gemalto
security to be free

WHITE PAPER

Understanding White Box Cryptography

Conventional means of cryptography are unable to provide a bulletproof solution that fully addresses diverse attacks scenarios attempting to exploit their inherent vulnerabilities.

Introduction

Traditionally, cryptography has offered a means of communicating sensitive (secret, confidential or private) information while making it unintelligible to everyone except for the message recipient. Cryptography, as was used in ancient biblical times, offered a technique in which text was manually substituted within a message as a means of hiding its original content. Many years later, during the Second World War, cryptography was extensively used in electro-mechanical machines (such as the infamous Enigma machine). Nowadays, cryptography is ever more pervasive heavily relying on computers supported by solid mathematical basis.

Cryptography, as the name implies, attempts to hide portions of text from malicious eyes using a variety of methods. In theory, the concept sounds ideal but real life experience has proven that a multitude of factors and environmental aspects come into play which have a negative impact on the cryptographic key's strength. Conventional means are unable to provide a bulletproof solution to fully address diverse attacks scenarios attempting to exploit cryptography's inherent vulnerabilities.

Professor Peter G. Neumann, a computer systems and networks trustworthiness and dependability, was quoted saying "If you think cryptography is the answer to your problem, you don't know what your problem is." [1]

This paper discusses traditional techniques while focusing on the White box cryptography implementation.

A Closer Look at Cryptography

In typical DRM (Digital Rights Management) implementations cryptographic algorithms are part of the security solution employing a known, strong algorithm while relying on the secrecy of the cryptographic key. In most cases, this is highly inappropriate since the platforms on which many of these applications execute on are subject to the control of potentially hostile end-users.

The conventional assumption for cryptography is a Black box setup that assumes the attacker has no access to the encryption key, can only control the encryption input (plaintext) and has access to the resulting output (ciphertext). For a long time this has been assumed to be true also for hardware devices like smartcards, but malicious attacks exploiting the information "leaking" from a Black box (such as Differential Power Analysis attacks—also known as DPA) have been developed allowing hackers to derive the secret keys used inside the Black box. This method has effectively allowed hackers to conduct non Black box attacks, and as a result turn these implementations into a "shade of grey" rather than Black. [2]

> Popular industry standard ciphers like AES were not designed to operate in environments where their execution could be observed. In fact, standard cryptographic models assume that endpoints, PC and hardware protection tokens for example, are to be trusted.

1. Peter G. Neumann, quoted in the New York Times, February 20 2001.

2. Amitabh Saxena, Brecht Wyseur, and Bart Preneel, Towards Security Notions for White-Box Cryptography

The Need for White Box Cryptography

Popular industry standard ciphers like AES were not designed to operate in environments where their execution could be observed. In fact, standard cryptographic models assume that endpoints, PC and hardware protection tokens for example, are to be trusted. If those endpoints reside in a potentially hostile environment then the cryptographic keys may be directly visible to attackers monitoring the application execution while attempting to extract the keys either embedded or generated by the application from memory.

This is a common problem for software based applications running on PC's, IPTV set-top boxes and other data consuming devices attempting to enforce DRM. By actively monitoring standard cryptographic APIs or memory dumps, hackers are then able to extract the key(s) whenever used. One example of a successful memory-based key extracting attack has enabled the BackupHDDVD tool to copy the content of a protected DVD and remove the DRM from Windows protected media content.

The White Box Challenge

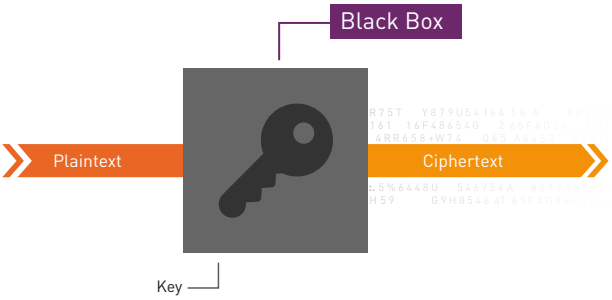
The notion of keeping valuable information such as licensing and other trade secrets hidden while operating in a fully transparent environment poses various challenges:

- How to encrypt or decrypt content without directly revealing any portion of the key and or the data?
- How to perform strong encryption mechanisms knowing that hackers can observe and or alter the code during execution?

The Various Cryptographic Models

Black Box (Traditional) Cryptography

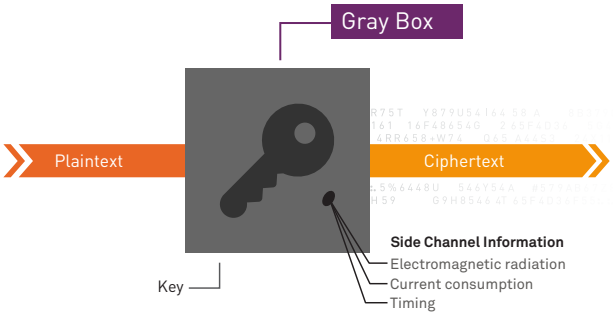
The Black box scenario, being a traditional model, assumes that the attacker has no physical access to the Key (algorithm performing the encryption or decryption) or any internal workings, rather can only observe external information and behavior. This information consists of either the plaintext (input) or the ciphertext (output) of the system while assuming zero visibility on code execution and dynamic encryption operations.



Gray Box Cryptography

The Grey box scenario assumes that the attacker has partial physical access to the Key or that it is "leaking" so called side channel information. Side Channel Analysis attacks (SCA) exploit information leaked from the physical implementation of a cryptographic system. The leakage is passively observed via timing information, power consumption, electromagnetic radiations, etc'. Protection against Side Channel Attacks is important because the attacks can be implemented quickly and at a low cost. Publicly available side channel information allows hackers to effectively reveal parts of the Key and as a result dramatically reduce its efficacy and demote the overall protection.[3]

Grey box cryptography is in fact a by-product of the traditional Black box implementation. It has been shown that even smartcards, perceived as being able to provide strong security, performing internal cryptography are in reality leaking information to the outside world. It is clear then that scenarios assumed to be a Black box are in reality only a shade of grey.



The Concept of White Box Cryptography

White box cryptography went head to head with the abovementioned traditional security models. As opposed to previous implementations where the attacker was only given a Black box, i.e. access to inputs and outputs and to the cryptographic algorithm under attack and assumed zero visibility into internal workings, White box provided full visibility instead.

White box cryptography techniques aim at protecting software implementations of cryptographic algorithms against key recovery even if the attacker has full control over the machine performing the encryption – especially useful in the DRM arena.

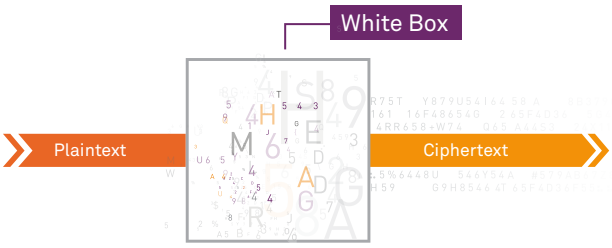
White Box Cryptography

The White box scenario, in contrast with previously described scenarios, handles far more severe threats while assuming hackers have full visibility and control over the whole operation. Hackers can freely observe dynamic code execution (with instantiated cryptographic keys) and internal algorithm details are completely visible and alterable at will. Despite of this fully transparent methodology, White box cryptography integrates the cipher in a way that does not reveal the key.

3. S. Chow, P. Eisen, H. Johnson, P.C. van Oorschot, A White-Box DES Implementation for DRM Applications

It is therefore clear that algorithms built for both Black and Grey box models are impractical in the face of operating on non-trusted hosts. Understandably, hackers will not try to break the cipher by only using the means available inBlack and Grey box scenarios, instead they will observe the execution when the unprotected key is used – directly stealing it.

Traditional cryptography algorithms, as exposed in the White box scenario, assume the presence of the key as part of the implementation.



The White box cryptography algorithm is protected in the White box scenario, as the key is not present in memory and cannot be extracted—not even dynamically.

Choosing the most appropriate, most secure cryptographic model is therefore the sole line of defense against malicious threats—precisely what White box cryptography attempts to achieve.

The Methodology Behind the White Box Implementation

How is it then possible to securely "hide" the key within the executed code assuming that one can fully monitor and alter each and every instruction?

Abstractly speaking, this is achieved by combining the effect of the secret key together with some implementation specific data using a mathematical operation, where it is made sure that the mathematical operation is virtually impossible to invert. [4]

As an example, the inherent strength of RSA is made possible through a simple multiplication to large numbers, although it is a mathematically hard problem to factor the result into its prime integers.

Additionally, and equally important, an implementation of a White box cryptography algorithm is solely able to either encrypt or decrypt.

The implementation is, as previously mentioned, based on a mathematical operation that is extremely hard to invert. This fact allows building a system that operates similarly to a full public/private key scheme, but at a performance level, is much closer to a standard symmetric cipher.

The decryption function can be implemented inside the distributed application but the key cannot be extracted and the decryption cannot be reversed as to perform an encryption operation. The attacker has no means possible to create the correct encrypted data that would decrypt back into the desired value.

This specific method is particularly useful for securing a communication channel protected with a hardware device, such as a hardware protection token. The attacker cannot extract the key used for the secure communication channel and is therefore unable to neither decrypt the data passing through the channel nor inject data into the channel, as he has no means of encrypting it correctly.

Solving the Challenge

Although the white box scenario is considered unsuitable for security-related tasks, White box cryptography shuffles all the cards and provides a highly secure method for performing encryption while operating in a fully transparent environment. Although fully transparent, both encrypt and decrypt operations allow maintaining sensitive data without revealing any portions of the key or the data itself. In addition, White box cryptography permits the execution of strong encryption mechanisms (in conjunction with other techniques) whilst knowing that malicious eyes potentially observe the code during execution.

An integral part of Gemalto's security measures

The secure communication channel provided by Gemalto's Sentinel products ensures that the communication between the protected application and the hardware token is encrypted and cannot be replayed. Unlike the previous implementation which aimed to hide the encryption key, the new implementation is centered on White box cryptography, where it is assumed that the attacker can trace the protected application and the run-time environment, in search for the encryption key. With this assumption as part of the design, the algorithm and encryption keys are replaced with special vendor-specific API libraries that implement the same encryption, but embed the encryption key as part of the algorithm, in a way that ensures that it's never present in the memory and therefore cannot be extracted. The generation of the vendor-specific libraries is performed on Gemalto servers utilizing several trade secrets. In addition, each application library is individually generated and obfuscated for a specific software vendor—making a generic hack virtually impossible.

A truly ground breaking solution

Gemalto is the first and only vendor to offer White box cryptography as an integral part of its Sentinel portfolio of software licensing solutions. This new technology allows protecting the cryptographic key at all times, rather than breaking it up and revealing it only a piece at a time. From a security perspective, this ensures that the protected key remains hidden from hackers and is therefore not susceptible to reconstruction during a potential attack process.

4. Amitabh Saxena, Brecht Wyseur, and Bart Preneel, Towards Security Notions for White-Box Cryptography

White box cryptography is an additional essential component that enables developers to protect their applications against reverse engineering, tampering, and automated attacks. Gemalto's White box cryptography methodology integrates into the software design process allowing embedding the additional layer of protection directly at the source code level thus providing a highly effective approach to software protection.

Conclusion

The overall security of a protected application is highly dependent on the implementation itself i.e. solely taking a strong cryptographic algorithm does not provide any security if it is not used in the context it was designed for—not using White box cryptography in a White box setup greatly helps hackers reverse engineer the protected software. Most common attacks have attempted to exploit software security flaws and not weaknesses in the cryptographic algorithms – but lately the attackers have recognized the vulnerability of classical cryptography in the open PC environment.

It is implicit that software protection must receive specific attention throughout the design and implementation stages in addition to being continuously enhance as part of the product life cycle and the release of new versions. In addition to White box cryptography, additional complementary security measures should be used to further strengthen the overall protection scheme.

Security comes at a certain cost and, as a direct result, cannot be air tight. It is therefore crucial to properly evaluate the required security level as dictated by the application itself i.e. the value of what needs to be protected in conjunction with the incurred losses assumed by neglecting potential risks.

Further publications

Additional information and detailed technical publications can be found in the below links:

1. Towards Security Notions for White box Cryptography www.cosic.esat.kuleuven.be/publications/article-1260.pdf
2. White box Cryptography: Formal Notions and (Im) possibility Results eprint.iacr.org/2008/273.pdf
3. White box (software engineering) on Wikipedia en.wikipedia.org/wiki/White_box_(software_engineering)
4. What is a white-box implementation of a cryptographic algorithm? crypto.stackexchange.com/questions/241/what-is-a-white-box-implementation-of-a-cryptographic-algorithm
5. Portable Executable Automatic Protection, Wikipedia https://en.wikipedia.org/wiki/Portable_Executable

About Gemalto's Sentinel Software Monetization Solutions

Gemalto, through its acquisition of SafeNet, is the market-leading provider of software licensing and entitlement management solutions for on-premises, embedded and cloud-based software vendors. Gemalto's Sentinel is the most trusted brand in the software industry for secure, flexible, and future-proof software monetization solutions.

Additional Resources on Software Monetization
Visit Gemalto's on-demand resource library to learn more about how you can better monetize your software.

Join the Conversation

- > Facebook
  www.facebook.com/licensinglive
- > LinkedIn
  bit.ly/LinkedInLicensingLive
- > Twitter
  twitter.com/LicensingLive
- > Google+
  plus.google.com/u/2/106533196287944993975/posts
- > Sentinel Video Cloud
  sentinelvideos.safenet-inc.com/
- > Blog
  http://www.licensinglive.com/
- > Sentinel Customer Community
  sentinelcustomer.gemalto.com

Contact Us: For all office locations and contact information, please visit www.gemalto.com/software-monetization

Follow Us: licensinglive.com

GEMALTO.COM

gemalto
security to be free