong [OF refl, of _ _ _ 3])
    apply (simp add: mod_mult_left_eq [of 6 _ 5] mod_pos_pos_trivial)
    done
qed
```

Proof of a Verification Condition with the Isabelle proof assistant performed to verify a part of our SHA-3 implementation

Introduction: In autumn 2012 the National Institute of Standards and Technology (NIST) selected the KECCAK algorithm to become the new standard hash algorithm called SHA-3. When implementing a hash algorithm like SHA-3 it is critical for the security that the resulting implementation strictly follows the specification of the algorithm since otherwise security flaws could be introduced. Program verification by "formal methods" allow one to proof, that the implementation of a program really meets its specification in contrast to the today commonly used tests which are only able to show that the implementation of a program fulfills its specification in the test cases. There are programming languages like SPARK that were explicitly designed to support formal program verification. The main goal of this thesis was to create an implementation of the SHA-3 hash algorithm in the SPARK programming language, to formally verify this implementation regarding its specification as far as possible with the help of the SPARK tools and to use classical tests to verify the parts of this implementation for which the SPARK tools were not powerful enough to formally verify them.

Proceeding: In order to achieve this goal, first the basics of the formal method behind SPARK, which is called Hoare Logic and their application in the SPARK programming language were investigated. Then the KECCAK algorithm, which was elected to become the SHA-3 hash algorithm, was examined. Finally, since at the time of writing the standardization of the SHA-3 hash algorithm was not yet completed, the KECCAK algorithm with default parameters was implemented as part of the libSPARKCRYPTO (a library of cryptographic algorithms which are written in SPARK) in SPARK and was verified with the help of the SPARK tools.

Result: For the resulting implementation of KECCAK (SHA-3) in SPARK the absence of runtime errors was proved with the help of the SPARK tools. It appeared that for the formal verification of this implementation of KECCAK regarding its specification the SPARK tools alone were not sufficient and one had to use a proof assistant like Isabelle in addition. With the help of Isabelle, a small part of the resulting implementation of KECCAK could be formally verified regarding its specification, thus partial correctness was shown for this part. The rest of the implementation was verified regarding its specification by implementing and performing classical tests based on the test vectors provided for KECCAK. With the current version of SPARK and its tools it should be possible for an engineer with basic training in formal methods (especially Hoare Logic) to implement a cryptographic algorithm in SPARK and to prove the absence of runtime errors for this implementation with the help of the SPARK tools. For the formal verification of such an implementation regarding its specification, a strong background in mathematical logic, formal methods and extensive knowledge of a proof assistant like Isabelle is necessary and thus an expert in this field is required to perform formal verification.