

# Using BCERT

Steve Burnett  
RSA Data Security Inc.  
burnetts@rsa.com

RSA Data Security Conference  
January 30, 1997



# Topics

---

- 1. What is BCERT**
- 2. The BCERT Model**
- 3. Examples**
- 4. X.509 v3 Extensions**
- 5. User-Defined Extensions**



# BCERT

- Certificate Management Toolkit
- Written in C
- Portable
- Object-Oriented
  - » flexible
  - » extensible



# BCERT

- Composes certificate requests, certificates and CRL's following the ITU-T X.509 standard.
  - DER encodes these constructs.
  - Parses BER or DER-encodings.
  - Computes and verifies digital signatures.
- Processes X.509 version 3 extensions.



# The BCERT Model

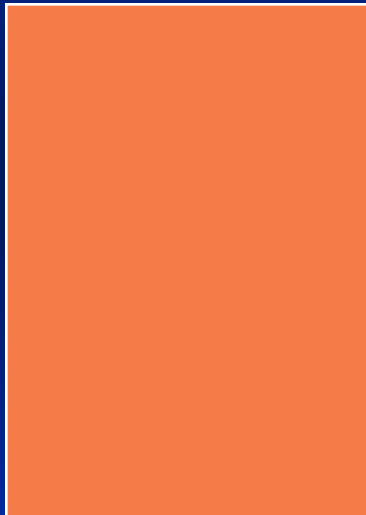
---

- **Theme 1 -- Producing Identification Constructs**
- **Theme 2 -- Reading Identification Constructs**



# The BCERT Model -- Theme 1

Object



# The BCERT Model -- Theme 1

## Input Data

“Darren Roscow”  
“Martin Bishop  
and Assoc.”  
<public key>



## Object



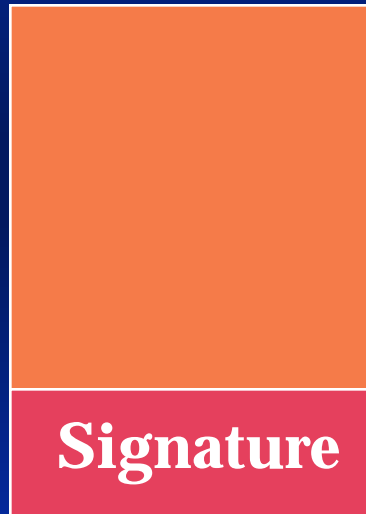
# The BCERT Model -- Theme 1

## Input Data

“Darren Roscow”  
“Martin Bishop  
and Assoc.”  
<public key>

  
private key

## Object



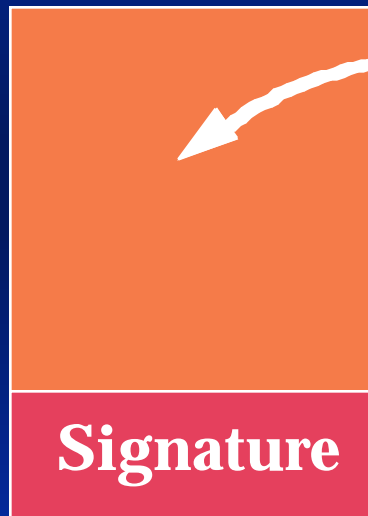
# The BCERT Model -- Theme 1

## Input Data

“Darren Roscow”  
“Martin Bishop  
and Assoc.”  
<public key>

  
private key

## Object



 **Pointer**



# The BCERT Model -- Theme 1

**Pointer**



30 82 26 f0 30 82 03 53  
30 82 02 bc a0 03 02 01  
02 02 04 20 01 00 04 30  
0d 06 09 2a 86 . . .



# The BCERT Model -- Theme

## 2

30 82 26 f0 30 82 03 53 30 82 02 bc  
a0 03 02 01 02 02 04 20 01 00 04 30  
0d 06 09 2a 86 48 86 f7 0d 01 01 05  
05 00 30 3a 31 0b 30 09 06 03 55 04  
06 13 02 55 53 31 11 30 0f 06 03 55  
04 0a 13 08 ...



# The BCERT Model -- Theme 2

Object



# The BCERT Model -- Theme 2

**Input Data**

**30 82 26 f0**

**30 82 03 53**

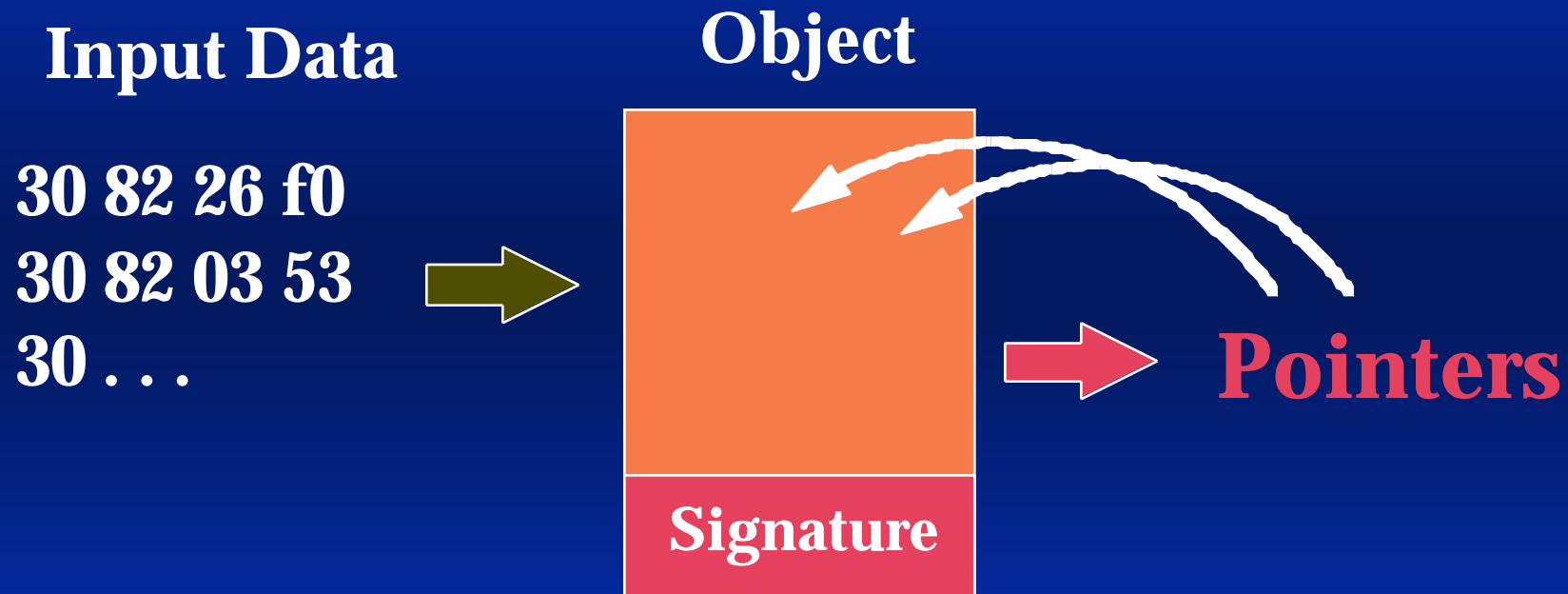
**30 ...**



**Object**



# The BCERT Model -- Theme 2



# The BCERT Model -- Theme 2

**Pointer**



**Darren Roscow  
Martin Bishop and Associates**

...

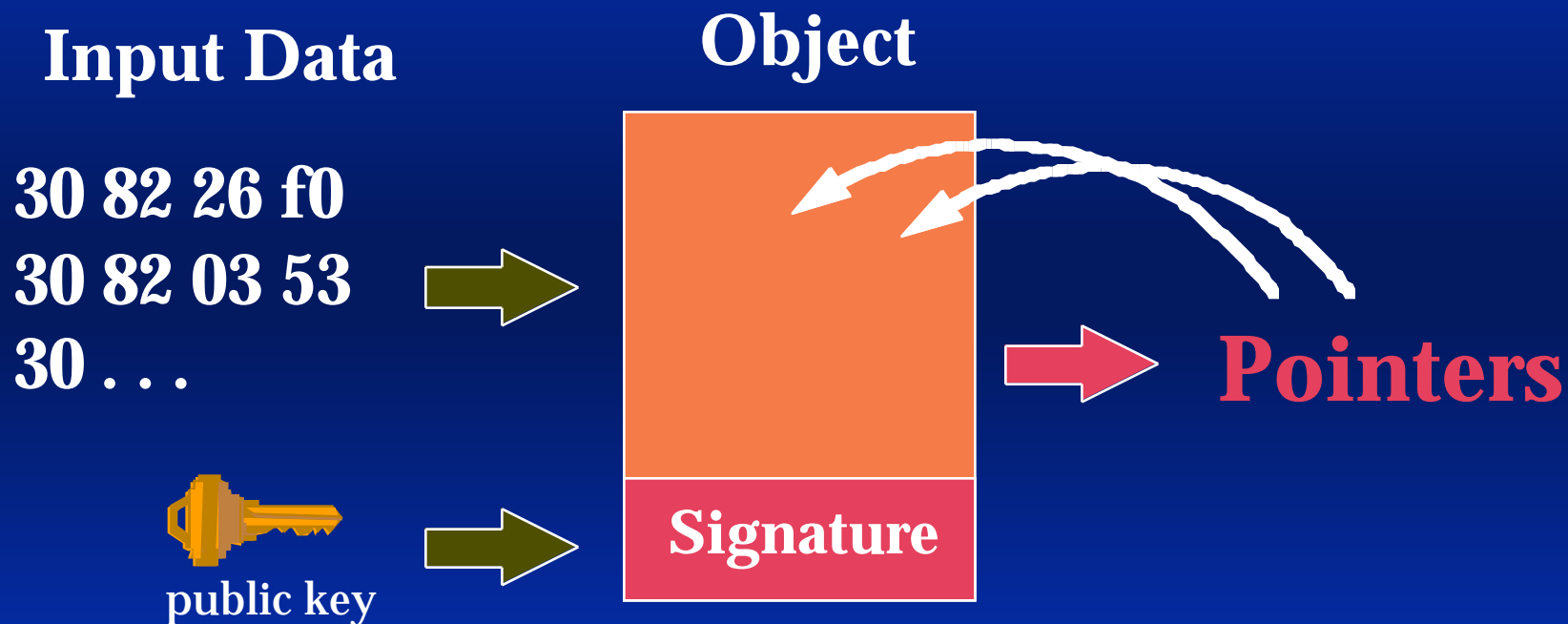
**Pointer**



**SUBJECT\_TYPE\_END\_ENTITY**



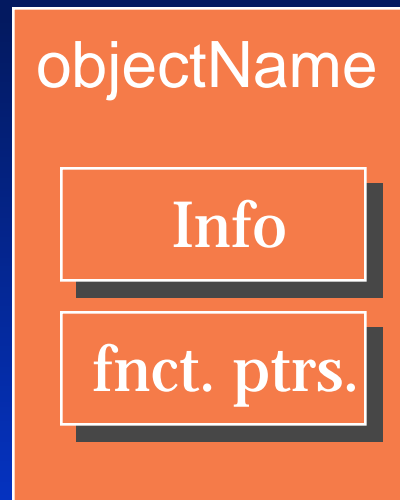
# The BCERT Model -- Theme 2



# Definition

An **Object** is the “vessel” that carries the information necessary to perform the desired operations.

## Object



# Five Steps -- Theme 1

---

1. Create
2. Enter Information
3. Sign (not always necessary)
4. Get DER
5. Destroy



# Five Steps -- Theme 2

---

1. Create
2. Set BER
3. Get Information
4. Verify (not always necessary)
5. Destroy



# BER and DER

**Basic Encoding Rules -- converts ASN.1 definitions into string of octets, computer readable data.**

**However, there are multiple encodings for objects.**

**Distinguished Encoding Rules -- unique encoding for objects  
a subset of BER**

**BCERT reads BER (which means it also reads DER)  
if sender produces BER or DER, BCERT is compatible  
BCERT produces DER  
if recipient reads only DER, or reads BER,  
BCERT is compatible**



# Theme 1: Producing ID Constructs

---

**Example:**

**The Name Object**



# Step 1: Create

**C\_CreateNameObject (&myName);**

**Object**

objectName: myName

DER:

AVA's:



# Step 2: Enter Information

## C\_AddNameAVA

### Object

objectName: myName

DER:

AVA's:

AT\_COUNTRY:  
"US"

AT\_ORGANIZATION:  
"Martin Bishop and  
Associates"

AT\_COMMON\_NAME:  
"Darren Roscow"



# Step 3: Sign

---

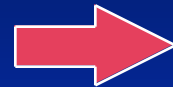
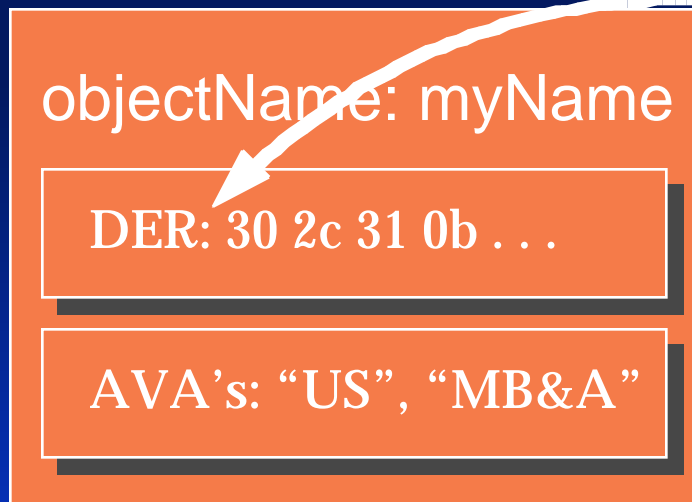
**No need to sign a name object.**



# Step 4: Get DER

**C\_GetNameDER**  
(myName, &nameDER, &nameDERLen)

**Object**



**Pointer**



# Save The DER

**Pointer**



30 2c 31 0b 30 09 06 03  
55 04 06 13 02 55 53 31  
0f 30 0d 06 03 55 04 03  
13 06 55 73 ...

```
fwrite (nameDER, 1,  
        (size_t)nameDERLen, fileHandle);
```



# Step 5: Destroy

```
C_DestroyNameObject (&myName);
```

Object



# Theme 2: Reading ID Constructs

---

**Example:**

**The Name Object**



# Step 1: Create

**C\_CreateNameObject (&newName);**

**Object**

objectName: myName

DER:

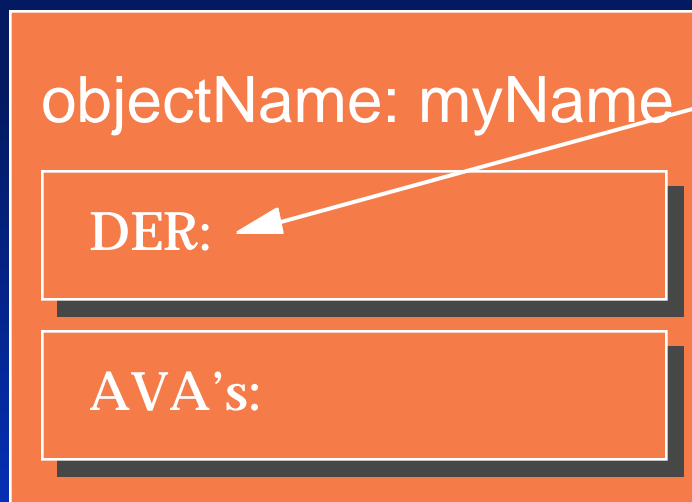
AVA's:



# Step 2: Set BER

## C\_SetNameBER

### Object



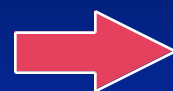
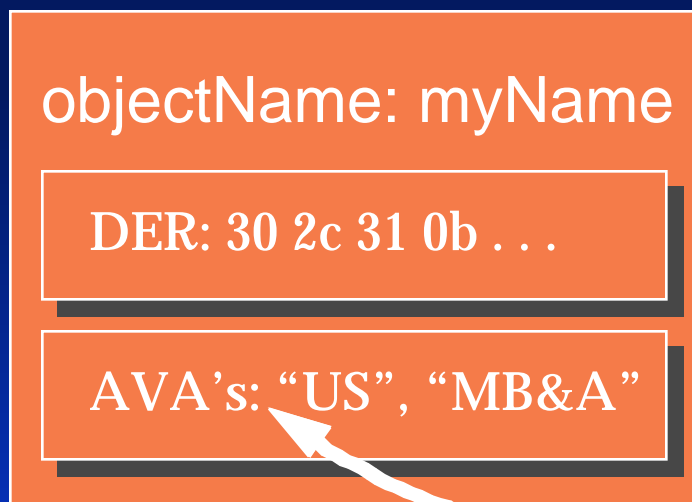
30 2c 31 0b 30 09 06 03  
55 04 06 13 02 55 53 31  
0f 30 0d 06 03 55 04 03  
13 06 55 73 ...



# Step 3: Get Information

**C\_GetNameAVACount**  
**C\_GetNameAVA**

**Object**



**Pointers**



# The BCERT Model -- Part 2

**Pointer**



value = "Darren Roscow"

**Pointer**



value = "Martin Bishop and  
Associates"

**Pointer**



value = "Sneaker"



# Step 4: Verify

---

**No signature to verify.**



# Step 5: Destroy

**C\_DestroyNameObject (&newName);**

**Object**



# Theme 1: Producing ID Constructs

---

**Example:**

**The Certificate Request Object**



# Preliminaries

```
B_KEY_OBJ publicKey = (B_KEY_OBJ)NULL_PTR;  
B_KEY_OBJ privateKey = (B_KEY_OBJ)NULL_PTR;  
  
if ((status = B_GenerateKeypair  
    (keypairGenerator, publicKey, privateKey,  
    random, &surrenderCtx)) != 0)  
    break;
```



# B\_GetKeyInfo

```
ITEM *publicKeyBER;  
  
if ((status = B_GetKeyInfo  
    ((POINTER *)&publicKeyBER, publicKeyObj,  
    KI_RSAPublicBER)) != 0)  
    break;
```



# Step 1: Create

**C\_CreateCertRequestObject (&myRequest);**

## Object

objectName: myRequest

DER:

- Inner DER:
- signature: (nothing yet)

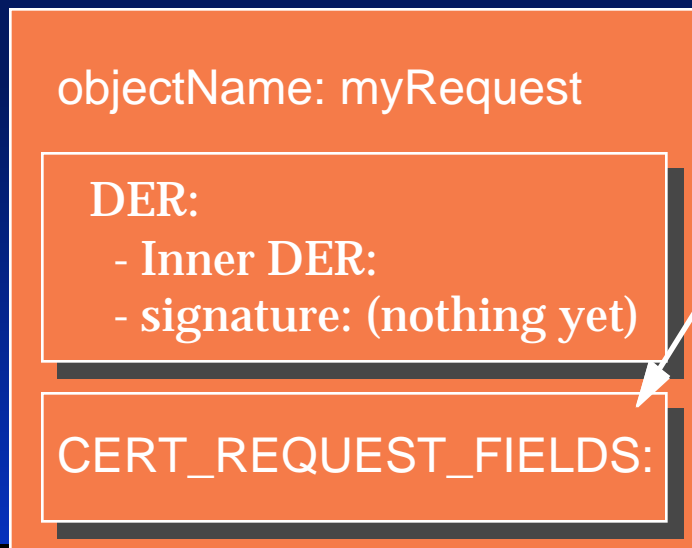
CERT\_REQUEST\_FIELDS:



# Step 2: Enter Information

## C\_SetCertRequestFields

### Object



```
typedef struct {  
    UINT2 version;  
    NAME_OBJ  
        subjectName;  
    ITEM publicKey;  
    ATTRIBUTES_OBJ  
        attribute;  
    POINTER reserved;  
} CERT_REQUEST_FIELDS;
```



# CERT\_REQUEST\_FIELDS

```
CERT_REQUEST_FIELDS certRequestFields;  
  
certRequestFields.version = CERT_VERSION_1;  
certRequestFields.subjectName = myName;  
certRequestFields.publicKey = *publicKeyBER;  
certRequestFields.attribute = NULL_PTR;  
certRequestFields.reserved = NULL_PTR;
```



# C\_SetCertRequestFields

```
if ((status = C_SetCertRequestFields  
    (myCertRequest, &certRequestFields)) != 0)  
    break;
```



# Step 2: Enter Information

## C\_SetCertRequestFields

### Object

objectName: myRequest

DER:

- Inner DER: 30 82 01 . . .
- signature: (nothing yet)

CERT\_REQUEST\_FIELDS:

← **version, name, key**



# Step 3: Sign

## C\_SignCertRequest

### Object

objectName: myRequest

DER: 30 82 01 ...

- Inner DER: 30 82 01 ...

- signature: 03 61 00 ...

CERT\_REQUEST\_FIELDS:

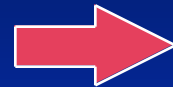
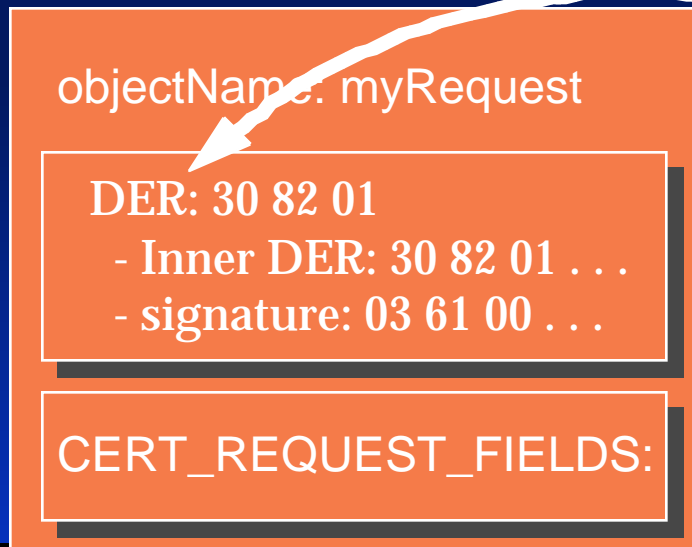
03 61 00 9e f8 3c 84 74  
42 65 56 b8 0c 08 4c 09  
4e 0d df f3 ee 79 29 75  
eb 3e d8 e2 ...



# Step 4: Get DER

**C\_GetCertRequestDER**  
(myRequest, &requestDER, &requestDERLen)

**Object**



**Pointer**



# Save The DER

**Pointer**



30 82 01 48 30 81 d3 02  
01 00 30 2c 31 0b 30 09  
06 03 55 04 06 13 02 55  
53 31 0f 30 ...

(including signature)

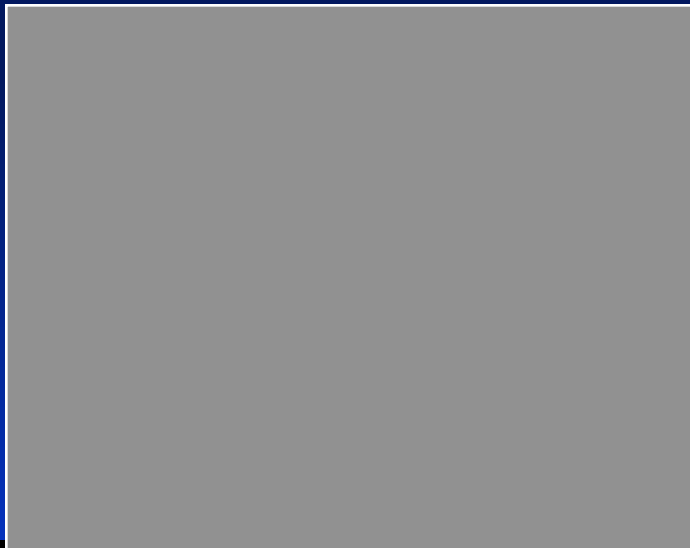
```
fwrite (requestDER, 1,  
        (size_t)requestDERLen, fileHandle);
```



# Step 5: Destroy

```
C_DestroyCertRequestObject (&myRequest);
```

Object



# Theme 2: Reading ID Constructs

---

**Example:**

**The Certificate Request Object**



# Five Steps

---

1. **C\_CreateCertRequestObject**
2. **C\_SetCertRequestBER**
3. **C\_GetCertRequestFields**
4. **C\_VerifyCertRequestSignature**
5. **C\_DestroyCertRequestObject**



# Step 1: Create

```
C_CreateCertObject (&newCert, NULL_PTR);  
C_CreateAttributesObject (&newAttr);
```

## Object

objectName: newCert

DER:

- Inner DER:
- signature: (nothing yet)

CERT\_FIELDS:

## Object

objectName: newAttr

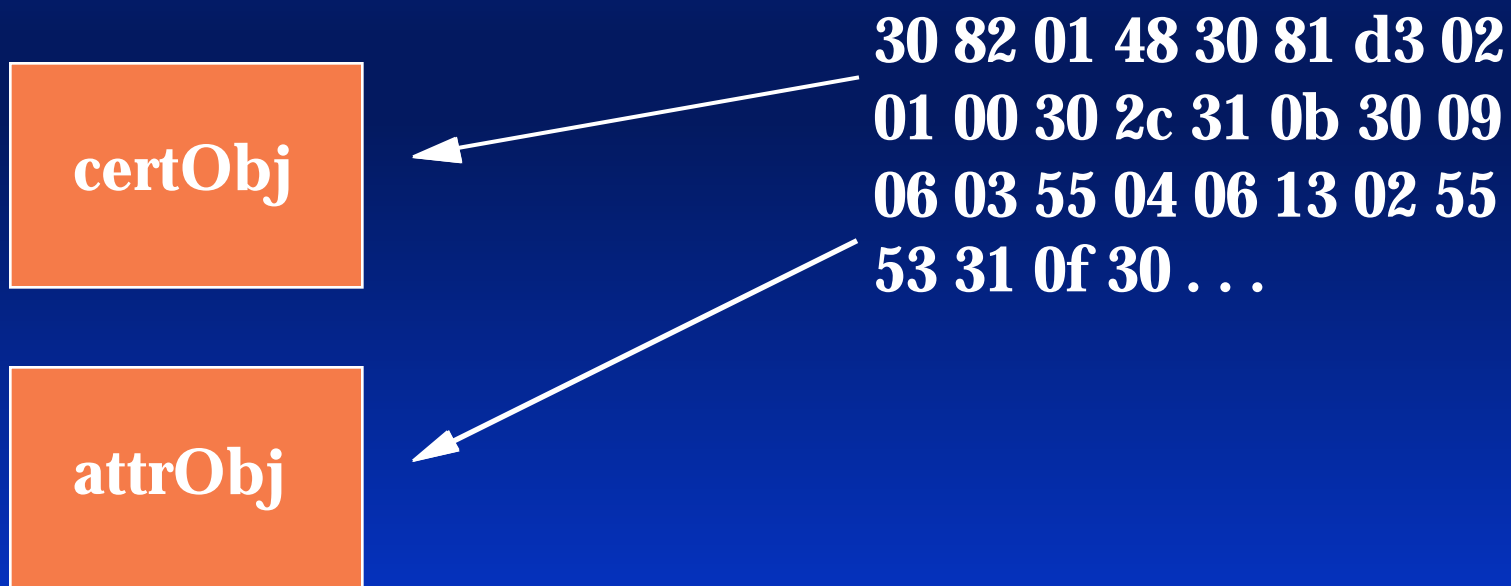
DER:

types:



# Step 2: Set BER

## C\_DecomposePKCSRequestBER



# Step 2: Set BER

## The certificate object

### Object

objectName: newCert

DER:

- Inner DER:
- signature: (nothing yet)

CERT\_FIELDS:

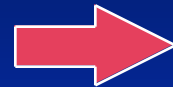
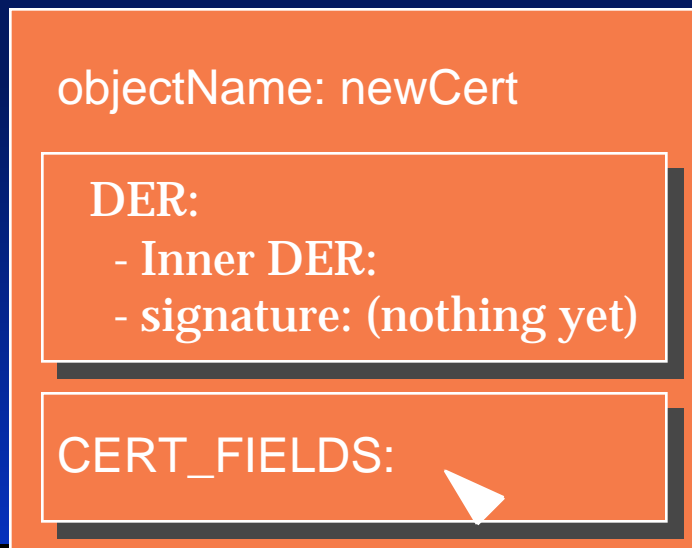
30 82 01 d1 30 82 01 57  
a0 03 02 01 02 02 04 01  
00 00 00 30 0d 06 09 2a  
86 48 86 f7 ...



# Step 3: Get Information

## C\_GetCertFields

### Object



**Pointers**



# CERT\_FIELDS

```
typedef struct {
    UINT2 version;
    ITEM serialNumber;
    int signatureAlgorithm;
    NAME_OBJ issuerName;
    struct {
        UINT4 start;
        UINT4 end;
    } validity;
    NAME_OBJ subjectName;
    ITEM publicKey;
    BIT_STRING issuerUniqueID;
    BIT_STRING subjectUniqueID;
    EXTENSIONS_OBJ certExtensions;
    POINTER reserved;
} CERT_FIELDS;
```



# CERT\_FIELDS

```
CERT_FIELDS certFields;  
  
if ((status = C_GetCertFields  
    (newCert, &certFields)) != 0)  
    break;
```



# CERT\_FIELDS

<b>version</b>	<b>CERT_</b> <b>VERSION_1</b>		<b>subjectName</b>	<b>requester's</b> <b>name</b>
<b>serialNum</b>	{NULL, 0}		<b>publicKey</b>	<b>DER-encoding</b> <b>from request</b>
<b>signAlg</b>	<b>MD5 w/ RSA</b>		<b>issuer</b> <b>UniqueID</b>	{NULL, 0}
<b>issuerName</b>	<b>requester's</b> <b>name</b>		<b>subject</b> <b>UniqueID</b>	{NULL, 0}
<b>validity.</b> <b>start</b>	<b>time of call to</b> <b>Decompose</b>		<b>extensions</b>	<b>created but</b> <b>empty object</b>
<b>validity.</b> <b>end</b>	<b>one year</b> <b>later</b>		<b>reserved</b>	<b>reserved for</b> <b>BCERT</b>



# Step 4: Verify

---

**C\_DecomposePKCS CertRequestBER  
verified the signature already.**



# Step 5: Destroy

**Call C\_DestroyCertObject and C\_DestroyAttributesObject to destroy the objects you created. You do not have to destroy the objects created by BCERT. They are the name objects and extensions object in the CERT\_FIELDS struct.**



# Or Delay Step 5

**We are building a certificate from a certificate request. We used the certificate request to produce a “first draft” of an object that will eventually become a certificate. Do not destroy that draft yet, edit it.**



# Theme 1: Producing ID Constructs

---

**Example:**

**The Certificate Object**



# Step 1: Create

---

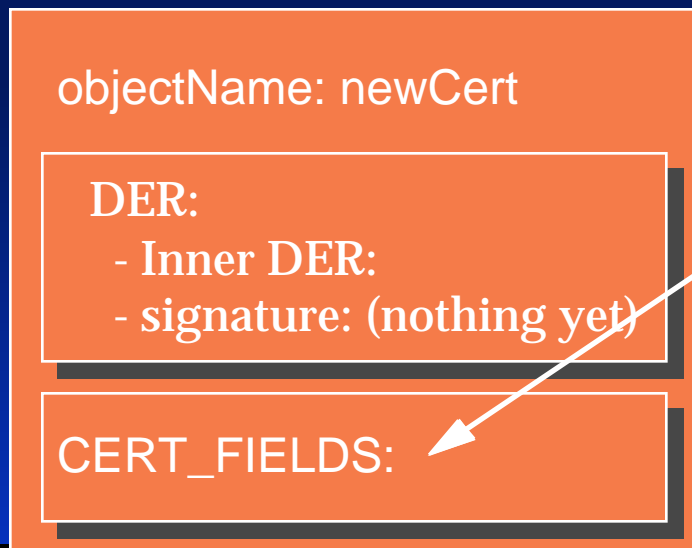
**We already created a certificate object in order to decompose the cert request.**



# Step 2: Enter Information

## C\_SetCertFields

### Object



**your CERT\_FIELDS struct, an “edited” version based on the certFields produced earlier.**



# CERT\_FIELDS

<b>version</b>	<b>CERT_</b> <b>VERSION_1</b>		<b>subjectName</b>	<b>requestor's</b> <b>name</b>
<b>serialNum</b>	{NULL, 0}		<b>publicKey</b>	<b>DER-encoding</b> <b>from request</b>
<b>signAlg</b>	<b>MD5 w/ RSA</b>		<b>issuer</b> <b>UniqueID</b>	{NULL, 0}
<b>issuerName</b>	<b>requestor's</b> <b>name</b>		<b>subject</b> <b>UniqueID</b>	{NULL, 0}
<b>validity.</b> <b>start</b>	<b>time of call to</b> <b>Decompose</b>		<b>extensions</b>	<b>created but</b> <b>empty object</b>
<b>validity.</b> <b>end</b>	<b>one year</b> <b>later</b>		<b>reserved</b>	<b>reserved for</b> <b>BCERT</b>



# CERT\_FIELDS

```
unsigned int nameDERLen;  
unsigned char *nameDER;  
unsigned char newNum[4] = {1 2 3 4};  
  
certFields.serialNumber.data = newNum;  
certFields.serialNumber.len = 4;  
  
if ((status = GetMyName  
    (&nameDER, &nameDERLen)) != 0)  
    break;  
  
if ((status = C_SetNameBER  
    (certFields.issuerName, nameDER,  
    nameDERLen)) != 0)  
    break;
```



# CERT\_FIELDS

```
if ((status = C_SetCertFields  
    (newCert, &certFields)) != 0)  
    break;
```



# Setting An Object

objectName: myObj

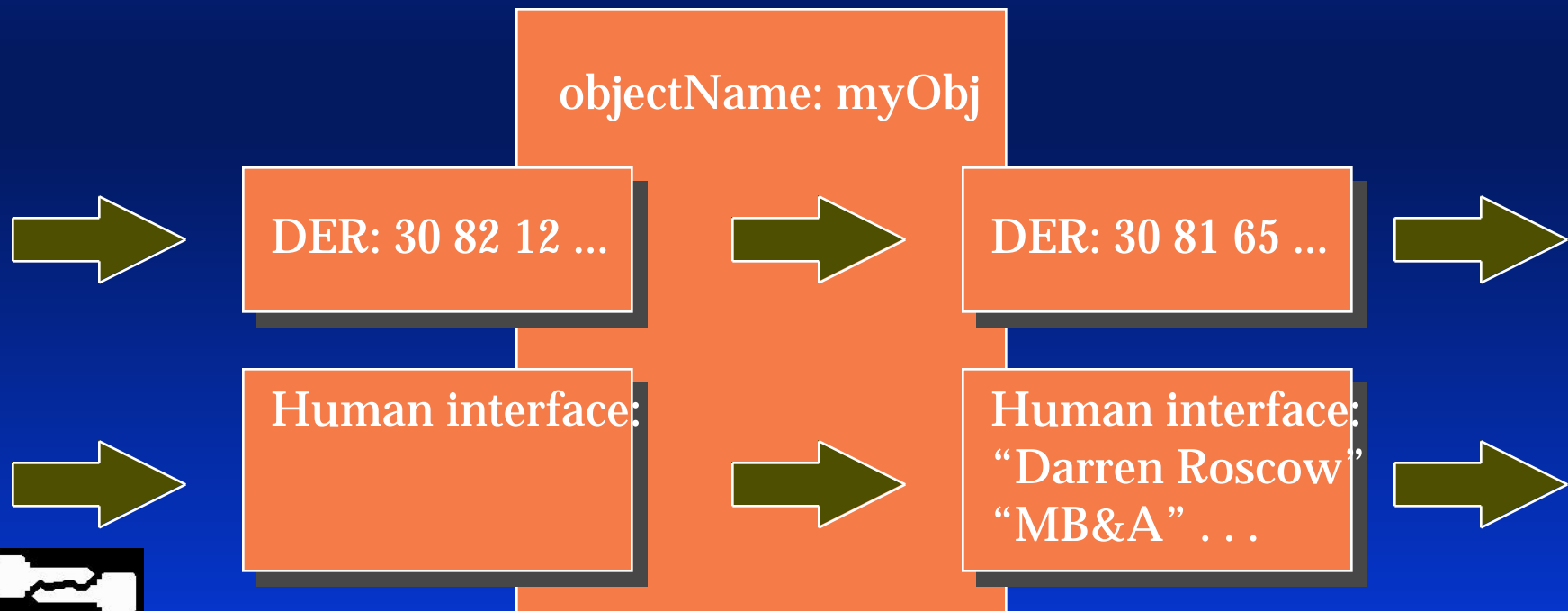
DER: 30 81 65 ...

Human interface:  
"Darren Roscow"  
"MB&A" ...



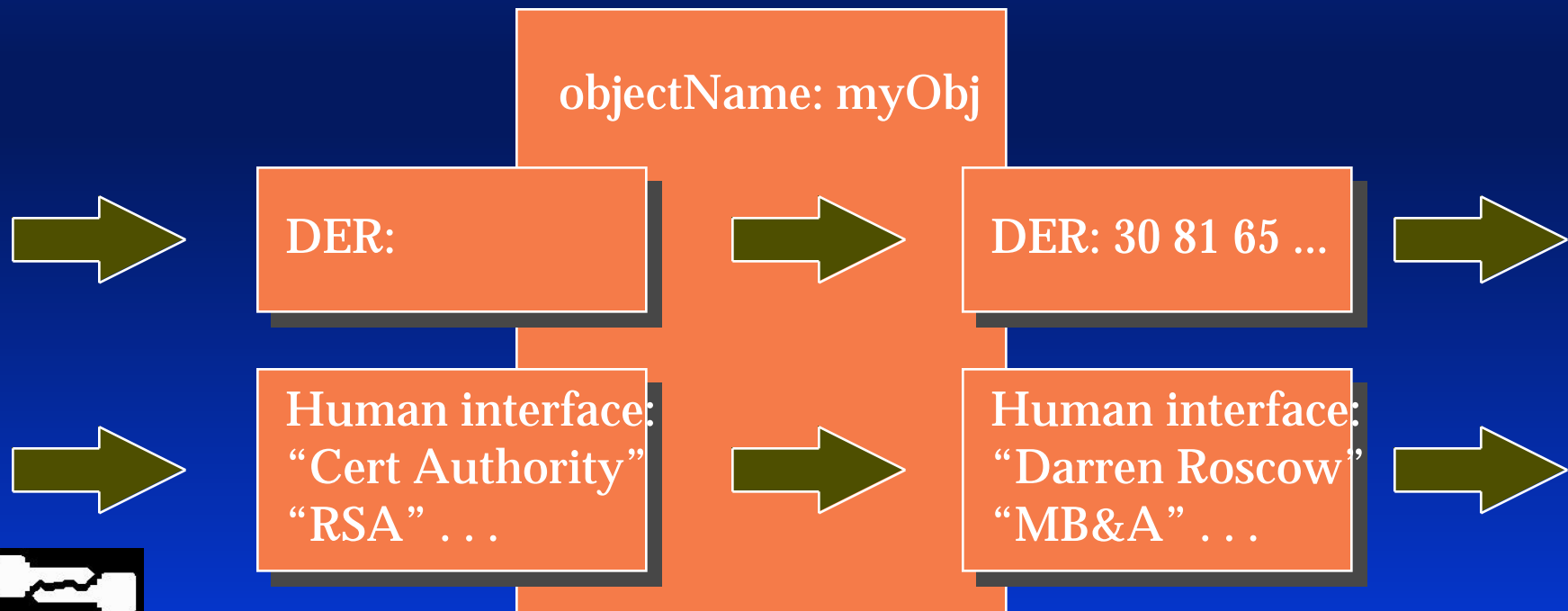
# Setting An Object

C\_Set\*BER



# Setting An Object

C\_Set\*Fields



# Setting An Object

objectName: myObj

DER: new DER

Human interface:  
new information



# Step 3: Sign

## C\_SignCert

### Object

objectName: newCert

DER: 30 82 01 ...

- Inner DER: 30 82 01 ...

- signature: 03 65 00 ...

CERT\_FIELDS: (updated)

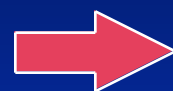
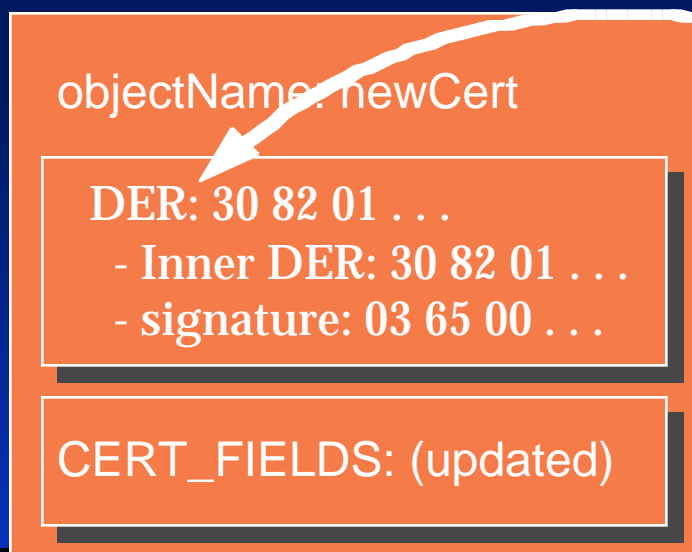
03 65 00 d9 43 0b d7 b3  
fd ba 1c e3 ad 41 8b 30  
7d 88 95 15 95 4d 8a de  
4e 02 bd 12 ...



# Step 4: Get DER

**C\_GetCertDER**  
(newCert, &certDER, &certDERLen)

**Object**



**Pointer**



# Save The DER

**Pointer**



30 82 01 e7 30 82 01 60  
a0 03 02 01 02 02 04 01  
00 00 00 30 0d 06 09 2a  
86 48 86 f7 ...

(including signature)

```
fwrite (certDER, 1,  
        (size_t)certDERLen, fileHandle);
```



# Step 5: Destroy

```
C_DestroyCertObject (&newCert);
```

Object



# Other ID Constructs

- Certificate Revocation List (CRL)
- CRL Entry
- Attributes
- X.509 V3 Certificate Extensions
- User-Defined Extensions



# Five Steps: Theme 1

---

1. **C\_CreateCRLObject**
2. **C\_SetCRLFields**  
    - **CRLentries**  
    (e.g. **C\_AddCRLentry**)
3. **C\_SignCRL**
4. **C\_GetCRLDER**
5. **C\_DestroyCRLObject**



# Five Steps: Theme 2

1. **C\_CreateCRLObject**
2. **C\_SetCRLBER**
3. **C\_GetCRLFields**
  - **CRLentries**  
(e.g. **C\_GetCRLentry**)
4. **C\_VerifyCRLSignature**
5. **C\_DestroyCRLObject**



# List Object

---

- General Utility Container
- User-Defined Handler
- Create and Destroy
- Add, Get, Insert and Delete Entries



# Using BCERT 1.0

---

## X.509 v3 Certificate Extensions



# Version 1 Digital Certificate

Version Number (0)	Signature Algorithm
Subject Name	Public Key
Issuer Name	Serial Number
Validity Start	Validity End

Issuer's Signature



# Version 2 Digital Certificate

Version Number (1)	Signature Algorithm
Subject Name	Public Key
Issuer Name	Serial Number
Validity Start	Validity End
<i>Subject Unique ID</i>	<i>Issuer Unique ID</i>

Issuer's Signature



# Version 3 Digital Certificate

Version Number (2)	Signature Algorithm
Subject Name	Public Key
Issuer Name	Serial Number
Validity Start	Validity End
<i>Subject Unique ID</i>	<i>Issuer Unique ID</i>
	<i>Extensions</i>
Issuer's Signature	



# Standard x.509 v3 Extensions

## Certificate Extensions

**Authority Key Identifier**  
**Subject Key Identifier**  
**Key Usage**  
**Private Key Usage Period**  
**Certificate Policies**  
**Subject Alternate Name**  
**Issuer Alternate Name**  
**Basic Constraints**  
**Policy Constraints**  
**Subject DirectoryAttributes**



# Standard x.509 v3 Extensions

## CRL Extensions

---

**CRL Number**

**Delta CRL**

**Authority Key Identifier**



# Standard x.509 v3 Extensions

## CRL Entry Extensions

---

**Reason Code**

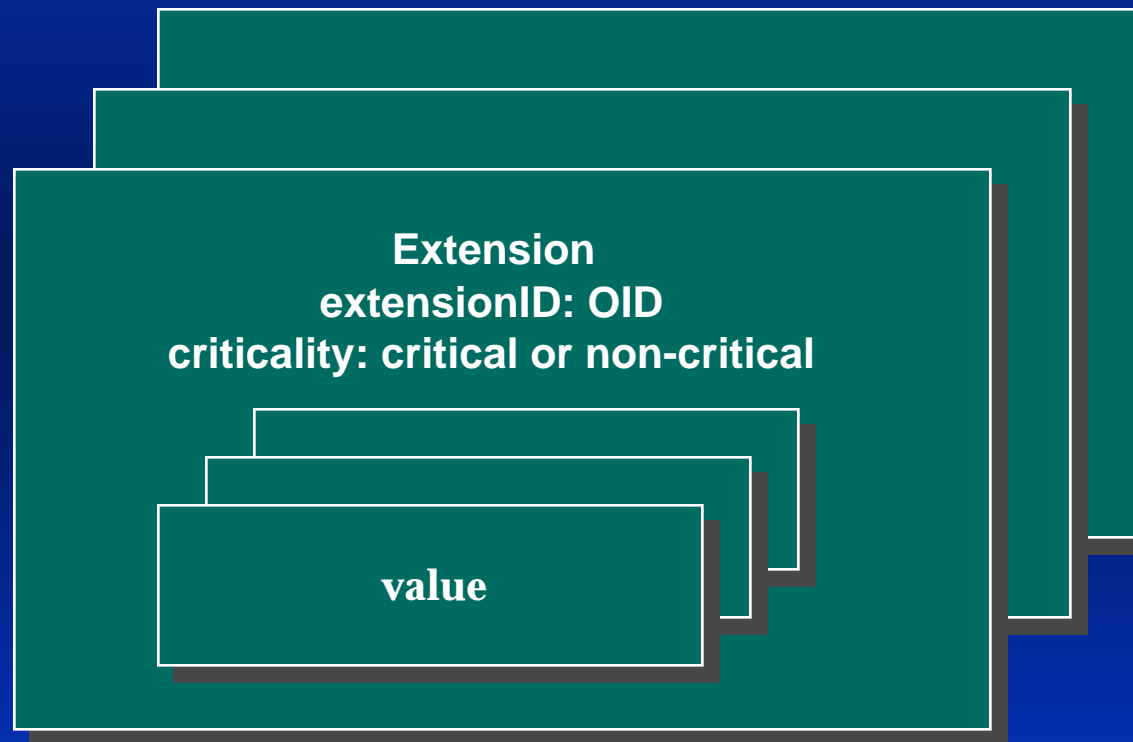
**Hold Instruction Code**

**Invalidity Date**



# Standard x.509 v3 Certificate Extensions

## Extensions



# Certificate Extensions

## Basic Constraints

**Describes the subject type the certificate holder is, CA or end entity, and if CA, the length of the certificate path to an end entity below it.**

**basicConstraints**  
**extensionID: OID = 0x55 0x1D 0x13**  
**criticality: critical**

**value**  
**subjectType**  
**pathLenConstraint**



# Theme 1: Producing ID Constructs

---

**Example:**

**The Extensions Object**



# Step 1: Create

**C\_CreateExtensionsObject**  
**(&myExtensions, CERT\_EXTENSIONS\_OBJ, NULL);**

**Object**

objectName: myExtensions

DER:

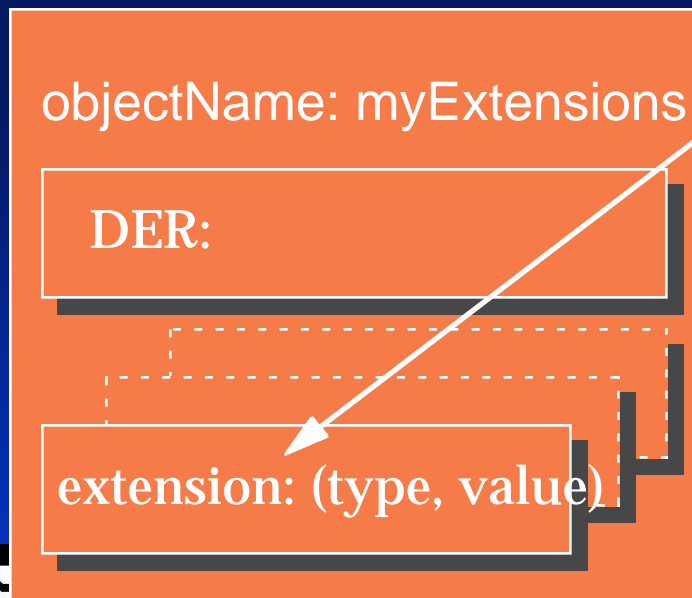
extension: (type, value)



# Step 2: Enter Information

**C\_CreateExtension**  
**C\_AddExtensionValue**

## Object



**type =**  
**ET\_BASIC\_CONSTRAINTS**

**value =**  
**(POINTER)&BASIC\_CONSTRAINTS**



# BASIC\_CONSTRAINTS struct

```
typedef struct BASIC_CONSTRAINTS {  
    unsigned int subjectType;  
    int pathLenConstraint;  
} BASIC_CONSTRAINTS;
```



# Extension Data Structures

authorityKeyID

```
typedef struct {  
    ITEM keyIdentifier;  
    ITEM serialNumber;  
    unsigned int issuerNameCount;  
    ALTERNATE_NAME *issuerNames;  
} AUTHORITY_KEY_ID;
```

keyUsage

UINT4

certificatePolicies

```
typedef struct {  
    ITEM policyID;  
    unsigned int qualifierInfoCount;  
    QualifierInfo *qualifierInfo;  
} POLICY_INFO;
```



# Step 2: Enter Information

```
BASIC_CONSTRAINTS basicConstraints;
unsigned int extIndex, valueIndex;

basicConstraints.subjectType = SUBJECT_TYPE_END_ENTITY;
basicConstraints.pathLenConstraint = NOT_IN_USE;

if ((status = C_CreateExtension
    (myExtensions, ET_BASIC_CONSTRAINTS,
     ET_BASIC_CONSTRAINTS_LEN, &extIndex, NOT_IN_USE,
     (EXTENSION_HANDLER *)NULL_PTR)) != 0)
    break;

if ((status = C_AddExtensionValue
    (myExtensions, extIndex, (POINTER)&basicConstraints,
     &valueIndex)) != 0)
    break;
```



# Step 3: Sign

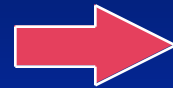
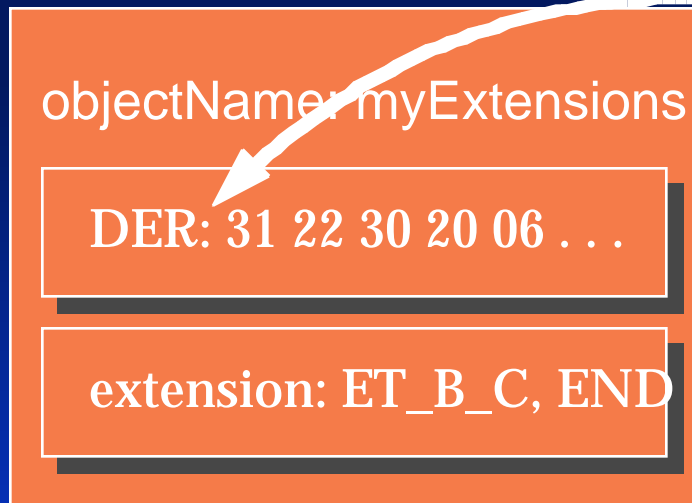
---

**No need to sign an extensions object.**



# Step 4: Get DER

**C\_GetExtensionsObjectDER**  
(myExtensions, &extensionsObjDER,  
&extensionsObjDERLen);  
**Object**



**Pointer**



# Save The DER

**Pointer**



31 22 30 20 06 09 2a 86  
48 86 f7 0d 01 09 0e 31  
13 14 11 30 ...

```
fwrite (extensionsObjDER, 1,  
        (size_t)extensionsObjDERLen, fileHandle);
```



# Step 5: Destroy

**C\_DestroyExtensionsObject (&myExtensions);**

**Object**



# Reading The Extensions Object

1. **C\_CreateExtensionsObject**
2. **C\_SetExtensionsObjectBER**
3. **C\_GetExtensionCount**  
**C\_GetExtensionInfo**  
**C\_GetExtensionValue**
4. **<no signature to verify>**
5. **C\_DestroyExtensionsObject**



# Extensions Object in a Certificate Object

```
if ((status = C_CreateCertObject  
    (&newCert, NULL)) != 0)  
    break;
```

```
if ((status = C_GetCertFields  
    (newCert, &certFields)) != 0)  
    break;
```

```
/* You now have a created but empty extensions  
   object: certFields.certExtensions */
```



# Extensions in Attributes Objects

```
CERT_REQUEST_FIELDS certRequestFields;  
  
certRequestFields.version = CERT_VERSION_1;  
certRequestFields.subjectName = myName;  
certRequestFields.publicKey = *publicKeyBER;  
certRequestFields.attribute = NULL_PTR;  
certRequestFields.reserved = NULL_PTR;
```



# Extensions in Attributes Objects

```
CERT_REQUEST_FIELDS certRequestFields;  
  
certRequestFields.version = CERT_VERSION_1;  
certRequestFields.subjectName = myName;  
certRequestFields.publicKey = *publicKeyBER;  
certRequestFields.attribute = NULL_PTR;  
certRequestFields.reserved = NULL_PTR;
```



# Extensions in Attributes Objects

```
if ((status = C_CreateExtensionsObject
    (&myExtensions, CERT_EXTENSIONS_OBJ, NULL)) != 0)
    break;

/* Enter all your extensions. */

if ((status = C_CreateAttributesObject
    (&myAttributes)) != 0)
    break;

if ((status = C_GetAttributeInExtensionsObj
    (myExtensions, myAttributes)) != 0)
    break;
```



# Extensions in Attributes Objects

```
CERT_REQUEST_FIELDS certRequestFields;  
  
certRequestFields.version = CERT_VERSION_1;  
certRequestFields.subjectName = myName;  
certRequestFields.publicKey = *publicKeyBER;  
certRequestFields.attribute = myAttributes;  
certRequestFields.reserved = NULL_PTR;
```



# Extensions in Attributes Objects

```
if ((status = C_DecomposePKCSCertRequestBER
    (certObj, attrObj, requestBER, requestBERLen,
    digestBuffer, &digestLen, surrender)) != 0)
    break;

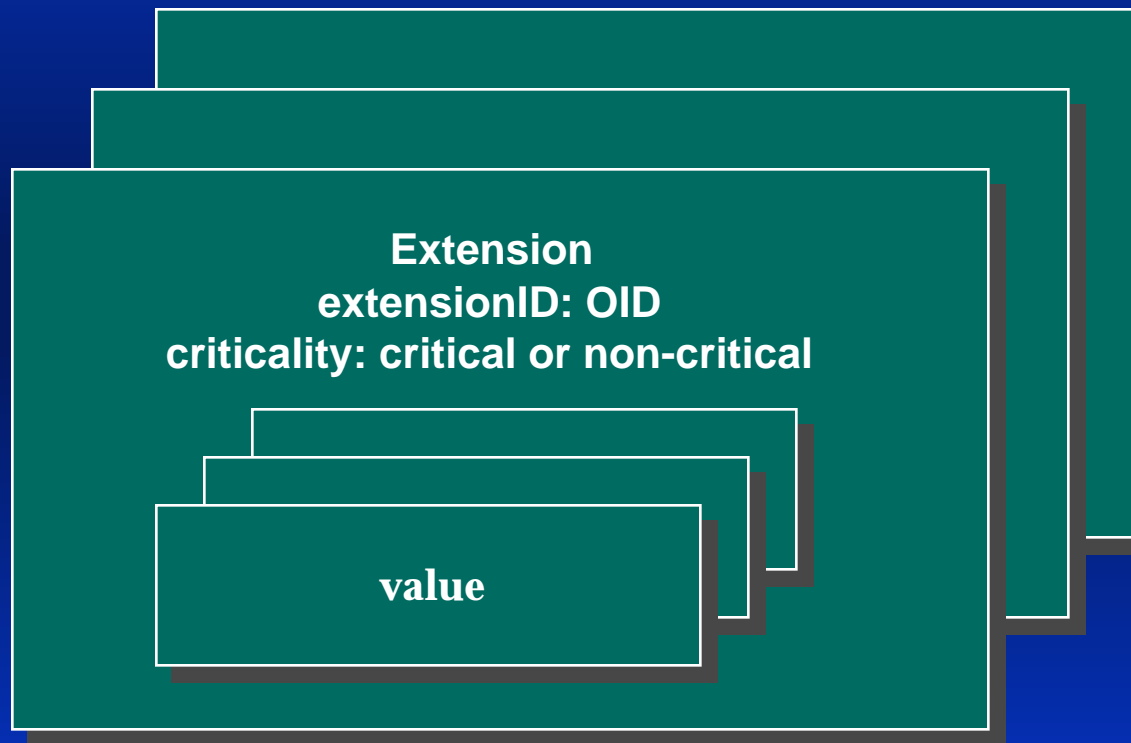
if ((status = C_GetCertFields
    (certObj, &certFields)) != 0)
    break;

if ((status = C_GetExtensionsInAttributesObj
    (certFields.certExtensions, attrObj)) != 0)
    break;
```



# User-Defined Extensions

## Extensions



# User-Defined Extension Employee Number

**employeeNumber**  
**extensionID: OID = "employee number"**  
**criticality: non-critical**

**value**  
**number**



# The Application Context

```
APPL_CTX myApplCtx = (APPL_CTX)NULL_PTR;  
EXTENSION_TYPE_INFO empNumTypeInfo;
```

```
if ((status = C_InitializeApplCtx  
      (&myApplCtx)) != 0)  
    break;
```

```
if ((status = C_RegisterExtensionType  
      (myApplCtx, &empNumTypeInfo)) != 0)  
    break;
```



# EXTENSION\_TYPE\_INFO

```
typedef struct {  
    ITEM type;  
    unsigned int criticality;  
    unsigned int overrideCriticality;  
    unsigned int overrideHandler;  
    UINT2 authenObjects;  
    unsigned int uniqueValue;  
    EXTENSION_HANDLER handler;  
} EXTENSION_TYPE_INFO;
```



# EXTENSION\_TYPE\_INFO

```
char *empNumType = "employee number";

empNumTypeInfo.type.data = empNumType;
empNumTypeInfo.type.len = T_strlen (empNumType);
empNumTypeInfo.criticality = NON_CRITICAL;
empNumTypeInfo.overrideCriticality = 0;
empNumTypeInfo.overrideHandler = 0;
empNumTypeInfo.authenObjects = CERT_EXTENSIONS_OBJ;
empNumTypeInfo.uniqueValue = 1;
empNumTypeInfo.handler = empNumHandler;

if ((status = C_RegisterExtensionType
      (myApplCtx, &empNumTypeInfo)) != 0)
    break;
```



# EXTENSION\_HANDLER

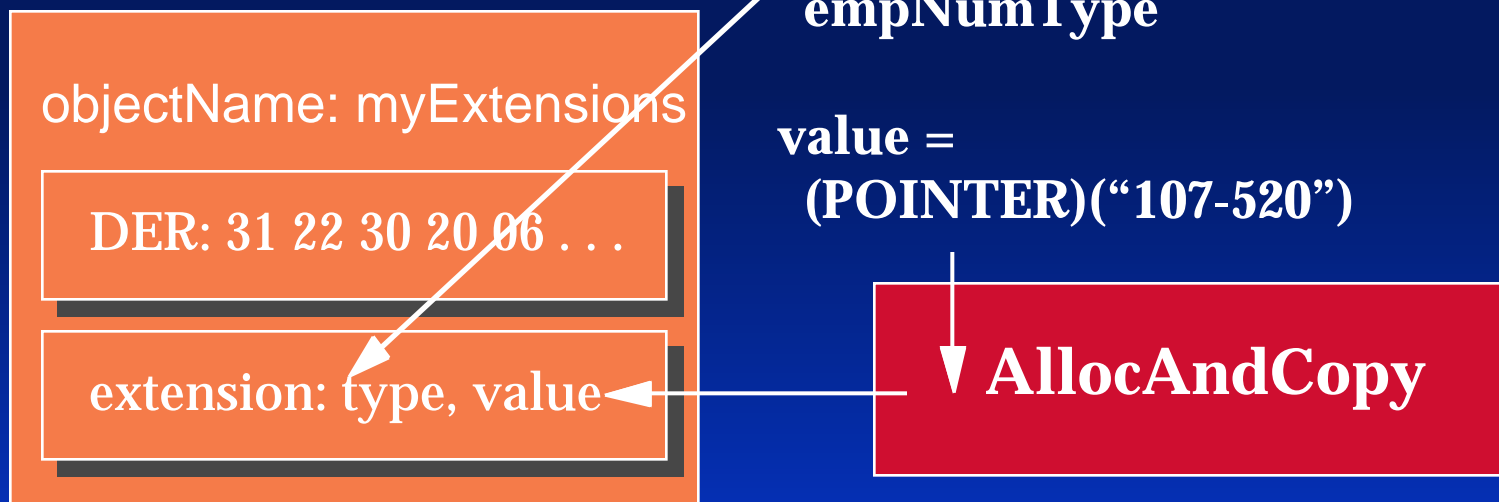
```
typedef struct {  
    int (*AllocAndCopy) (POINTER *, POINTER);  
    void (*Destructor) (POINTER);  
    int (*GetEncodedValue)  
        (LIST_OBJ, unsigned char **, unsigned int *);  
    int (*SetEncodedValue)  
        (LIST_OBJ, unsigned char *, unsigned int,  
        LIST_OBJ_ENTRY_HANDLER *);  
} EXTENSION_HANDLER;
```



# Step 2: Enter Information

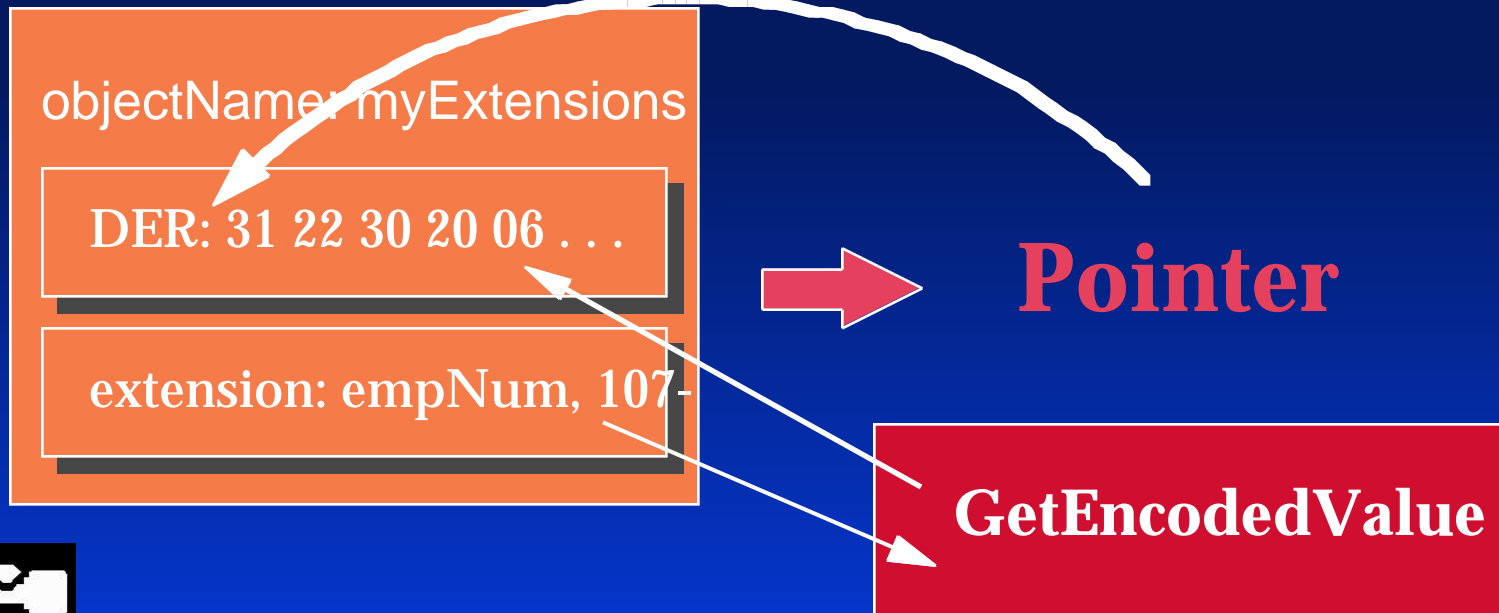
**C\_CreateExtension**  
**C\_AddExtensionValue**

## Object



# Step 4: Get DER

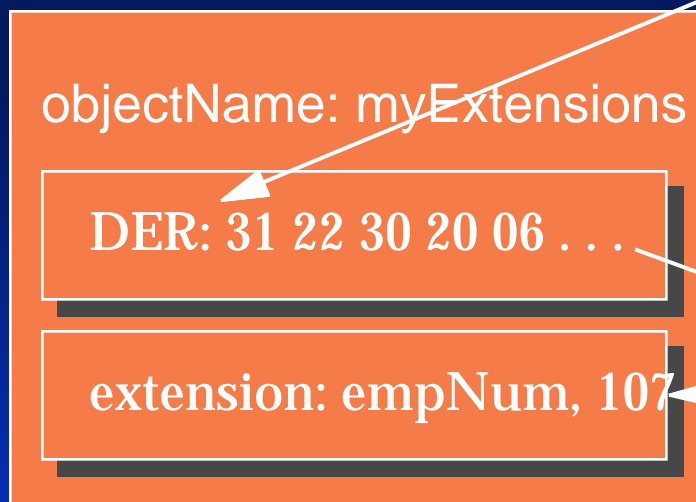
**C\_GetExtensionsObjectDER**  
(myExtensions, &extensionsObjDER,  
&extensionsObjDERLen);  
**Object**



# Step 2: Set BER

## C\_SetExtensionBER

### Object



31 22 30 20 06 09 2a 86  
48 86 f7 0d 01 09 0e 31  
13 14 11 30 ...

**SetEncodedValue**



# Step 5: Destroy

**C\_DestroyExtensionsObject (&myExtensions);**

**Object**



**Destructor**



# AllocAndCopy

```
int MyAllocAndCopy (POINTER *destination, POINTER source)
{
    ITEM *sourceItem, *destinationItem;

    *destination = T_malloc (sizeof (ITEM));
    sourceItem = (ITEM *)source;
    destinationItem = (ITEM *)(*destination);

    destinationItem->data = T_malloc (sourceItem->len);

    T_memcpy (destinationItem->data, sourceItem->data,
              sourceItem->len);
    destinationItem->len = sourceItem->len;

    return (0);
}
```



# Destructor

```
void MyDestructor (POINTER value)
{
    T_free (((ITEM *)value)->data);
    T_free (value);

    return;
}
```



# GetEncodedValue

```
int MyGetEncodedValue (LIST_OBJ valueList,
    unsigned char **encodedValue, unsigned int *encodedValueLen)
{
    int status;
    ITEM *value;

    if ((status = C_GetListObjectEntry
        (valueList, 0, (POINTER *)&value)) != 0)
        return (status);

    *encodedValue = T_malloc (value->len + 2);
    (*encodedValue)[0] = 0x04;
    (*encodedValue)[1] = (unsigned char)value->len;
    T_memcpy ((*encodedValue + 2), value->data, value->len);
    *encodedValueLen = value->len + 2;

    return (0);
}
```



# SetEncodedValue

```
int MySetEncodedValue (LIST_OBJ valueList,  
    unsigned char *encodedValue, unsigned int encodedValueLen,  
    LIST_OBJ_ENTRY_HANDLER *listHandler)  
{  
    int status;  
    ITEM value;  
  
    value.data = encodedValue + 2;  
    value.len = encodedValueLen - 2;  
  
    status = C_AddListObjectEntry  
        (valueList, (POINTER)&value, (unsigned int *)NULL_PTR,  
        listHandler);  
  
    return (status);  
}
```



# EXTENSION\_HANDLER

```
EXTENSION_HANDLER empNumHandler;
```

```
empNumHandler.AllocAndCopy = MyAllocAndCopy;  
empNumHandler.Destructor = MyDestructor;  
empNumHandler.GetEncodedValue = MyGetEncodedValue;  
empNumHandler.SetEncodedValue = MySetEncodedValue;
```



# Using the APPL\_CTX

```
if ((status = C_CreateCertObject
    (&newCert, myApplCtx)) != 0)
    break;

if ((status = C_CreateExtensionsObject
    (&myExtensions, CERT_EXTENSIONS_OBJ,
    myApplCtx)) != 0)
    break;
```



# Finalizing the APPL\_CTX

```
/* In object-oriented programming, what you create  
you must destroy. Similarly, when you  
Initialize an application context, you must  
Finalize it. Before the application ends,  
call the following routine. */
```

```
if ((status = C_FinalizeApplContext  
    (&myApplCtx)) != 0)  
    break;
```



# Using BCERT

Steve Burnett  
RSA Data Security Inc.  
burnetts@rsa.com

RSA Data Security Conference  
January 30, 1997

