Watermarking of SAT using Combinatorial Isolation Lemmas

Rupak Majumdar
Electrical Engineering & Computer Science Department
University of California
Berkeley, California

Jennifer L. Wong
Computer Science Department
University of California
Los Angeles, California

ABSTRACT

Watermarking of hardware and software designs is an effective mechanism for intellectual property protection (IPP). Two important criteria for watermarking schemes are credibility and fairness. In this paper, we present the unique solution-based watermarking technique which provides, in a sense, the ultimate answer to both credibility and fairness requirements. Leveraging on a combinatorial theorem of Valiant and Vazirani, we demonstrate how ultimate credibility and complete fairness can almost always be achieved with high probability during the watermarking of the solution of the satisfiability (SAT) problem. The effectiveness of the technique is demonstrated on both specially created examples where the number of solutions is known, as well as on common CAD and operation research SAT instances.

1. INTRODUCTION

The reuse of intellectual property (IP) such as IC cores and software libraries is widely considered to be the most economically efficient way to close an increasing gap between the ability of designers to develop integrated circuits and the potential of silicon. One of the prerequisites of hardware and software IP is the development of IPP techniques. Watermarking is the embedding of information into a design for the purpose of identification or proof of authorship. It is one of the most effective IPP techniques due to its flexibility, strong proof of authorship, and very low overhead in terms of speed, area, and power. In the last several years a number of watermarking based IPP techniques have been developed at all levels of the design process, including system synthesis, behavioral synthesis, logic synthesis, and physical design [5, 6, 8, 13]. The key observation on which all watermarking-based IPP techniques are based on is the fact that many synthesis problems are associated with computationally intractable or difficult optimization problems for which there exists a high number of different solutions with identical or very similar quality. Watermarking-based techniques leverage on this fact by incorporating the design signature in the design specification as additional constraints and therefore to ensure that the completed design satisfies both the initial specification as well as the new constraints. Therefore, the design is unique and only the author of the design knows the encrypted signature. While many of these techniques perform well in practice there is very little known about their theoretical underpinnings. In particular, two issues deserve more sound and more effective treatment: calculation of *credibility of the ownership* and *fairness.*

Credibility of ownership can be defined as the number of solutions to a given instance of the problem after imposing signature related additional constraints versus the number of solutions of the same or better quality before the addition of the signature. Obviously, a perfect watermarking scheme would minimize this ratio. For decision optimization problems this means that a perfect watermarking scheme would add constraints in such a way that the problem has exactly one solution.

Fairness is the assurance that for all possible signatures of a given length, the number of solutions to the synthesis problem is identical or at least similar. Recently, Qu et al. [15] introduced the first watermarking technique which embeds instance specific constraints in order to ensure fairness.

In this paper we present a new watermarking method based on constraint addition and provide a complete theoretical analysis of its properties, including credibility and fairness. We formally and experimentally demonstrate that the method provides in some sense a complete solution to both credibility and fairness problems by reducing the number of solutions of an instance of an optimization problem to exactly one. Moreover, the optimality does not depend on any particular probability distribution over the input space. By a series of experiments on actual problem instances we also show that the method works quite well on practical benchmarks. In our research, we focused our attention on the propositional satisfiability (SAT) problem. We were mainly motivated by the fact that SAT is widely used to model many optimization and verification tasks in CAD as well as in other application areas such as artificial intelligence and optimization research. It is important to emphasize that the new method is completely general and can be easily adapted to watermarking any other NP-complete problem.

Our method is based on a combinatorial result of Valiant and Vazirani [16] that randomly reduces a given instance of SAT to an instance which has (with high probability)

DAC2001, June 18-22, 2001, Las Vegas, Nevada, USA.

Sig. Len (bits)	4	8	12	16	20	24	28	32
Ave. # of Sols	4.1	2.1	1.3	0.5	0.2	0.3	0.1	0.0
Min/Max Sols	2/5	1/4	0/3	0/1	0/1	0/1	0/1	0/0
Ave. Discrep.	.5	.6	.72	.5	.32	.42	.18	0

Table 1: The relationship between the number of solutions and the length of the signature for the motivational example.

exactly one satisfying assignment. The construction successively conjoins constraints to the original formula to produce a series of formulas that have a monotonically decreasing number of solutions. By utilizing a binary search one can efficiently identify the maximal length of the signature related constraints which still do not make the formula unsatisfiable. Another important property is that if additional constraints are added at random, there is very high probability that all signatures will terminate with unique solutions at very similar lengths.

We illustrate the two main properties, maximal credibility and fairness, using the following example. Consider the SAT formula

$$f = (x_1 \vee x_3 \vee \overline{x}_4)(x_2 \vee x_4)(\overline{x}_1 \vee x_2 \vee \overline{x}_3 \vee x_4)(\overline{x}_2 \vee x_3 \vee \overline{x}_4)$$

over four variables x_1, x_2, x_3, x_4. Originally, we have nine satisfying assignments or solutions. By embedding signatures (method to be described later) which are multiples of length four to the instance we obtain the results shown in Table 1.

The first row of Table 1 indicates the length of the embedded message and the second row indicates the average number of solutions to the SAT instance for randomly selected strings of indicated length after ten trials. The final two rows present the minimum and maximum number of solutions found as well as the average discrepancy between the number of solutions. We can see from the results That as the length of the signature increases, the number of solutions is approximately halved.

In order to illustrate intrinsic fairness of the proposed technique even on very small examples, such as our motivational example, we conducted the following study. We embedded as signature all strings of length four which have at least a single one bit into our motivational example. There are 15 such strings. Note that out technique requires that at least one bit in the string is not zero, which obviously happens with high probability even for strings of moderate length.

The number of solutions for these 15 strings varies between two and six. For one of them there are only 2 solutions. For two strings there are 3 solutions. Five signature strings results in 4 solutions , and five in 5 solutions. Finally, for one of the strings there are 6 solutions. The average distance of the number of solutions from the average case is 0.85 and the average variance is 0.21. This clearly indicates that even on very small examples and very short messages the technique performs well. A much more comprehensive evaluation of the technique is given in Section 6.

The remainder of the paper is organized as follows. In Section 2 we survey related work. In order to make the paper self-contained, we provide in Section 3 technical background on constraint based watermarking schemes. Section 4 is the technical core of the paper which describes the procedure to create a unique solution to the SAT problem. In Section 5, we briefly describe our software experimentation environment. Finally, we present the experimental results and conclude.

2. RELATED WORK

2.1 Watermarking

There are two different domains for watermarking: static artifacts and functional artifacts. A variety of techniques have been developed to watermark static multimedia artifacts [4, 17]. Watermarking techniques have also been developed for functional artifacts. Functional artifacts can be described at several levels of abstraction such as system level designs, high level logic synthesis, and physical designs for FPGA [18]. The majority of watermarking techniques are vertical, but there are several horizontal techniques [5, 8]. In *constraint based watermarking* [14, 15], the designer embeds additional constraints which denote the users signature in a unique way. Fair watermarking [15] is a constraint-based scheme that provides, in addition, high credibility to each user.

There are two main advantages of the new technique over previously published techniques. First, the new technique provides ultimate proof of credibility by enabling the designer to impose a number of constraints in such a way that there exist only unique solutions. These solutions correspond to the signature. Second, the technique provides strong probabilistic proof that the fairness property is enforced during the watermarking process. While previous techniques approached the problems of credibility and fairness heuristically at best, the new method provides a theoretical justification and proof of optimality.

2.2 Satisfiability Problem

SAT was proven to be the first NP-complete problem [3]. SAT has numerous applications both in VLSI CAD and other domains such as artificial intelligence, operations research, and combinatorial optimization (see, *e.g.,* [7]). Several efficient techniques have been developed to solve the SAT problem and many efficient public domain packages are available [1, 2, 11, 10]. The economic importance of efficient SAT solving is well illustrated by a number of FPGA based application specific computers exclusively built for solving SAT problems [9].

Valiant and Vazirani [16] proved that the number of solutions of a NP-complete problem, which can vary from zero to exponentially many, does not impact its inherent intractability: in fact, if there is a polynomial time algorithm for finding solution to SAT instances having a unique solution, then NP=RP. The main technical construction of their result is an isolation lemma that reduces an instance of SAT to an instance with a unique solution in randomized polynomial time. Valiant and Vazirani's theorem have been applied to prove several important theorems of computational complexity theory (see, *e.g.* [12]).

3. PRELIMINARIES

In this section, we briefly survey constraint-based watermarking methodology. The watermarking problem takes two inputs: the initial instance of the optimization problem (which corresponds to optimization synthesis or compilation problem) and the owner's signature in some standard format. The optimization problem maximizes an objective function with respect to some constraints. The signature is translated to a set of additional constraints which should satisfy tests for randomness. The signature is applied as input to cryptographic hash function whose output is used as a seed to a binary random number generator. The binary random generator is used to generate an infinite string of zeros and ones.

The crucial observation for creating additional constraints is that for components of the instance (in the case of the SAT problem components are variables and clauses) one has to establish a well defined ordering. This can be done in two ways, either by using industry imposed standards or by using properties of the designs. For example, for the SAT problem, we can use orderings according to both clauses and variables. In the case of variables, for example, we can use rank order rules such as the number of appearances of variables in all clauses, the number of complemented forms of each variable, and the number of occurrences of variables and uncomplemented variables in clauses of odd length. With this well defined ordering one can then add additional constraints in the most efficient and systematic way. By following the ordering, one can reverse engineer the original signature, which is one of key objectives when showing proof of authorship.

The next step is the superposition of signature constraints on to the instance of the problem. After that the problem is solved using off-the-shelf problem solvers. The output is a watermarked design which can be analyzed according to standard watermarking desiderata. The essence of constraint-based watermarking is to restrict the users solution to the part of solution space which is characterized by the signature constraints. The key essential assumption is that there are numerous solutions of high and very similar quality. In the case of decision problems, such as SAT, the addition of extra constraints should not change the positive answer to the initial instance of the problem to a negative answer after the addition of the watermarking constraints. Ideally, one should add a signature long enough that only a single solution is found in the part of the solution space denoted by signature constraints. In addition, different signatures of the same length should have fairly assigned a number of positive answers, that is, there should be the same number of solutions which is a function only on the length of the signature. We show next how our method achieves all these properties.

4. CREATING UNIQUE AND FAIR SOLUTIONS TO SAT

Valaint and Vazirani's construction essentially isolates a solution of a CNF formula by a randomized reduction. Given an instance f of SAT, the method successively conjoins constraints to f to obtain a series of formulas $f_1, f_2, \ldots, f_n$ that will have decreasing number of solutions. If f is satisfiable, we can prove that with probability at least $\frac{1}{4}$ one of the formulas will have a unique solution. If we choose one of the formulas at random, then the probability that it has a unique solution is at least $\frac{1}{4n}$. This probability can be boosted as usual. On the other hand, if f is not satisfiable, then each of the formulas will be unsatisfiable. The watermarking method constructs, given an instance of SAT, a formula with a unique satisfying assignment, and produce the unique assignment as the solution. The construction will ensure that this assignment satisfies the original formula f. However, the probability that a random algorithm selects exactly this satisfying assignment is low. We now outline the construction. The treatment is from [16]. For the sake of brevity, we omit the proofs of correctness.

We shall select constraints at random from some suitable set. Ideally, we would like to eliminate each solution independently with a certain probability. This is not possible with only a polynomial number of random choices. However, the use of $GF[2]$ inner products with polynomially few vectors over $GF[2]^n$ suffices for our purposes. Let f be a CNF formula over the variables $x_1, x_2, \ldots, x_n$. We shall view truth assignments to the variables $x_1, x_2, \ldots, x_n$ as n-dim-ensional $\{0, 1\}$ vectors over the vector space $GF[2]^n$. The satisfying assignments of f form a set of vectors from this space. For $u, v \in GF[2]^n$, let $u \cdot v$ denote the inner product over $GF[2]$ of u and v.

LEMMA 1. **[16]** *If f is any CNF formula in $x_1, x_2, ..$ $, x_n$ and $w_1, \ldots, w_k \in \{0,1\}^n$, then one can construct in linear time a formula f'_k whose satisfying assignments v satisfy f and the equations $v \cdot w_1 = v \cdot w_2 = \cdots = c \cdot w_k = 0$. Furthermore, one can construct a polynomial-size CNF formula f_k in variables $x_1, \ldots, x_n, y_1, \ldots, y_m$ for some m such that there is a bijection between solutions of f_k and f'_k defined by equality on the values of $x_1, \ldots, x_n$.*

We show the addition of one constraint. The general case follows easily. The formula f'_1 is

$$f \wedge (x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_j} \oplus 1),$$

where $\oplus$ denotes the exclusive-or function, and $i_1, .., i_j$ are the indices of the x_i that have value 1 in w_1. The function f_1 is simply the CNF equivalent of f'_1:

$$f \wedge (y_1 \Leftrightarrow x_{i_1} \oplus x_{i_2}) \wedge (y_2 \Leftrightarrow y_1 \oplus x_{i_3}) \wedge \cdots \wedge$$
$$(y_{j-1} \Leftrightarrow y_{j-2} \oplus x_{i_j}) \wedge (y_j \Leftrightarrow y_{j-1} \oplus 1) \wedge y_j.$$

The intuition behind the construction is the following surprising fact. Let S be a subset of $\{0,1\}^n$. Define the sets $S_1 = \{v \mid v \in S, v \cdot w = 0\}$, and $S'_1 = \{v \mid v \in S, v \cdot w = 1\}$. Then, if w is chosen randomly, any S will be partitioned in this way into two roughly equal halves with high probability. In our construction, S is the set of satisfying assignments of f, we choose $w_1, \ldots, w_k$ at random, and constructing f_k we obtain a formula with roughly $2^{-k}|S|$ satisfying assignments. Note that we do not know $|S|$ other than it lies between 0 and 2^n. Therefore, we need to "guess" the size of $|S|$. This is where the random choice of k comes in: with probability $\frac{1}{n}$, we make the right guess.

The overall construction is the following: Given a CNF formula f, choose an integer k at random from $\{1, \ldots, n\}$, randomly choose vectors $w_1, \ldots, w_k$ and output f_k. We now present the technical result that formalizes the above intuition.

THEOREM 1. **[16]** *Let $S \subseteq \{0,1\}^n$. Suppose $w_1, .., w_k$ are chosen at random. For each $i \leq n$, let $S_i = \{v \mid v \in S, v \cdot w_1 = \cdots = v \cdot w_i = 0\}$, and let $P_n(S)$ be the probability that, for some $i \leq n$, $|S_i| = 1$. Then:*

i $P_n(S) \geq \frac{1}{4}$;

ii if $w_1, \ldots, w_n$ are chosen to be linearly independent in addition, then $P_n(S) \geq \frac{1}{2}$.

```
/* Precondition: f is of the form x_1 ⊕ x_2 ⊕ ... x_k ⊕ 1 */
convertToCNF(formula f){
    let {y_1, y_2, ... , y_k} be new variables;
    /* let a ≡ b ⊕ c denote the CNF formula
    (¬a ∨ ¬b ∨ ¬c)(¬a ∨ b ∨ c)(a ∨ b ∨ ¬c)(a ∨ ¬b ∨ c) */
    return
    ((y_1 ≡ x_1 ⊕ x_2) ∧ (y_2 ≡ y_1 ⊕ x_3) ∧ ...
        ∧ (y_{k-1} ≡ y_{k-2} ⊕ x_k) ∧ (y_k ≡ y_{k-1} ⊕ 1) ∧ y_k);
}

formula addOneConstraint(formula f){
    pick w uniformly at random from {0,1}^n;
    let {i_1, ... , i_k} be the positions of the 1 entries in w;
    return f ∧ convertToCNF(x_{i_1} ⊕ x_{i_2} ⊕ ··· ⊕ x_{i_k} ⊕ 1);
}

/* Precondition:
f is a CNF formula over the variables {x_1, ... , x_n} */
formula genConstrainedFormula(formula f){
    pick k uniformly at random from {1, ... , n};
    for (i = 1 to k) do f = addOneConstraint(f); od
    return f;
}
```

Figure 1: Pseudo code for generating watermarked formula.

Figure 1 shows the algorithm to produce the final formula (conjoined with the additional constraints). The function *addOneConstraint*() adds one more constraint to the current formula, thus making the number of solutions drop to roughly half the original number (with high probability). The function *genConstrainedFormula*() is a loop that calls *addOneConstraint* k times. Note that every call effectively reduces the number of satisfying assignments by half. The function *convertToCNF* takes a formula of the form $x_1 \oplus \cdots \oplus x_k \oplus 1$ and converts it to a CNF formula (with some new variables). The beauty of the technique is that the final algorithm is extremely simple (it involves only several random choices), yet it yields optimal results with high probability.

5. EXPERIMENTAL ENVIRONMENT

In this section, we present the software environment which is used for experimental evaluation of the new watermarking technique. Specifically, we developed three programs: (i) program which generates an instance of the SAT problem with user specified number of solution for requested number of variables (ii) program for branch and bound based exhaustive enumeration of solutions in a SAT instance (iii) program for the watermarking of SAT instance using the combinatorial isolation lemmas. In addition, we used also several public domain SAT solvers.

At the intuitive level the program for creating an instance of SAT for known number of solution relies on four phases. First, we use a random number generator and linear hash function based algorithm for avoidance of collision to generate n different assignments of variables which will constitute solutions to the instance of the created SAT problem. In the second phase, we order the variables according to the amount of discrepancy between pair of variables. We do this to cut the solution space as quickly as possible. Next, we add clauses which eliminate all non-valid solutions. Finally, in the last phase we alter some of the clauses and add additional clauses to hide the structure of the instance. Figure 2 shows pseudo code for the creation of the SAT instance.

```
Input:     k number of variables.
           m number of solutions.
Output: SAT instance and solutions.
Algorithm:
createSATInstance(){
1. generateSolutions(k, m);
2. orderVariables();
3. eliminateNonSolutions();
4. Addition of Confusion();}
5. generateSolutions(k, m){
6.     Random number generation of m
     solutions using linear hash function}
7. orderVariables(){
8.     Order pairs of variables according to
     their consistency. }
9.eliminateSolutions(){
10.     Create clauses which prevent non-solutions
     from being solutions.}
```

Figure 2: Algorithm for the creation of a SAT instance with a specific number of solutions.

To clarify the process, consider the following example. We wish to create a SAT instance with four variables x_1, x_2, x_3, x_4 and five solutions. Using the random number generator and linear hash function we generate the following assignments of variables, which are the solutions to the instance.

$$(\overline{x}_1, \overline{x}_2, x_3, x_4)(\overline{x}_1, \overline{x}_2, \overline{x}_3, x_4)(x_1, \overline{x}_2, \overline{x}_3, \overline{x}_4)$$
$$(x_1, \overline{x}_2, x_3, \overline{x}_4)(x_1, x_2, x_3, \overline{x}_4)$$

Now, we order the variables according to discrepancy. As a result of pair x_1 and x_4 appearing together in only two forms, we order these two variables first. We then compare the rest of the variables to the previous pair. The resulting ordering is x_1, x_4, x_2, x_3. We begin adding clauses by building a branch and bound binary search tree. We begin by adding variable 1. By examining our solutions, we see that x_1 appears in both forms x_1 and $\overline{x}_1$, therefore we cannot terminate any branches. We continue for x_4. We see that no solutions have the form $\overline{x}_1, \overline{x}_4$ or x_1, x_4. We terminate these branches and create clauses which will eliminate all solutions of these forms. In this case, we add clauses $(x_1 \vee x_4)$ and $(\overline{x}_1 \vee \overline{x}_4)$. We continue this process until all branches which lead to non-valid solutions have been terminated by the creation of clauses. The created instance is

$$f = (\overline{x}_1 \vee \overline{x}_4)(x_1 \vee x_4)(x_1 \vee \overline{x}_2 \vee x_4)(\overline{x}_1 \vee \overline{x}_2 \vee x_3 \vee x_4)$$

The last step is to add additional clauses and to alter clauses to hide the structure of the instance and increase the complexity of the instance.

The second program, again uses the branch and bound technique to enumerate all solutions of a given instance. We implemented the algorithm show in Figure 1 for watermarking SAT instances using the combinatorial isolation lemmas. The watermarked instances were tested on the public domain SAT solvers WalkSAT [11], GSAT [10], NTAB [2], and Rel_SAT [1].

6. EXPERIMENTAL RESULTS

Instance	# Vars	# Clauses	Orig. (sec)	1x	2x	5x	10x	25x	50x	100x
par8-1-c.cnf	64	254	0.1	38	83	89.8	95.9	112.7	135	205.3
				1079.1	0.66	0.17	0.32	0.67	0.44	2.9
jnh1.cnf	100	850	0.7	0.86	11.2	17	16.7	18.0	22.3	31.8
				0.1	47.5	0.004	0.006	0.003	0.009	0.052
uf225-097.cnf	225	960	0.1	3.8	66.8	94.2	100	133.3	183.3	254.9
				1.95	1357.2	0.17	0.22	1.34	1.57	.99
par16-3-c.cnf	334	1332	9.6	1.02	1.04	1.08	1.18	1.68	2.23	3.06
				3.5e-5	4.3e-5	2.4e-5	5.4e-5	1.2e-4	7.35e-5	2.0e-4
f600.cnf	600	2550	1.3	7.18	7.32	8.2	10.02	13.8	18.1	25.4
				1.58e-3	5.92e-4	1.2e-2	5.3e-3	4.5e-3	4.5e-3	1.5e-2
hanoi4.cnf	718	4934	12.2	.926	.901	.986	1.15	1.59	2.1	2.99
				0	6.72e-6	3.06e-5	7.54e-5	5.67e-5	7.3e-3	6.64e-5
ii8c2.cnf	950	6689	0.1	1.6	3.3	120.9	137.7	182.6	249.3	364.4
				0.26	0.45	0.98	0.67	0.48	0.67	1.6
f1000.cnf	1000	4250	0.4	23.95	24.9	29.7	35.4	47.7	64.3	91.2
				0.011	0.017	0.028	0.017	0.034	0.053	0.145
par16-1.cnf	1015	3310	7.9	1.99	1.11	2.66	1.74	2.45	3.28	4.68
				8.95	5.16e-5	16.0	6.41e-5	1.35e-4	6.41e-5	9.54e-4
par32-2-c.cnf	1303	5206	12.7	1.03	1.07	1.17	1.28	1.64	2.17	3.09
				6.2e-6	2.82e-5	1.45e-5	3.03e-5	1.03e-4	1.68e-4	2.04e-4
ii16a1.cnf	1650	19368	0.2	1.95	6.9	85.4	90.25	118.7	154.1	225.2
				0.025	7.77	0.267	69.68	6.178	230.77	1.74
ii16b1.cnf	1728	24792	0.4	2.2	4.37	64.6	61.05	68.88	85.87	119.2
				0.178	0.78	0.18	0.16	0.46	0.36	2.74
g125.17.cnf	2125	66272	63.3	.76	0.65	0.53	0.50	0.55	0.68	-
				2.2e-5	2.0e-05	6.77e-6	2.88e-6	1.56e-5	6.9e-6	-
par32-1.cnf	3176	10277	10.9	1.1	1.17	1.33	1.61	2.28	3.18	-
				2.71e-5	2.71e-5	5.61e-5	1.43e-3	7.89e-4	2.1e-3	-
Average				6.17	15.27	36.93	39.6	50.42	66.13	110.9

Table 2: Experimental Results for DIMACS instances.

Instance	# Vars	# Clauses	Original	1x	2x	3x	4x	5x	10x
50s-10v.cnf	10	214	50	0.484	0.212	0.12	0.04	0.36	0
				9.6e-4	3.73e-4	0	0	9.6e-4	0
100s-10v.cnf	10	328	100	0.48	0.27	0.114	0.065	0.036	0
				0	0	7.11e-5	2.5e-4	2.67e-5	0
100s-20v.cnf	20	1328	100	0.506	0.252	0.127	0.06	0.023	0
				4.27e-4	3.73e-4	2.33e-5	0	1.57e-4	0
1000s-25v.cnf	25	15024	1000	0.501	0.25	0.125	0.063	0.034	0.002
				4.18e-6	1.13e-5	4.32e-6	7.67e-7	1.04e-4	6.77e-6
1000s-50v.cnf	50	40024	1000	0.499	0.251	0.137	0.063	0.037	0.0005
				1.23e-5	9.51e-6	1.62e-3	1.57e-6	1.65e-4	7.22e-7
10s-10000v.cnf	10000	99966	10	0.54	0.28	0.11	0.06	0.06	0
				9.3e-3	8.4e-3	9.89e-3	2.67e-3	4.88e-3	4.88e-3
100s-2000v	2000	199328	100	0.506	0.246	0.113	0.066	0.028	0
				2.27e-4	5.38e-4	4.01e-4	4.89e-5	1.51e-4	0
1000s-1000v.cnf	1000	990024	1000	0.502	0.251	0.126	0.062	0.032	1.67e-3
				6.77e-6	4.22e-6	4.0e-7	3.76e-4	5.0e-7	1.34e-6
10000s-200v.cnf	200	10866384	10000	0.502	0.251	0.126	0.062	0.031	9.0e-4
				7.87e-6	9.25e-6	4.17e-6	3.28e-6	2.38e-6	1.24e-7
25000s-150v.cnf	150	3382768	25000	0.499	0.25	0.124	0.063	0.031	0.001
				3.41e-6	2.34e-6	7.27e-7	3.12e-7	5.36e-7	1.1e-8
Average				0.502	0.251	0.122	0.061	0.035	6.2e-4

Table 3: Experimental Results for created SAT instance with known number of solutions.

In order to verify the effectiveness of the proposed approach, we conducted two sets of experiments. The first set consists of popular DIMACS examples shown in Table 2 we selected examples which greatly differ in terms of variables, number of clauses, and ratio of number of clauses to number of variables; it is often a good measure of difficulty to solve the problem. The first column indicates the name of the example. The next two columns indicate the number of variables and clauses in the instance respectively. The next column indicates the runtime to solve the initial instance. The next seven columns indicate the average runtime when ten different random messages are embedded as the watermark. The length of the messages are 1, 2, 5, 10, 50, and 100 times the number of variables in the example. The runtimes shown in the table are normalized against the runtime required to solve the original instance. We show the average runtime, and in addition we show the variance, in the second row for each instance as an indicator of fairness for the new watermarking procedure. Since for this type of large examples, there is no known technique to calculate all known solutions, we use runtime as the indicator of difficulty to solve a particular SAT instance. The last row shows the average normalized runtime for each signature length. A dash in our table indicates that the created instance is to large to run on the solvers.

The experimental results shown in Table 3 aim to remove the indirect measurement of fairness and credibility. We used our program to generate SAT instances with a known number of solutions to generate this test set. We calculated the number of solutions to each instance using the branch and bound exhaustive search algorithm of Section 5. The first column is the name of the instance. The next two columns indicate the number of variables and the number of clauses in each of the created instance. The next column indicates the number of solutions to created instance. The next six columns indicate the average number of solutions and variance for messages of 1, 2, 3, 4, 5, and 10 times the number of variables. We ran test for messages of 25 and 50 times the number of variables, but could not find a solution for any of these cases. For each example, we embedded ten random messages of the specified length. The last row of the table indicates the average normalized number of solutions remaining after messages of each length had been embedded. It can be seen that the average normalized number of solutions is halved for each multiple of the number of variables. It is easy to see that the variance, which appears on the second row for each instance, in all examples is very low, which clearly indicates very high fairness of the new watermarking procedure. The results in Table 3 strongly demonstrate that the procedure is able to rapidly and efficiently produce watermarks of very high credibility.

7. CONCLUSION

We presented a new SAT watermarking technique which provides the maximal possible credibility and almost always full fairnes and established a connection between the Valiant-Vazirani combinatorial isolation lemmas and the watermarking process. In order to validate the theoretical results, we applied the technique to a variety of real life and specially created instances of SAT.

8. REFERENCES

[1] R.J. Bayardo and R. Schrag."Using CSP look-back techniques to solve exceptionally hard SAT instances." Principles and Practice of Constraint Programming. pp.46-60. 1996.

[2] J.M. Crawford. "Solving Satisfiability Problems Using a Combination of Systematic and Local Search". Second DIMACS Challenge. 1993.

[3] M. Garey and D. Johnson. "Computers and intractability. A Guide to the theory of NP-completeness". W. H. Freeman and Company, New York, 1979.

[4] F. Hartung and M. Kutter. "Multimedia watermarking techniques". IEEE vol. 87, no. 7. pp. 1079-1107, June 1999.

[5] I. Hong and M. Potkonjak. "Behavioral synthesis techniques for intellectual property protection". Design Automation Conference. pp. 849-854. 1999.

[6] A. Kahng, et al. "Watermarking techniques for intellectual property protection". Design Automation Conference. pp. 776-781. 1998.

[7] J. Marques-Silva and K.A. Sakallah. "Boolean Satisfiability in Electronic Design Automation". ACM/IEEE Design Automation Conference. pp. 675–680. 2000.

[8] A.L. Oliviera. "Robust Techniques For Watermarking Sequential Circuit Designs." Design Automation Conference. pp.837-842. 1999.

[9] T. Pagarani and F. Kocan and D.G. Saab and J.A. Abraham. "Parallel and scalable architecture for solving SATisfiability on reconfigurable FPGA". IEEE 2000 Custom Integrated Circuits Conference. pp.147-150. 2000.

[10] B. Selman and H. Kautz. "Domain-Independent Extensions to GSAT: Solving Large Structured Satisfiability Problems". International Conference on Artificial Intelligence.pp. 290-295. 1993.

[11] B. Selman, H. Kautz, and B. Cohen."Local Search Strategies for Satisfiability Testing". Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge. pp. 521–532. 1993.

[12] S. Toda. "On the computational power of PP and $\oplus$P". Proc. IEEE Foundations of Computer Science, pp. 514–519, 1989.

[13] I.Torunoglu and E. Charbon. "Watermarking-based copyright protection of sequential functions". IEEE Custom Integrated Circuits Conference. pp.35-38. 199.

[14] G. Qu and M. Potkonjak. "Fingerprinting intellectual property using constraint-addition". Design Automation Conference. pp. 587-592. 2000.

[15] G. Qu, J.L. Wong, and M. Potkonjak. "Fair Watermarking Techniques". IEEE/ACM Asia and South Pacific Design Automation Conference. pp. 55-60. 2000.

[16] L.G. Valiant and V.V. Vazirani. "NP is as easy as detecting unique solutions". Theoretical Computer Science, vol.47. pp. 85-93. 1986.

[17] G. Voyatzis and I. Pitas. "The user of watermarks in the protection of digital multimedia products". Proceedings of IEEE, Special Issue on Identification and Protection of Multimedia Information, vol. 87, no. 7. pp. 1197-1207, July 1999.

[18] P. Zhong and M. Martonosi and P. Ashar. "FPGA-based SAT solver architecture with near-zero synthesis and layout overhead". IEEE Proceedings-Computers and Digital Techniques, vol.147, no.3. pp. 135-41. 2000.