

Technical whitepaper

Secure Boot Customization Guide

July 2017
L01780-001



Disclaimer

The information contained in this document, including URL, other web site references, and other specification documents are subject to change without notice and are provided for informational purposes only. No licenses concerning any intellectual property are being granted, expressly or impliedly, by the disclosure of the information contained in this document. Furthermore, neither Hewlett Packard nor any of its subsidiaries makes any warranties of any nature regarding the use of the information contained in this document, and thus the entire risk, if any, resulting from the use of information within this document is the sole responsibility of the user. Also, the names of the technologies, actual companies, and products mentioned in this document may be trademarks of their respective owners. Complying with all applicable copyright and trademark laws is the sole responsibility of the user of this document. Without limiting any rights under copyright, no part of this document may be reproduced, stored, or transmitted in any form or by any means without the express written consent of HP Development Company, L.P.

HP Development Company, L.P. or its subsidiaries may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering the subject matter in this document. Except where expressly provided in any written license from HP Development Company, L.P. or its subsidiaries, the furnishing of this document, or any ideas contained within, does not grant any license to these ideas, patents, trademarks, copyrights, or other intellectual property.

| Version No. | Revised by | Changes |
|--------------------|----------------------------|---|
| 0.1 | Chris Stewart | Initial Baseline |
| 0.2 | Chris Stewart | Augment PK and KEK import procedures to show sample for self-signed keys. |
| 1.0 | Chris Stewart | Add disclaimer |
| 1.1 | Joe David, Jason Aydelotte | Clarifications and formatting revisions |

Table of contents

| | |
|---|----|
| 1 Introduction..... | 7 |
| 2 Setting up a customized Secure Boot environment | 8 |
| 2.1 Backup existing Secure Boot configuration | 8 |
| 2.2 Place your HP PC in Secure Boot setup mode | 9 |
| 2.3 Obtain PK and KEK public keys | 10 |
| 2.4 Self-signing certificates..... | 10 |
| 2.4.1 Generate a new PK | 11 |
| 2.4.2 Generate a new KEK | 13 |
| 2.5 Install the new PK..... | 13 |
| 2.5.1 PK: Create a valid SetVariable() package..... | 15 |
| 2.5.2 Import PK using Windows tools | 15 |
| 2.6 Install the new PK-signed KEK..... | 16 |
| 2.6.1 KEK: Create a valid SetVariable() package..... | 17 |
| 2.6.2 Import KEK Using Windows Tools | 18 |
| 2.7 Install the New KEK-signed DB and DBX..... | 19 |
| 2.7.1 DB..... | 19 |
| 2.7.2 DBX..... | 22 |
| 2.8 Enable Secure Boot Once More | 24 |
| 2.9 Add Additional Certificates to DB or DBX | 24 |
| 2.9.1 DB..... | 25 |
| 2.9.2 DBX..... | 27 |
| 3 References | 28 |

List of figures

| | | |
|------------------|--|----|
| Figure 1 | Run PowerShell as Administrator | 8 |
| Figure 2 | Sample Get-SecureBootUEFI Commands to backup default Secure Boot configuration ... | 9 |
| Figure 3 | Sample backup of default Secure Boot configuration | 9 |
| Figure 4 | Place HP PC in Secure Boot setup mode | 10 |
| Figure 5 | Sample command line for generation of a self-signed certificate | 11 |
| Figure 6 | Sample output of generation of self-signed certificate | 11 |
| Figure 7 | Sample command line to create a PFX file for signing | 11 |
| Figure 8 | Sample output of creation of PFX file | 12 |
| Figure 9 | Sample command line for conversion of certificate to DER format | 12 |
| Figure 10 | Sample output of successful DER format conversion | 12 |
| Figure 11 | Sample command lines to generate KEK | 13 |
| Figure 12 | Successful PK format | 14 |
| Figure 13 | Command line to create signed PK | 14 |
| Figure 14 | Successful output of properly formatted UEFI variable | 15 |
| Figure 15 | Successful import of PK to Windows | 16 |
| Figure 16 | Successful output of formatted KEK | 17 |
| Figure 17 | Command line switches to sign KEK with PK private key | 17 |
| Figure 18 | Successful creation of SetVariable() package | 18 |
| Figure 19 | Successful import of KEK | 18 |
| Figure 20 | Successful output | 19 |
| Figure 21 | Command line to sign the signature list with private key | 20 |
| Figure 22 | Successful creation of package | 21 |
| Figure 23 | Successful import | 21 |
| Figure 24 | Successful output | 22 |
| Figure 25 | Command line to sign DBX using PFX file | 23 |
| Figure 26 | Successful creation of variable package | 23 |
| Figure 27 | From support.hp.com: How to enable Secure Boot | 24 |
| Figure 28 | Command line to sign signature list for DB | 26 |
| Figure 29 | Successful import | 26 |
| Figure 30 | Successful import | 27 |

List of tables

| | | |
|-----------------|--|----|
| Table 1 | List of switches useful for Format-SecureBootUEFI command to format the Platform Key (PK)..... | 14 |
| Table 2 | Command line switches to create SetVariable() package | 15 |
| Table 3 | Command line switches to import PK to Windows..... | 15 |
| Table 4 | Command line switches to format the KEK | 16 |
| Table 5 | Command line switches to create SetVariable() package for KEK | 17 |
| Table 6 | Command line switches to import KEK..... | 18 |
| Table 7 | Command line switches to create signature list for three default DB..... | 19 |
| Table 8 | Command line switches to create SetVariable() package for DB | 20 |
| Table 9 | Command line switches to import KEK-signed DB..... | 21 |
| Table 10 | Command line switches to format DBX..... | 22 |
| Table 11 | Command line switches to create SetVariable() package | 23 |
| Table 12 | Command line switches to import the KEK-signed DBX..... | 24 |
| Table 13 | Command line switches to format DB key | 25 |
| Table 14 | Successful output with formatted DB key | 25 |
| Table 15 | Command line switches to import KEK-signed certificate | 26 |
| Table 16 | Command line switches to import the KEK-signed DB certificate | 27 |

1 Introduction

This document offers an overview of how to configure Secure Boot in a customized environment, specifically one in which the machine owner claims ownership of the machine by installing his own Secure Boot Platform Key. Doing this requires the platform owner to configure Secure Boot further to allow the machine to boot. This guide makes several assumptions:

1. The default HP Platform Key (PK) will be replaced with a new PK that is exclusively under the control of the platform owner.
2. The default HP Key Exchange Key (KEK) will be replaced with a new KEK that has been signed with the PK mentioned in #1, above.
3. The default Signature Database (DB) will be modified in such a way that all database entries are imported because they have been signed with the platform owner's KEK mentioned in #2, above. The default DB may or may not be included, but if it *does* include the default DB, then the default DB will be exported and re-signed with the platform owner's KEK before being imported again into the DB. Then any additional keys to place into the DB will also be signed with the platform owner's KEK.
4. The default Forbidden Signature Database (DBX) will be modified in such a way that all database entries are imported because they have been signed with the platform owner's KEK mentioned in #2, above. The default DBX may or may not be included, but if it *does* include the default DBX, then the default DBX will be exported and re-signed with the platform owner's KEK before being imported again into the DBX. Then any additional keys to place into the DBX will also be signed with the platform owner's KEK.

This document assumes the reader is familiar with Secure Boot architecture. For a good overview, please reference Microsoft's [Windows 8.1 Secure Boot Key Creation and Management Guidance](#).

2 Setting up a customized Secure Boot environment

2.1 Backup existing Secure Boot configuration

The first step is to back up the default PK, KEK, DB, and DBX. Partly, this is intended as a failsafe because the ultimate protection against loss of access to a Secure Boot environment is to have a backup copy of the default configuration¹. Mostly, however, this is required so that the default DB and DBX can be re-signed and reimported after the PK and KEK are updated if this is the desire of the system administrator.

It is necessary to run PowerShell as Administrator to back up the existing Secure Boot configuration. From the Windows 8.1 or Windows 10 Start screen, press the Windows key. Then start typing PowerShell. Choose Windows PowerShell ISE from the list, right-click on it, and choose Run as administrator.

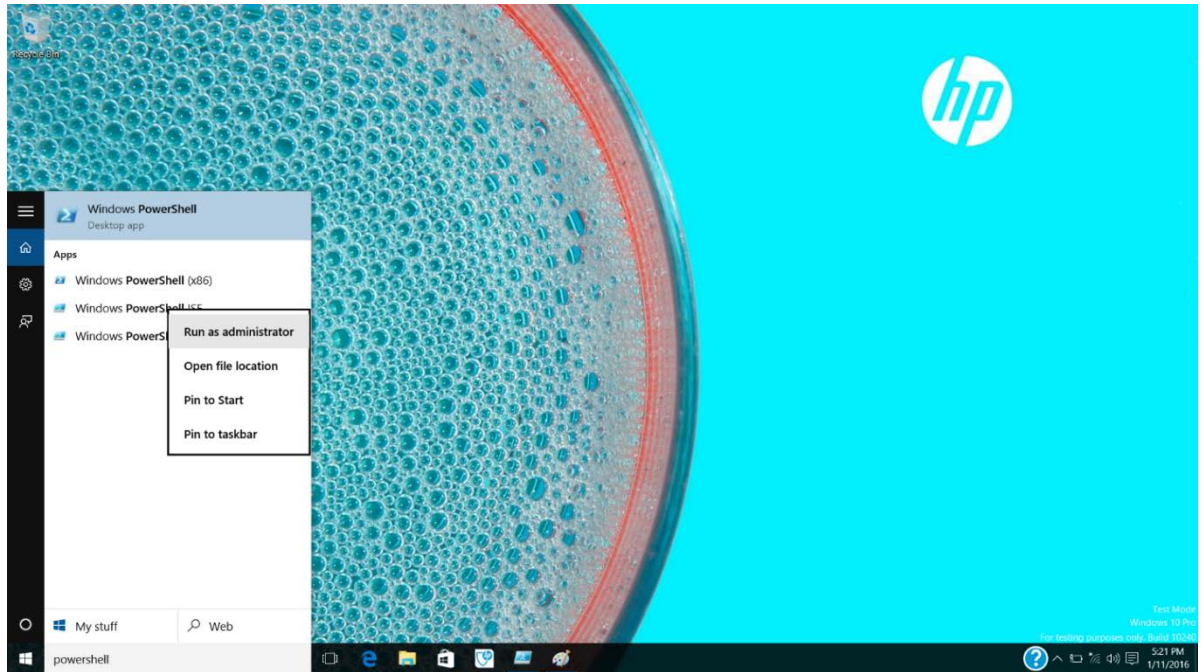


Figure 1 Run PowerShell as Administrator

Now, from the PowerShell command line, back up the PK, KEK, DB, and DBX, each, in turn, using the [Get-SecureBootUEFI](#) command. In these examples, each Secure Boot configuration setting is backed up to an individual file in the Secure Boot directory of an attached USB Key, configured here as drive F:

¹ This is not strictly necessary on HP platforms, because HP platforms can recover the default Secure Boot configuration and re-apply by accessing the proper configuration settings via F10 setup. The procedure for restoring the default Secure Boot configuration on an HP platform is provided as an appendix to this guide.


```
Get-SecureBootUEFI -Name PK -OutputFilePath F:\SecureBoot\HpPk.BAK

Get-SecureBootUEFI -Name KEK -OutputFilePath F:\SecureBoot\HpKek.BAK

Get-SecureBootUEFI -Name DB -OutputFilePath F:\SecureBoot\HpDb.BAK

Get-SecureBootUEFI -Name DBX -OutputFilePath F:\SecureBoot\HpDbx.BAK
```

Figure 2 Sample `Get-SecureBootUEFI` Commands to backup default Secure Boot configuration

Your output should look something like the following:

```
PS C:\windows\system32> Get-SecureBootUEFI -Name PK -OutputFilePath F:\SecureBoot\HpPk.BAK
Name Bytes          Attributes
----
PK  {161, 89, 192, 165...} NON VOLATILE...

PS C:\windows\system32> Get-SecureBootUEFI -Name KEK -OutputFilePath F:\SecureBoot\HpKek.BAK
Name Bytes          Attributes
----
KEK {161, 89, 192, 165...} NON VOLATILE...

PS C:\windows\system32> Get-SecureBootUEFI -Name DB -OutputFilePath F:\SecureBoot\HpDb.BAK
Name Bytes          Attributes
----
db  {161, 89, 192, 165...} NON VOLATILE...

PS C:\windows\system32> Get-SecureBootUEFI -Name DBX -OutputFilePath F:\SecureBoot\HpDbx.BAK
Name Bytes          Attributes
----
dbx {38, 22, 196, 193...} NON VOLATILE...
```

Figure 3 Sample backup of default Secure Boot configuration

Once this is complete, you have a back-up of the four Secure Boot configuration settings. This document will assume that these backup files are the following, but you may, of course, use any file names you like:

- PK backup: HpPk.BAK
- KEK backup: HpKek.BAK
- DB backup: HpDb.BAK
- DBX backup: HpDbx.BAK

2.2 Place your HP PC in Secure Boot setup mode

Next, it is necessary to place your HP PC into Secure Boot Setup Mode. To accomplish this, reboot your machine. At the HP logo screen during boot-up, press **F10** to enter **F10** setup. At the **F10** Main screen, select the **Advanced** tab. Then select Secure Boot Configuration. Ensure that the Configure Legacy Support and Secure Boot option is set to **Legacy Support Disable and Secure Boot Disable**. If needed, set the configuration accordingly and choose **F10** to Save and Exit. Then enter

F10 setup again and return to the [Advanced...Secure Boot Configuration](#) screen. Finally, check the box next to the Clear Secure Boot keys option and press F10 to [Save and Exit](#).

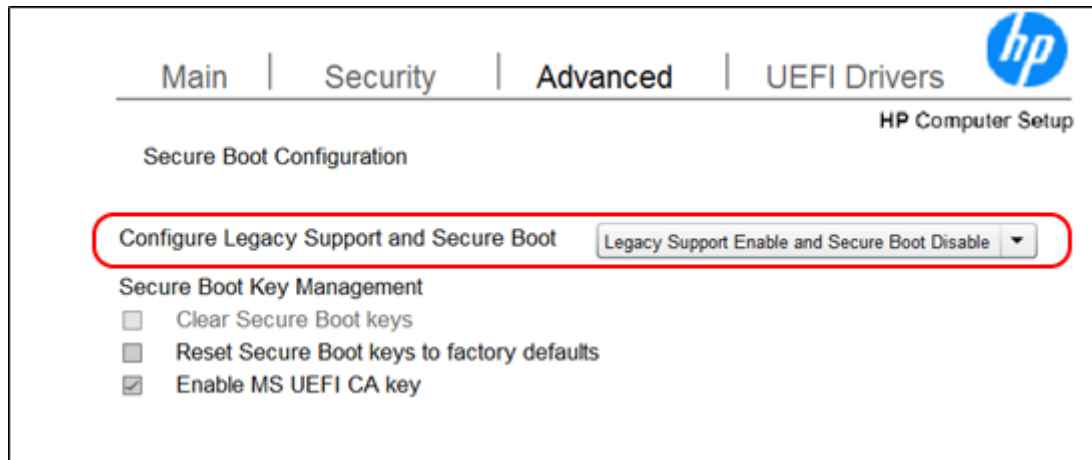


Figure 4 Place HP PC in Secure Boot setup mode

When the machine reboots, allow it to boot to the Windows desktop. From here, it is possible to install your PK, KEK, DB, and DBX configuration settings.

2.3 Obtain PK and KEK public keys

In a typical production environment, a *Hardware Security Module* (HSM) constrains the ability to sign with the PK and KEK private keys. An HSM is a physical device that stores private keys and from which it is difficult or impossible to steal the private keys independent of the HSM physical device itself. There are several HSM vendors in the market providing solutions of greater or lesser security; analyzing the specifics of these vendors is beyond the scope of this document. However, it is important to note that HP recommends that any infrastructure deployed for digital signing be secured using an HSM device. Microsoft publishes a good guide on entitled, [Secure Boot Key Generation and Signing Using HSM](#), which goes over this process in greater detail.

It is necessary to obtain both the PK and KEK *public* keys from your security administrator to continue with a customized configuration for Secure Boot. These keys are *required* to be in [DER format](#), and in this case, it is assumed that they have each been saved to a separate file with a CER extension. Assuming you have PK.cer, which is your new Platform Key, and KEK.cer, which is your new Key Exchange Key, submit your KEK to your HSM key signing process to be signed by PK's private key. Once complete, you are ready for the next step.

2.4 Self-signing certificates

Just for the sake of completeness, this document shows how to use OpenSSL and Microsoft tools to generate a set of self-signing keys. Such an approach does not provide an adequately secure system, but it is helpful for illustration. The References section at the end of this document contains URL links that can direct you where to obtain OpenSSL and Microsoft tools for this purpose.

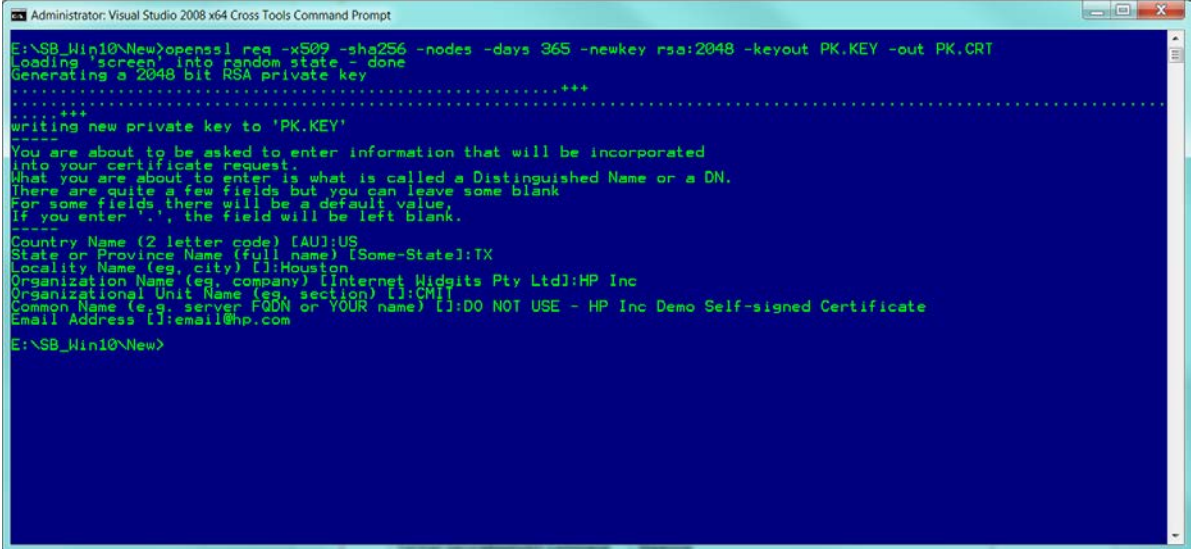
2.4.1 Generate a new PK

First, generate a self-signed certificate.

```
openssl req -x509 -sha256 -nodes -days 365 -newkey rsa:2048 -keyout PK.KEY -out PK.CRT
```

Figure 5 Sample command line for generation of a self-signed certificate

You will have to answer several questions. Sample output follows:



```
Administrator: Visual Studio 2008 x64 Cross Tools Command Prompt
E:\SB_Hin10\New>openssl req -x509 -sha256 -nodes -days 365 -newkey rsa:2048 -keyout PK.KEY -out PK.CRT
Loading 'screen' into random state - done
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to 'PK.KEY'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:TX
Locality Name (eg, city) []:Houston
Organization Name (eg, company) [Internet Widgits Pty Ltd]:HP Inc
Organizational Unit Name (eg, section) []:CMI
Common Name (e.g. server FQDN or YOUR name) []:DO NOT USE - HP Inc Demo Self-signed Certificate
Email Address []:email@hp.com
E:\SB_Hin10\New>
```

Figure 6 Sample output of generation of self-signed certificate

The command generates both a KEY file and a CRT file. To sign with a self-signing certificate, you will also need to create a PFX file:

```
openssl pkcs12 -export -out PK.PFX -inkey PK.KEY -in PK.CRT
```

Figure 7 Sample command line to create a PFX file for signing

In response to this command, you will have to provide an export password. If the command is successful, you will see output similar to the following:

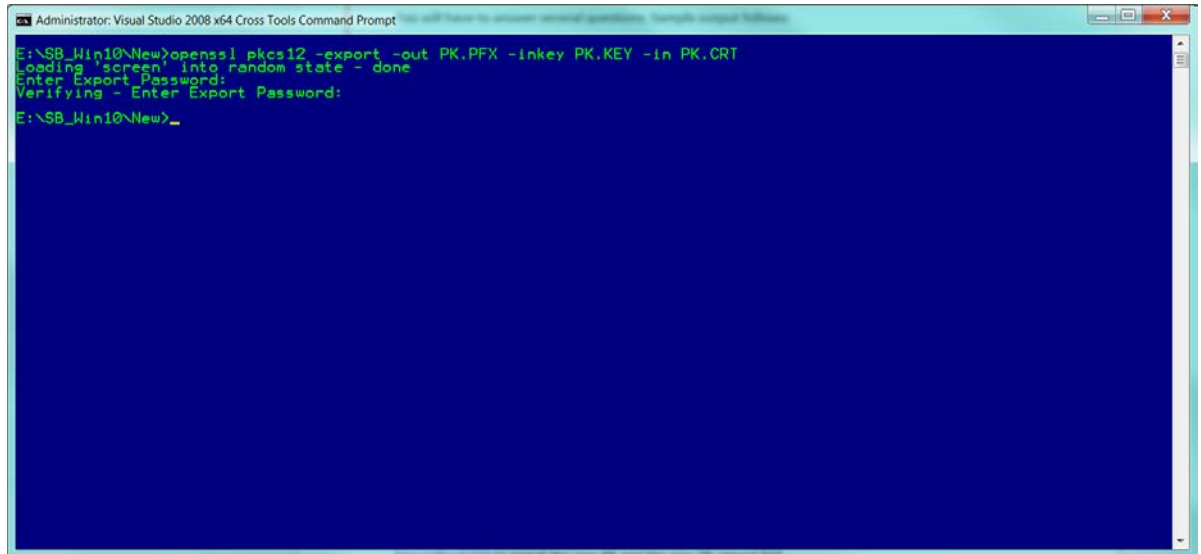


Figure 8 Sample output of creation of PFX file

At this point, you have the following files: PK.CRT, PK.KEY, and PK.PFX. The certificate you need to use is PK.CRT, but it is in a base64-encoded format. For Secure Boot, this certificate must be in DER format. Thus, you must convert it to DER format:

```
openssl x509 -outform der -in PK.CRT -out PK.CER
```

Figure 9 Sample command line for conversion of certificate to DER format

If successful, you will see output similar to the following:

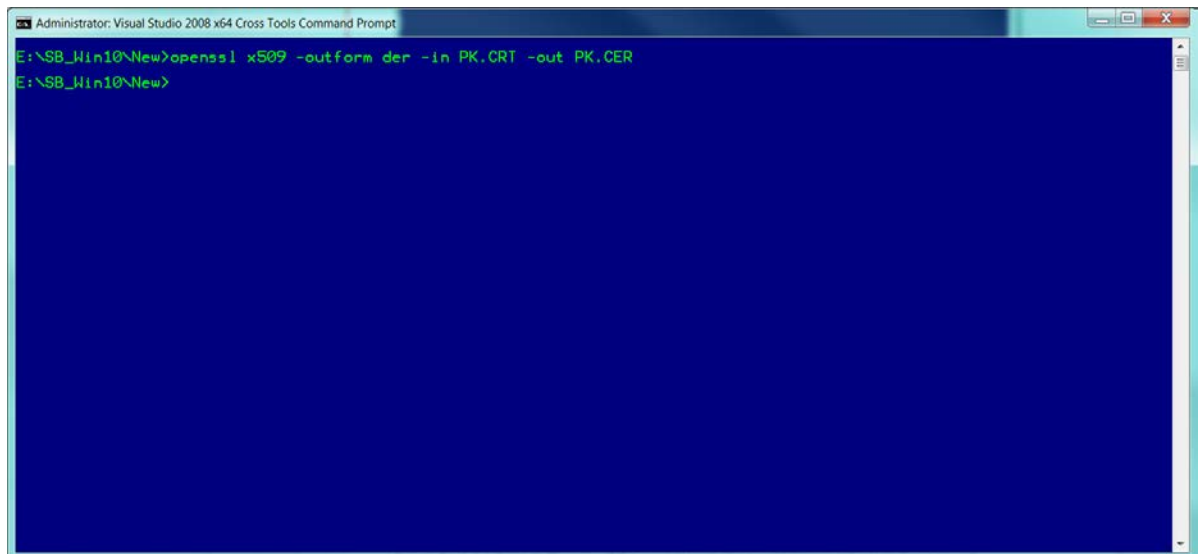


Figure 10 Sample output of successful DER format conversion

PK.CER is the DER-encoded x509 certificate that you can use for the Secure Boot customization. Again, this is for illustrative purposes only. A real-world deployment would obtain this DER-encoded certificate from an HSM.

2.4.2 Generate a new KEK

To generate a new self-signed KEK, follow the steps in the immediately preceding section, but replace all filename references to PK with filename references to KEK. You now have two sets of self-signed keys and certificates which can be used for Secure Boot customization. We reproduce the commands here but not the command output:

```
Create certificate:  
openssl req -x509 -sha256 -nodes -days 365 -newkey rsa:2048 -keyout KEK.KEY -out  
KEK.CRT  
  
Export private key into PFX file:  
openssl pkcs12 -export -out KEK.PFX -inkey KEK.KEY -in KEK.CRT  
  
Save certificate in DER format:  
openssl x509 -outform der -in KEK.CRT -out KEK.CER
```

Figure 11 Sample command lines to generate KEK

2.5 Install the new PK

IMPORTANT! It is critical that all time values in the following steps be identical and that they fall within the valid date/time ranges of your custom certificates. If this is not the case, the process will fail.

Again, launch PowerShell as administrator. This time, you will use [Format-SecureBootUEFI](#) and [Set-SecureBootUEFI](#) to install the new PK and the new PK-signed KEK.

NOTE: Secure Boot is not enabled on your system by default. It is important that Secure Boot is not enabled until a Signature Database is installed on the system. If you plan to use the default Windows signatures with Secure Boot, you may wish to sign the default Secure Boot Signature Database that was previously backed up. More on this later.

Once you have launched PowerShell as administrator, you are ready to import your new PK into the Secure Boot database. In the case of the current example, my signing procedure produces signed packages with a P7 extension. Thus, the PK-signed KEK file is KEK.CER.P7; and the signed DB and DBX files (described later) will also have a P7 extension. Depending on your process and HSM provider, your signed files might have a different extension from this, or they might have the same extension as the unsigned file. The PK is the root of trust and is not signed because nothing more trusted exists with which to sign it. After importing the PK into the platform, all other artifacts must be signed by the appropriate private key using your HSM solution.

For step 1, you must obtain your PK public key certificate in DER format. In this example, we use our own PK.CER as the public key certificate. You should obtain this certificate from your HSM provider. First, the PK key must be formatted using the [Format-SecureBootUEFI](#) command inside Windows PowerShell.

| Format-SecureBootUEFI Command Line Parameter | Meaning |
|---|---|
| -Name PK | Indicates that you are working with the Platform Key (PK) |
| -SignatureOwner DEF16466-F946-4E71-BE22-CF8B1B7B36A0 | The hexadecimal number is a GUID that uniquely identifies you to the platform. You can generate a GUID using the Microsoft GuidGen.exe tool, among other means. |
| -ContentFilePath .\PK_SigList.bin | This file is created to hold the content that is generated by Format-SecureBootUEFI, i.e. the formatted content. |
| -FormatWithCert | Tells Format-SecureBootUEFI to integrate the entire certificate into the formatted content. |
| -Certificate .\PK.CER | Indicates the path to the desired certificate, in this case, the PK certificate. |
| -SignableFilePath .\PK_SigList_Serialization_for_PK.bin | Specifies the file that should be signed after formatting. |
| -Time 2016-02-01T13:30:00Z | Specifies the current date and time, which must be specified. |

Table 1 List of switches useful for [Format-SecureBootUEFI](#) command to format the Platform Key (PK)

If successful, the command should produce output similar to the following:

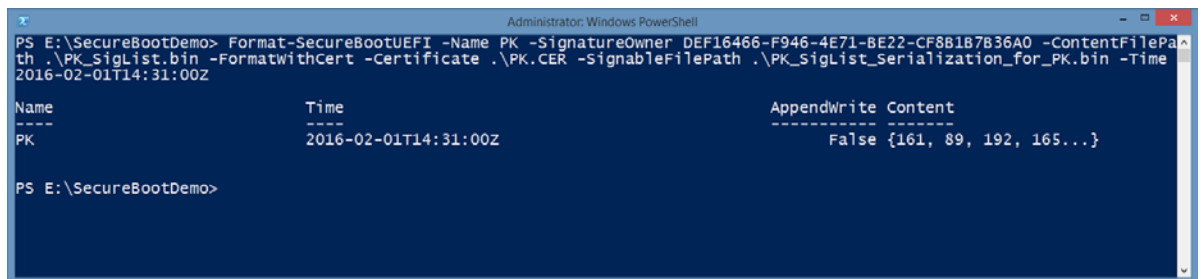


Figure 12 Successful PK format

It is the SignableFilePath file, in this case, PK_SigList_Serialization_for_PK.bin, which is submitted to your HSM solution for signing. This file should be signed using the private key for your new PK. A proper signing command for PK_SigList_Serialization_for_PK.bin, if using a PFX file², is as follows. In this case, [signtool](#) must be in your path:

```

signtool.exe sign /fd sha256 /p7 .\ /p7co 1.2.840.113549.1.7.1 /p7ce DetachedSignedData /a /f .\PK.PFX /p <password> PK_SigList_Serialization_for_PK.bin
    
```

Figure 13 Command line to create signed PK

Replace <password> with the actual private key password for your PFX file. The result of the above command is a signed PK serialized into a file called PK_SigList_Serialization_for_PK.bin.p7. You should, of course, use the signed file that provided by your HSM provider.

Once you have the signed PK, it is ready for import to your platform. Importing is done with the [Set-SecureBootUEFI](#) command inside Windows PowerShell. There are two steps possible here. The first step simply creates a valid time-authenticated variable package which could be imported using a simple UEFI [SetVariable\(\)](#) command. This package is then saved to a file called PK_NewKey_Import_PK.bin. This step is worth running even if you plan to use Windows tools to import

² This would be the approach if you used self-signing certificates, but it is strongly recommended that you perform the same action in the context of your own HSM provider.

the new PK onto your platform. The second step uses Windows tools to write the new PK directly to your platform BIOS storage repository.

2.5.1 PK: Create a valid SetVariable() package

| Set-SecureBootUEFI Command Line Parameter | Meaning |
|--|---|
| -Name PK | Indicates that you are working with the Platform Key (PK) |
| -Time 2016-02-01T13:30:00Z | Specifies the current date and time, which must be specified. |
| -ContentFilePath .\PK_SigList.bin | Specifies the name of the file which contains the unsigned, formatted PK. |
| -SignedFilePath .\PK_SigList_Serialization_for_PK.bin.p7 | Specifies the name of the file which contains the signed, formatted PK. |
| -OutputFilePath .\PK_NewKey_Import_PK.bin | Specifies the file which will contain the output of the command upon successful completion. |

Table 2 Command line switches to create SetVariable() package

If successful, the command should produce output similar to the following:

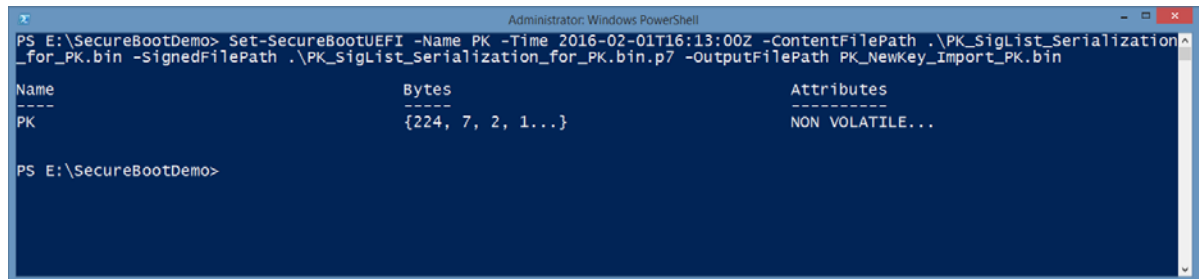


Figure 14 Successful output of properly formatted UEFI variable

This step has produced a properly-formatted UEFI time-authenticated variable which may be used for direct import into Secure Boot using a simple UEFI `SetVariable()` command. The file `PK_NewKey_Import_PK.bin` contains this properly-formatted UEFI time-authenticated variable artifact.

2.5.2 Import PK using Windows tools

There is one more step required to use the Windows tools to import the KEK: writing the KEK itself to Non-volatile Random Access Memory (NVRAM). Use the `Set-SecureBootUEFI` command inside Windows PowerShell for this purpose

| Set-SecureBootUEFI Command Line Parameter | Meaning |
|--|---|
| -Name PK | Indicates that you are working with the Platform Key (PK) |
| -Time 2016-02-01T13:30:00Z | Specifies the current date and time, which must be specified. |
| -ContentFilePath .\PK_SigList.bin | Specifies the name of the file which contains the <i>unsigned</i> , unformatted PK, created in a previous step. |
| -SignedFilePath .\PK_SigList_Serialization_for_PK.bin.p7 | Specifies the name of the file which contains the <i>signed</i> , formatted PK. |

Table 3 Command line switches to import PK to Windows

If successful, the command should produce output similar to the following:

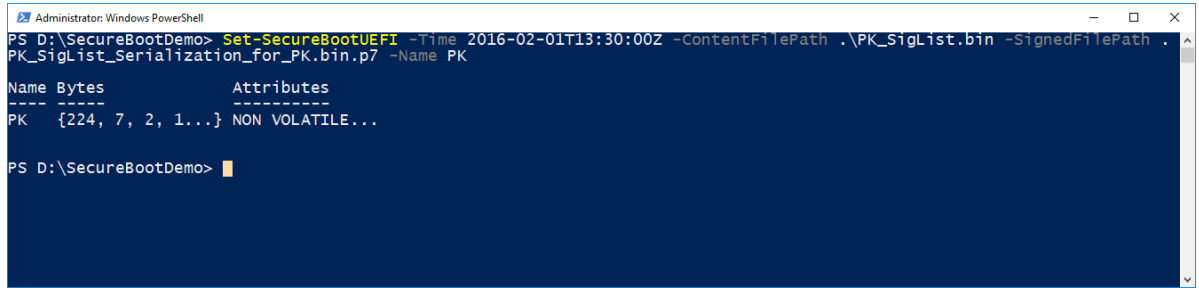


Figure 15 Successful import of PK to Windows

This command imports the PK into your system.

2.6 Install the new PK-signed KEK

Launch PowerShell as Administrator. Again, the [Format-SecureBootUEFI](#) and [Set-SecureBootUEFI](#) commands are used to install the new PK-signed KEK.

Obtain your KEK public key as a DER-encoded certificate file. You should obtain this certificate from your HSM provider. In this case, I assume that the KEK filename is KEK.CER. The KEK key must be formatted using the [Format-SecureBootUEFI](#) command inside Windows PowerShell before being imported.

| Format-SecureBootUEFI Command Line Parameter | Meaning |
|---|---|
| -Name KEK | Indicates that you are working with the Key Exchange Key (KEK) |
| -SignatureOwner DEF16466-F946-4E71-BE22-CF8B1B7B36A0 | The hexadecimal number is a GUID that uniquely identifies you to the platform. Since this represents the signature owner, it should be the same GUID used to import the PK. |
| -ContentFilePath .\KEK_SigList.bin | This file is created to hold the content that is generated by Format-SecureBootUEFI, i.e. the formatted content. |
| -FormatWithCert | Tells Format-SecureBootUEFI to integrate the entire certificate into the formatted content. |
| -Certificate .\KEK.CER | Indicates the path to the desired certificate, in this case, the KEK certificate. |
| -SignableFilePath .\KEK_SigList_Serialization_for_KEK.bin | Specifies the file that should be signed after formatting. |
| -Time 2016-02-01T13:30:00Z | Specifies the current date and time, which must be specified. |

Table 4 Command line switches to format the KEK

If successful, the command should produce output similar to the following:

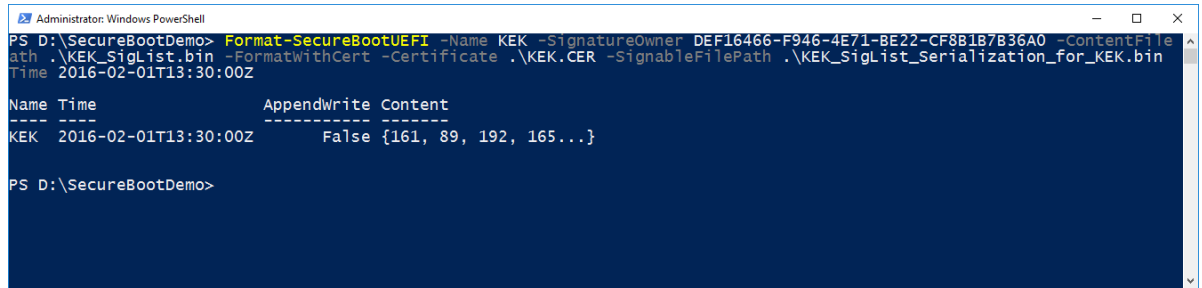


Figure 16 Successful output of formatted KEK

It is the [SignableFilePath](#) file, in this case, [KEK_SigList_Serialization_for_KEK.bin](#), which is submitted to your HSM solution for signing. This file should be signed using the private key for your new PK. A proper signing command for [KEK_SigList_Serialization_for_KEK.bin](#), if using a PFX file³, is as follows. In this case, [sintool](#) must be in your path:

```
sintool.exe sign /fd sha256 /p7 .\ /p7co 1.2.840.113549.1.7.1 /p7ce DetachedSignedData /a /f .\PK.PFX /p <password> KEK_SigList_Serialization_for_KEK.bin
```

Figure 17 Command line switches to sign KEK with PK private key

Replace <password> with the actual private key password for your PFX file. The result of the above command is a PK-signed KEK serialized into a file called [KEK_SigList_Serialization_for_KEK.bin.p7](#). Use the signed file that is provided by your HSM provider.

Once you have the PK-signed KEK, it is ready for import to your platform. Importing is done with the [Set-SecureBootUEFI](#) command inside Windows PowerShell. Again, there are two steps possible here. The first step simply creates a valid time-authenticated variable package which could be imported using a simple UEFI SetVariable() command. This package is then saved to a file called [KEK_NewKey_Import_KEK.bin](#). This step is worth running even if you plan to use Windows tools to import the new PK onto your platform. The second step uses Windows tools to write the new PK directly to your platform BIOS storage repository.

2.6.1 KEK: Create a valid SetVariable() package

| Set-SecureBootUEFI Command Line Parameter | Meaning |
|--|---|
| -Name KEK | Indicates that you are working with the Platform Key (PK) |
| -Time 2016-02-01T13:30:00Z | Specifies the current date and time, which must be specified. |
| -ContentFilePath .\KEK_SigList.bin | Specifies the name of the file which contains the <i>unsigned</i> , formatted KEK. |
| -SignedFilePath .\KEK_SigList_Serialization_for_KEK.bin.p7 | Specifies the name of the file which contains the <i>signed</i> , formatted KEK. |
| -OutputFilePath .\KEK_NewKey_Import_KEK.bin | Specifies the file which will contain the output of the command upon successful completion. |

Table 5 Command line switches to create SetVariable() package for KEK

³ This would be the approach if you used self-signing certificates, but it is strongly recommended that you perform the same action in the context of your own HSM provider

If successful, the command should produce output similar to the following:

```

Administrator: Windows PowerShell
PS D:\SecureBootDemo> Set-SecureBootUEFI -Name KEK -Time 2016-02-01T13:30:00Z -ContentFilePath .\KEK_SigList_Serialization_for_KEK.bin -SignedFilePath .\KEK_SigList_Serialization_for_KEK.bin.p7 -OutputFilePath .\KEK_NewKey_Import_KEK.bin
Name Bytes Attributes
----
KEK {224, 7, 2, 1...} NON VOLATILE...
PS D:\SecureBootDemo>
    
```

Figure 18 Successful creation of SetVariable() package

2.6.2 Import KEK Using Windows Tools

There is one more step required to use the Windows tools to import the KEK: writing the KEK itself to Non-volatile Random Access Memory (NVRAM). Use the [Set-SecureBootUEFI](#) command inside Windows PowerShell for this purpose.

| Set-SecureBootUEFI Command Line Parameter | Meaning |
|--|--|
| -Name KEK | Indicates that you are working with the Key Exchange Key (KEK) |
| -Time 2016-02-01T13:30:00Z | Specifies the current date and time, which must be specified. |
| -ContentFilePath .\KEK_SigList.bin | Specifies the name of the file which contains the unsigned, unformatted KEK, created in a previous step. |
| -SignedFilePath .\KEK_SigList_Serialization_for_KEK.bin.p7 | Specifies the name of the file which contains the signed, formatted KEK. |

Table 6 Command line switches to import KEK

If successful, the command should produce output similar to the following:

```

Administrator: Windows PowerShell
PS D:\SecureBootDemo> Set-SecureBootUEFI -Name KEK -Time 2016-02-01T13:30:00Z -ContentFilePath .\KEK_SigList.bin -SignedFilePath .\KEK_SigList_Serialization_for_KEK.bin.p7
Name Bytes Attributes
----
KEK {224, 7, 2, 1...} NON VOLATILE...
PS D:\SecureBootDemo>
PS D:\SecureBootDemo>
    
```

Figure 19 Successful import of KEK

At this point, both the new PK and the new PK-signed KEK are on the system, and you officially own the platform. The next step is to import your DB and DBX files, each of which must be signed by the KEK. For purposes of this example, we shall simply sign the DB and DBX files previously backed-up as [HpDb.BAK](#) and [HpDbx.BAK](#).

2.7 Install the New KEK-signed DB and DBX

2.7.1 DB

The simplest way to get to the default HP DB configuration using the Windows command line tools is to create a Signature List serialization file using the three HP default certificates. If you wish to omit one or more or exclusively use your DER-encoded certificates, you can do that, of course. The following command parameters create a signable signature list file using the three default HP DB certificates, embedded in this document. The command parameters below assume that you have unpacked the three DB certificates into your local working directory.



HpDb.zip

The first step uses the [Format-SecureBootUEFI command](#).

| Format-SecureBootUEFI Command Line Parameter | Meaning |
|--|---|
| -Name DB | Indicates that you are working with the Secure Boot DB. |
| -Time 2016-02-01T13:30:00Z | Specifies the current date and time, which must be specified. |
| -SignatureOwner DEF16466-F946-4E71-BE22-CF8B1B7B36A0 | The hexadecimal number is a GUID that uniquely identifies you to the platform. Since this represents the signature owner, it should be the same GUID used to import the PK. |
| -ContentFilePath .\HpDb_SigList.bin | Specifies the name of the file which contains the <i>unsigned</i> , unformatted collection of DB certificates. |
| -CertificateFilePath .\HPDB2013.der, .\MsUEFCA2011_2011-06-27.cer, .\MsWinProDb2011_2011-10-19.cer | These are the three HP default DER-encoded certificate files. Each filename is separated by a comma (,) on the command line. |
| -FormatWithCert | Tells Format-SecureBootUEFI to integrate the entire certificate into the formatted content. |
| -SignableFilePath .\HpDb_SigList_Serialization_for_DB.bin | Specifies the file that should be signed after formatting. |

Table 7 Command line switches to create signature list for three default DB

If the command succeeds, you should see something like the following.

```

Administrator: Windows PowerShell
PS D:\SecureBootDemo> Format-SecureBootUEFI -Name DB -Time 2016-02-01T13:30:00Z -SignatureOwner DEF16466-F946-4E71-BE22-CF8B1B7B36A0 -ContentFilePath HpDb_SigList.bin -CertificateFilePath .\HpDb\HPDB2013.der, .\HpDb\MsUEFCA2011_2011-06-27.cer, .\HpDb\MsWinProDb2011_2011-10-19.cer -FormatWithCert -SignableFilePath HpDb_SigList_Serialization_for_DB.bin
Name Time Appendwrite Content
----
db 2016-02-01T13:30:00Z False {161, 89, 192, 165...}

PS D:\SecureBootDemo>
    
```

Figure 20 Successful output

Again, the file to submit to your HSM solution for signing is the signature list serialization file, in this case, [HpDb_SigList_Serialization_for_DB.bin](#). This file should be signed using the private key for your new KEK. A proper signing command for [HpDb_SigList_Serialization_for_DB.bin](#), if using a PFX file⁴, is as follows. In this case, [signtool](#) must be in your path:

```
signtool.exe sign /fd sha256 /p7 .\ /p7co 1.2.840.113549.1.7.1 /p7ce DetachedSignedData /a /f .\KEK.PFX /p <password> HpDb_SigList_Serialization_for_DB.bin
```

Figure 21 Command line to sign the signature list with private key

Replace <password> with the actual private key password for your PFX file. The result of the above command is a KEK-signed DB serialized into a file [HpDb_SigList_Serialization_for_DB.bin.p7](#). Use the signed file that is provided by your HSM provider.

Once you have the KEK-signed DB, it is ready for import to your platform. Importing is done with the [Set-SecureBootUEFI](#) command inside Windows PowerShell. As with the PK and KEK steps previously detailed, we will demonstrate the creation of a valid time-authenticated SetVariable() binary package and demonstrate how to import the newly signed DB into your platform.

2.7.1.1 DB: Create a Valid SetVariable() Package

| Set-SecureBootUEFI Command Line Parameter | Meaning |
|--|--|
| -Name DB | Indicates that you are working with the Secure Boot certificate database (DB). |
| -Time 2016-02-01T13:30:00Z | Specifies the current date and time, which must be specified. |
| -ContentFilePath .\HpDb_SigList.bin | Specifies the name of the file which contains the <i>unsigned</i> , formatted DB, which is the previously backed-up DB in this case. |
| -SignedFilePath .\ HpDb_SigList_Serialization_for_DB.bin.p7 | Specifies the name of the file which contains the <i>signed</i> , formatted DB. |
| -OutputFilePath .\HpDb_Output_for_DB.bin | Specifies the file which will contain the output of the command upon successful completion. |

Table 8 Command line switches to create SetVariable() package for DB

This command creates a valid SetVariable() package for use in direct import to the BIOS using a SetVariable() call, contained in the file called [HpDb_Output_for_DB.bin](#) and is also useful for archival purposes.

⁴ This would be the approach if you used self-signing certificates, but it is strongly recommended that you perform the same action in the context of your own HSM provider.

If successful, the command should produce output similar to the following:

```

Administrator: Windows PowerShell
PS D:\SecureBootDemo> Set-SecureBootUEFI -Name DB -Time 2016-02-01T13:30:00Z -ContentFilePath .\HpDb_SigList.bin -SignedFilePath .\HpDb_SigList_Serialization_for_DB.bin.p7 -OutputFilePath .\HpDb_output_for_DB.bin
Name Bytes Attributes
----
db {224, 7, 2, 1...} NON VOLATILE...

PS D:\SecureBootDemo>
    
```

Figure 22 Successful creation of package

2.7.1.2 Import KEK-Signed DB Using Windows Tools

There is one more step required to use the Windows tools to import the KEK: writing the KEK itself to Non-volatile Random Access Memory (NVRAM). Use the [Set-SecureBootUEFI](#) command inside Windows PowerShell for this purpose

| Set-SecureBootUEFI Command Line Parameter | Meaning |
|--|---|
| -Name DB | Indicates that you are working with the Secure Boot certificate database (DB). |
| -Time 2016-02-01T13:30:00Z | Specifies the current date and time, which must be specified. |
| -ContentFilePath .\HpDb_SigList.bin | Specifies the name of the file which contains the <i>unsigned</i> , unformatted DB, created in a previous step. |
| -SignedFilePath .\HpDb_SigList_Serialization_for_DB.bin.p7 | Specifies the name of the file which contains the <i>signed</i> , formatted DB. Signed in the previous step. |

Table 9 Command line switches to import KEK-signed DB

If successful, the command should produce output similar to the following:

```

Administrator: Windows PowerShell
PS D:\SecureBootDemo> Set-SecureBootUEFI -Name DB -Time 2016-02-01T13:30:00Z -ContentFilePath .\HpDb_SigList.bin -SignedFilePath .\HpDb_SigList_Serialization_for_DB.bin.p7
Name Bytes Attributes
----
db {224, 7, 2, 1...} NON VOLATILE...

PS D:\SecureBootDemo>
PS D:\SecureBootDemo>
    
```

Figure 23 Successful import

At this point, you have a fully functional Secure Boot configuration, installed into NVRAM using your custom PK and KEK, which have been used to extend trust to the default set of HP DB certificates. There is no DBX available as yet, but you may now enable Secure Boot. To add a DBX to your implementation, read on.

2.7.2 DBX

To import the previously backed-up DBX file, you should follow the same procedure as for the DB, except that you need to substitute the desired proscribed certificates for DBX. The default set of HP-proscribed certificates is provided here as an attachment, immediately following.



HP_Default_DBX_Certificates.zip

Again, these certificates need to be formatted properly using the [Format-SecureBootUEFI](#) command.

| Format-SecureBootUEFI Command Line Parameter | Meaning |
|---|---|
| -Name DBX | Indicates that you are working with the Secure Boot DBX. |
| -Time 2016-02-01T13:30:00Z | Specifies the current date and time, which must be specified. |
| -SignatureOwner DEF16466-F946-4E71-BE22-CF8B1B7B36A0 | The hexadecimal number is a GUID that uniquely identifies you to the platform. Since this represents the signature owner, it should be the same GUID used to import the PK. |
| -ContentFilePath .\HpDbx_SigList.bin | Specifies the name of the file which contains the <i>unsigned</i> , unformatted collection of DB certificates. |
| -CertificateFilePath .\ HP_DBX_Default1.DER, .\ HP_DBX_Default2.DER | These are the two HP default DER-encoded certificate files. Each filename is separated by a comma (,) on the command line. |
| -FormatWithCert | Tells Format-SecureBootUEFI to integrate the entire certificate into the formatted content. |
| -SignableFilePath .\ HpDbx_SigList_Serialization_for_DBX.bin | Specifies the file that should be signed after formatting. |

Table 10 Command line switches to format DBX

If the command succeeds, you should see something like the following.

```

Administrator: Windows PowerShell
PS D:\SecureBootDemo> Format-SecureBootUEFI -Name DBX -Time 2016-02-01T13:30:00Z -SignatureOwner DEF16466-F946-4E71-BE22-CF8B1B7B36A0 -ContentFilePath HpDbx_SigList.bin -CertificateFilePath .\HP_DBX_Default1.DER, .\HP_DBX_Default2.DER -FormatWithCert -SignableFilePath .\HpDbx_SigList_Serialization_for_DBX.bin
Name Time Appendwrite Content
----
dbx 2016-02-01T13:30:00Z False {161, 89, 192, 165...}
PS D:\SecureBootDemo>
    
```

Figure 24 Successful output

Here, the file to submit to your HSM solution for signing is the signature list serialization file, in this case, [HpDbx_SigList_Serialization_for_DBX.bin](#). This file should be signed using the private key for your new KEK. A proper

signing command for [HpDbx_SigList_Serialization_for_DBX.bin](#), if using a PFX file⁵, is as follows. In this case, [signtool](#) must be in your path:

```
signtool.exe sign /fd sha256 /p7 .\ /p7co 1.2.840.113549.1.7.1 /p7ce DetachedSignedData /a /f .\KEK.PFX /p <password> HpDbx_SigList_Serialization_for_DBX.bin
```

Figure 25 Command line to sign DBX using PFX file

Replace <password> with the actual private key password for your PFX file. The result of the above command is a KEK-signed DB serialized into a file [HpDbx_SigList_Serialization_for_DBX.bin.p7](#). Use the signed file that is provided by your HSM provider.

Once you have the KEK-signed DBX, it is ready for import to your platform. Importing is done with the [Set-SecureBootUEFI](#) command inside Windows PowerShell. As with the PK and KEK steps previously detailed, we will demonstrate the creation of a valid time-authenticated SetVariable() binary package and demonstrate how to import the newly signed DBX into your platform.

2.7.2.1 DBX: Create a Valid SetVariable() Package

| Set-SecureBootUEFI Command Line Parameter | Meaning |
|--|--|
| -Name DBX | Indicates that you are working with the Secure Boot DBX. |
| -Time 2016-02-01T13:30:00Z | Specifies the current date and time, which must be specified. |
| -ContentFilePath .\HpDbx_SigList.bin | Specifies the name of the file which contains the <i>unsigned</i> , formatted DBX, which is the previously backed-up DBX in this case. |
| -SignedFilePath .\ HpDbx_SigList_Serialization_for_DBX.bin.p7 | Specifies the name of the file which contains the <i>signed</i> , formatted DBX. |
| -OutputFilePath .\HpDbx_Output_for_DBX.bin | Specifies the file which will contain the output of the command upon successful completion. |

Table 11 Command line switches to create SetVariable() package

This command creates a valid SetVariable() package for use in direct import to the BIOS using a SetVariable() call, contained in the file called [HpDbx_Output_for_DBX.bin](#) and is also useful for archival purposes.

If successful, the command should produce output similar to the following:

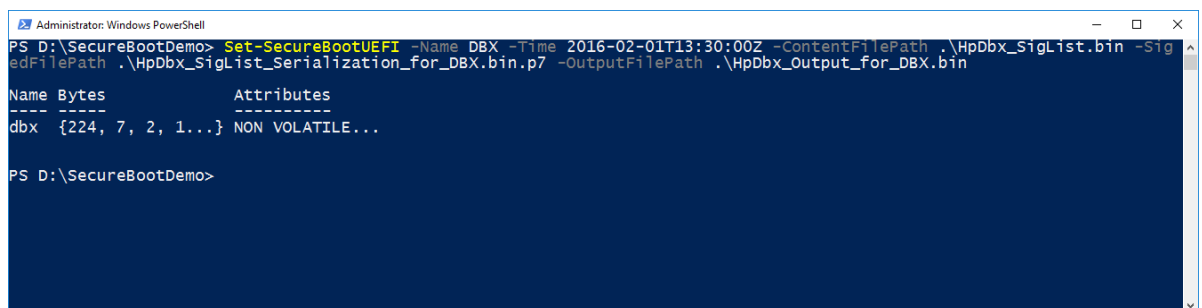


Figure 26 Successful creation of variable package

⁵ This would be the approach if you used self-signing certificates, but it is strongly recommended that you perform the same action in the context of your own HSM provider.

2.7.2.2 Import KEK-Signed DBX Using Windows Tools

There is one more step required to use the Windows tools to import the KEK: writing the KEK itself to Non-volatile Random Access Memory (NVRAM). Use the [Set-SecureBootUEFI](#) command inside Windows PowerShell for this purpose

| Set-SecureBootUEFI Command Line Parameter | Meaning |
|--|--|
| -Name DBX | Indicates that you are working with the Secure Boot DBX. |
| -Time 2016-02-01T13:30:00Z | Specifies the current date and time, which must be specified. |
| -ContentFilePath .\HpDbx_SigList.bin | Specifies the name of the file which contains the <i>unsigned</i> , unformatted DBX, created in a previous step. |
| -SignedFilePath .\HpDbx_SigList_Serialization_for_DBX.bin.p7 | Specifies the name of the file which contains the <i>signed</i> , formatted DBX. Signed in the previous step. |

Table 12 Command line switches to import the KEK-signed DBX

At this point, you have a Secure Boot configuration whose *functionality* matched the default functionality shipped from HP's factories. However, you are the owner of PK and KEK.

2.8 Enable Secure Boot Once More

The final step is to re-enable Secure Boot. At the HP logo screen during boot-up, press **F10** to enter **F10** setup. At the **F10 Main** screen, select the **Advanced** tab. Then select **Secure Boot Configuration**. Ensure that the **Configure Legacy Support and Secure Boot** option is set to **Legacy Support Disable and Secure Boot Enable**. Then press **F10** again to **Save and Exit**. The unit will now boot in Secure Boot mode, and the newly signed Secure Boot keys will verify the existing Windows boot loader.

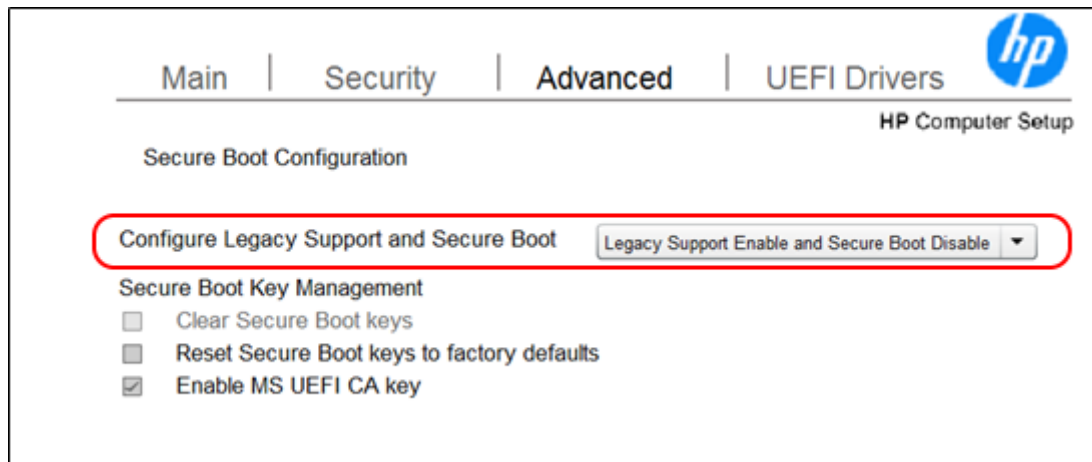


Figure 27 From support.hp.com: How to enable Secure Boot

At this point, the unit works in the same *functional* fashion as when it shipped from the factory. However, the PK and KEK owners are not the default factory-shipped PK and KEK but rather the PK and KEK certificates that *you* have generated. With this complete, you are now the platform owner for the purposes of Secure Boot.

2.9 Add Additional Certificates to DB or DBX

Adding additional certificates is a standard Secure Boot administration task, but it is worth documenting here in some detail. First, you must obtain the public key certificate in DER format that you wish to import into the DB or DBX. In the example shown here, we will import this certificate into the DB, but the procedure is identical (except for the storage location) for the DBX. In the example, the public key certificate was obtained and saved to a file named NewDbCert.CER, which is a public key

certificate in DER format. The first step is to format this certificate for Secure Boot import, using [Format-SecureBootUEFI](#). Once formatted for import, the certificate must be signed by the KEK, using your HSM solution. Finally, the formatted and signed certificate must be imported into the DB (or DBX) using [Set-SecureBootUEFI](#) using the `-Append` flag to avoid overwriting the existing DB.

2.9.1 DB

Obtain your new DB public key as a DER-encoded certificate file. You should obtain this certificate from your HSM provider. In this case, I assume that the DB file name is DB.CER. The DB key must be formatted using the [Format-SecureBootUEFI](#) command inside Windows PowerShell before being imported.

| Format-SecureBootUEFI Command Line Parameter | Meaning |
|--|---|
| -Name DB | Indicates that you are working with the collection of certificates in your Secure Boot database (DB). |
| -SignatureOwner DEF16466-F946-4E71-BE22-CF8B1B7B36A0 | The hexadecimal number is a GUID that uniquely identifies you to the platform. Since this represents the signature owner, it should be the same GUID used to import the PK. |
| -ContentFilePath .\NewHpDb_SigList.bin | This file is created to hold the content that is generated by Format-SecureBootUEFI, i.e. the formatted content. |
| -FormatWithCert | Tells Format-SecureBootUEFI to integrate the entire certificate into the formatted content. |
| -Certificate .\NewDbCert.CER | Indicates the path to the desired certificate, in this case, the DB certificate. |
| -SignableFilePath .\NewHpDb_SigList_Serialization_for_DB.bin | Specifies the file that should be signed after formatting. |
| -Time 2016-02-05T13:30:00Z | Specifies the current date and time, which must be specified. The time specified may be different from the <code>-Time</code> flags used previously because it must be within the validity range of the new certificate. Moreover, that validity range might be later than the validity of the original certificates. |

Table 13 Command line switches to format DB key

If successful, the command should produce output similar to the following:

```

Administrator: Windows PowerShell
PS D:\SecureBootDemo> Format-SecureBootUEFI -Name DB -SignatureOwner DEF16466-F946-4E71-BE22-CF8B1B7B36A0 -ContentFilePath .\NewHpDb_SigList.bin -FormatWithCert -Certificate .\NewDbCert.CER -SignableFilePath .\NewHpDb_SigList_Serialization_for_DB.bin -Time 2016-02-05T13:30:00Z
Name Time AppendWrite Content
----
db 2016-02-05T13:30:00Z False {161, 89, 192, 165...}

PS D:\SecureBootDemo>
    
```

Figure 28 Successful output with formatted DB key

Again, it is the [SignableFilePath](#) file, in this case, [NewHpDb_SigList_Serialization_for_DB.bin](#), which is submitted to your HSM solution for signing. This file should be signed using the private key for your new KEK, already imported into the Secure Boot database. A proper signing command for [NewHpDb_SigList_Serialization_for_DB.bin](#), if using a PFX file⁶, is as follows. In this case, [signtool](#) must be in your path:

```
signtool.exe sign /fd sha256 /p7 .\ /p7co 1.2.840.113549.1.7.1 /p7ce DetachedSignedData /a /f .\KEK.PFX /p <password> NewHpDb_SigList_Serialization_for_DB.bin
```

Figure 29 Command line to sign signature list for DB

Replace <password> with the actual private key password for your PFX file. The result of the above command is a KEK-signed DB certificate serialized into a file called [NewHpDb_SigList_Serialization_for_DB.bin.p7](#). Use the signed file that is provided by your HSM provider.

Once you have the new KEK-signed certificate for your Secure BootDB, it is ready for import to your platform. Importing is done with the [Set-SecureBootUEFI](#) command inside Windows PowerShell.

| Set-SecureBootUEFI Command Line Parameter | Meaning |
|---|---|
| -Name DB | Indicates that you are working with the collection of certificates in your Secure Boot database (DB). |
| -Time 2016-02-01T13:30:00Z | Specifies the current date and time, which must be specified. |
| -ContentFilePath .\NewHpDb_SigList.bin | Specifies the name of the file which contains the <i>unsigned</i> , formatted DB certificate. |
| -SignedFilePath .\NewHpDb_SigList_Serialization_for_DB.bin.p7 | Specifies the name of the file which contains the <i>signed</i> , formatted DB certificate. |
| -OutputFilePath .\NewHpDb_Output_for_DB.bin | Specifies the file which will contain the output of the command upon successful completion. |

Table 14 Command line switches to import KEK-signed certificate

If successful, the command should produce output similar to the following:

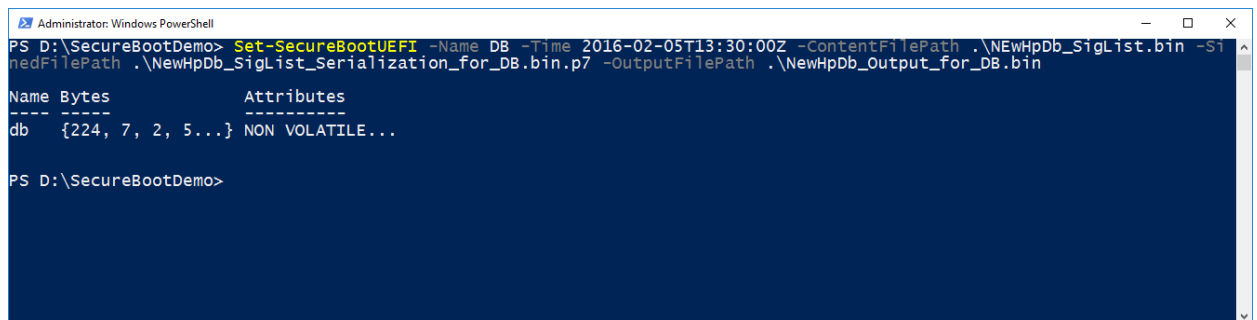


Figure 30 Successful import

Note that this command *adds* another certificate to the Secure Boot database (DB). It does *not* overwrite the existing set of certificates. This command creates a valid SetVariable() package in [DB_NewKey_ImportDB.bin](#) but does not set that value into the BIOS.

⁶ This would be the approach if you used self-signing certificates, but it is strongly recommended that you perform the same action in the context of your own HSM provider.

There is one more step required to use the Windows tools to import the KEK: writing the KEK itself to Non-volatile Random Access Memory (NVRAM). Use the [Set-SecureBootUEFI](#) command inside Windows PowerShell for this purpose

| Set-SecureBootUEFI Command Line Parameter | Meaning |
|---|---|
| -Name DB | Indicates that you are working with the Secure Boot certificate database (DB). |
| -Time 2016-02-01T13:30:00Z | Specifies the current date and time, which must be specified. |
| -ContentFilePath .\NewHpDb_SigList.bin | Specifies the name of the file which contains the <i>unsigned</i> , unformatted DB, created in a previous step. |
| -SignedFilePath .\NewHpDb_SigList_Serialization_for_DB.bin.p7 | Specifies the name of the file which contains the <i>signed</i> , formatted DB. Signed in the previous step. |

Table 15 Command line switches to import the KEK-signed DB certificate

If successful, the command should produce output similar to the following:

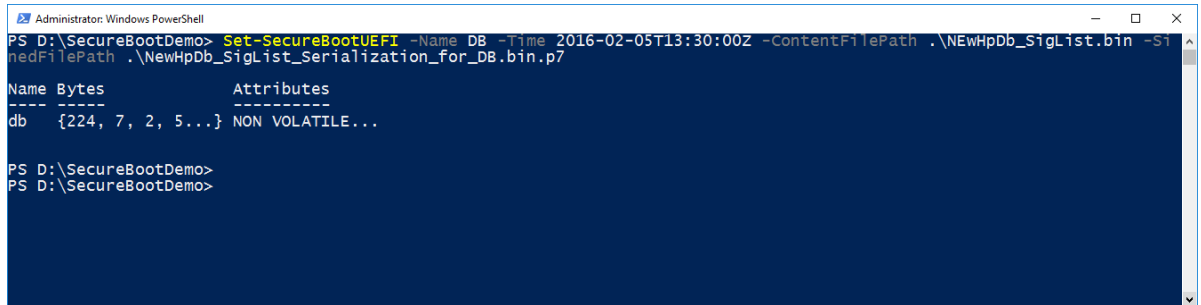


Figure 31 Successful import

2.9.2 DBX

To append a new DBX certificate, follow the instructions itemized under the DB heading immediately above, but use DB for the `-Name` parameter on the command line. Doing so adds a certificate to the Secure Boot DBX rather than to the Secure Boot DB.

3 References

- Windows 8.1 Secure Boot Key Creation and Management Guidance. <https://technet.microsoft.com/en-us/library/dn747883.aspx>
- Get-SecureBootUEFI command. [https://technet.microsoft.com/en-us/library/jj603039\(v=wps.630\).aspx](https://technet.microsoft.com/en-us/library/jj603039(v=wps.630).aspx)
- Secure Boot Key Generation and Signing Using HSM (Example). <https://technet.microsoft.com/en-us/library/dn747881.aspx>
- A Layman's Guide to a Subset of ASN.1, BER, and DER. <http://luca.ntop.org/Teaching/Appunti/asn1.html>
- Format-SecureBootUEFI command. [https://technet.microsoft.com/en-us/library/jj603044\(v=wps.630\).aspx](https://technet.microsoft.com/en-us/library/jj603044(v=wps.630).aspx)
- Set-SecureBootUEFI command. [https://technet.microsoft.com/en-us/library/jj603045\(v=wps.630\).aspx](https://technet.microsoft.com/en-us/library/jj603045(v=wps.630).aspx)
- Create GUID (guidgen.exe). [https://msdn.microsoft.com/en-us/library/ms241442\(v=vs.80\).aspx](https://msdn.microsoft.com/en-us/library/ms241442(v=vs.80).aspx)
- SignTool. [https://msdn.microsoft.com/en-us/library/windows/desktop/aa387764\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/aa387764(v=vs.85).aspx)
- OpenSSL Wiki. <https://wiki.openssl.org/index.php/Binaries>
- The Most Common OpenSSL Commands. <https://www.sslshopper.com/article-most-common-openssl-commands.html>