Services

Preventing Unauthorized Purchases with Receipts

Session 305

James Wilson
Software Engineering

Using Receipts

Using Receipts

Know exactly what the customer has paid for

Using Receipts

Know exactly what the customer has paid for

Within your app, on the device and on your servers

Using Receipts

Know exactly what the customer has paid for

Within your app, on the device and on your servers

Protect your revenue

The Receipt

The Receipt

Trusted record of App and In-App Purchases

The Receipt

Trusted record of App and In-App Purchases

Stored on device

The Receipt

Trusted record of App and In-App Purchases

Stored on device

Issued by the App Store

The Receipt

Trusted record of App and In-App Purchases

Stored on device

Issued by the App Store

Signed and verifiable

The Receipt

Trusted record of App and In-App Purchases

Stored on device

Issued by the App Store

Signed and verifiable

For your app, on that device only

Flexibility

Flexibility

Apple provides you with

Flexibility

Apple provides you with

- The receipt format specification

Flexibility

Apple provides you with

- The receipt format specification
- The receipt itself

Flexibility

Apple provides you with

- The receipt format specification
- The receipt itself
- Instructions for on-device receipt validation

Flexibility

Apple provides you with

- The receipt format specification
- The receipt itself
- Instructions for on-device receipt validation
- Online service for server-to-server validation

Flexibility

Apple provides you with

- The receipt format specification
- The receipt itself
- Instructions for on-device receipt validation
- Online service for server-to-server validation

You chose a security level appropriate for your products

Flexibility

Apple provides you with

- The receipt format specification
- The receipt itself
- Instructions for on-device receipt validation
- Online service for server-to-server validation

You chose a security level appropriate for your products

You decide the complexity of the implementation

Decisions

Decisions

How to verify the signature?

Decisions

How to verify the signature?

How to verify the device?

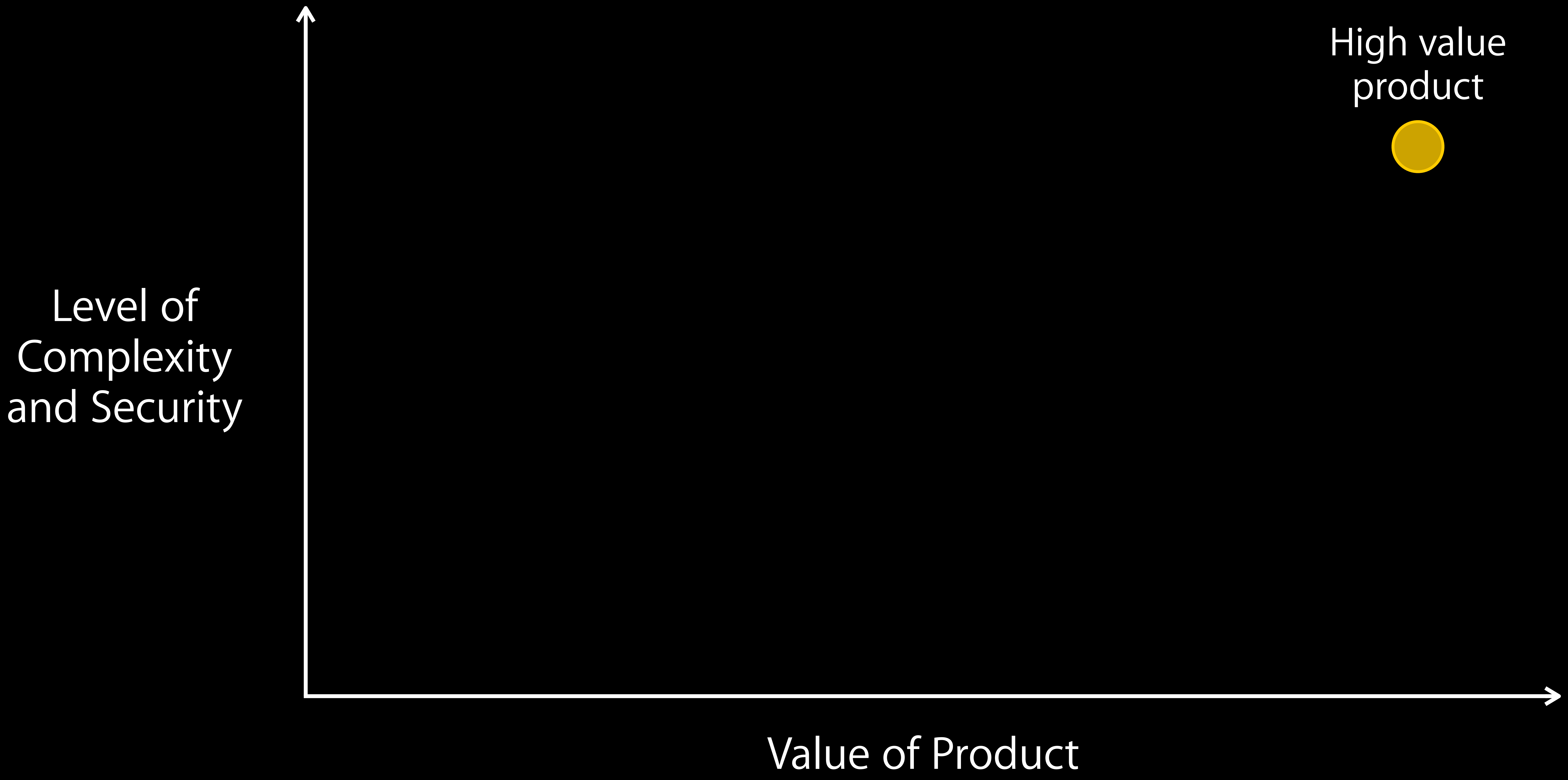
Decisions

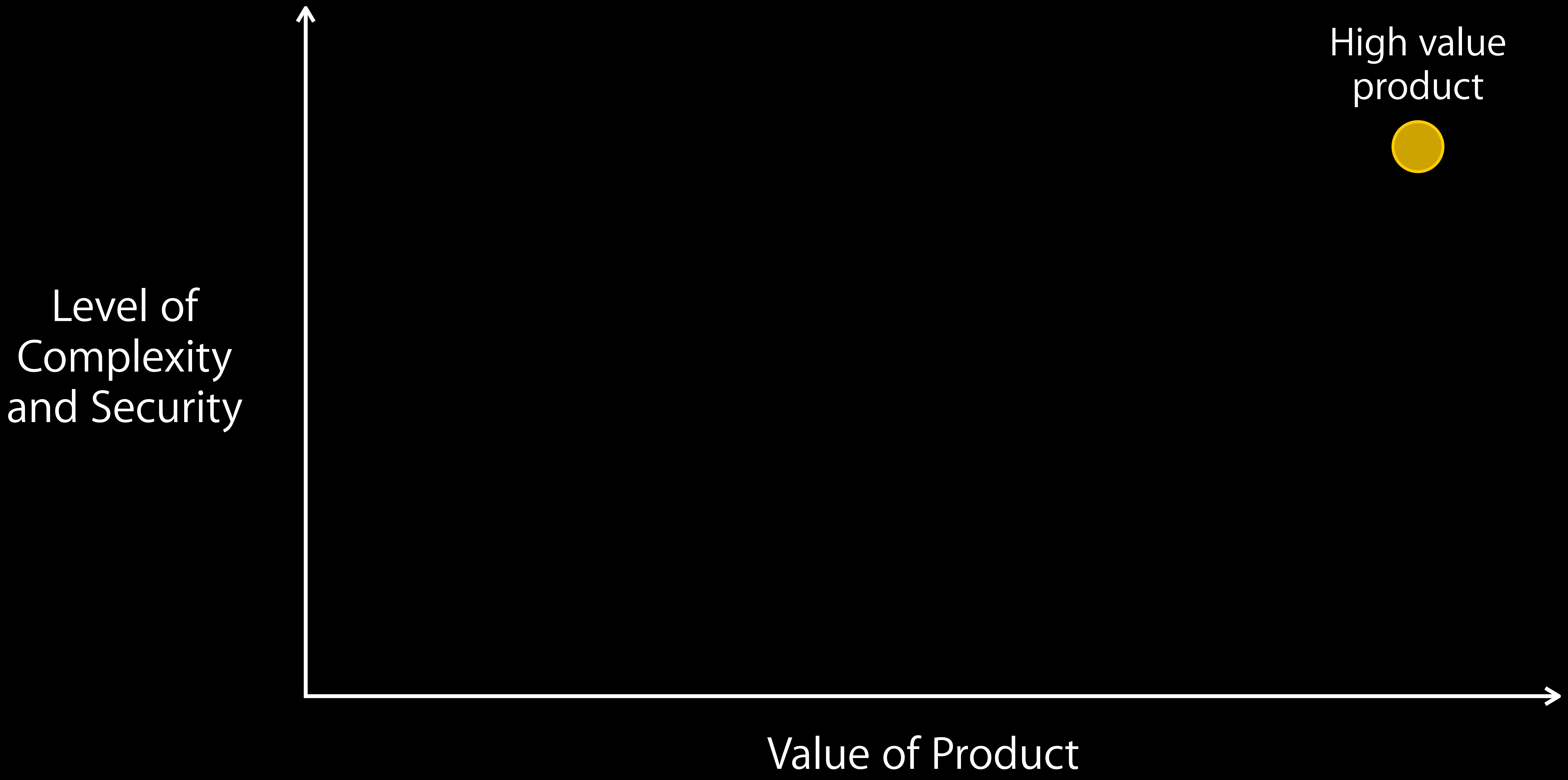
How to verify the signature?

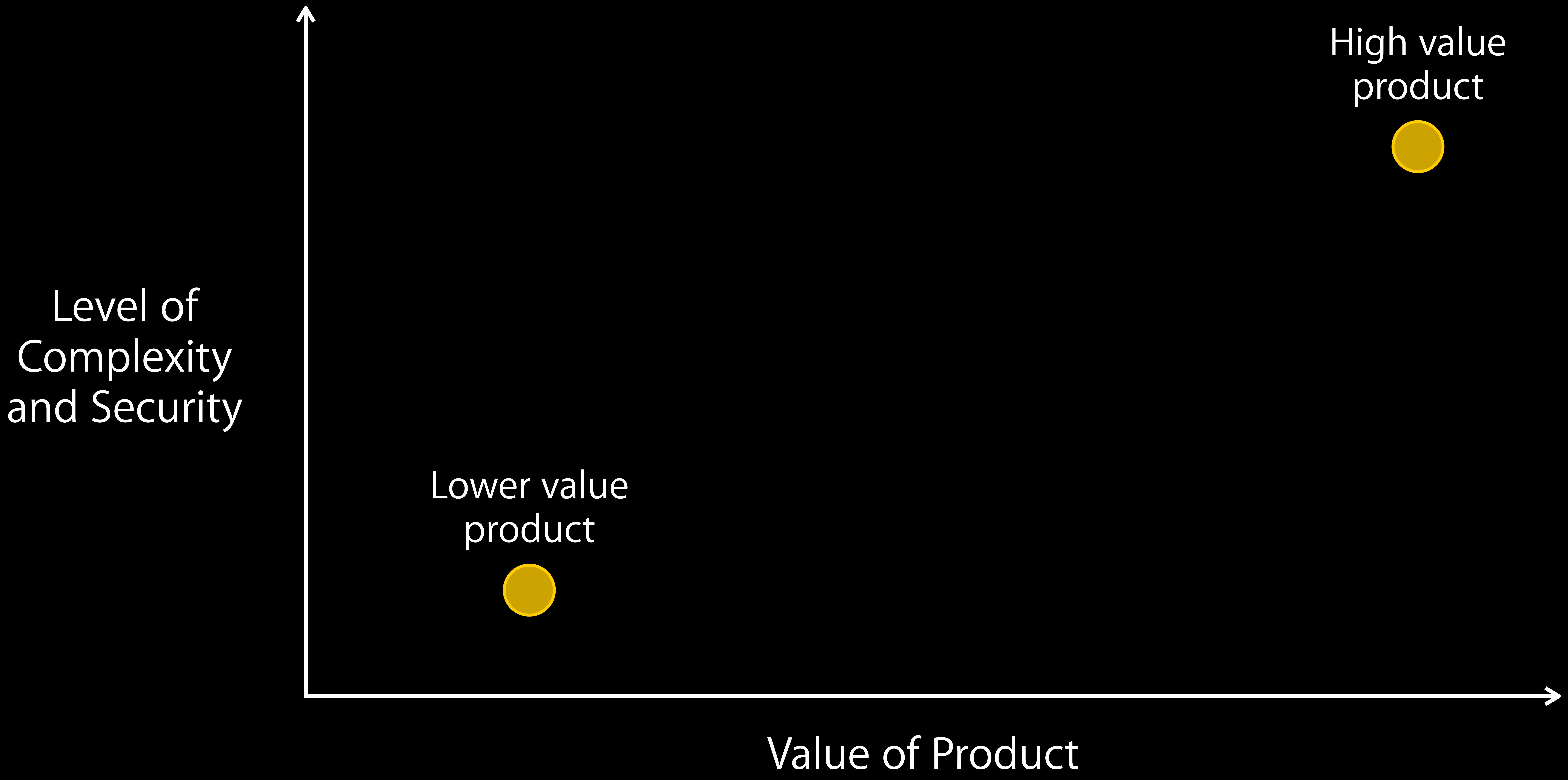
How to verify the device?

How to interpret the data in the receipt?

Level of Complexity and Security
Value of Product

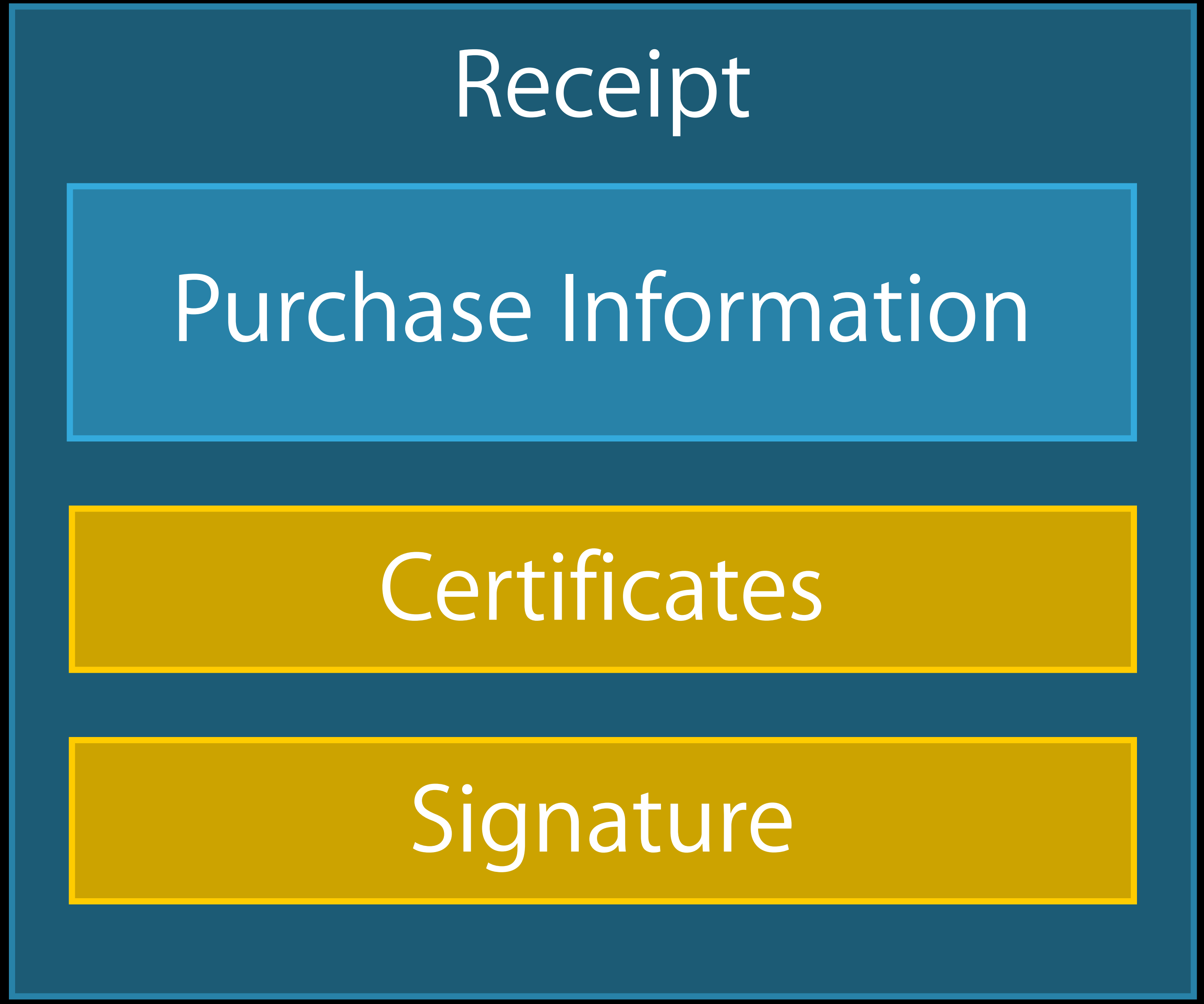

High value product
Level of Complexity and Security
Value of Product


Level of Complexity and Security
Value of Product
High value product


High value product
Level of Complexity and Security
Lower value product
Value of Product

Working with Receipts

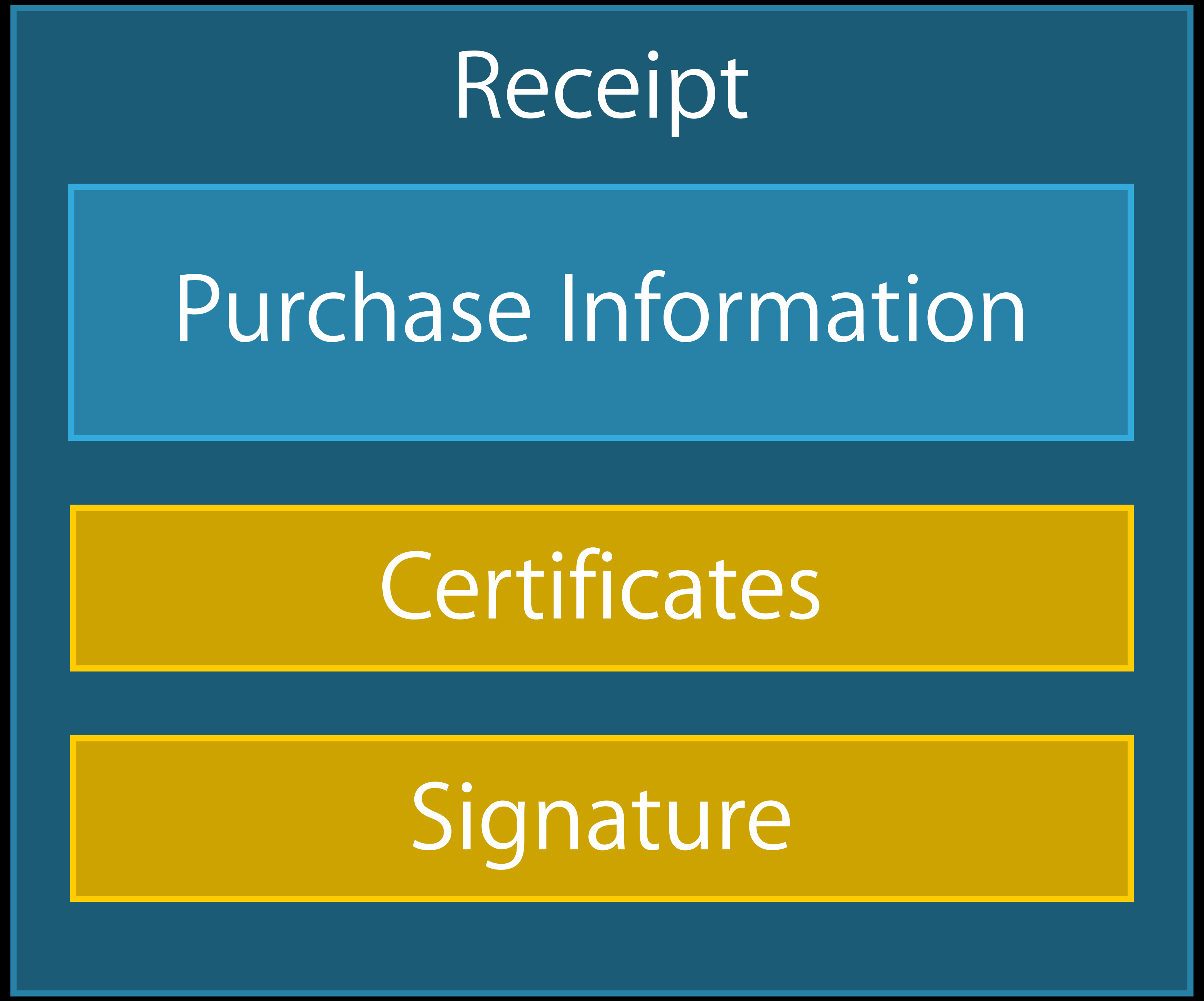
The Basics



The Basics

Stored in the App Bundle

- API to get the path



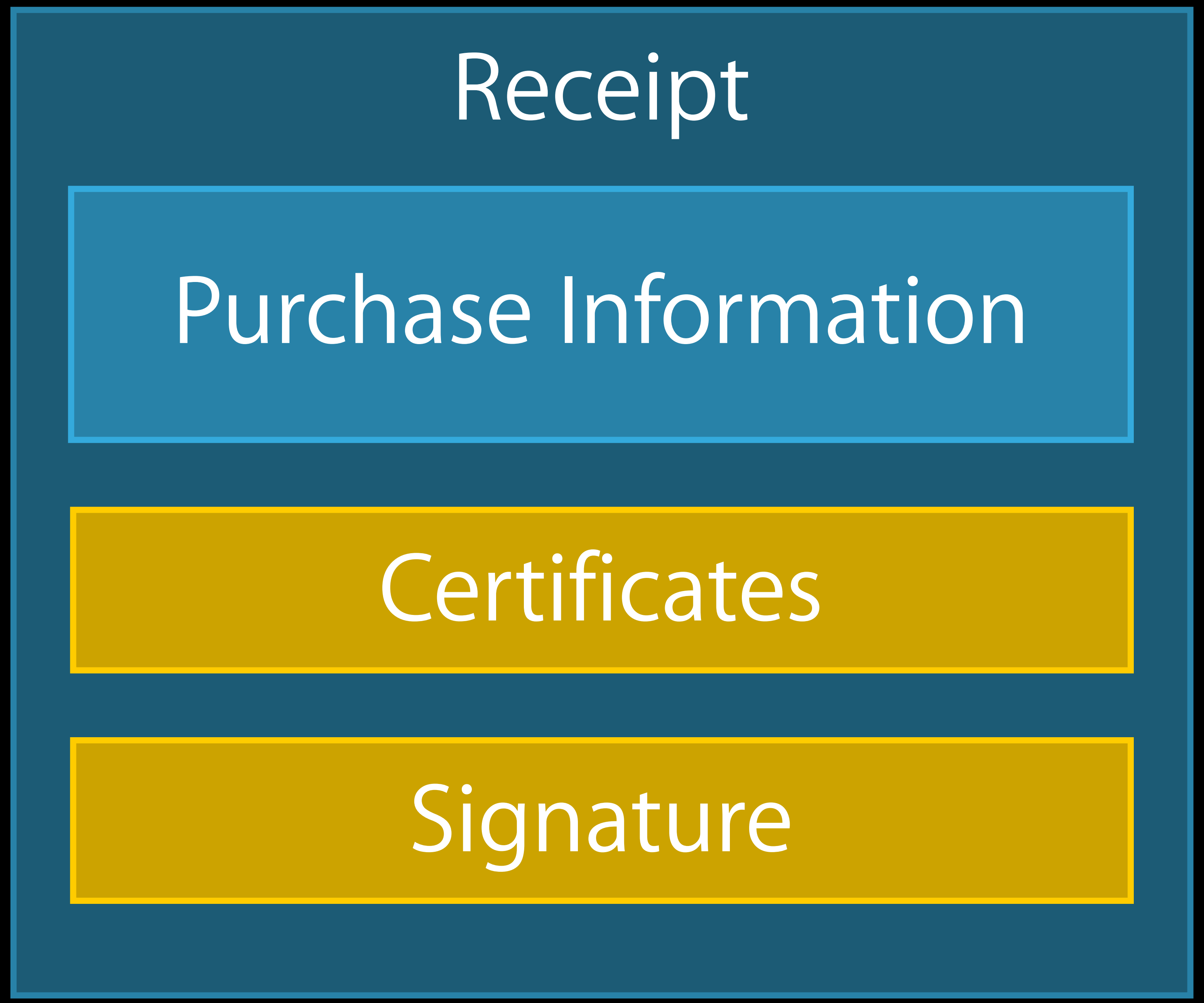
The Basics

Stored in the App Bundle

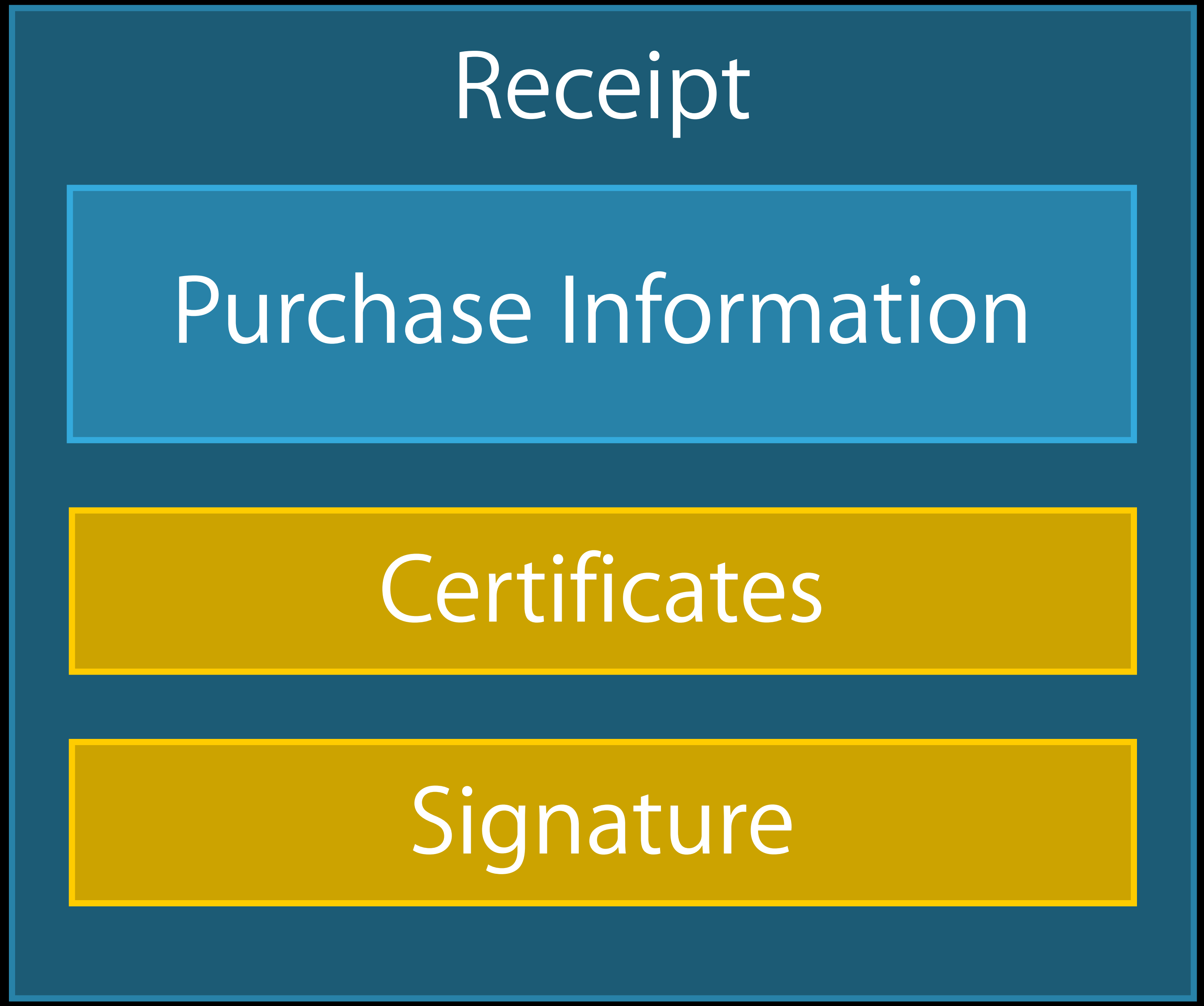
- API to get the path

Single file

- Purchase data
- Signature to check authenticity



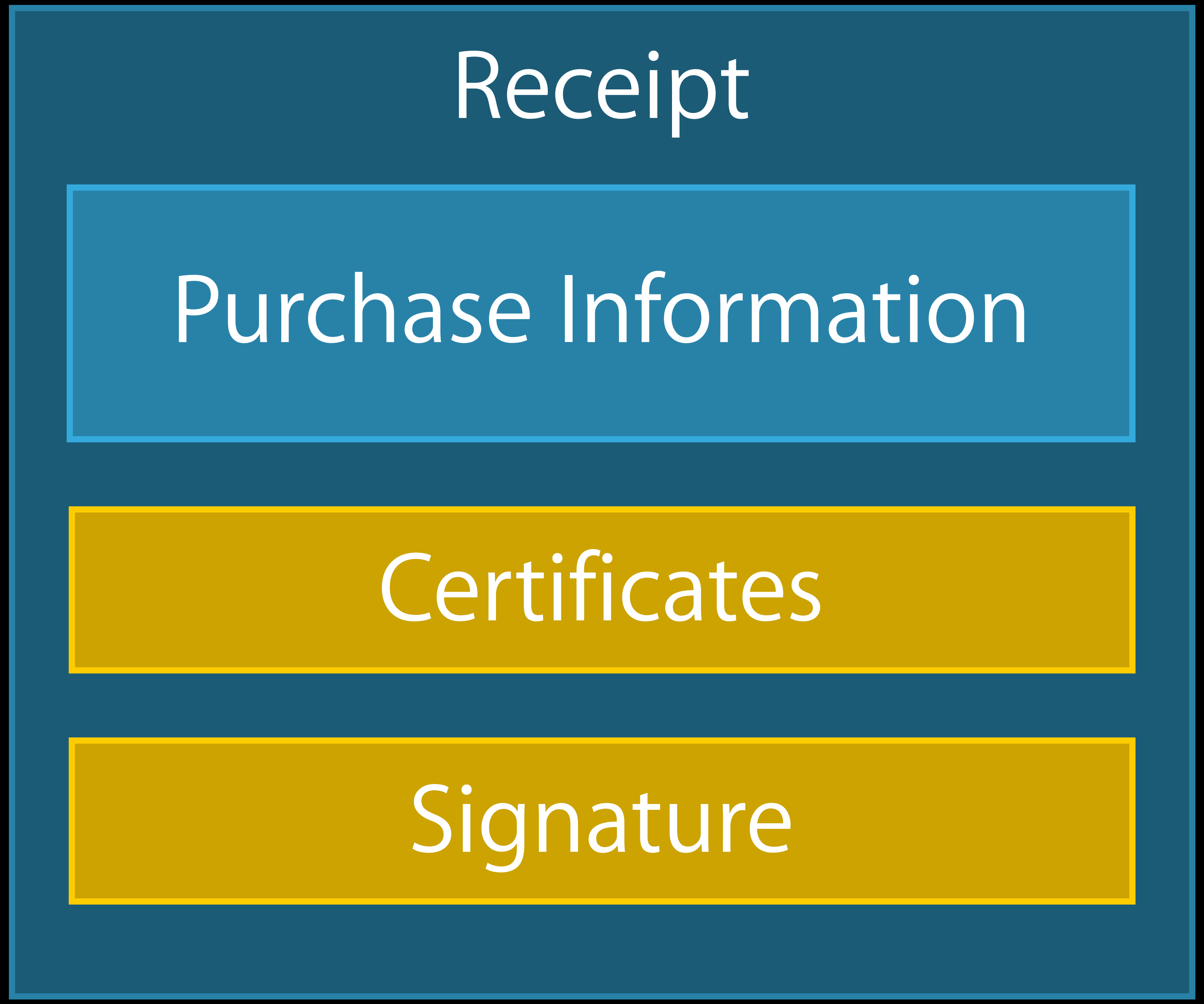
Standards



Standards

Signing

- PKCS#7 Cryptographic Container



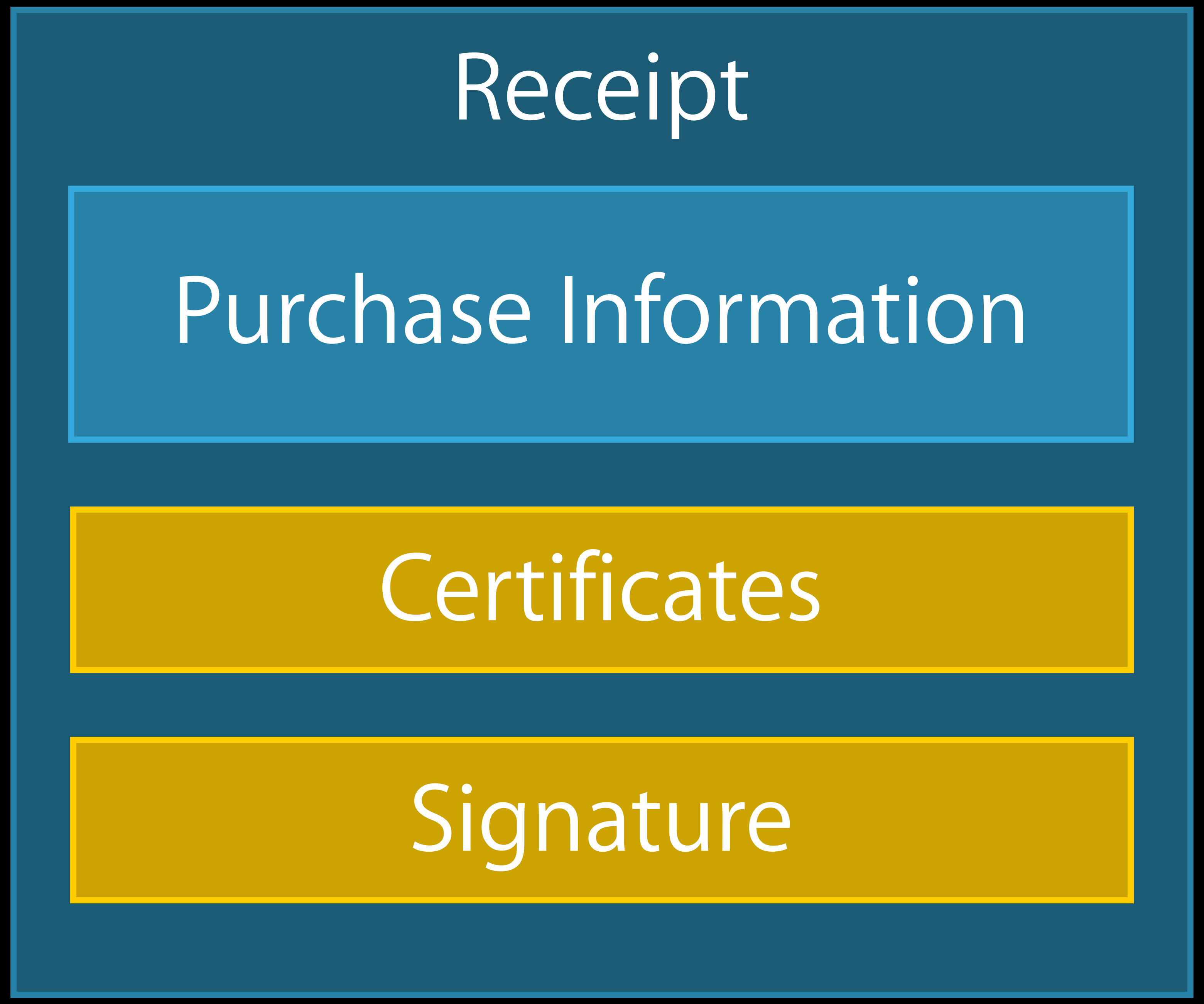
Standards

Signing

- PKCS#7 Cryptographic Container

Data Encoding

- ASN.1



Standards

Signing

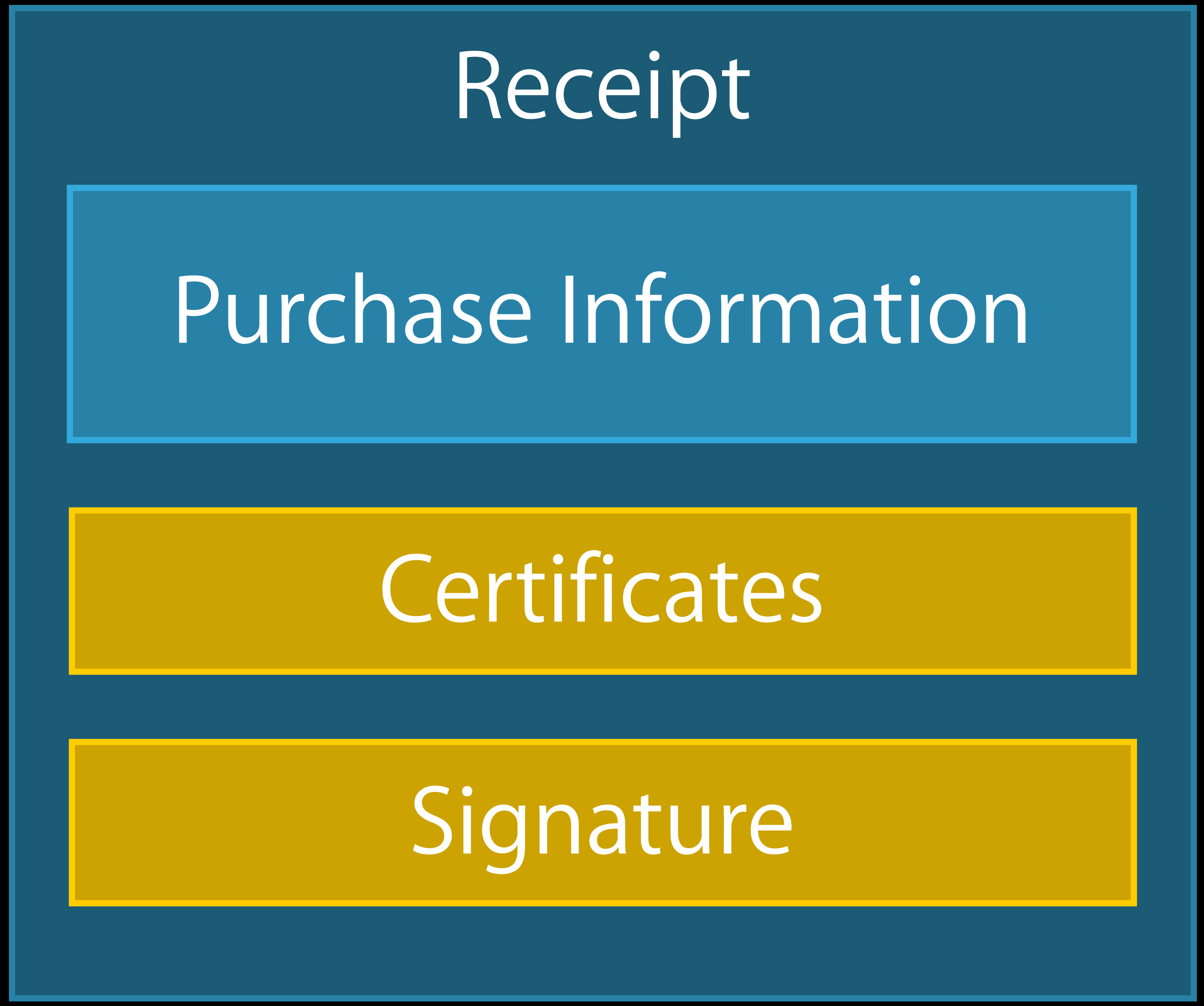
- PKCS#7 Cryptographic Container

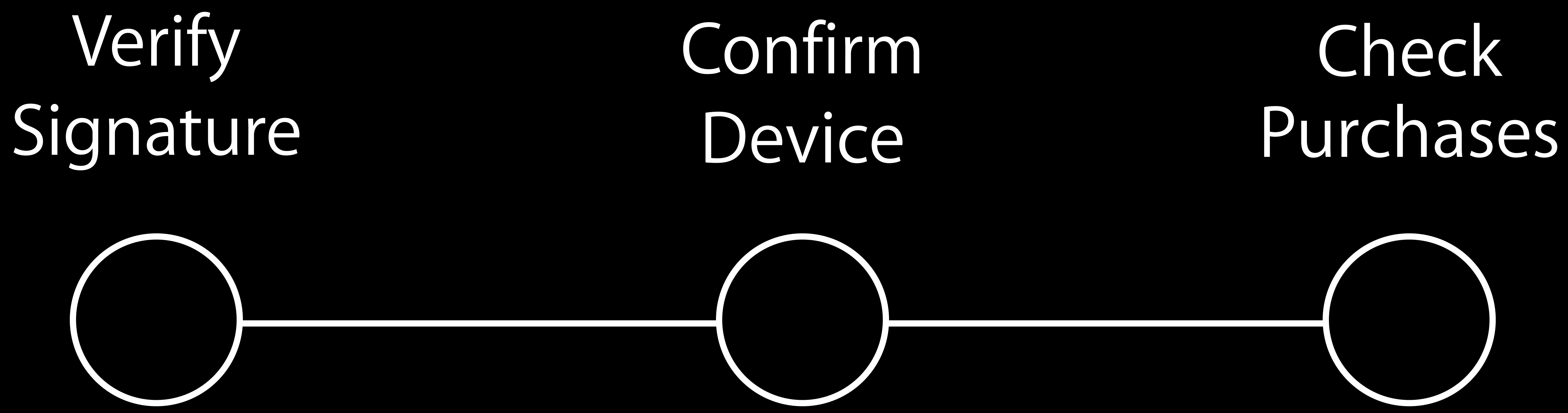
Data Encoding

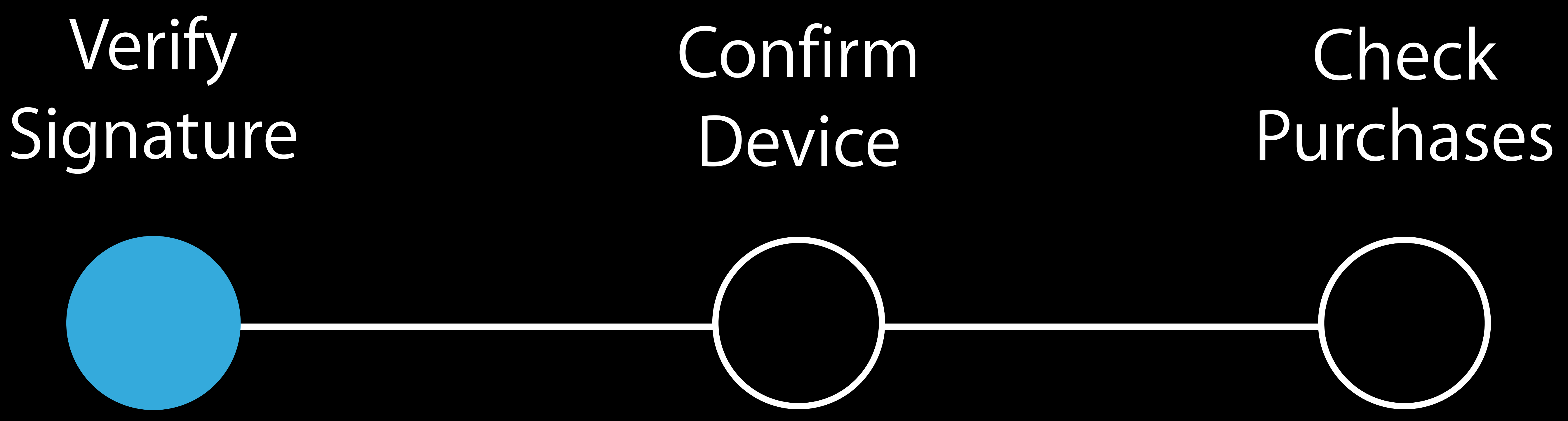
- ASN.1

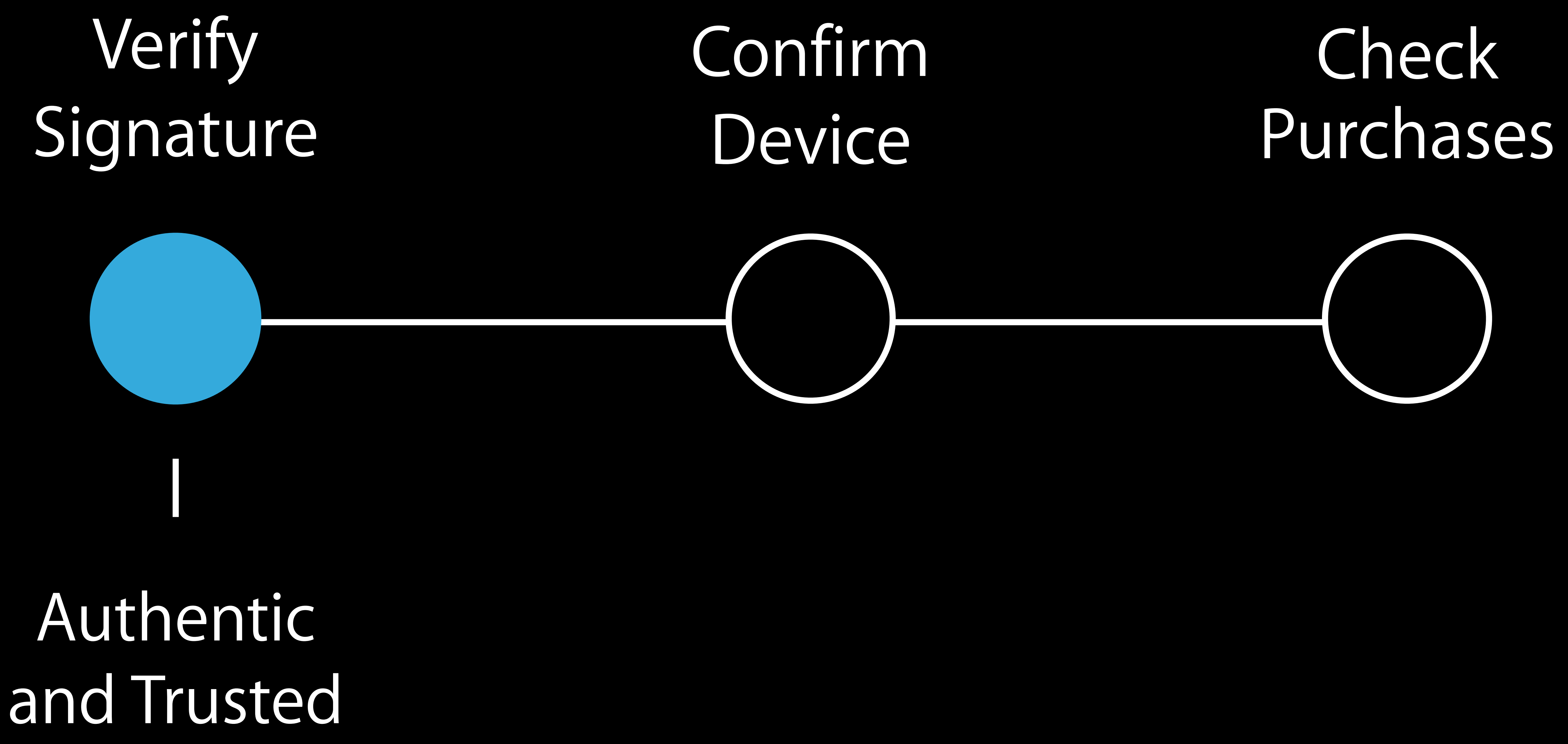
Options for verifying and reading

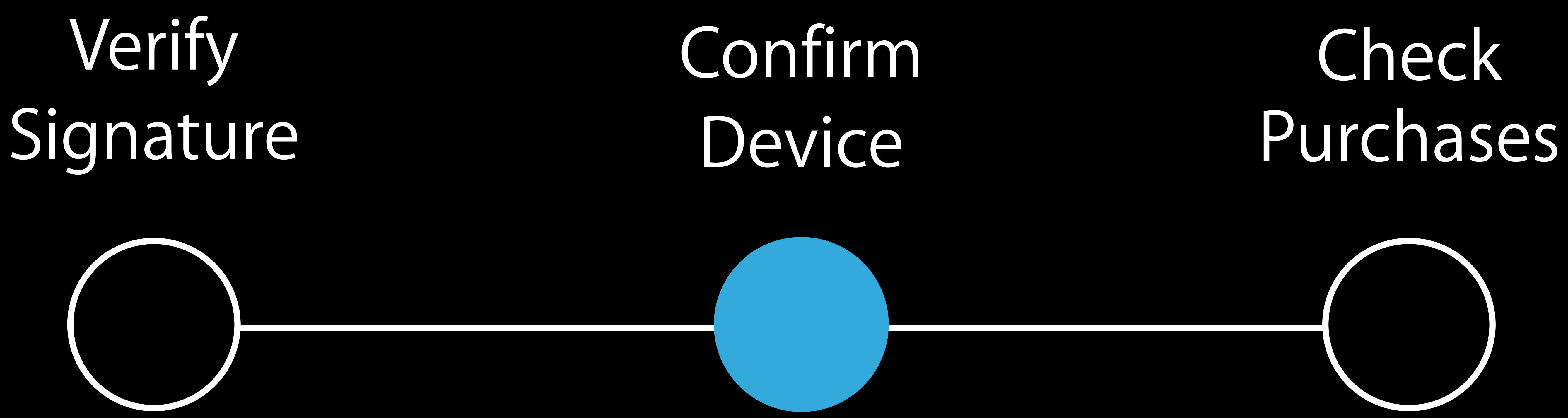
- OpenSSL, asn1c, etc.
- Create your own

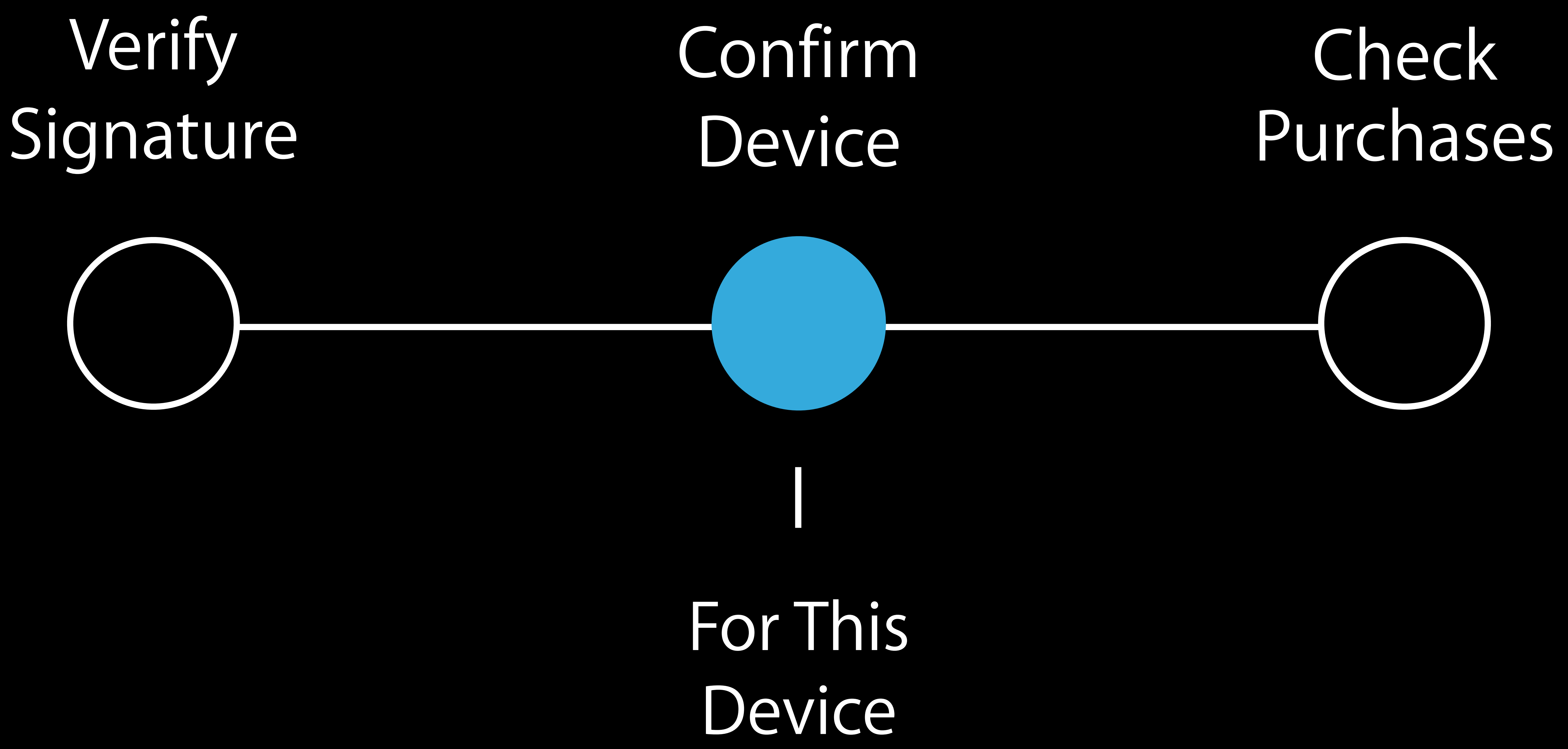


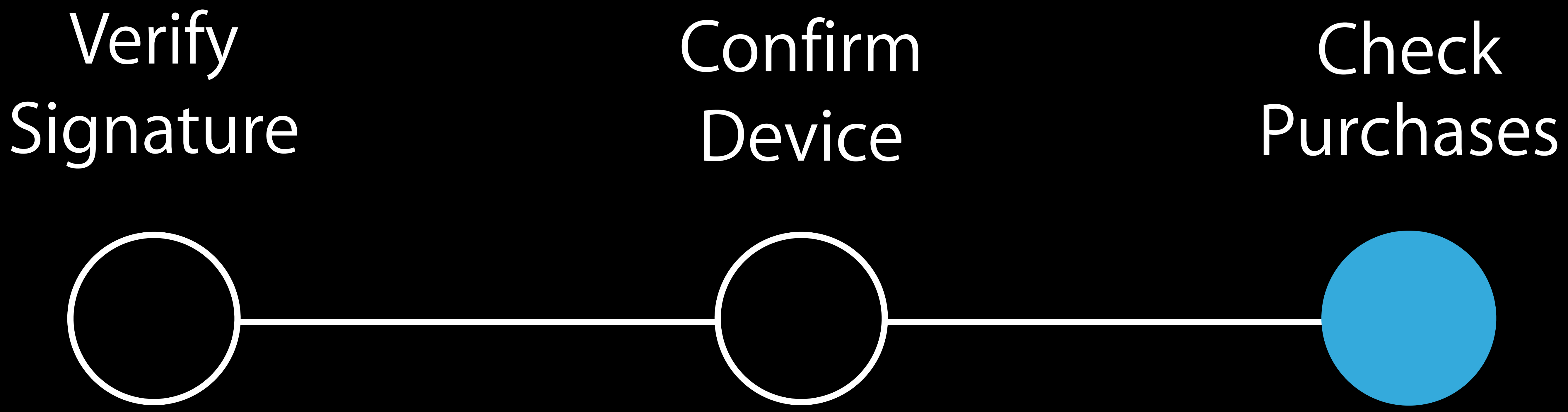

Verify Signature
Confirm Device
Check Purchases

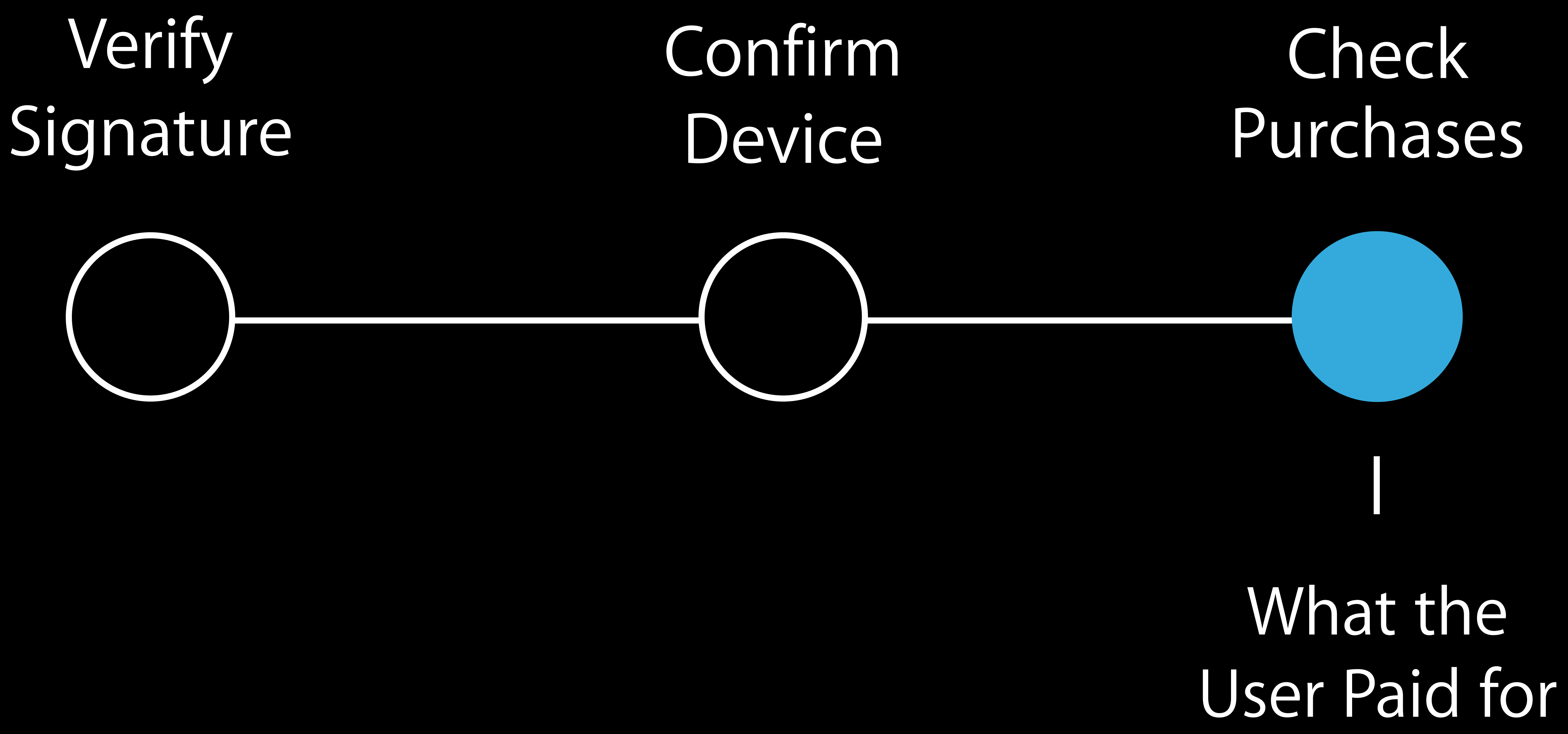

Verify Signature
Confirm Device
Check Purchases


Verify Signature
Confirm Device
Check Purchases
Authentic and Trusted


Verify Signature
Confirm Device
Check Purchases


Verify Signature
Confirm Device
Check Purchases
For This Device

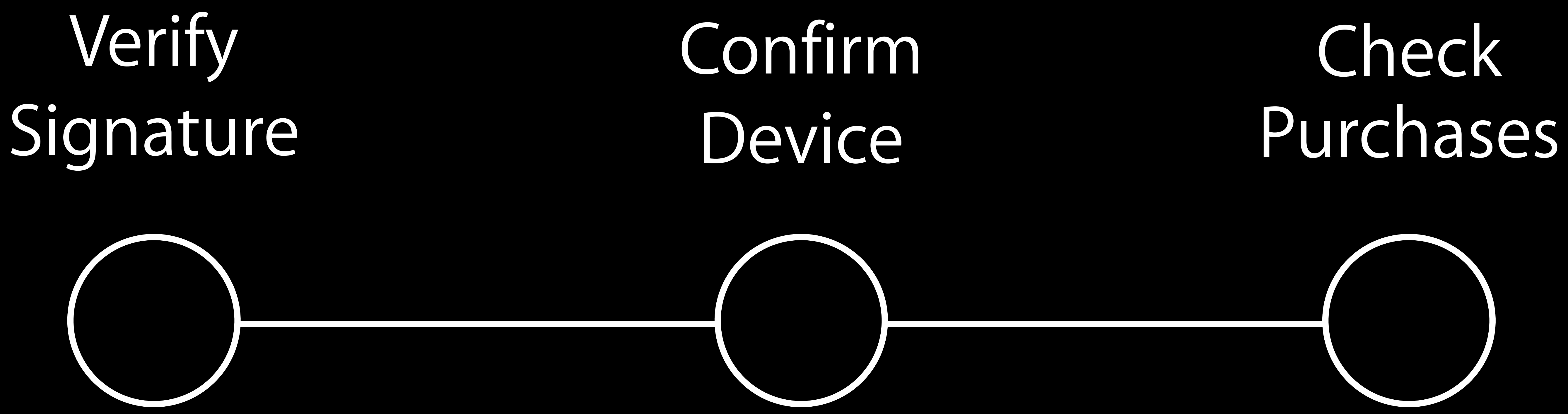

Verify Signature
Confirm Device
Check Purchases

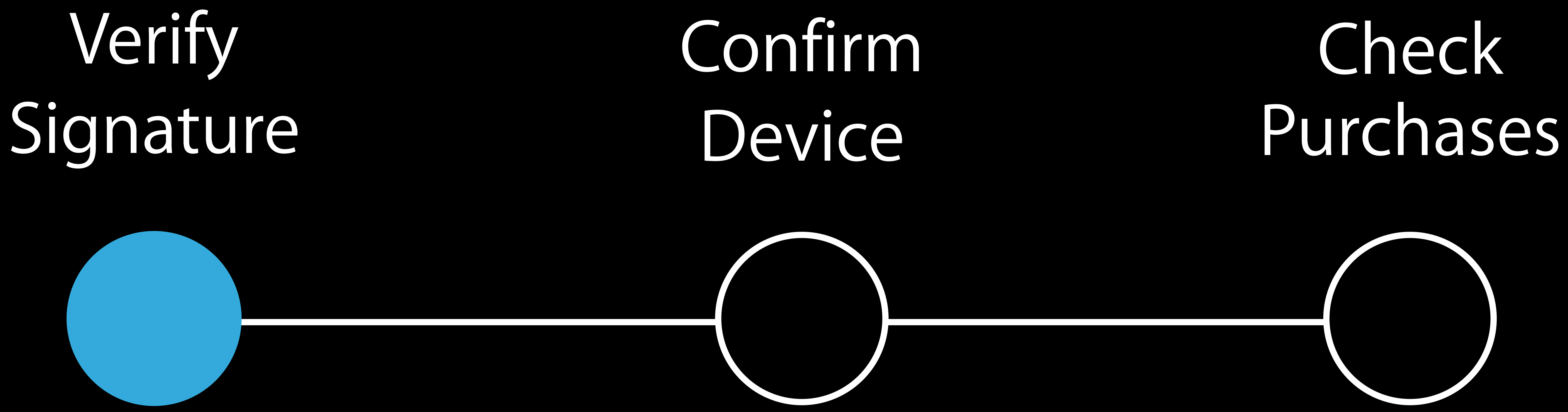

Verify Signature
Confirm Device
Check Purchases
What the User Paid for

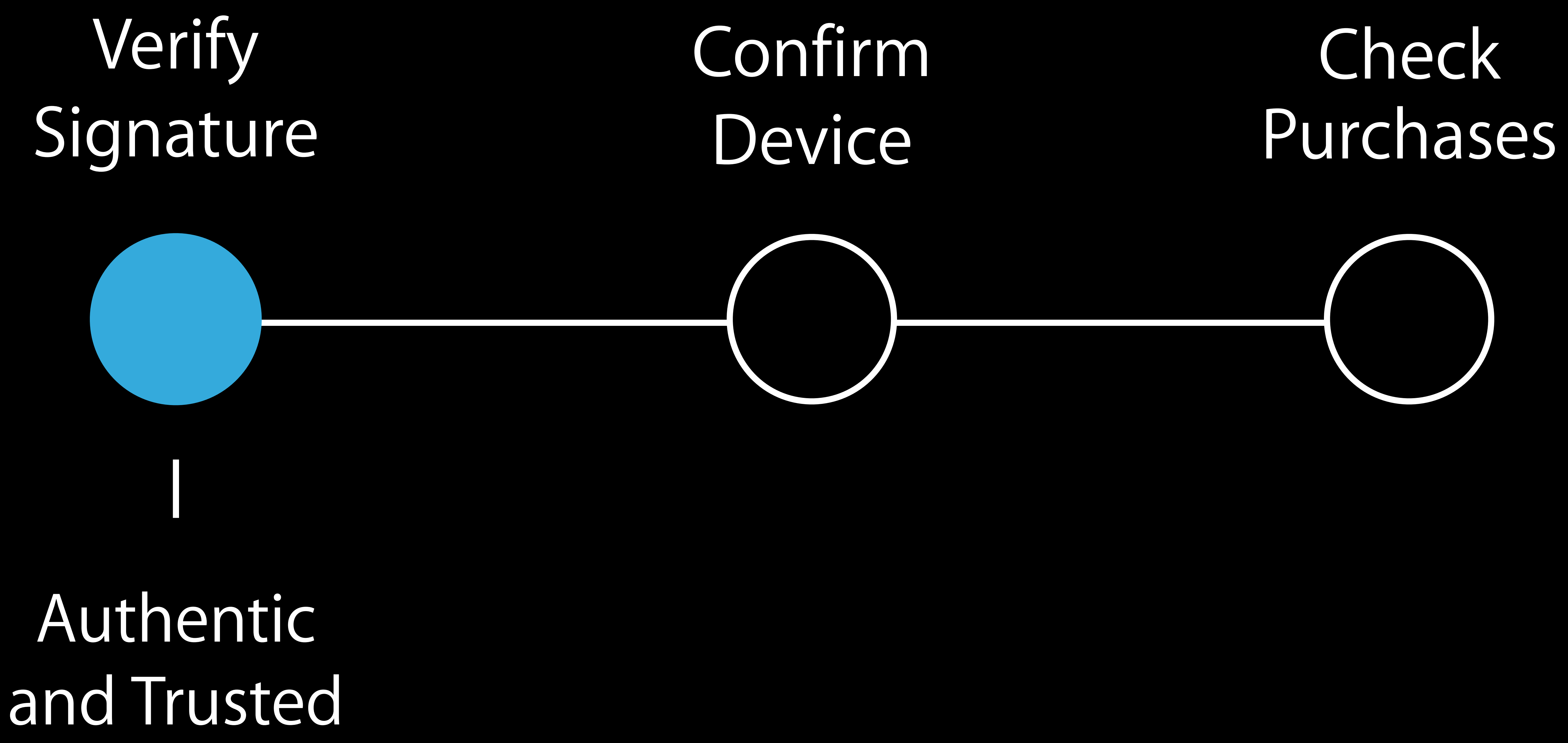
Demo

Getting a receipt

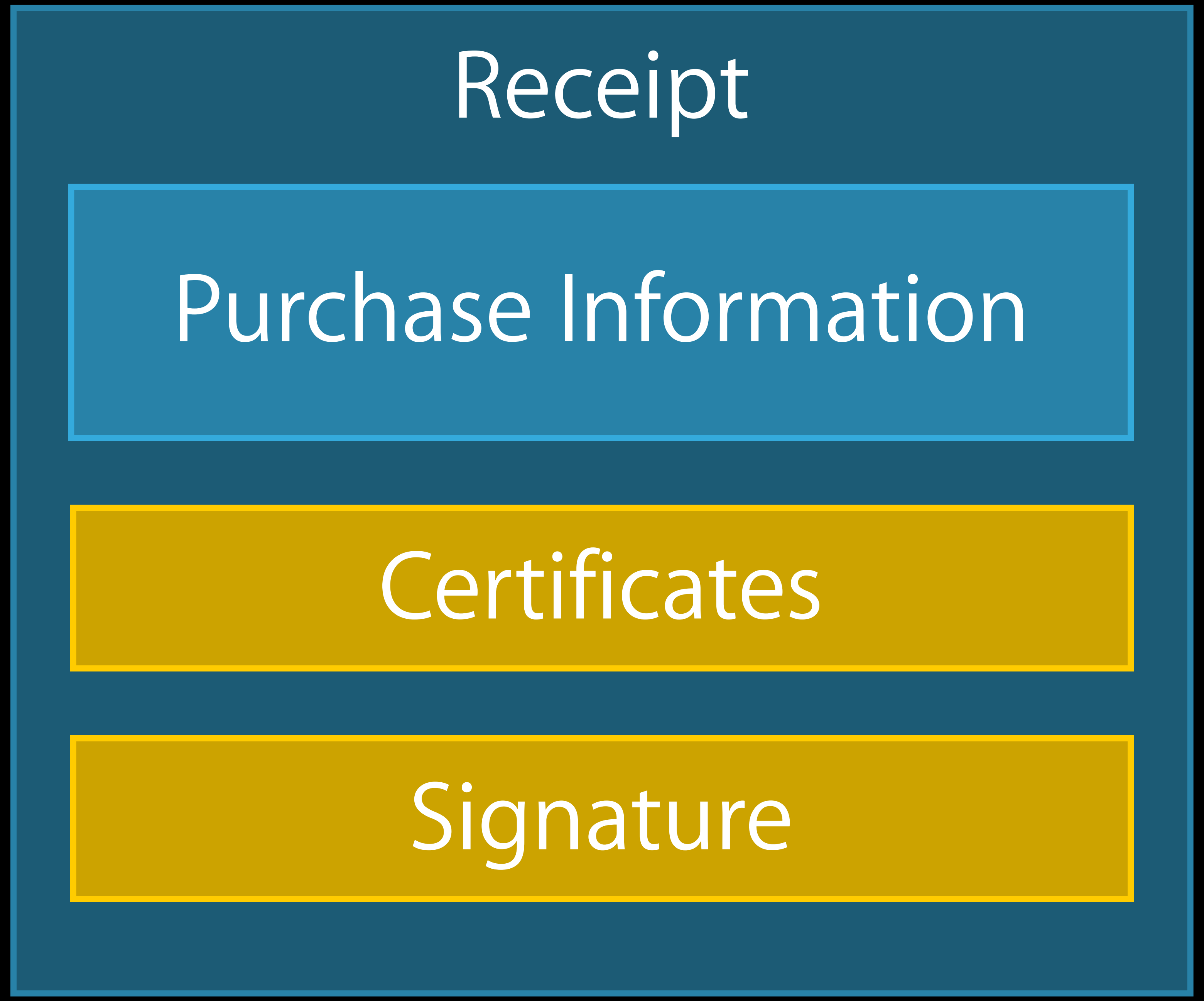
Validating on Device


Verify Signature
Confirm Device
Check Purchases


Verify Signature
Confirm Device
Check Purchases

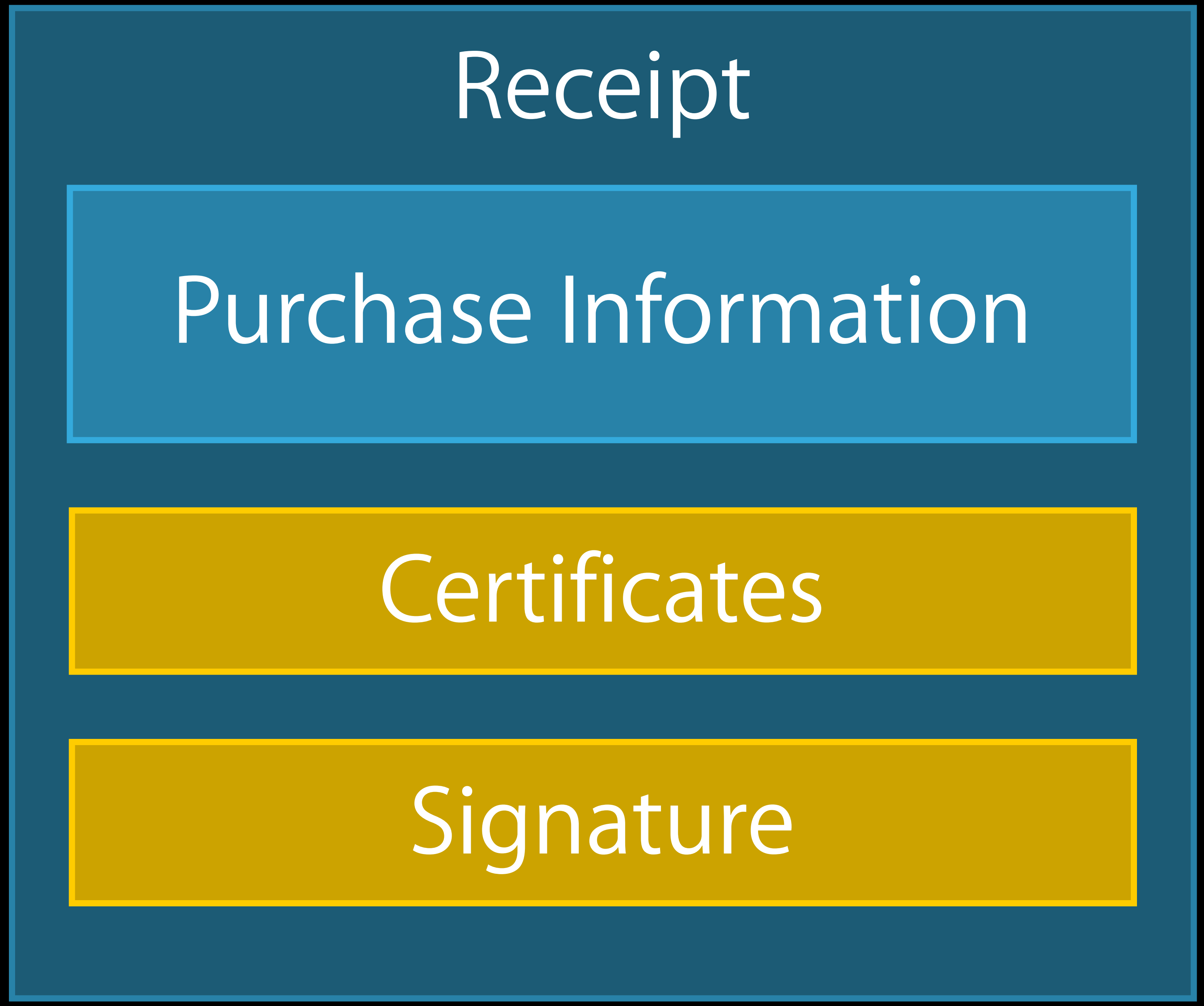

Verify Signature
Confirm Device
Check Purchases
Authentic and Trusted

Verify the Signature



Verify the Signature

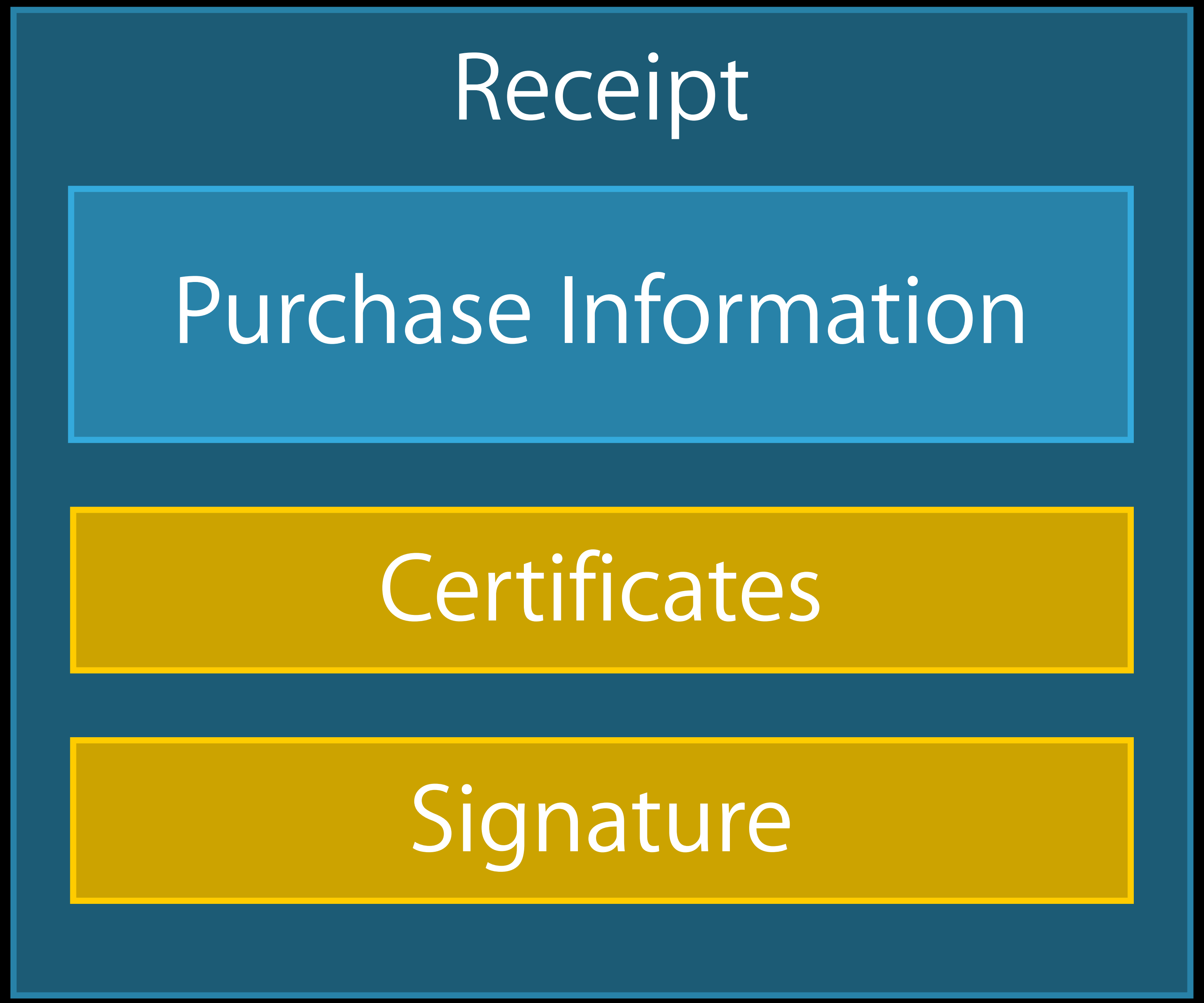
Confirms that the receipt



Verify the Signature

Confirms that the receipt

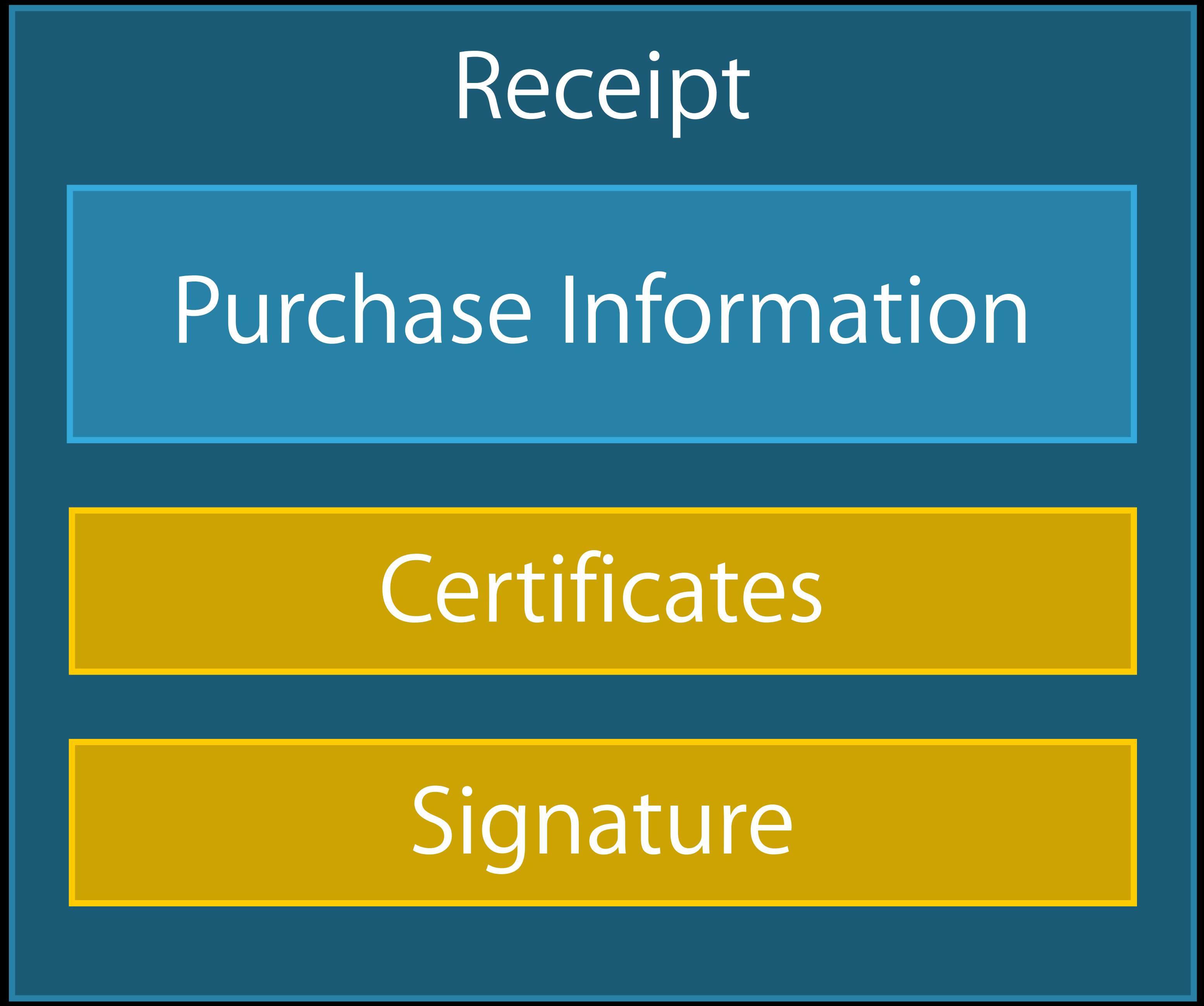
- Has not been altered



Verify the Signature

Confirms that the receipt

- Has not been altered
- Came from Apple

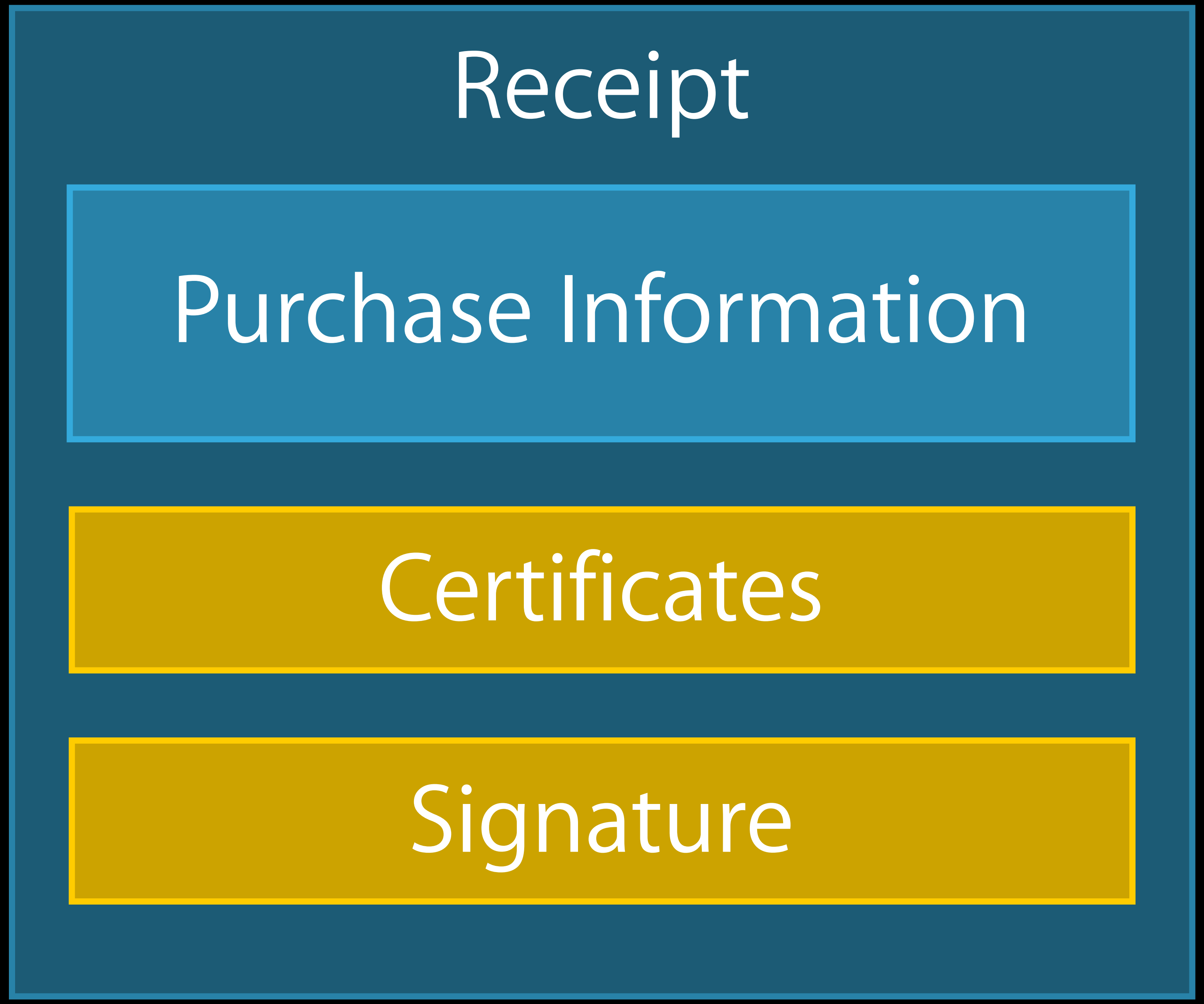


Verify the Signature

Confirms that the receipt

- Has not been altered
- Came from Apple

PKCS#7 Cryptographic Container



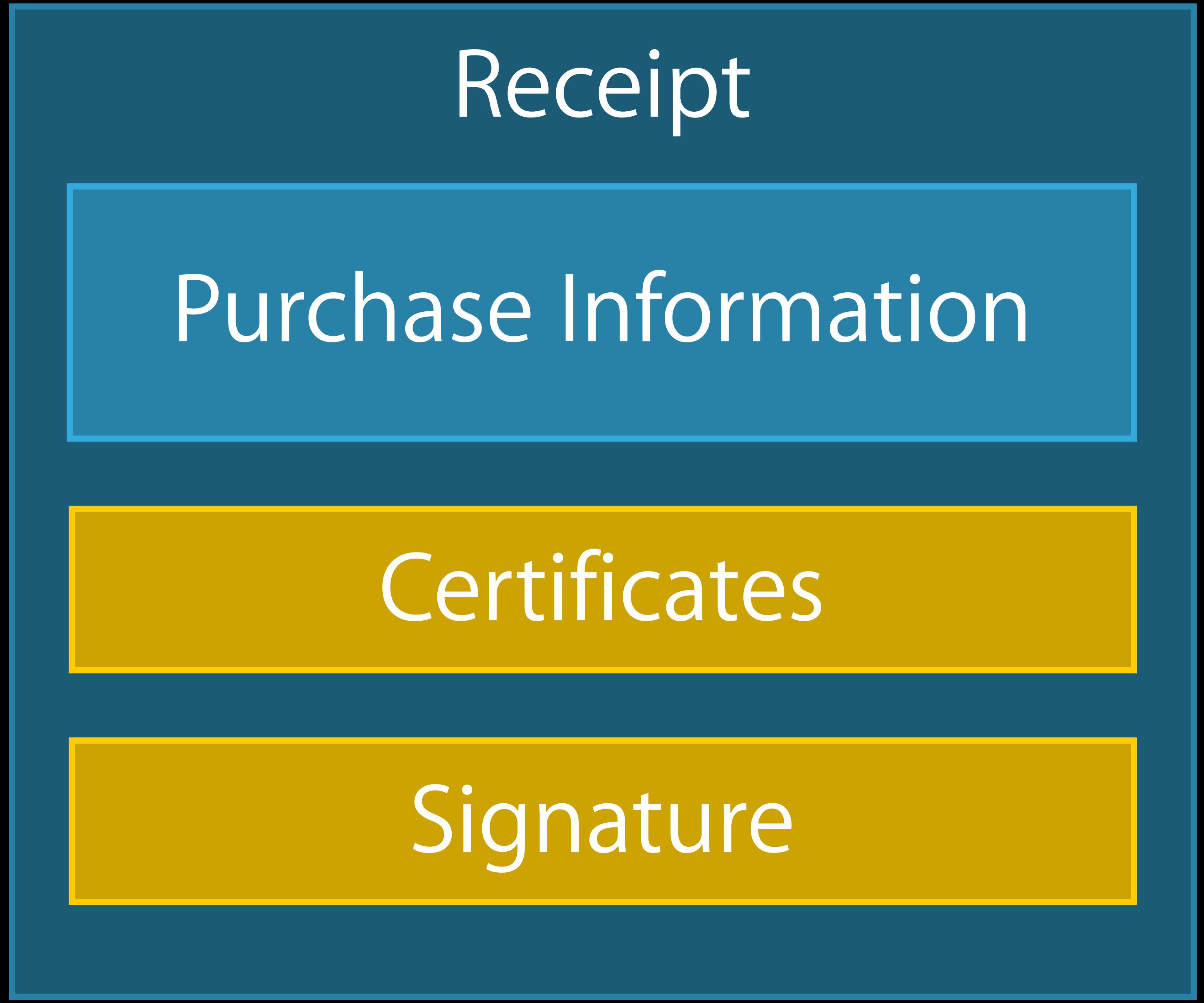
Verify the Signature

Confirms that the receipt

- Has not been altered
- Came from Apple

PKCS#7 Cryptographic Container

Options



Verify the Signature

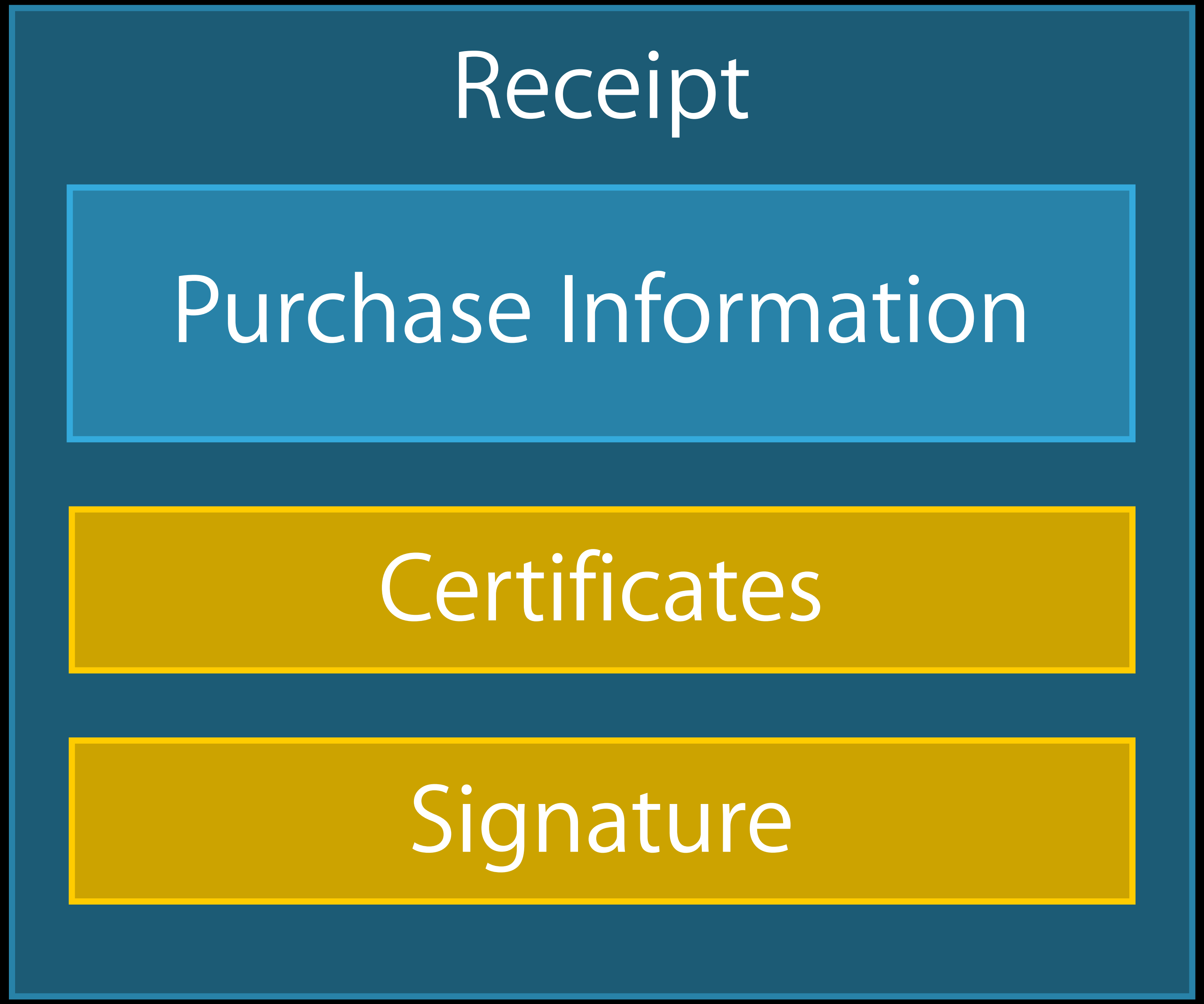
Confirms that the receipt

- Has not been altered
- Came from Apple

PKCS#7 Cryptographic Container

Options

- OpenSSL, other frameworks, etc.



Verify the Signature

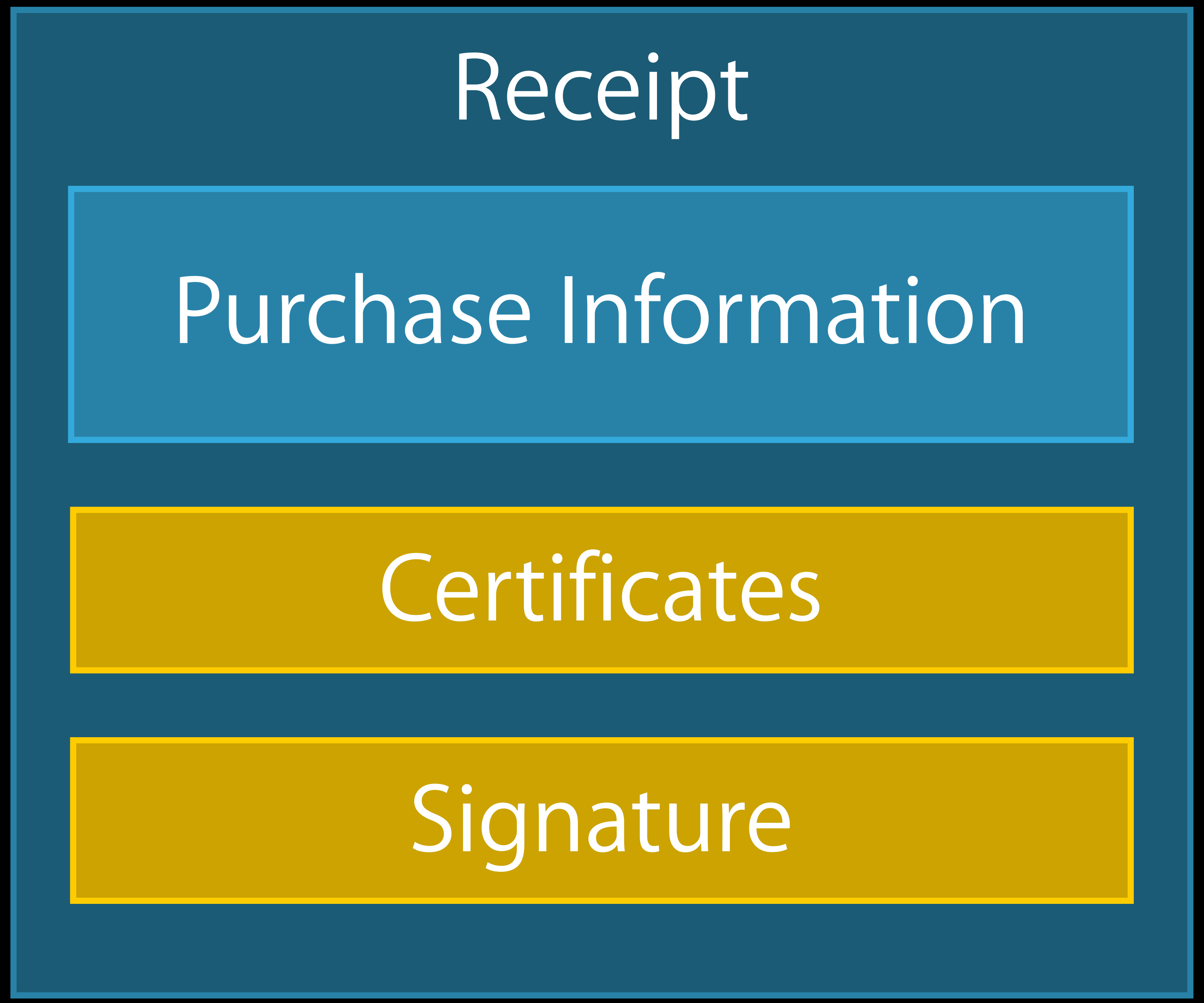
Confirms that the receipt

- Has not been altered
- Came from Apple

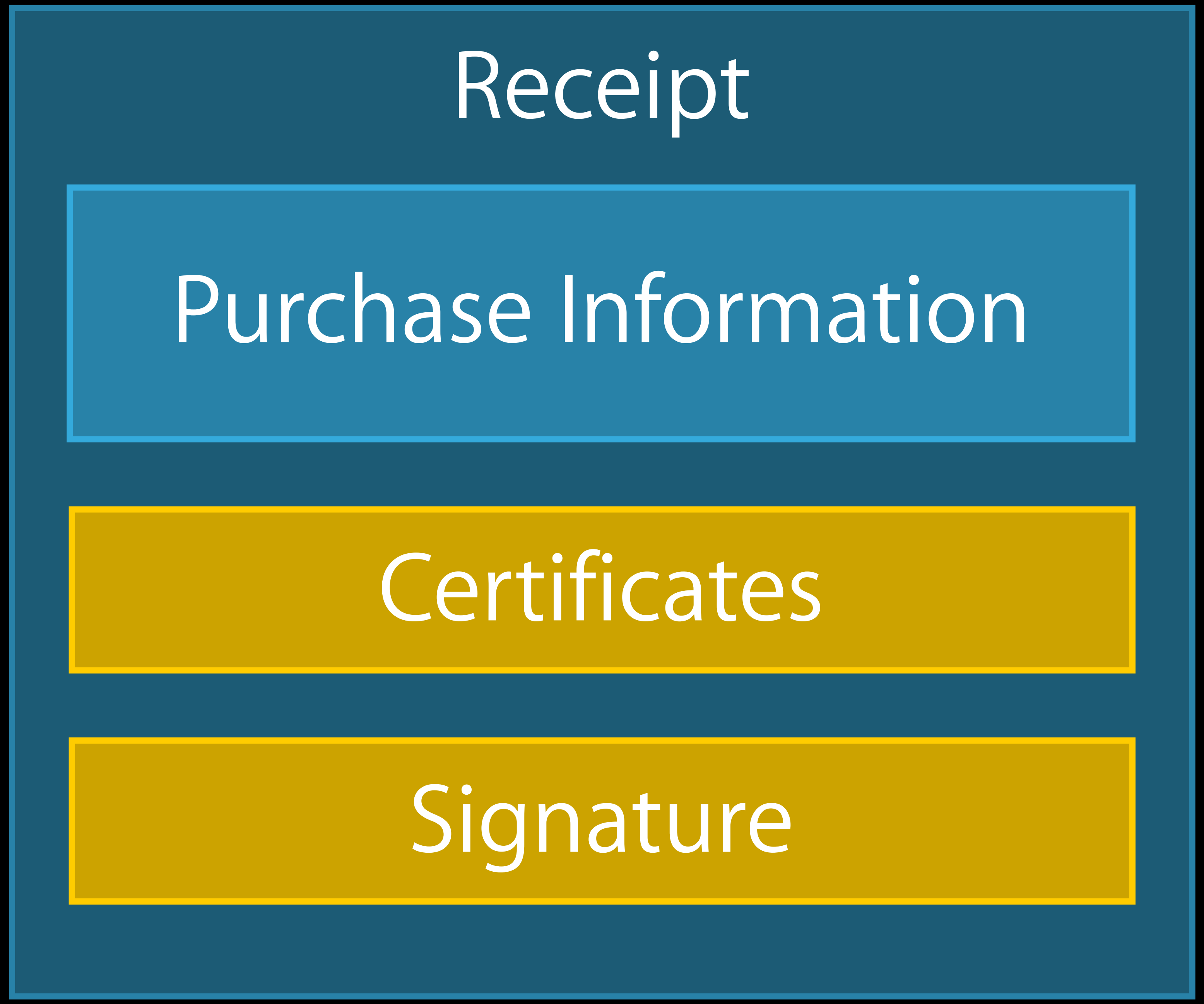
PKCS#7 Cryptographic Container

Options

- OpenSSL, other frameworks, etc.
- Custom implementation

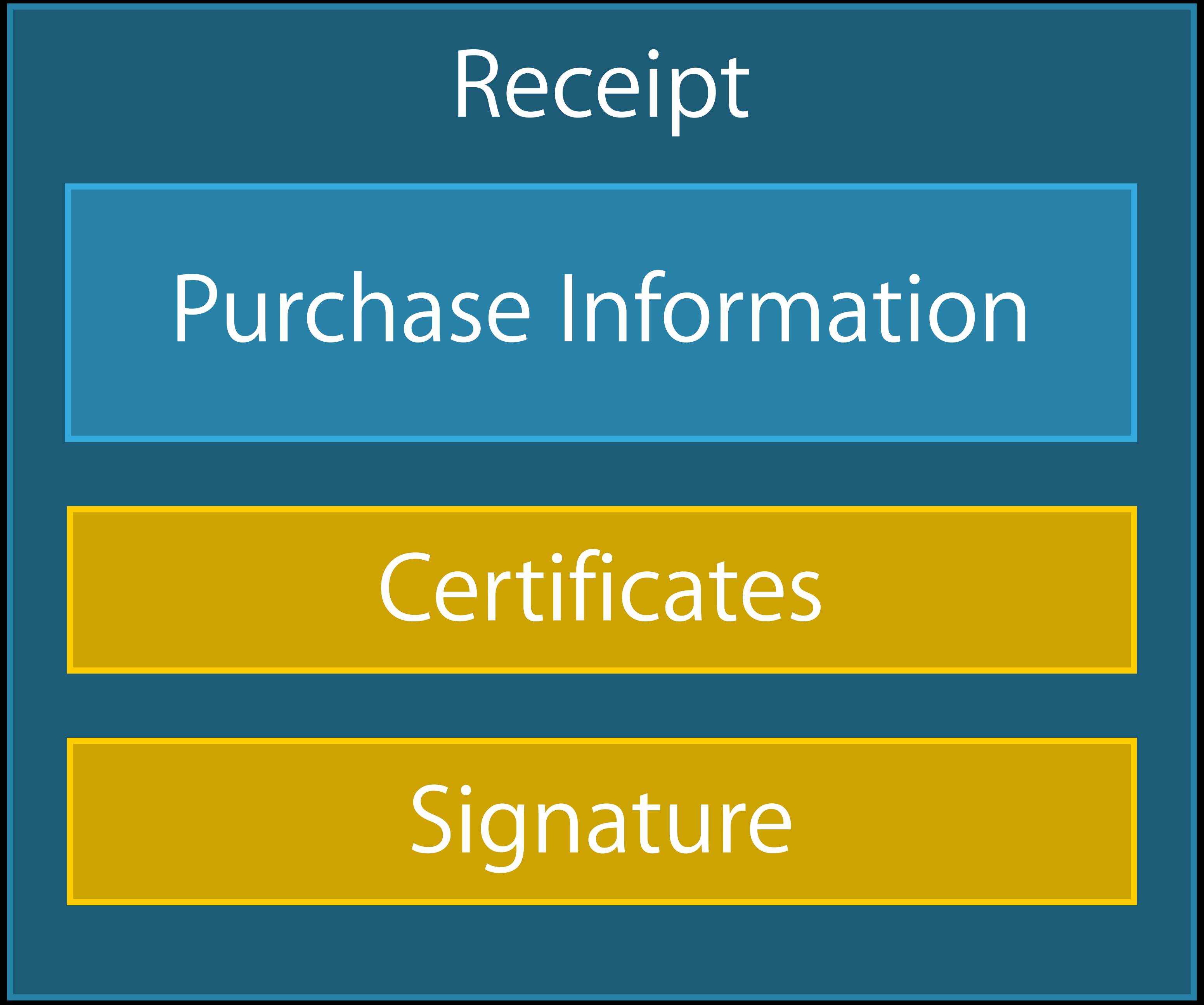


Getting Started



Getting Started

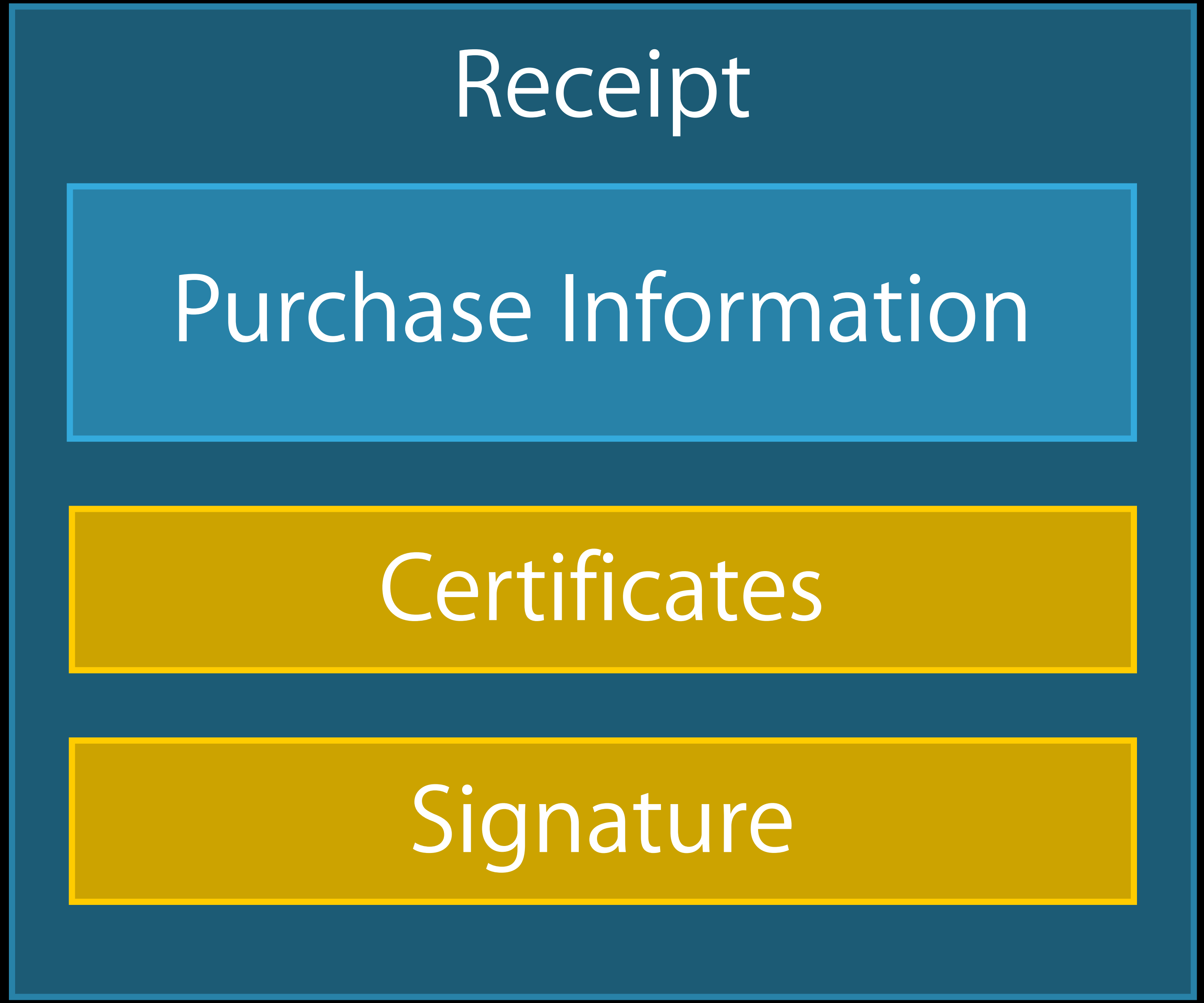
Locate the file



Getting Started

Locate the file

```
    // Locate the Receipt
    [[NSBundle mainBundle] appStoreReceiptURL];
```

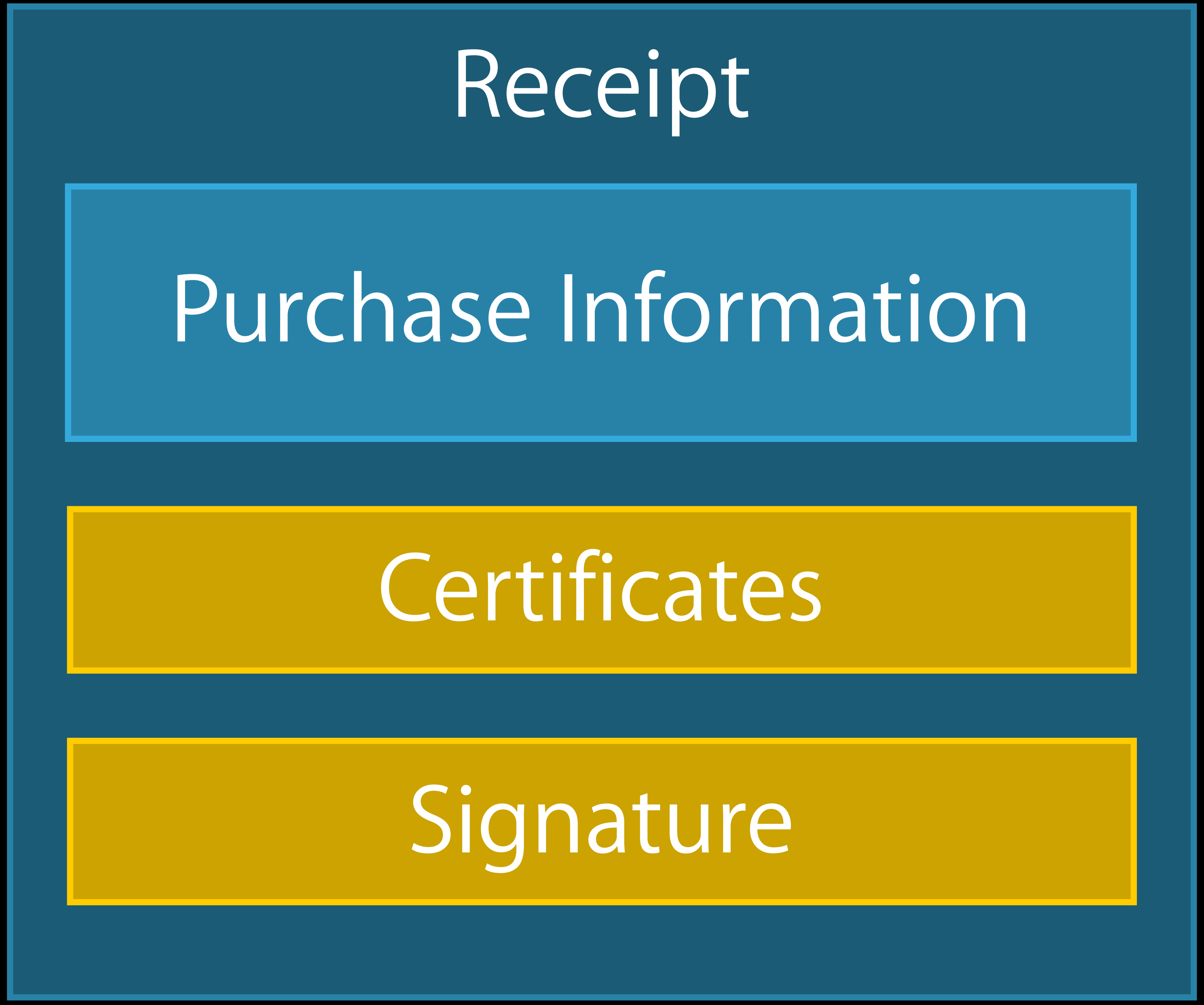


Getting Started

Locate the file

```
// Locate the Receipt
[[NSBundle mainBundle] appStoreReceiptURL];
```

Read the contents



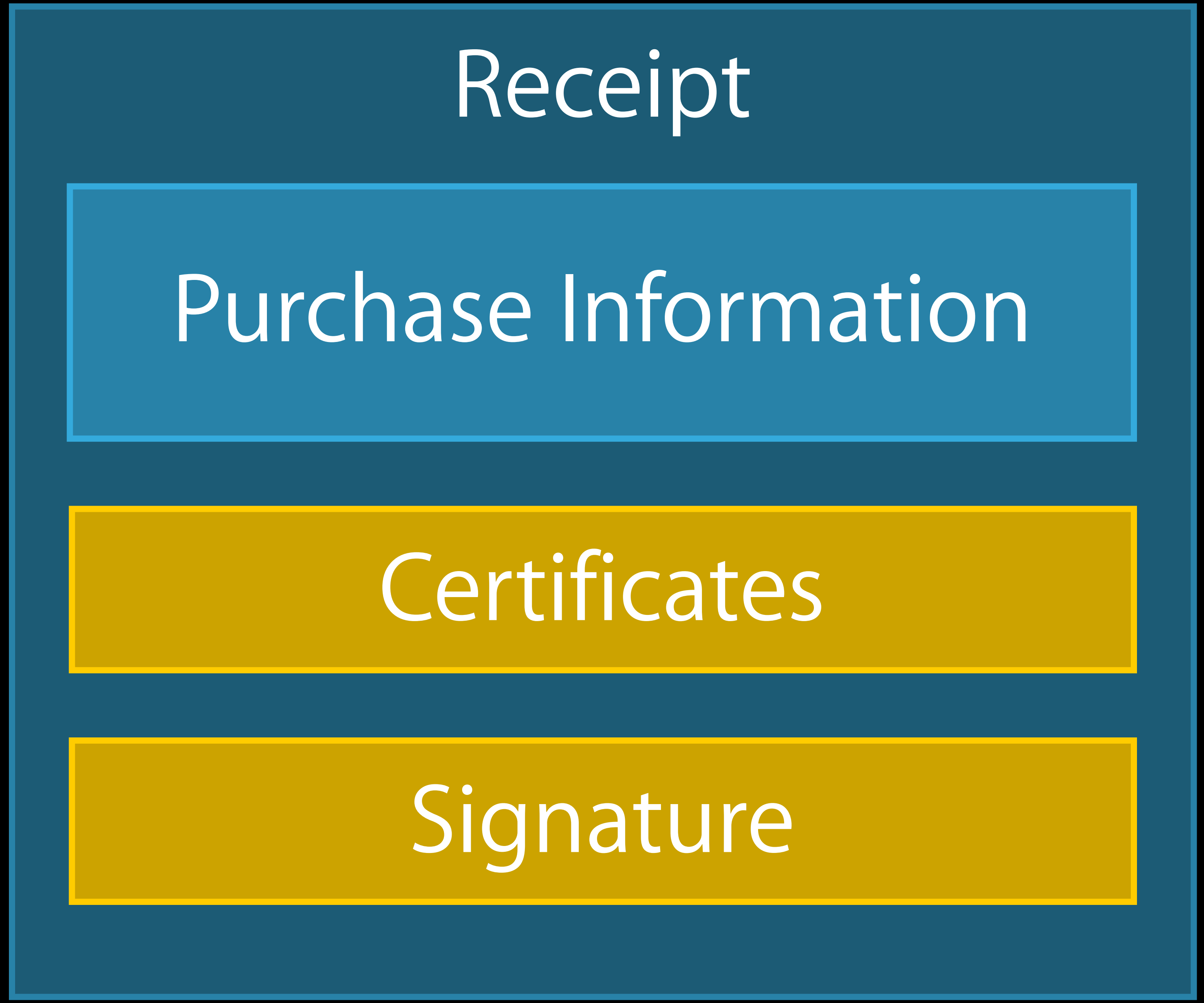
Getting Started

Locate the file

```
// Locate the Receipt
[[NSBundle mainBundle] appStoreReceiptURL];
```

Read the contents

```
// Read the receipt
[[NSData alloc] initWithContentsOfURL:]
```



OpenSSL Example

```
BIO *b_receipt;
BIO *b_x509;
```

Load the Receipt and Apple Root CA Certificate
Binary data from receipt plus certificate

OpenSSL Example

```
BIO *b_receipt;
BIO *b_x509;
```

Load the Receipt and Apple Root CA Certificate
Binary data from receipt plus certificate

```
// Convert receipt data to PKCS #7 Representation
PKCS7 *p7 = d2i_PKCS7_bio(b_receipt, NULL);
```

OpenSSL Example

```
BIO *b_receipt;
BIO *b_x509;
```

Load the Receipt and Apple Root CA Certificate
Binary data from receipt plus certificate

```
// Convert receipt data to PKCS #7 Representation
PKCS7 *p7 = d2i_PKCS7_bio(b_receipt, NULL);

// Create the certificate store
X509_STORE *store = X509_STORE_new();
X509 *appleRootCA = d2i_X509_bio(b_x509, NULL);
X509_STORE_add_cert(store, appleRootCA);
```

OpenSSL Example

```
BIO *b_receipt;
BIO *b_x509;
```

Load the Receipt and Apple Root CA Certificate
Binary data from receipt plus certificate

```
// Convert receipt data to PKCS #7 Representation
PKCS7 *p7 = d2i_PKCS7_bio(b_receipt, NULL);

// Create the certificate store
X509_STORE *store = X509_STORE_new();
X509 *appleRootCA = d2i_X509_bio(b_x509, NULL);
X509_STORE_add_cert(store, appleRootCA);

// Verify the Signature
BIO *b_receiptPayload = BIO_new(BIO_s_mem());
int result = PKCS7_verify(p7, NULL, store, NULL, b_receiptPayload, 0);
if (result == 1)
{
    // Receipt Signature is VALID

}
```

OpenSSL Example

```
BIO *b_receipt;
BIO *b_x509;
```

Load the Receipt and Apple Root CA Certificate
Binary data from receipt plus certificate

```
// Convert receipt data to PKCS #7 Representation
PKCS7 *p7 = d2i_PKCS7_bio(b_receipt, NULL);

// Create the certificate store
X509_STORE *store = X509_STORE_new();
X509 *appleRootCA = d2i_X509_bio(b_x509, NULL);
X509_STORE_add_cert(store, appleRootCA);

// Verify the Signature
BIO *b_receiptPayload = BIO_new(BIO_s_mem());
int result = PKCS7_verify(p7, NULL, store, NULL, b_receiptPayload, 0);
if (result == 1)
{
    // Receipt Signature is VALID
    // b_receiptPayload contains the payload
}
```

Demo

Using OpenSSL for signature verification

Building OpenSSL

Building OpenSSL

Build a static library (.a file), not a dynamic library

Building OpenSSL

Build a static library (.a file), not a dynamic library

Building for multiple architectures

Building OpenSSL

Build a static library (.a file), not a dynamic library

Building for multiple architectures

- Separate .a per architecture (e.g. arm, arm64, x86_64, etc.)

Building OpenSSL

Build a static library (.a file), not a dynamic library

Building for multiple architectures

- Separate .a per architecture (e.g. arm, arm64, x86_64, etc.)
- Use ‘lipo’ to create a single .a with multiple arch slices

Building OpenSSL

Build a static library (.a file), not a dynamic library

Building for multiple architectures

- Separate .a per architecture (e.g. arm, arm64, x86_64, etc.)
- Use 'lipo' to create a single .a with multiple arch slices

Configuration script

Building OpenSSL

Build a static library (.a file), not a dynamic library

Building for multiple architectures

- Separate .a per architecture (e.g. arm, arm64, x86_64, etc.)
- Use 'lipo' to create a single .a with multiple arch slices

Configuration script

- Use darwin64-x86_64-cc host type for OS X 64bit

Building OpenSSL

Build a static library (.a file), not a dynamic library

Building for multiple architectures

- Separate .a per architecture (e.g. arm, arm64, x86_64, etc.)
- Use ‘lipo’ to create a single .a with multiple arch slices

Configuration script

- Use darwin64-x86_64-cc host type for OS X 64bit
- Use iphoneos-cross host type for iOS

Building OpenSSL

Build a static library (.a file), not a dynamic library

Building for multiple architectures

- Separate .a per architecture (e.g. arm, arm64, x86_64, etc.)
- Use 'lipo' to create a single .a with multiple arch slices

Configuration script

- Use darwin64-x86_64-cc host type for OS X 64bit
- Use iphoneos-cross host type for iOS

Lots of examples available online…

Verification

Verification

Do not check the expiry date on the certificate

Verification

Do not check the expiry date on the certificate

Do evaluate trust up to Root CA

Examples and Samples

Examples and Samples

Convenience comes at a price

Examples and Samples

Convenience comes at a price

- Re-using code brings with it bugs and vulnerabilities

Examples and Samples

Convenience comes at a price

- Re-using code brings with it bugs and vulnerabilities
- Single exploit affects many

Examples and Samples

Convenience comes at a price

- Re-using code brings with it bugs and vulnerabilities
- Single exploit affects many

It's *your* revenue stream

Examples and Samples

Convenience comes at a price

- Re-using code brings with it bugs and vulnerabilities
- Single exploit affects many

It’s your revenue stream

- Make decisions that suit your product

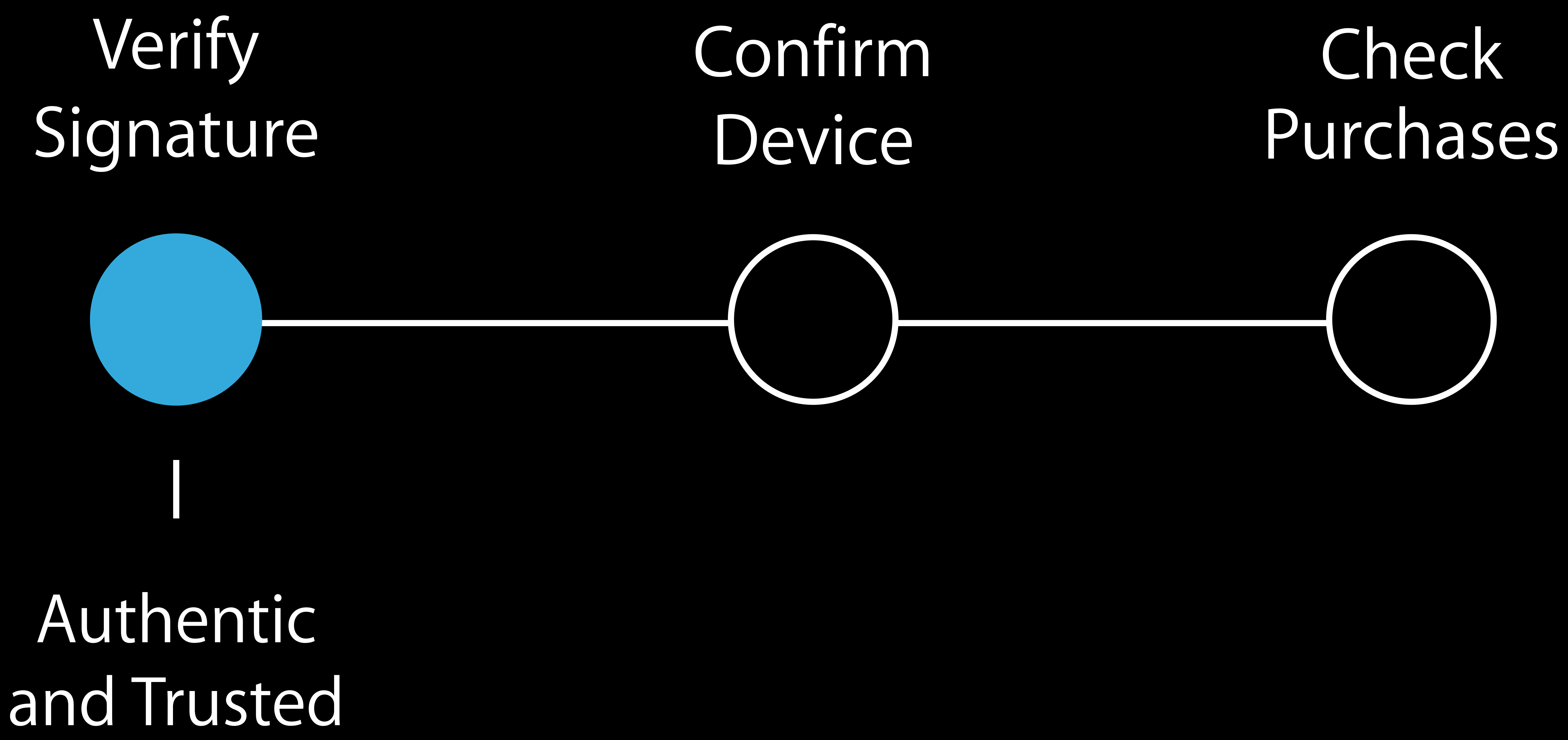
Examples and Samples

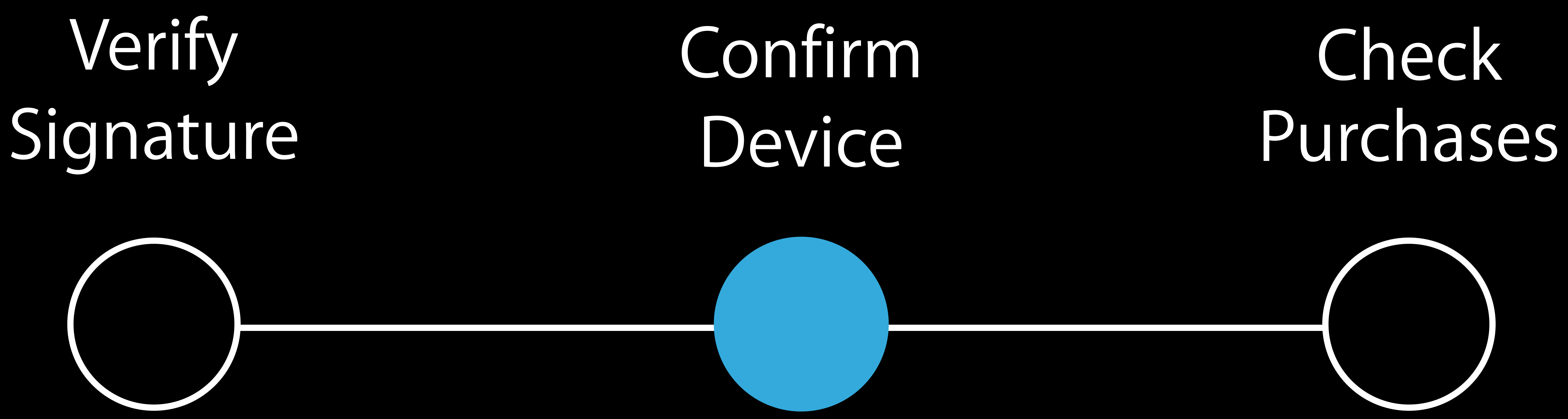
Convenience comes at a price

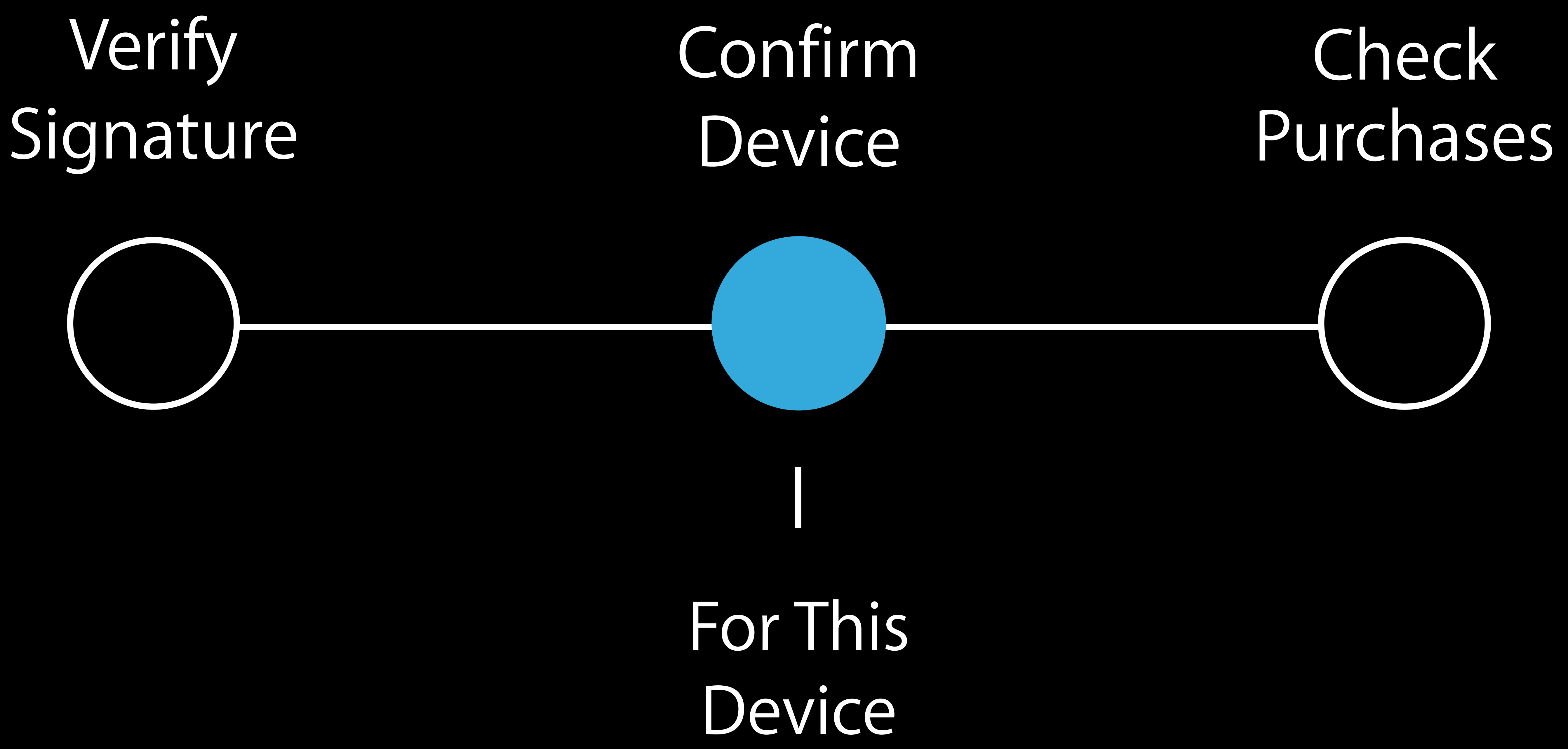
- Re-using code brings with it bugs and vulnerabilities
- Single exploit affects many

It’s your revenue stream

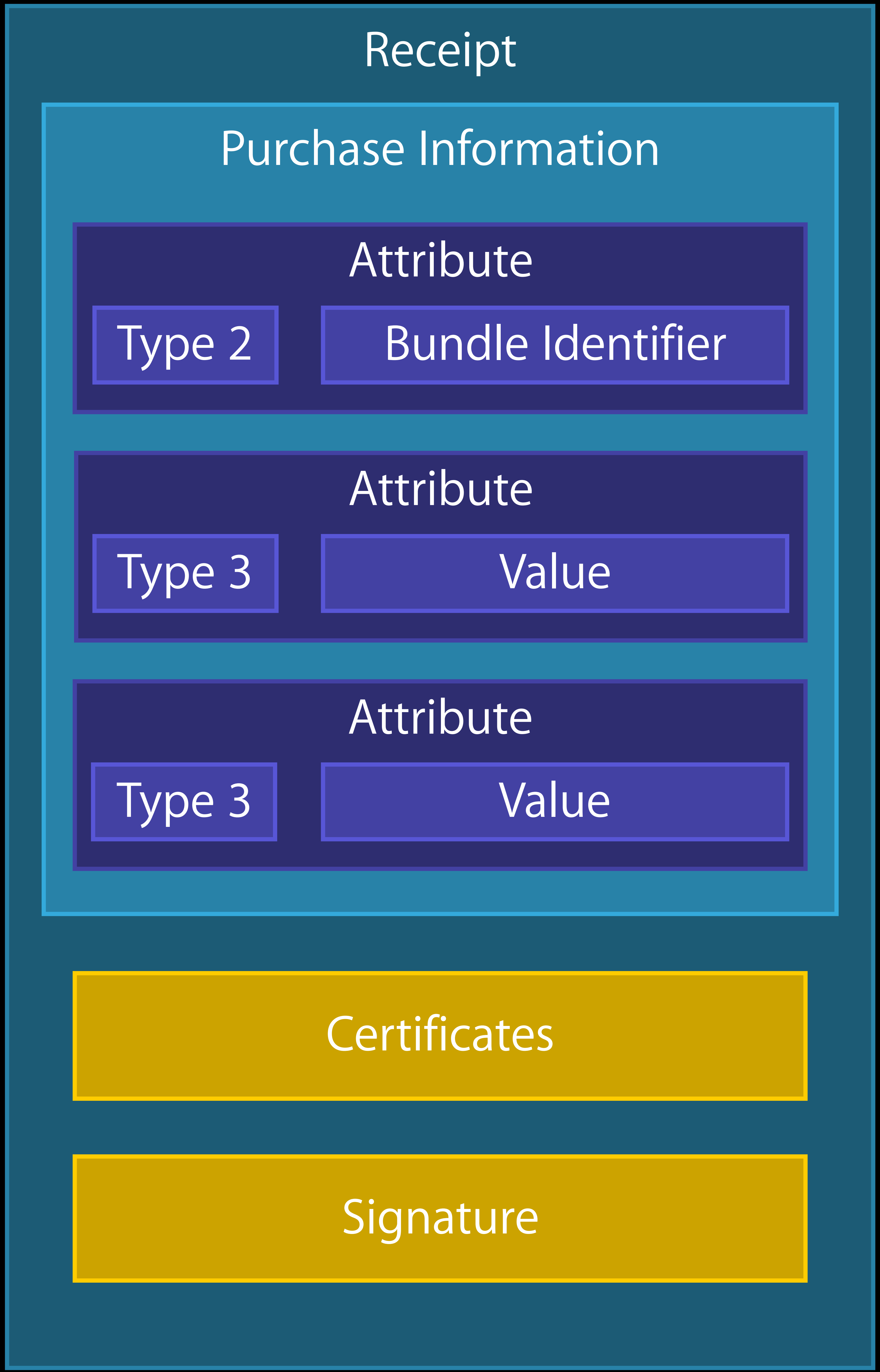
- Make decisions that suit your product
- Know and own the risks


Verify Signature
Confirm Device
Check Purchases
Authentic and Trusted


Verify Signature
Confirm Device
Check Purchases

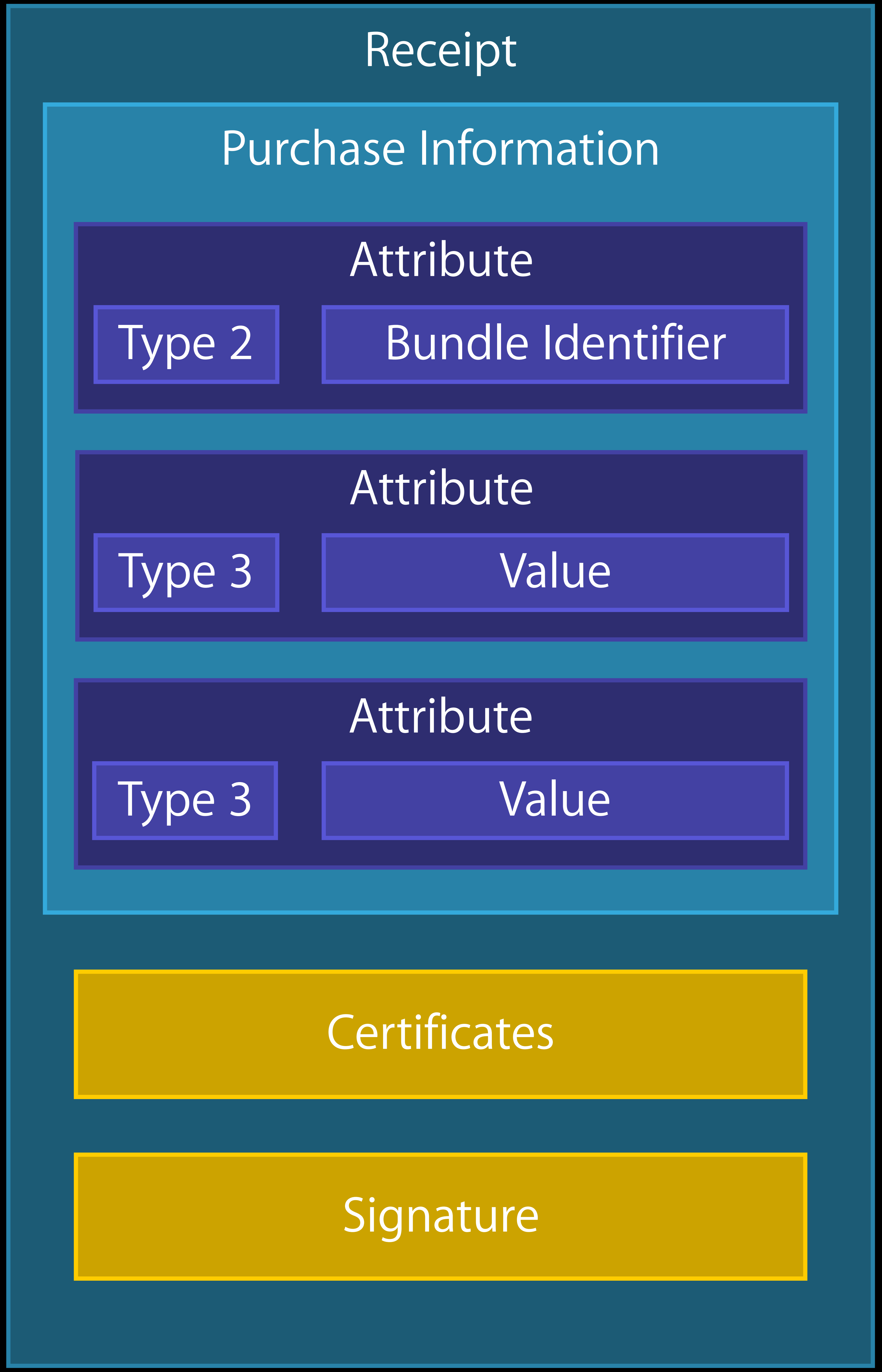

Verify Signature
Confirm Device
Check Purchases
For This Device

Receipt Payload



Receipt Payload

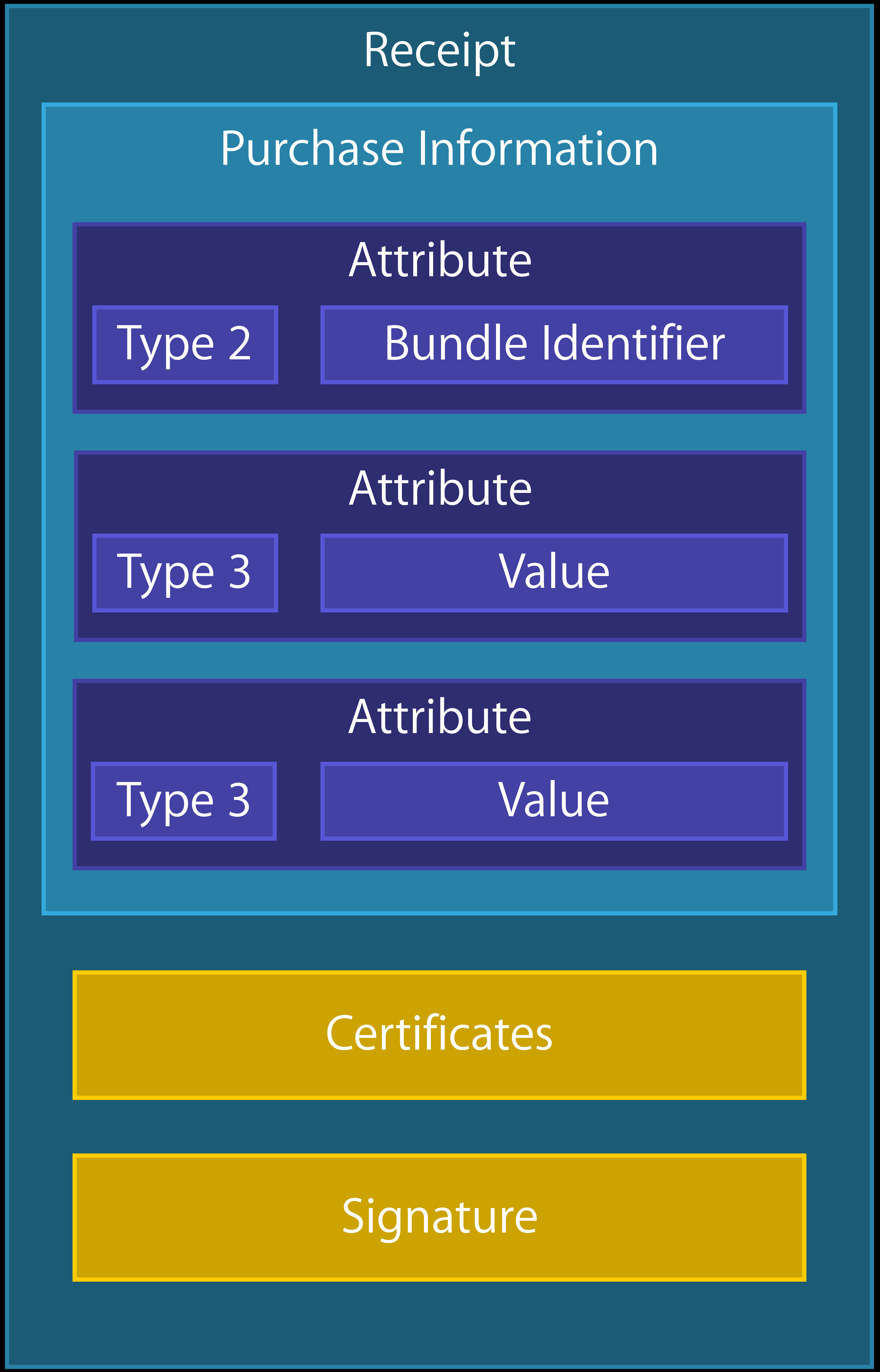
Series of attributes



Receipt Payload

Series of attributes

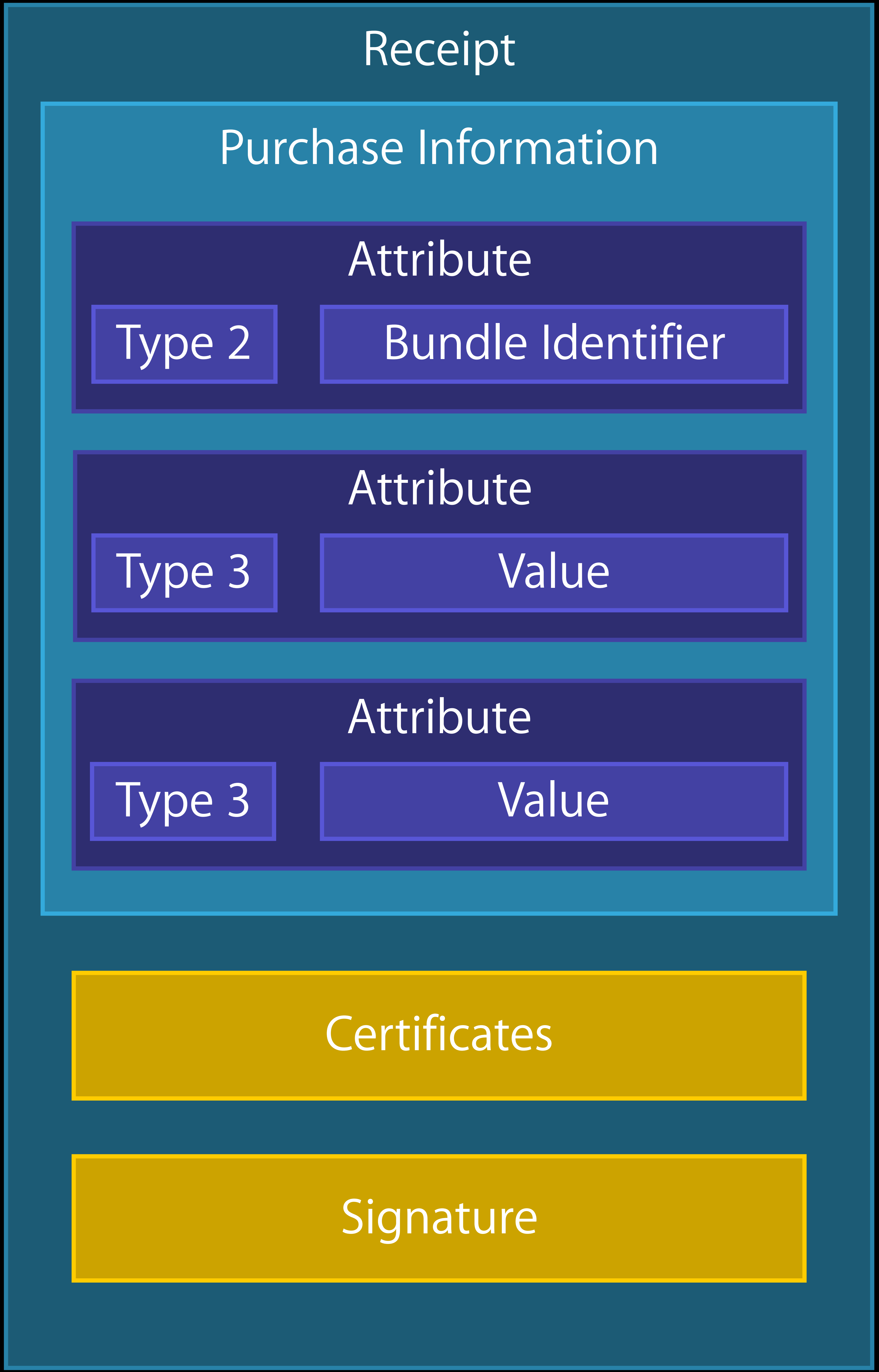
- Type



Receipt Payload

Series of attributes

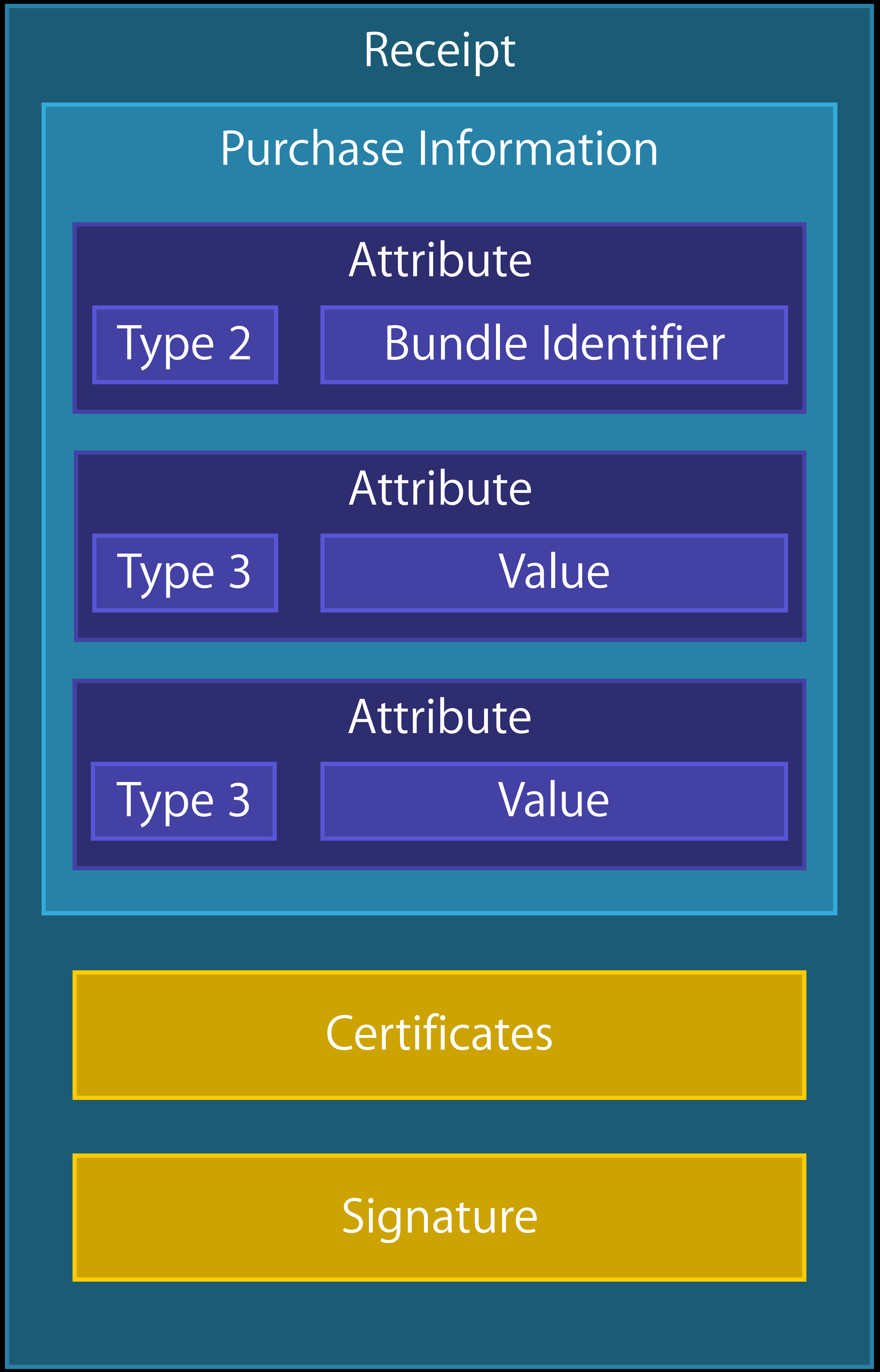
- Type
- Value



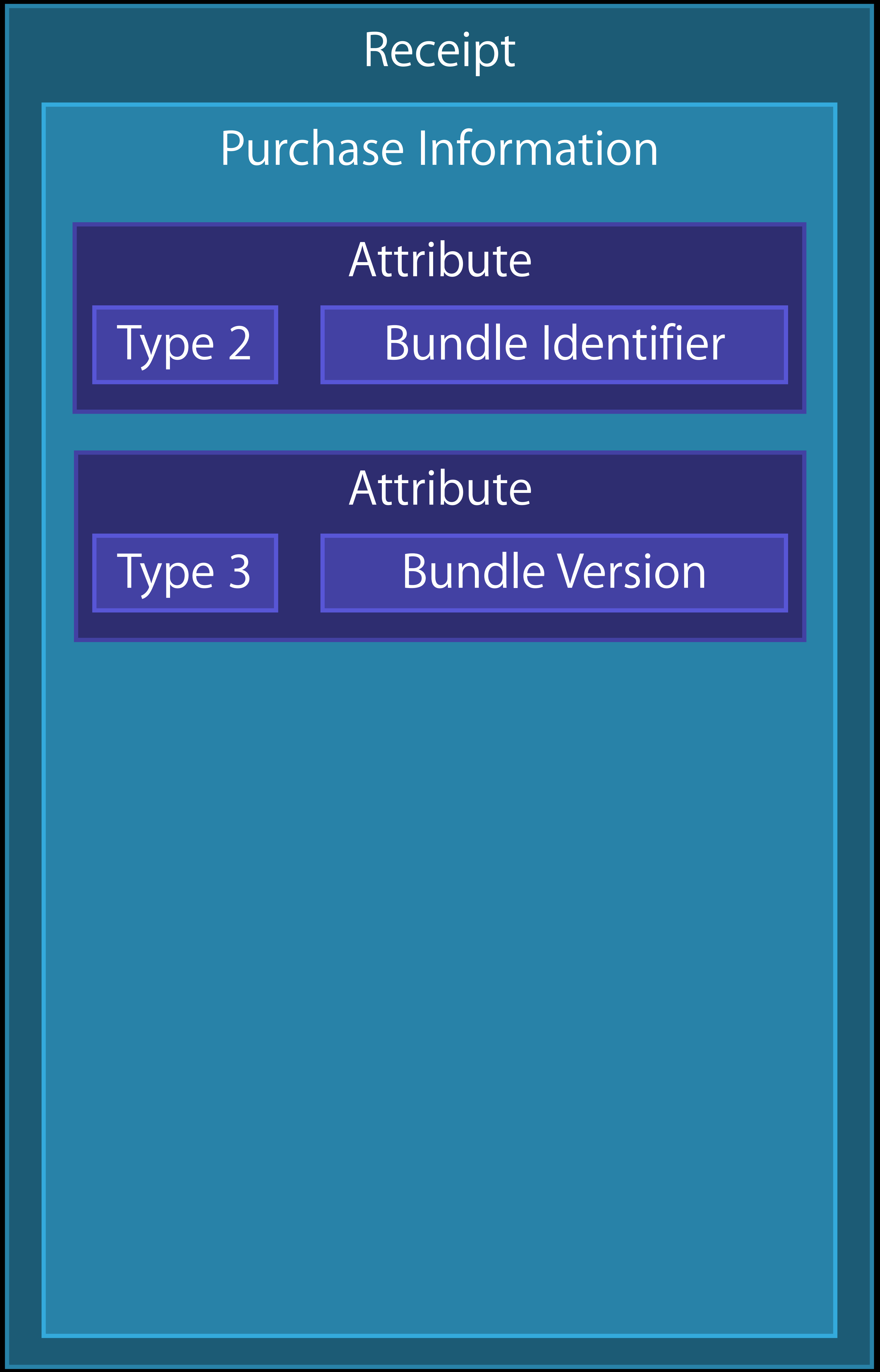
Receipt Payload

Series of attributes

- Type
- Value
- (Version)

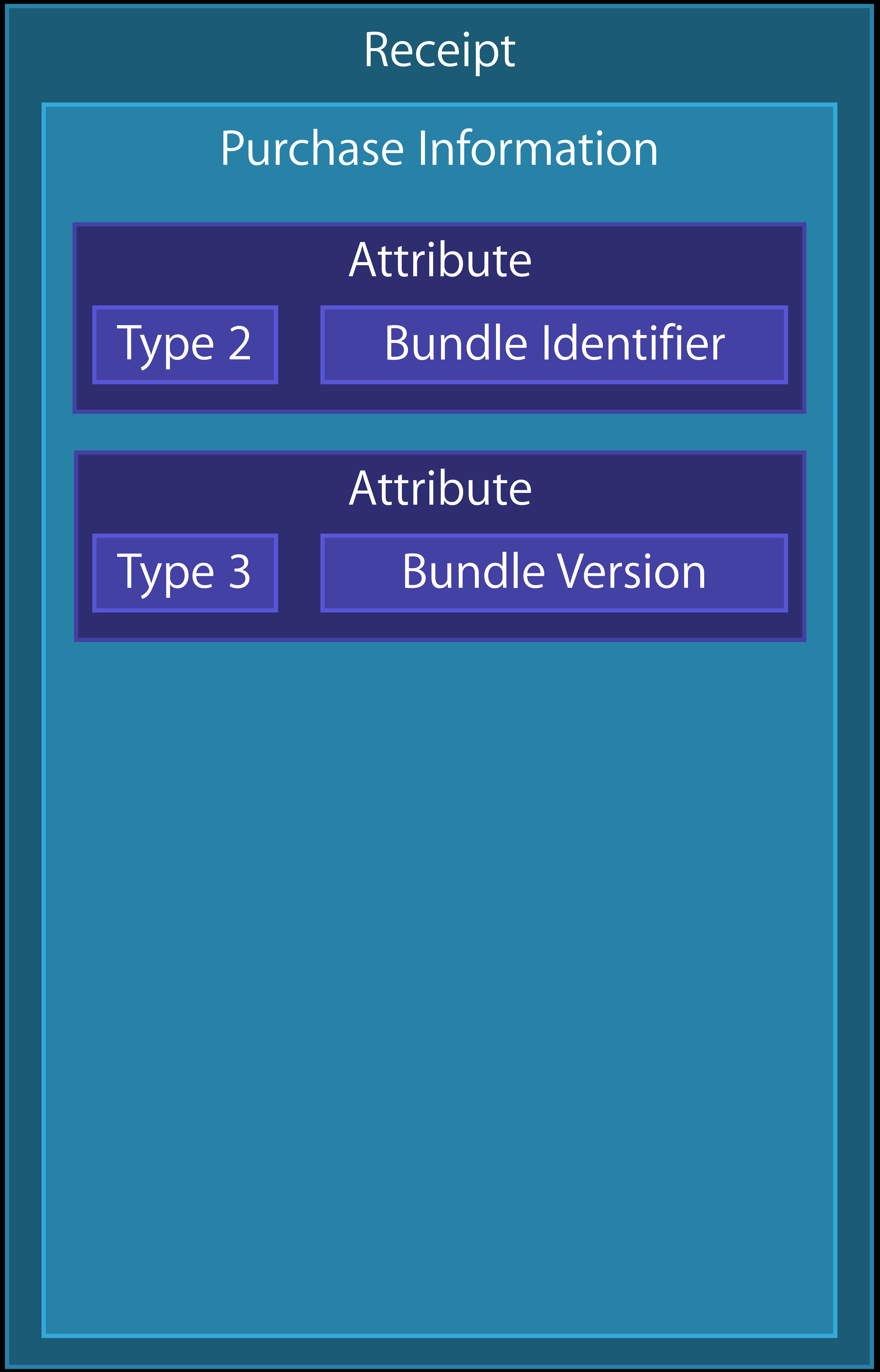


Verify Application



Verify Application

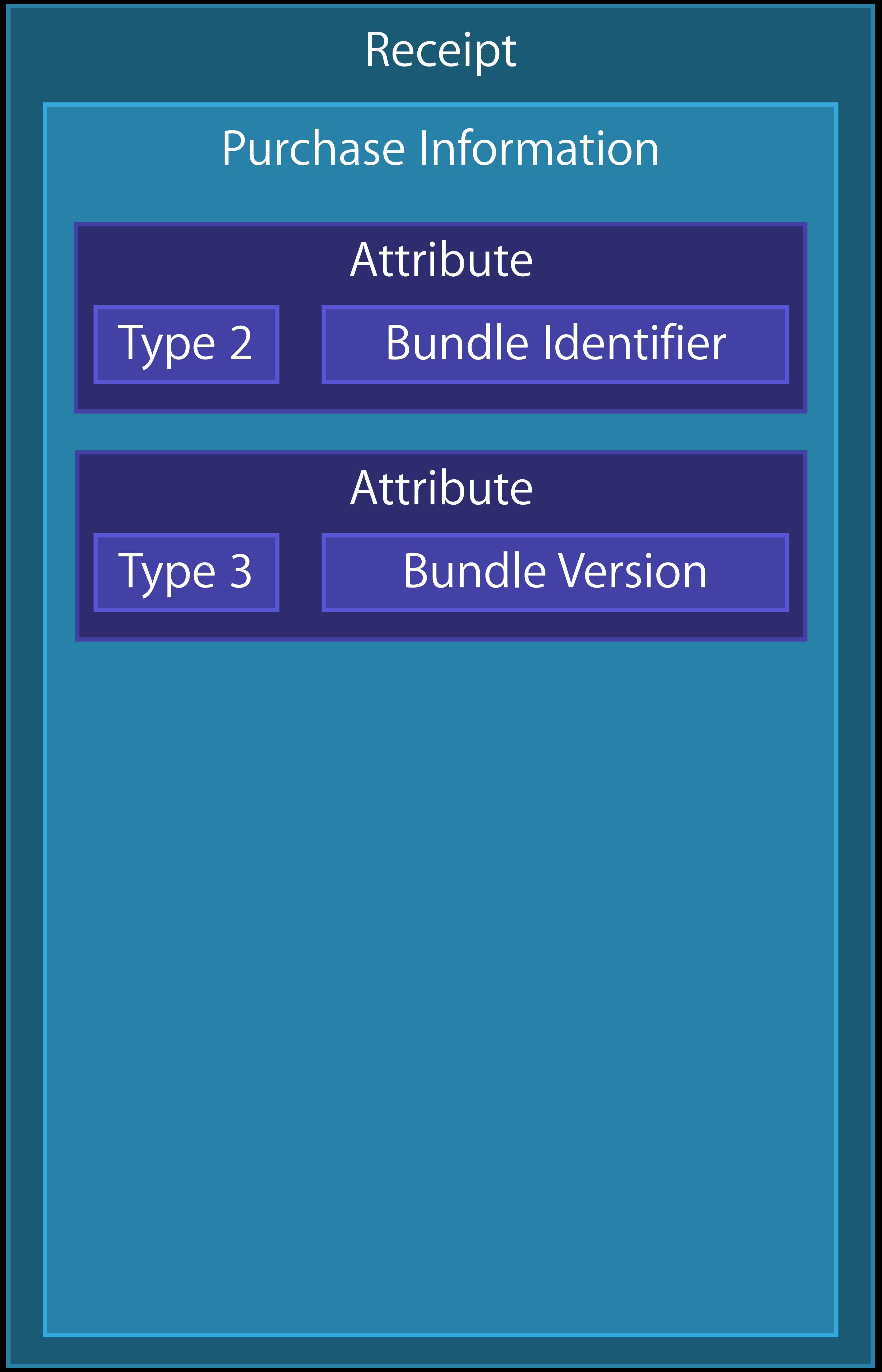
Check the Bundle Identifier



Verify Application

Check the Bundle Identifier

Check the Bundle Version

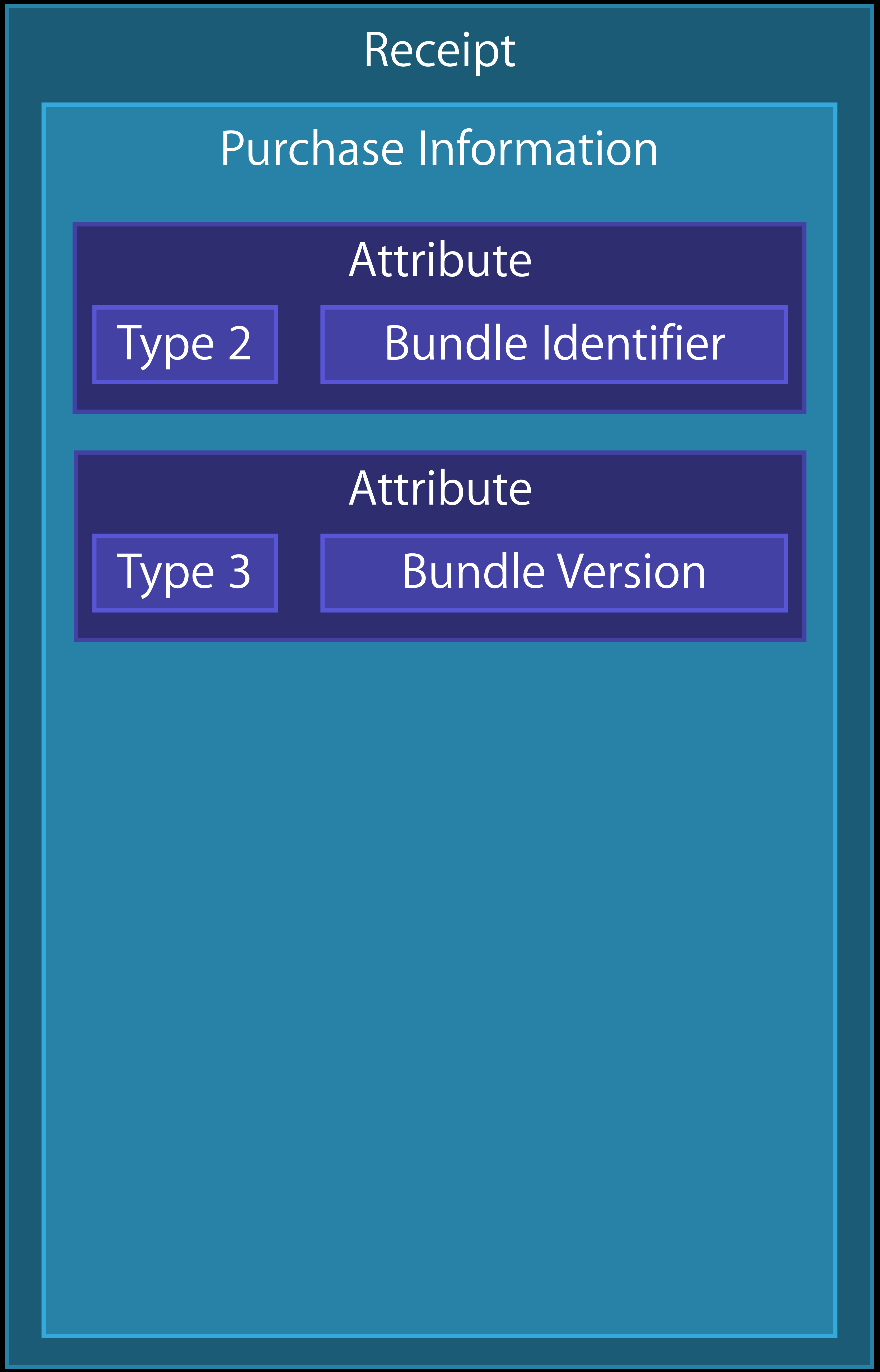


Verify Application

Check the Bundle Identifier

Check the Bundle Version

Use hardcoded values



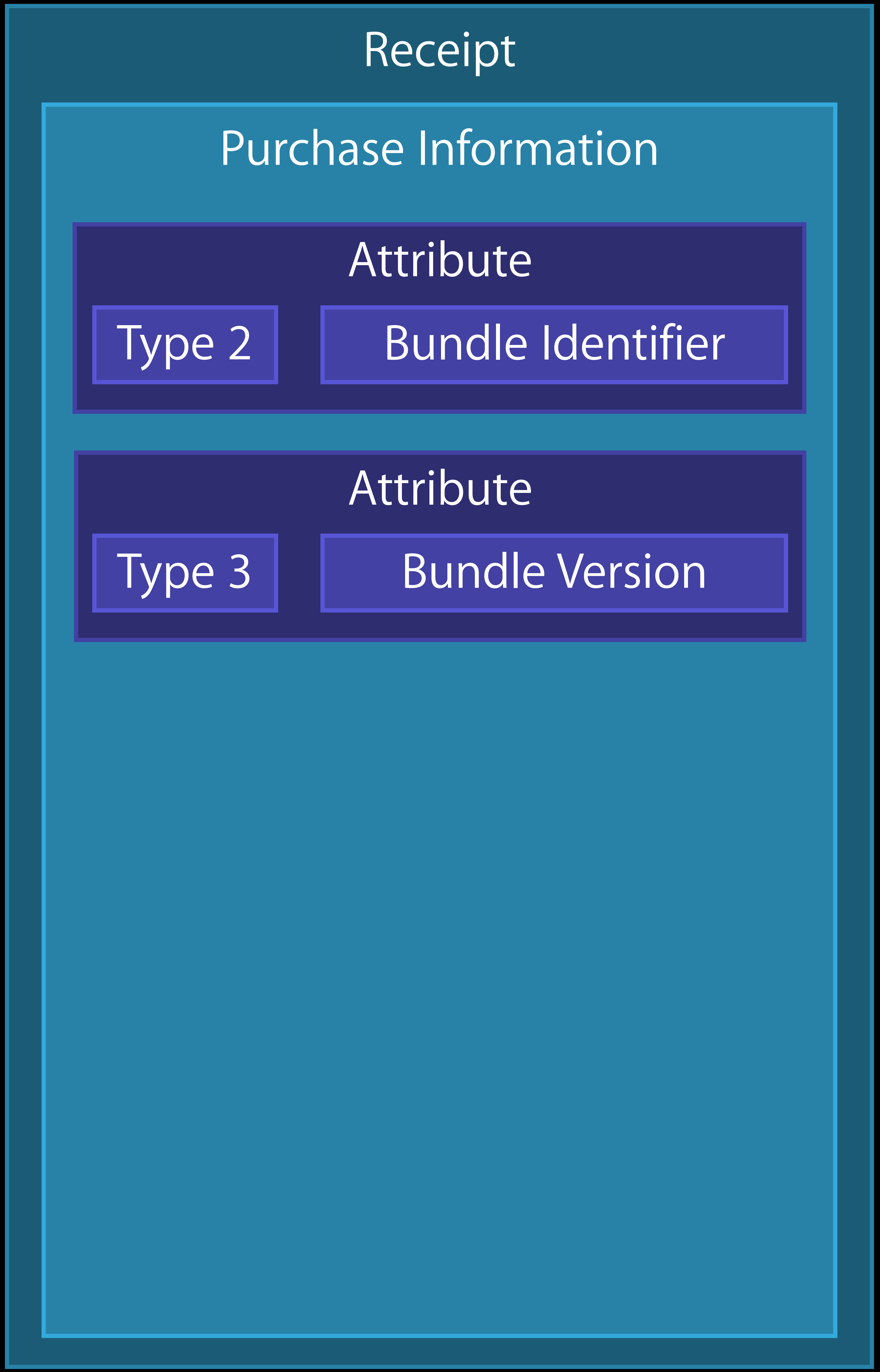
Verify Application

Check the Bundle Identifier

Check the Bundle Version

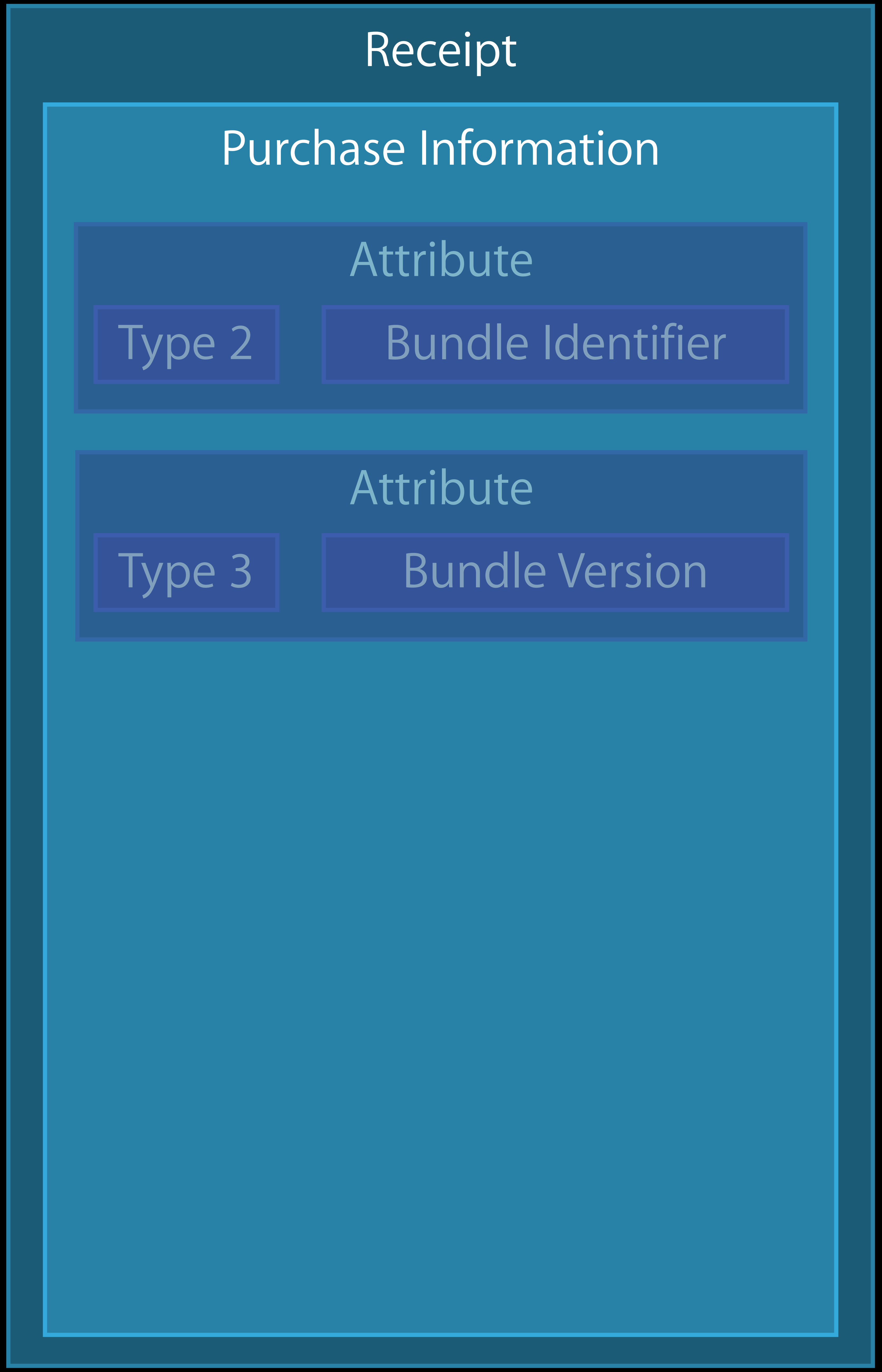
Use hardcoded values

- Not Info.plist values



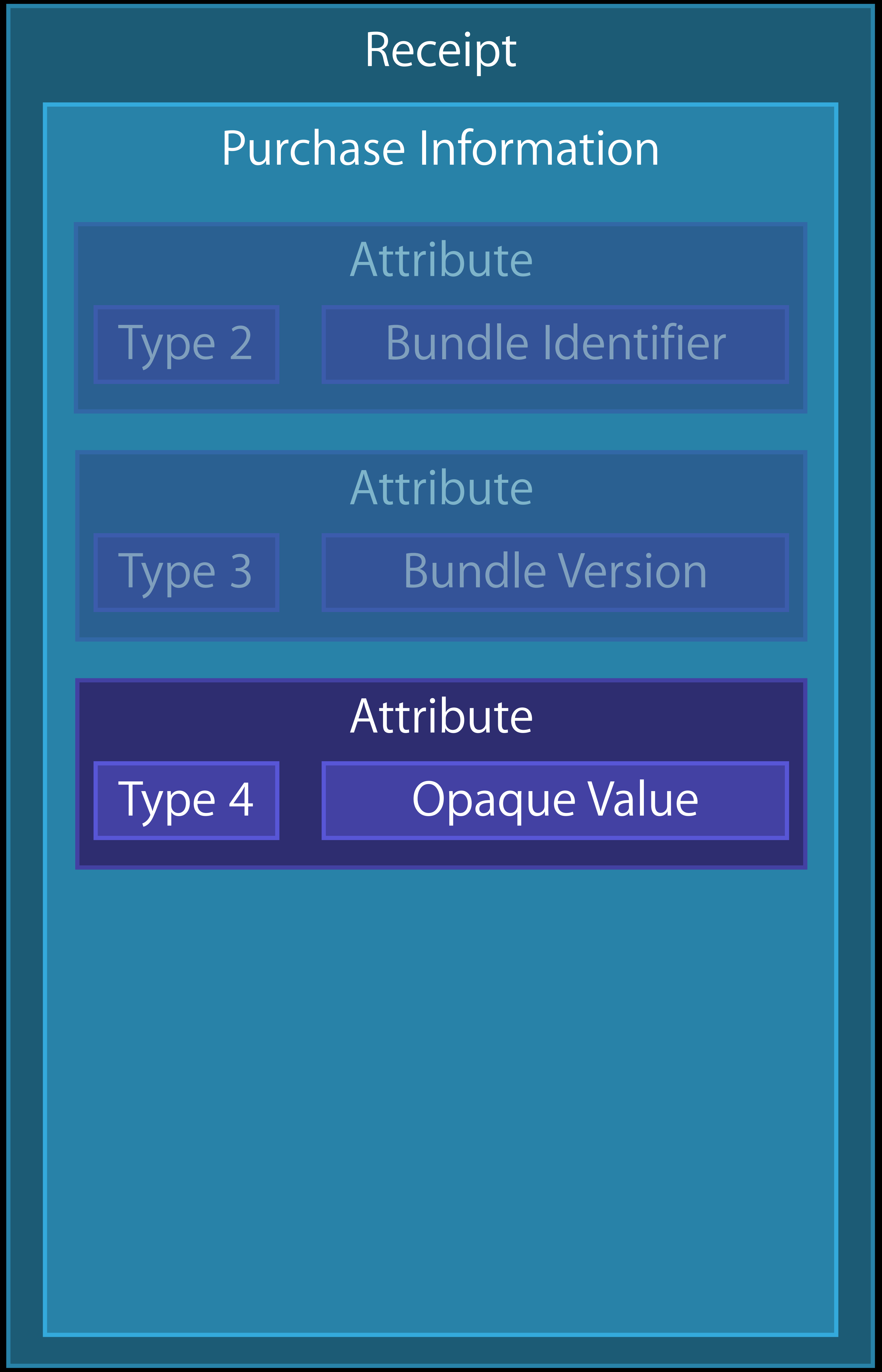
Verify Device

ASN.1 format



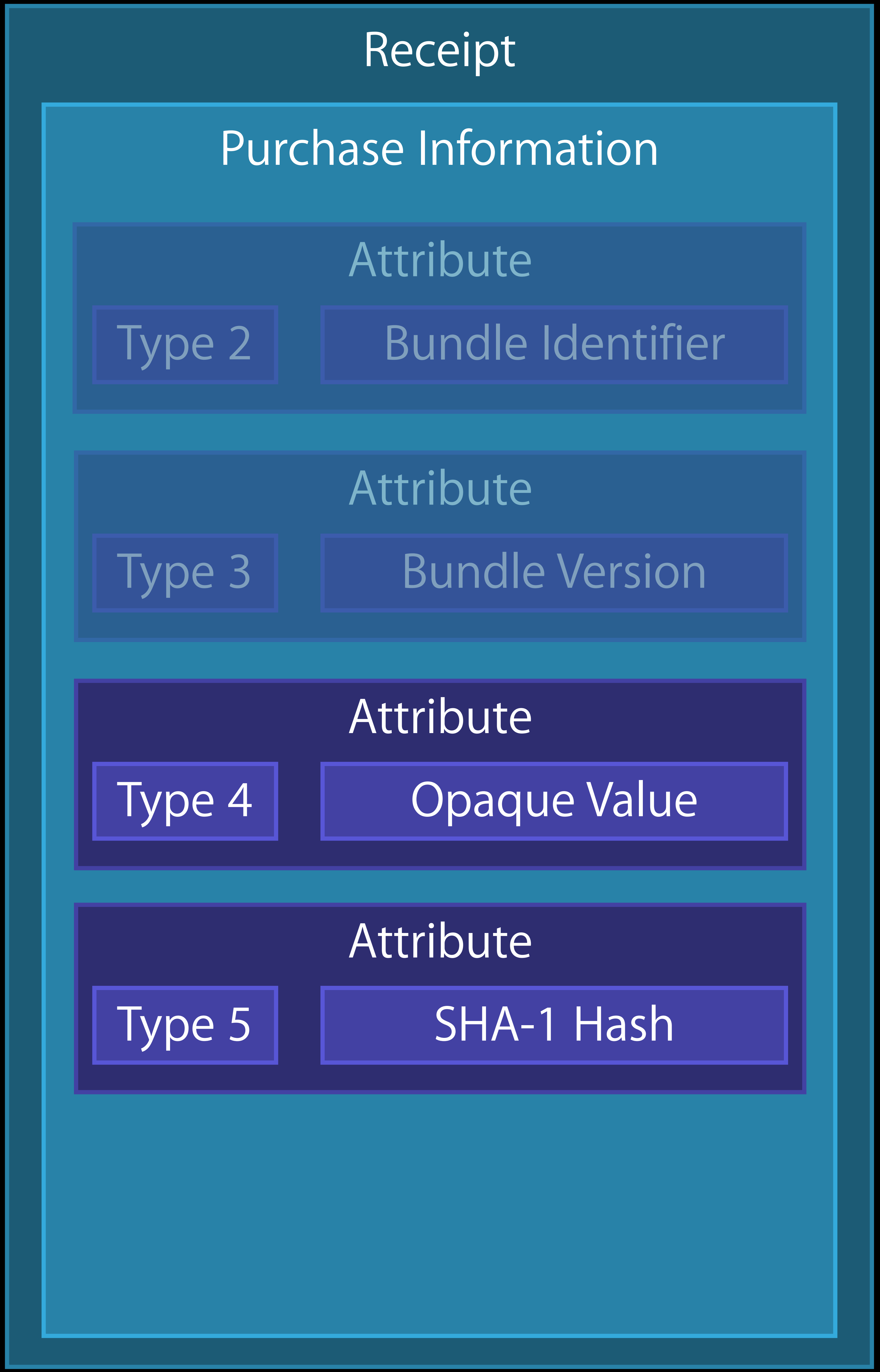
Verify Device

ASN.1 format



Verify Device

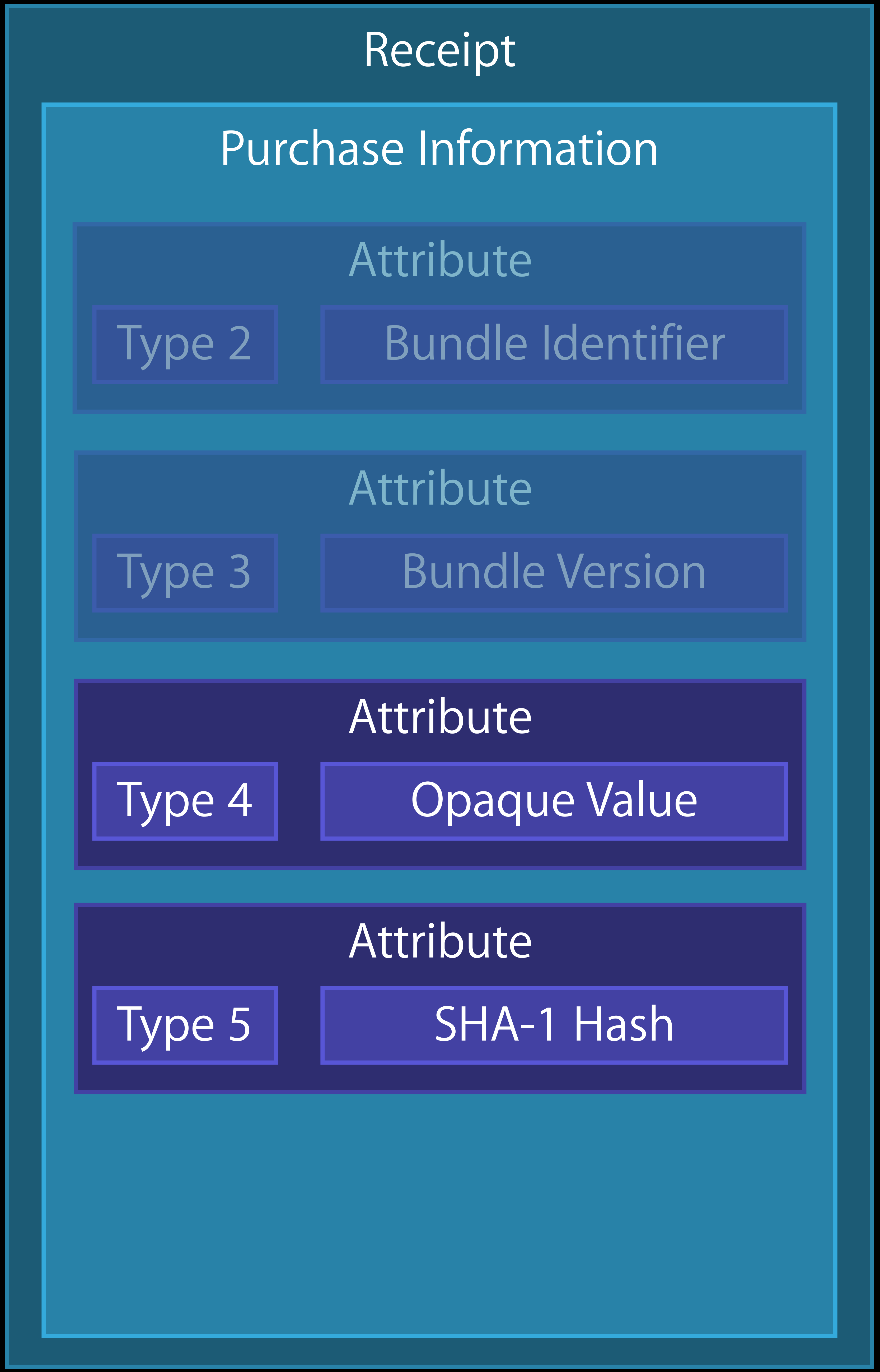
ASN.1 format



Verify Device

ASN.1 format

Attribute 5 is a SHA-1 hash of 3 key values

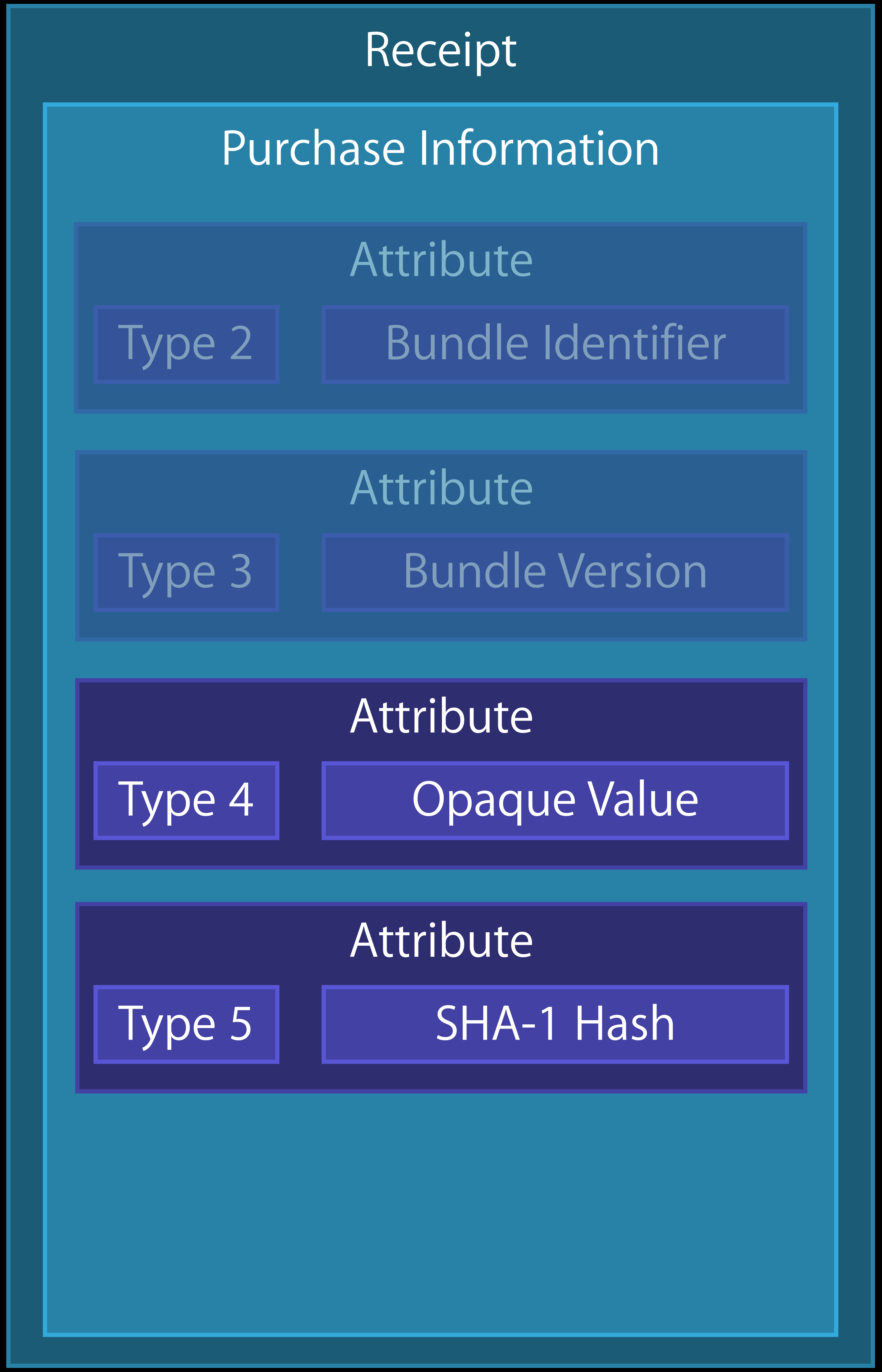


Verify Device

ASN.1 format

Attribute 5 is a SHA-1 hash of 3 key values

- Bundle ID

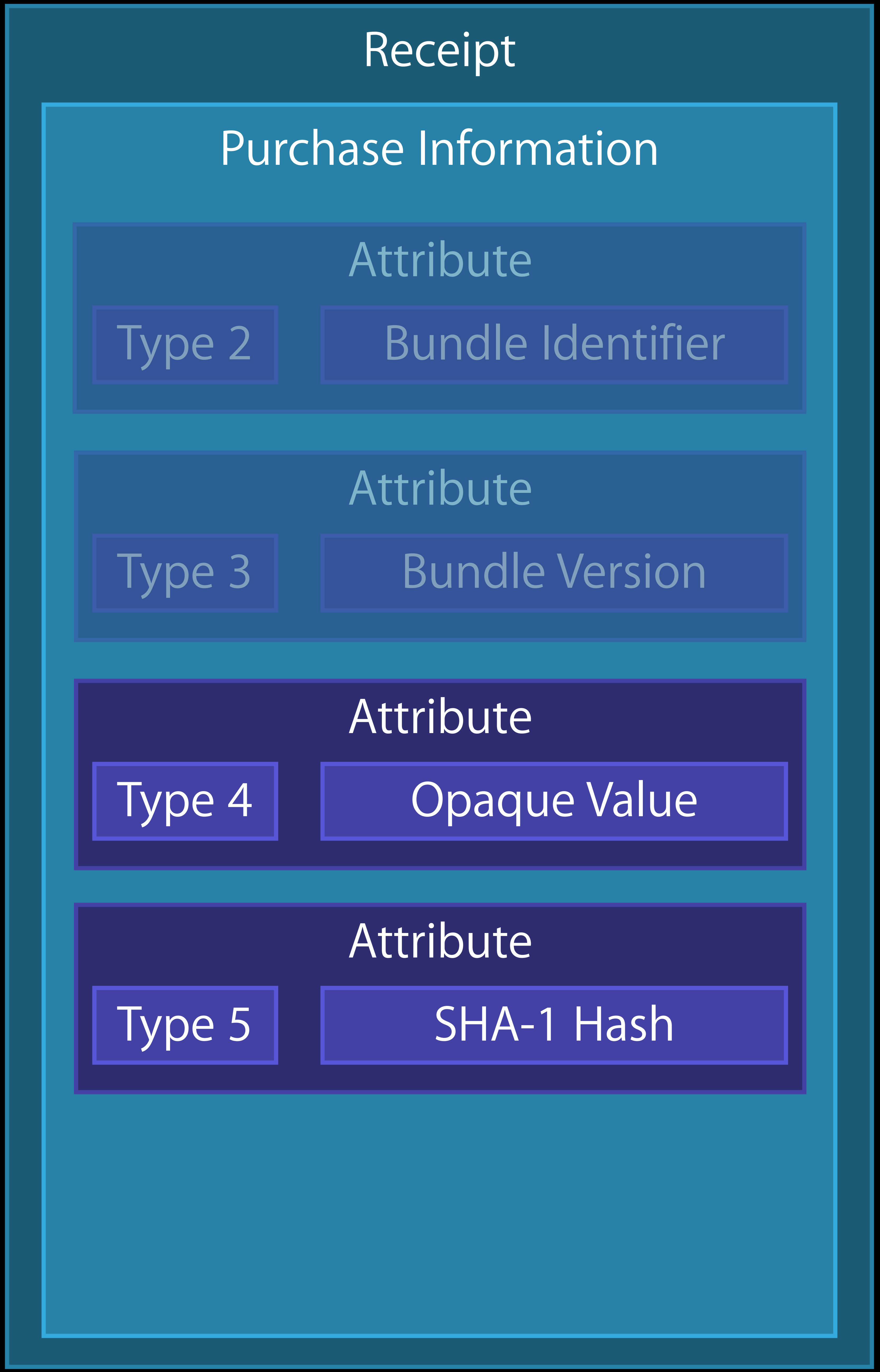


Verify Device

ASN.1 format

Attribute 5 is a SHA-1 hash of 3 key values

- Bundle ID
- Device Identifier

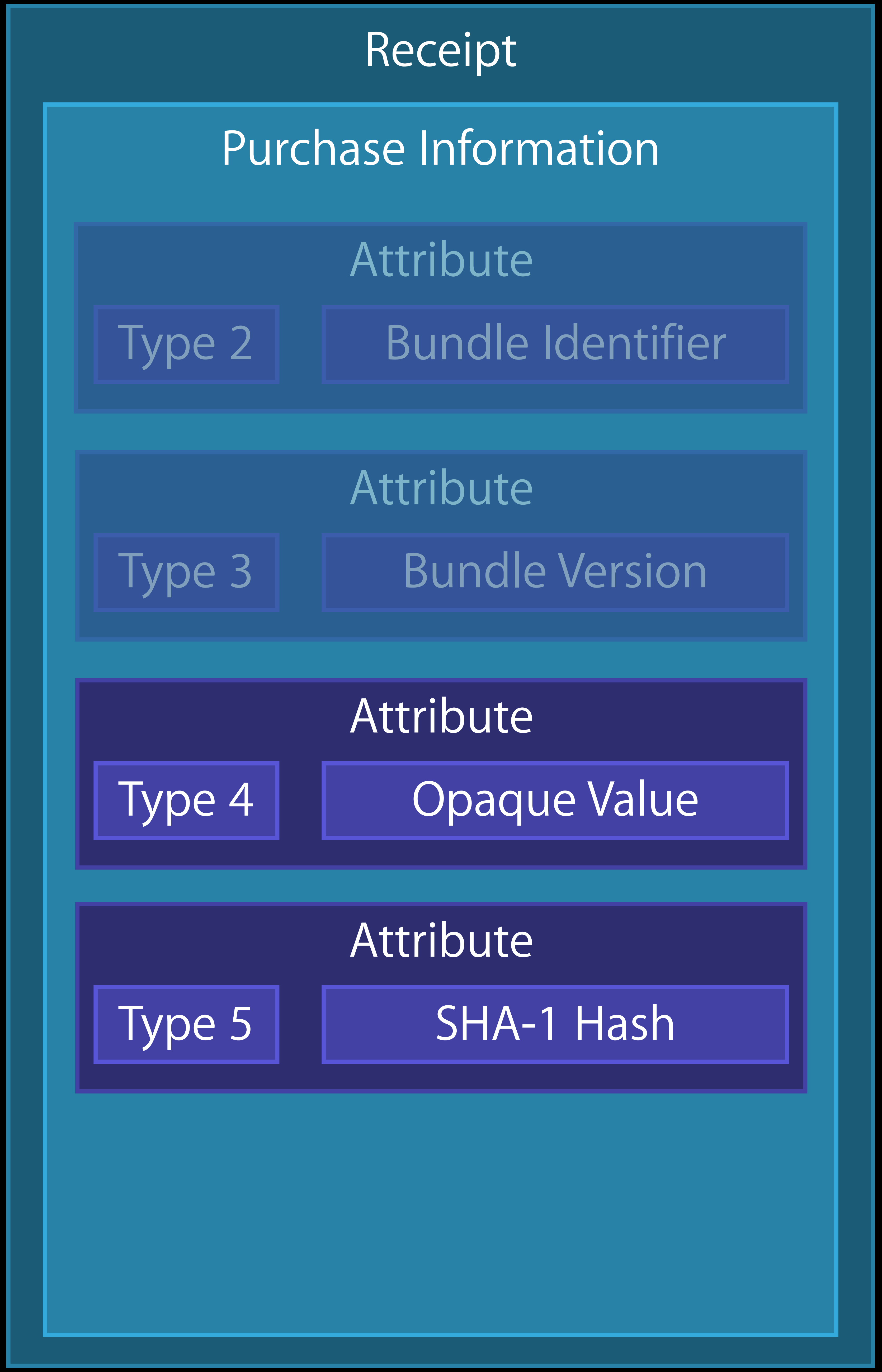


Verify Device

ASN.1 format

Attribute 5 is a SHA-1 hash of 3 key values

- Bundle ID
- Device Identifier
- Opaque Value



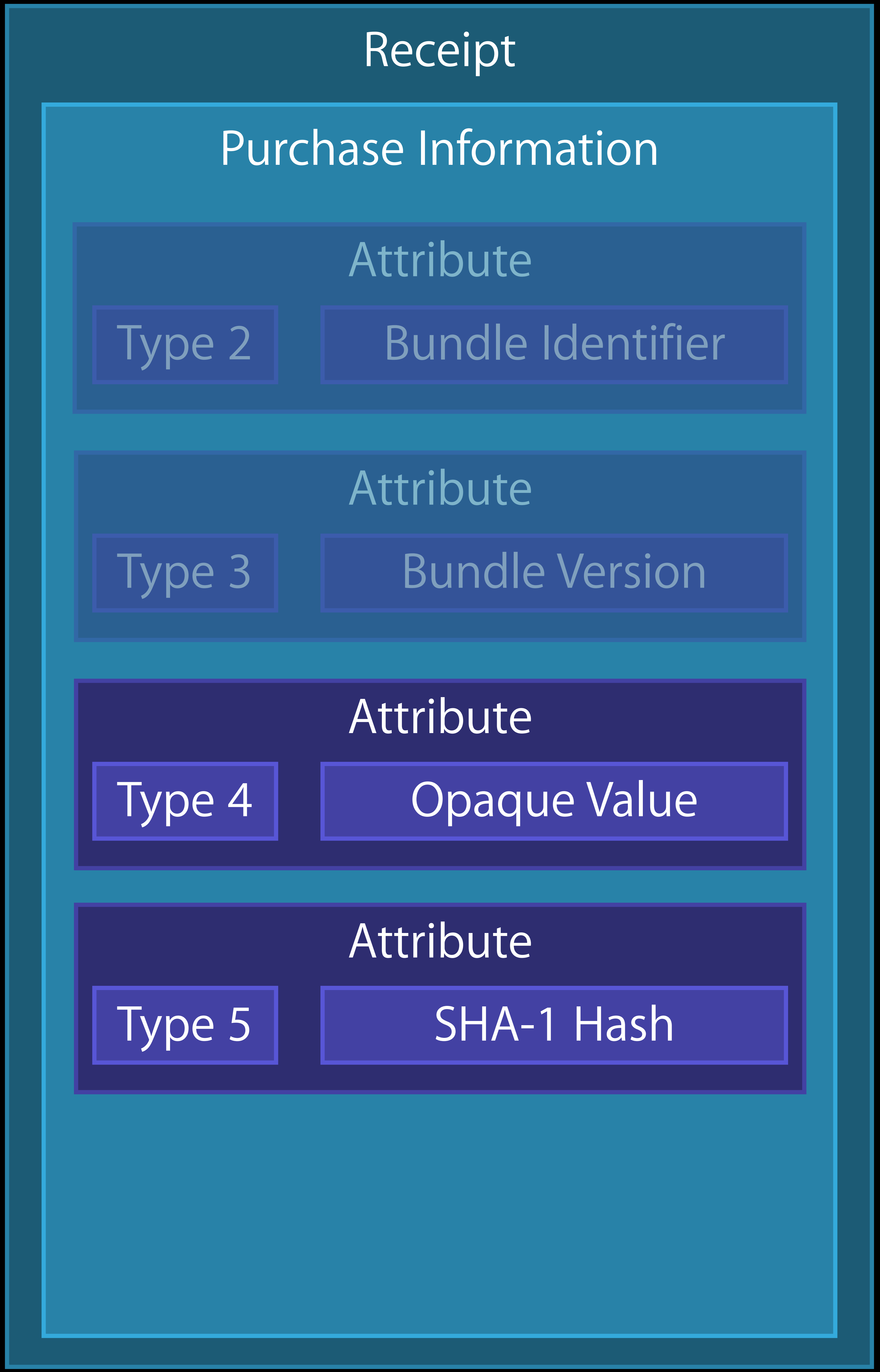
Verify Device

ASN.1 format

Attribute 5 is a SHA-1 hash of 3 key values

- Bundle ID
- Device Identifier
- Opaque Value

The App Store knows these at time of purchase



Verify Device

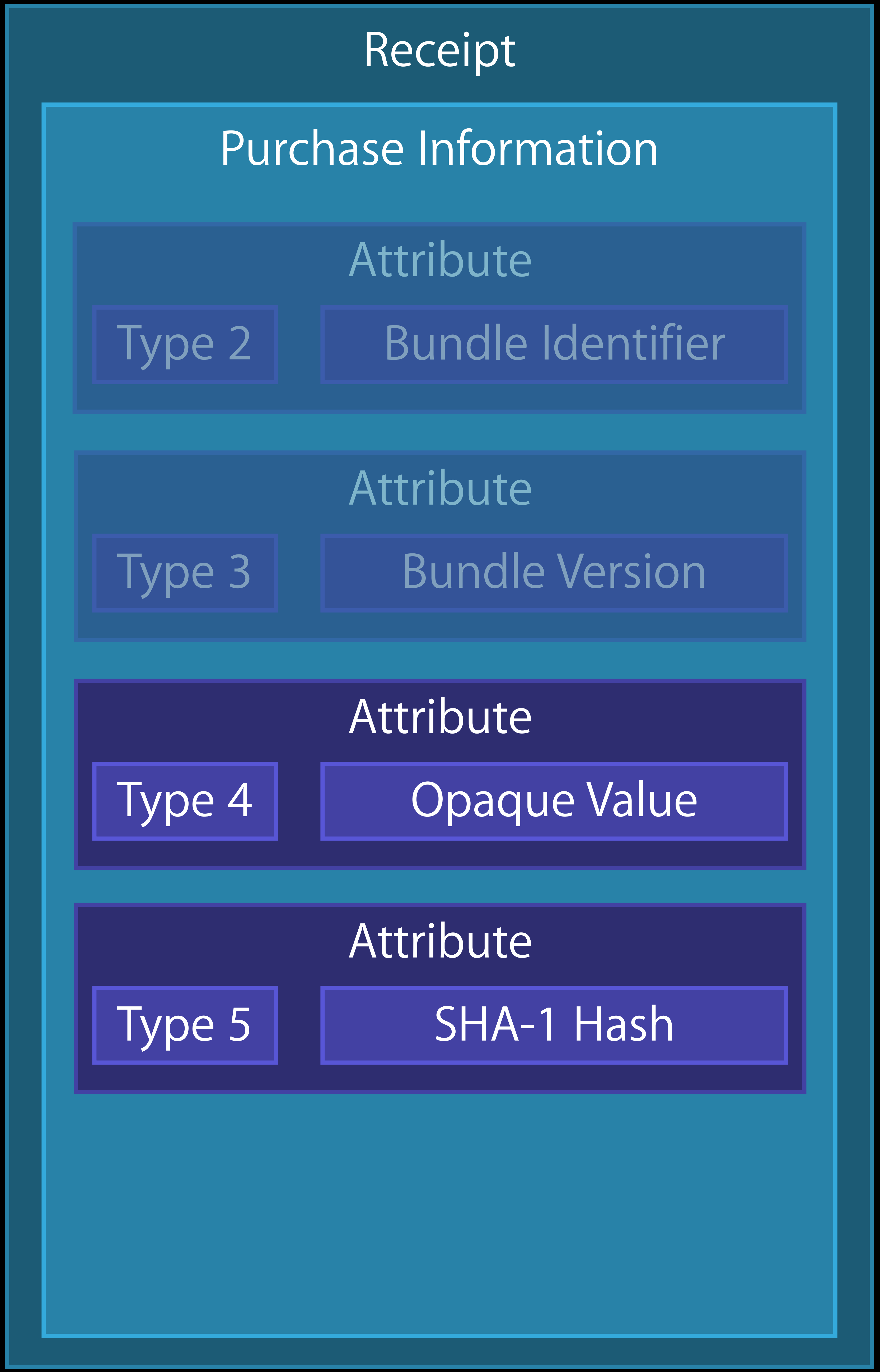
ASN.1 format

Attribute 5 is a SHA-1 hash of 3 key values

- Bundle ID
- Device Identifier
- Opaque Value

The App Store knows these at time of purchase

Your app knows them at time of verification



Verify Device

ASN.1 format

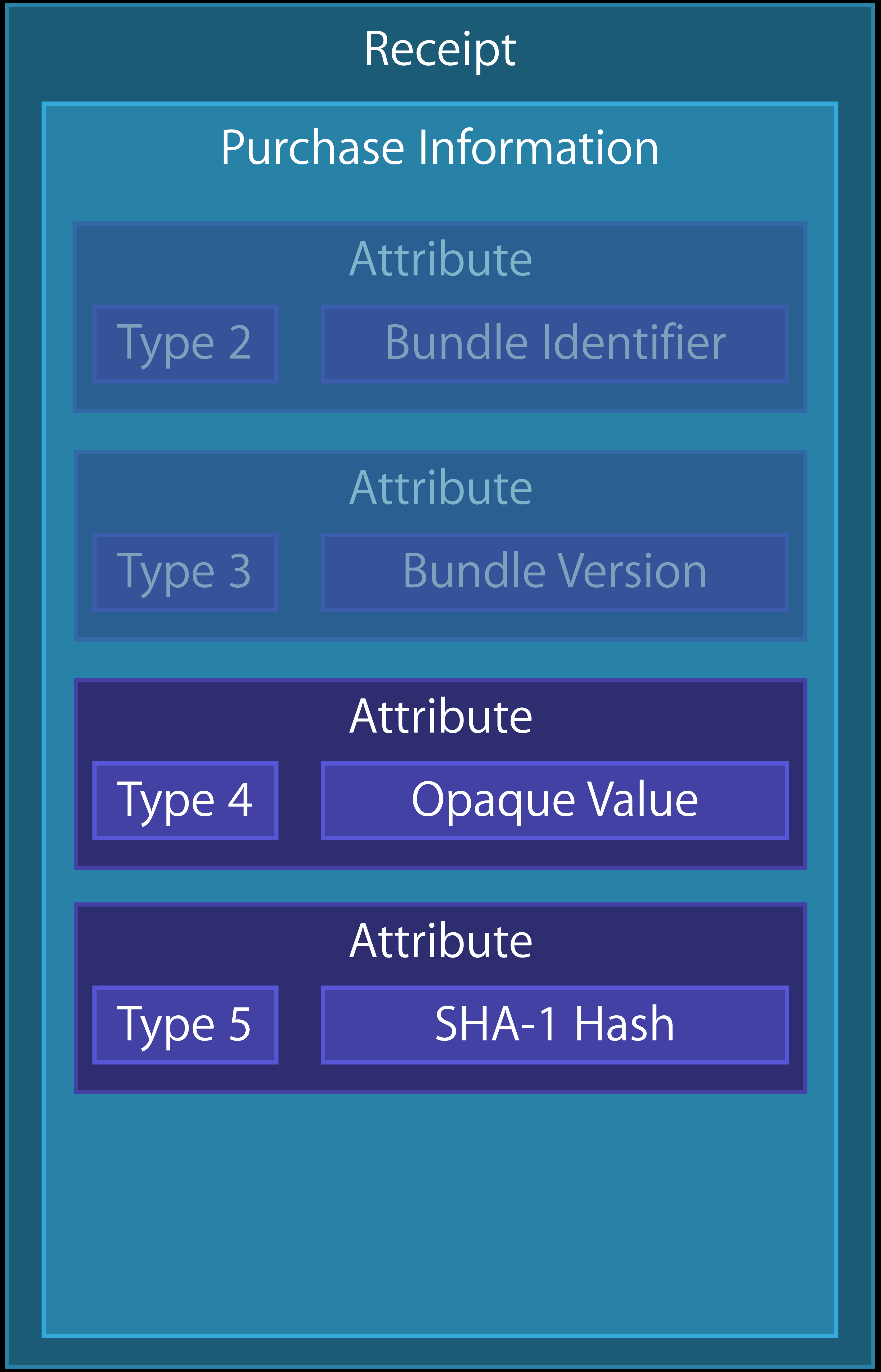
Attribute 5 is a SHA-1 hash of 3 key values

- Bundle ID
- Device Identifier
- Opaque Value

The App Store knows these at time of purchase

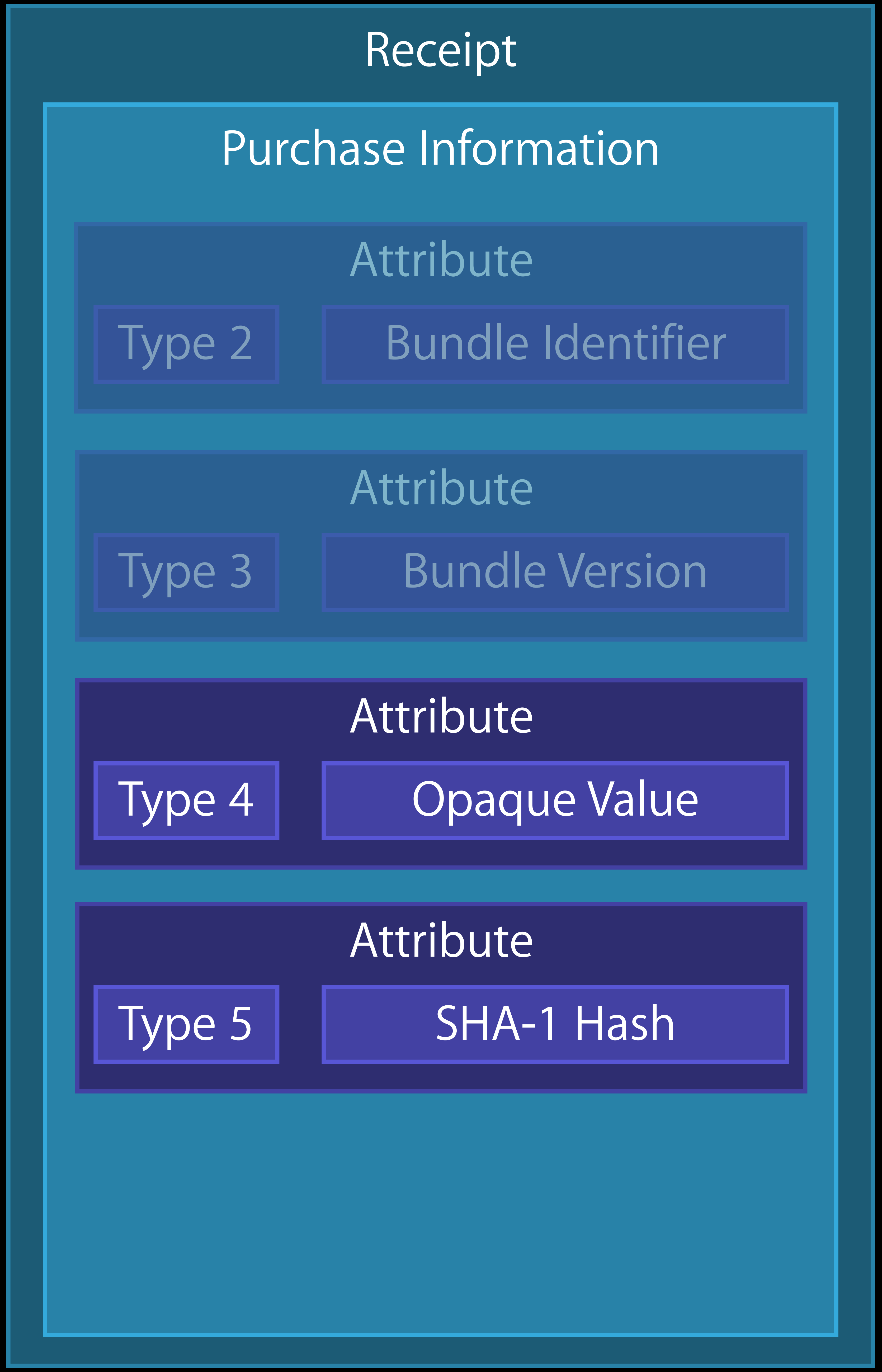
Your app knows them at time of verification

Unique to your app on this device



Data Encoding

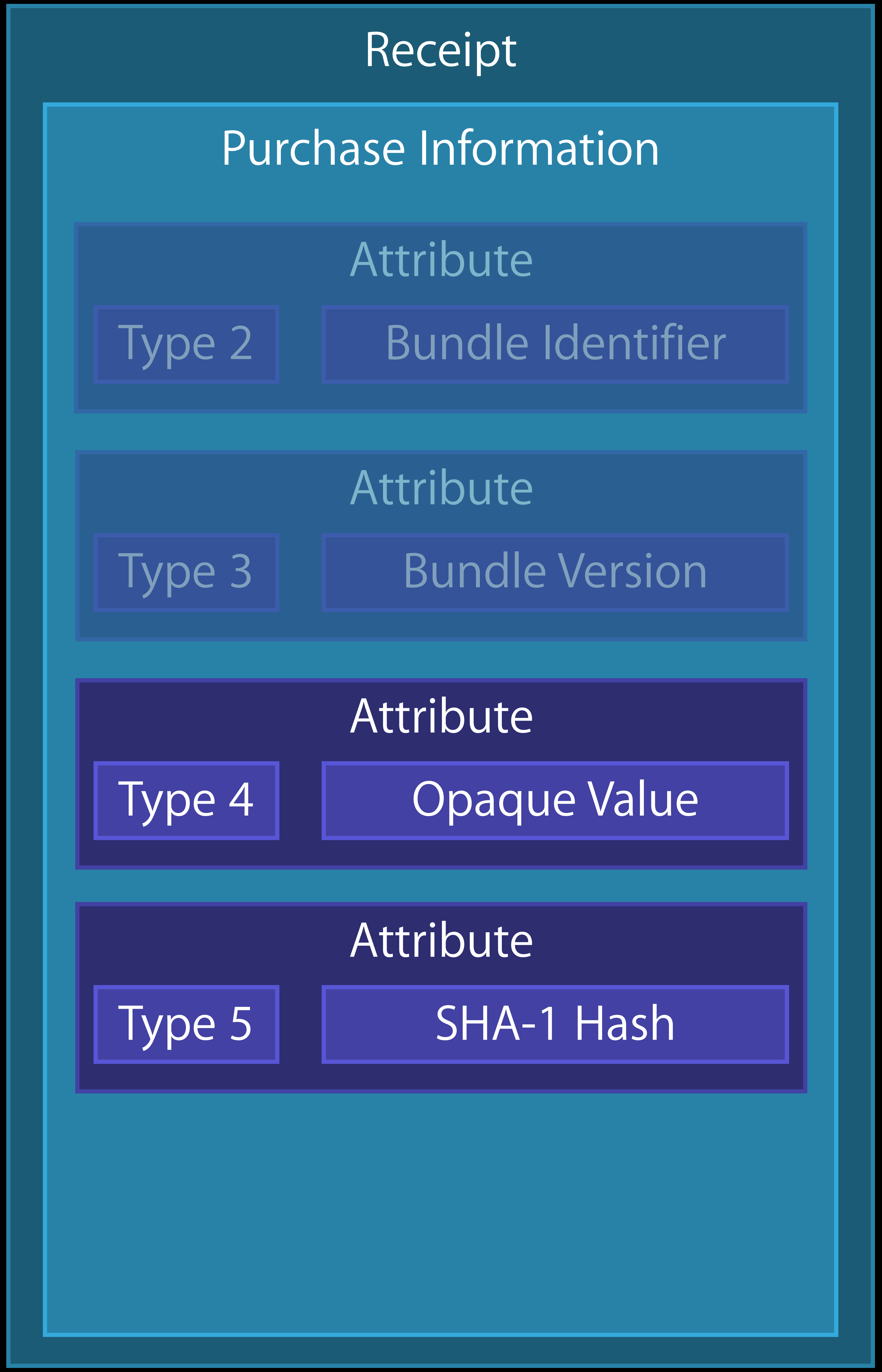
ASN.1 Format



Data Encoding

ASN.1 Format

Receipt payload format definition

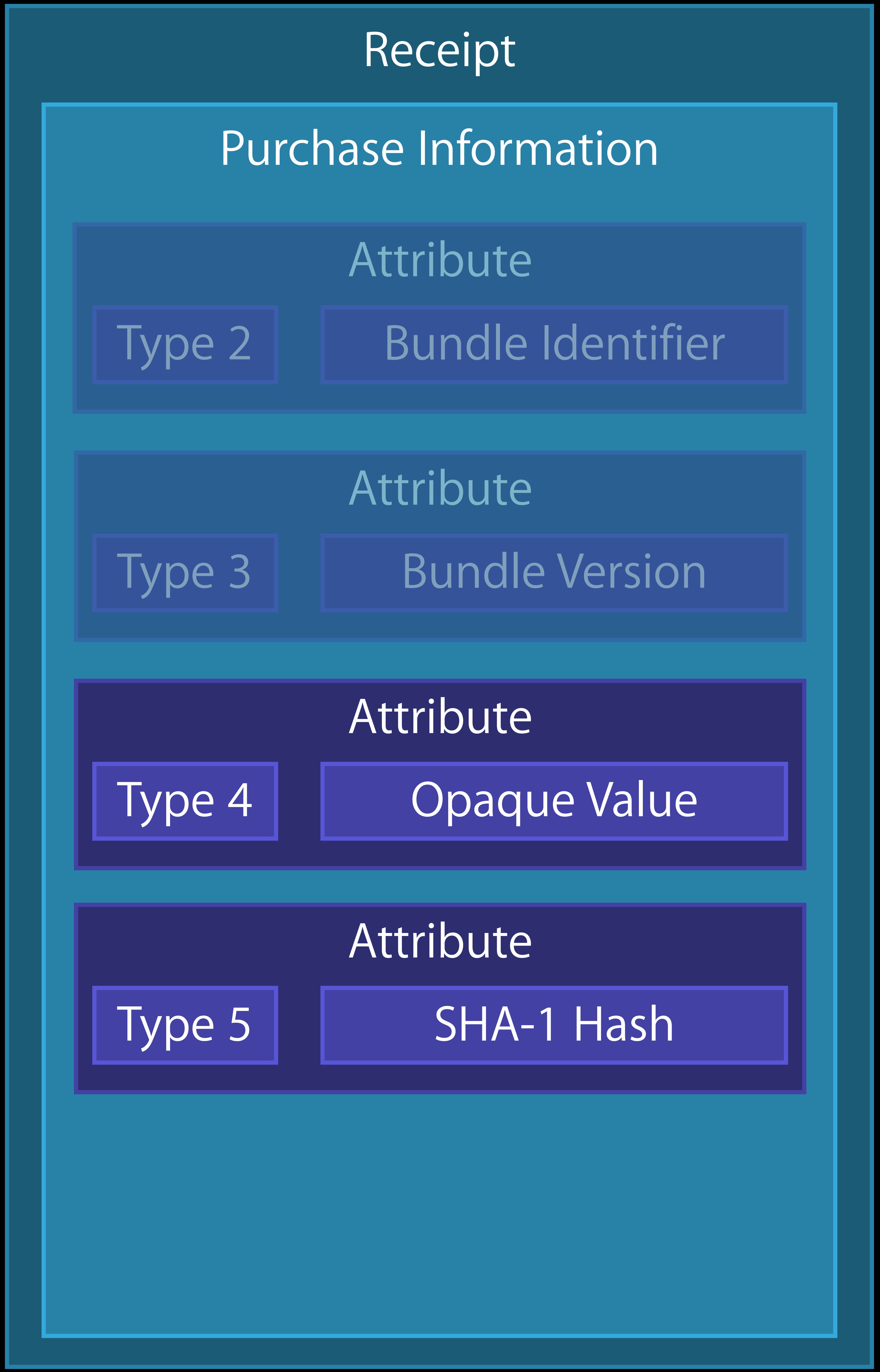


Data Encoding

ASN.1 Format

Receipt payload format definition

```
ReceiptModule DEFINITIONS ::=
BEGIN

ReceiptAttribute ::= SEQUENCE {
    type    INTEGER,
    version INTEGER,
    value   OCTET STRING
}

Payload ::= SET OF ReceiptAttribute

END
```

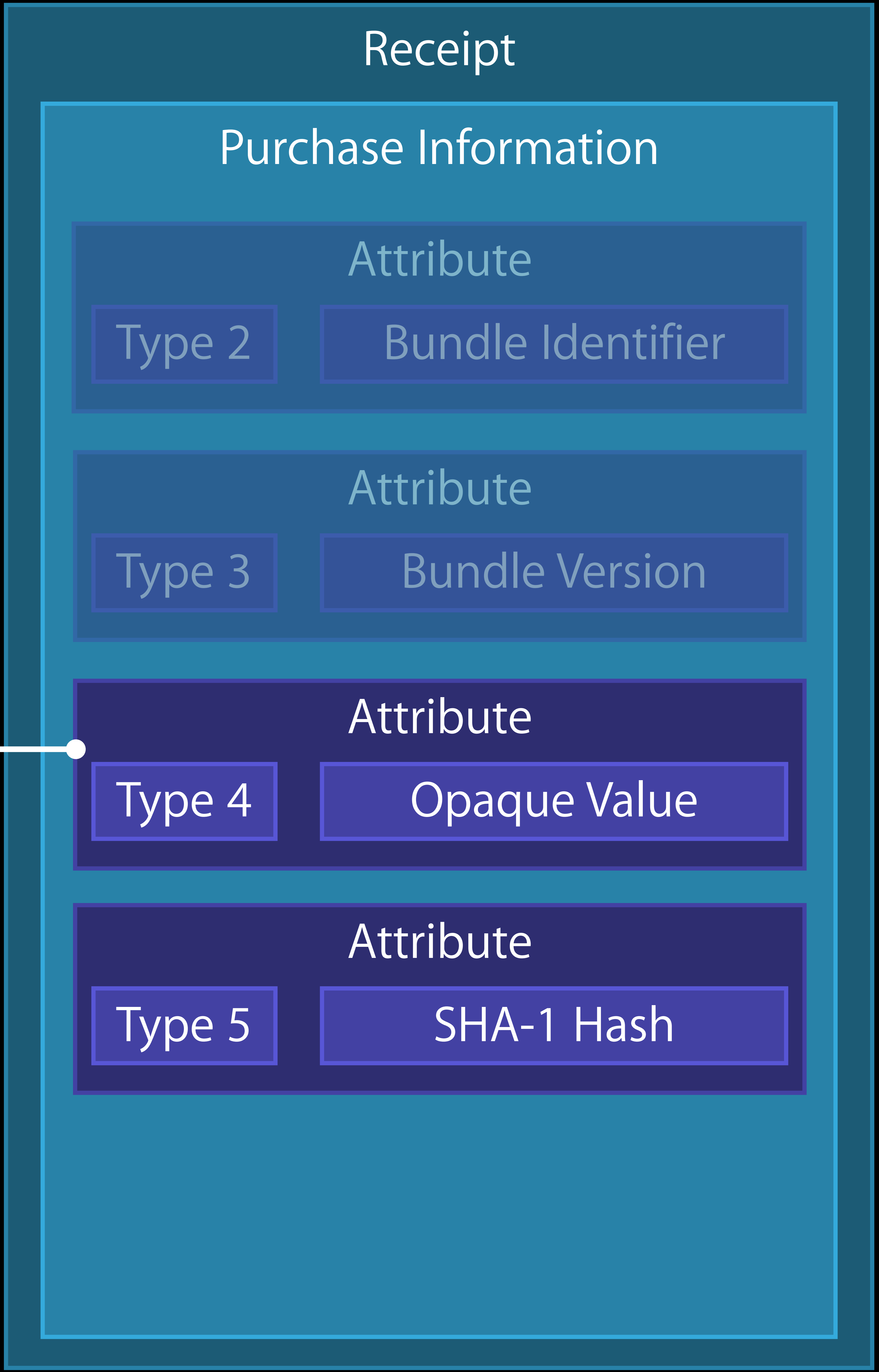


Data Encoding

ASN.1 Format

Receipt payload format definition

```
ReceiptModule DEFINITIONS ::=
BEGIN

ReceiptAttribute ::= SEQUENCE {
    type    INTEGER,
    version INTEGER,
    value   OCTET STRING
}

Payload ::= SET OF ReceiptAttribute

END
```

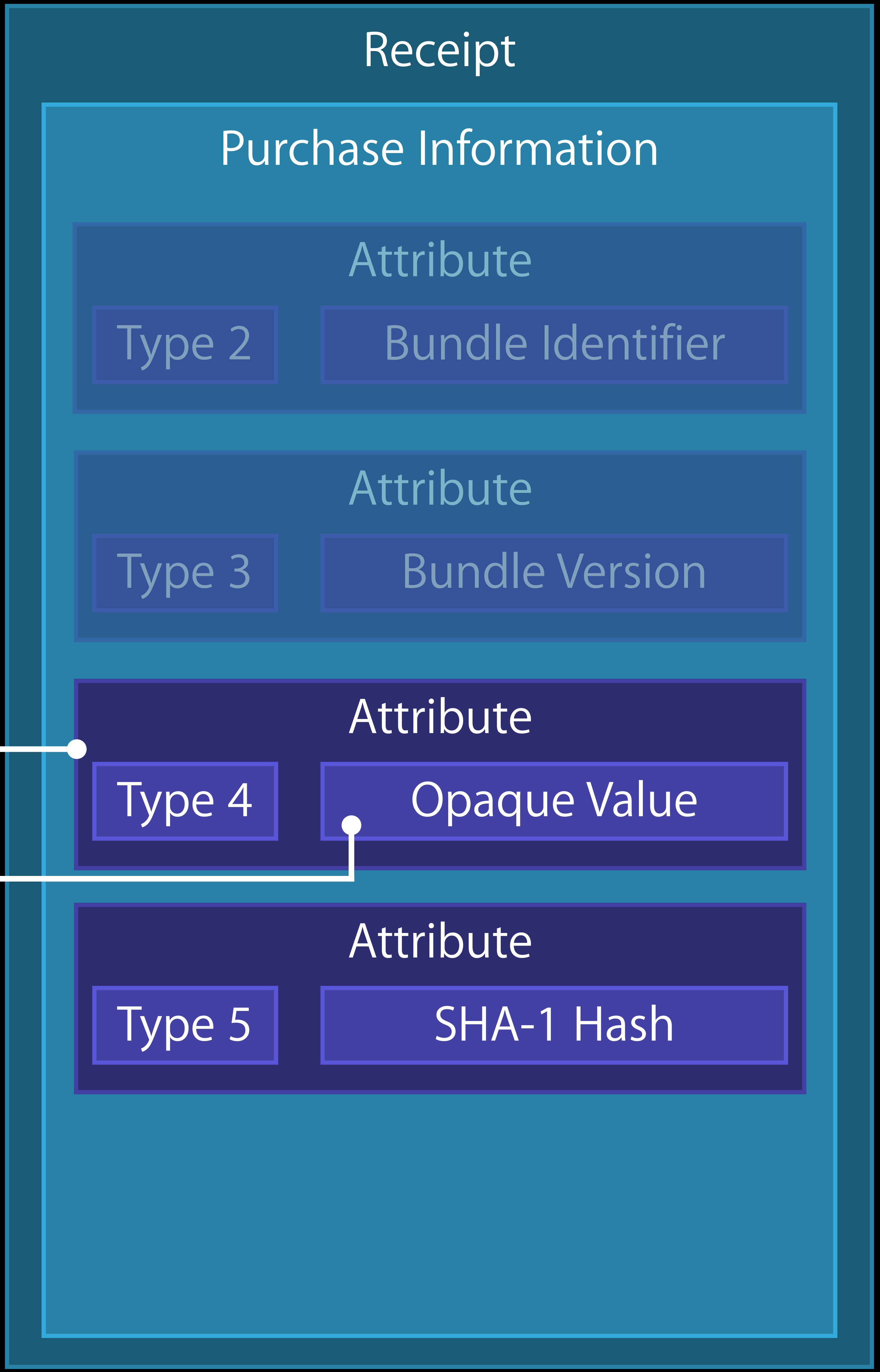


Data Encoding

ASN.1 Format

Receipt payload format definition

```
ReceiptModule DEFINITIONS ::=
BEGIN

ReceiptAttribute ::= SEQUENCE {
    type    INTEGER,
    version INTEGER,
    value   OCTET STRING
}

Payload ::= SET OF ReceiptAttribute

END
```



Working with ASN.1

Working with ASN.1

Open Standard

Working with ASN.1

Open Standard

Very widely used

Working with ASN.1

Open Standard

Very widely used

Options

Working with ASN.1

Open Standard

Very widely used

Options

- OpenSSL, ASN1C, etc.

Working with ASN.1

Open Standard

Very widely used

Options

- OpenSSL, ASN1C, etc.
- Create your own parser

OpenSSL Example

OpenSSL Example

```
// p7 is the same PKCS7 Structure
```

OpenSSL Example

```
// p7 is the same PKCS7 Structure
ASN1_OCTET_STRING *octets = p7->d.sign->contents->d.data;
```

OpenSSL Example

```
// p7 is the same PKCS7 Structure
ASN1_OCTET_STRING *octets = p7->d.sign->contents->d.data;
// Call ASN1_get_object to parse objects
```

OpenSSL Example

```
// p7 is the same PKCS7 Structure
ASN1_OCTET_STRING *octets = p7->d.sign->contents->d.data;
// Call ASN1_get_object to parse objects
ASN1_get_object(…);
```

OpenSSL Example

```
// p7 is the same PKCS7 Structure
ASN1_OCTET_STRING *octets = p7->d.sign->contents->d.data;
// Call ASN1_get_object to parse objects
ASN1_get_object(…);
```

```
ReceiptModule
-> Payload
   -> ReceiptAttribute
       -> Type
       -> Version
       -> Value
   -> ReceiptAttribute
       -> Type
       -> Version
       -> Value
```

OpenSSL Example

```
// p7 is the same PKCS7 Structure
ASN1_OCTET_STRING *octets = p7->d.sign->contents->d.data;
// Call ASN1_get_object to parse objects
ASN1_get_object(…);
```

```
ReceiptModule
-> Payload
   -> ReceiptAttribute
       -> Type
       -> Version
       -> Value
   -> ReceiptAttribute
       -> Type
       -> Version
       -> Value
```

```
ReceiptModule DEFINITIONS ::=
BEGIN

ReceiptAttribute ::= SEQUENCE {
    type    INTEGER,
    version INTEGER,
    value   OCTET STRING
}

Payload ::= SET OF ReceiptAttribute

END
```

OpenSSL Example

OpenSSL Example

```
// Iterate over child objects (Attributes)
```

OpenSSL Example

```
// Iterate over child objects (Attributes)
while (p < end)
{
```

OpenSSL Example

```
// Iterate over child objects (Attributes)
while (p < end)
{
  ASN1_get_object(&p, &length, &type, &xclass, end - p);   // Attribute
  const unsigned char *seq_end = p + length;
```

OpenSSL Example

```
// Iterate over child objects (Attributes)
while (p < end)
{
  ASN1_get_object(&p, &length, &type, &xclass, end - p);   // Attribute
  const unsigned char *seq_end = p + length;

  ASN1_get_object(&p, &length, &type, &xclass, seq_end - p);  // Type
  int attr_type = p[0];
  p += length; // Move the pointer to the next object
```

OpenSSL Example

```
// Iterate over child objects (Attributes)
while (p < end)
{
  ASN1_get_object(&p, &length, &type, &xclass, end - p);   // Attribute
  const unsigned char *seq_end = p + length;

  ASN1_get_object(&p, &length, &type, &xclass, seq_end - p);  // Type
  int attr_type = p[0];
  p += length; // Move the pointer to the next object

  ASN1_get_object(&p, &length, &type, &xclass, seq_end - p);
```

OpenSSL Example

```
// Iterate over child objects (Attributes)
while (p < end)
{
  ASN1_get_object(&p, &length, &type, &xclass, end - p);   // Attribute
  const unsigned char *seq_end = p + length;

  ASN1_get_object(&p, &length, &type, &xclass, seq_end - p);  // Type
  int attr_type = p[0];
  p += length; // Move the pointer to the next object

  ASN1_get_object(&p, &length, &type, &xclass, seq_end - p);
  switch (attr_type) {
```

OpenSSL Example

```
// Iterate over child objects (Attributes)
while (p < end)
{
  ASN1_get_object(&p, &length, &type, &xclass, end - p);   // Attribute
  const unsigned char *seq_end = p + length;

  ASN1_get_object(&p, &length, &type, &xclass, seq_end - p);  // Type
  int attr_type = p[0];
  p += length; // Move the pointer to the next object

  ASN1_get_object(&p, &length, &type, &xclass, seq_end - p);
  switch (attr_type) {
     case 2: {
```

OpenSSL Example

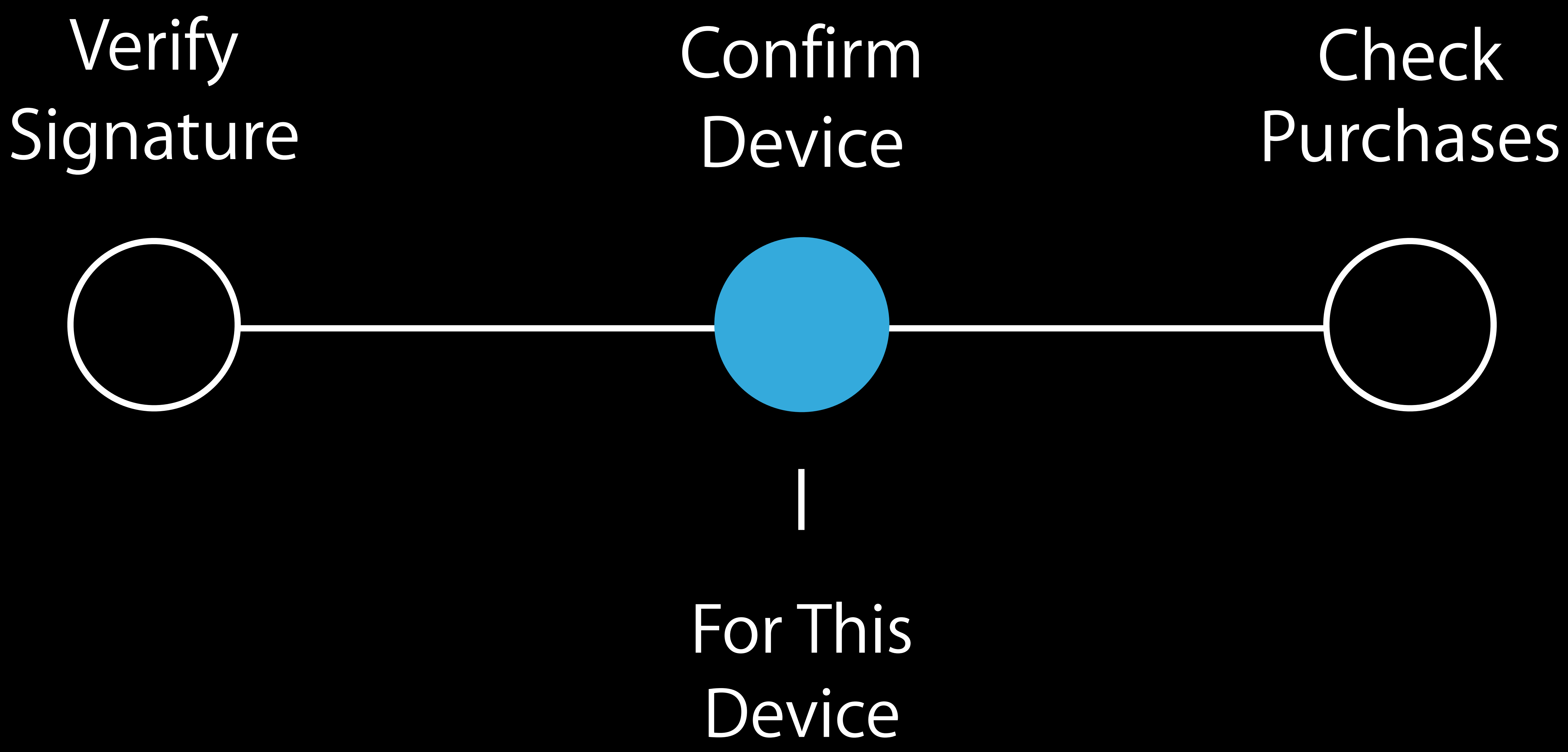
```
// Iterate over child objects (Attributes)
while (p < end)
{
  ASN1_get_object(&p, &length, &type, &xclass, end - p);   // Attribute
  const unsigned char *seq_end = p + length;

  ASN1_get_object(&p, &length, &type, &xclass, seq_end - p);  // Type
  int attr_type = p[0];
  p += length; // Move the pointer to the next object

  ASN1_get_object(&p, &length, &type, &xclass, seq_end - p);
  switch (attr_type) {
      case 2: {
          // BundleID (Attribute 2)
```

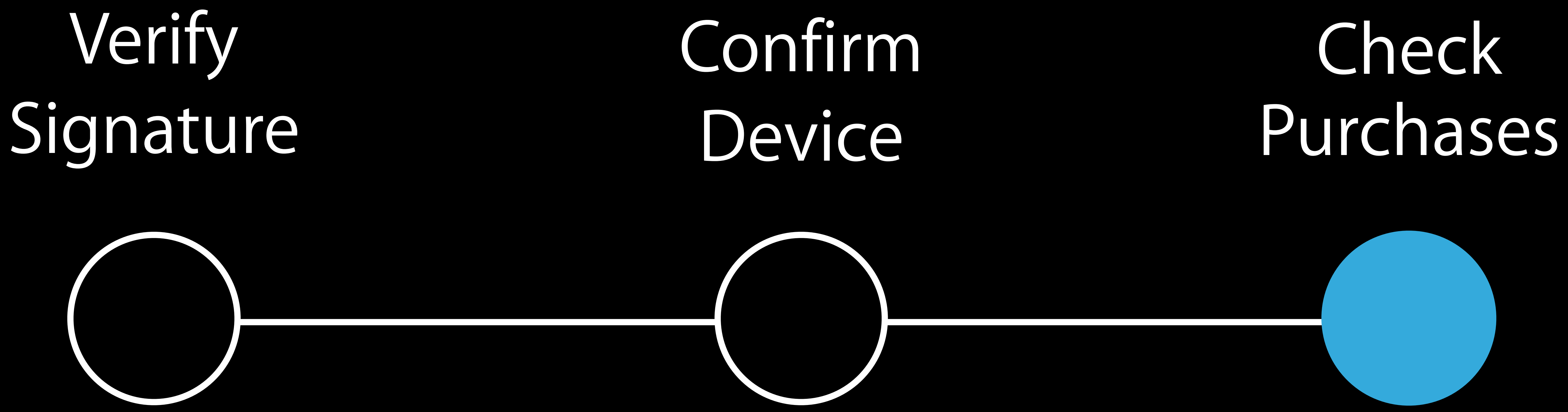
OpenSSL Example

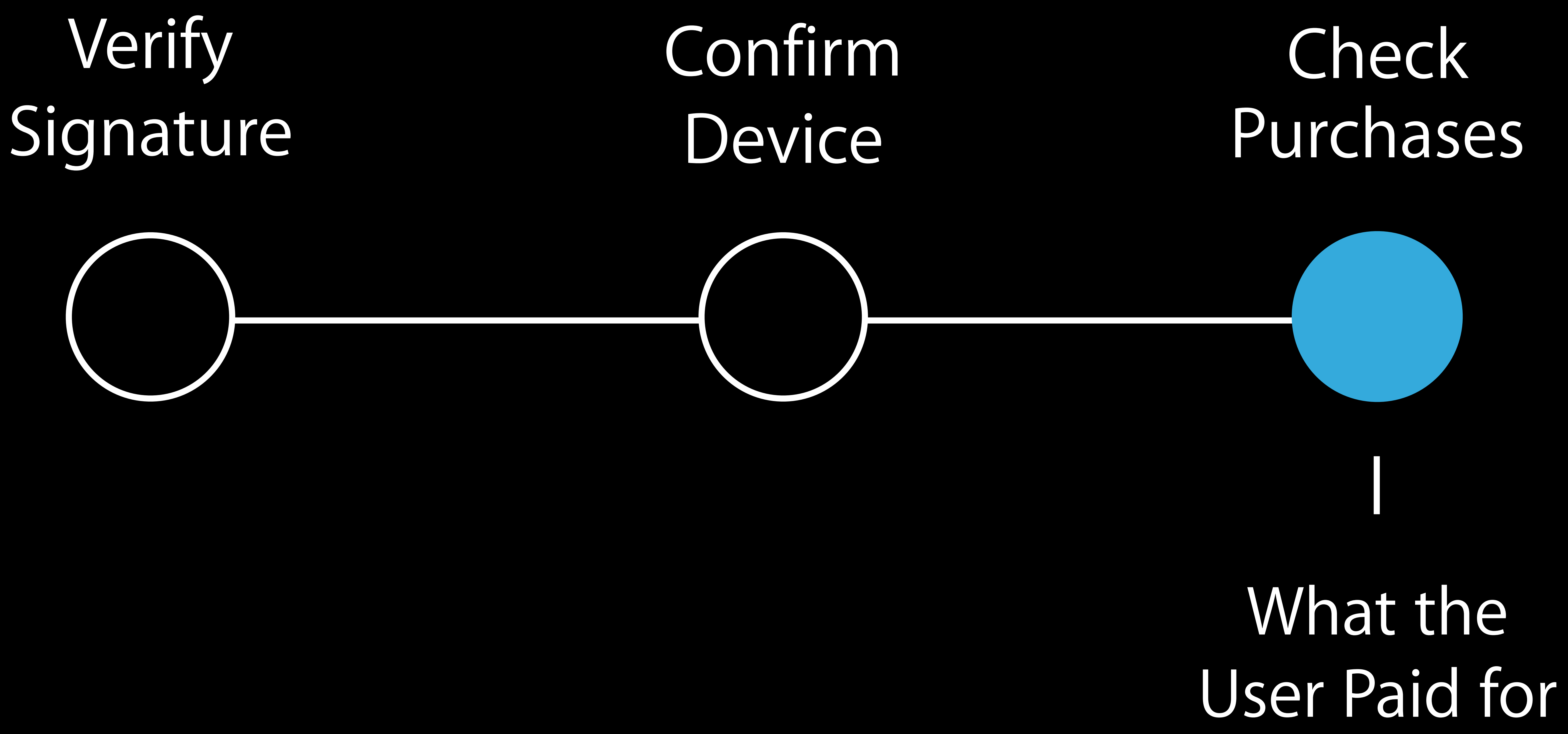
```
// Iterate over child objects (Attributes)
while (p < end)
{
  ASN1_get_object(&p, &length, &type, &xclass, end - p);   // Attribute
  const unsigned char *seq_end = p + length;

  ASN1_get_object(&p, &length, &type, &xclass, seq_end - p);  // Type
  int attr_type = p[0];
  p += length; // Move the pointer to the next object

  ASN1_get_object(&p, &length, &type, &xclass, seq_end - p);
  switch (attr_type) {
      case 2: {
          // BundleID (Attribute 2)
          // Use ASN_get_object again to get the string
      }
  }
}
```

Demo

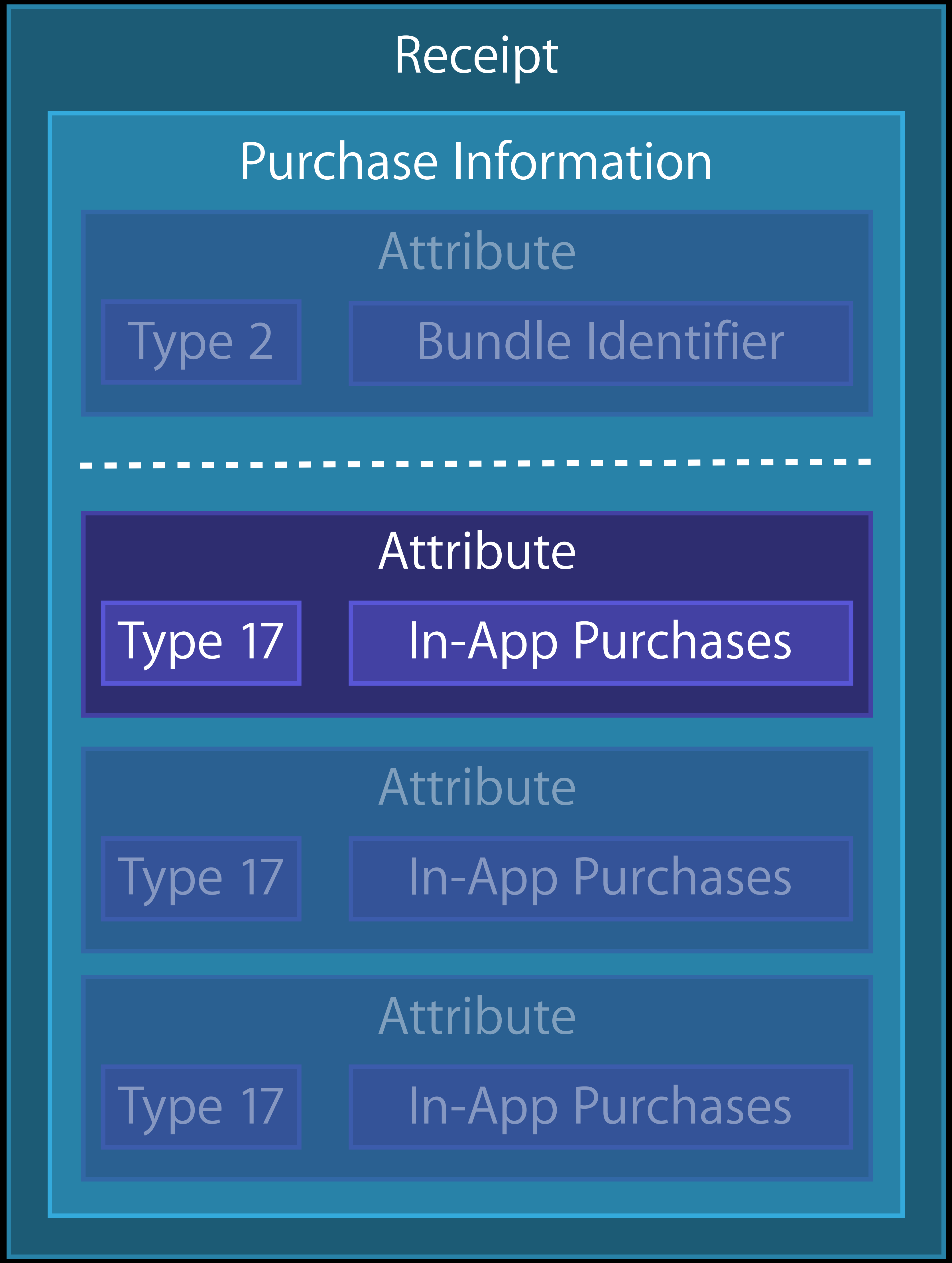
Using OpenSSL for ASN.1 parsing


Verify Signature
Confirm Device
Check Purchases
For This Device

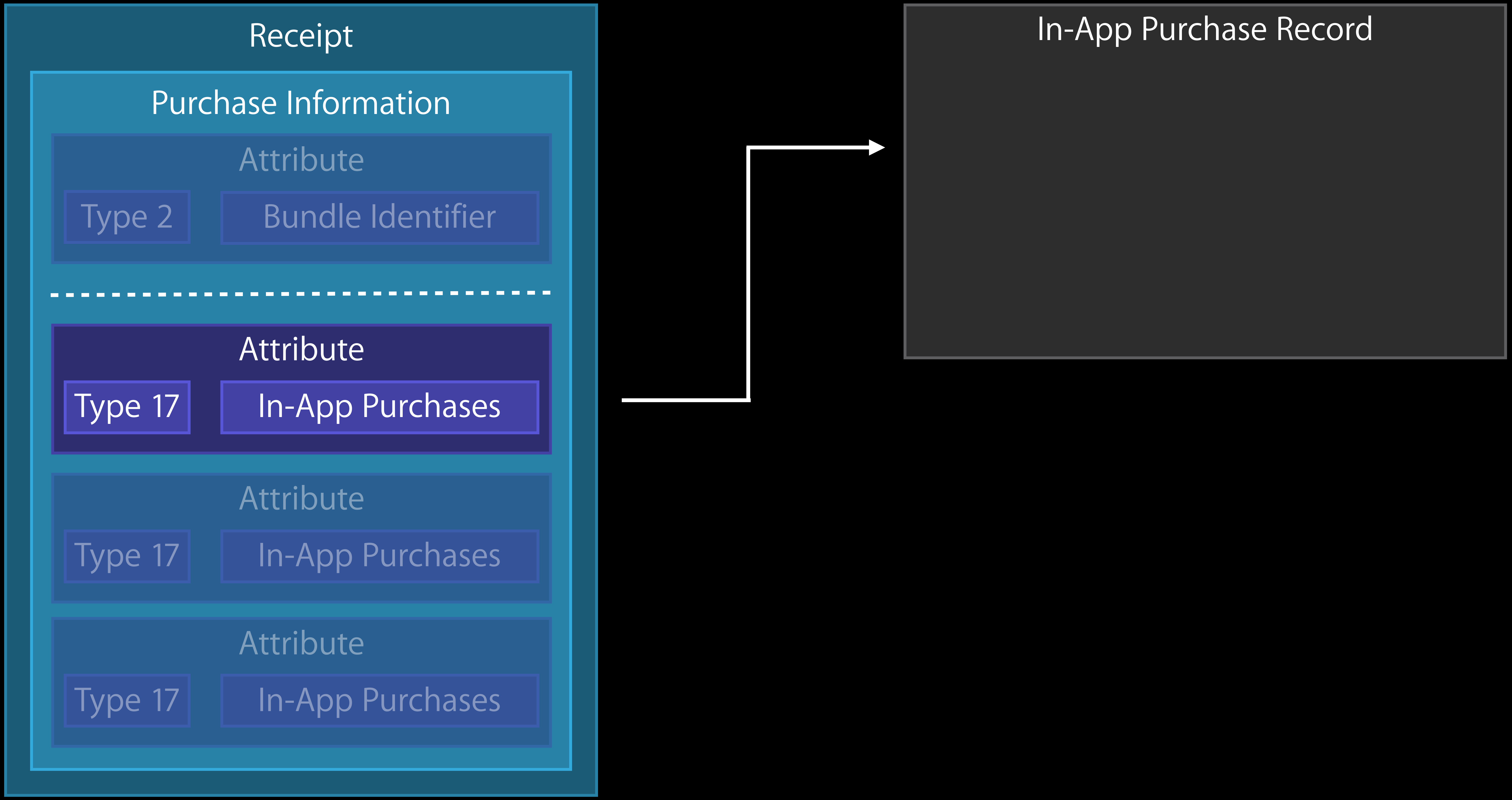

Verify Signature
Confirm Device
Check Purchases


Verify Signature
Confirm Device
Check Purchases
What the User Paid for

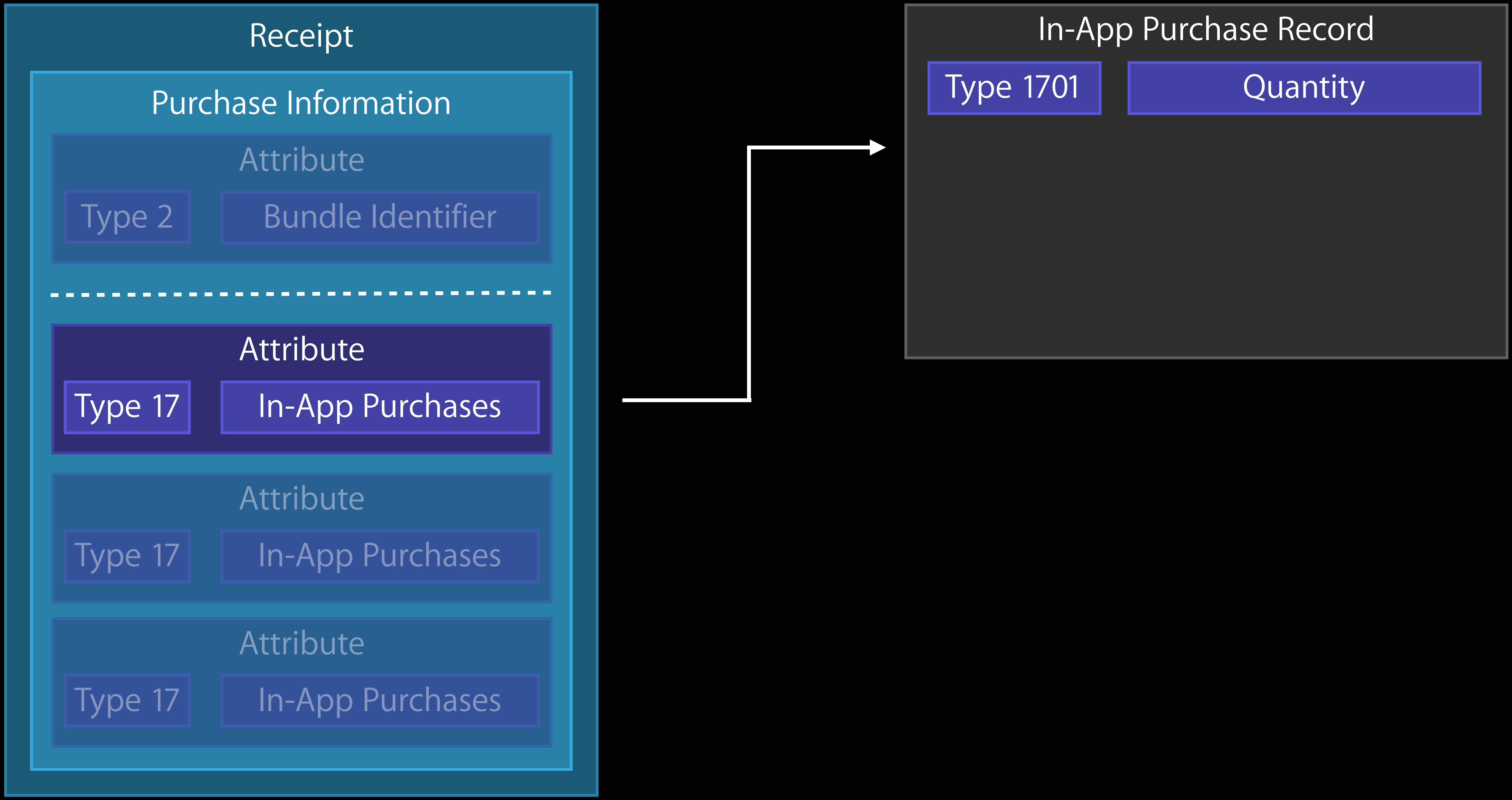
In-App Purchase Attributes



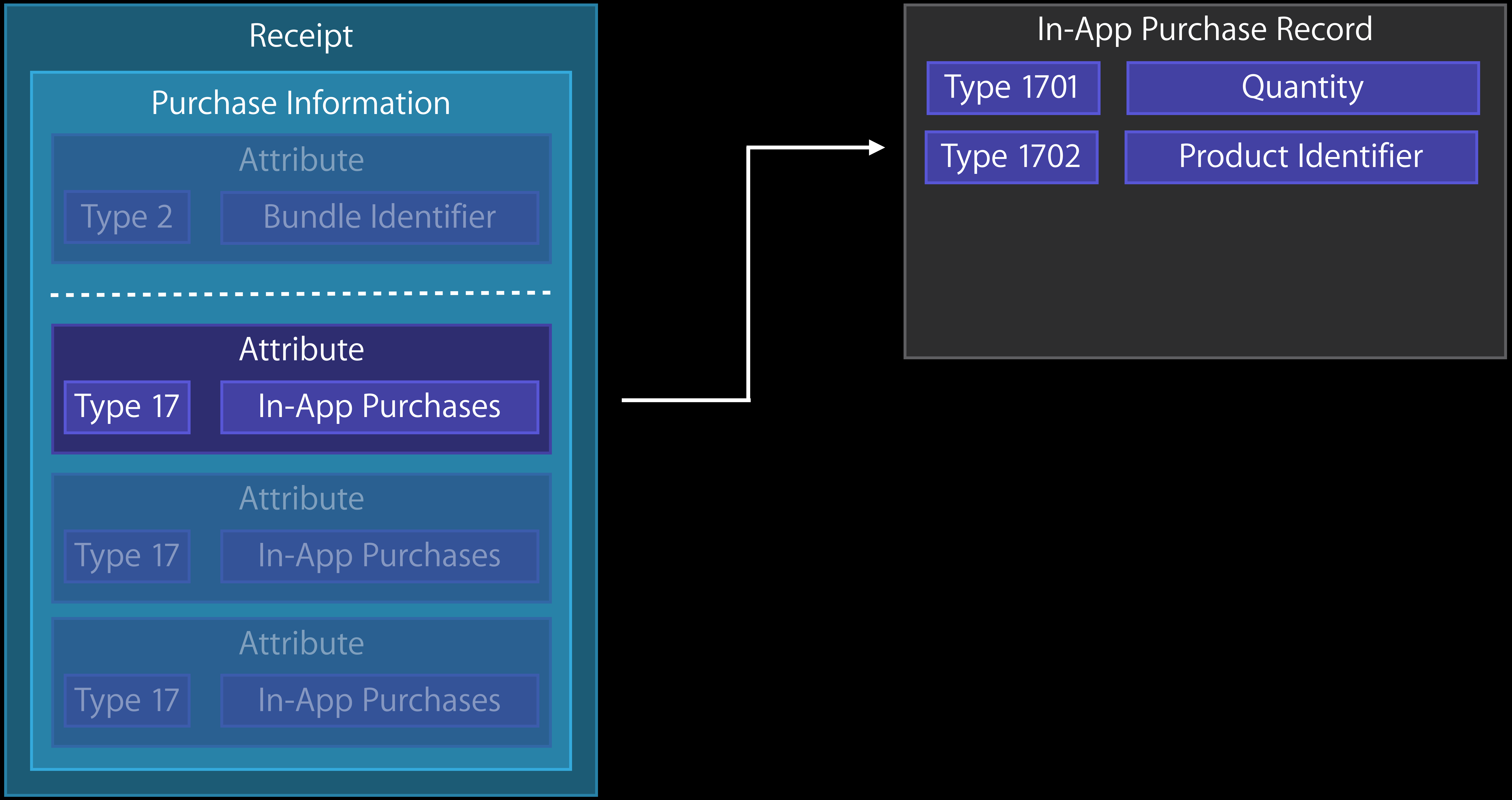
In-App Purchase Attributes



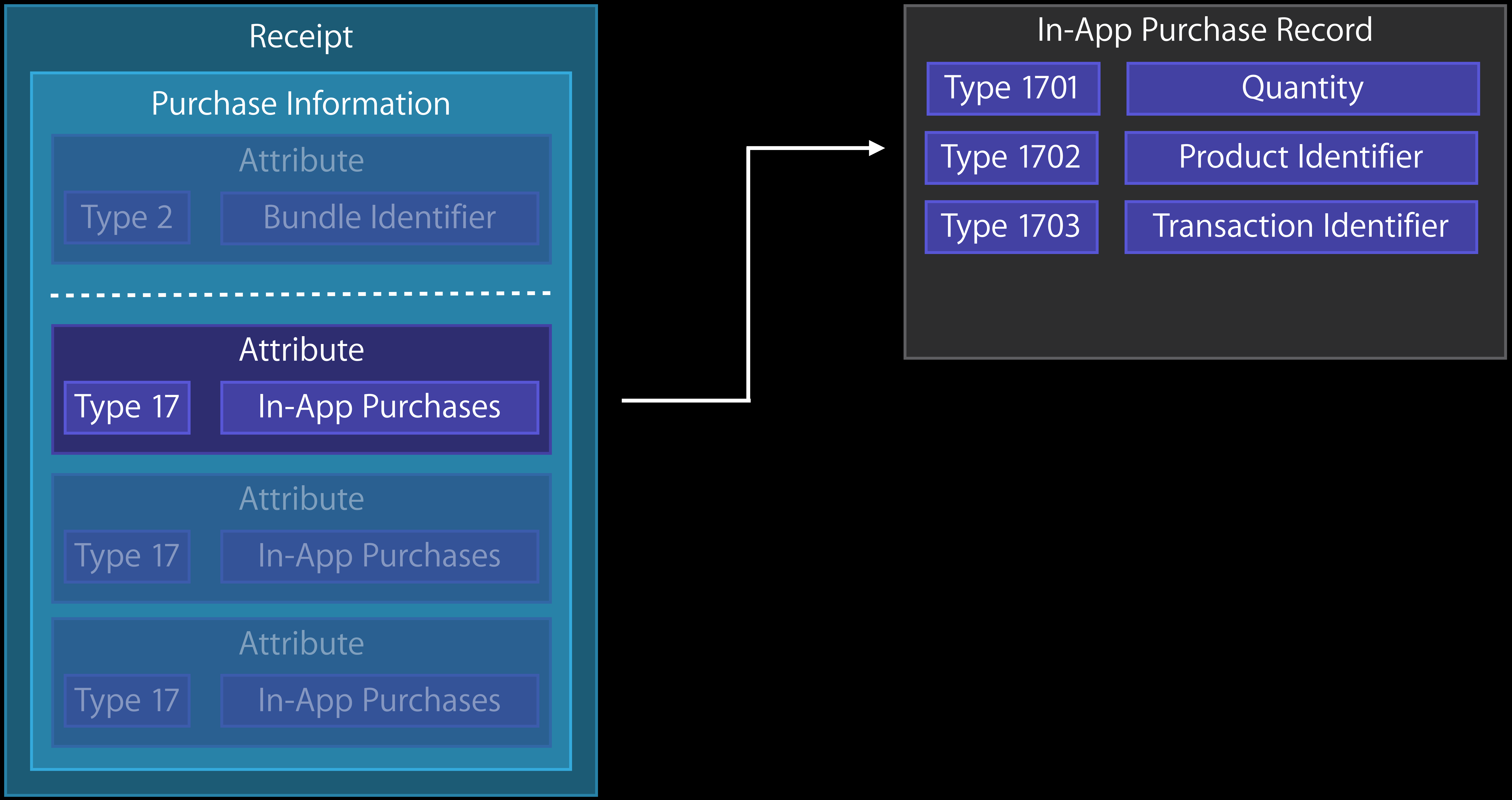
In-App Purchase Attributes



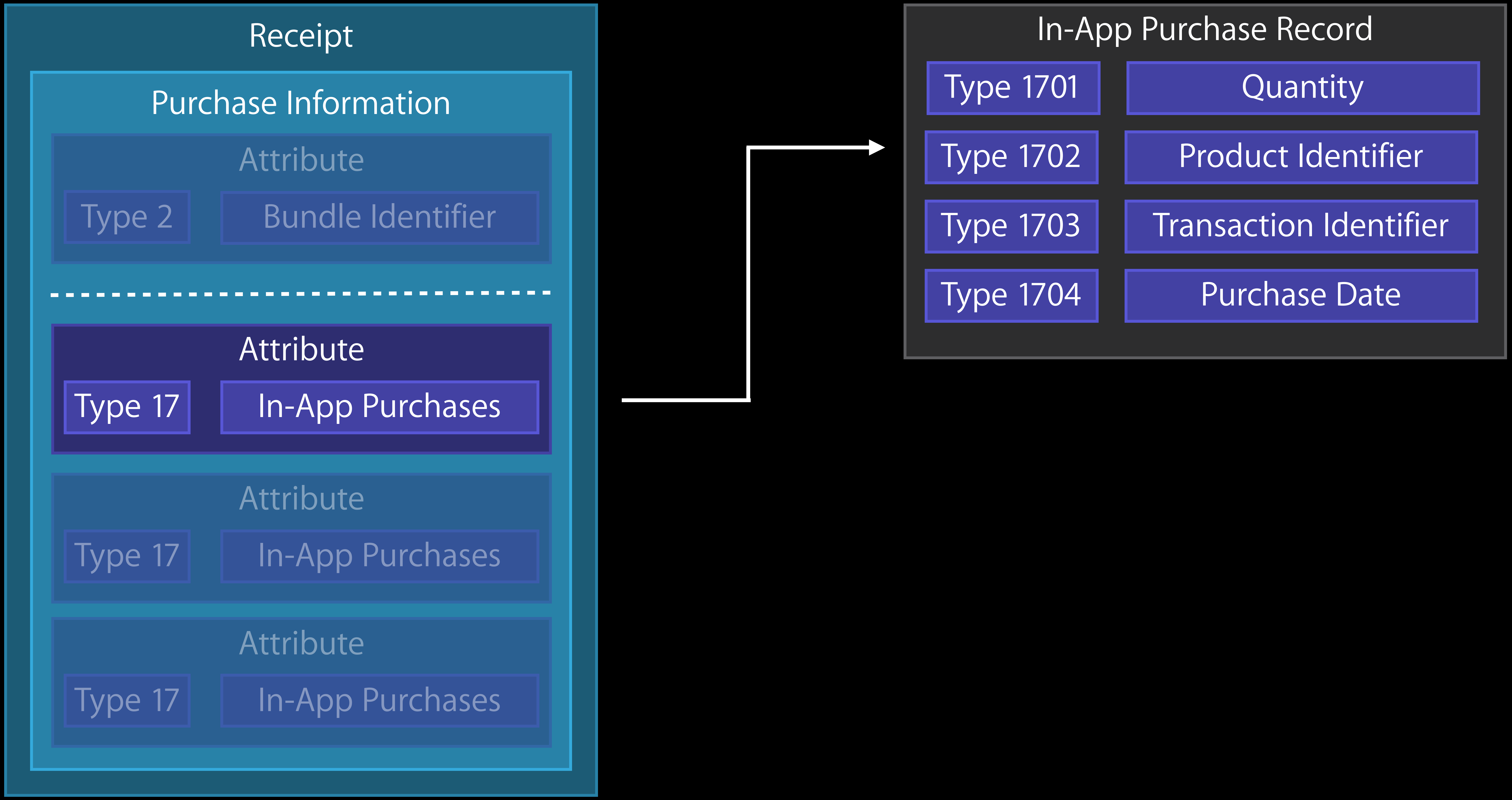
In-App Purchase Attributes



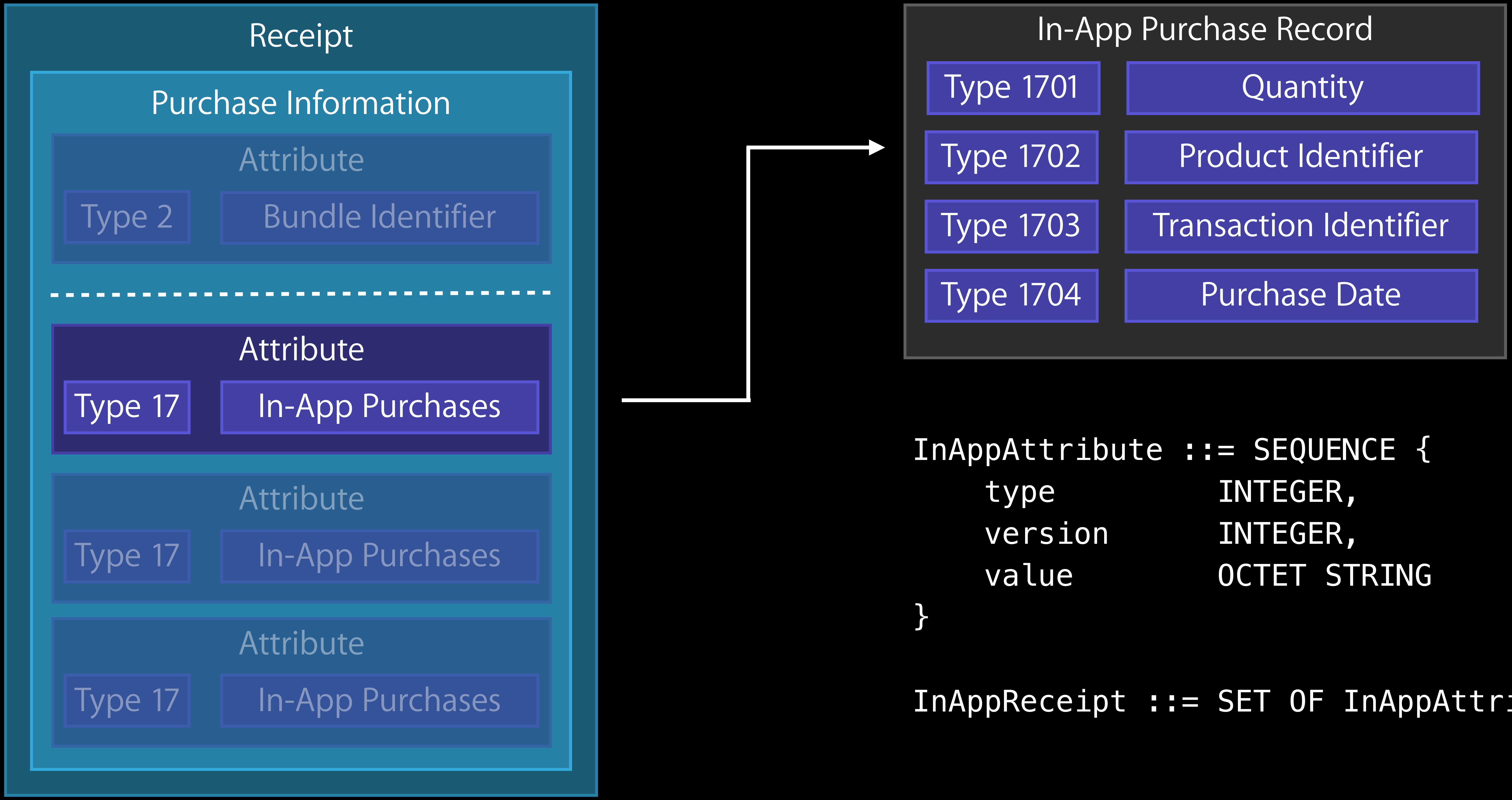
In-App Purchase Attributes



In-App Purchase Attributes

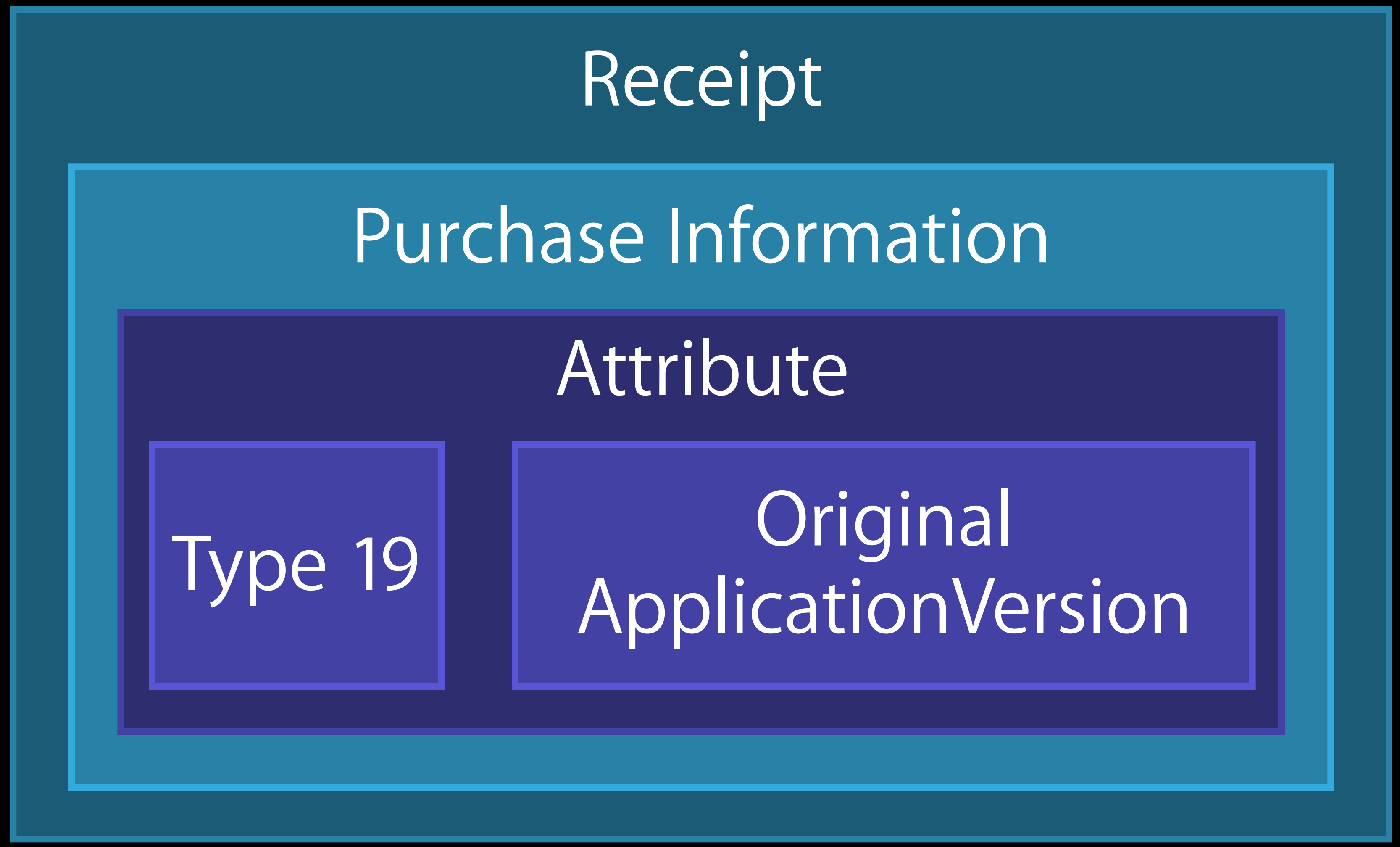


In-App Purchase Attributes



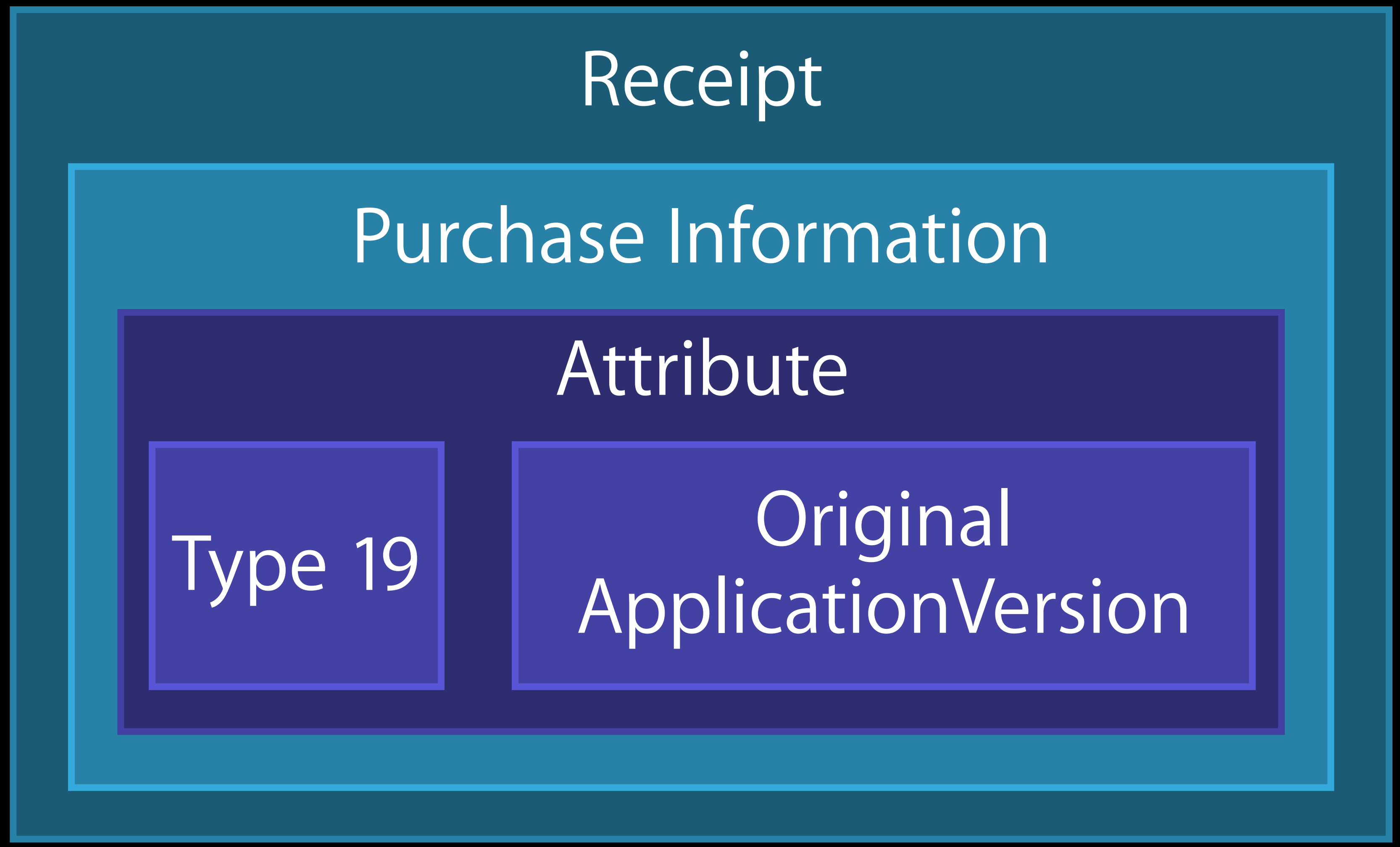
```
InAppAttribute ::= SEQUENCE {
    type        INTEGER,
    version     INTEGER,
    value       OCTET STRING
}

InAppReceipt ::= SET OF InAppAttribute
```

Transition to Freemium



Transition to Freemium

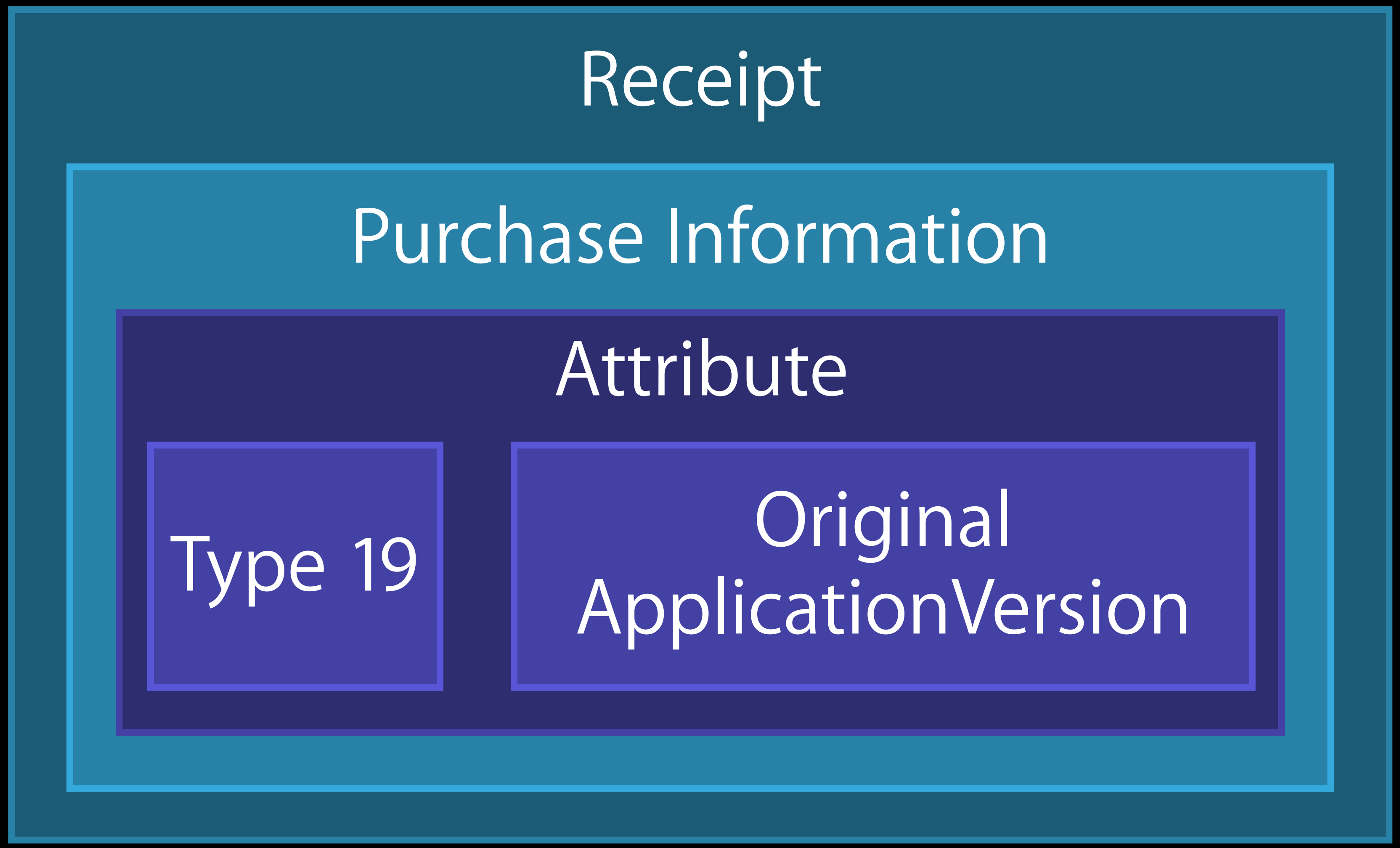
Original application version in the receipt



Transition to Freemium

Original application version in the receipt

Know whether to treat the app as the paid version, or the freemium version



Transaction Lifecycle

Transaction Lifecycle

Consumable and non-renewing subscriptions

Transaction Lifecycle

Consumable and non-renewing subscriptions

- Will only appear once

Transaction Lifecycle

Consumable and non-renewing subscriptions

- Will only appear once
- In the receipt issued at time of purchase

Transaction Lifecycle

Consumable and non-renewing subscriptions

- Will only appear once
- In the receipt issued at time of purchase
- Will not be present in subsequent receipts issued

Transaction Lifecycle

Consumable and non-renewing subscriptions

- Will only appear once
- In the receipt issued at time of purchase
- Will not be present in subsequent receipts issued

Non-consumable and auto-renewable subscriptions

Transaction Lifecycle

Consumable and non-renewing subscriptions

- Will only appear once
- In the receipt issued at time of purchase
- Will not be present in subsequent receipts issued

Non-consumable and auto-renewable subscriptions

- Always in the receipt

Transaction Lifecycle

Consumable and non-renewing subscriptions

- Will only appear once
- In the receipt issued at time of purchase
- Will not be present in subsequent receipts issued

Non-consumable and auto-renewable subscriptions

- Always in the receipt
- Can be restored via StoreKit API

Handling Invalid Receipts

Receipt Refresh on iOS

Receipt Refresh on iOS

If the receipt doesn’t exist or is invalid

Receipt Refresh on iOS

If the receipt doesn’t exist or is invalid

Refresh the receipt using StoreKit

Receipt Refresh on iOS

If the receipt doesn't exist or is invalid

Refresh the receipt using StoreKit

```
// Refresh the Receipt
SKReceiptRefreshRequest *request = [SKReceiptRefreshRequest alloc] init];
[request setDelegate:self];
[request start];
```

Receipt Refresh on iOS

If the receipt doesn’t exist or is invalid

Refresh the receipt using StoreKit

```
// Refresh the Receipt
SKReceiptRefreshRequest *request = [SKReceiptRefreshRequest alloc] init];
[request setDelegate:self];
[request start];
```

Receipt refresh will require network

Receipt Refresh on iOS

If the receipt doesn't exist or is invalid

Refresh the receipt using StoreKit

```
// Refresh the Receipt
SKReceiptRefreshRequest *request = [SKReceiptRefreshRequest alloc] init];
[request setDelegate:self];
[request start];
```

Receipt refresh will require network

Store sign-in will be required

Receipt Refresh on iOS

If the receipt doesn't exist or is invalid

Refresh the receipt using StoreKit

```
// Refresh the Receipt
SKReceiptRefreshRequest *request = [SKReceiptRefreshRequest alloc] init];
[request setDelegate:self];
[request start];
```

Receipt refresh will require network

Store sign-in will be required

Avoid continuous loop of validate-and-refresh

Receipt Refresh on OS X

Receipt Refresh on OS X

If the receipt is invalid

Receipt Refresh on OS X

If the receipt is invalid

Exit with code 173 to refresh receipt

Receipt Refresh on OS X

If the receipt is invalid

Exit with code 173 to refresh receipt

```
// Receipt is invalid
exit(173);
```

Receipt Refresh on OS X

If the receipt is invalid

Exit with code 173 to refresh receipt

```
// Receipt is invalid
exit(173);
```

Receipt refresh will require network

Receipt Refresh on OS X

If the receipt is invalid

Exit with code 173 to refresh receipt

```
// Receipt is invalid
exit(173);
```

Receipt refresh will require network

Store sign-in will be required

Invalid Receipt User Experience

Invalid Receipt User Experience

Invalid or missing receipts will happen

Invalid Receipt User Experience

Invalid or missing receipts will happen

- Receipt refresh may not be possible

Invalid Receipt User Experience

Invalid or missing receipts will happen

- Receipt refresh may not be possible
- You decide how to handle this

Invalid Receipt User Experience

Invalid or missing receipts will happen

- Receipt refresh may not be possible
- You decide how to handle this

Match the user experience to the product value

Invalid Receipt User Experience

Invalid or missing receipts will happen

- Receipt refresh may not be possible
- You decide how to handle this

Match the user experience to the product value

- Allow full access to content and features

Invalid Receipt User Experience

Invalid or missing receipts will happen

- Receipt refresh may not be possible
- You decide how to handle this

Match the user experience to the product value

- Allow full access to content and features
- Limit access

Invalid Receipt User Experience

Invalid or missing receipts will happen

- Receipt refresh may not be possible
- You decide how to handle this

Match the user experience to the product value

- Allow full access to content and features
- Limit access
- Block functionality

Invalid Receipt User Experience

Invalid or missing receipts will happen

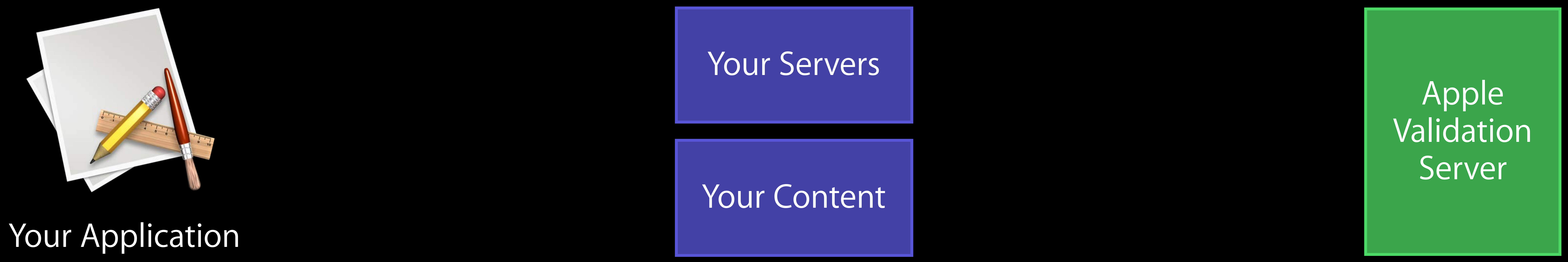
- Receipt refresh may not be possible
- You decide how to handle this

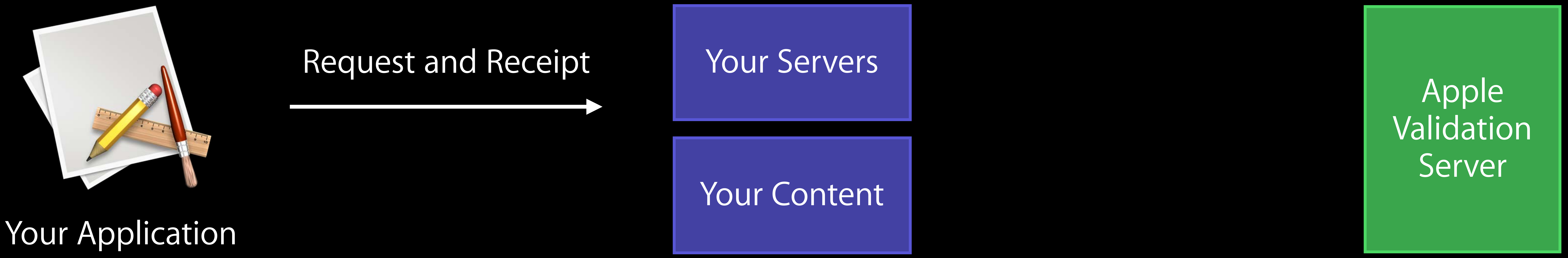
Match the user experience to the product value

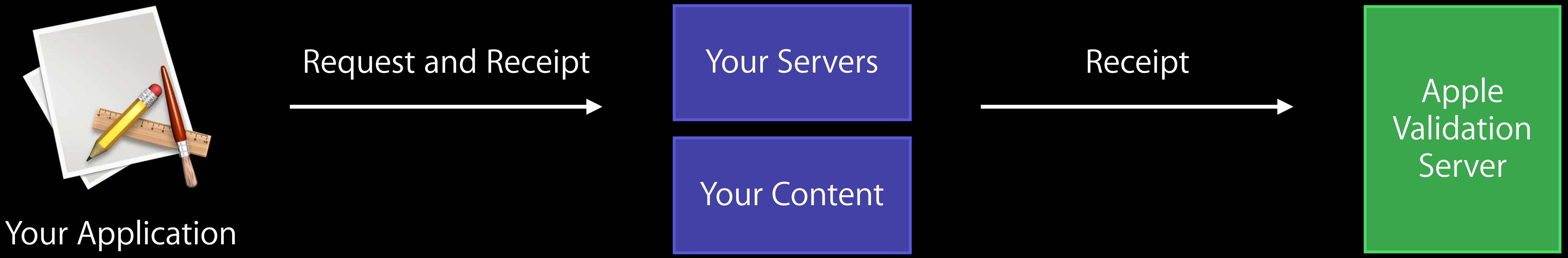
- Allow full access to content and features
- Limit access
- Block functionality
- Quit app (OS X only)

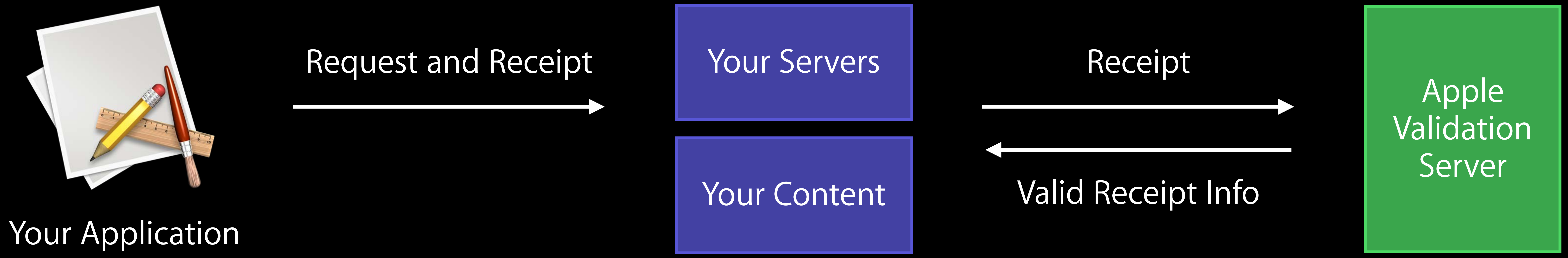
Online Validation

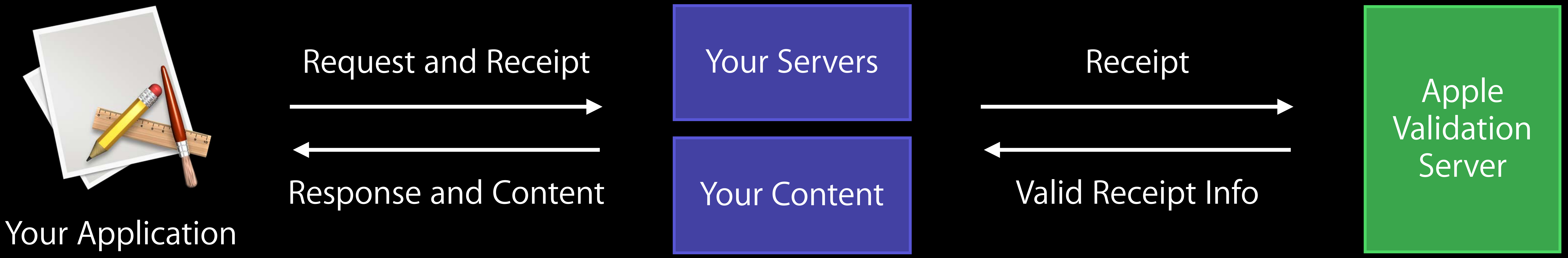
Server-to-Server


Your Application
Your Servers
Your Content
Apple Validation Server


Your Application
Request and Receipt
Your Servers
Your Content
Apple Validation Server


Your Application
Request and Receipt
Your Servers
Your Content
Receipt
Apple Validation Server


Your Application
Request and Receipt
Your Servers
Your Content
Receipt
Valid Receipt Info
Apple Validation Server


Your Application
Request and Receipt
Response and Content
Your Servers
Your Content
Receipt
Valid Receipt Info
Apple Validation Server

Server-to-Server Validation

Server-to-Server Validation

Allows your servers to validate the receipt before issuing content

Server-to-Server Validation

Allows your servers to validate the receipt before issuing content

Your app sends the receipt to your servers

Server-to-Server Validation

Allows your servers to validate the receipt before issuing content

Your app sends the receipt to your servers

- Your server sends the receipt to Apple’s server

Server-to-Server Validation

Allows your servers to validate the receipt before issuing content

Your app sends the receipt to your servers

- Your server sends the receipt to Apple's server
- Never send the receipt directly from your app to Apple's server

Server-to-Server Validation

Allows your servers to validate the receipt before issuing content

Your app sends the receipt to your servers

- Your server sends the receipt to Apple’s server
- Never send the receipt directly from your app to Apple’s server

Response is in JSON

Test Your Implementation

Using the App Store Test Environment

Test Thoroughly

Test Thoroughly

No receipt

Test Thoroughly

No receipt

Invalid receipt

Test Thoroughly

No receipt

Invalid receipt

Valid on refresh

Test Thoroughly

No receipt

Invalid receipt

Valid on refresh

Invalid on refresh

Test Thoroughly

No receipt

Invalid receipt

Valid on refresh

Invalid on refresh

Volume Purchase Program receipts

Test Thoroughly

No receipt

Invalid receipt

Valid on refresh

Invalid on refresh

Volume Purchase Program receipts

These are not edge cases!

Testing on iOS

Testing on iOS

Run the app from Xcode

Testing on iOS

Run the app from Xcode

Perform an In-App Purchase to get a receipt

Testing on iOS

Run the app from Xcode

Perform an In-App Purchase to get a receipt

Must be signed with Development Certificate

Testing on OS X

Testing on OS X

Build the app in Xcode

Testing on OS X

Build the app in Xcode

Run the app from Finder

Testing on OS X

Build the app in Xcode

Run the app from Finder

Exit with code 173 to get a receipt

Testing on OS X

Build the app in Xcode

Run the app from Finder

Exit with code 173 to get a receipt

Must be signed with Development Certificate

Must be signed with Development Certificate

App Submission

App Review and Receipts

	Certificate	Receipt Type
Development	Development	Test
For Sale	Production	Production

App Review and Receipts

	Certificate	Receipt Type
Development	Development	Test
App Review	Production	
For Sale	Production	Production

App Review and Receipts

	Certificate	Receipt Type
Development	Development	Test
App Review	Production	Test
For Sale	Production	Production

App Review and Receipts

	Certificate	Receipt Type
Development	Development	Test
App Review	Production	Test
For Sale	Production	Production

Do not invalidate Test Environment receipts—Your app will be rejected

More Information

Evangelism
evangelism@apple.com

Documentation
Receipt Validation Programming Guide
https://developer.apple.com

Apple Developer Forums
http://devforums.apple.com

Related Sessions

●	Optimizing In-App Purchases	Nob Hill	Wednesday 3:15PM
●	Designing a Great In-App Purchase Experience	Nob Hill	Wednesday 11:30AM

Labs

●	StoreKit and Receipts Lab	Services Lab	Friday 10:15AM
●	Services Open Lab	Services Lab	Friday 2:00PM

WWDC14