Securing Cryptographic Assets for the Internet of Things

White Paper

September 2015

Securing Cryptographic Assets for the Internet of Things

Introduction

THIS ARTICLE SURVEYS various white-box cryptography techniques for protecting critical cryptographic operations and data in an environment where adversarial users have complete control of the host computing platform (a **white-box attack context**). Microsemi® review s the need for white box cryptography, describes the techniques and technologies behind a typical white-box cryptography implementation, reviews how white-box cryptography resists attacks on critical cryptographic data and operations, and discusses important features in any white-box implementation. Finally, due to the expanding need for software cryptography combined with a rise in threats and attacks in the Internet of Things, Microsemi recommends white-box cryptography as an essential technology for protecting cryptographic operations in any software system.

The Need for White-Box Cryptography

The economic growth of the Internet of Things is unprecedented. With estimates of over 200 billion connected devices by 2020, Internet-connected devices are influencing many aspects of modern life. The Internet of Things is impacting a multitude of markets from robotics to point-of-sale systems; from mobile computing devices to 3D-printing. Embedded systems produced in these markets are helping us in the following ways:

- to inform us
- to make autonomous decisions on our behalf
- to communicate with business associates
- to manage our finances

Access to data, information systems, and digital content on these systems is commonly restricted using encryption. For encryption to provide effective access control, it is imperative that the cryptographic key used to encrypt the data is never revealed. Typical cryptographic implementations leave both the algorithm and key vulnerable to tampering and reverse engineering: the most vulnerable point for any crypto system implementation is the first moment at which the key is used. This point is easily identifiable in modern systems using signature, pattern, and memory analysis. As an example, key extraction attacks against keys coded as literal data arrays in unprotected software can be successfully completed in a matter of hours.

White-Box Cryptography Overview

White-box cryptography refers to a collection of methods for obfuscating cryptographic algorithms in order to hide their key material from unauthorized observers. White-box cryptography aims to prevent sight-sensitive information (such as, a key) in cryptographic operations from being revealed to an attacker even when he has full access to the system.

The name **white-box cryptography** is an analogy to **white-box testing**, where the tester is presumed to have access to all internal details of the system. A **white-box attack context** is therefore a situation in which the attacker has full control and observation of the host system. In contrast, a black-box attack context arises when the attacker may only observe and control the inputs and outputs to the host system at its external interfaces. In a white-box attack context, one assumes the attacker has full access to the system, its memory, its software routines, and so on. One can safely assume, as modern systems have become more open and mobile (laptops, tablets, phones), that they have become more accessible and therefore vulnerable to white-box attacks.

Figure 1 shows the relationship between a classical key and one possible white-box representation. It is a non-trivial relationship making it impractical to reconstruct the classical key using the tools available to a network-based attacker.

White Box AES Key:

```
0000000 0e18 80bc bae7 e250 708d ea28 04dd 9a18
0000010 f615 0d93 cf64 b9b6 7c5c 73be 2282 2176
0000020 d870 ade7 656b c188 48a4 cbe0 6ec6 9f1f
0000030 2c54 dd21 30f3 bdc7 d438 3b61 6850 8094
0000040 83e7 d907 57e8 db00 a39a 1ddb 59ec 7e29
0000050 6bae fd3d b2b0 604e edf7 98a3 c519 56c4
0000060 deb8 93c4 432d 4146 9fa6 5637 9d8e d7df
0000070 986e b925 d5a4 b1ed c4c6 3778 9cc8 aa8b
0000080 8006 3b73 2a7e 87d9 3248 157c b5f6 f9b3
0000090 c30c 3a62 0dbe 5bb4 0cc7 f788 664e 2f69
00000a0 57d0 ad7d 0b70 1a92 b251 efb3 60c0 bdf4
00000b0 27d2 ebdf 916d dae5 c981 be66 667c c9cc
00000c0 2634 e17c 082d d0f8 338f 3e58 c9ee 3780
0000040 2a01 9224 6d71 6344 66bb b037 5e96 2320
00000e0 13d7 d7aa 9f42 f210 5dfa 66b0 dc5b 070e
00000f0 a2dc 5fb3 7e53 bd5e 0830 e021 83cf 3764
0000100 e870 30a5 3320 8d0b aa3b f86a 3a75 e71c
0000110 5e85 84e8 1db4 6d82 0ee4 c64a 1bf7 2657
```

Based on the Classical 256-bit AES Key:

```
0000000 e502 d48a 18d7 95cd 5992 b8b0 d88b 65f1
0000010 78e8 264f 3652 bb4b fbb9 6802 c914 c4d0
```

***Figure 1* • Relationship between a White-box Representation and the Corresponding Classical Key**

A white-box implementation typically seeks to leverage combinatorial problems against an attacker such that access to or knowledge of the implementation does not compromise the key material even under direct observation of cryptographic operations. A typical white-box implementation of a cryptographic standard encrypts, decrypts, signs, and verifies sensitive data in the same way as a classical implementation, yet it attempts to force an attacker to reverse engineer complex mathematical transformations to obtain the secret key.

White-box cryptography is useful wherever cryptography must be performed in a potentially vulnerable environment, where the crypto keys and/or plaintext data must be protected, or where an untrusted user could take control of the host system. Such use cases include compromise of networked systems, software delivered to business competitors, or commercially deployed software with private keys.

Preventing Attacks with White-Box Cryptography

One relevant example of a high-profile attack is the 2014 Heartbleed vulnerability that allowed an attacker to retrieve memory contents from vulnerable server-side software (namely OpenSSL). A properly constructed Heartbleed attack exploits an input validation error in a way that causes the server to send a small portion of its memory contents to the attacker. The memory contents delivered to an attacker may contain portions of cryptographic key material used to secure communications between the server and the outside world. Exposing keys can lead to compromise of the (very sensitive) data being protected by the secure communications channel.

Had the key-material been resident in memory in a non-standard form, the impact of memory-exposure vulnerabilities such as Heartbleed would have been lessened. Generally speaking, any attack based on capturing or replacing cryptographic keys becomes more difficult when a white-box key representation is in use.

Important Techniques in a White-Box Cipher Implementation

White-box products and technologies vary from institution to institution. The following features should be considered when evaluating white-box technologies:

- Diversity
- Cipher Specific Obfuscation
- Hardware Binding
- Side-Channel Resistance
- Support for Obfuscated I/O

Diversity

Rather than implementing a single white box cryptography algorithm for all users (which would lead to break-once-run-everywhere attacks), code generators should be used to produce unique variants of the algorithms.

Cipher Specific Obfuscation

White-box implementations should be designed using alternate mathematical methods and obfuscation techniques tailored to the target algorithm. White-box implementations should not simply apply automated transformations to classical cipher implementations.

Each algorithm or cipher should be modified in ways that leverage the specific properties of the underlying mathematics; blanket transformation should never be applied over all algorithms. As a rule-of-thumb, the more general a transformation is, the easier it is to reverse-engineer.

Many times, standard cryptographic algorithm designs result in implementations that have fundamental vulnerabilities to white-box attacks because they make an explicit assumption of executing on a secure host. A strong white-box implementation should mitigate these vulnerabilities.

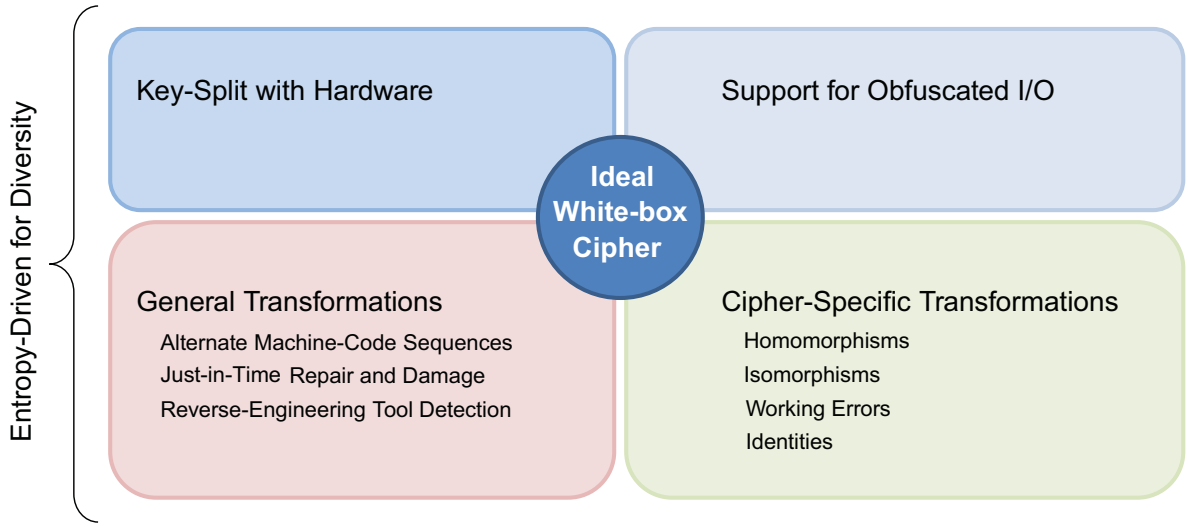

***Figure 2* • Important Techniques in White-box Cipher Implementation**

Hardware Binding

Software is inherently easier to attack than hardware. By simply copying the original software system bit for bit, an attacker is guaranteed unlimited attempts to break the system. However, hardware can enforce more permanent penalties. A strong white-box cryptography implementation should take advantage of hardware when available to limit the reverse engineering attempts on the obfuscated algorithm(s).

It is possible to construct a cryptographic key as a split between the data derived from a hardware challenge, and the data stored in a non-volatile storage. White-box cipher implementations that support such a split can then leverage hardware sensing and anti-tamper features as a prerequisite to cipher operation.

Side-Channel Resistance

Resistance against side-channel attacks (such as simple or differential power analysis) are paramount to protecting the key material from exposure. A solid white-box cryptography implementation should utilize numerous side-channel analysis countermeasures to resist exposing the key to such attacks.

Support for Obfuscated I/O

In many cases, systems employing cryptography must periodically update or refresh keys, a requirement referred to as **key management**. This requirement presents one of the primary security risks for systems exposed to white-box attacks: How can one receive or derive new keys without exposing the systems to an observer?

One approach is to produce ciphers that have obfuscated I/O interfaces. That is, ciphers that support consuming obfuscated data, producing obfuscated data, or both. If this is combined with an ability to produce white-box key representations from obfuscated representations of keys, the base technology is present for building a key-management system.

As a simple example, consider a device that supports obfuscated-out RSA decryption, and white-box AES-key preparation from obfuscated data. Such a device can inter-operate with a non-white-box-aware key-management server in the following way:

1. The server can send an RSA-wrapped AES key to the device.
2. The device decrypts the wrapped key to obtain an obfuscated version of the AES key, and subsequently runs the routine to prepare a white-box representation of the AES key from the obfuscated result of the RSA decryption.

In this way, the device obtains an appropriate white-box representation of the AES key without exposing the corresponding classical key to the observer.

Conclusion

Given the rise in mobile Internet connected devices combined with a growing need for secure operations and communications, a strong white box cryptography implementation using (at a minimum) the techniques described above should be considered an essential component to any software system that requires cryptography. White-box cryptography is an important tool in the systems-security-engineering toolbox; but like any security technology, it has strengths and weaknesses, which must be accommodated in the overall security design. Refer to the *Threat-Driven Security* white paper for more information on how to conduct systems-level security analysis.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA

Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

E-mail: sales.support@microsemi.com

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense & security, aerospace and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 3,600 employees globally. Learn more at **www.microsemi.com**.