Information Hiding through Reversible Computing

Ali J. Zubairy and Dr. Najmi Ghani Haider
SZABIST
Karachi, Pakistan

Abstract: *Information hiding has become an interesting topic that is receiving more and more attention. It is used in communicating the most confidential information such as use in back channel diplomacy. Recently, many hiding techniques have been proposed to directly conceal secret information. Images are the most common type used for information hiding. This paper focuses on a new technique, which is based on the ongoing research topic of reversible computing.*
This paper presents a design of a logically equivalent reversible machine that can hide information while running forward and can reveal information when run in the reverse direction.

Keywords: *Information hiding, steganography, reversible computing*

1. INTRODUCTION

As evidenced from history, in difficult political times, limitations are placed on the individual's freedom of opinion and expression. Nowadays, there is heavy dependence on electronic media and automated systems in the collection and dissemination of information. The transfer of information using these communication channels is open to the threat of information leakage.

Clearly, it is highly desirable to protect the communicating parties' freedom to communicate securely in the presence of eavesdroppers, who seek to control the communication channels to detect and even penalize the communication deemed by them to be non-permissible. This is a well-understood setup in information-security studied in the context of steganography.

1.1. Cryptography and Steganography

Cryptography refers to the technique of information hiding by encryption, i.e. the sender uses an encryption key to scramble the message, which is then transmitted through a normal communication channel to the intended recipient. The reconstruction of the original, unencrypted message is possible only if the receiver has the appropriate decryption key.

In the steganography approach, the secret message is embedded in another message. Using this technology, even the fact that a secret is being transmitted has to be secret. The main purpose of steganography is to hide the occurrence of such communication. While most methods in use today are invisible to the observer's senses, mathematical analysis may reveal statistical discrepancies in the steganographic medium. These discrepancies expose the fact that hidden communication is taking place, making it detectable.

2. REVERSIBLE COMPUTING

Reversible computing is motivated by the Von Neumann Landauer (VNL) principle [1], [2], a theorem of modern physics stating that ordinary irreversible logic operations (which destructively overwrite previous outputs) incur a fundamental minimum energy cost. Such operations typically dissipate roughly the logic signal energy, itself irreducible due to thermal noise. This fact threatens to end improvements in practical computer performance within the next few decades. This particular limit could only possibly be avoidable through reversible computing. It "de-computes" unwanted bits, rather than obliviously erasing them. This can avoid entropy generation, enabling the signal energy to be preserved for later re-use, rather than being dissipated.

The digital computer usually perceived as logically irreversible as it performs operations that discard the unwanted byproduct information, and utilize the same memory space for some other instruction because its transaction value lacks single valued inverse and hence, are considered to be logically irreversible. Every digital computer can be simulated by general purpose computing machines (i.e. Turing Machine), hence, they are also considered to be logically irreversible.

Landauer in his paper [1] raised the debatable question of whether logical irreversibility is an unavoidable feature of modern digital computers, arguing that it is, and demonstrating the physical importance of this question by showing that whenever a physical computer discards information about its previous state it must generate a corresponding amount of entropy.

A reversible digital logic operation can be defined as any operation that performs an invertible (one-to-one) transformation of the device's local digital state space, or at least, of that subset of states that are actually used in a design.

Every machine can be made logically reversible by saving the information it would otherwise discard, e.g. if the machine saves information regarding all its operation to an extra space with sufficient detail such that at a later stage the previous stage can be uniquely identified, then every machine can be made reversible. The major

disadvantage of reversibility is that it requires extra amount of space for saving historical data.

2.1 Research in Reversible Computing

The greatest motivation for the study of hardware and software technologies aimed for implementing reversible computing is that the machines have a thermodynamic loophole; that implies that they might become quite useful as CPUs, become more and more powerful. Ordinary electronic circuits waste some energy every time they make a decision, but reversible computers do not. This wasted energy exits the integrated circuit chip as heat, which is why the newest and fastest CPUs come with heat sink attached on top of the CPU. Some of the fastest machines are cooled by liquid coolants, which can absorb even more heat. The buildup of waste heat is a serious problem and if not removed, will result in CPU malfunction and eventual destruction.

Reversible computing offers what seems to be the only potential way to improve the energy efficiency of computers beyond the fundamental Von Neumann-Landauer limit [1], [2] of kTln2 energy dissipated per irreversible bit operation, where, k is the Boltzmann's constant of $1.38 \times 10{-}23$ J/K, and T is the temperature of the environment into which unwanted entropy will be absorbed.

Reversible computing has been researched over several years. The motivation for much of the research has been the first seminal paper on the topic, which was published by Charles H. Bennett of IBM research [3] in 1973. In his paper, Bennett based on the Landuer machine model that has energy dissipation per step of about kT; demonstrated that any machine computation can be done with a reversible Turing machine [5]. He has created a few basic examples of reversible Turing machine with well-defined commands for moving the read/write head of the tape and changing the state of the machine. The transaction rules for these machines often look quite similar to the Fredkin gate [4]. There is just as much information coming out at each step as going into the gate. This is balanced correctly so the position that leads to another position can always be inferred and the machine can be reversed.

The result showed that anything that can be done with a computer could be done with a reversible computer. All that is necessary is to find a way to save the information from each step, so that it can be effectively run in reverse.

The original work on reversible computing was done by Ed Fredkin [4], in which he proposed a type of logic gate that would not expend energy. Normal gates that take the AND of two bits are not reversible. For instance, if x AND y is 1, then both x AND y can be recovered because both must have been 1. But, if x AND y is 0, then nothing conclusive can be deduced about either x or y; either x or y may have been 1. This makes it impossible to run such a normal gate in reverse. On the other hand, the Fredkin gate does not discard the information so it can be reversed.

Today, the field has a substantial body of academic literature [1], [2], [3], [5], [6]. A wide variety of reversible device concepts, logic gates, electronic circuits, processor architectures, programming languages, and application algorithms have been designed and analyzed by physicists, electrical engineers, and computer scientists.

This field of research awaits the detailed development of a high quality, cost-effective, nearly reversible logic device technology, one that includes, highly energy-efficient clocking and synchronization mechanisms. This sort of solid engineering progress will be needed before the large body of theoretical research on reversible computing can find practical application in enabling real computer technology to circumvent the various near-term barriers to its energy efficiency, including the Von Neumann-Landauer bound.

2.2 Concept of Using Reversible Computing for Information Hiding

Many scientists have been studying reversible computing for quite some time. There is a significant advancement in this technology with some real implementation, but still much work remains to be done for practical implementation of this technology. The same concept can be applied for the purpose of information hiding. Assume, information can be hidden by just doing any computation, and revealed by running it in reverse.

To design such a system requires the creation of a universal, reversible Turing machine format. A standard program running on any computer would be able to read in a Turing machine and run it forward or backward. To send a message, load the data in and run the machine until it halts. The result would be the output and the waste temporary data that must be retained in order to run the machine in reverse.

At the receiver's end, this info would be loaded into the same universal Turing machine and run backwards to recover the data. One problem with this scenario is that any attacker could also possess the same universal, reversible Turing machine and could intercept the message and reverse it recovering the message. For the technique to be useful, some data must be kept secret from the attacker. This could be sent separately, independent of the message.

If such a machine can be designed, then any computation can be used to hide information in the final outcome. This can hide as much information as needed depending upon the calculation, e.g. to hide large amounts of data, a network based interactive game program could be used.

3. REVERSIBLE TURING MACHINE

If a Turing machine can be designed in a reversible manner, then every possible machine is a candidate for being converted into a vehicle for hiding information. Some of them are more interesting than others, e.g. computer programs that are quite simple and can only respond with either success or failure. This is just one bit of information, and it seems unlikely that anyone will be able to hide much of anything in just that one bit. On the other hand, programs that produce complex worlds of game like "Need 4 Speed" spit out billions of bits, there is ample room in the noise to hide the intended message; e.g. the secret information could be encoded in the car track data. The two parties communicating the message could join the internet group game of "Need for Speed" and the messages could come across the internet disguised as instructions for where to draw the car on the screen. The second version of "Need for Speed" could extract this.

This paper suggests a simple reversible Turing machine based on the work of Charles Bennett [6] demonstrating how to convert a simple non-reversible Turing machine into a reversible machine by tweaking it; so that, there is only one possible state that could lead to another, this makes it possible to rewind the behavior.

3.1 Design of a Machine

This section focuses on the requirement to build a reversible Turing equivalent machine that could perform basic computation functions, but still be reversible. There are numerous problems that need to be confronted in the design of this machine.

It will get much of its performance by emulating the Fredkin gate, which merely swaps information instead of destroying it.

"Wasted bits" are by-products (such as the use of temporary variables to hold intermediate values) produced when a computation is performed. These waste bits need to be sent to the recipient to enable computation in reverse.

There are various ways to send these bits to the recipient, the most simple but least secure is to send the bits through the same channel. A more secure way is to send it through another channel encoded in the least significant bits of an image or in an audio chat session, or through some other covert channel.

3.1.1 Minimizing Extra State

All extra bits are transported through a separate channel at the end. These should be kept to a minimum. For this reason, all strings should be constant, and then the final state of all strings would need to be communicated to the recipient, and this would become too much overhead.

3.1.2 Arithmetic

Arithmetic is generally not reversible, e.g. 3+2 is not reversible, but it is, if one component of the equation is retained. Thus, adding the contents of register A1 and register A2 and storing the result in register A1 is reversible. The content of A2 can be subtracted from A1 to recover the original value of A2.

For most part, addition, subtraction, multiplication and division are reversible; if they are expressed in this format. The only problem is multiplication by zero. This would not be allowable.

3.1.3 Structure of Memory

What form will be the memory take? Obviously, ordinary computers allow a programmer to access and move blocks of memory at a time. This is not feasible because it would require that too many extra waste bits be retained. Block move of data is not reversible, but swapping of information is.

For that reason, there is simply an array of registers. Each one holds one number that can be rounded off in some case. The final state of the register will be shipped as extra state to the recipient, so any programmer should aim to use them sparingly. Unfortunately, the rules of reversibility can make this difficult.

3.1.4 Conditional Statement

Most conditional statements that choose between branches of a program can be reversed, but sometimes this may not be so. Consider the case where "if x is less than 100 then add 1 to x; otherwise, add 1 to p" which path is to be taken when running in reverse and x is 100? Should 1 be subtracted from p or not? Either case is valid.

One possibility is to execute each branch storing results in temporary variables. When the conditional statement is encountered, the proper choices can be swapped into place.

The solution is to disallow the program from changing the contents of those variables that were used to choose a branch. This would rule out many standard-programming idioms. The following is one way to work around the problem:

```
if x < 100  then {
        swap x,k;
        k = k + 1}
else {
        p = p + 1;
        swap x, k+1; }
swap x,k;
```

3.1.5 Loops

Loops may be very handy devices for a programmer, but can often lead to ambiguities when a program is run in reverse. A straight forward example of this is the while loop used to test for the last element in a string in C programs, i.e. a counter moves down the string until the termination character, a /0 character is found. It may be easy to move backward up the string, but it is impossible to know where to stop.

The problems with a loop can be eliminated if the structure is better defined. It is not enough to simply give a test condition for the end of the loop. The dependent variable of the loop, its initial setting, and the test condition has to be specified. When the loop is reversed, it will run the contents of the loop until the dependent variable reaches its initial setting.

This structure is often not strong enough. Consider the loop:

```
i = 1;
j = I;
while ( i < 2 ) do {
        j = j + 0.01;
        i = floor ( j ); }
```

The floor (x) function finds the largest integer less than or equal to x. This function will execute 100 times before it stops. If it is executed in reverse, then it will only go through the loop twice before *i* is set to its initial value, 1. It is clear that *i* is the defining variable for the loop, but it is clear that *j* has an important role.

There are two ways to resolve this problem, the first is to warn programmers and hope that they will notice the mistake before they use the code to send an important message. This leaves some flexibility in their hands.

Another solution is to constrain the nature of loops further. There is no reason why loops cannot be restricted to “for loops” that apply a particular function to every element in a list. Both are quite useful and easy to reverse without conflicts.

4. STRENGTH ANALYSIS OF THIS MACHINE

The security of this machine lies in the program code. A steganalyst may be able to reveal the message by decoding, i.e. reverse engineering the program code.

The major security drawback in this technique is that, anyone can design the same reversible Turing machine. They could intercept the message and recover the message. In order to avoid this problem, some part of data must be kept separate, i.e. it must be transferred through some separate covert channel, which should also be encrypted.

The second solution might be to keep the structure of the program secret and let it act as a key. Only the output and extra waste bits will be transmitted to the recipient. The adversary, who is trying to intercept the message, cannot read it without the copy of the program that created it. So, the strength of this scheme is in the ability to keep adversaries from getting access to it.

How difficult would it be for an attacker to form the structure of the program with these outputs? Obviously, there are some programs that are quite easy to crack; e.g. a program that copies the data to be hidden and spits it out, would be easy to figure out. The output maintains the structure of the document. A more complicated program would be harder to crack. Eventually, there must be some thing of complexity that is really extremely difficult to crack. The major issue is; is there some measure such that a program exceeding it could be considered completely safe to use for the purpose?

REFERENCES

[1] R. Landauer & R.W. Keyes, IBM J. *Res Develop. 14, 152 (1970),* Investigate a specific model computer whose energy dissipation per step is about kT.

[2] J. Von Neumann. Theory of Self-Reproducing Automata, Univ. of Illinois Press, 1966.

[3] C. H. Bennett, "The Thermodynamics of Computation -- A Review," *International Journal of Theoretical Physics*, Vol. 21, no. 12, pp. 905-940, 1982.

[4] Edward Fredkin and Tommaso Toffoli “Conservative Logic”, *International Journal of Theoretical Physics* 21 (1982), 219-253.

[5] Charles Bennett and Rolf Landauer. The fundamental physical limits of computation. *Scientific American*, pages 48-56, July 1985.

[6] C. H. Bennett, "Logical reversibility of computation," *IBM Journal of Research and Development*, Vol. 17, no. 6, pp. 525-532, 1973.