
Compressed Certificate Definition

Introduction

This document provides the details required to integrate a Microchip ATECC CryptoAuthentication™ secure element into a third-party X.509 certificate chain. Due to the size requirements of an X.509 Certificate, only a portion of the certificate can be stored in the secure element, with the remainder being stored elsewhere in the system. The portion of the certificate that is stored in the secure element is known as the Compressed Certificate¹. The host system has the responsibility to fully reconstitute the complete X.509 Certificate by combining the different portions stored in the secure element and elsewhere in the system or microcontroller memory.

Features

- Compressed Certificate Details
- Certificate Template
- Support for X.509 Certificates
- Microchip ATECC Device Supported Authentication Chaining Overview

¹ This technique is not limited to ATECC CryptoAuthentication devices. It can be used with any device that supports Elliptic Curve Cryptography (ECC). The initial use of this method was however defined for Microchip ATECC devices.

Table of Contents

Introduction.....	1
Features.....	1
1. Technical Overview.....	3
1.1. Provisioning Overview.....	3
1.2. Device Authentication Overview.....	4
1.2.1. Supported ATECC Chain.....	4
1.2.2. Device Authentication Sequence.....	4
2. Compressed Certificates.....	6
2.1. P256 Compressed Certificate.....	6
2.2. Signature.....	6
2.3. Encoded Dates.....	6
2.4. Signer ID.....	8
2.5. Template ID.....	8
2.6. Chain ID.....	8
2.7. Serial Number Source.....	8
2.7.1. Stored Serial Numbers (SN Source = 0x0).....	9
2.7.2. Generated from Subject Public Key (SN Source = 0xA).....	9
2.7.3. Generated from Device Serial Number (SN Source = 0xB).....	9
2.8. Compressed Certificate Format Version.....	10
2.9. Reserved Byte.....	10
3. X.509 Certificate.....	11
3.1. X.509 Compressed Certificate Elements.....	11
3.2. Signature Reconstruction.....	12
3.2.1. Encoding steps.....	13
4. References.....	15
5. Revision History.....	16
The Microchip Website.....	17
Product Change Notification Service.....	17
Customer Support.....	17
Microchip Devices Code Protection Feature.....	17
Legal Notice.....	17
Trademarks.....	18
Quality Management System.....	18
Worldwide Sales and Service.....	19

1. Technical Overview

1.1 Provisioning Overview

Prior to deployment, the ATECC device needs to be provisioned. This consists of defining the configuration and the programming of device memory with secrets and other application-specific data. The way to do this is beyond the scope of this document. The specific tools that are used to do this may be implemented by the customer or they can be based on Microchip software tools or the tools of other vendors.



However the provisioning is done the customer must ensure that for production-level devices, the tools used to provision the ATECC device ensure the confidentiality and security of ECC Private keys, Symmetric secrets and other forms of secret data.



Important: The Microchip Trust Platform Design Suite does provide the ability to provision Microchip secure elements through some software utilities. However, while these software functions are adequate to create prototype units this is not a secure environment overall and should never be used for actual production devices.

For shared secret data and keys, the protected memory of the device is written so the keys can be used in the secure execution environment of the device and secure memory can be accessed with encrypted reads.

For ECC keys, the device internally creates a unique private key, stores the private key into a protected key slot, calculates the associated public key, then returns the public key. The private key can then be used to sign messages and the public key will be signed into a trusted chain.

Device Provisioning Steps

1. Send a command to the device to create a unique public-private key pair.
2. Construct a *To Be Signed* (TBS) certificate for the device using the device certificate template, the device public key, and other device-specific certificate elements.
3. A SHA-256 digest of the TBS certificate is signed by the Signer⁽¹⁾.
4. The compressed certificate of the device is written to the device. The compressed certificate includes:
 - Device public key⁽²⁾
 - Signature of TBS by Signer
 - Device-specific certificate elements
5. The compressed certificate of the signer is written to the device. Compressed certificate includes:
 - Signer public key
 - Signature of Signer TBS by Issuer
 - Signer-specific certificate elements

Notes:

1. The term “Signer” is used to refer to an intermediate Certificate Authority (CA) that is used to sign a device certificate.
2. The public key for the device certificate is typically regenerated from the private key, but may optionally be stored in a separate device slot.



Notice: The provisioning results in the device contain two certificates: One for the device itself and one for the signer that signed it.

1.2 Device Authentication Overview

1.2.1 Supported ATECC Chain

The device is preconfigured as shown in [1.2.2 Device Authentication Sequence](#). All elements required for authenticating the chain through the Issuer are stored on the device.

The certificate chain that is supported by the device is defined in the table below.

Table 1-1. Certificate Chain Definition

Certificate	Signed By	Signer For	Notes
Device	Signer	Challenge	A challenge is sent to the device to demonstrate possession of the Private Key that is associated with the Public Key in the Device Certificate.
Signer	Issuer	Device	The Signer Private Key is contained in the Signer CA that is used in production.
Issuer	Root	Signer	Used as the root in some systems.
Optional Root	Optional Root	Issuer	Represents a chain of one or more levels above the Issuer to include Certificate Authorities. These certificates may use larger ECC curves.


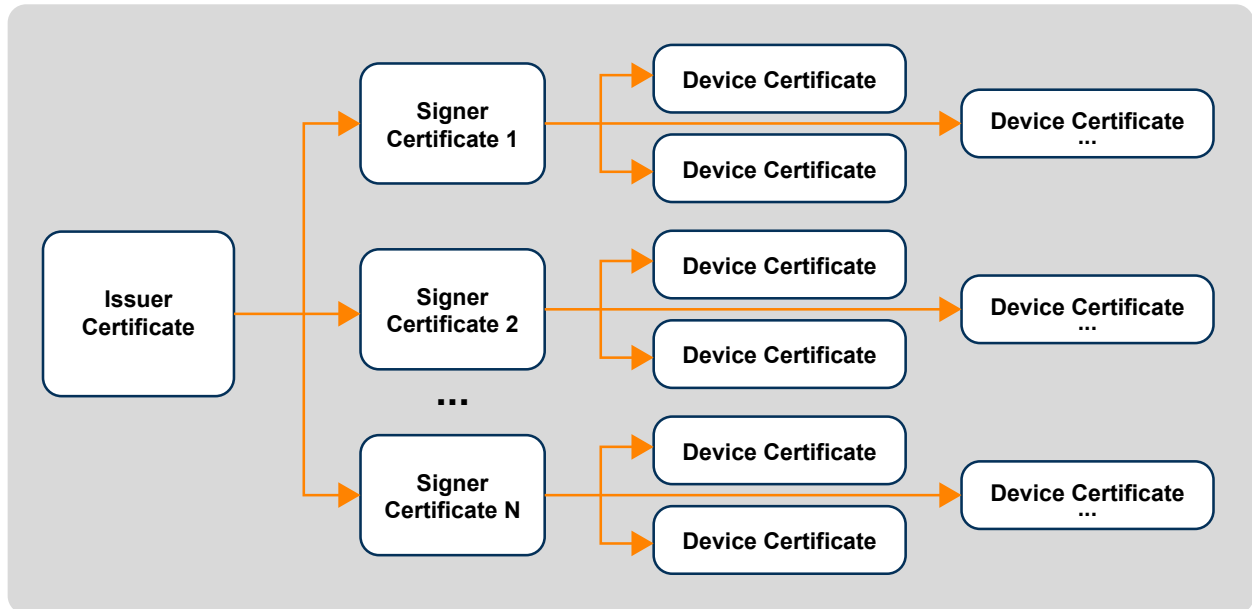
 **Notice:** The Issuer, Signer, and Device Certificates *must* use the same ECC curve.

Figure 1-1. Certificate Chain Diagram

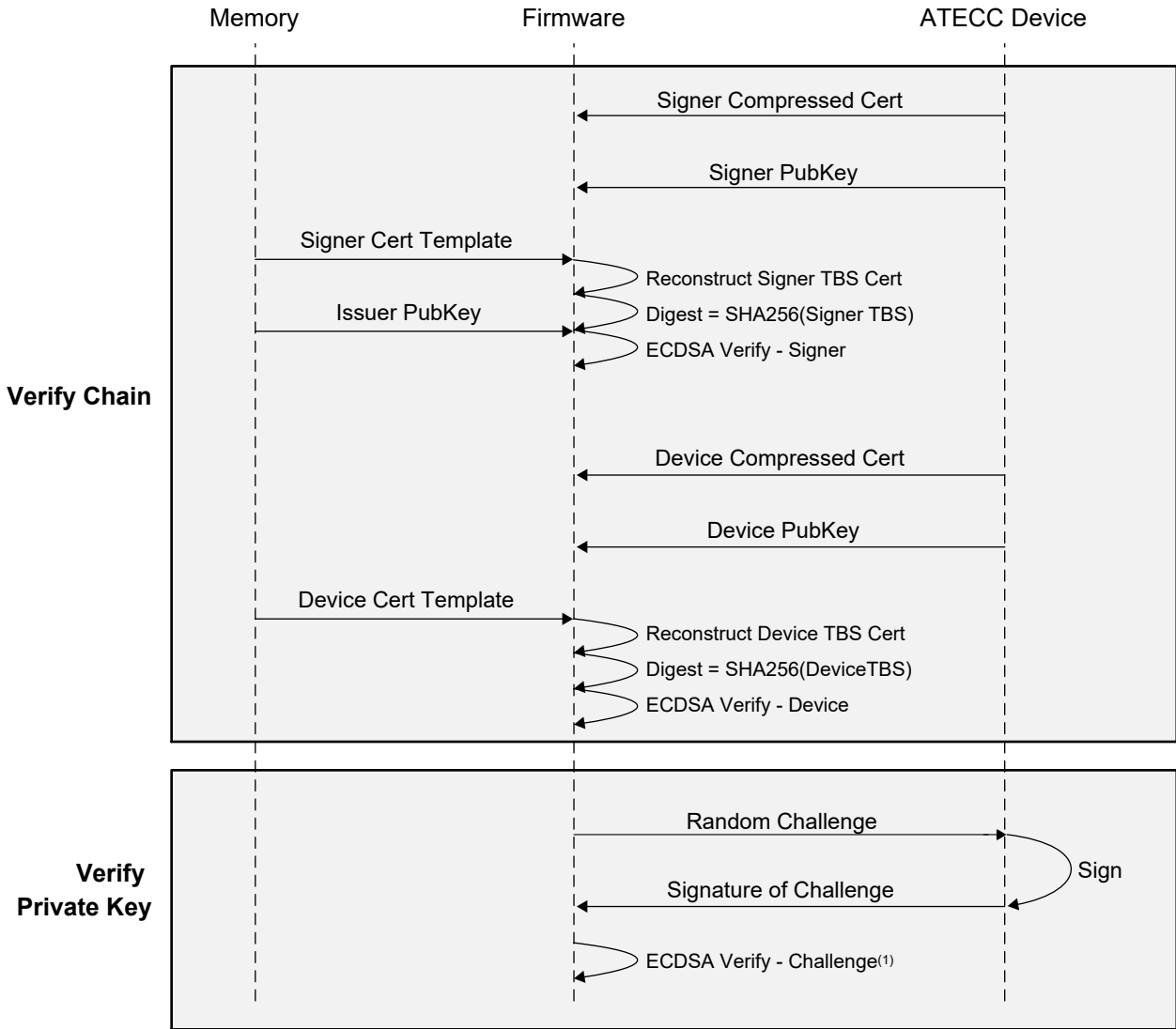


1.2.2 Device Authentication Sequence

Since the signing of the device is performed by the signer, it allows Microchip to produce devices that can be validated into the certificate chain while managing the other production considerations.

With this scheme, the device authentication only needs the issuer certificate to be installed on the platform. All remaining components of the device certificate and its signer certificate can be produced via information stored within the device. The following sequence diagram shows the authentication of the device.

Figure 1-2. Device Authentication Sequence Diagram



Note:

1. The ECDSA Verify operations use the "Device PubKey."

2. Compressed Certificates

Since the ATECC device does not have enough storage for two full X.509 certificates, the concept of a Compressed Certificate and a Certificate Template is introduced. The Compressed Certificate includes the Public Key, Certificate Elements, and the Signature. These elements can be combined with the Certificate Template to construct the original Certificate.

This certificate reconstruction is required for both the device and the signer of the device; therefore, the details are defined for both certificate types. Each device stores the elements to reconstruct both the Signer Certificate and the Device Certificate.

2.1 P256 Compressed Certificate

For P256 curves, the Compressed Certificate can be stored with the signature in one of the 72 byte slots (8 to 15) of the ATECC device.

The P256 Compressed Certificate is defined as follows:

Table 2-1. P256 Compressed Certificate

Signature: 64 Bytes Bytes 0 to 63									
Byte 64	Byte 65	Byte 66	Byte 67	Byte 68	Byte 69		Byte 70	Byte 71	
Encoded Dates (24 bits)			Signer ID (16 bits)		Template ID (4 bits)	Chain ID (4 bits)	SN Source (4 bits)	Format Version (4 bits)	Reserved (8 bits)

Table 2-2. P256 Compressed Certificate Definition

Element	Size (bits)	Description
Signature	512	Certificate signature stored as the 32 byte R and S unsigned big-endian integers.
Encoded Dates	24	Certificate issue and expiration dates in a bit-packed format.
Signer ID	16	ID of the specific signer used to sign the certificate (device cert) or of the signer itself (signer cert).
Template ID	4	ID the certificate template to be used to reconstruct the full X.509 certificate.
Chain ID	4	ID of the certificate chain being used.
SN Source	4	Indicates where to find or how to generate the certificate serial number.
Format Version	4	Version of the compressed certificate format. 0 is the only version.
Reserved	8	Reserved byte.

2.2 Signature

An elliptic curve signature has two components, R and S. For a P256 curve, these components are 32 bytes. The signature in the compressed certificate stores the R integer first, then S in an unsigned big-endian format.

2.3 Encoded Dates

The encoded dates are three bytes that represent the issue and expiration dates of the certificate in a compressed (bit packed) format. The issue date is assumed to be in Universal Time Coordinated² (UTC) format.

Table 2-3. Encoded Date Format

Byte 0								Byte 1								Byte 2							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Year (5 bits)								Month (4 bits)				Day (5 bits)				Hour (5 bits)				Expire Years (5 bits)			

Table 2-4. Encoded Dates Definition

Element	Size (bits)	Description	Range
Year	5	Issue date year starting from 2000.	0 to 31 (2000 to 2031)
Month	4	Issue date month.	1 to 12
Day	5	Issue date day.	1 to 31
Hour	5	Issue date hour.	0 to 23
Expire Years	5	How many years the certificate is valid for.	0 to 31

The expiration date is a set number of years (Expire Years) from the issue date. Use 0 for the Expire Years field to indicate no expiration date.



Notice: The issue date only has a resolution of hours. Minutes and seconds are assumed to be zero.

Example 1: Parsing Encoded Dates Example

The example encoded date represents the following dates:

Issue date: 2014-10-15 16:00:00 UTC

Expire date: 2028-10-15 16:00:00 UTC (14 years after issue date)

```
unsigned char enc_dates[] = {0x75, 0x3E, 0x0E};

int issue_date_year = (enc_dates[0] >> 3) + 2000;
int issue_date_month = ((enc_dates[0] & 0x07) << 1) | ((enc_dates[1] & 0x80) >> 7);
int issue_date_day = ((enc_dates[1] & 0x7C) >> 2);
int issue_date_hour = ((enc_dates[1] & 0x03) << 3) | ((enc_dates[2] & 0xE0) >> 5);
int expire_years = (enc_dates[2] & 0x1F);
```

If the expiration in years is 0, then the certificate is supposed to have no expiration date. Since X.509 certificates have no provision for this, the maximum possible date is used. X.509 defines two different date formats in [RFC5280](#).

Table 2-5. Time Formats and Maximum Certificate Expiration Date

Time Format	Max Date	Decoded Value
UTC Time Format	491231235959Z	2049-12-31 23:59:59 UTC
Generalized Time Format	99991231235959Z	9999-12-31 23:59:59 UTC

² Also known as Coordinated Universal Time

2.4 Signer ID

The Signer ID is a two-byte identifier for the specific signer that is used to sign the certificate. The Microchip internal provisioning system uses the following format.

Table 2-6. Signer ID Format

Byte 0								Byte 1							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Module ID (8 bits)								ECC ID (4 bits)				Slot ID (4 bits)			

Table 2-7. Signer ID Definition

Element	Size (bits)	Description
Module ID	8	Specific signing module used.
ECC ID	4	ECC device on the specified signing module used.
Slot ID	4	Slot on the specified ECC device used.

2.5 Template ID

The Template ID provides the necessary information to expand a compressed certificate into a full X.509 certificate. A typical chain may have a different template for the signer certificate than for the device certificate. The Template ID allows for up to 16 different templates to be used for unique templates at each level.

Table 2-8. Conventions

Template ID	Description
0	Use the Device Template.
1	Use the Signer Template.
n	Issuer Template or higher.

2.6 Chain ID

The Chain ID provides the necessary information to allow for multiple certificate chains to be defined for a particular customer. Most customers have only one certificate chain that uses a Chain ID of 0.

2.7 Serial Number Source

The Serial Number Source determines where the serial number comes from when reconstructing the X.509 certificate from the compressed certificate. A typical X.509 certificate serial number is at least 8 bytes, can be up to 20 bytes, and must be unique. Each certificate must have a different serial number and there is no room in the compressed certificate slot to store it. Therefore, the serial number needs to be either stored in another slot on the ATECC device or generated from already known data that would result in a unique value. In all of the options listed, the most significant two bits should be forced to 01. This ensures the serial number is positive and un-trimmable per RFC5280 Section 4.1.2.2 (See Reference [Serial Number](#)) and ASN.1 integer encoding.

Table 2-9. Serial Number Source

SN Source	Value	Description
Stored SN	0x0	Random number generated and written to a slot.
Generated with Public Key	0xA	Use the Public Key and encoded dates to generate the certificate serial number.
Generated with Device SN	0xB	Use the unique device serial number and encoded dates to generate the certificate serial number.

2.7.1 Stored Serial Numbers (SN Source = 0x0)

During certificate generation, the serial number is a random number. The resulting serial number is stored in a slot on the ATECC device. The two upper most significant bits are set to 01. When reconstructing the certificate, the serial number is read from the slot where it was saved and inserted into the certificate template.

```
const int cert_sn_size = 16;
unsigned char cert_sn[cert_sn_size];
// Fill the serial_number with random data (pseudo-function)
random_bytes(cert_sn, cert_sn_size);
// Set two most significant bits to 01 to ensure a positive, untrimmable serial number
cert_sn[0] &= 0x7F;
cert_sn[0] |= 0x40;
```

2.7.2 Generated from Subject Public Key (SN Source = 0xA)

Serial number is generated from a hash of the subject public key (device public key for the device certificate and signer public key for the signer certificate) and encoded dates from the compressed certificate:

- SHA256 (subject public key [64 bytes] + encoded dates [3 bytes])
- Two upper most significant bits are set to 01

This method uses already known information to generate the serial number and doesn't need to be saved to a slot. Serial number should be truncated to the size of the serial number specified for the certificate. Refer to the ECC Key Formatting section in the data sheet for information on how ECC public keys are stored in the EEPROM slots.

```
const int cert_sn_size = 16;
unsigned char cert_sn[cert_sn_size];
unsigned char subject_public_key[72];
unsigned char comp_cert[72];
unsigned char msg[67];
unsigned char digest[32];
// Read the subject key from the relevant slot (pseudo-function)
get_subject_public_key(subject_public_key);
// Read the compressed certificate from the relevant slot (pseudo-function)
get_compressed_cert(comp_cert);
// Build the input message for the hash
memcpy(msg[0], &subject_public_key[4], 32); // Subject public key, X
memcpy(msg[32], &subject_public_key[40], 32); // Subject public key, Y
memcpy(msg[64], &comp_cert[64], 3); // Encoded dates from comp cert
// Perform the SHA256 hash on the message and put the results into digest
sha256(msg, 67, digest);
// Copy only the portion of the digest for the serial number size
memcpy(cert_sn, digest, cert_sn_size);
// Set two most significant bits to 01 to ensure a positive, untrimmable serial number
cert_sn[0] &= 0x7F;
cert_sn[0] |= 0x40;
```

2.7.3 Generated from Device Serial Number (SN Source = 0xB)

Serial number is generated from a hash of the device's serial number and encoded dates from the compressed certificate:

- SHA256 (device SN [9 bytes] + encoded dates [3 bytes])
- Two upper most significant bits are set to 01

This method uses already known information to generate the serial number and doesn't need to be saved to a slot. Serial number should be truncated to the size of the serial number specified for the certificate. This method is also only available to the device certificate, since the signer has no "device serial number."

```
const int cert_sn_size = 16;
unsigned char cert_sn[cert_sn_size];
unsigned char device_config[128];
unsigned char comp_cert[72];
unsigned char msg[12];
unsigned char digest[32];
// Read the config zone from the device (pseudo-function)
get_device_config(device_config);
// Read the compressed certificate from the relevant slot (pseudo-function)
get_compressed_cert(comp_cert);
// Build the input message for the hash
memcpy(&msg[0], &device_config[0], 4); // Device SN[0:3] from config zone
memcpy(&msg[4], &device_config[8], 5); // Device SN[4:8] from config zone
memcpy(&msg[9], &comp_cert[64], 3); // Encoded dates from compressed cert
// Perform the SHA256 hash on the message and put the results into digest
sha256(msg, 12, digest);
// Copy only the portion of the digest for the serial number size
memcpy(cert_sn, digest, cert_sn_size);
// Set most significant bit to 0 to ensure a positive serial number
cert_sn[0] &= 0x7F;
// Set two most significant bits to 01 to ensure a positive, untrimmable serial number
cert_sn[0] &= 0x7F;
cert_sn[0] |= 0x40;
```

2.8 Compressed Certificate Format Version

This is a 4-bit value that can be used to indicate a change in the format of the compressed certificate. There's only one version for now, so this is just set to 0x0.

2.9 Reserved Byte

This byte is reserved for future use and should be set to zero.

3. X.509 Certificate

This section describes how the full X.509 certificates are formatted for the two certificates (Signer and Device) stored within the ATECC device.

3.1 X.509 Compressed Certificate Elements

The elements to recreate the Signer TBS certificate and the Device TBS are stored on the Device. The storage locations for each of the elements can be the Certificate Template, a Device Slot, or a Calculated value.

The table below shows the location of each element of the Signer Certificate.

Table 3-1. Signer Certificate Location

Element	Storage Location	Size	Transform
Serial Number ⁽¹⁾	Device Slot	var	None.
	Calculated	var	See Section 2.7 Serial Number Source .
IssueDate	Compressed Cert	13	See Section 2.3 Encoded Dates . In ASCII YYMMDDHHMMSSZ.
ExpireDate	Compressed Cert	13	See Section 2.3 Encoded Dates . In ASCII YYMMDDHHMMSSZ.
Signer ID	Compressed Cert	4	In ASCII as four upper-case hex digits within the Subject field, Common Name attribute.
Public Key X	Device Slot	32	None.
Public Key Y	Device Slot	32	None.
authorityKeyIdIdentifier	Calculated	20	SHA1 (04 + Issuer Public Key X&Y).
subjectKeyIdIdentifier	Calculated	20	SHA1 (04 + Signer Public Key X&Y).
Signature R	Compressed Cert	32	None.
Signature S	Compressed Cert	32	None.

Note: (1) Customer chooses whether the serial number is stored or calculated.

Table 3-2. Device Certificate Location

Element	Storage Location	Size	Transform
Serial Number ⁽¹⁾	Device Slot	var	None.
	Calculated	var	See Section 2.8, "Serial Number Source."
Signer ID	Compressed Cert	4	In ASCII as four upper-case hex digits within the Issuer field, Common Name attribute.
IssueDate	Compressed Cert	13	See Section 2.4. In ASCII YYMMDDHHMMSSZ.
ExpireDate	Compressed Cert	13	See Section 2.4. In ASCII YYMMDDHHMMSSZ.
Public Key X	Device Slot	32	None.
Public Key Y	Device Slot	32	None.
authorityKeyIdIdentifier	Calculated	20	SHA1 (04 + Signer Public Key X&Y).
subjectKeyIdIdentifier	Calculated	20	SHA1 (04 + Device Public Key X&Y).

.....continued			
Element	Storage Location	Size	Transform
Signature R	Compressed Cert	32	None.
Signature S	Compressed Cert	32	None.

Note: (1) Customer chooses whether the serial number is stored or calculated.

3.2 Signature Reconstruction

The X.509 signature presents a unique case as it is not possible to perform a simple “copy and paste” into the template as with the other elements. This is because the signature’s R and S components are stored as ASN.1 integers, whose format and length can change depending on the actual value. Since signature values are different for each certificate, the rules have to be followed closely.

ASN.1 Integers are stored in a big-endian signed format. ASN.1 encoding rules (see section 4. [References](#)) require that if the upper-most nine bits are all ones or zeros, then the upper-most byte must be trimmed, reducing the encoded size of the integer.

Furthermore, since the R and S integers for a signature are unsigned, if either integer has a one as its uppermost bit, then a zero byte must be added to the pad the integer and prevent it from being interpreted as a negative number.

Example 1: ASN.1 Signature With Padded Integer

The following example shows all the elements of an ASN.1 signature with the S integer requiring padding. Below are the raw bytes from a signature. This starts from the signature offset.

```

03 48 00 30 45 02 20 37 4A DD 5A B5 7E 48 F8 EA
59 AB C6 E6 09 54 E8 46 25 8C CA 1E 63 25 F4 A4
86 55 20 B0 FA 48 AE 02 21 00 9C 92 55 1E 8B 85
5E 30 EA A0 9B C8 47 3C 79 27 A4 60 E8 16 11 93
5D 60 C2 D6 D8 34 BF 99 B5 CF
  
```

Note: **Bold** bytes are tags; *italic* bytes are lengths; underlined are the actual R and S Integers of the signature.

ASN.1 data has a tag, length, value format, and tends to follow a tree structure. The above data can be broken down into the following structure:

The above data can be broken down into the following structure:

BIT STRING (tag=03, length=48 (72 bytes), unused bits=00)

SEQUENCE (tag=30, length=45 (69 bytes))

INTEGER (tag=02, length=20 (32 bytes)) – R integer

INTEGER (tag=02, length=21 (33 bytes)) – S integer

This ASN.1 signature encodes the raw signature bytes:

```

37 4A DD 5A B5 7E 48 F8 EA 59 AB C6 E6 09 54 E8      R
46 25 8C CA 1E 63 25 F4 A4 86 55 20 B0 FA 48 AE

9C 92 55 1E 8B 85 5E 30 EA A0 9B C8 47 3C 79 27      S
A4 60 E8 16 11 93 5D 60 C2 D6 D8 34 BF 99 B5 CF
  
```



Notice: Notice the S integer is 33 bytes long in the ASN.1 encoding. That is because its upper-most bit is a one (0x9C = 0b10011100) and the entire integer is interpreted as a negative number unless padded with a 00 byte.

Example 2: ASN.1 Signature With Trimmable Bytes

The next example illustrates when bytes need to be trimmed, since dealing only with unsigned numbers, it is only required to look at the case where nine or more 0 bits are in the uppermost position.

The following signature has too many 0 bits in the upper-most position.

```

00 55 DD 5A B5 7E 48 F8 EA 59 AB C6 E6 09 54 E8      R
46 25 8C CA 1E 63 25 F4 A4 86 55 20 B0 FA 48 AE

00 00 7F 1E 8B 85 5E 30 EA A0 9B C8 47 3C 79 27      S
A4 60 E8 16 11 93 5D 60 C2 D6 D8 34 BF 99 B5 CF
    
```

This would get encoded as the following ASN.1 signature:

```

03 44 00 30 41 02 1F 55 DD 5A B5 7E 48 F8 EA 59
AB C6 E6 09 54 E8 46 25 8C CA 1E 63 25 F4 A4 86
55 20 B0 FA 48 AE 02 1E 7F 1E 8B 85 5E 30 EA A0
9B C8 47 3C 79 27 A4 60 E8 16 11 93 5D 60 C2 D6
D8 34 BF 99 B5 CF
    
```

Note: **Bold** bytes are tags; *italics* bytes are lengths; underlined are the actual R and S Integers of the signature.

The above data can be broken down into the following structure:

```

BIT STRING (tag=03, length=44 (68 bytes), unused bits=00)
SEQUENCE (tag=30, length=45 (65 bytes))
    INTEGER (tag=02, length=1F (31 bytes)) – R integer
    INTEGER (tag=02, length=1E (30 bytes)) – S integer
    
```



Notice: The bit string and sequence lengths have adjusted for the different integer sizes.

Because the R integer has at least nine upper-most 0 bits (0x00 0x55 = 0b**00000000** 01010101), the upper byte gets trimmed. Because the S integer has at least 17 upper-most 0 bits (0x00 0x00 0x7F = 0b**00000000** **00000000** 01111111), the upper two bytes get trimmed.

The last piece that needs to be considered is the size of the certificate as a whole. When the signature changes size, the length field for the whole certificate needs to be adjusted as well. The length of the certificate is a 2-byte big-endian number found as the third and fourth bytes of the certificate (index 2 and 3). For example, the following is the first 4 bytes of an example certificate template:

```
30 82 01 B8
```

The bytes in bold are the length (0x01B8 = 440 bytes). If a signature changed from the first example above to the second example, where bytes had to be trimmed, then the ASN.1 signature shrunk in size from 74 bytes to 70 bytes. This change in four bytes would need to be reflected in the certificate size:

```
30 82 01 B4
```

The certificate size is now 436 bytes, reflecting the ASN.1 signature change.

3.2.1 Encoding steps

The encoding steps can be summarized as such:

1. Encode the R integer
 - If the uppermost bit is a one, pad the integer with a zero (encoded integer is now 33 bytes).
 - Otherwise, while the upper-most nine bits are zeros, trim the upper-most byte.

2. Encode the S integer
 - If the uppermost bit is a one, pad the integer with a zero (encoded integer is now 33 bytes).
 - Otherwise, while the upper-most nine bits are zeros, trim the upper-most byte.
3. Set the sequence and bit string length fields based on the encoded R and S integer sizes.
4. Adjust the certificate length field by the change in signature size.

4. References

1. ASN.1 Length encoding X.690 (www.itu.int/rec/T-REC-X.690/e) Section 8.1.3
2. ASN.1 Integer encoding X.690 (www.itu.int/rec/T-REC-X.690/e) Section 8.3
3. X.680 (www.itu.int/rec/T-REC-X.680/e) Section 19.8 for tag value
4. ECDSA-Sig-Value encoding RFC 5480 Appendix A (tools.ietf.org/html/rfc5480)
5. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile (tools.ietf.org/html/rfc5280)
 - Sections 4.1.2.5.1 for UTC time format and 4.1.2.5.2 for Generalized time format
 - Section 4.1.2.2 Serial Number
6. SECG SEC1 (www.secg.org/sec1-v2.pdf)
7. Microchip CryptoAuthentication Family (www.microchip.com/design-centers/security-ics/cryptoauthentication)

5. Revision History

Doc Rev.	Date	Comments
A	06/2020	Initial release of this document. Replaces Atmel Document #8974A.

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with

your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TempTrackr, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, Vite, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2020, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-6167-8

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
<p>Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Tel: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com</p> <p>Atlanta Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455</p> <p>Austin, TX Tel: 512-257-3370</p> <p>Boston Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088</p> <p>Chicago Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075</p> <p>Dallas Addison, TX Tel: 972-818-7423 Fax: 972-818-2924</p> <p>Detroit Novi, MI Tel: 248-848-4000</p> <p>Houston, TX Tel: 281-894-5983</p> <p>Indianapolis Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380</p> <p>Los Angeles Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800</p> <p>Raleigh, NC Tel: 919-844-7510</p> <p>New York, NY Tel: 631-435-6000</p> <p>San Jose, CA Tel: 408-735-9110 Tel: 408-436-4270</p> <p>Canada - Toronto Tel: 905-695-1980 Fax: 905-695-2078</p>	<p>Australia - Sydney Tel: 61-2-9868-6733</p> <p>China - Beijing Tel: 86-10-8569-7000</p> <p>China - Chengdu Tel: 86-28-8665-5511</p> <p>China - Chongqing Tel: 86-23-8980-9588</p> <p>China - Dongguan Tel: 86-769-8702-9880</p> <p>China - Guangzhou Tel: 86-20-8755-8029</p> <p>China - Hangzhou Tel: 86-571-8792-8115</p> <p>China - Hong Kong SAR Tel: 852-2943-5100</p> <p>China - Nanjing Tel: 86-25-8473-2460</p> <p>China - Qingdao Tel: 86-532-8502-7355</p> <p>China - Shanghai Tel: 86-21-3326-8000</p> <p>China - Shenyang Tel: 86-24-2334-2829</p> <p>China - Shenzhen Tel: 86-755-8864-2200</p> <p>China - Suzhou Tel: 86-186-6233-1526</p> <p>China - Wuhan Tel: 86-27-5980-5300</p> <p>China - Xian Tel: 86-29-8833-7252</p> <p>China - Xiamen Tel: 86-592-2388138</p> <p>China - Zhuhai Tel: 86-756-3210040</p>	<p>India - Bangalore Tel: 91-80-3090-4444</p> <p>India - New Delhi Tel: 91-11-4160-8631</p> <p>India - Pune Tel: 91-20-4121-0141</p> <p>Japan - Osaka Tel: 81-6-6152-7160</p> <p>Japan - Tokyo Tel: 81-3-6880-3770</p> <p>Korea - Daegu Tel: 82-53-744-4301</p> <p>Korea - Seoul Tel: 82-2-554-7200</p> <p>Malaysia - Kuala Lumpur Tel: 60-3-7651-7906</p> <p>Malaysia - Penang Tel: 60-4-227-8870</p> <p>Philippines - Manila Tel: 63-2-634-9065</p> <p>Singapore Tel: 65-6334-8870</p> <p>Taiwan - Hsin Chu Tel: 886-3-577-8366</p> <p>Taiwan - Kaohsiung Tel: 886-7-213-7830</p> <p>Taiwan - Taipei Tel: 886-2-2508-8600</p> <p>Thailand - Bangkok Tel: 66-2-694-1351</p> <p>Vietnam - Ho Chi Minh Tel: 84-28-5448-2100</p>	<p>Austria - Wels Tel: 43-7242-2244-39 Fax: 43-7242-2244-393</p> <p>Denmark - Copenhagen Tel: 45-4450-2828 Fax: 45-4485-2829</p> <p>Finland - Espoo Tel: 358-9-4520-820</p> <p>France - Paris Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79</p> <p>Germany - Garching Tel: 49-8931-9700</p> <p>Germany - Haan Tel: 49-2129-3766400</p> <p>Germany - Heilbronn Tel: 49-7131-72400</p> <p>Germany - Karlsruhe Tel: 49-721-625370</p> <p>Germany - Munich Tel: 49-89-627-144-0 Fax: 49-89-627-144-44</p> <p>Germany - Rosenheim Tel: 49-8031-354-560</p> <p>Israel - Ra'anana Tel: 972-9-744-7705</p> <p>Italy - Milan Tel: 39-0331-742611 Fax: 39-0331-466781</p> <p>Italy - Padova Tel: 39-049-7625286</p> <p>Netherlands - Drunen Tel: 31-416-690399 Fax: 31-416-690340</p> <p>Norway - Trondheim Tel: 47-72884388</p> <p>Poland - Warsaw Tel: 48-22-3325737</p> <p>Romania - Bucharest Tel: 40-21-407-87-50</p> <p>Spain - Madrid Tel: 34-91-708-08-90 Fax: 34-91-708-08-91</p> <p>Sweden - Gothenberg Tel: 46-31-704-60-40</p> <p>Sweden - Stockholm Tel: 46-8-5090-4654</p> <p>UK - Wokingham Tel: 44-118-921-5800 Fax: 44-118-921-5820</p>