thalestct.com

THALES

Roots of Trust

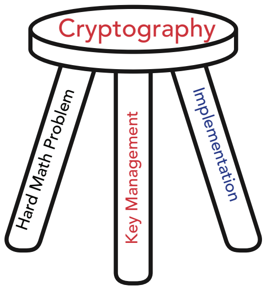
Five Things You Must Know

White Paper

The term Root of Trust (RoT) is commonly used in information security circles, but what does it mean? Why do we care? How does it apply to cryptographic controls? Modern computer systems are incredibly powerful and flexible. They can be molded to accomplish things that were unimaginable a mere decade ago. This same property makes them almost impossible to control and all too easy for malicious actors to find ways to disrupt them. To counter these threats, security experts have resorted to a wide range of cryptographic tools, and for these tools to function they need a trust worthy beginning.

Why do we need a root of trust?

Cryptography is a powerful defense against all kinds of threats as it creates impenetrable walls around our digital assets. It keeps information from prying eyes; prevents unauthorized changes to it and attests to its authenticity. However, cryptography can only achieve this when it is deployed correctly. To achieve this, three critical factors must be met:



Use qualified hard mathematical problems
Keep the cryptographic keys secret with best-practices key management
Ensure the implementation is trust worthy.

Like a three-legged stool, a failure in any one of these foundational elements potentially leads to catastrophic failure of the system's security controls.

Identifying the hard mathematical problems is the job of cryptographers with support from standards organizations such as the National Institute of Standards and Technology (NIST). Cryptographic algorithms change over time, but there are well established processes in place to evaluate new algorithms and clearly define which ones are trustworthy. Solving the key management and implementation challenges are a little more difficult. It is all too easy to lose control of a key if you haven't carefully deployed strong controls.

There are many examples of industry failing on these counts; some great examples are Poodle, Heartbleed and Spectre/Meltdown. The Poodle attack is a popular example of implementation errors. The core, hard math problems represented by RSA, ECC and AES were not challenged, but the way they were deployed at the protocol level failed. Well before identifying the specific flaw uncovered by Poodle, researchers recognized that SSL 3.0 had weaknesses, so they addressed them by introducing Transport Layer Security (TLS). Support for TLS rolled out across web servers and browsers, but to maximize interoperability, implementers chose to leave SSL 3.0 active much longer than necessary. Since the Poodle flaw only existed in SSL 3.0, the vulnerability uncovered by Poodle could have been removed from systems before it had even been discovered. Poodle actually illustrates two implementation flaws: unforeseen ways to bypass protocol implementations and the unforeseen consequences of deployment decisions that make compromises with security best practices.

Another example of combined implementation and key management failures can be found in the highly publicized Heartbleed and Spectre/Meltdown vulnerabilities. Both represent flaws that allow remote attackers to steal keys or data from any server that can be reached. Heartbleed can be used to extract keys and data held in the web server's application memory. Spectre/Meltdown can be used to extract keys and data from any memory on the server: the web server application, the operating system, other virtual machines on the same system, and even code trying to protect itself using Intel's advanced Software Guard Extensions (SGX). Of course, keys need to be on the server for them to be stolen, so the key management mistake in these examples is choosing to allow keys to be used on the server – a device with so much complexity that fully locking down the implementation is a daunting task.

The existence of Heartbleed brought into question every TLS private key on every server that used OpenSSL. Spectre/Meltdown were similar in nature, but broader in scope bringing into question every key on every server using the flawed hardware. That is, not just the TLS key for the web server, but any private or symmetric key that was on the physical server for any purpose.

Similar flaws do not affect just the confidentiality of your cryptographic keys, they also represent a serious risk to your server's integrity. Through them, malware and rootkits can be installed that compromise your system on an ongoing basis. Malware can enter your server through any crack in its access controls – Heartbleed, Spectre/Meltdown are just the tip of the iceberg. Once on your server, ebooting or even reinstalling everything will often not clear it. This can result in an ongoing compromise of all keys that get loaded on the server. The only way to combat this kind of malware is to use a secure boot process that starts from a root of trust.

What is a root of trust typically used for?

On a typical computer, a secure boot creates a Root of Trust (RoT) by leveraging a physically integrated key store and initial boot code. Typically, a Trusted Platform Module (TPM) stores critical keys, and the initial code is physically integrated into the computer's motherboard. In mobile phones and advanced IoT devices, the RoT is boot code and keys fused directly in the specialized chips used in the device. In all cases, the primary characteristic of the RoT is that it is physically unchangeable with highly restricted interfaces. These properties achieve trustworthiness by preventing any change to its behavior and using highly restricted interfaces to ensure there are no unforeseen ways to misuse the RoT. However, computers, mobile devices, and IoT are highly cost sensitive and require operational models that are transparent to their user base. This puts constraints on these standard RoTs.

The cost pressures lead to limiting the RoT performance and complexity. This causes the RoT to focus on the trustworthy beginning. Once the system is up and running, the RoT releases keys to the system so performance requirements can be achieved. This exposes some RoT-protected keys to vulnerabilities like Heartbleed and Spectre/Meltdown.

The users of computers, mobile devices and IoT devices also have limited to no training on security practices. This leads the RoT to relying on limited well-defined management roles that are largely transparent to the users. This is a great attribute for these types of devices but restricts the ability to bind RoT authorizations to trusted individuals within an organization.

The core features allow the standard RoT to accomplish critical security functions for their systems: ensuring the system integrity by detecting the presence of even the most sophisticated malware during boot. However, since cryptographic keys must be protected always, even when in use, a standard RoT is not enough. Cryptographic keys require a cryptographic RoT. One that is up to the challenge of protecting keys through their entire life cycle; and one that mitigates against the likes of Heartbleed, Spectre/Meltdown and more.

How does a cryptographic root of trust work?

It is generally known that public key infrastructures (PKI) create a web of trust. At the root of every PKI is a Certificate Authority (CA) that ultimately attests to the authenticity of every party's keys. The CA is the inherently trusted RoT at the foundation of every PKI. Without it, parties could not efficiently authenticate any other party it communicates with. If a CA's private key is lost or misused, the integrity of the entire system built on the PKI is lost. It is critical to protect the CA's private key.

The CA is just the root of the trust tree; as you extend out across the system there are many other critical points where the loss of a key compromises substantial parts of a system. For example, loss of a TLS private key enables attackers to impersonate a system's web presence for the life of the lost certificate – often meaning years! Sure, certificates can be revoked in theory, but revocation checking is far from a guaranteed failsafe. Lose a document signing key and the authenticity of every signed document, past, present, and future, is now in question. Revoking credential issuance keys leads to massively expensive re-provisioning of user tokens or devices, and when used to issue IoT device identities, revocation may require dispatching maintenance staff to every end-point. Losing any of these keys can be catastrophic.

For all these use cases, a high assurance cryptographic module is required to protect the keys. It is easy to jump to the conclusion that a software based cryptographic module is good enough, after all the server has many layers of access controls. However, it should now be clear that this conclusion is wrong. Even a TPM-protected software module does not protect keys from real world threats like Heartbleed, and Spectre/Meltdown. As the potential impact of a key compromise increases, a stronger mechanism is needed to protect keys against many real-world threats.

Where does a hardware security module fit in all this?

Hardware Security Modules (HSMs) take over when software, and even TPM-protected software, fall short. HSMs are high-assurance cryptographic modules that easily mitigate against attacks like Heartbleed, Spectre/Meltdown. They do this by leveraging and extending standard RoT principles – by further anchoring keys in the physical world.

Most obviously, the HSM keeps keys physically separate from the server by storing and using them inside the HSM security boundary. No matter where malware looks on a server, it will not find the key. Using the key within the HSM is the only way to mitigate against vulnerabilities that allow an attacker to gain access to the server's memory. It is also important to store keys within the HSM. Storing them externally allows attackers to bypass the HSM's physical protections and relies on the key encryption implementation as the single layer of defense.

HSMs also provide highly restricted server interfaces that allow keys to be used only within a strictly defined set of policies. This prevents unauthorized parties from using the keys and prevents even authorized users from using keys in potentially insecure ways. Like standard RoT devices, an HSM maintains its implementation integrity by always booting from trusted code and verifying its firmware just like a server's secure boot. This ensures the HSM always enforces its policies and access controls. There is no way for an attacker to change the HSM's behavior.

An HSM's strong physical barriers prevent physical tampering and probing, and its logical access controls keep out malicious insiders. Finally, if the HSM is good at its job, it clearly defines a key's locality. That is, when all copies of a key are strongly tied to a physical device it is easy to demonstrate control to auditors. More than a checkbox, the FIPS 140-2 Level 3 validation process ensures the implementation is correct – something all too commonly not achieved by lesser commercial solutions. All this allows an HSM to anchor critical keys in the physical world, realizing a cryptographic RoT.

This makes an HSM the mandatory element of many keys within cryptographic systems. Whether it is protecting the elements of a Certificate Authority, TLS server private keys, document signing keys, or human or device identity issuance keys. If the impact of a key loss goes beyond a single entity, the key should probably be protected by the physical controls offered by HSMs.

What to look for in an HSM?

Selecting an HSM can be a daunting task. There are many choices on the market, all making similar security claims but having very different operational characteristics. How do you choose? The task becomes easy once you realize a core goal of the HSM is to implement a cryptographic RoT for your system. Just determine which HSM does the best job rooting itself in the physical world and you will have found the best HSM.

Keys-in-Hardware

When an HSM relies on external storage it exposes keys to many threats. Even when stored keys are encrypted, they have lost several of the physical barriers that an HSM should provide. Storing keys within the HSM clearly defines their locality, and this ensures all the physical access control's – safes, guards, secure data centers, and the HSM's own physical security measures – actually protect all copies of your key all the time. If your HSM uses external storage mechanisms who knows how many copies of the backup file have been made! Padding Oracle vulnerabilities like Poodle – a flaw in the way encryption is implemented - should make it clear a single layer of protection is inadequate for your high-value keys.

Fixed Key Management Policies

An HSM should implement critical key management policies directly in the hardware in a physically unchangeable manner. Of course, administrators need the ability to tailor the HSM to each application. However, certain critical policies should not be left to applications, or be put at risk to malicious insiders. For example, when using an HSM for keys that need to stay put, use an HSM that has no key export capability. Making this a physically unchangeable policy within the hardware anchors the policy in the physical world.

Strong Role Separation

Administration roles implemented by an HSM should enable separation of duties that map well to the HSM's operational realities. A single super-user style administrator is certainly convenient, but it also forces the role to be shared with too many individuals. Instead, the roles should separate service activation, policy management activities, and HSM domain authorization. Isolating the role of domain authorization, the act of authorizing a new HSM to store and use specific keys, enables tight control of key locality. That is, simply lock this role up in a safe and bring it out only when new devices need to be added to the pool of authorized HSMs. When the role that puts an HSM into service also authorizes new devices to hold the keys, a single point of failure is created and placed in the hands of junior infrastructure staff.

Trustworthy Supply Chain

Your HSM supply chain is another critical factor in the physical security of your HSM. Most HSMs are developed on foreign soil and are thus subject to influence by foreign actors whose interests may not be aligned with the US government and its citizens. Ensure you HSM is designed and manufactured by a U.S. based provider of high-assurance data security products.

What are you waiting for?

So now you understand RoT technology and why HSMs have become an industry best practice by realizing cryptographic RoT for information systems. They're not just about satisfying the mandate to use FIPS 140-2 validated products. HSMs are a fundamental part of what makes your security controls effective. If you don't have HSMs deployed for every high-value key in your information system, there may be an exposure that could have substantial impact to your system's integrity. And if you have HSMs deployed, but their architecture doesn't follow key RoT principles, then you should be wondering if you really got the benefit HSMs are intended to deliver.

About Thales Trusted Cyber Technologies

Thales Trusted Cyber Technologies, a business area of Thales Defense & Security, Inc., protects the most vital data from the core to the cloud to the field. We serve as a trusted, U.S. based source for cyber security solutions for the U.S. Federal Government. Our solutions enable agencies to deploy a holistic data protection ecosystem where data and cryptographic keys are secured and managed, and access and distribution are controlled.

Contact Us: For more information, visit www.thalestct.com

3465 Box Hill Corporate Center Drive, Suite D, Abingdon, MD 21009 • 443-484-7070 • info@thalestct.com
thalestct.com