Computational Soundness of Hash Functions
(Work in Progress)

Dominique Unruh
Saarland University

Goal of our work. We develop a computational soundness result for cryptographic hash functions. That is, we show that if a protocol that uses encryp-tions, signatures, and hash functions is secure with respect to a suitable symbolic security model, then it is also secure with respect to a cryptographic (computa-tional) security model. Our goal is to work with a standard cryptographic assumption, namely collision-resistance of hash functions. That is, instead of using non-standard cryptographic assumptions such as non-malleability or the random oracle hypothesis, we deal with the limitations of cryptographic hash functions by mimicking these limitations in the symbolic deduction rules. We be-lieve that this approach helps us to better understand the nature of symbolic and cryptographic hash functions. The work presented here is early work in progress.

Cryptographic limitations of hash functions. If we assume that a function $H$ is collision-resistant, but not anything else, an adversary can (at least for suitably constructed hash function) perform the following attacks:

- **Hash cycles.** Collision-resistance does not exclude the possibility of finding a bitstring $x$ with $x = H(x)$. Similarly, for other functions $f$, one might find bitstrings $x$ with $x = H(f(x))$. To mimic this symbolically, the adversary needs to be able to construct terms $t$ with $t = H(C[t])$ for contexts $C$. Such a term equation is only solvable if we endow the symbolic model with coinductive (i.e., recursive) terms.

- **Malleability.** It is possible that the adversary can, given, e.g., a hash $H(N, M)$ construct another hash $H(M, N)$ from it. To mimic such operations symbolically, we need the rule that if the adversary can deduce $t$ from $u$, then the adversary can also deduce $H(t)$ from $H(u)$.

- **Multi-hash leakage.** Given two hashes containing the same nonce $N$, e.g., $H(N, M)$ and $H(N, M')$, a computational adversary could extract $N$ because the combined leaked information of the two hashes is too large. Thus, symbolically we need a rule that, given hashes $H(t)$ and $H(u)$, the adversary can extract all nonces that appear both in $t$ and $u$. 