Microsemi
Power Matters.™

Cryptographic Key Protection

OpenSSL is the backbone of a large majority of Internet connected information systems, protecting digital content and sensitive user information with encryption. As robust as OpenSSL is, the single point of failure for this or any crypto system is the instance in which the cryptographic key is used an easily identifiable point in software routines using signature, pattern, and memory analysis. Typically, key extraction attacks against keys coded as literal data arrays in unprotected applications can be successfully completed in a matter of hours. With the recent Heartbleed vulnerability, the world was shocked that such attacks can be performed remotely and without trace.

Microsemi's WhiteboxSSL™ product is a drop-in replacement for OpenSSL, combining mathematical algorithms, data, and code obfuscation techniques to transform the key and related crypto operations in complex ways requiring deep knowledge in multiple disciplines to attack. Importantly, **the key is never present in static or runtime memory**. Rather, the key becomes an inert collection of data that is useless without the uniquely generated white box algorithm. WhiteboxSSL™ supported crypto algorithms include:

WhiteboxAES™ 128, 192, and 256-bit key protection

WhiteboxRSA™ Encrypt/decrypt/sign/verify all key sizes

WhiteboxECC™ p160/192/224/256/384/521 prime curve

WhiteboxSHA™ SHA1, SHA2 224/256/384/512 + HMAC

WhiteboxOXD™ Obfuscation for sensitive data transfer

WhiteboxSSL™ provides protection from known and future OpenSSL vulnerabilities and attack vectors

WhiteboxFFC™ Finite Field Crypto/Diffie-Hellman-Merkle

WhiteboxJCE™ Facilitates the use of third party implementations through Java Cryptography Extension (JCE)

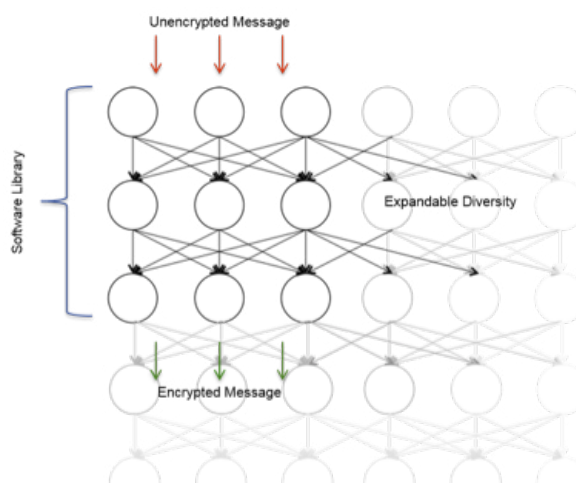**WhiteboxTLS™** Transport Layer Security (TLS) protocol

Whitebox Key Transformation

To protect encrypted information, it is imperative that the key never reveals itself in memory or on disk. Standard crypto implementations leave both the algorithm and key vulnerable to tampering and reverse engineering.

WhiteboxSSL™ mathematically transforms the key into a complex graph of numbers and executable code. This graph has multiple valid paths randomly chosen at runtime based on a user-supplied random source.

> WhiteboxSSL™ provides protection of your keys against unknown future attack vectors!

Hardware ID binding allows integration of a hardware identifier into the WhiteboxSSL™ library forcing an attacker to reverse engineer a complex, dynamically changing key-graph tied to a single hardware system.

WhiteboxSSL™
Cryptographic Key Protection - "Your server keys…NEVER Exposed"™

The WhiteboxSSL™ product allows you to: Generate a unique crypto library for each application shipped, and encode the same classical key for each; produce a single library and encode many keys to work with it; or encode multiple keys for multiple libraries. By simply supplying a classical crypto key, tweaking a few configuration variables, and running WhiteboxSSL™ in any build environment, a new library will be generated which can be integrated into an application using a fully documented software API.

Supported Platforms

WhiteboxSSL™ supports configurable key sizes, are little and big endian compatible, run on both 32 and 64-bit systems, and are fully compatible with any environment that can link C libraries. WhiteboxSSL™ is a drop-in replacement for OpenSSL so minimal changes are necessary on systems on which OpenSSL is already installed. WhiteboxSSL™ was developed in the United States of America, is EAR export controlled, and is immediately ready for deployment in C and C++ software applications running on nearly any hardware and operating system configuration.

Use Cases

WhiteboxSSL™ is useful wherever cryptography must be performed in a potentially vulnerable environment or where the crypto keys and/or plaintext data must be protected even if an untrusted user has taken complete 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.

Additionally, WhiteboxSSL™ can receive input and produce output in an obfuscated data format, suitable for use with other algorithms in the WhiteboxCRYPTO suite. In this way, WhiteboxCRYPTO can keep data secured in addition to key material.

Features and Benefits

Features	WhiteboxCRYPTO™ Benefits
Hides Keys Completely	Actual key bits never form in memory, thwarting various memory attacks. The obfuscated white box form is resistant to break-once-run-everywhere exploits.
Simple, Documented API	A simple, fully documented API enables quick implementation of secure encryption, decryption, signing, and verifying functionality.
Binds Keys to Hardware	Hardware identifiers can be mathematically integrated into a key, binding an application and sensitive data to a particular hardware platform.
Customizable Performance	Tunable encrypt/decrypt throughput allows full performance vs. security tradeoffs.
Randomizes at Runtime	For each execution, WhiteboxSSL™ takes a random path, confusing and forcing an attacker to reverse engineer a complex node graph.
Highly Portable	Source code based implementation is portable to all platforms and compatible with any software protection technique. WhiteboxSSL™ functions as little or big endian, 32 or 64-bit, compatible with any environment that can link C libraries.
Managed Keys Solution	The white box version of the key can be stored externally to a WhiteboxSSL™ library enabling key updates, key escrow, etc.
Protects data in-transit	WhiteboxSSL™ produces data in obfuscated form usable by other algorithms within the WhiteboxSSL™ suite. Thus, data is protected during intermediate stages of a sequence of cryptographic operations.