Software Pirates: a Criminal Investigation

Hayson Tse, K.P. Chow, Pierre Lai
The University of Hong Kong,
Pokfulam Road, Hong Kong
{hkstse, chow, kylai}@cs.hku.hk

***Abstract*—Computer program infringing materials are difficult to identify. There are common techniques to disguise the origin of copied codes. In order to decide on a legal basis whether a substantial part of copyright work has been taken, it is necessary to consider both the quality and quantity of the part taken. Various researches have carried out in relation to authorship identification and plagiarism identification. In a criminal case in Hong Kong, we used a common software to compare files contents and folders between a copyright work and the infringing copy instead of complex and technical metrics. We conclude that the source codes of the defendant started from the source codes of his previous employer using simple and easy to understand measurements. Though the magistrate was satisfied beyond reasonable doubt of the defendant's guilt, the evidence in the case did not enable a scientific calculation in respect of the likelihood that a computer program may look like a derivative of another program by chance.**

I. Introduction

Copying other's computer program or sharing them with others has an innocent origin. It began in the 1980s when computer technology was still in its infancy. Computer geeks at universities wrote computer programs. They formed computer clubs to share computer programs to help each other. Those clubs soon became opportunities for sharing other computer programs. Such opportunities were enlarged by the invention of Bulletin Board Systems. In those days, computer programs were either used by hobbyists or large corporations. Later, computer programs were copied not for the sole purpose of helping others in the development of computer programs [6], e.g. for any purpose of trade or business.

In the education sector, copying other's computer program is known as computer program plagiarism. In 2001, Bull et al. [3] carried out a survey on 293 academics. Fifty percent of the academics felt that there had been an increase in plagiarism by students. Those students copied other's computer programs and used them as if they were their own inventions for their assignments.

In the business sector, employees access employers' computer program codes on employers' hard drives. They copy them onto external hard drives and use those codes in connection with their own private business. Various methods were proposed to detect computer program piracy.

Computer program privacy has now become a serious and widespread problem. In 2010, for every US dollar spent on legitimate software, an additional 63 cents worth of unlicensed software also made its way into the market. In the same year, computer program privacy caused a loss of US $59 billion worldwide [1].

Computer programs infringing materials are difficult to identify. It is not uncommon for courts to spend a considerable amount of time, effort, and resources to determine whether a computer program is reproduced from another computer program without the licence of the owner. Researches have been carried out regarding authorship identification and plagiarism identification.

In a criminal case of copyright infringement in Hong Kong, a commonly available software which quickly and easily compared files and folders was used to examine whether the defendant's computer program was substantially a copy of the victim's program. The victim was the former employee of the defendant, a programmer. Though the defendant was convicted, the evidence in the case did not enable a scientific calculation in respect of the likelihood that a computer program may look like a derivative of another program by chance.

In this paper, we first examine in sections II and III to see whether object codes of a computer program is or is not protected by copyright laws in Hong Kong, and the modes of copying. In section IV, we outline common techniques used to disguise the origin of copied codes and we give a survey of ways to detect copying. In section V, we describe our findings in the criminal case mentioned above. We conclude in section VI.

II. Protection of source codes and object codes

We consider the first issue whether the source codes and object codes of a computer program are protected or not protected by copyright laws in Hong Kong. This leads to an examination of the nature of computer programs.

A. Source codes and object codes

In order to design a computer program, a programmer writes in ordinary language with formula, flow charts and diagrams, of which represented a set of instructions to a computer. The programmer then writes out the program in details by "*source code*" in one of the recognized computer languages. Source codes are "high level language". They are so called because they are not far removed from ordinary language.

A computer does not understand ordinary language and cannot read the source code. Therefore, it is necessary to process them into binary codes by another program called a compiler. The words and algebraic symbols of the "high level language" then become binary numbers. This binary program code is called "*object code*" or "machine code".

The program in "object codes" in the first instance are often stored on a magnetic disk, tape, compact disc, or a read only memory silicon chip. These devices are installed in the computer. When electrical power is applied to them, a sequence of electrical impulse is generated. Those impulses cause the computer to take action according to the program in "object codes". The descriptions of the electrical impulses are displayed on a visual display unit of the computer, or print out on paper [23].

B. *Hong Kong protection*

In Hong Kong, copyright is a property right which subsists in, amongst other things, literary works. Literary works include a computer program [19].

There is no statutory definition provided for the phrase "computer program". The Concise Oxford Dictionary defines the phrase as "a series of coded instructions to control the operation of a computer". "Object codes" are a series of coded instructions to control the operation of a computer. Therefore, "object codes" are a computer program.

Despite this definition, it has been argued that object codes were not literary works [23], a material form perceived by the senses of a human being . The reasons are:

1) Electrical impulses in a silicon chip, a magnet disk or tape, or on a device cannot be perceived by sight, touch or hearing.
2) The electrical pulses were not intended to convey any message to a human being.
3) The re-arrangement of electrons in a programmed Read-Only-Memory is not visible to the human eye too.
4) The electrical impulses also do not represent words, letters, figures or symbols as a literary work.
5) The electrical impulses do not communicate the letters or figures by which an object program may be represented.

On the other hand, source code are not far removed from human languages. Source code conveys messages to a human being as a literary work.

Those arguments ignore two important points. First, the adaptation of a computer program includes a conversion of a version of a computer program from a computer code into a different computer code. The making of an adaptation of a literary work is an act restricted and protected by the copyright legislation in Hong Kong [18].

Second, as a member of World Trade Organization, Hong Kong must give effect to the Trade-Related Aspects of Intellectual Property Rights which protects both source and object code [35].

Also, Hong Kong gives such effect by applying a fair, large and liberal interpretation [16] to all its legislations.

III. Modes of Copying

A. *Hong Kong*

In Hong Kong, copying means a reproduction of *a whole or any substantial part* of a copyright work [17]. Copying includes:

1) The exact or literal reproduction of all or part of a computer program;
2) Reproduction of the program in a re-written form, perhaps in a different language; or
3) Reproduction on a higher level, for example the reproduction of the structure of the program.

The scope is wide because copyright law protects the relevant skill and labour of the programmer. Infringement occurs if there has been an appropriation of a part of the work on which a substantial part of the programmer's skill and labour was expended [11].

In deciding whether a substantial part of copyright work has been taken, it is necessary to consider both the quality and quantity of the part taken. It is also necessary to consider all relevant factors before any conclusion can be drawn. Those factors include [15]:

1) Whether the claimant's work constitutes or does not constitute an original copyright work by reason of the knowledge, skill and labour employed in the production of the claimant's work; and
2) Whether or not the defendant's work has been or has not been produced by the substantial use of those features of the claimant's work.

In view of the above, the copying of commonplace routines is not a copyright infringement because the writing of the commonplace routines involved no great skill or labour.

What amounts to "substantial" varies from context to context. In summary [15]:

1) If an idea, function or concept is *sufficiently worked out and expressed in computer codes* and is the *result of independent skill and labour*, that expression of the idea is protected.
2) Under certain circumstances, there will inevitably be similarities in two *independently written* pieces of coding. This does not mean there is no copyright in such similar codes. Copyright will subsist if it was the product of substantial skill and labour. The circumstances may provide a reason for the similarities.
3) Sometimes, a piece of coding is in the "public domain" or is a standard piece of coding in which no one claims copyright. If someone has combined it with other coding using skill and labour, copyright subsists in the piece of "public domain" coding in combination with the other codings.

B. *Some cases*

We summarised four cases to demonstrate the need for computer applications to be used to detect software piracy.

1) First case: Nowadays, computer users are used to graphical user interface (GUI) and other features, e.g. windows and icons, of at least 2 well known operating systems. Before those features became part of the computing landscape, there have been disputes between them regarding whether or not certain features originated from the other.

There was a time when DOS operating system was a widely used application software. It ran on the Intel processor. Xerox first developed GUI during the 1970's. It was not put into the market. In 1979, Xerox demonstrated its GUI to Apple which was developing new interface for Lisa and Macintosh. At the same time, Intel was also developing new chips which also ran GUI. In late 1983, Microsoft announced developments of Windows platform, version 1.0 of which was put into the market in 1985. In 1985, Apple granted Microsoft a license to used the windows and icons in the development of version 1.0. In exchange, Microsoft agreed to develop software for the Macintosh platform.

Microsoft released Windows version 2.03. On March 17, 1988, Apple filed a law suit against Microsoft and its sub-licensee Hewlett-Packard [12]. The main argument was that both had infringed Apple's copyrights in the presentation and control of on-screen information on the ground that Microsoft's GUI was substantially similar to that of Apple's.

On the basis of the response of an ordinary reasonable person and whether or not the 2 works were virtually identical, the US District Court for the Northern District of California held that Apple could not get patent-like protection for the idea of a graphical user interface, or the idea of a desktop metaphor. This was because all the elements identified as similar by Apple fell into one of the limiting categories, i.e. they were either not subject to broad protection or not copyrightable at all. The only other basis for protection left to Apple was to compare the compilation and arrangement of these elements. Apple's only remaining argument was that the environments were virtually identical. Since this was not a sound argument, Apple chose not to make it. Therefore, when the Court of Appeals applied the same standard, Apple necessarily lost.

Those 2 tests adopted by the trial court were ambiguous in nature [29]. This was not the last duel between Apple and Microsoft.

2) Second case: An ex-programmer of Computer Associates went to work for Altai and wrote a software Oscar by literally copying around 30% of the codes of Computer Associates. In 1989, after Altai found out that they were using copyrighted codes, they employed another programmer to rewrite Oscar without using the copied codes, marketed it and offered free upgrade to users of the old version. Computer Associates filed a law suit against Altai [30]. One of the issues for the trial court to decide was whether or not the second version of Oscar, which did not derive from literal copying, was an infringing copy of the work of Computer Associates. Nonliteral elements include the organization, structures and dynamic sequences of a "running" program.

In deciding the issue, the trial court traced the steps which a programmer took to create the program, except in reverse order. Once the levels of abstraction are laid out, the trial court examine each level to separate idea from expression and decided whether or not the expression has been copied.

The recreation of a programmer's path in the reverse order is undoubtedly difficult and should be done by someone well-versed in programming [4].

The trial District Court held that the second program did not infringe the copyright. On appeal, the Court of Appeal found that the District Court was correct.

3) Third case: The amount of work to be examined can be demonstrated in the law suit filed by IBCOS [9]. IBCOS developed a program called ADS which was largely written by an employee, Mr Poole. The software managed the financing of agricultural equipment. The software comprised of 335 program files, 171 record layout files and 46 screen layout files. Mr Poole later left IBCOS and began working for Barclay's. Mr Pooles leaving agreement with IBCOS provided that all software that had been developed was IBCOS's property.

While working for IBCOS Mr Poole produced an improved version of the ADS program called "Unicorn". When Barclays began marketing "Unicorn", IBCOS brought proceedings for infringement of the copyright that it allegedly owned in the source code of the ADS program.

As a result of the integration of files that came together for software to execute, the trial judge concluded that copyright subsisted on more than one conceptual level in the package of IBCOS. Copyright subsisted in the individual program files that make up a software package, as well as the entire software package. Where a computer program - whether a file or the package as a whole - has been altered sufficiently, a new copyright arises in the amended version of the software.

The Court also held that copying was not restricted to "literal" and can be "non-literal". Copying of a detailed structure where it was a substantial part of a program could infringe copyright in the program. The Unicorn program infringed copyright in the ADS program because there was literal copying of source code, overall similarity and even the replication of mistakes!

The trial judge in the case also noted that: (a) The unimportant parts of a computer program like comments and documentation may betray a conclusion that the source code has been copied. These include common spelling mistakes in the code, comments, variables, procedure calls or function calls; (b) If the capitalization of the words are similar, then they are also relevant to demonstrating copying; (c) comments may include hyphens or asterisks in either side of words which emphasise the structure in the software; (d) unusual similarities also go to the assertion that the code has been copied; (e) redundant code in the allegedly infringing copied code may be identical to the original software; (f) names of program files or any other part of the respective programs may be the same; (g) changes to files that would be identical but for a change that has been made by a global replacement.

4) Fourth case: The fourth case concerns the software implementing an airline booking system principally employed by low-cost airlines who employ 'ticketless' booking. The claimant's system is called OpenRes. The first defendant ('easyJet') is a low-cost airline. The second defendant ('BulletProof') is a software developer, who is responsible for writing the code of the allegedly infringing system, which is called eRes, in consultation with easyJet's IT department. easyJet wanted a new system that was substantially indistinguishable from the OpenRes system. The commands alleged to have been copied by easyJet amounted to some 44% of the OpenRes commands.

The trial judge described the case as being "factually and technically complex" and "There is a great amount of technical detail in the case. Some of the issues cannot be properly approached without some understanding of the technical issues."

[10]

The Court held in relation to these elements single words were not protectable as copyright works. Complex commands and the collection of commands as a whole were not protectable because they amounted to a claim for copyright in a computer language, which is precluded by the European Union Computer Programs Directive. VT100 screen displays were not protectable because they amounted to "ideas which underlie the program's interface". The GUI screen display and icons were protected as distinct artistic works. The "business logic" of the flight reservation program was not protectable as to allow otherwise would be an unjustifiable extension of copyright protection and circumvention of the Directive.

IV. Hide and Seek

A. Common ways to hide copying

Whale [37] listed thirteen common techniques used to disguise the origin of copied codes. The techniques are "changing comments, changing formatting, changing identifiers, changing the order of operands in expressions, changing data types, replacing expressions by equivalents, adding redundant statements, changing the order of time-independent statements, changing the structure of iteration statements, changing the structure of selection statements, replacing procedure calls by the procedure body, introducing non-structured statements, combining original and copied program fragments". Arwin et al. [2] listed an additional technique: the translation of source code from one language to another, or inter-lingual plagiarism. For example, source code written in C may be copied across to an implementation in Java.

Further, an infringer may apply a decompilation process to a computer program and recompile its output in a way that generates a binary with identical functionality but with seemingly different codes. In this case it might be far more difficult to prove that the source code had actually been "stolen" [7].

B. Authorship identification

In digital forensics, Bayesian Network models have been developed to analyze various cybercrime scientifically, such as Internet piracy [25] [26]. Software forensic has been described as the science of software authorship identification [32]. Various methods have been suggested.

Computer programs are generally written in source code. From a linguistic perspective, the source codes are in some respects like a form of human language. In that regard, programmers develop a style and approach that is identifiable [20].

Krsul [24] argued that programmers were humans. Humans were creatures of habit. Habits tended to persist. Krsul identified a large range of metrics that can be used to help determine the author of a program. He concluded that within a closed environment, and for a specific set of programmers, it was possible to identify a particular programmer and the probability of finding two programmers that share exactly those same characteristics should be small .

Other methods include identifying authorship by byte-level N-grams [14], [13], and the use of software metrics such as layout, style and structure.

C. Plagiarism identification

According to El-Wahed, et al., computer program plagiarism detection has been in existence for over twenty years [8].

Plagiarism detection and authorship identification, software forensic, are different. Krsul [24] gave an example of a program "X" that is a plagiarized version by programmer "A" of an original work done by programmer "B". After programmer "A" has copied the original program, he makes stylistic changes. Specifically, old comments are removed and new comments were added. Indentation and placement of brackets are changed to match the style of programmer "A". Variables are renamed. The order of functions is altered and "for" loops are changed to "while" loops. Plagiarism detection detects the similarly between those two programs. Authorship analysis does not. For the purpose of authorship identification, these two programs have distinct authors.

Many software tools have been developed for detecting source-code plagiarism, the most popular being Plague, YAP3, and JPlag [3].

1) Traditional ways of detection of copying: The traditional method used to decide whether a computer program copies or does not copy another program is to compare the source code of one against the other [5].

Prechelt et al. [31] identified two main methods used in automated plagiarism detection: feature comparison and structure comparison. In feature comparison, the similarity of two computer programs are measured by the similarity of various software metrics, e.g the average number of characters per line and the number of comment lines. Jones divided such metrics into three profiles: physical profile, Halstead profile and Composite profile [22].

Physical profile characterizes a computer program by its physical attributes, such as the number of lines, words and characters. Halstead profile characterizes a computer program by its token types and frequencies. Closeness of two computer programs is the Euclidean distance between their profiles. Another way to detect infringing computer programs is to find code clone pairs between the two programs. One of the code clone detection tool is called CCFinder. By using that tool, Tamada et al. found that "sys/net/zlib.c" of FreeBSD and "drivers/net/zlib.c" of Linux are almost identical [34]. There are other applications available such as "UltraCompare" and "Beyond Compare".

These current plagiarism detection tools appear sufficient for academic use, like finding copied programs in programming classes. First, they are short for fighting against serious plagiarists [28]. Second, proof of duplication is not sufficient. El-Wahed, et al. [8] identified the main issue which was whether or not the code was actually copied or whether or not there were other reasons for the similarities, such as using third-party source code, code generation tools, commonly used identifier names, common algorithms or common author. Third, a defendant may argue that his computer program is too different to infringe the copyright in the plaintiff's copyright work. His arguments rely on one or both of two reasons. First, he may be saying that he did not copy or has only copied to an insignificant extent. Secondly, he may take the point that what he copied is not relevant for copyright law purposes because the resemblance between the two programs is too insignificant or is too general to take into account. An example is the way of expressing an idea. If there is only one effective way of expressing an idea, that expression was insufficiently original, and was not therefore protectable [27] under copyright.

However, it is not always the case that the source code of the infringing copy of a copyright work are available for comparison with the source code of the copyrighted work. Sometimes, it is the object codes which are available.

2) Requirements for copy-detection schemes: Schleimer et al. [33] proposed that a copy-detection algorithm should have three properties. They are:

1) Whitespace insensitivity
2) Noise suppression
3) Position independence

Wang et al. [36] identified five requirements. They are abilities to:

1) Cope with semantics-preservation obfuscation techniques;
2) Detect theft of small components of a program;
3) Handle detection for large scale commercial or open source software theft;
4) Detect binary executables;
5) Be independent of platforms, operating systems and program languages.

Wang et al. classified current detection schemes into 4 classes. They are:

1) Static source code based birthmark;
2) Static executable code based birthmark;
3) Dynamic Whole Program Path based birthmark;
4) Dynamic Application Programming Interface based birthmark.

3) Detecting copied computer source code by examining computer object code: Zeidman [38] patented a set of methods and systems for detecting copied computer code. The detection may be performed by comparing source code of a first program to object code of a second program. It may also be performed by comparing object code of a first program to object code of a second program.

Zeidman examined a number of plagiarism detection programs currently available and concluded that comparison by creation of a detailed block diagram of the control flow or data flow of the executable program. Other than those plagiarism detection programs, detection may be performed by comparison of detailed block diagram of the control flow or data flow of the executable program. Zeidman identified three drawbacks, which are the creation of extremely complex diagram, time consuming comparison and difficult to automate.

Zeidman's method comprises of building one or more source code data structures from a program source code. The structures have entries correspond to components of a program represented by program strings or identifiers. From a program object code file, object data structures are built. Those structures also have entries corresponding to text sequence of the object code file. Similar entries are obtained by a comparison of the 2 sets of entries of the data structures. A correlation score is calculated from the similar entries. The magnitude of the score indicates copying.

4) Other means of proof: There are other ways which indicate the hand of a copycat.

One of them is the presence of similar redundancies, mistakes and idiosyncrasies in coding, as opposed to standard routines. This is because computer code evolves during initial development, testing and upgrading. It is common to find that there is residual and redundant code left within a computer program.

Some programmers deliberately include "smoking guns" into their codes. Those codes have no function at all but lie there to prove copying. In the worst case of computer program piracy, there exists exact reproduction of the whole copyright work by simple duplication. Then, proving identity of the infringing copy is a mechanical task by comparing the source code.

Each program has inherent characteristics. Birthmarking relies on such characteristics to show that one program is a copy of another. Grover [21] defined birthmarking as those characteristics which appeared in a program by chance. Such characteristics could be used in program identification. Such characteristics are inherent to the code.

V. A CASE STUDY

In 2005, a programmer left his employer, an accounting software company, G, and started his business, an accounting software company, T. The programmer wrote various programs in the course of his employment with G. After he left, he started his own business and sold a computer program purportedly designed by him. G alleged that T was selling infringing copies of programs of G.

G confirmed that the copyright works were developed using Delphi. T did not dispute that the purported infringing copy was also developed using Delphi.

A. *Criminal Investigation*

In the past, software copying cases were handled through civil litigations in Hong Kong. In the current case, the software company G filed a complaint to the law enforcement agency in Hong Kong. The agency then started a criminal investigation process and the investigator conducted the investigation with the following steps:

1) Preliminary investigation: the purpose of the preliminary investigation is to confirm if there are reasonable cause to suspect that there was an offence;
2) Arrest and seizure: if the result from preliminary investigation is positive, the agency will then plan for an operation to arrest the suspect and collect all necessary evidence;
3) Forensics analysis: forensic analysis will be performed on the collected evidence and an examination report will be produced to assist the public prosecutor;

4) Prosecution: if the public prosecutor is satisfied that there is a reasonable prospect of obtaining a conviction, i.e. evidence to make out the case and evidence to rebut any defence which might be available and which might be raised, and the public interest requires a prosecution, the public prosecutor then puts forward the charge(s) to the suspect and presents the case to the court;
5) Defence: the suspect may present his/her explanation to all charges and evidence produced by the public prosecutor.

B. Preliminary Investigation using Decompilation

When the investigator received the complaint from the software company G, the evidence available were executables of the copyright work of G, executables of (infringing) works of T and source codes of the copyright work of G (provided by the plaintiff). Since only executables from G and T were available, most techniques described in Section IV were not applicable. Instead, investigator used simple and easy to understand approach. Therefore, in the preliminary assessment, the executables of G and T were decompiled for comparison. Whether the case could be established would depend on the findings of the preliminary assessment.

This simple and easy approach was feasible because Delphi executables can be decompiled. After decompilation, original names of files or forms can be recovered. Codes are represented as assembly language. We listed out some of the examples of recovered symbols after decompilation:

```
unit main;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics,
   Controls, Forms, Dialogs, StdCtrls
type
  TMainForm=class(TForm)
    Panel1: TPanel;
     MainMenu1: TMainMenu;
     System1: TMenuItem;
    Help1: TMenuItem;
    About1: TMenuItem;
    Contents1: TMenuItem;
```

Examples of assembly codes recovered are:

```
procedure TMainForm.Exit1Click(Sender : TObject);
begin
(*
005B3E98 55 push ebp
005B3E99 8BEC mov ebp, esp
005B3E9B 6A00 push $00
005B3E9D 6A00 push $00
005B3E9F 53 push ebx
005B3EA0 8BD8 mov ebx, eax
```

C. Preliminary Findings and Expert Opinion

After the decompilation, we observed certain features. We summarized the number of ".pas" files and number of lines of codes in Table I:

G's files	Files (.pas)	Lines	T's files	Files (.pas)	Lines
foo1.exe	15	20,656	foo5.exe	16	26,754
foo2.exe	170	767,265	foo6.exe	314	1,438,387
foo3.exe	139	548,625			
foo4.exe	129	496,758			
Total	453	183,304		330	1,465,141

Table I: No. of .pas file and no. of lines

	G	T
File names exist in both	209	209
File names exist in G	16	
File names exist in T		105
Total	225	314

Table II: Recovered file names

Some file names are identical between G's software and T's software. They are summarized below in Table II:

At the stage, there are two competing set of evidence. Evidence in support of copying were:

1) All .pas filenames in G /start.exe existed in T /tom.exe;
2) 209 amongst 225 .pas file names in G's executables existed in T's executable.

On the other hand, evidence negating copying were:

1) Sizes of executables in both systems were significantly different;
2) Number of lines in recovered assembly codes and source codes (with inline assembly) in both systems were significantly different.

We have to carry out further examination. The evidence of further examination were summarized in Table III, Table IV and Table V:

	No. of source files	No. of source lines
Copyright owner G	288	191,440
Company T	321	219,077

Table III: Size of the 2 software systems

T source code directory	No. of files	G source code directory	No. of files	No. of identical file names
\	1	\	1	1
foo7	5	foo8	5	5
foo9	79	foo10	48	46
foo11	177	foo12	114	110
foo13	6	foo14	6	6
foo15	53	foo3/foo2/foo4	48	32
Total	321		222	200

Table IV: Structure and file names comparison

Similarity of 2 files with identical file names from T's source codes and G's source codes	No. of files
Identical file	23
No. of different lines between 2 files <= 20	37

Table V: Identical or very similar files

On the basis of the above findings, the preliminary opinion was that T's source codes had used G's source codes as a starting point for T's development and a certain amount of work had been done in the T's source codes to provide additional functionalities.

The forensics expert gave an neutral and unbiased opinion based on the evidence collected during the preliminary investigation. It is the responsibility of the public prosecutor to decide whether there was reasonable grounds for the agency to plan an operation to arrest the suspect and to collect further evidence.

D. The Arrest and seizure

Based on the preliminary findings, a case was established. A search warrant was obtained to carry out a search at T's premises. An examination was then carried out on the hard disks found in T's premises. The following evidence were found:

1) Source codes existed in the seized hard disks;
2) The source codes might be used to generate different versions of T's software system.

Later, 16 sets of compact discs were also seized during the operation.

E. Beyond Reasonable Doubt

The main difference between a criminal case and a civil litigation is the public prosecutor is required to prove beyond reasonable doubt that the criminal act was performed by the suspect. In this case, the prosecutor was required to prove that the suspect had made infringing copies of the copyright work for sale which violated the copyright laws in Hong Kong. During the operation, 16 sets of compact discs were collected, which were the copyright work for sale. On the other hand, the infringing copies were made out of the source code which was reproduced from copyright work. The making and reproduction violated the copyright laws. Therefore, the 16 CDs contained T's software system were submitted for forensics examination. The purpose was to find out:

1) Is there any source code that can reproduce T's software system in the 16 compact discs?
2) If there is such set of source code, what are the similarities between the T's source code and the G's source code?
3) Is there any evidence to indicate that T's source code was copied from G's source code?

During the examination, the investigator was unable to rebuild the software from the source code found on the hard disks. It means the investigator was unable to prove that the copyright work for sale in the CDs were produced from the source code, and the production chain was broken. The investigator then decided to use other evidence to prove the offence and the following circumstantial evidence were found from the source codes in the hard disks. They are set out below:

1) Development directories were found in the seized hard drive and different versions were stored in different directories;
2) The executables in one of the development directory match the executables of 1 set of the 16 compact discs;
3) Each executable has the following file information:
    a) Description;
    b) Company;
    c) File Version.
4) The file information associated with executables in different CDs were consistent (not identical), and also consistent with the file information in the development directory.

In view of the fact that the findings needed to be presented to the Court at a later stage, the investigator decided to use simple and easy to understand techniques to perform the comparison instead of the complex methods which might confuse the Court. It is always the objective of the defence lawyer to create confusion at the court so that the magistrate is unable to decide whether there is an offence or not.

Based on the above findings, the investigator performed a more detail comparison in the following manner:

1) Name analysis;
    a) File name comparison;
    b) Function and procedure name comparison;
    c) Database comparison.
2) Source code comparison by Line by line comparison;
3) Search for "core functions" as identified by the copyright owner, e.g. IncOrDecStockLocQty and CheckJnlNo.

The findings are set out in Table VI, Table VII and Table VIII.

	No. of distinct names in G	No. of distinct names in T	No. of matching names	No. of names exist in G only	No. of names exist in T only
Function	121	113	109	12	14
Procedure	4,696	6,897	3,943	753	2,954
Total	4,817	7,010	4,052	765	2,958

Table VI: Comparing function and procedure names

The approach adopted in source code comparison are:

1) File comparison tool Beyond Compare 2.5 was used;
2) Comparison was file-based, i.e. files with identical files names from G's source code and T's source codes were compared.

There are two investigation highlights. They are summarized in Table IX and Table X:

	No. of table names	No. of matching names	No. of names exist in G only	No. of names exist in T only	No. of
Table Name	91	122	91	0	31

Table VII: Comparing file names of database scripts

	No. of lines names	No. of matching lines	No. of names exist in G only	No. of names exist in T only	No. of
Lines	2,884	3,881	2,660	224	1,221

Table VIII: Comparing no. of lines in database scripts

F. Other evidence

In the hard disks of T, we also found:

1) Copyright Notice of G;
2) Dead program statements and commented program statements of G;
3) Dead files of G.

On the basis of all the above findings, and the nexus between G and the defendant, we came to the conclusion that the source codes of T's software were started from the source codes of G's software.

G. Prosecutor

The public prosecutor put forward the report by the forensics expert. The expert appeared at Court to explain the findings in the report. Because the standard of proof is "beyond reasonable doubt", the expert needs to ensure that the magistrate understand all the findings in the report. Any confusion will be to the benefit of the defence.

H. Defence

The defence tendered at trial were that both programs were written by the defendant (G's programmer who left and set up T) and they were therefore similar. Further, some codes had a common source, i.e. they were sourced from the Internet, reference books and samples. It was also argued that the 2 programs for similar functions would have similar code patterns.

I. Verdict

The magistrate heard evidence from prosecution and defence experts. At the end, the magistrate found the prosecution had proved its allegation beyond all reasonable doubt and convicted the defendant, i.e. the defendant's program is an infringing copy of G's program.

	Matching lines	Total lines (Matching files) G	Total lines (Matching files) T	Total lines (All files) G	Total lines (All files) T
No. of lines	10,987	123,504	162,550	140,483	219,063

Table IX: Lines in files

Matching lines	Total lines (Matching files) G	Total lines (Matching files) T	Total lines (All files) G	Total lines (All files) T
10,987	123,504	162,550	140,483	219,063
	88.94%			
			78.19%	
		67.58%		
				50.14%

Table X: Percentage of Matching Lines

VI. Conclusion

This paper has summarily examined the copyright laws of Hong Kong which protect both the source code and object code of a computer program. We also surveyed various researches regarding detection of copying, though various techniques were used to disguise the origin of copied codes. The basis of those methods is the comparison of 2 sets of codes to decide whether one is an exact or literal reproduction of another or a reproduction of a substantial part of another. Establishing whether a program is or is not a derivative of another can be a difficult and subjective task. It depends on the judgment of the individual expert and the methodology used by the expert. The model, the technique and the metrics used depend greatly on the purpose of the analysis and on the information available

In a criminal case in Hong Kong, we simply used a tool to compare files, folders and directories of the original copyright work and the infringing copy of copyright work. We discovered characteristics which proved that the defendant reproduced the copyright work to become his work. Though the defendant was convicted, the evidence in this case did not enable a scientific calculation to evaluate the likelihood that a computer program may look like a derivative of another program by chance.

Further work is to be carried out to design a model for quantitative and qualitative measurements made on computer program source code and object code automatically extracted by analysis tools and calculated by an expert.

References

[1] Business Software Alliance. Eigth Annual BSA and IDC Global Software 2010 Piracy Study. Technical report, Business Software Alliance and International Data Corporation, 2010.

[2] Christian Arwin and S.M.M. Tahaghoghi. Plagiarism Detection across Programing Languages. In *ACSC 06: Proceedings of the 29th Australasian Computer Science Conference, Hobart, Australia*, pages 277 – 286, 2006.

[3] J. Bull, C. Colins, E. Coughlin, and D. Sharp. Technical Review of Plagiarism Detection Software Report. Technical report, JISC, 2000.

[4] John H. Butler. Pragmatism in Software Copyright: Computer Associates v. Altai. *Harvard Journal of Law and Technology*, 6:183, 1992.

[5] Hong Kong Special Administration Region Court of First Instance. *Palm Computing Inc. v Echolink Design Ltd. and Kessel Electronics (H.K.) Limited. HCA 11787/1999 & HCA 13420/1999*, page 1, 2003.

[6] Paul Craig. *Software Piracy Exposed*. Syngress, 2005.

[7] Eldad Eilam. *Reversing: Secrets of Reverse Engineering*. Wiley Publishing Incorporation, 2005.

[8] Samer Abd El-Wahed, Ahmed Elfatatry, and Mohamed S. Abougabal. A New Look at Software Plagiarism Investigation and Copyright Infringement. In *ITI 5th International Conference on Information and Communications Technology*, pages 315 – 318, 2007.

[9] England and Wales High Court of Justice. IBCOS Computers Ltd. v. Barclays Mercantile Highland Finance Ltd. *FSR*, page 275, 1994.

[10] England and Wales High Court of Justice. SAS Institute Inc. v. World Programming Limited. *EWHC (Ch)*, page 1829, 2010.

[11] English High Court of Justice Chancery Division. *Cantor Fitzgerald v. Tradition (U.K.) Limited. RPC*, page 95, 2000.

[12] United States District Court for the Northern District of California. Apple Computer, Inc. v. Microsoft Corporation and Hewlett-Packard Company. *821 F. Supp. 616*, 1993.

[13] G. Frantzeskou, S. Gritzalis, and S. MacDonell. Source Code Authorship Analysis for Supporting the Cybercrime Investigation Process. In *1st International Conference on E-Business and Telecommunication Networks. Setúbal, Portugal, INSTICC Press*, 2004.

[14] Georgia Frantzeskou, Efstathios Stamatatos, Stefanos Gritzalis, Carole E. Chaski, and Blake Stephen Howald. Identifying Authorship by Byte-Level N-Grams: The Source Code Author Profile (SCAP) Method. *International Journal of Digital Evidence*, 6(1), 2007.

[15] Kevin Garnett, Gillian Davies, and Gwilym Harbottle. *Copinger an Skone James on Copyright*. Sweet and Maxwell, 2005.

[16] Government of the Hong Kong Special Administration Region. Section 19 of the Interpretation and General Clauses Ordinance, Chapter 1. *Laws of Hong Kong*, 1997.

[17] Government of the Hong Kong Special Administration Region. Section 22 of the of the Copyright Ordinance, Chapter 528. *Laws of Hong Kong*, 2007.

[18] Government of the Hong Kong Special Administration Region. Section 29 of the Copyright Ordinance, Chapter 528. *Laws of Hong Kong*, 2007.

[19] Government of the Hong Kong Special Administration Region. Sections 2 and 4(1) of the Copyright Ordinance, Chapter 528. *Laws of Hong Kong*, 2007.

[20] Andrew Gray, Philip Sallis, and Stephen Macdonell. Software Forensics: Extending Authorship Analysis Techniques to Computer Programs. In *Proceedings of the 3rd Biannual Conference of the International Association of Forensic Linguists (IAFL), Durham NC, USA*, pages 1–8, 1997.

[21] Derrick Grover. *The Protection of Computer Software - Its Technology and Applications*. Cambridge University Press, 1989.

[22] Edward L. Jones. Metrics based Plagiarism Monitoring. In *Proceedings of the sixth annual CCSC northeastern conference on he Journal of Computing in Small Colleges, Middlebury, Vermont, United States*, pages 253 – 261, 2001.

[23] Hong Kong Court of Appeal. HKSAR v. Chan Tak Tim. *Hong Kong Law Report Digest*, 3:112, 2004.

[24] I. Krsul. AuthorshipAnalysis: Identifying the Author of a Program, Technical Report CSD-TR-94-030. Technical report, Department of Computer Science, Purdue Univesity, 1994.

[25] Michael Y. K. Kwan, Kam-Pui Chow, Frank Y. W. Law, and Pierre K. Y. Lai. Reasoning About Evidence Using Bayesian Networks. In *IFIP Int. Conf. Digital Forensics*, pages 275–289, 2008.

[26] Michael Y. K. Kwan, Richard E. Overill, Kam-Pui Chow, Hayson Tse, Frank Y. W. Law, and Pierre K. Y. Lai. Sensitivity Analysis of Bayesian Networks Used in Forensic Investigations. In *IFIP Int. Conf. Digital Forensics*, pages 231–243, 2011.

[27] Honourable Sir Hugh Laddie, Peter Prescott, Mary Vitoria, Adrian Speck, and Lindsay Lane. *The Modern Law of Copyright and Design*. Butterworths, 3rd edition, 2000.

[28] Chao Liu, Chen Chen, Jiawei Han, and Philip S. Yu. PLAG: Detection of Software Plagiarism by Program Dependence Graph Analysis. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 872 – 881, 2006.

[29] Joseph Myers. Apple v. microsoft: Virtual identity in the gui wars. *Rich. J. L. & Tech.*, 1:5, 1995.

[30] United States Court of Appeals Second Circuit. Computer Associates International, Inc., v. Altai, Inc. *61 F.3d 6*, 1995.

[31] Lutz Prechelt, Guido Malpohl, and Michael Philippsen. Finding Plagiarisms among a Set of Programs with JPlag. *J. UCS*, 8(11):1016, 2002.

[32] Philip Sallis, Asbjorn Aakjaer, and Stephen MacDonel. Software Forensics, Old Methods for a New Science. In *Proceedings of Software Engineering: Education & Practice (SE:E & P' 96), Dunedin, New Zealand*, pages

481 – 485. IEEE CComputer Society Press, 1996.

[33] Saul Schleimer, Daniel S. Wilkerson, and Alex Aiken. Winnowing: Local Algorithms for Document Fingerprinting. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 76 – 85, New York, NY, USA, 2003. ACM.

[34] Haruaki Tamada, Masahide Nakamura, Akito Monden, and Ken-Ichi Matsumoto. Java Birthmarks Detecting the Software Theft. *IEICE - Trans. Inf. Syst.*, E88 - D(9):2148–2158, September 2005.

[35] World Trade Organisation. Articles 1 and 10. *Trade Related Aspects of Intellectual Property Rights*, 2001.

[36] Xinran Wang, Yoon-Chan Jhi, Sencun Zhu, and Peng Liu. Behavior based Software Theft Detection. In *CCS '09: Proceedings of the 16th ACM Conference on Computer and Communications Security*, pages 280 – 290, New York, NY, USA, 2009. ACM.

[37] G. Whale. Detection of Plagiarism in Student Programs. In *Proceedings of the Ninth Australian Computer Science Conference, Canberra*, pages 231 – 241, 1986.

[38] Robert Zeidman. Patent: Detecting Copied Computer Source Code by Examining Computer Object Code, 2009.