Fair Watermarking using Combinatorial Isolation Lemmas

Jennifer L. Wong, Rupak Majumdar, Miodrag Potkonjak
University of California, Los Angeles, Los Angeles, CA 90095

Abstract

In addition to silicon fingerprinting and forensic engineering, watermarking is one of the most effective proactive mechanisms for intellectual property protection (IPP) of hardware and software. Numerous watermarking-based IPP techniques have been proposed that satisfy a spectrum of IPP desiderata, including full preservation of functionality, low timing, area and power overhead, transparency to the synthesis and compilation process, and resiliency against attacks. Two objectives that are very important but until now have not yet been properly addressed are credibility and fairness.

We present new watermarking techniques that specifically target credibility and fairness. Leveraging on the Valiant-Vazirani theorem, we demonstrate how these two desiderata can be achieved during the watermarking of a satisfiability (SAT) instance. 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 operational research SAT benchmark instances.

***Index Terms*—**

I. Introduction

A. Motivation

Historically, the level of integration of a state of the art system has grown at the rate of 44% annually. In 1970, the first Intel microprocessor i4004 had 2,250 transistors. Today, the state of the art microprocessor has up to 287 million transistors. The growth rate has increased even faster in the last decade. The ultra high integration level and emerging system-on-chip technologies create a productivity gap between the ability of designers to develop integrated circuits and the potential of silicon. Software, in terms of the number of lines of code for a typical application, grows at an even faster rate, almost twice as fast as hardware.

It is widely considered that the reuse of intellectual property (IP) such as IC cores and software libraries is the most economically efficient way to close this gap. One of the prerequisites for hardware and software IP reuse is the development of IPP techniques. Several approaches have been proposed, including fingerprinting of silicon dies and forensic engineering. It seems, however, that watermarking is the most effective IPP technique 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. The key observation on which all watermarking-based IPP techniques are based is the fact that many synthesis problems are associated with computationally intractable or difficult optimization problems which often have a high number of different solutions with identical or very similar quality. The key idea of watermarking-based techniques is to leverage this fact by incorporating the design signature into the design specification as additional constraints and therefore ensuring 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, relatively little is known about their theoretical underpinnings. Two issues, in particular, deserve more sound and effective treatment: the calculation of a watermark's credibility and fairness.

It is important to emphasize that neither credibility nor fairness can be defined in a unique and ultimately correct way. This is so because both credibility and fairness are related to finding a single or multiple solutions of a NP-complete problem - class of problems that are intrinsically intractable. Even from the conceptual point of view, it is not clear if a definition based on the number of solutions or one based on the required effort (e.g. time) to find a solution is more adequate. Even if one prefers one of these two options, the unavoidable logistic problem is that one cannot provide a unique operational definition for either. In the case of the number of solutions, since the problem is computationally intractable, it is impossible, to count the number of solutions for standard benchmarks. While time or memory space required to find a solution can be measured, both greatly depend on the particular algorithm and particular system used to solve the particular instance.

Intuitively, we propose to measure the strength of the proof that a considered solution is indeed produced after the addition of watermarking constraints by measuring the likelihood of the solution being selected at random. Therefore, one can define credibility as the ratio of the number of solutions that satisfy the watermarking constraints. We focus our research on the boolean satisfiability (SAT) problem due to its use to model many optimization and verification tasks in CAD and other areas such as artificial intelligence and operational research. Credibility can be defined as the overall number of solutions for the initial instance of the SAT problem. In this case, a lower ratio indicates higher credibility. Therefore, we can conclude that ultimate credibility is achieved if after the addition of a watermark, only a single solution exists for an instance.

Credibility can also be defined with respect to the effort required to find a particular solution. One can claim that credibility is high if after the addition of watermarking constraints, all or at least a majority of state-of-the-art solvers will produce with high likelihood a watermarked solution and that in the absence of the watermark they will produce with high likelihood a different solution. Note that many watermarking techniques can provide very high credibility if the length of the message is increased. From a practical point of view, a good aim for addressing credibility would be to evaluate the technique according to its ability to provide a variety of tradeoffs: strength of watermarking proof versus quality of solutions, or in the case of SAT, time required to find a solution.

Fairness can be formally defined as a function which takes into account the difficulty of finding a solution after a variety of watermarks of a specific length are embedded. The definition of difficulty can be established in multiple ways. One approach is to define fairness as the number of solutions after the watermark is embedded. Another approach is to measure how difficult it is to find a solution once the signature is embedded, which can be quantified using the runtime on common solvers. Note that one can define a variety of statistics that combine the difficulty of specific instances. For example, if we have n different watermarks and we denote by x_i either the number of solutions or the runtime required to find solution for the i^{th} watermark, then the following formula is one such option. Note that one can define a variety of statistical measures that aggregate the measured difficulties of a specific set of instances.

$$\sum_{i=0}^{n} \frac{(x_i - \overline{x})^2}{n} \tag{1}$$

Recently, Qu et al. [1] introduced the first watermarking technique which embeds instance specific constraints to ensure fairness. However, that technique is empirical and does not provide any theoretical guarantees. The new method is based on the Valiant-Vazirani result that probabilistically guarantees the isolation of a single solution of the SAT instance. The Valiant-Vazirani approach uses randomized reduction by successively adding 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 pursue the maximal length of the signature which still does 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 watermark lengths.

II. Related Work

The related work can be traced in three different research directions: the satisfiability (SAT) problem, IPP using watermarking, and algorithmic techniques for producing instances with unique solutions.

A. Satisfiability Problem

SAT is an exceptionally important optimization problem. For example, SAT was the first problem determined to belong to the class of NP-complete problems [2]. The proof of computational intractability for all other NP-complete problems has been established by direct or indirect polynomial time transformations from the SAT problem.

The SAT problem is a decision problem defined for a boolean expression. The goal is that for a given boolean expression E, decide if there is some assignment to the variables in E such that E is true. Formally, Garey and Johnson [3] define the problem as:

Problem: *Boolean Satisfiability*
Instance: *A set U of variables and a collection C of clauses over U.*
Question: *Is there a satisfying truth assignment for C?*

SAT has numerous applications both in VLSI CAD and other domains including artificial intelligence, operations research, and combinational optimization. Probably the most well known application for SAT in CAD is Automatic Test Pattern Generation (ATPG) [4], [5], [6]. SAT has also been used for deterministic test pattern generation [7], delay fault testing [8], logic verification, and timing analysis. In addition, SAT has been used for FPGA routing [9], [10], logic synthesis [11], physical design [12], and combinational equivalence checking which has been developed by [13], [14], [15].

Furthermore, SAT has also been used to solve covering problems [16], [17], physical design problems [18], linear integer linear programming problems [19], and finding prime implicants of boolean functions [20]. Also, it has been used as an optimization engine for solving 0-1 integer linear problems (ILP) in CAD and other areas [21], [19], [22], [23].

Several techniques have been developed to solve the SAT problem which include: backtrack search [24], [25], [26], local search [27], algebraic manipulation [28], [29], continuous formulation [30] and recursive learning [14], [25]. An excellent survey on SAT in Electronic Design Automation is [31].

B. Watermarking

Ultra high system-on-chip integration depends on the availability of third party hardware and software components. Therefore, to facilitate viable business models, there is a strong need for a variety of IPP techniques. A number of IPP techniques such as watermarking [32], [33], [34], [35], [1], [36], fingerprinting [37], [38], software obfuscation [39], and forensic engineering [40] have been proposed. We focus our review of related work on watermarking due to its direct relevance to this work. There are two conceptually different domains where watermarking is applied: for static artifacts and functional artifacts. A variety of techniques have been developed to watermark static artifacts [41], [42] such as images [43], [44], video [45], audio [46], and text [47]. Watermarks can also be placed in graphical objects such as 3D graphics [48], [49] and animation [50]. The essential property of all watermarking techniques for static artifacts is that they leverage on the imperfections of human perception. The main objectives of watermarking techniques for static artifacts include requirements for global placement of the watermark in the artifact, resiliency against removal, and suitability for rapid detection.

Watermarking techniques have also been developed for functional artifacts, such as software and integrated circuits design. The common denominator for functional artifacts is that they must fully preserve their functional specifications and therefore can not leverage the same principles as used for watermarking static artifacts. Functional artifacts can be specified and therefore watermarked at several levels of abstraction such as system level designs, FPGA designs [51], at the behavioral and logic synthesis levels, and physical designs [33], [34]. Techniques have also been developed for watermarking of DSP algorithms, sequential circuits, sequential functions [52], [53], [54], [55], and analog designs [56], [57], [58], [59].

There are two different types of watermarking techniques: horizontal and vertical [60], [61]. Horizontal techniques embed watermarks into designs by structurally altering the design itself. The majority of hardware IPP techniques are vertical watermarking approaches that embed the watermark into the design by superimposing additional constraints during synthesis or compilation. The key idea of watermarking is to embed constraints into the design which represent the user's signature in a unique way while introducing low overhead to the design. The VSI Alliance [62] is currently developing the industry standard for watermarking hardware.

There are two main advantages of the new technique over previously published techniques. First, the new technique provides 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.

C. Algorithmic Techniques

The basis for our work is a paper by Valiant and Vazirani [63]. They 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. One of many corollaries of their result is a lemma for reducing the number of solutions of an arbitrary SAT instance.

There have been a number of techniques that leverage the Valiant-Vazirani results. For example, Watanabe [64] used it as a starting point to develop a framework for testing average performance of algorithms for a given NP search problem with respect to some distribution on the instances. Also, Emden-Weinert et al. [65] proved that some types of graphs have a unique k coloring solution.

III. Preliminaries

In this Section, we briefly survey the constraint-based watermarking methodology. In particular, we summarize several answers to frequently asked questions about the most sensitive steps in the watermarking process. Figure 1 illustrates the generic watermarking technique. There are two inputs, the initial instance of the optimization problem (which corresponds to optimization synthesis or a compilation problem) and the owner's signature. The signature is translated into a set of additional constraints using the proposed watermarking technique, which should satisfy tests for randomness. Once the additional constraints are defined, they are combined with the original instance specification creating the overconstrained specification of the problem instance. The overconstrained instance is then solved using any solver for the problem. The solution obtained from the solver is a watermarked solution, since the solution satisfies both the initial instance specification and the additional constraints added from the signature bitstream. The last step of the process flow is to evaluate the effectiveness of the watermark.

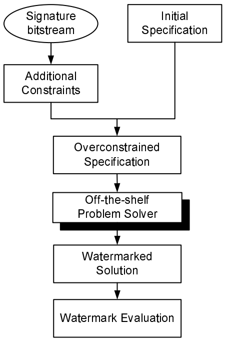

Fig. 1. Watermarking-Based IPP Process Flow.

In order to generate a bitstream for watermark encoding which is representative of the encoder's signature we introduce the following process shown in Figure 2. By using the PGP encoding scheme, a sufficiently randomized bitstream is created resulting in watermarking constraints which are difficult to imitate and detect.

The figure illustrates the encoding and verification process for watermarking. The Pretty Good Privacy (PGP) software package is used to encode the watermark signature with the users private key. A seedfile is generated from PGP software that is used to create a signature-related bit-stream. The bit-stream which is the output of the RC4 stream cipher is a cryptographically strong pseudorandom bit-stream. This bitstream is the basic signature which is used by the watermarking techniques.

To verify a signature, one must show that both the signature is present in the SAT solution and that the signature corresponds to the textfile and the PGP public key of the supposed owner. Demonstrating that the signature exists in the SAT solution is achieved by demonstrating that the solution satisfies all the constraints claimed by the author. One can show that the signature corresponds to the textfile and the owner's public key by running PGP.

The crucial observation is that during the creation of additional constraints suitable for intentional watermarking is that one has to establish a well defined ordering for the components of the instance. In the case of the SAT problem the components are variables and clauses. This can be done in two ways, either by using industry imposed standards or by using properties

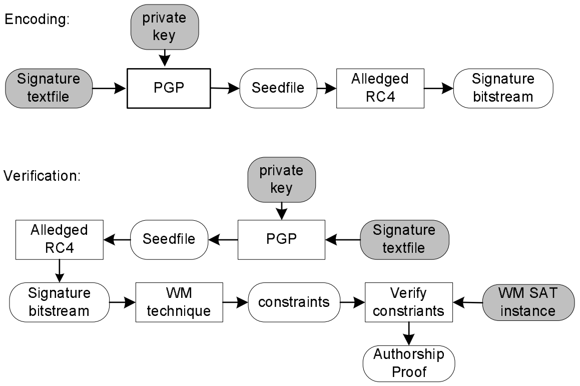

Fig. 2. Procedure for translation of arbitrary signature to infinite random string.

of the designs. For example, for the second option, 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. Once when this well defined ordering is available, we can assign a specific rank to each variable and to each clause. After this step, additional constraints can be added in a unique way to the instance and one can establish a unique relationship between each bit in the users signature and each new constraint added to the instance. Furthermore, by following the ordering, one can conduct reverse engineering of the original signature, which is one of the key objectives for establishing proof of authorship.

The output of the process is a watermarked design which can be analyzed according to standard watermarking desiderata, which include high credibility, high resiliency against attacks, low overhead, complete transparency to the standard problem solving tools, and partial protection and fairness. 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.

A. Motivational Example

We illustrate the metrics, credibility and fairness, and how they are achieved using the new watermarking approach on the following example. For the sake of simplicity, clarity, and brevity, we adopt the definitions of credibility and fairness that are based on the number of solutions before and after watermarking. Consider the following SAT formula over four variables x_1, x_2, x_3, x_4. Originally, this instance has nine satisfying assignments.

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

According to the proposed metric for fairness, if all watermarks of a given length reduce the number of solutions remaining for the instance to the same number, we conclude that the watermarking approach is completely fair. In that case, no preference is given by the technique to any particular signature.

Consider all watermark signatures of length four bits. Table I shows the 16 different watermark signatures in the first column. We embed each of the signatures into the SAT formula f, using three watermarking techniques. The first two techniques are previously proposed watermarking techniques [36]: adding clauses (AC) and deleting literals (DL). The third technique is the proposed isolation lemma-based (IL) watermarking technique.

The AC clauses technique embeds the signature into the SAT formula f by creating new clauses over the existing variables. In this case, we use every two bits of the signature to create a clause. The first bit of the signature corresponds to variable x_1 and the second to x_2. If the bit is zero, the variable is added to the clause in the uncomplemented form. Otherwise, it is added in the complemented form. For example, for the watermarking signature 0011 the following clauses are added to the SAT formula f: $(x_1 \vee x_2)(\overline{x}_1 \vee \overline{x}_2)$.

For the DL watermarking technique the signature is embedded by removing appearances of variables from a clause. By removing literals (appearances of variables in either complemented or uncomplemented form) from a clause, the clause becomes more constrained and therefore more difficult to satisfy. For every two literals in a clause, we select one of them to removed based on a bit in the signature. For example, to embed a signature we consider each clause one at a time. In the first clause $(x_1 \vee x_3 \vee \overline{x}_4)$, we consider the first pair of literals x_1 and x_3. If the bit in the signature is zero, we delete variable x_1, if it is 1 then variable x_3 is removed. To embed watermark signature 0011, literal x_1 is deleted. Since there is no other pair of literals in this clause, the second clause is considered $(x_2 \vee x_4)$. The second bit of the signature is a 0, therefore x_2 is removed from the clause. The remaining two bits of the signature are embedded by removing literal x_2 and x_4 from the third clause. The final SAT formula in this case is $f = (x_3 \vee \overline{x}_4)(x_4)(\overline{x}_1 \vee \overline{x}_3)(\overline{x}_2 \vee x_3 \vee \overline{x}_4)$.

The final scheme is the new isolation lemma-based (IL) approach. The scheme embeds the watermarking signature into the SAT formula by creating new clauses over variables selected by the signature. The additional clauses are created in such a way that they eliminate a subset of the possible signatures. Note that the proposed watermarking technique requires at least one bit in the string to be a one, which happens with high probability even for strings of moderate length.

For example, consider a substring of the watermarking signature with its length equal to the number of variables in the instance. In this case, the SAT formula f has four variables. In order to embed the signature 0011 into the instance, we consider the position of the one's. In this case there are two one's and they correspond to variables x_3 and x_4. The Valiant-Vazirani isolation lemma-based approach is used to translate these variables into the following additional clauses (along with the introduction of a new variable y_1):

Sig. of Length 4	AC	DL	IL
0000	8	4	-
0001	5	2	3
0010	6	4	5
0011	5	2	3
0100	5	6	2
0101	6	6	4
0110	4	4	6
0111	3	4	4
1000	6	4	5
1001	4	2	5
1010	7	4	5
1011	4	2	5
1100	5	6	4
1101	3	6	4
1110	4	4	4
1111	6	4	4
Var	1.929	2.133	1.028

TABLE I
MOTIVATIONAL EXAMPLE: NUMBER OF SOLUTIONS REMAINING AFTER EMBEDDING SIGNATURES OF LENGTH 4 INTO THE SAT INSTANCE USING THREE DIFFERENT WATERMARKING TECHNIQUES.

$(\overline{y}_1 \vee \overline{x}_3 \vee \overline{x}_4)(\overline{y}_1 \vee x_3 \vee x_4)(y_1 \vee \overline{x}_3 \vee x_4)(y_1 \vee x_3 \vee \overline{x}_4)(\overline{y}_1)$. The detection of the signature 0011, for the motivation example, can be accomplished by checking that the solution which the owner claims as his own satisfies this overcontrained SAT instance.

After embedding each watermark signature using each of the three watermarking approaches into the above instance, the number of solutions which satisfy the new SAT formula is enumerated in Table I. The second column indicates the number of solutions found using exhaustive search when watermarking with the adding clauses (AC) technique, the third column corresponds to the deleting literals (DL) technique. Finally, the last column displays the number of solutions remaining after embedding the watermarks using the proposed isolation-lemma (IL) technique. The last row of the table displays the variance in the number of remaining solutions for each approach. The variance for the proposed technique is significantly lower than that of the other two techniques. This small variance indicates that even on very small examples using very short messages the technique performs fairly.

In order to illustrate the impact of credibility of a specific watermarking technique, we consider adding watermarks of length, 4, 8, 12, 16, 20, 24, 28 and 32 bits to the same SAT formula using the three techniques. Twenty different watermarks of each length were embedded using the three aforementioned techniques. The number of solutions after embedding the watermark signature was enumerated in each case and the measured results are presented in Table II. The first row of the table indicates the length of the watermark message. For each of the proposed techniques three rows are given. The first of these rows indicates the average number of solutions after embedding the twenty randomly selected strings of the indicated watermark length. The second row shows the minimum and maximum number of solution which satisfy the watermarked instance, while the final row denotes the variance in the number of solutions over the twenty different watermark signatures.

The results shown in Table II indicate how the number of solutions for each technique reduces as longer watermarking signatures are embedded. For the AC watermarking technique, as the watermark signature increases in length more clauses are added to the instance. However, many of the added clauses are identical to each other and therefore place no additional constraints on the instance. The small signature lengths significantly reduce the number of solutions, due to the addition of unique constraints, but as the signature lengths increase the number of solutions decreases very slowly.

In the case of the deleting literals (DL) technique, only a few variables can be eliminated from each clause, until the clause becomes overconstrained (a single variable clause). Therefore after embedding five bits of the signature, the remaining watermark signature is not used. This result is seen in the column for the signature length of eight bits. The average number of solutions decreases in this case from the four bit signatures case. However, for all signatures greater than this length, the results are unchanged.

Finally, for the proposed isolation lemma technique as the length of the signature increases, the average number of solutions is approximately halved. Therefore, there is a predictable trade-off between the number of solutions remaining after watermarking (credibility) and the length of the watermark that is embedded. Note that the other proposed techniques have no predictability for the number of solutions as the length of the watermark signature increases. Furthermore note that the data in Table II shows low variance between signatures of equal length which additionally illustrates the fairness of the technique. A more comprehensive evaluation of the technique is given in the Experimental Results Section.

IV. CREATING UNIQUE AND FAIR SOLUTIONS TO SAT

In this Section, we present the mathematical foundation for the new watermarking technique. We start by presenting the Valiant-Vazirani Isolation Lemma and an illustration of its application on a small example. We conclude the Section by briefly discussing how the technique can be applied to other computationally intractable problems beyond SAT.

A. Valiant-Vazirani Isolation Lemma

The method is based on a combinatorial result of [63] that isolates a solution of a conjunctive normal form (CNF formula) by 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 a 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 will consequently construct, 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

	Sig. Length (bits)	4	8	12	16	20	24	28	32
Adding Clauses	Ave. # of Solutions	5.15	2.7	1.2	0.4	0.05	0.05	0.05	0.0
	Min/Max Solutions	3/8	0/5	0/3	0/3	0/1	0/1	0/1	0/0
	Variance	1.40	2.22	1.85	0.88	0.05	0.05	0.05	0.0
Deleting Literals	Ave. # of Solutions	4.05	3.3	3.3	3.3	3.3	3.3	3.3	3.3
	Min/Max Solutions	1/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6
	Variance	2.47	2.64	2.64	2.64	2.64	2.64	2.64	2.64
Isolation Lemma	Ave. # of Solutions	3.89	1.84	1.11	0.58	0.37	0.16	0.16	0.05
	Min/Max Solutions	2/5	0/3	0/3	0/3	0/3	0/1	0/1	0/1
	Variance	0.99	0.70	0.88	0.70	0.58	0.14	0.14	0.05

TABLE II

MOTIVATIONAL EXAMPLE: RELATIONSHIP BETWEEN THE NUMBER OF SOLUTIONS AND THE LENGTH OF THE SIGNATURE.

the original formula f. However, the probability that a random algorithm picks exactly this satisfying assignment is low.

We now outline the construction. The treatment is from [63]. We omit the proofs of correctness. For an alternate treatment of the process using 2-universal hash functions, see [66].

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-dimensional $\{0,1\}$ 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. *If f is any CNF formula in $x_1, x_2, \ldots, x_n$ 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 $v \cdot w_1 = v \cdot w_2 = \cdots = v \cdot w_k = 0$. 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 $x_1, \ldots, x_n$.*

Proof. We show the lemma for $k = 1$. 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 $i_1, \ldots, i_j$ are the indices of the x_i that have value 1 in w_1. The function f_1 is the CNF equivalent of f'_1:

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

The intuition behind the construction is the following surprising fact. Let S be a subset of $\{0,1\}^n$. Define the sets

$$S_1 = \{v \mid v \in S, v \cdot w = 0\} \text{ and } S'_1 = \{v \mid v \in S, v \cdot w = 1\}.$$

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.

```
/* Precondition: $f$ is of the form $x_1 \oplus x_2 \oplus \ldots x_k \oplus 1$ */
$convertToCNF$(formula $f$){
    $let\ \{y_1, y_2, \ldots, y_k\}\ be\ new\ variables$;
    /* let $a \Leftrightarrow b \oplus c$ denote the CNF formula
    $(\overline{a} \vee \overline{b} \vee \overline{c})(\overline{a} \vee b \vee c)(a \vee b \vee \overline{c})(a \vee \overline{b} \vee c)$ */
    return $((y_1 \Leftrightarrow x_1 \oplus x_2)(y_2 \Leftrightarrow y_1 \oplus x_3) \ldots$
                $(y_{k-1} \Leftrightarrow y_{k-2} \oplus x_k)(y_{k-1} \oplus 1))$; }
formula $addOneConstraint$(formula $f$){
    $pick\ n\ bitsfor\ w\ from\ signature\ bitstream$;
    $let\ \{i_1, \ldots, i_k\}\ be\ positions\ of\ the\ 1\ entries\ in\ w$;
    return $f \wedge convertToCNF(x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_k} \oplus 1)$; }
/* Precondition: $f$ is a CNF formula over variables
    $\{x_1, \ldots, x_n\}$ */
formula
$generateConstrainedFormula$(formula $f$){
    $get\ t\ from$ {1, ..., n} $from\ signature\ bitstream$;
    for ($i$ = 1 to $t$) do
        $f = addOneConstraint(f)$;
    od }
```

Fig. 3. Pseudo code for watermarking using combinatorial isolation lemmas.

The overall construction is simply 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 give the technical result that formalizes the above intuition.

Lemma 2. *Let $S \subseteq \{0,1\}^n$. Suppose $w_1, \ldots, w_k$ 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 $P_n(S) \geq \frac{1}{2}$.*

Proof. This is the main construction of [63].

Figure 3 shows the algorithm to produce the final formula (conjoined with the additional constraints). The function *addOne-Constraint*() 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 $generateConstrainedFormula()$ 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 new variables).

Example. As an example of the application of method, consider the CNF formula

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

over the variables $\{x_1, x_2, x_3, x_4\}$. This expression has ten satisfying assignments, namely $\{0000, 0001, 0011, 0110, 0111, 1001,$ $1011, 1101, 1110, 1111\}$. The watermarking process begins with *generateConstrainedFormula*. In this step, we select a number t from our bitstream which is between 1 and n, where n is the number of variables in the original instance specification. In our case, $n = 4$. Suppose that our bitstream generates $t = 2$. Therefore, *addOneConstraint(f)* is called twice.

In the first invocation, we select a watermark string w_1 from the bitstream of length equal to the number of variables in the original instance. Suppose the first watermark w_1 is 0101. We associate each bit of the watermark with a single variable in the original instance. Therefore, x_1 is associated to 0, x_2 to 1, x_3 to 0, and x_4 to 1. In this case, there are two 1's in the watermark that are associated with x_2 and x_4. The following formula f_1 is then created and passed to the *convertToCNF* function.

$$f_1 = (x_2 \oplus x_4 \oplus 1) = (y_1 \Leftrightarrow x_2 \oplus x_4)(y_1 \oplus 1)$$

In *convertToCNF*, the formula f_1 is converted to CNF form. In this case, the first term $(y_1 \Leftrightarrow x_2 \oplus x_4)$ translates into four new CNF clauses $(\overline{y_1} \vee \overline{x_2} \vee \overline{x_4})(\overline{y_1} \vee x_2 \vee x_4)(y_1 \vee \overline{x_2} \vee x_4)(y_1 \vee x_2 \vee \overline{x_4})$. The final term $(y_1 \oplus 1)$ translates solely to a single clause, $(\overline{y_1})$. These clauses are appended to the original formula f creating f' and returned to *generateConstraindFormula*.

$$f' = (x_1 \vee \overline{x}_2 \vee x_3)(\overline{x}_1 \vee x_3 \vee x_4)(x_2 \vee \overline{x}_3 \vee x_4)(\overline{y_1} \vee \overline{x_2} \vee \overline{x_4})$$
$$(\overline{y_1} \vee x_2 \vee x_4)(y_1 \vee \overline{x_2} \vee x_4)(y_1 \vee x_2 \vee \overline{x_4})(\overline{y_1})$$

Formula f' now has four satisfying assignments $\{0000, 0111, 1101, 1111\}$. In this first step, the number of solutions was reduced from 10 to 4, approximately half. In the second iteration, *addOneConstraint* is called with f'. Note, only the original variables (x_i) are used to watermark the instance throughout the watermarking process despite the fact that new variables (y_i) are created. A second watermark string from the signature bitstream is selected, $w_2 = 0011$. In this case, the 1's correspond to variables x_3 and x_4. The constraint below, f_2, is created and encoded to CNF form.

$$f_2 = (x_3 \oplus x_4 \oplus 1) = (y_2 \Leftrightarrow x_3 \oplus x_4)(y_2 \oplus 1)$$

The formula f'' is the overconstrained instance after both w_1 and w_2 have been encoded into the original formula f. This formula has only three satisfying assignments $\{0000, 0111, 1111\}$. The assignment 1101 was eliminated by the constraint $(y_2 \vee x_3 \vee \overline{x_4})$. Note, that now the instance has the following form.

$$f'' = (x_1 \vee \overline{x}_2 \vee x_3)(\overline{x}_1 \vee x_3 \vee x_4)(x_2 \vee \overline{x}_3 \vee x_4)(\overline{y_1} \vee \overline{x_2} \vee \overline{x_4})$$
$$(\overline{y_1} \vee x_2 \vee x_4)(y_1 \vee \overline{x_2} \vee x_4)(y_1 \vee x_2 \vee \overline{x_4})(\overline{y_1})$$
$$(\overline{y_2} \vee \overline{x_3} \vee \overline{x_4})(\overline{y_2} \vee x_3 \vee x_4)(y_2 \vee \overline{x_3} \vee x_4)(y_2 \vee x_3 \vee \overline{x_4})(\overline{y_2})$$

The beauty of the technique is that the final algorithm is extremely simple (it involves only some random choices), yet it yields **optimal** results with high probability.

B. Beyond SAT

In this Subsection, we briefly discuss potential applications of the new watermarking technique to other optimization problems. All the proposed applications have one common aspect: they leverage the ability to polynomially transform an instance of an intractable optimization problem into an instance of the SAT problem. We begin with three examples from literature and finish this Subsection with an example.

Barth presented the generalization of the Davis-Putnam enumeration technique for solving the SAT problem in such a way that the procedure can be used to solve an arbitrary 0-1 integer linear program [19]. Three years later, a new SAT algorithm, GRASP, was proposed as an optimization engine for computation of prime implicants via solving the binate covering problem [20]. These techniques have been further generalized within frameworks of SAT-based linear search and SAT-based branch and bound procedures to further enhance experimental results [31].

Recently, several, conceptually similar, approaches has been developed that enable reduction of a particular 0-1 ILP problem into a SAT instance with a pseudopolynomial overhead [21], [22], [23]. These techniques conduct translation of each ILP constraint into a linear number of SAT constraints. The coefficient in front of the linear term is a logarithmic function of the constants that appear in the corresponding ILP constraint.

One can also construct customized pseudopolynomial approaches that do not require any specific SAT algorithm and that only leverage the transformation to an instance of the SAT problem. We will demonstrate one such approach by transforming an arbitrary instance of the maximal independent set (MIS) problem into an instance of the SAT problem. The transformation is pseudo-polynomial, i.e. the instance size of the SAT problem is a polynomial function which has as the largest exponent the value of the cardinality of the MIS.

Therefore, it can be used either for watermarking small to medium MIS solutions, or parts of large MIS solutions. The MIS problem can be stated in the following way. A graph G with n vertices is given. Also, the incident matrix with entries e_{ij} is given, where a value of 1 indicates that there is an edge between nodes i and j in G. Otherwise, the entry has a value of zero. The goal is to find a set of elements with cardinality at least K, such that no two nodes that are elements of the set have an edge between them. The starting point for the transformation is the specification of the MIS problem as an instance of 0-1 linear program. Specifically, with each node i, we associate variable x_i that has value 1 if the node is selected for the MIS solution and has value 0 otherwise. The objective function is to maximize the sum of x_i. The constraints are that for each entry (i,j) that has value 1, we have constraint that $x_i + x_j \leq 1$.

An instance of the 0-1 linear program can be translated into an instance of SAT in the following way. Each variable x_i corresponds to variable x_i in the SAT instance. We use

```
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 constancy. }
9. eliminateSolutions() {
10. Create clauses which prevent non-solutions
      from being solutions. }
```

Fig. 4. Algorithm for the creation of a SAT instance with a specific number of solutions.

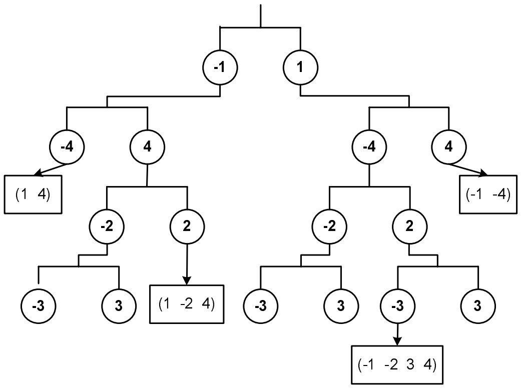

Fig. 5. Branch and Bound Tree for the creation of a SAT instance with four variable and five solutions.

the same notation for both problem formulations to enhance intuition. It is easy to see that each constraint corresponds to a clause. In order to ensure that at least K variables are assigned to 1, we create all clauses of size $K+1$ that contain all combinations of the variables. There are (n over K) such clauses. All that is required is to conduct the watermarking procedure on the instance of the SAT problem in order to create a watermarked solution of the MIS instance.

V. Software Experimental Environment

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

At the intuitive level the program for creating an instance of SAT with a known number of solutions consists of 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 diversity of their mutual literal appearance. Note, that two variable x_i and x_j can appear in a total of four combinations together: $x_i x_j$, $\overline{x}_i x_j$, $x_i \overline{x}_j$, $\overline{x}_i \overline{x}_j$. We order the variable pairs which appear in the fewest number of combinations first in order to cut the solution space as rapidly as possible. Next, we add clauses to eliminate all non-valid solutions. Finally, in the last phase we alter some of the clauses and add additional clauses to better hide the structure of the instance. Figure 4 shows pseudo code for the creation of the SAT instance.

To clarify the process, consider the following example. Our goal is to create a SAT instance defined using 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 the least number of pair combinations. As a result of pair x_1 and x_4 appearing together in only two forms $\overline{x}_1 x_4$ and $x_1 \overline{x}_4$, 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 as shown in Figure 5. We begin by adding variable x_1. By examining our solutions, we see that x_1 appears in both forms x_1 and $\overline{x}_1$, therefore we can not 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 that 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 appropriate clauses. The created instance is as follows.

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

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

The second program, also uses the branch and bound technique to enumerate all solutions of a given instance. We implemented the algorithm shown in Figure 3 for watermarking SAT instances using the combinatorial isolation lemmas. The watermarked instances were tested on the following public domain SAT solvers: WalkSAT [67], zChaff [68], and Rel_SAT [69].

VI. Experimental Results

In this Section, we present simulation results that evaluate the effectiveness of the combinatorial isolation lemma-based watermarking technique. We first present the experimental result related to credibility. Then we present experimental results

Instance	# Vars	# Clauses	Orig. WalkSAT (sec)	Orig. zChaff (sec)
par8-1-c.cnf	64	254	0.1	0.01
jnh1.cnf	100	850	0.7	0.01
uf225-097.cnf	225	960	0.1	4.86
par16-3-c.cnf	334	1332	9.6	4.38
f600.cnf	600	2550	1.3	-
hanoi4.cnf	718	4934	12.2	2.67
ii8c2.cnf	950	6689	0.1	0.05
f1000.cnf	1000	4250	0.4	-
par16-1.cnf	1015	3310	7.9	1.46
par32-2-c.cnf	1303	5206	12.7	-
ii16a1.cnf	1650	19368	0.2	-
ii16b1.cnf	1728	24792	0.4	57.51
g125.17.cnf	2125	66272	63.3	-
par32-1.cnf	3176	10277	10.9	-

TABLE III

DIMACS instances with original runtime on WalkSAT [67] and zChaff solvers.

for fairness. Both techniques are analyzed using measures based on the number of solutions as well as required runtime.

SAT is a decision problem. Therefore, there is no overhead in terms of impact on the quality of the solution. All solutions are of equal quality. The overhead in terms of runtime can be directly recorded from the increase in runtime of the solvers. The proposed watermarking technique is completely transparent to all available SAT solvers. SAT instances are intrinsicly non-partitionable and therefore it is inappropriate to discuss partial protection in the this case. Finally, it is important to emphasize that the main protection of any integrated circuit watermarking technique against an arbitrary attack is not that the attacker can not reuse a part of the solution to find another solution, but in the inherent structure of the design process where alternation of the design at the higher levels of synthesis process inevitably have as a side effect requirements for more comprehensive alternations of the solution at lower levels of abstractions. Therefore, the attacker is forced to spend significant time and effort to find a suitable new solution; essentially he/she must redo the entire design.

There are two main conceptual difficulties in evaluating the proposed watermarking technique. The first is that the technique is used to watermark instances of an NP-complete problem. Therefore, it is unlikely that one can guarantee to find a solution, and even less likely to find all solutions. The second problem is that the exact performance of an experimentally evaluated watermarking technique is most likely correlated, at least to some unknown extent, to a particular solver used to solve the instances of the SAT problem.

In order to resolve these two conceptual problems and to quantify the effectiveness of the proposed watermarking approach, we conducted two sets of experiments. The goal of the first set was to establish how well the technique performs on standard benchmarks. Note that for this type of instances, the number of solutions is unknown. The set consists of popular DIMACS examples shown in Table III. In order to enhance the diversity, we selected examples that significantly differ in terms of the number variables, number of clauses, and the ratio of number of clauses to number of variables. The last criteria, ratio of number of clauses to number of variables, is often a good measure of the difficulty to solve the instance. We present the name of the instance along with the number of variables and number of clauses for a subset of the DIMACS instances. The last two columns indicates the runtime, in seconds, to solve the initial instance using the WalkSAT solver and the zChaff solver. The second set of experiments was performed on a set of instances that were constructed in such a way that the number of solutions is known, as described in the previous Section.

A. Credibility

We first evaluated runtime-based credibility using DIMACS instances. In Figure 6, we present the average normalized runtime for each DIMACS instance after 100 watermarks of each signature length are embedded. The length of the watermark signatures are 1, 2, 5, 10, 25, 50, 100 times the number of variables in each instance. On the two horizontal axes we present the DIMACS instances and the length of the embedded watermarks. On the vertical axis, the normalized runtime using the WalkSAT solver is shown in seconds. Note that the new watermarking technique provides a continuous smooth trade-off between the length of watermark and runtime and therefore it is well suited to be used as a high credibility IPP technique.

Additionally, we performed an identical analysis using the zChaff solver. The normalized runtime for instances which could be solved by the zChaff solver after embedding the various length watermarks are shown in Table IV. Instances which are not listed in this Table, could not be solved by zChaff under the standard time limit.

In order to evaluate credibility trade-offs using a measure that is based on the number of solutions, we tested the approach on instances with the known number of solutions. These instances were created using the approach presented in Section V. The number of variables for each instance varied from 10 to 10,000 and the number of solutions ranged from 10 to 25,000. One hundred watermarks of lengths 1, 2, 3, 4, 5, and 10 times the number of variables in each instance were embedded using the proposed technique. After the addition

	par8-1-c	jnh1.cnf	uf225-097	par16-3-c	hanoi4	ii8c2	par16-1	ii16b1	ii16a1
1x	0.10	1.70	0.42	2.35	1.68	1.62	0.80	1.14	0.96
2x	0.73	1.60	1.25	1.18	1.35	1.82	1.21	1.54	0.93
5x	0.10	1.70	0.42	2.35	1.68	1.62	0.80	1.14	0.88
10x	9.60	7.78	1.54	-	-	2.38	-	1.12	1.39
25x	-	-	108.15	-	-	2611.47	-	32.20	-
50x	-	-	-	-	0.04	-	-	-	-
100x	-	416.00	-	299.86	-	-	-	-	-

TABLE IV

THE CREDIBILITY IS DEMONSTRATED BY CONTINUOUS TRADE-OFF BETWEEN THE STRENGTH OF WATERMARK (LENGTH) AND REQUIRED RUNTIME (ZCHAFF) ON DIMACS EXAMPLES.

of the watermark, the number of solutions still valid were enumerated. In Figure 7 we present the results. The created instances are labelled as the number of solutions - the number of variables and are denoted on one horizontal axis. The other horizontal axis shows the length of the watermark. The average normalized number of solutions remaining after the embedding of the watermark of a given length is shown on the vertical axis.

We present the normalized average number of solutions remaining after the addition of the watermark of a given length for each instance in Table V. The normalization is conducted against the initial number of solutions before watermarking. It is easy to see that in Figure 7 and Table V, the number of solutions scales almost exactly as predicted by the combinatorial isolation lemma.

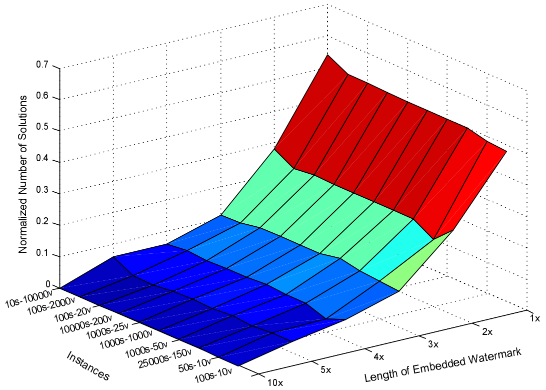

Fig. 7. Experimental results for credibility on created SAT instance with known number of solutions.

B. Fairness

To evaluate the fairness of the proposed technique, we evaluated both runtime and solution count-based measures. Experiments were performed on the DIMACS instances, shown in the columns of Table VI. The original runtime required by the WalkSAT solver to solve each instance (in seconds) is shown in the second row. For each instance, we embedded 100 watermarks of 5, 10, 25, 50 and 100 times the number of variables in the instance. For each watermark length, we present five rows of measured parameters: average runtime, minimum runtime, maximum runtime, best and worst case difference in runtime, and the variance in runtime. In the case of par32-1 and g125.17 watermarks of 100x the number of variables we added, but the WalkSAT solver was unable to find a solution in a reasonable amount of time. Note that if variance is used as the measure, zero indicates perfect fairness. In more than 90% of the cases the variance is less than 0.1, and in 95% of cases the worst case difference is less than 0.1. Additionally, in all cases where there is high variance (greater than 1), this is due to a few outlier runtimes, which in all cases were runtimes below the average.

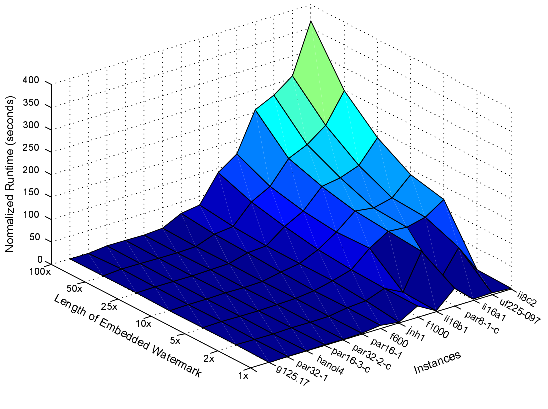

Fig. 6. The credibility is demonstrated by continuous trade-off between the strength of watermark (length) and required runtime (walkSAT) on DIMACS examples.

Evaluation of the technique using instances with the known number of solutions was performed on examples created with a specified number of variables and solutions. For this purpose we used the same instances evaluated in the experimental results for credibility validation. As in the case of the DIMACS instances, data for the average, minimum, maximum, worst case (WC) number of solutions and variance in the number of solutions are presented in Table VII. The maximum variance between the number of solutions remaining after the addition of the watermark of given length is 0.009. In all but one case when the worst case (WC) difference in number of solutions is greater than 1, there were a few watermarks which resulted in no remaining solutions.

VII. CONCLUSION

We present a new SAT watermarking technique that provides mechanisms for producing high credibility and strong fairness. This is accomplished by establishing a connection

	1x	2x	3x	4x	5x	10x
Average	0.502	0.251	0.122	0.061	0.035	6.2e-4

TABLE V

EXPERIMENTAL RESULTS FOR CREDIBILITY. AVERAGE RUNTIME FOR CREATED INSTANCES WHERE NUMBER OF SOLUTIONS IS KNOWN.

Instance	uf225-097	ii8c2	ii16a1	f1000	ii16b1	jnh1	f600	par16-1	par16-3-c	par32-1	hanoi4	par32-2-c	g125.17
Orig Runtime	0.1	0.1	0.2	0.4	0.4	0.7	1.3	7.9	9.6	10.9	12.2	12.7	63.3
5x													
Ave Runtime	9.42	12.09	17.08	11.67	25.84	11.9	10.66	10.99	10.4	14.6	12.03	14.87	33.76
Min Runtime	9.4	12.0	16.9	11.6	25.6	11.8	10.4	10.9	10.3	14.4	11.9	14.8	33.5
Max Runtime	9.5	12.2	17.2	11.8	26.1	12.0	10.8	11.1	10.5	14.7	12.1	14.9	34.1
WC Difference	0.011	0.017	0.018	0.017	0.019	0.017	0.038	0.018	0.019	0.021	0.017	0.007	0.018
Variance	0.002	0.010	0.011	0.005	0.029	0.002	0.020	0.005	0.002	0.007	0.005	0.002	0.027
10x													
Ave Runtime	10	13.77	18.05	14.16	24.42	11.69	13.03	13.78	11.35	17.59	14.07	16.36	31.76
Min Runtime	9.9	13.6	13.5	14.1	24.3	11.6	12.8	13.7	11.3	17.3	13.9	16.3	31.6
Max Runtime	10.1	13.9	19.4	14.2	24.8	11.8	13.1	13.9	11.5	18.5	14.2	16.5	31.9
WC Difference	0.020	0.022	0.327	0.007	0.020	0.017	0.023	0.015	0.018	0.068	0.021	0.012	0.009
Variance	0.002	0.007	2.787	0.003	0.026	0.003	0.009	0.004	0.005	0.170	0.011	0.005	0.012
25x													
Ave Runtime	13.33	18.26	23.74	19.09	27.55	12.58	17.99	19.32	16.2	24.86	19.42	20.81	34.97
Min Runtime	13.1	18.2	23.4	19.0	27.0	12.5	17.8	19.2	16.1	24.4	19.3	20.7	34.6
Max Runtime	13.5	18.4	25.1	19.2	28.0	12.6	18.1	19.5	16.3	25.5	19.6	21.0	35.5
WC Difference	0.030	0.011	0.072	0.010	0.036	0.008	0.017	0.016	0.012	0.044	0.015	0.014	0.026
Variance	0.013	0.005	0.247	0.005	0.074	0.002	0.008	0.008	0.011	0.094	0.008	0.017	0.062
50x													
Ave Runtime	18.33	24.93	30.82	25.72	34.35	15.6	23.51	25.92	21.43	34.71	25.45	27.64	43.21
Min Runtime	18.1	24.8	22.2	25.6	34.1	15.5	23.4	25.8	21.3	34.2	22.5	27.4	43.0
Max Runtime	18.6	25.1	32.2	25.9	34.9	15.7	23.7	26.0	21.5	35.9	25.9	27.9	43.5
WC Difference	0.027	0.012	0.324	0.012	0.023	0.013	0.013	0.008	0.009	0.049	0.134	0.018	0.012
Variance	0.016	0.007	9.231	0.008	0.058	0.004	0.008	0.004	0.007	0.250	1.083	0.027	0.028
100x													
Ave Runtime	25.49	36.44	45.05	36.49	47.68	22.29	32.99	36.98	29.39	-	36.51	39.28	-
Min Runtime	25.3	36.3	44.7	36.3	47.2	22.1	32.7	36.8	29.1	-	36.4	39.1	-
Max Runtime	25.6	36.6	45.7	36.8	49.4	22.6	33.2	37.6	29.5	-	36.7	39.6	-
WC Difference	0.012	0.008	0.022	0.014	0.046	0.022	0.015	0.022	0.014	-	0.008	0.013	-
Variance	0.010	0.016	0.069	0.023	0.440	0.025	0.025	0.060	0.019	-	0.010	0.033	-

TABLE VI

FAIRNESS RESULTS FOR THE DIMACS BENCHMARKS. THE FIRST COLUMN INDICATES MEASURED PARAMETER. THE NEXT THIRTEEN COLUMNS INDICATE THE DIMACS INSTANCES. THE SECOND ROW SHOWS THE RUNTIME TO FIND A SOLUTION OF THE ORIGINAL INSTANCE. EACH SET OF FIVE ROWS FOLLOWING SHOW THE AVERAGE, MINIMUM, MAXIMUM RUNTIME, WORST CASE (WC) DIFFERENCE IN RUNTIME, AND THE RUNTIME VARIANCE FOR 100 WATERMARKS OF EACH LENGTH.

between the Valiant-Vazirani combinatorial isolation lemma and the watermarking process. We add clauses used by the Valiant-Vazirani lemma in such a way that they correspond to the owner's binary signature which is generated as a random binary stream by the cipher whose key corresponds to the user's secret text message. We validated our technique on a variety of application-derived SAT instances and created SAT instances with specified structure. We demonstrate a close match between the theoretically predicted credibility and fairness and the ones obtained after the use of the standard SAT solvers: WalkSAT and ZChaff.

REFERENCES

[1] G. Qu, J. Wong, and M. Potkonjak, "Fair watermarking techniques," in *Asia and South Pacific Design Automation Conference*, 2000, pp. 55–60.

[2] S. Cook, "The complexity of theorem proving procedures," in *ACM Symposium on Theory of Computing*, 1971, pp. 151–158.

[3] M. R. Garey and D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*. W. H. Freeman, 1979.

[4] T. Larrabee, "Test pattern generation using boolean satisfiability," *Transactions on Computer-Aided Design*, pp. 6–22, 1992.

[5] J. Marques-Silva and K. Sakallah, "Robust search algorithms for test pattern generation," in *Fault-Tolerant Computing Symposium (FTCS)*, 1997, pp. 1–10.

[6] P. R. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Combinational test generation using boolean satisfiability," *IEEE Transactions on Computer-Aided Design*, vol. 15, no. 9, pp. 1167–1176, 1996.

[7] I. Hamzaoglu and J. Patel, "New techniques for deterministic test pattern generation," *Journal of Electronic Testing*, vol. 15, no. 1-2, pp. 63–73, 1998.

[8] C. Chen and S. Gupta, "A satisfiability-based test generator for path delay faults in combinational circuits," in *Design Automation Conference*, 1996, pp. 209–214.

[9] G. Nam, K. Sakallah, and R. Rutenbar, "Satisability based FPGA routing," *International Conference on VLSI Design*, pp. 574–577, 1999.

[10] G.-J. Nam, K. A. Sakallah, and R. A. Rutenbar, "Satisfiability-based layout revisited: Detailed routing of complex FPGAs via search-based boolean SAT," in *International Symposium on Field-Programmable Gate Arrays*, 1999, pp. 167–75.

Instance	50s-10v	100s-10v	100s-20v	1000s-25v	1000s-50v	10s-10000v	10000s-200v	1000s-1000v	25000s-150v	100s-2000v
Orig Solns	50	100	100	1000	1000	10	10000	1000	25000	100
1x										
Ave Solns	0.484	0.48	0.506	0.5008	0.4991	0.54	0.5027	0.5001	0.499496	0.506
Min Solns	0.46	0.48	0.49	0.497	0.492	0.4	0.4992	0.495	0.49712	0.48
Max Solns	0.52	0.48	0.53	0.505	0.504	0.6	0.5076	0.504	0.5018	0.53
WC Solns	0.124	0	0.079	0.0159	0.0240	0.37	0.0167	0.018	0.009	0.099
Variance	0.00096	0	0.0004	4.18E-06	1.23E-05	0.009	7.87E-06	6.77E-06	3.41E-06	0.0002
2x										
Ave Solns	0.212	0.27	0.252	0.2503	0.2512	0.28	0.25131	0.25	0.250336	0.246
Min Solns	0.2	0.27	0.24	0.245	0.247	0.1	0.2451	0.247	0.2482	0.21
Max Solns	0.24	0.27	0.28	0.254	0.255	0.4	0.255	0.254	0.25244	0.28
WC Solns	0.188	0	0.158	0.0359	0.0318	1.071	0.0393	0.028	0.0169	0.284
Variance	0.0003	0	0.0003	1.13E-05	9.51E-06	0.0084	9.25E-06	4.22E-06	2.34E-06	0.0005
3x										
Ave Solns	0.12	0.114	0.127	0.1251	0.1374	0.11	0.1262	0.1252	0.124748	0.113
Min Solns	0.12	0.11	0.12	0.122	0.122	0	0.1244	0.124	0.1234	0.09
Max Solns	0.12	0.13	0.13	0.128	0.252	0.3	0.1316	0.126	0.126	0.15
WC Solns	0	0.175	0.0787	0.0479	0.946	2.727	0.057	0.0159	0.021	0.530
Variance	0	7.11E-05	2.33E-05	4.32E-06	0.0016	0.009	4.17E-06	4.00E-07	7.27E-07	0.0004
4x										
Ave Solns	0.04	0.065	0.06	0.0629	0.0627	0.06	0.0624	0.0679	0.062528	0.066
Min Solns	0.04	0.05	0.06	0.062	0.06	0	0.0593	0.06	0.0618	0.06
Max Solns	0.04	0.08	0.06	0.064	0.064	0.1	0.0652	0.123	0.0634	0.08
WC Solns	0	0.461	0	0.0318	0.0638	1.666	0.0945	0.928	0.0255	0.303
Variance	0	0.00025	0	7.67E-07	1.57E-06	0.0026	3.28E-06	0.0003	3.12E-07	4.89E-05
5x										
Ave Solns	0.036	0.036	0.023	0.0341	0.0372	0.06	0.03103	0.0315	0.031324	0.028
Min Solns	0	0.03	0	0.029	0.03	0	0.0276	0.03	0.03004	0
Max Solns	0.06	0.04	0.03	0.063	0.063	0.2	0.0324	0.032	0.03248	0.04
WC Solns	1.66	0.277	1.304	0.997	0.887	3.333	0.154	0.0635	0.0779	1.428
Variance	0.00096	2.67E-05	0.00015	0.0001	0.0001	0.004	2.38E-06	5.00E-07	5.36E-07	0.0001
10x										
Ave Solns	0	0	0	0.0021	0.0005	0	0.0009	0.0017	0.001008	0
Min Solns	0	0	0	0	0	0	0	0	0.00084	0
Max Solns	0	0	0	0.008	0.002	0	0.0013	0.004	0.0012	0
WC Solns	-	-	-	3.809	4	-	1.44	2.353	0.357	-
Variance	-	-	-	6.77E-06	7.22E-07	-	1.24E-07	1.34E-06	1.10E-08	-

TABLE VII

Evaluation of fairness using the instances with known number of solutions. The first column indicates measured parameter. The next ten columns represent the created SAT instances. Each set of five rows following show the average, minimum, maximum number of solutions found, worst case (WC) difference in number of solutions, and the variance for 100 watermarks of each length.

[11] L. A. Entrena and K.-T. Cheng, "Combinational and sequential logic optimization by redundancy addition and removal," *IEEE Transaction on CAD*, pp. 909–916, 1995.

[12] S. Devadas, "Optimal layout via boolean satisfiability," in *IEEE International Conference on Computer-Aided Design*, 1989, pp. 294–297.

[13] A. Gupta and P. Ashar, "Integrating a boolean satisfiability checker and BDDs for combinational equivalence checking," in *International Conference in VLSI Design*, 1998, pp. 222–225.

[14] W. Kunz and D. Stoffel, *Reasoning in Boolean Networks*. Kluwer Academic Publishers, 1997.

[15] J. Silva and T. Glass, "Combinational equivalence checking using satisability and recursive learning," in *Design Automation and Test in Europe Conference*, 1999, pp. 145–149.

[16] O. Coudert, "On solving binate covering problems," in *Design Automation Conference*, 1996, pp. 197–202.

[17] V. Manquinho and J. Marques-Silva, "On using satisfiability-based pruning techniques in covering algorithms," in *Design Automation and Test in Europe*, 2000, pp. 356–363.

[18] N. Sherwani, *Algorithms for VLSI physical design automation*. Kluwer Academic Publishers, 1993.

[19] P. Barth, "A Davis-Putnam based enumeration algorithm for linear pseudo boolean optimization," Max-Planck-Institut fur Informatik, Tech. Rep., 1995.

[20] V. Manquinho, A. Oliveira, and J. Marques-Silva, "Models and algorithms for computing minimum-size prime implicants," in *International Workshop on Boolean Problems (IWBP)*, 1998, pp. 83–92.

[21] F. A. Aloul, A. Ramani, I. L. Markov, and K. A. Sakallah, "Generic ILP versus specialized 0-1 ILP: an update," in *ICCAD*, 2002, pp. 450–457.

[22] J.P.Warners, "A linear-time transformation of linear inequalities info conjunctive normal form," in *Inf. Proc. Letters*, ser. 2, vol. 68, 1998, pp. 63–68.

[23] H. Xu, R. Rutenbar, and K. Sakallah, "sub-SAT, a formulation for related boolean SATisfiability with approximate routing," in *International Symposium on Physical Design*, 2002, pp. 182–187.

[24] M. Davis, G. Logemann, and D. Loveland, "Machine program for theorem-proving," in *Communications of the ACM*, vol. 5, 1962, pp. 394–397.

[25] J. Marques-Silva and K. Sakallah, "GRASP: a search algorithm for propositional satisfiability," *Transactions on Computers*, vol. 48, no. 5, pp. 506–521, 1999.

[26] H. Zhang, "SATO: An efficient propositional prover," in *Conference on Automated Deduction and LNAI 1249*, 1997, pp. 272–275.

[27] B. Selman, H. Levesque, and D. Mitchell, "A new method for solving hard satisability problems," in *AAAI*, 1992, pp. 440–446.

[28] J. Groote and J. Warners, "The propositional formula checker Heerhugo," The National Research Institute for Mathematics and Computer Science in the Netherlands, Tech. Rep. SEN-R9905, 1999.

[29] M. Sheeran and G. Stalmarck., "A tutorial on Stalmarck's proof pro-

cedure for propositional logic," in *International Conference on Formal Methods in Computer-Aided Design*, 1998, pp. 82–99.

[30] Y. Shang and B. W. Wah, "A discrete lagrangian-based global-search method for solving satisfiability problems," *Journal of Global Optimization*, vol. 12, no. 1, pp. 61–100, 1998.

[31] J. Marques-Silva and K. Sakallah, "Boolean satisfiability in electronic design automation," in *ACM/IEEE Design Automation Conference*, 2000, pp. 675–680.

[32] A. B. Kahng, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Constraint-based watermarking techniques for design IP protection," *IEEE Transactions on CAD*, vol. 20, no. 10, pp. 1236–1252, 2001.

[33] A. Kahng, S. Mantik, I. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Robust IPP watermarking methodologies for physical design," in *Design Automation Conference*, 1998, pp. 782–787.

[34] D. Kirovski, Y.-Y. Hwang, M. Potkonjak, and J. Cong, "Intellectual property protection by watermarking combinational logic synthesis solutions," in *International Conference on Computer-Aided Design*, 1998, pp. 194–198.

[35] G. Qu and M. Potkonjak, "Analysis of watermarking techniques for graph coloring problem," in *International Conference on Computer-Aided Design*, 1998, pp. 190–193.

[36] G. Qu, J. L. Wong, and M. Potkonjak, "Optimization-intensive watermarking techniques for decision problems," in *DAC Design Automation Conference*, 1999, pp. 33–36.

[37] A. Caldwell, H. Choi, A. Kahng, S. Mantik, M. Potkonjak, G. Qu, and J. Wong, "Effective iterative techniques for fingerprinting design IP," in *Design Automation Conference*, 1999, pp. 843–848.

[38] G. Qu and M. Potkonjak, "Fingerprinting intellectual property using constraint-addition," in *Design Automation Conference*, 2000, pp. 587–592.

[39] C. Collberg and C. Thomborson, "Watermarking, tamper-proofing, and obfuscation - tools for software protection," *Transactions on Software Engineering*, vol. 28, no. 2, pp. 735–746, 2002.

[40] D. Kirovski, D. Liu, J. L. Wong, and M. Potkonjak, "Forensic engineering techniques for VLSI CAD tools," in *Design Automation Conference*, 2000, pp. 581–586.

[41] F. Hartung and M. Kutter, "Multimedia watermarking techniques," *Proceedings of IEEE*, vol. 87, no. 7, pp. 1079–1107, 1999.

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

[43] ——, "Applications of toral automorphisms in image watermarking," in *International Conference on Image Processing*, 1996, pp. 237–240.

[44] R. Wolfgang and E. Delp, "A watermark for digital images," in *International Conference on Images Processing*, 1996, pp. 219–222.

[45] R. B. Wolfgang, C. I. Podilchuk, and E. J. Delp, "Perceptual watermarks for digital images and video," *International Conference on Security and Watermarking of Multimedia Contents*, vol. 3657, pp. 40–51, 1999.

[46] I. Cox, J. Killian, T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for images," in *International Conference on Image Processing*, 1996, pp. 243–246.

[47] J. T. Brassil, S. Low, and N. F. Maxemchuk, "Copyright protection for the electronic distribution of text documents," in *Proceedings of the IEEE*, vol. 87, 1999, pp. 1181–1196.

[48] R. Ohbuchi, H. Masuda, and M. Aono, "Watermarking three-dimensional polygonal models," in *ACM International Multimedia Conference*, 1997, pp. 261–272.

[49] ——, "Shape-preserving data embedding algorithm for NURBS curves and surfaces," in *Computer Graphics International*, 1999, pp. 180–187.

[50] P. Su, J. Kuo, C.-C., and H. Wang, "Blind digital watermarking for cartoon and map images," *The International Society for Optical Engineering*, vol. 3657, pp. 296–306, 1999.

[51] J. Lach, W. Mangione-Smith, and M. Potkonjak, "FPGA fingerprinting techniques for protecting intellectual property," in *Proceedings of CICC*, 1998, pp. 299–302.

[52] R. Chapman, T. Durrani, and A. Tarbert, "Watermarking DSP algorithms for system on chip implementation," *International Conference on Electronics, Circuits and Systems*, vol. 1, pp. 377–380, 1999.

[53] R. Chapman and T. Durrani, "IP protection of DSP algorithms for system on chip implementation," *IEEE Transactions on Signal Processing*, vol. 48, no. 3, pp. 854–861, 2000.

[54] I. Torunoglu and E. Charbon, "Watermarking-based copyright protection of sequential functions," in *Custom Integrated Circuits Conference*, 1999, pp. 35–38.

[55] A. Oliveira, "Techniques for the creation of digital watermarks in sequential circuit designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 9, pp. 1101–1117, 2001.

[56] R. Newbould, J. Carothers, J. Rodriguez, and W. Holman, "A hierarchy of physical design watermarking schemes for intellectual property protection of IC designs," in *IEEE International Symposium on Circuits and Systems*, vol. 4, 2002, pp. 862–865.

[57] R. Newbould, D. Irby, J. Carothers, and J. Rodriguez, "Watermarking ICs for IP protection," in *IEE Electronics Letters*, vol. 38, no. 6, 2002, pp. 272–274.

[58] R. Newbould, D. Irby, J. D. Carothers, J. J. Rodriguez, and W. T. Holman, "Mixed signal design watermarking for IP protection," in *Symposium on Mixed Signal Design*, February 2001.

[59] D. Irby, R. Newbould, J. D. Carothers, J. J. Rodriguez, and W. T. Holman, "Placement of watermarking of standard-cell designs," in *Symposium on Mixed Signal Design*, February 2001.

[60] I. Hong and M. Potkonjak, "Behavioral synthesis techniques for intellectual property protection," in *Design Automation Conference*, 1999, pp. 849–854.

[61] A. Oliviera, "Robust techniques for watermarking sequential circuit designs," in *Design Automation Conference*, 1999, pp. 837–842.

[62] V. Alliance, "http://www.vsi.org/."

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

[64] O. Watanabe, "Test instance generation for promised NP search problems," in *Structure in Complexity Theory Conference*, 1994, pp. 205–216.

[65] T. Emden-Weinert, S. Hougardy, and B. Kreuter, "Uniquely colourable graphs and the hardness of colouring graphs of large girth," *Combinatorics Porbability & Computing*, vol. 7, no. 4, pp. 375–386, 1998.

[66] M. Luby and A. Wigderson, "Pairwise independence and derandomization," University of California at Berkeley CSD-95-880, Tech. Rep., 1995.

[67] B. Selman, H. Kautz, and B. Cohen, "Local search strategies for satisfiability testing," in *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challeng*, 1993, pp. 521–532.

[68] L. Zhang and S. Malik, "Conflict driven learning in a quantified boolean satisfiability solver," in *International Conference on Computer Aided Design*, 2002.

[69] R. Bayardo and R. Schrag, "Using CSP look-back techniques to solve exceptionally hard SAT instances," in *Principles and Practice of Constraint Programming*. USENIX, 1996, pp. 46–60.

Jennifer L. Wong received her B.S. degree in Computer Science and Engineering and M.S. in Computer Science from the University of California, Los Angeles in 2000 and 2002. Currently, she is undergoing her Ph.D. studies at the University of California, Los Angeles. Her research interests include intellectual property protection, optimization for embedded systems, and mobility in ad-hoc sensor networks.

Rupak Majumdar Biography text here.

Miodrag Potkonjak received his Ph.D. degree in Electrical Engineering and Computer Science from University of California, Berkeley in 1991. In 1991, he joined C&C Research Laboratories, NEC USA, Princeton, NJ. Since 1995, he has been with Computer Science Department at UCLA. He received the NSF CAREER award, OKAWA foundation award, UCLA TRW SEAS Excellence in Teaching Award and a number of best paper awards. His watermarking-based Intellectual Property Protection research formed a basis for the Virtual Socket Initiative Alliance standard. His research interests include system design, embedded systems, computational security, and intellectual property protection.