Communications and Network, **2013, 5, 430-433**
http://dx.doi.org/10.4236/cn.2013.53B2079 Published Online September 2013 (http://www.scirp.org/journal/cn)

The Anti-Piracy Measure Using Encryption of Embedded Products May Mitigate the Security Strength

Zhilong Xiong, Rui Zhang, Xin Zhan, Zhenglin Liu
School of Optical and Electronic, Huazhong University of Science and Technology, Wuhan, China
Email: liuzhenglin@hust.edu.cn

Received June 2013

ABSTRACT

In this paper, firstly we describe the piracy problem of embedded products. Then we formulate the security features of anti-piracy embedded products. Finally we prove that the anti-piracy measure using encryption of embedded products may mitigate the security strength.

Keywords: Embedded Products; Anti-Piracy; Encryption; Security Strength; Copyright

1. Introduction

There are a vast number of embedded devices in a wireless network, such as wireless routers, handheld terminals and access points. In the fierce competition of the communications and consuming electronics market, there is a real-world problem: A design house takes several years to design an embedded product. Then they send the design information to a manufacturer, they find that the manufacturer produces their embedded products stealthily. In view of this situation, many design houses have provided their anti-piracy schemes. Even so, the problem of piracy is serious. The reason for this issue is that they don't realize the security features for the anti-piracy embedded products and often misuse the encryption technology against piracy.

2. Describe the Embedded Product Piracy

The situation is always like this: The pirate pretends to be a consumer and buy an embedded product. After that he can analyze the embedded product and get some design information about it. Then he produces the pirated products on large scale based on the design information. Finally, the manufacturer will be responsible for production. And from the **Figure 1**, we know that the manufacturer can easily take the design information rather than

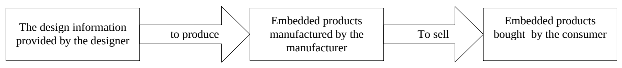
The design information provided by the designer
to produce
Embedded products manufactured by the manufacturer
To sell
Embedded products bought by the consumer

Figure 1. Embedded products from design to production and to sale.

*Corresponding author.

the consumer. So, by analysis, it is evident that the key problem between the designer and the manufacturer can be represented in **Figure 2**:

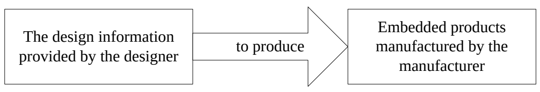
The design information provided by the designer
to produce
Embedded products manufactured by the manufacturer

Figure 2. Embedded products from design to production.

Obfuscation, Watermarking, and Tamper proofing for Software Protection was introduced in [1]. "Code packing transforms a program into a packed program by compressing or encrypting the original code and data into packed data and associating it with a restoration routine" in [4]. Security framework for embedded systems was discussed, and then copyright protection model has been proposed based on specific crypto memory IC in [3]. In [5], some software protection solutions in embedded condition, including the implementation schemes in pure software without any support from hardware, and the higher strength schemes bonding with specified hardware have been introduced.

Surreptitious software, proposed by C. Collberg and J. Nagra in [1], is a promising technology which can be used as the anti-piracy measure of embedded products. However the main goal of the surreptitious software is to protect the software which is running on the embedded product, not the embedded product itself. Code packing in [4] is effectively used to protect embedded software against reverse engineering. But, if embedded software has the portability, the pirate can simply bypass this security feature by transplanting embedded software. The limitation in [3,5] is their security strength based on a specified hardware. They assume that the information in the specified hardware can't be got by the pirate. But the pirates often get the information when they pretend to be the manufacturer.

Our aim is to prevent the manufacturer from producing embedded products stealthily. In other words, if the designer wants to produce one million products, we should insure that the production quantity of the manufacturer will be not more than one million. To achieve it, the anti-piracy embedded products should have some special security features. Now let's formulate these security features.

3. The Security Features of Anti-Piracy Embedded Product

Some terms are defined as follows:

E (The abbreviation of "embedded products") is the collection of embedded products.

H (The abbreviation of "hardware") is the collection of the embedded hardware. We use the hardware identifier to represent it, for example we always use the serial number of the CPU to represent hardware.

S (The abbreviation of "software code") is the collection of the embedded software. We use the software identifier to represent it, for example we often use the consumer data to represent the embedded software.

e is an element of collection *E*.

h, *g* is an element of collection *H*.

s is an element of collection *S*.

We assume that *e* is the design information of embedded products. Because embedded products are composed of hardware and software. We can represent *e* as $<h,s>$, namely $e=<h,s>$. Then the design house sends it to manufacturer for production.

The properties of the anti-piracy embedded products are as follows:

- Firstly they can't be duplicated by the manufacturer. $\Leftrightarrow$ Namely e is unable to be duplicated. $\Leftrightarrow$ $e=<h,s>$, and s is able to be duplicated, so h can't be duplicated by the manufacturer. $\Rightarrow$ Of course, h must be unique.
- What's more, the embedded software should have the non-portability. $\Leftrightarrow$ Namely, if $<h,s>=<g,s>$, $h=g$ ($h,g \in H$), that means there is function relation between h and g. $\Rightarrow$ Of course s must be unique.

By analysis above, the security features of anti-piracy embedded product can be summarized as follows:

- *h* is unique and immutable. And it can't be duplicated or modified.
- *s* is unique.
- If we represent the above function relation as f, we can get the function equation: $h=f(s)$. We call it "the blinding function equation". In practice, the manufacturer is easy to get the hardware information. It would be best to design the blinding function f as an irreversible function, so that the manufacturer can't conclude the information of *s* from *h* on the base of the blinding function *f*.

Firstly, I will introduce the immutability of *h*. Many designers use OTP or security chip to record the software information. In such a way, users can't modify the information. So, they can't transplant the software. However, the manufacturers have the access to that information. What's more, the money people spend on getting the information can be neglected, compared with the benefit people can get from the piracy. So this method is invalid. The immutability means that if the manufactures rewrite the information, it will pay a huge price. And the price is much higher than the value of embedded products.

Secondly, *h* should be unique, which means that different products can't have the same hardware identifier. For example, each processor has its unique identifier.

Thirdly, *s* should be unique, which means that different products can't have the same software identifier. Though the main software code is always the same, we can add some unique consumer data to make sure that no two software code can be the same.

Finally, *f* would better be irreversible. The SHA can be a good choice. Though this requirement is not necessary, we believe that it can improve the security of the product.

For example: if embedded product is Set Top Box, the serial number of the CPU could be *h*. It is unique and immutability. And the manufacturer can't duplicate or modify it. The consumer data could be *s*. And we can design the SHA as the blinding function *f*.

4. The Anti-Piracy Measure Using Encryption of Embedded Products May Mitigate the Security Strength

So, by analysis, the manufacturer is unable to duplicate or modify the hardware identifier *h*. Consequently, they can't copy embedded products simply. In order to pirate, they have to transplant the embedded software. The key to transplant the embedded software is to break the relation between *h* and s. In other worlds, their main object of attack is blinding function *f* and the software identifier *s*. So our main protected targets are *s* and *f*. If we design the blinding function *f* as the one-way hash function, the integrality of *f* will be more important than the confidentiality of *f*. Because, even if the pirate knows the blinding function *f*, he can't derive *s* from *h*, the only way for him to transplant the embedded software successfully is to modify *f*.

It is commonly observed that encryption techniques are preferred and regarded as top choice. It is the best choice for the confidentiality and the encryption technology such as digital signature is used to guard the integrality of a software code. Consequently, the encryption technology is often used to defend embedded products against piracy. But is it really works? Actually it is not the case because of following reasons:

1) The protection schemes using encryption cost some overhead, the more overhead we give to it, less overhead we can give to other protection schemes.

2) The encryption protects the confidentiality of data, but assuming that a secret key remains hidden. This means that we have to add some new protected target such as the secret key. If we use the encryption to protect our new protected target, we still have to add some new protected target for the same reason. At last, we have to use other protection schemes for the new protected target.

So, by analysis, it is found that anti-piracy measure using encryption of embedded product may mitigate the security strength. To prove it, let's analyze the relation between our protected targets and the new protected targets.

We can represent the new protected targets as the encryption function equation, namely $y = e(k, x)$. And *k* is the secret key, *x* is plaintext, *e* is the encryption function, and *y* is ciphertext.

For a specific *k*, each plaintext *x* has only one ciphertext *y* for it. This is just like our relation between *h* and *s*. In fact, if we have the ability to keep *y*, *e*, *k* and *x* safe, we must have the ability to keep *h*, *s*, *f* safe. Because we can design a secure blinding function *S* as $h = y$, $f = e$ and $s = x$. Furthermore, the new protected target, namely $y = e(k, x)$, has one more protected element '*k*'. So the new protected target is more difficult to protect than our original protected target.

We assume that *C* is the maximum overhead of our production schemes which embedded products can afford. G is a protection scheme we used, *c* is the protection overhead, and $G(c)$ represents the security strength of the protection scheme G. For the same kind of production scheme, we assume that the security strength is proportional to the protection overhead, namely $c1 < c2$, $G(\mathrm{c}1) < G(c2)$. Obviously $G(\mathrm{C})$ represents the maximum security strength.

$G1$ is the protection scheme using encryption, while $G2$ is not. Obviously $G2(C)$ represents the maximum security strength of the protection scheme without using encryption. Now we add the protection scheme $G1$. We assume that the protection overhead of $G1$ is *c*. And we can record the maximum security strength for the new mix protection scheme as $\{G1(c), G2(C-c)\}$.

1) If we are unable to keep the encryption function $y = e(k, x)$ safe, the protection scheme $G1$ will be invalid. And the maximum security strength of the new mix protection scheme $\{G1(c), G2(C-c)\}$ will reduce to $\{G2(C-c)\}$, namely $G2(C-c)$.

2) If we have the ability to keep the encryption function $y = e(k, x)$ safe. From the analysis given above, we know that the added protected target $y = e(k, x)$ is more difficult to protect than our original protected target $h = f(s)$. And the maximum security strength for the added protected target is $G2(C-c)$. Even if we make sure that our original protected target is safe so that the maximum security strength of the new mix protection scheme is totally depended on our added protected target, the maximum security strength of the new mix protection scheme will still reduce to $G2(C-c)$.

In a nutshell, regardless of whether the protection scheme $G1$ is valid, the maximum security strength of the new mix protection scheme will reduce. We can also get three deductions below:

1) More encryption technology we add, more new protected targets will be added, and more difficulties in protection.

2) More encryption technology we add, the less maximum security strength we can have.

3) When $G2$ can't fulfill the requirement security strength of our protected target, adding $G1$ can't fulfill the requirement too.

5. Conclusions

This paper concerns the piracy problem of embedded products and their anti-piracy schemes using encryption. We formulate the security features of anti-piracy embedded products and prove that the anti-piracy measures using encryption of embedded products may mitigate the security strength.

In this paper, the basic assumption here is that the embedded software is in NVRAM (non-volatile memory). Our ongoing research work is to analyze the security features when the embedded software is loaded into RAM (Random Access Memory). It is our ultimate goal to provide the perfect theoretical guidance for design houses against piracy.

6. Acknowledgment

This work was supported in part by National Natural Science Foundation of China (No.60973034, Grant No.61176026).

REFERENCES

[1] C. Collberg and J. Nagra, "Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection," Addison-Wesley Professional, 2009.

[2] H. B. Enderton, "Set Theory—An Introduce to Independence Proofs," North Holland Publishing Company, 1977.

[3] C. Y. Luo, G. S. Deng and Y. H. Guo, "Copyright Protection Model of Embedded Systems and Its Applications in Digital TV SET-Top-Box," 2008 *International Symposium on Computational Intelligence and Design*, Pro-

posed a System Copyright Protection Model Based on Specific Crypto Memory IC, 2008.

[4] Y. Park, “Design and Performance Evaluation of Binary Code Packing for Protecting Embedded Software against Reverse Engineering,” 13*th IEEE International Symposium on Object /Component/Service-Oriented Real-Time Distributed Computing*, 2010.

[5] Z. Y. Yang, “Software Protection Solutions for Embedded Systems,” *Computer Applications and Software*, Vol. 26, No. 8, 2009.