The Role of Cryptography in Combating Software Piracy

Jeff Bilger

ABSTRACT

Anti-software piracy techniques such as copy protection reached it's height in the late eighties and has been considered a failure by many. An entire cottage industry arose just to crack and release software as fast as possible, mostly for prestige. In this paper, I'll cover why cryptography is well suited[1] for use in anti-software piracy techniques as well as what can cause cryptographic techniques to fail. I will then briefly cover cryptographic techniques employed in both early and modern anti-piracy schemes and explain why some failed while others have significantly deterred piracy.

Introduction

The rationale for deterring software piracy is based on economics. If a company wishes to maximize their profits they may employ techniques such as copy protection or license keys in order to delay the time it takes for a cracked copy to become widely available. Most early attempts at copy protection were based on obfuscation and proprietary disk formatting techniques. These techniques were easily breakable and provided little in the way of deterrence, yet they increased the cost of the software. In some cases, software publishers attempted to use various cryptographic techniques in order to improve their copy protection, obfuscation, or license key schemes, yet implementation bugs or poor key management rendered those attempts trivially breakable as well. Even if cryptographically secure primitives and techniques are used, cost and engineering tradeoffs can result in unintended consequences such that the cryptographic-based anti-piracy scheme becomes vulnerable to attack. Despite these challenges, cryptography – if implemented correctly – is will suited to the task of deterring software piracy as well as delaying the time it takes for cracked copies to appear.

Why cryptography?

In order to deter software piracy and delay the time it takes for cracked copies of the software to appear, the cost to defeat the anti-piracy measures must exceed the value of cracking the software. Obfuscation and proprietary disk formatting techniques alone are not sufficient deterrents since pirate groups can easily crack software employing these

[1] Note: No anti-piracy technique is unbreakable. In the worst case, companies want to increase the time it takes to crack the software. In the best case, they want the cost of cracking the software to exceed any benefit.

techniques. Additionally, a common complaint users have against copy protection is that it prevents them from making legitimate backups of their software.

Therefore, techniques that are prohibitive to crack and allow users to copy the software freely yet only allow authorized users to run the software are desirable. Cryptography is well suited for this task due to the following cryptographic primitives:

- Digital Signatures
- Encryption
- Hashes
- Key Exchange

Digital Signatures

Digital signatures can be leveraged for multiple purposes. For example, to ensure that the code being run came from the software publisher, it can be digitally signed by the publisher. Techniques could then be employed to ensure only the code signed by the publisher would run on the user's computer. If pirates cracked the software and by definition modified some of the code, they could never digitally sign that code since they do not have the software publisher's public key. This technique is especially well suited for game consoles where the manufacturer has a high degree of confidence that software based debuggers and monitors could not be used.

Encryption

Encrypting the software using public or symmetric key cryptography and a strong key will provide a means of ensuring confidentiality and increasing obfuscation. However, if the code will be run on an insecure system such as a PC, techniques such as thwarting debuggers and monitors must be utilized since the encrypted code must be decrypted in unprotected memory in order to be executed by the CPU.

Hashes

Hashes can be used to ensure integrity of the code since if the hash of the code was different from the expected value, this would imply that the code had been modified. Hashes can also be used to uniquely identify a user's computer based on the unique signature of their hardware. This in turn could be used by the software publisher at activation in order to generate a license key as well as track license key usage. This hash has the added benefit that the original information used to compute the hash can not be recovered by the software publisher, thus alleviating privacy concerns.

Key Exchange

In distributed environments (i.e. anything outside a cryptographically secure system) cryptography can be used to exchange public keys in order to negotiate a symmetric session key. Once this session key has been agreed on, sensitive copy protection/verification code can be efficiently transmitted between the user and the game server over insecure communication links.

Any combination of the four cryptographic primitives can be utilized for anti-piracy purposes instead of relying on copy protection mechanisms. However, their efficacy is a function of the type of system the software being protected will be run on. In an insecure environment (such as a typical PC) an adversary will normally be able to access the RAM where the code is executing. As a result, they could use real time debuggers or monitors to reverse engineer and bypass the anti-piracy measures. In contrast, proprietary devices such as game consoles can have cryptographically secure areas (such as RAM) where the code executes. In such an environment, real-time debuggers or monitors would be of little use.

If implemented correctly, cryptographic techniques can make bypassing anti-piracy mechanisms prohibitive yet still allow legitimate users to backup their software. However, care should be taken since vulnerabilities can easily be introduced due to engineering tradeoffs and poor key management practices.

What can cause cryptographic techniques to fail?

Although the individual primitives of a cryptographic-based anti-piracy scheme may be secure, the protocol that employs these components can be is exploited. Additionally, engineering trade-offs and bugs in the cryptographic routines will have a significant impact on the efficacy of any anti-piracy solution.

Brute force attacks and cryptanalysis techniques are infeasible unless poor choices have been made regarding the size of the key, the key value, or the cipher algorithm used. As a result, an adversary will usually focus their attention on exploiting the cryptographic protocols or looking for weaknesses due to engineering trade offs.

Engineering Trade offs

Cost is the major factor in causing poor engineering trade off decisions. However, the capabilities of the target system which will run the software can also force engineers to make concessions regarding implementation decisions. For example, if the CPU is too slow or concurrent capacity is an issue, it may force the engineers to reduce the size of the key so that encryption/decryption operations take less time. If the size of RAM or ROM is not sufficient to store code, data, and required cryptographic routines, compromises must be made. Additionally, power consumption issues may force engineers to decide to not encrypt data over high speed busses.

Bugs and poor implementation

Introducing bugs when developing cryptographic-based algorithms and protocols is likely because of all the nuances of implementing cryptography correctly. Poor implementation decisions such as choosing algorithms that have previously been broken (SHA-1) or using cryptographically insecure ciphers such as substitution ciphers can be easily detected and exploited by an adversary. Moreover, choosing a poor pseudo-random function can result in reducing the strength of the chosen key since an adversary could detect that not all bits are equally likely to change.

Poor key management

The most cryptographically secure systems are only as secure as the private keys. Compared to designing and implementing secure, bug free cryptographic algorithms and protocols, key management is considered more difficult. The following key management issues can considerably reduce and, in some cases, nullify the benefit of utilizing cryptography:

- Poor key generation (can reduce the strength of the key)
- Poor key choice (exploitable by dictionary attack)
- Bribery (why perform cryptanalysis when you can pay for the key?)
- Storing the keys in an insecure manner (keys can be easily compromised)
- Transferring keys in an insecure manner (keys can be easily compromised)

Examples of cryptography in anti-piracy schemes

Since the early 1980's software publishers attempted to incorporate cryptographic techniques in the hope that it would reduce piracy. Despite their efforts, most attempts failed due to poor implementations or engineering trade offs.

Alternate Reality (1985)

This game was one of the first that used cryptographic techniques in order to make it more difficult to crack. This is significant since the computer it was designed to run on only had a 1.8MHz CPU and 48K bytes of RAM. It implemented a simple Block Chained multi-encryption cipher based on Leventhall/Seville cryptography (designed by Dr. Carl Meyer of Lucifer and DES fame). Unfortunately, the pre-generated keys were stored in minimally secured memory and disk. Additionally, the seeds used by the encryption routine suffered from a flaw such that if both seeds were zero, the plaintext code would not be encrypted in memory.

Despite these flaws, it was one of the most difficult software titles to crack of its time.

Windows Product Activation for XP RC1 (2001)

Activation was partly based on the hash of a set of 10 predefined hardware components. If more than 3 hardware components changed after activation, the user was forced to reactivate the product. Tests showed that you could easily modify the identifying traits for 6 of the 10 predefined hardware components. As a result, if the new computer you installed a pirated copy of Windows XP on had the same amount of RAM as the original then you could effectively defeat WPA.

Xbox (2001)

The cracking of the Xbox is the conical case of exploiting engineering tradeoffs in order to break a seemingly cryptographically secure system[2]. Microsoft knew they would lose money on the Xbox hardware itself so their business model was based on selling games. As a result, their design was centered around reducing the manufacturing cost of the

[2] The boot code used on the Xbox was encrypted with RC-4 and a 128 bit key so a brute force attack was not feasible.

Xbox hardware while providing a cryptographically secure system in which only authorized software[3] could be executed on the hardware. Due to economic pressures, the following engineering tradeoffs ultimately resulted in the Xbox being compromised:

- All Xbox's used the same secret key.
- The secret boot code and secret key were stored in ROM on an existing custom chip instead of on the CPU itself, thus requiring these two chips to communicate sensitive information via a high performance bus.
- The high performance bus, which was used by the cryptographically secure components to communicate, was not encrypted.
- The custom chip used to store the secret boot code and secret key only had 512 bytes of ROM available. This was insufficient to contain all the necessary cryptographic logic, resulting in the designers incorrectly relying on RC4[4] as a hash as well as using a constant checksum value to ensure validity of the boot code.

Despite the perceived security of the system, a hacker was able to capture the trace activity on the bus between two of the cryptographically secure custom chips. Since this bus was not encrypted, the hacker was able to reverse engineer the secret boot code as well as recover the secret RC4 key. Since the secret boot code was not hashed, it was possible for hackers to patch the boot code to suit their needs. This allowed hackers to run any software (authorized or not) on the Xbox.

As a response, Microsoft fixed some bugs and changed the RC4 secret key. Even though they replaced the use of a constant checksum with a hash in the secret boot code ROM, they made a poor choice in algorithms and used the Tiny Encryption Algorithm (TEA) to compute the hash. This was a poor choice because as early as 1997, it was known that TEA was insecure if used to compute hashes.

Other vulnerabilities existed that allowed the anti-piracy measures to be bypassed such as the MIST attack and Visor backdoor[5].

Valve Corporation's Steam platform

The Steam platform leverages both cryptographic techniques as well as required online registration in order to authorize and activate the software that Valve distributes. Although some cracks have appeared, the platform has allowed Valve to quickly detect and address the specific threats as they appeared.

During activation (which requires a Steam account), the user must authenticate their copy of the software by providing the license/CD key. Once authenticated, the RSA-encrypted

[3] In order to maximize software sales, Microsoft wanted to prevent users from running other OS's (Linux) on the subsidized Xbox hardware, prevent users from running PC software titles, as well as prevent the running of unauthorized or copied versions of Xbox games.
[4] Unfortunately, RC4 can not be used to calculate hashes, although RC5 can.
[5] For more information, see http://tinyurl.com/ff9r6

game code is downloaded from Value's servers, installed, and decrypted on the client's machine at runtime in RAM.

Conclusions

Software piracy is difficult to stop, especially on open systems where adversaries can utilize debuggers and monitors in order to bypass anti-piracy mechanisms. However by employing cryptographic techniques, software publishers can delay the time it takes for a cracked copy to become widely available, thus maximizing profits. Additionally, cryptographically secure server-based activation and authorization techniques can allow software publishers to detect and respond to unauthorized software use. Care should be taken because if cryptographic techniques are not implemented correctly, or if poor engineering tradeoffs are made, the cryptographic system or protocols can be vulnerable to attack.

References

Schneier, Bruce. Applied Cryptography. John Wiley & Sons, 1996.

Hartman, Mike. "Windows Product Activation compromised." http://www.tecchannel.de/client/windows/401701/index.html, 2001.

Steil, Michael. "Mistakes Microsoft Made in the Xbox Security System." http://tinyurl.com/ff9r6, 2005.

BBC News. "Ban Hits Half-Life 2 pirates hard." http://news.bbc.co.uk/2/hi/technology/4041289.stm, 2004.