Publicly Detectable Techniques for the Protection of Virtual Components

Gang Qu
Electrical & Computer Engineering Department and Institute for Advanced Computer Studies
University of Maryland, College Park, MD 20742 USA

Abstract

Highlighted with the newly released intellectual property (IP) protection white paper by VSI Alliance, the protection of virtual components (VCs) has received a large amount of attention recently. Digital signature is one of the most promising solutions among the known protection mechanisms. However, the trade-off between hard-to-attack and easy-to-detect and the lack of efficient detection schemes are the major obstacles for digital signatures to thrive. In this paper, we propose a new watermarking method which (i) allows the watermark to be public detected without forensic experts, (ii) gives little advantage to attackers for forgery, and (iii) does not lose the strength of protection provided by other watermarking techniques. The basic idea is to make part of the watermark public. We explain the concept of this public-private watermark and discuss the generation and embedding of such marks. We use popular VLSI CAD problems, namely technology mapping, partitioning, graph coloring, FPGA design, and Boolean satisfiability, to demonstrate its easy detectability, high credibility, low design overhead, and robustness. Finally, this technique is compatible with all the known watermarking and fingerprinting techniques.

1 Introduction

The advances in VLSI semiconductor technology and system-on-a-chip design paradigm, coupled with the shrinking time-to-market window, have changed the traditional system design methodology. Design reuse and intellectual property (IP) based design become more and more important. Unlike design from scratch, the real challenge nowadays for system designers is to find IPs and make necessary modification, as little as possible, to meet customer's requirements in a timely fashion.

The Virtual Socket Interface Alliance (VSIA), an international organization that includes representatives from system houses, semiconductor vendors, electronic design automation companies, and IP providers, specifies open standards to facilitate the mix and match of virtual components from multiple sources in order to accelerate system-chip development. According to VSIA, virtual component (VC) is a block that meets the virtual socket interface specification and is used as a component in the design environment[13]. Vendors offer VCs in three forms: soft VC (in the form of behavioral description), firm VC (in the form of structural description), and hard VC (in the form of physical description).

VC trading plays a central role in the design-for-reuse methodology and the potential of infringement is growing fast. However, the global awareness of VC protection remains low. The goals of VC protection are to enable IP providers to protect their VCs against unauthorized use, to detect and to trace the use of VCs. Of the early efforts on VC protection, only detection mechanisms enable designers to do the so-called self-protection. Other approaches require either time, or money, or both[1]. Using the detection methods, designers embed digital signatures or other traceable marks into VCs during the design and implementation phases, such that unauthorized usage can be detected and the source of theft can be traced[1, 2, 5, 7, 9, 10].

1.1 Related Work and Motivation

There have been various proposals for the identification of VC's ownership. In tagging and tracking technique, labels are attached to VCs in the manufacturing phase for the purpose of tracing. Examples include the "physical tagging standard" from VSIA and silicon fingerprinting technology from SiidTech Inc.(*www.siidtech.com/*). Digital watermarking techniques enable VC owners to show their authorship by demonstrating their hidden digital signatures[2, 5, 7] and fingerprinting makes it possible to trace each individual VC by creating a unique copy for each user[1, 9].

Kahng et al.[5] first explain how to embed digitalized signature into VCs as additional design constraints and demonstrated this approach in the context of physical design. These extra constraints are selected in such a way that the VC's value is maintained. The indistinguishability of such constraints from the original design constraints provides the watermark's robustness. Later on, similar ideas have been applied on behavioral level, logic synthesis, FPGA design, and standard cell place and route algorithms[4, 7, 9, 10]. Meanwhile, Charbon[2] proposes the hierarchical watermarking, where signatures are created and hidden at multiple levels of hierarchy to enhance the robustness. Lach et al.[9] utilize certain special structures in FPGA design to embed IP buyer's fingerprint. Caldwell et al.[1] give a generic and efficient approach to create fingerprinted IPs.

Clearly the success of digital signatures relies on the detectability and traceability of the copyright marks. However, a general copy detection process is equivalent to the problems of pattern matching or subgraph isomorphism which are well-known NP-hard. Up to date, three different approaches for copy detection have been reported. Charbon and Torunoglu[3] discuss copy detection under a design environment that involves IPs from multiply sources that requires IP providers to register their IPs in a trusted agent. For several instances (namely scheduling, graph coloring, and gate-level layout), Kahng et al.[6] choose signatures selectively and develop fast comparison schemes to detect such signatures. More recently, Kirovski et al.[8] propose a forensic engineering technique to identify solutions generated by strategically different algorithms.

Efficient detection technique is an essential piece of the protection mechanism and is as important as watermarking techniques. Comparing to watermarking and fingerprinting, we see the research on copy detection lack both in breadth and in depth. Due to the hardness of the detection problem in general, most of the existing watermarking and fingerprinting literature focus on how to make the marks more secure and leave copy detection as an open challenge problem[4, 7]. The trade-off is that, in most cases, the more secure watermarks or fingerprints are, the more difficult to detect

[1] According to the IP protection white paper released recently by VSIA, there are three approaches to the problem of securing a VC: deterrent approach like patents, copyrights, and trade secrets; protection via licensing agreements or encryption; detection mechanism such as physical tagging, digital watermarking and fingerprinting[14].

them, even for the authors.

The lack of detection mechanisms may cause problems for both IP providers and buyers who obtain IPs from other brokers and distributors. On one hand, if IP providers cannot detect their digital signatures, such marks become useless and IP's copyright is lost. On the other hand, dishonest parties may illegally sell the reproduced IPs to innocent buyers at a much lower price, knowing that the end users are unable to tell the real source of the IP. Things become even worse in the latter scenario, since IP buyers usually do not possess the knowledge that IP providers have for copy detection or the required expertise for forensic engineering.

1.2 New Approach and Contributions

In this paper, we propose a new watermarking technique to solve the copy detection problem. The core concept is to divide the watermark into two parts: the public part which is visible to the public, and the private part which is only visible for authorized people. Both public watermark and private watermark are in the form of addition design constraints. Their difference is that public watermark is embedded in designated locations with known method to guarantee public detectability, while private part are embedded the same way as traditional constraint-based watermark. We use cryptographic techniques for data integrity to deter any attempt of removing or modifying the public watermark.

The separation of public watermark and private watermark provides the following advantages:

- It facilitates easy public copy detection. A relatively convincing authorship can be verified by end users without forensic experts and in great extend deters illegal redistribution.
- IP providers can select private watermark as before to obtain the desired level of credibility.
- There is little extra cost to trade for easy detectability. The new technique is compatible with all the existing watermarking/fingerprinting methods.

1.3 Motivational Example

Researchers in UCLA propose a watermarking technique to hide signatures during logic synthesis where the marks are in the form of a set of primary outputs which are not necessary to be primary in the original design[7]. Constraints are introduced to enforce the selected gates to be visible in the final technology mapping solution. Suppose there are 100000 gates in a design out of which 10000 nodes are visible, and 1000 visible nodes are selected based on designer's secret key and the encryption scheme being used. The authors argue that the possibility that others accidentally obtain exactly the same solution is 10^{-1000}. The strength of this watermark relies on the uniqueness of these 1000 nodes. Designer's secrete key is necessary for watermark detection.

Now we illustrate how public detectability can be achieved with the same example using the same watermarking method.

1. select 160 gates, $G_1, G_2, \cdots, G_{160}$, and make them public (the selection of such gates will be discussed later);
2. hash a 4-letter design company symbol (32 bits in ASCII) by one-way hash function such as MD5;
3. append the 128-bit hash result to the 32-bit company symbol to make a 160-bit string: $m_1 m_2 \cdots m_{160}$;
4. for each gate G_i, make it visible if $m_i = 1$ and invisible otherwise. Suppose that half of them (80 gates) are made visible.
5. select 920 more gates other than the 160 public gates based on designer's secret key. Enforce these gates to be visible to embed private watermark.

It is clear that the new scheme chooses 1000 (920+80) gates to be visible, therefore it can achieve the same level of protection as the previous one[2]. Moreover, one can detect the (public) watermark without knowing the designer's secret key as follows:

1. check the visibility of gates $G_1, G_2, \cdots, G_{160}$ in that order. Let $m_i = 1$ if G_i is visible, otherwise let $m_i = 0$.
2. pick the first 32 bits from the 160-bit string $m_1 m_2 \cdots m_{160}$. This is the provider's plain text message in ASCII;
3. hash the selected 32-bit and compare the hash result with the remaining 128 bits. If they are the same, the authorship is established. A mismatch indicates a sign of piracy and further careful moves should be considered.

In the next section, we briefly discuss the IP-based design flow with the public-private watermarking. Then in Section 3, we explain the selection, embedding, and detection of public-private watermark. Besides the above logic synthesis example, we consider several other well-studied problems in the context of VLSI: the Boolean satisfiability (SAT), partitioning, FPGA layout, and graph coloring. The proposed watermarking technique is validated and experimental results are reported before we draw the conclusion.

2 IP-based Design Flow with the Public-Private Watermark

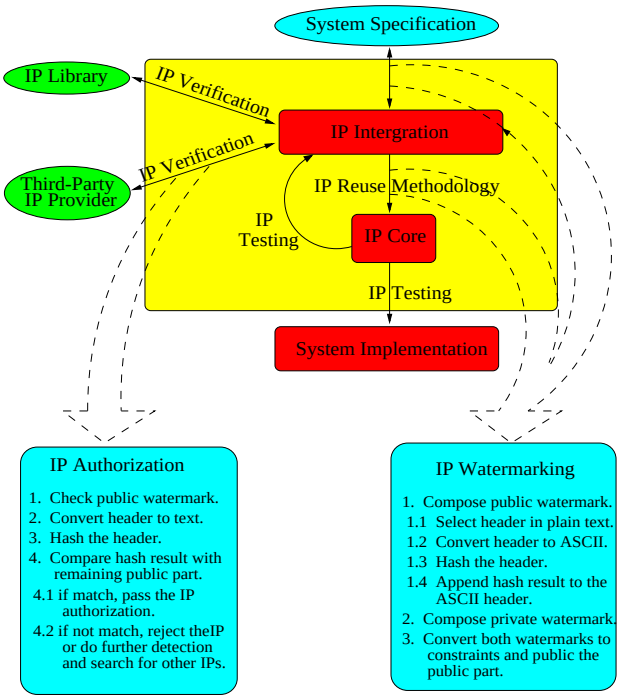


Figure 1: IP-based design flow with the detection and embedding of public-private watermarking.

Figure 1 depicts the global design flow based on IP reuse. With the system specification, the designers will take the necessary IP blocks

[2] In fact, the new watermark is stronger since 80 gates are forced to be invisible.

from the IP library and the third-party IP providers[3]. IP verification process is required for external IPs and IPs from third-party IP providers. An IP authentication is performed during this phase too as shown in the lower left text box in Figure 1. The purpose of this is to avoid getting illegal IPs. Notice that the easy and public detectability of the public watermark reduces the cost for this authentication. Then designers can exploit the reuse methodology to build the core with the public-private watermark embedded. After IP testing is accomplished, this design can be added to the IP library for later use and will have market value.

3 Public-Private Watermarking Technique

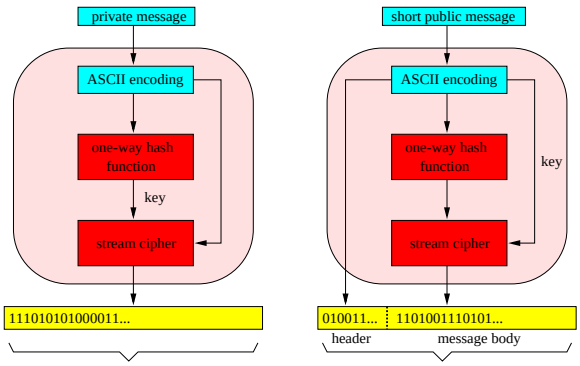
Watermarking and fingerprinting are indirect protection schemes in that they provide a deterrent to infringers by offering the ability to demonstrate ownership of a VC to its originator[14]. The most popular watermarking and fingerprinting techniques are based on the addition of a pseudo-random bitstream as design constraints[1, 4, 5, 7, 10]. Our approach is a direct extension of the same ideas.

3.1 Watermark Selection and Embedding

Our watermark consists of two parts: public and private, which are selected separately. We first explain how to create two bit steams based on public and private message, and then discuss how to encode them to watermarks as extra design constraints.

Selection of Private Watermark Message

The private watermark is the same as the traditional digital watermark discussed in early works[1, 4, 5, 7, 10]. A typical watermark is a cryptographical strong pseudo-random bit stream created by crypto systems using designer's digital signature as the secret key. Figure 2(a) shows the procedure of how to create such bit streams. We hash the plain text message and get a 128-bit or higher hash result, which is used next as the key for a stream cipher to make the plain text message pseudo random.



(a) private watermark message (b) public watermark message

Figure 2: construction of public-private watermark messages.

Selection of Public Watermark Message

The public watermark message is composed of a header and a body. We first pick a short plain text message containing design information such as ownership, project title, and starting date. A good example may be the 4- or 3-letter symbol for the design company. The ASCII code of this short text is put as the header for the public watermark. Then we use a one-way hash function to hash this message. The hash result is put into the stream cipher with the plain text message as key. The output from the stream cipher makes the body of the public watermark message as shown in Figure 2(b).

Watermark Embedding

Once we have both watermark messages, we can use encoding schemes to translate the binary strings to design constraints. Developing such schemes needs to use the characteristics of a given problem and we will discuss it in Section 4 for specific VLSI CAD problems). To ensure the public detectability of the public watermark, we make the followings public: (i) the hash function being used in the construction of public watermark, (ii) the watermarking scheme we use to create the constraints, and (iii) the place where we embed these constraints. We keep the secret key out of the reach of public to make the private watermark secure[4].

As illustrated in Figure 2, we can use the same cryptographic tools to generate both watermark messages and convert them to constraints using the same watermarking scheme. Therefore, comparing to the traditional watermark, there is no extra computational cost or a second pass to embed the public-private watermark. We gain the public detectability at the expense of releasing part of the watermark, which we call public watermark.

Finally, notice that we have not used any special properties of the watermarking scheme, therefore the idea of public-private watermark is compatible with all existing watermarking techniques.

3.2 Watermark Detection and Security

We limit our discussion to the detection of public watermark, the private part can be detected by the existing copy detection techniques with the secret key[3, 6, 8].

Since we have made (i) the hash function, (ii) the watermark scheme, and (iii) the place that hosts the (public) watermark public, one can check for the existence of constraints in the part of the design that carries the public watermark to reveal the entire public watermark message. Next the message header is taken and hashed, if the hash result coincides with the the message body, the public watermark is detected and the authorship is established.

The public watermark can only provide a limited level of confidence on the authorship, further evidence is possible when secret key is available or forensic tools are used to detect the private watermark. The combined public-private watermark can provide the same level of credibility as a traditional constraint-based watermark.

The private watermark is as secure as before, however the public part is visible to everyone and may be vulnerable to attacks. In most known constraint-based watermarking techniques, attackers will have a great amount of advantage if they can detect the watermark. That is not the case in our scheme due to the fact that the message body in the public watermark is the hash of the header. We add security to public watermark by data integrity.

Suppose an adversary follows the same steps and get our public watermark. It is relatively easy for him to remove or modify this mark. It is also trivial for him to compute the hash based on the modified watermark. However, the following two fundmental facts make this attack hard and unrealistic[5]:

- The faked hash will be different from the original in half of the bits statistically. We can make message body long such that this change will be significant.

[3] We refer here IP library as both the internal IPs and external IPs in the public domain where watermarks may not be imposed.

[4] The security of the cryptographic function depends on the secret key, not on which hash function and stream cipher we use to encrypt the message. Also it is the digital signature, which is independent of the watermarking schemes, that carries the proof of authorship.

[5] By *unrealistic*, we mean that the degradation of performance is so large that one will not accept it and the design loses its value.

- Design is an integrated process, it is unlikely one can make one change without altering the behavior of the design. At least some level of local modification is expected.

The message body in public watermark is pseudo random, but the header is not. We make the header relatively short to keep the majority of the public watermark stream "pseudo random". The selection and embedding of private watermark are almost independent of the public watermark[6]. So attackers gain little on how to break the private watermark[7].

4 Validation and Experimental Results

We have explained how to create the public-private watermark which is a pseudo-random bit stream (except the header of public watermark). In this section, we conduct several case studies to validate this approach. More specifically, we discuss where to embed the (public) watermark, how to detect it, what is the impact to system's performance, and how much damage can an attacker do to the (public) watermark.

4.1 Partitioning

Partitioning, which enables the powerful divide and conquer approach, plays a key role in VLSI design. Given a hypergraph $G = (V, E)$ on a set of vertices V and a set of hyperedges E, the partitioning problem is to partition V into disjoint nonempty subsets. The constraints and objective functions for the partitioning vary with the level at which partitioning is performed and different design styles being used. Typical objective functions include minimize interconnections and delay under constraints such as number of nodes in each partition (balance constraint), area of each partition and number of partitions. Two most popular classes of algorithms for this NP-hard problem are group migration algorithms (e.g., Kernighan-Lin and Fiduccia-Mattheyses) and simulated annealing and evolution based algorithms[12].

A k-bit public watermark is hidden in a graph partitioning solution as follows: we select k pairs of vertices and order them randomly; for each pair, we enforce them to be in different subsets to embed a bit 1 and enforce them to be in the same partition to embed a bit 0 by adding proper constraints[8].

Figure 3(a) shows two partitioning solutions: the one separated by the dashed line with arrow heads has letter 'p' as its public watermark, and letter 'O' is hidden in the other solution. To detect these watermark, one only needs to know the above public watermarking scheme and the 8 pairs of vertices holding the watermark as we marked from 0 to 7 in Figure 3(b). The way these 16 vertices are partitioned shows an 8-bit public watermark[9]. For example, the two vertices in pairs 0,1,2,3, and 7 are on the same side of the dashed line with arrow heads, which implies 0's at the corresponding bit positions. The message has bit 1 at the other positions because the other pairs are separated by the line. Consequently, assuming bit 0 is the least significant bit, we get an 8-bit message from this solution: "0111000", which is 'p' in ASCII code.

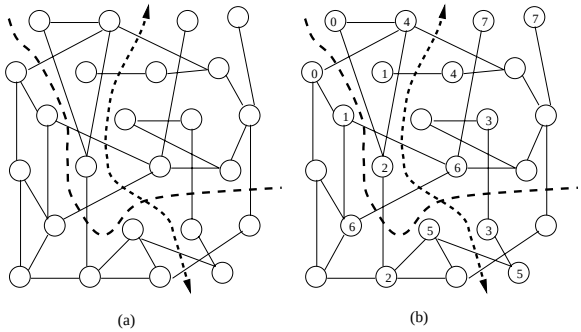


Figure 3: Public-private watermark messages hidden in graph partitioning solutions. The solution separated by the dashed line with arrow heads has message 'p', and the other hides the letter 'O'.

One can easily verify that the other solution hides the bit stream "01001111", i.e. letter 'O'.

4.2 FPGA Layout

Lach et al.[9] propose an FPGA fingerprinting technique that utilize the FPGA design flexibility to put a unique identification mark into the design for each customer. For example, the four tiles in Figure 4, each contains four configurable logic blocks, all implement the same Boolean function $Z = A + B + C \cdot D$. Moreover, they have the same interfaces and thus are interchangeable. The timing of the circuit may vary due to the changes in routing.

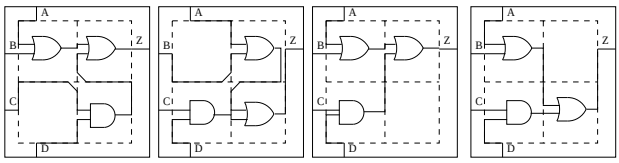


Figure 4: Four instances of the same function with fixed interfaces (redrawn from [9]).

This observation is used to create different design for different customer to trace the use of the design[9]. However, the same property can be used to embed public watermark. We first label the four CLBs as 00, 01, 10, 11 clockwise from the upper left to the lower left. To hide 2 bits from the public watermark message, one can choose one of these four implementations, with the unused CLB has the same label as the given 2 bits. For example, from left to right, the four design in Figure 4 have "11", "00", "10", and "01" as the embedded message respectively. With a few of such tiles, one can find sufficient space for public watermark messages.

Forgery is a problem for this approach. Given a FPGA layout with the public-private watermark embedded, an attacker can go to the tiles where public watermark is hidden and obtain the bit stream easily. Then he can change the message header at his wish, use one-way hash function and stream cipher on his new message header to forge a message. Next, he can do the necessary modifications in these tiles to replace the original public watermark by his faked message. This will be a successful attack unless private watermark is revealed. However, this is the same problem as what FPGA watermarking and fingerprinting techniques are facing. The solution lies on the difficulty of reverse engineering and the fact that most FPGA vendors will not reveal the specification of their configuration streams[5, 9].

[6] We say *almost independent* because the selection and embedding of private watermark are restricted by the existence of public watermark. For example, the addition of private watermark should not change the public watermark.

[7] What the attacker gains are: the place that public watermark sits, which is small comparing to the entire design place; and the (public) watermarking scheme being used which may not be the same as the scheme that private watermark is embedded.

[8] The type of constraints depends on the objective function of the partition. For example, if we want to minimize the interconnection cost in a weighted graph, two vertices will go to different partitions if we change the weight of the edge between them to $-\infty$ (when they are connected) or add an extra edge of weight $-\infty$; similarly, they will stay together if the edge between them has a ∞ weight.

[9] Due to the small size of the example, we assume that we have only the public watermark message header here. The encrypted message body can be embedded and detected in the same way.

4.3 Boolean Satisfiability

The Boolean satisfiability problem (SAT) seeks to decide, for a given formula, whether there is a truth assignment for its variables that makes the formula true. SAT appears in many contexts in the field of VLSI CAD, such as automatic pattern generation, logic verification, timing analysis, delay fault testing and channel routing. We necessarily assume that the SAT instance to be protected is satisfiable and that there is a large enough solution space to accommodate the watermark.

Given a formula $\mathcal{F}$ on a set of boolean variables V, the simplest watermarking technique for public detectability is to hide the public watermark behind a known subset of variables $\{v_1, v_2, \cdots, v_k\}$. Suppose the public watermark message is $m_k \cdots m_2 m_1$, we embed it by forcing $v_i = m_i$ in the solution. This can be done by adding to the formula $\mathcal{F}$ single-literal clause v_i (if $m_i = 1$) or v_i' (if $m_i = 0$)[10].

We pick four 4-letter messages A, B, C, and D. We use MD5[11] (*ftp://ftp.sunet.se/pub3/vendor/sco/skunkware/uw7/fileutil/md5/src*) as the one-way hash function to obtain four 128-bit messages H(A), H(B), H(C), and H(D). Next we use RC4 (*ftp://ftp.ox.ac.uk/pub/crypto/misc/rc4.tar.gz*) to encrypt these messages using their ASCII codes as the encryption keys. The resulting pseudo-random bit streams are appended to the ASCII codes of the corresponding plain text to form the four public watermark messages as illustrated in Figure 2(b).

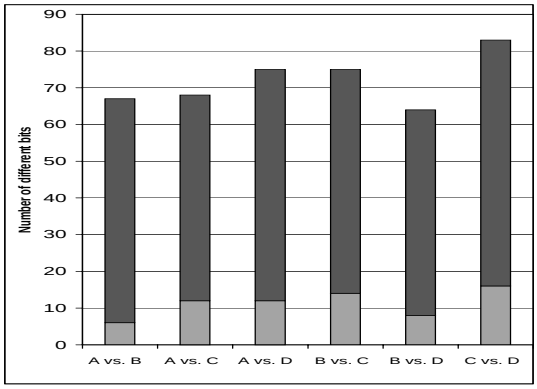


Figure 5: Hamming distance among the four public watermark message. The bottom half comes from the message header(plain text part), and the top half comes from the message body(results of RC4).

Figure 5 shows pairwisely the Hamming distance among these four public watermark message. A and B, B and D are relatively close because each pair has one letter in common accidentally[11].

We now embed these public watermark messages to DIMACS SAT benchmarks, where the instances are generated from the problem of inferring the logic in an 8-input, 1-output "blackbox" (*http://dimacs.rutgers.edu/*). We first select 32 variables for the message header, then choose 128 (or 64 for instances of small size, e.g., with less than 600 variables) more variables for the message body. We then assign values to these variables based on the public watermark and solve for the assignment of the rest variables to get the original solution.

[10] A single-literal clause imposes a very strong constraint to the formula. Statistically it will cut the entire solution space by one half. Therefore we may use a short public watermark message, in particular for instances with not so many variables. However, the credibility can always be enhanced by adding private watermark using other techniques, such as those proposed in [5].

[11] The ASCII codes for messages A, B, C, and D are: "01010011 01000111 01001001 00100000", "01000011 01000100 01001110 00100000", "01010011 01001110 01010000 01010011", and "01001101 01000101 01001110 01010100".

With the given solution (and variables that carry the public watermark), an adversary retrieve the public message header, modify it and compute the new message body. He then embed this forged message and resolve the problem. Our goal is to show that there is little correlation between the original solution and adversary's new solution, i.e., attacker has little advantage from the original solution or it is equally difficult to obtain a solution.

		4 bits in header		8 bits in header		16 bits in header		24 bits in header	
$\mathcal{F}$	N	body	sol.	body	sol.	body	sol.	body	sol.
ii8b1	336	31.2	148	32.8	150	31.8	168	32.6	170
ii8b2	576	33.6	260	30.6	258	32.4	265	32.0	272
ii8b3	816	62.2	363	64.0	376	67.4	358	61.6	387
ii8b4	1068	65.8	489	66.2	472	63.4	492	62.6	513
Ave. Dist. (%)		40.2%	-	43.5%	-	50.5%	-	56.2%	-
Ave. Dist. (%)		-	44.9%	-	44.9%	-	46.5%	-	48.3%

Table 1: Average number of different bits in public message body ("body"), average distance (rounded to integer) from the original solution ("sol.") when 4-bit, 8-bit, 16-bit, and 32-bit forgery is conducted to the public message header on SAT benchmarks.

Table 1 shows our experimental results, where messages A, B, C, D are embedded to the four SAT instances respectively. The second column gives the number of variables N in these instances. We consider the adversary changes randomly 4 bits, 8 bits, 16 bits, and 24 bits in the 32-bit message header. We repeat each trial 5 times, the columns labeled "body" show the average number of bits changed in the faked message body from the original. We solve each instance with this faked message (both header and body) embedded and calculate the Hamming distance between the new solution and the original solution. The average distances (rounded to the nearest integer) are reported in columns with label "sol.".

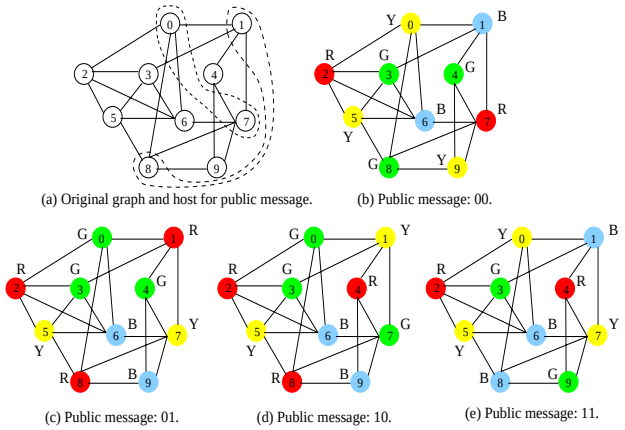
The last two rows report these average distances percentage-wise. The first is the distance in public domain, which is very close to 50% if we exclude the mandatory header part. It is independent of the number of bits being modified in the header and shows the robustness of our cryptographic tools in generating pseudo-random bit streams. The last row shows that the new solutions are not close to the original solution. (When we solve the original instances for multiple solution, their average distance is also about 45%.) Therefore, we can conclude that the new solutions are independent of the given solution, which means that once the public watermark has been modified, the adversary loses almost all the advantage from the given solution. This is further verified by the fact that the run time difference for resolving the problem and solving from scratch is so small (within 5%) that we consider they are the same.

4.4 Graph Coloring

The NP-hard graph vertex coloring optimization seeks to color a given graph with as few colors as possible, such that no two adjacent vertices receive the same color. We propose the following public-private watermarking technique for graph coloring problem and use it to demonstrate our approach's impact to the quality of the solution:

> For a given graph, we select pairs of vertices that are not connected directly by an edge. We hide one bit of information behind each pair as follows: adding one edge between the two vertices and thus making them to be colored by different colors to embed 1; collapsing this pair and thus forcing them to receive the same color to embed 0.

Consider Figure 6, two pairs of unconnected vertices, nodes 0 and 7, and nodes 1 and 8, are selected as shown in the dashed circles in 6(a). The rest of Figure 6 shows four different coloring schemes with a 2-bit public watermark message embedded. To detect such



(a) Original graph and host for public message. (b) Public message: 00.

(c) Public message: 01. (d) Public message: 10. (e) Public message: 11.

Figure 6: Four GC solutions with different public watermarks added to the same graph.

watermark, one can simply check the colors received by nodes 0, 1, 7, and 8. For example, in Figure 6(c), nodes 0 and 7 are colored by G(reen) and Y(ellow) respectively, which means the first bit (the most significant bit) is 0. Similarly, the observation that nodes 1 and 8 are both colored by R(ed) tells us the second bit of the message is 1. Therefore, we detect a public message "01".

To evaluate the trade-off between protection and solution degradation (in the case of graph coloring, the number of extra colors), we first color the original graph, then color the watermarked graph and comparing the average number of colors required. We consider two classes of real life graphs (the *fpsol2* and *inithx* instances from *http://mat.gsia.cmu.edu/COLOR/instances.html*) and the DIMACS on-line challenge graph (available at *http://dimacs.rutgers.edu/*).

Table 2 shows the number of vertices in each graph, the optimal solutions (the DSJC1000 problem is still open. The number in the table is the average of 10 trials with 85-color solutions occur several times), and the overhead introduced by public watermark messages of various length. For each instance, we create ten 32-bit and ten 64-bit public watermark messages randomly. We add the message to the graph and color the modified graph. The average number of colors and the best solution we find are reported. One can easily see that the proposed approach causes little overhead for real life instances, but loses best solutions for the randomized DSJC1000 graph. The reason is that there exist localities in real life graph of which we can take advantage of. However, such localities do not exist or are very difficult to find in random graphs.

original instance			32-bit message		64-bit message	
	vert.	opt.	overhead	best	overhead	best
fpsol2.i.1	496	65	0.2	65	0.7	65
fpsol2.i.2	451	30	0.1	30	0.5	30
fpsol2.i.3	425	30	0.1	30	0.5	30
inithx.i.1	864	54	0.0	54	0.2	54
inithx.i.2	645	31	0.9	31	1.8	32
inithx.i.3	621	31	1.1	31	1.9	32
DSJC1000	1000	85.8	0.5	86	2.0	87

Table 2: Embedding public watermark to real life graph and randomized graph.

5 Conclusion

We propose the first watermarking technique that facilitates easy and public detection. We achieve this by allowing part of the watermark to be public. We use cryptographic techniques, in particular techniques for data integrity, to protect the public watermark from forgery. Using the traditional constraint-based watermark as private part, this public-private watermarking scheme is capable of providing public detectability with no degradation on the watermark's strength. We explain the basic approach and develop specific techniques for various classes of VLSI CAD problems. The new approach is compatible with all the existing watermarking techniques. With the help from organizations pushing for design standards, for example VSIA, this method has the potential of solving eventually the IP protection problem.

References

[1] A.E. Caldwell, H. Choi, A.B. Kahng, S. Mantik, M. Potkonjak, G. Qu, and J.L. Wong. "Effective Iterative Techniques for Fingerprinting Design IP", *36th ACM/IEEE Design Automation Conference Proceedings*, pp. 843-848, June 1999.

[2] E. Charbon. "Hierarchical Watermarking in IC Design", *IEEE 1998 Custom Integrated Circuits Conference*, pp. 295-298, May 1998.

[3] E. Charbon and I. Torunoglu. "Copyright Protection of Designs Based on Multi Source IPs", *IEEE/ACM International Conference on Computer Aided Design*, pp. 591-595, November 1999.

[4] I. Hong and M. Potkonjak. "Behavioral Synthesis Techniques for Intellectual Property Protection", *36th ACM/IEEE Design Automation Conference Proceedings*, pp. 849-854, June 1999.

[5] A.B. Kahng, J. Lach, W.H. Magione-Smith, S. Mantik, I.L. Markov, M. Potkonjak, P. Tucker, H. Wang and G. Wolfe. "Watermarking Techniques for Intellectual Property Protection", *35th ACM/IEEE Design Automation Conference Proceedings*, pp. 776-781, June 1998.

[6] A.B. Kahng, D. Kirovski, S. Mantik, M. Potkonjak, and J.L. Wong. "Copy Detection for Intellectual Property Protection of VLSI Design", *IEEE/ACM International Conference on Computer Aided Design*, pp. 600-604, November 1999.

[7] D. Kirovski, Y. Hwang, M. Potkonjak, and J. Cong. "Intellectual Property Protection by Watermarking Combinational Logic Synthesis Solutions", *IEEE/ACM International Conference on Computer Aided Design*, pp. 194-198, November 1998.

[8] D. Kirovski, D. Liu, J.L. Wong, and M. Potkonjak. "Forensic Engineering Techniques for VLSI CAD Tools", *37th ACM/IEEE Design Automation Conference Proceedings*, pp. 581-586, June 2000.

[9] J. Lach, W.H. Mangione-Smith, and M. Potkonjak. "FPGA Fingerprinting Techniques for Protecting Intellectual Property", *Proceedings of the IEEE 1998 Custom Integrated Circuits Conference*, pp. 299-302, May 1998.

[10] A.L. Oliveira. "Robust Techniques for Watermarking Sequential Circuit Designs", *36th ACM/IEEE Design Automation Conference Proceedings*, pp. 837-842, June 1999.

[11] R.L. Rivest. "The MD5 Message-Digest Algorithm", *http://www.cis.ohio-state.edu/htbin/rfc/rfc1321.html*, April 1992.

[12] N.A. Sherwani. "Algorithms for Vlsi Physical Design Automation", 3rd edition, Kluwer Academic Publishers, June 1999.

[13] Virtual Socket Interface Alliance. "Architecture Document Version 1.0", March 1997.

[14] Virtual Socket Interface Alliance. "Intellectual Property Protection White Paper: Schemes, Alternatives and Discussion Version 1.0", September 2000.