

Applications of SAT Solvers to Cryptanalysis of Hash Functions

Ilya Mironov and Lintao Zhang

Microsoft Research, Silicon Valley Campus
{mironov, lintaoz}@microsoft.com

Abstract. Several standard cryptographic hash functions were broken in 2005. Some essential building blocks of these attacks lend themselves well to automation by encoding them as CNF formulas, which are within reach of modern SAT solvers. In this paper we demonstrate effectiveness of this approach. In particular, we are able to generate full collisions for MD4 and MD5 given only the differential path and applying a (minimally modified) off-the-shelf SAT solver. To the best of our knowledge, this is the first example of a SAT-solver-aided cryptanalysis of a non-trivial cryptographic primitive. We expect SAT solvers to find new applications as a validation and testing tool of practicing cryptanalysts.

1 Introduction

Boolean Satisfiability (SAT) solvers have achieved remarkable progress in the last decade [MSS99, MMZ⁺01, ES03]. The record-breaking performance of the state-of-the-art SAT solvers opens new vistas for their applications beyond what conventionally has been thought feasible. Still, most of the successful real world applications of SAT solvers belong to the traditional domains of formal verification and AI. In this paper we explore applications of SAT solvers to cryptanalysis of hash functions.

Several applications of SAT solvers to cryptanalysis have been described in the literature [Mas99, MM00, FMM03, JJ05]. Their strategy can be regarded as a “head-on” approach, in the sense that they are not using any new or existing cryptanalytic methods in their attacks. Unsurprisingly, these efforts failed to produce any attacks of interest to cryptologists.

Despite the previous (arguably unsuccessful) attempts, we are convinced that SAT solvers could be of use in practical cryptanalysis. Our strategy may be described as “meet-in-the-middle”: after initial, highly creative work of cryptanalysts, we are able to delegate the more laborious parts of the attack to the SAT solver.

Recently, several important cryptographic hash functions were shown to be vulnerable to collision-finding attacks [WY05, WYY05b]. The original attacks consisted of several steps each of which involves a lot of bit-tweaking and manual work. It suffices to say that the attack on the simplest function of the family, MD4, requires keeping track of as many as 122 boolean conditions.

In this paper, we show that SAT solvers can be used to automate certain elements of these attacks. In particular, we demonstrate that SAT solvers may obviate the need for compiling tables of sufficient conditions and designing clever message-modifications techniques. Our successful attacks on MD4 and MD5 suggest that SAT solvers could be a valuable addition to cryptanalysts' toolkit.

The paper is structured as follows. Section 2 is a short primer on theory and practical constructions of hash functions. Section 3 covers recent attacks on hash functions; Section 4 presents experimental results of applying SAT solvers to automation of these attacks. We conclude in Section 5.

2 Theory and Constructions of Hash Functions

Cryptographic hash functions are essential for security of many protocols. Early applications of hash functions in systems security include password tables [JKW74] and signature schemes [RSA78, Lam79]; since then virtually any cryptographic protocol uses directly or indirectly a secure hash function as a building block.

The properties required of a secure hash function differ and often depend on the protocol in question. Still, the property of being *collision-resistant* is recognized as the “gold standard” of security of hash function. The first formal definition of collision-resistant hash functions (CRHF) was given by Damgård [Dam88]. A function H is said to be collision-resistant if it is infeasible to find two different inputs x, y such that $H(x) = H(y)$. Since any compressing function has collisions, a guarantee of collision-resistance may only be computational.

A first standard hash function, MD4, was designed by Ron Rivest [Riv91]; its strengthened version MD5 followed shortly thereafter [Riv92]. A first NIST-approved hash function, SHA (Secure Hash Algorithm), adopted the general structure (and even some constants!) of MD4 [NIS93]; it was withdrawn in 1995 and replaced with a new version, dubbed SHA-1 [NIS95], that differed in one additional instruction. To avoid confusion, the original SHA is commonly referred to as SHA-0. As of 2004, two hash functions were in wide-spread (and almost exclusive) use: MD5 and SHA-1. It is fair to say that all of these functions belong to one family that shares similar design principles.

Compression function. The basic construction block of CRHFs is a collision-resistant *compression function*, which maps a fixed-length input into a shorter fixed-length output.

The heart of the construction is a *block cipher*, which is defined as a function of two inputs $E: \{0, 1\}^k \times \{0, 1\}^n \mapsto \{0, 1\}^n$. Although $E(\cdot, \cdot)$ compresses its input by mapping $k + n$ bits into n bits, as is it is trivially invertible. However, the following trick, called the Davies-Meyer construction, results in a CRHF F under the assumption that E is an ideal block cipher (i.e., $E(x, \cdot)$ is an indexed collection of random permutations on $\{0, 1\}^n$):

$$F(M, x) = E(x, M) \oplus M.$$