presented by

Implementing and Using the UEFI Key Management Service (KMS)

UEFI 2020 Virtual Plugfest

September 17, 2020

Presented by Zachary Bobroff and Alex Podgorsky

Meet the Presenters

Zachary Bobroff
Technical Marketing Manager
Member Company: AMI

Alex Podgorsky
Firmware Engineer
Member Company: AMI

Agenda

- Introduction
- Data Security
- UEFI KMS Implementation
- Call to Action

Implementing and Using the UEFI Key Management Service (KMS)

Introduction

The UEFI KMS Protocol

- The UEFI specification has included the Key Management Service (KMS) protocol definition since version 2.3.1 in 2011. This provides services to generate, store, retrieve and manage cryptographic keys.
- What are some use cases of the KMS protocol in modern systems?
- What does the interaction between UEFI firmware and a KMS server look like?
- How can you implement the UEFI KMS protocol over Key Management Interoperability Protocol (KMIP)?

What is KMS?

- Key Management Service (KMS) provides a way to generate, store, and authenticate cryptographic keys
  - KMS servers are an integral part in data security and can communicate with UEFI firmware to securely transfer keys and protect encrypted data
  - KMS servers are already used extensively in enterprise environments
    - For example: Windows volume licensing, Cloud KMS services, etc.

EFI_KMS_PROTOCOL Structure

```
typedef struct _EFI_KMS_SERVICE_PROTOCOL {
    EFI_KMS_GET_SERVICE_STATUS GetServiceStatus;
    EFI_KMS_REGISTER_CLIENT RegisterClient;
    EFI_KMS_CREATE_KEY CreateKey;
    EFI_KMS_GET_KEY GetKey;
    EFI_KMS_ADD_KEY AddKey;
    EFI_KMS_DELETE_KEY DeleteKey;
    EFI_KMS_GET_KEY_ATTRIBUTES GetKeyAttributes;
    EFI_KMS_ADD_KEY_ATTRIBUTES AddKeyAttributes;
    EFI_KMS_DELETE_KEY_ATTRIBUTES DeleteKeyAttributes;
    EFI_KMS_GET_KEY_BY_ATTRIBUTES GetKeyByAttributes;
    EFI_KMS_KEY_ATTRIBUTE *KeyAttributes;
    …
    …
} EFI_KMS_PROTOCOL;
```

No explicit references to hardware implementation

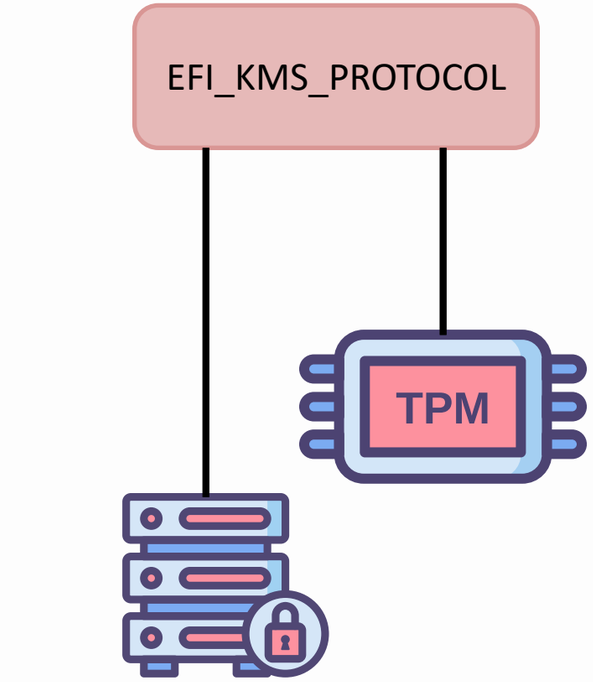
The protocol is abstract, and a hardware wrapper driver must be included to provide access to the security service (e.g. TPM, HSM, KMS, etc.)

Progression of UEFI KMS Protocol

- When the protocol was introduced, the UEFI Network Stack did not have secure communication capabilities
  - For example, there was no HTTP with TLS
- Current UEFI interfaces – HTTP(s) (UEFI 2.5, 2015) – make it possible to develop a fully generic and portable EFI KMS solution employing existing EFI Network Stack protocols to communicate securely with KMS servers
- For example, encrypted drives can be decrypted utilizing secure authentication using the UEFI KMS Protocol

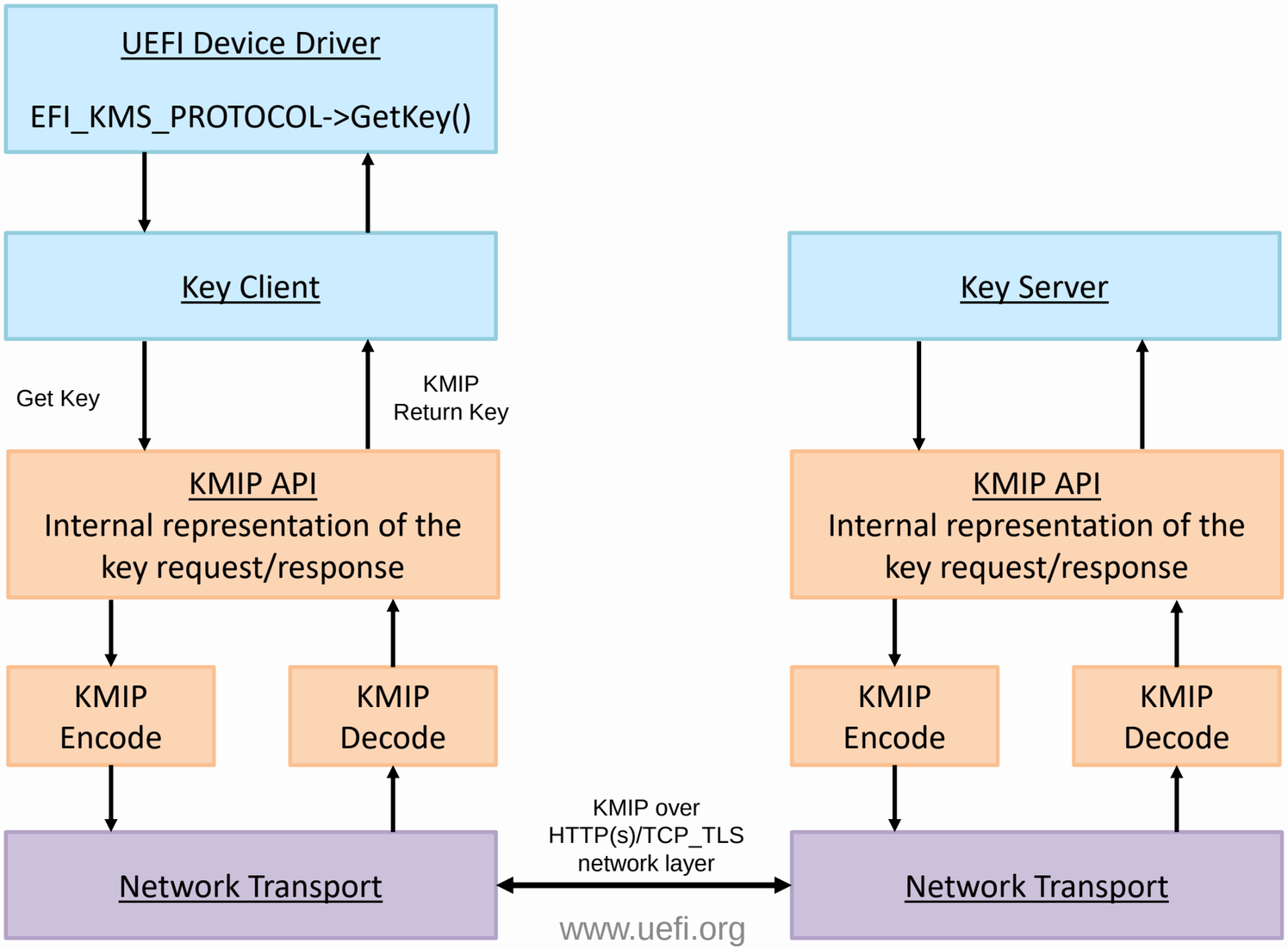
KMIP Over KMS

- As usual, the UEFI specification provides just the definition for the service and the underlying implementation can vary
- There are various possible implementations of the KMS protocol
  - A simple implementation is to build it on top of something already in the system such as a Trusted Platform Module (TPM)
  - The most practical UEFI implementation involves interfacing with a KMS server using the KMIP industry standard over a secure network connection
    - KMIP is an OASIS standard for the management of objects stored and maintained by key management systems
    - KMIP defines how key management operations and data should be encoded and communicated between client and server applications
    - For more information on KMIP, check out the OASIS KMIP Technical Committee



Key Management Server

UEFI Driver Consuming KMS Protocol via KMIP

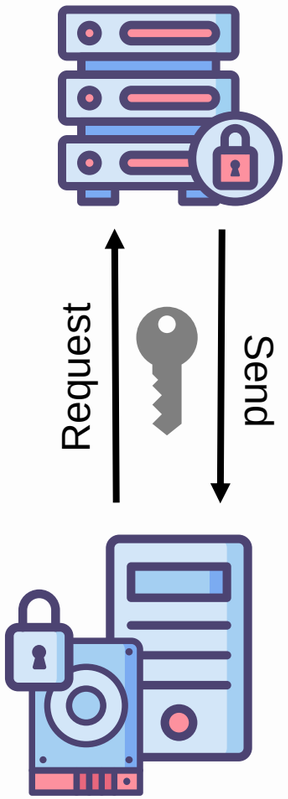


Implementing and Using the UEFI Key Management Service (KMS)

Data Security

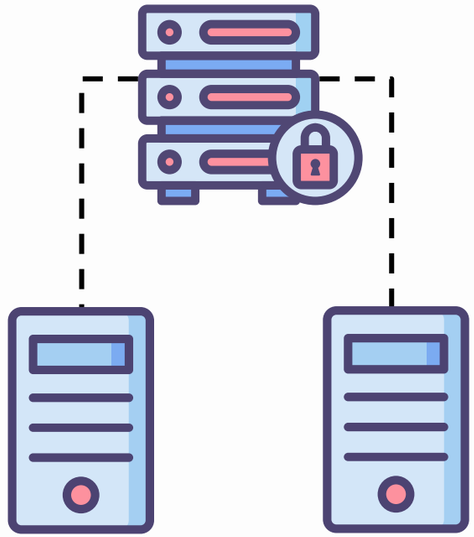
Keeping Data Secure Using KMS

- Data privacy continues to be a high-priority concern, especially in enterprise and cloud environments
  - Data privacy and protection policies and regulations have strict requirements
    - Many companies encrypt their storage devices to comply with these regulations
  - Self-encrypting drives (SED) are a popular solution to data security, but are commonly “married” to their platform today
- How can you boot to an encrypted drive?
- How can an SED migrate to another platform without breaking functionality?



Using the UEFI KMS Protocol to Boot to an Encrypted Drive

- Encrypted drives can be unlocked during boot or in the OS, but what if such a drive is the boot device or has data needed during the boot process?
- Traditionally, encrypted drives have needed user input or static keys on a platform to unlock the drives during the boot process
- KMS offers a secure way to manage cryptographic keys used by encrypted drives for authentication
  - Automatic authentication does not require user input
  - Greatly beneficial to enterprise and HyperScale environments
  - In case of system failures, encrypted drives can be migrated to other systems and remain operable and secure

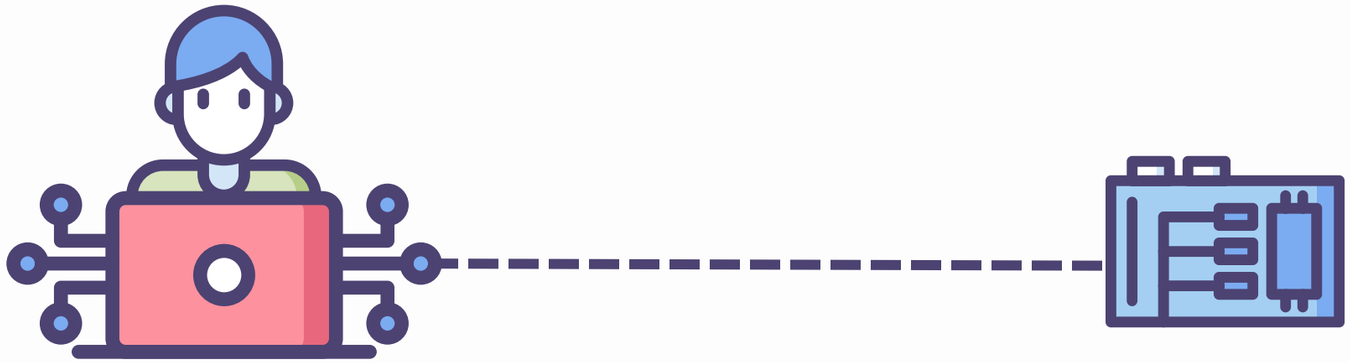


Implementing and Using the UEFI Key Management Service (KMS)

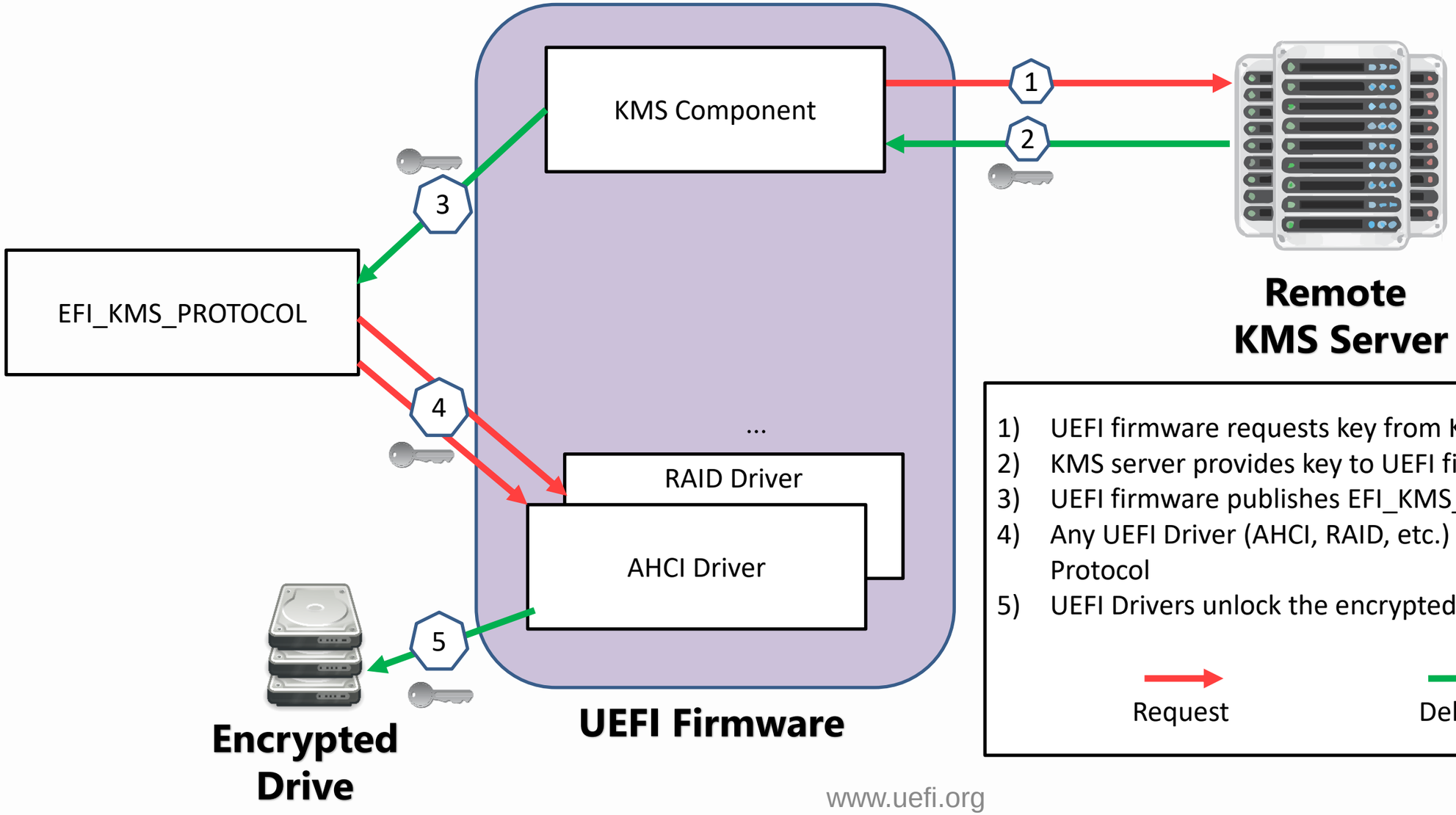
UEFI KMS Implementation

The Role of Independent Hardware Vendors (IHVs)

- Successful implementation of the UEFI KMS Protocol requires storage controllers to consume it
  - Tight collaboration with system firmware is required by IHVs when designing device drivers



UEFI KMS Implementation



1) UEFI firmware requests key from KMS server
2) KMS server provides key to UEFI firmware
3) UEFI firmware publishes EFI_KMS_PROTOCOL
4) Any UEFI Driver (AHCI, RAID, etc.) consumes the Protocol
5) UEFI Drivers unlock the encrypted drives

Request Delivery

Implementing and Using the UEFI Key Management Service (KMS)

Call to Action

Call to Action

- The UEFI KMS Protocol can be very useful in enterprise and HyperScale environments, especially as drive encryption and other applications for cryptographic keys become more common
- UEFI stakeholders should understand how to implement KMS authentication in their firmware to encourage a higher adoption rate
  - Independent BIOS Vendors (IBVs) should support the UEFI KMS Protocol in their firmware
  - IHVs should implement the UEFI solution by consuming the UEFI KMS Protocol
  - Independent Software Vendors (ISVs) should also look to leverage the UEFI KMS Protocol to further secure their solutions

Questions?

Thanks for attending the UEFI 2020 Virtual Plugfest

For more information on UEFI Forum and UEFI Specifications, visit http://www.uefi.org

presented by