z/TPF EE V1.1
z/TPFDF V1.1
TPF Toolkit for WebSphere® Studio V3
TPF Operations Server V1.2

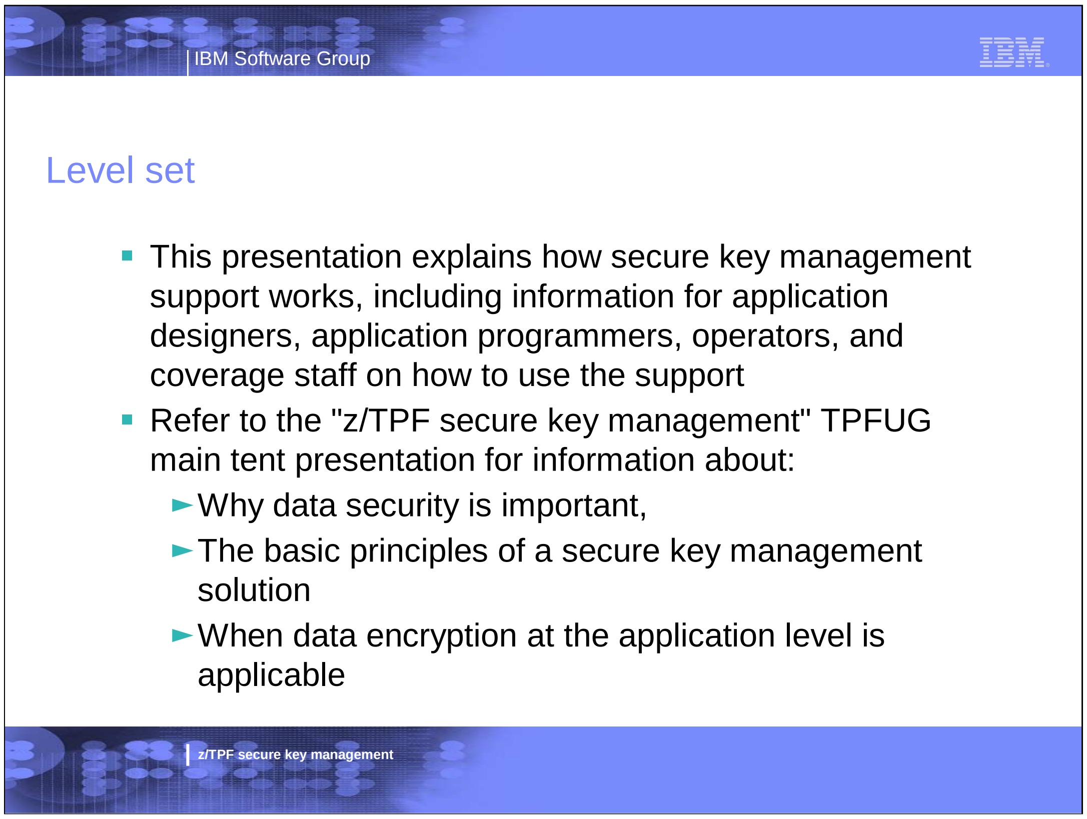
IBM Software Group

TPF Users Group Spring 2007

z/TPF secure key management

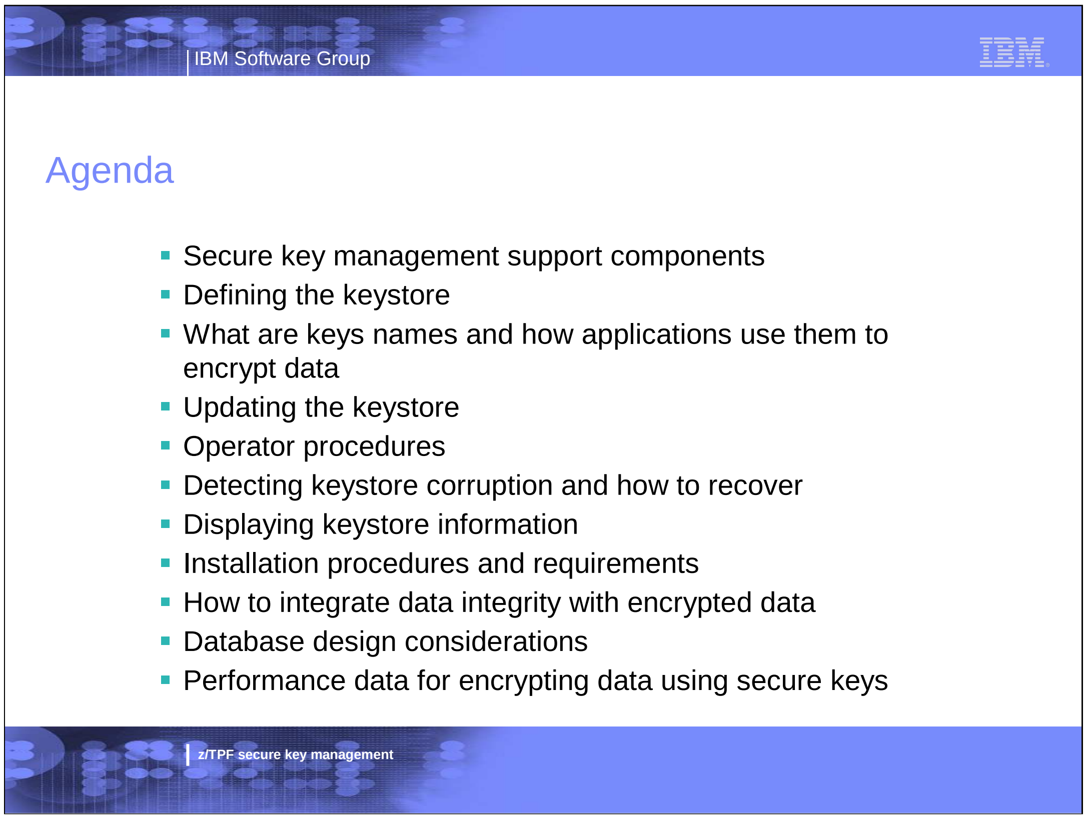
Subcommittee presentation, part 1

AIM Enterprise Platform Software
IBM z/Transaction Processing Facility Enterprise Edition 1.1.0



Level set

- This presentation explains how secure key management support works, including information for application designers, application programmers, operators, and coverage staff on how to use the support
- Refer to the "z/TPF secure key management" TPFUG main tent presentation for information about:
  - ►Why data security is important,
  - ►The basic principles of a secure key management solution
  - ►When data encryption at the application level is applicable



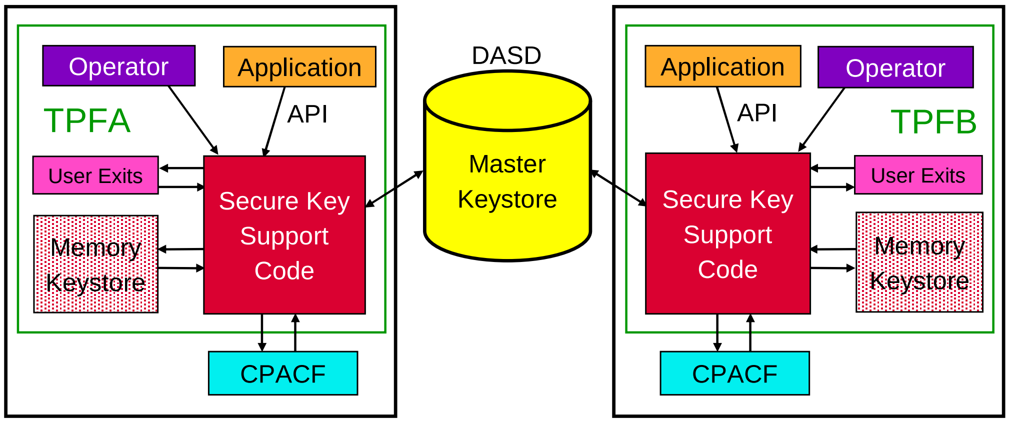
IBM Software Group

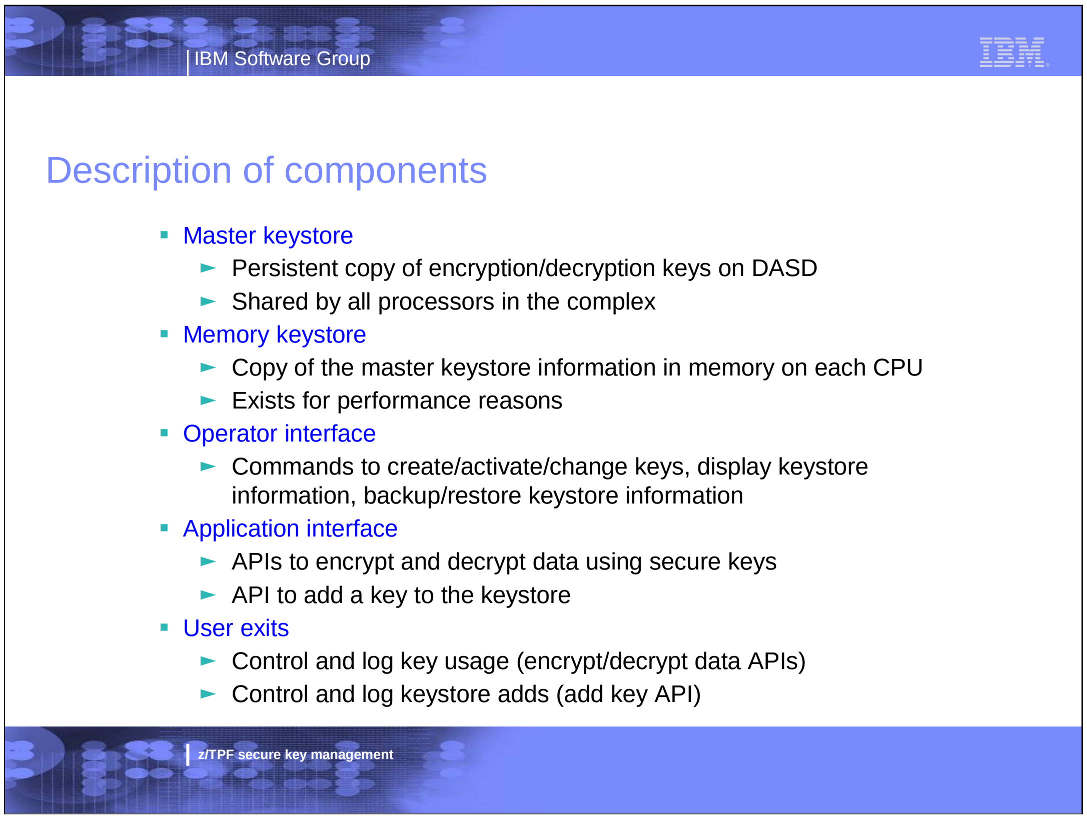
Agenda

- Secure key management support components
- Defining the keystore
- What are keys names and how applications use them to encrypt data
- Updating the keystore
- Operator procedures
- Detecting keystore corruption and how to recover
- Displaying keystore information
- Installation procedures and requirements
- How to integrate data integrity with encrypted data
- Database design considerations
- Performance data for encrypting data using secure keys

z/TPF secure key management

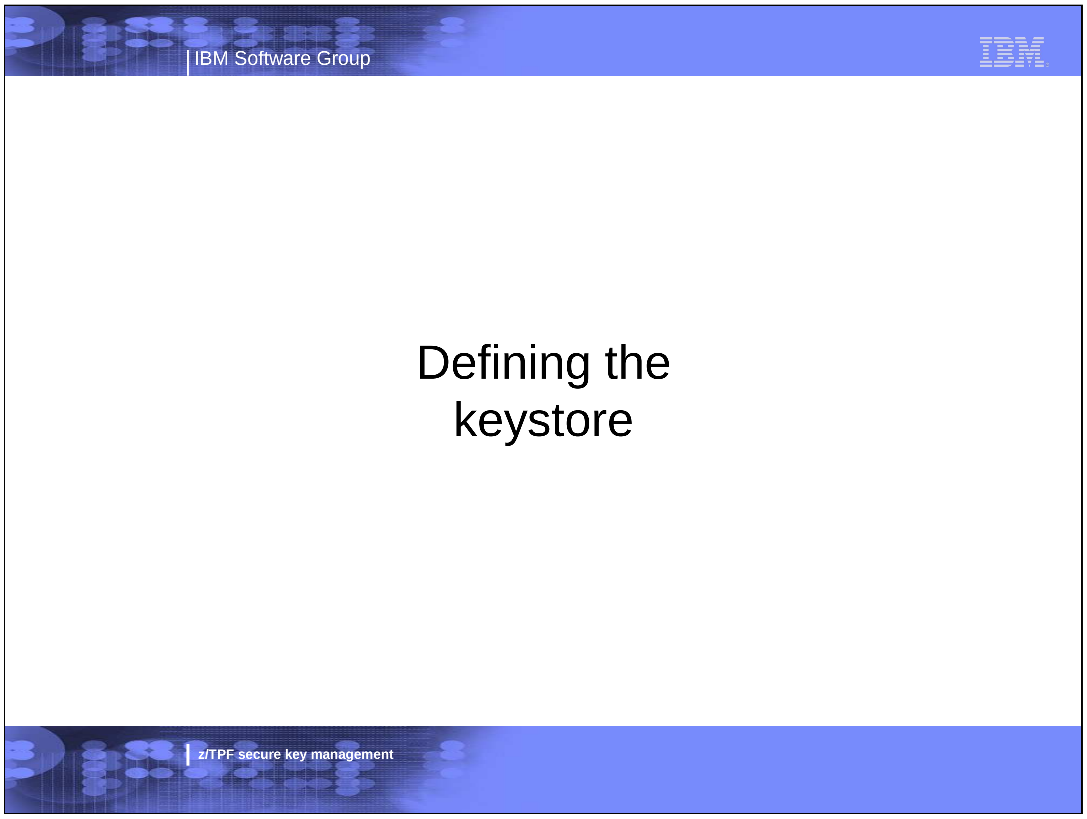
z/TPF secure key management components





Description of components

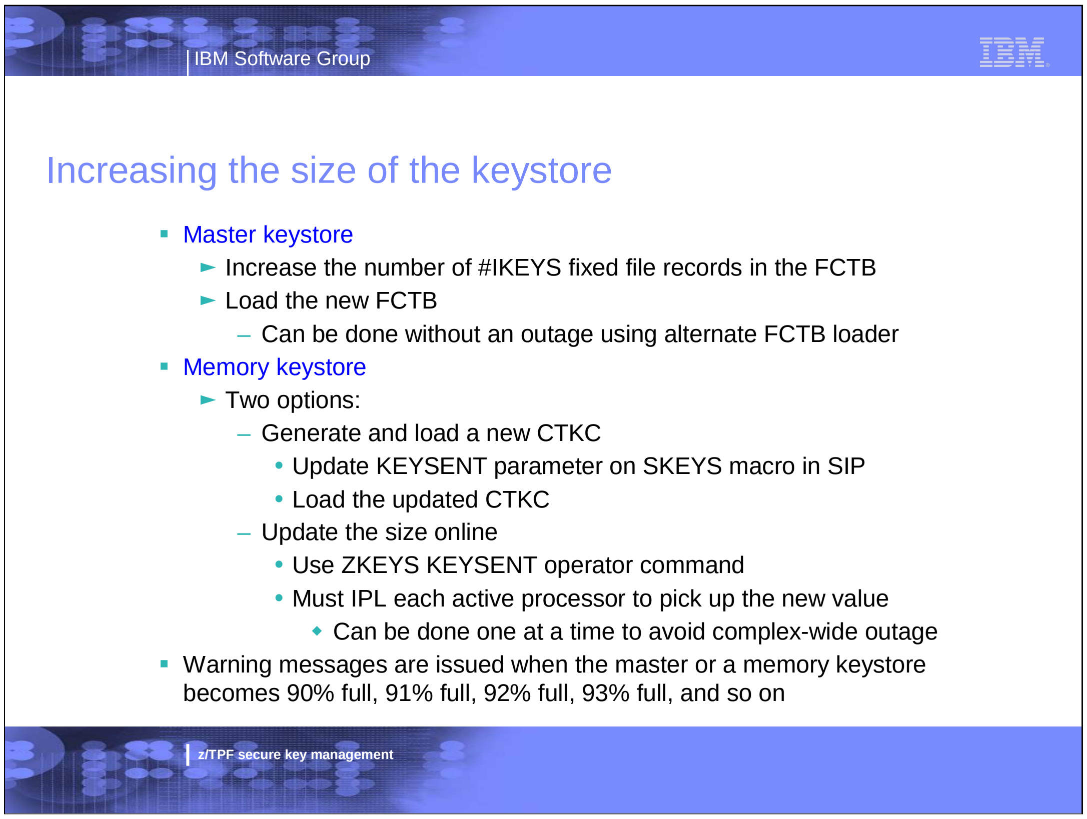
- Master keystore
  - Persistent copy of encryption/decryption keys on DASD
  - Shared by all processors in the complex
- Memory keystore
  - Copy of the master keystore information in memory on each CPU
  - Exists for performance reasons
- Operator interface
  - Commands to create/activate/change keys, display keystore information, backup/restore keystore information
- Application interface
  - APIs to encrypt and decrypt data using secure keys
  - API to add a key to the keystore
- User exits
  - Control and log key usage (encrypt/decrypt data APIs)
  - Control and log keystore adds (add key API)



Defining the keystore

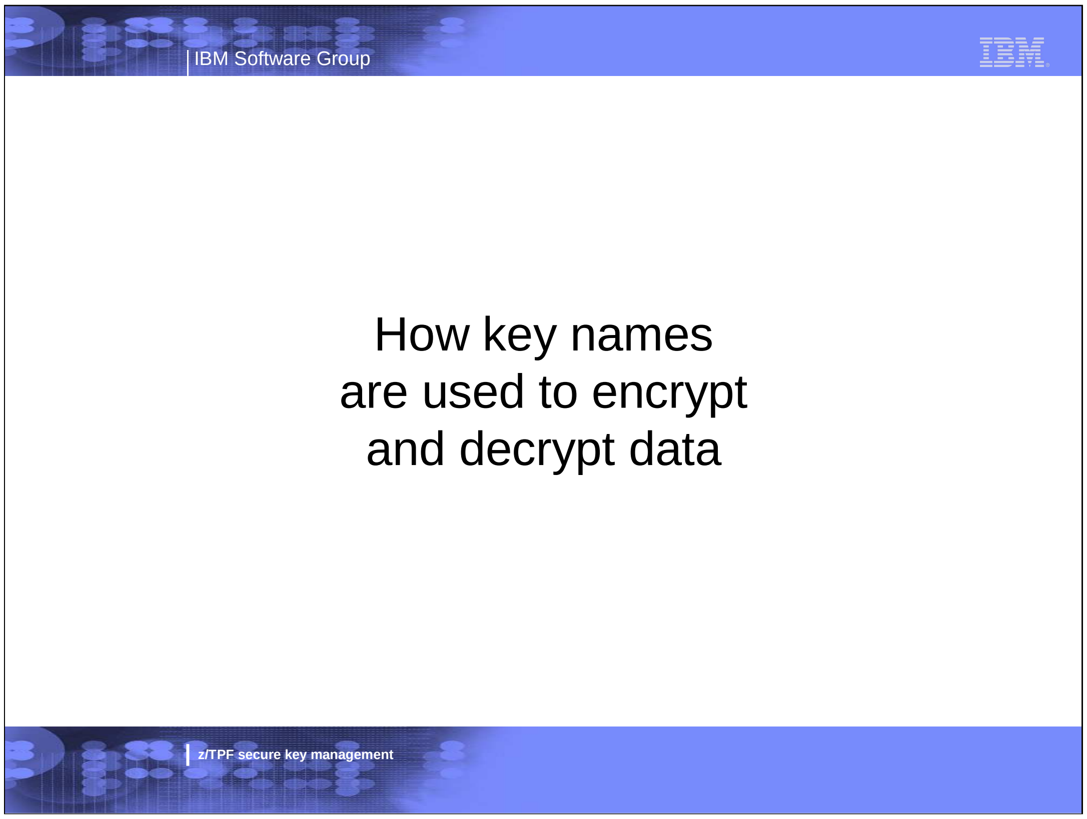
Defining keystore restores

- Determine how many secure keys you will need
- Master keystore
  - Define #IKEYS fixed file records in the FACE table (FCTB)
    - Formula for how many records are needed based on the number of keys is in the z/TPF documentation
  - Load the new FCTB
- Memory keystore
  - Define the size (number of entries) in Keypoint C (CTKC)
    - KEYSENT parameter on the SKEYS macro in SIP
    - CTKC is processor shared
- Defining the keystore enables secure key management support
- Sizes of the master keystore and memory keystore can be different
  - Memory keystore must be large enough to hold all the keys defined in the master keystore

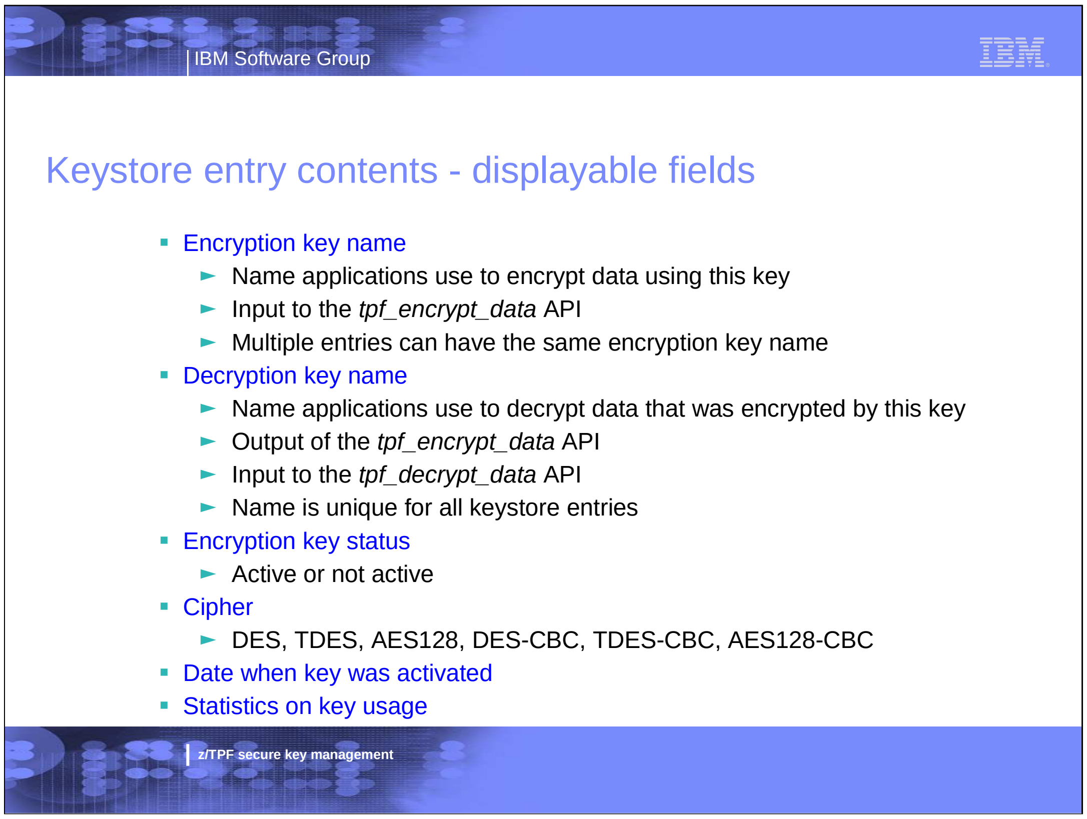


Increasing the size of the keystore

- Master keystore
  - Increase the number of #IKEYS fixed file records in the FCTB
  - Load the new FCTB
    - Can be done without an outage using alternate FCTB loader
- Memory keystore
  - Two options:
    - Generate and load a new CTKC
      - Update KEYSENT parameter on SKEYS macro in SIP
      - Load the updated CTKC
    - Update the size online
      - Use ZKEYS KEYSENT operator command
      - Must IPL each active processor to pick up the new value
        - Can be done one at a time to avoid complex-wide outage
- Warning messages are issued when the master or a memory keystore becomes 90% full, 91% full, 92% full, 93% full, and so on

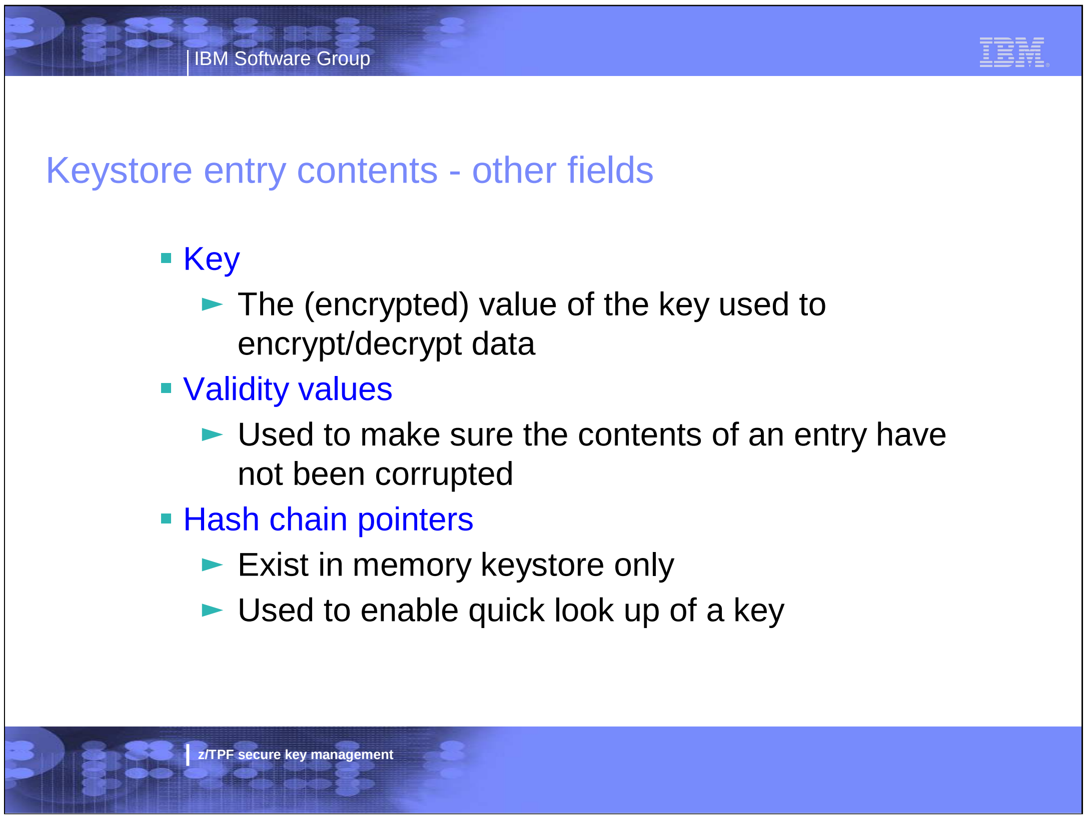


How key names are used to encrypt and decrypt data



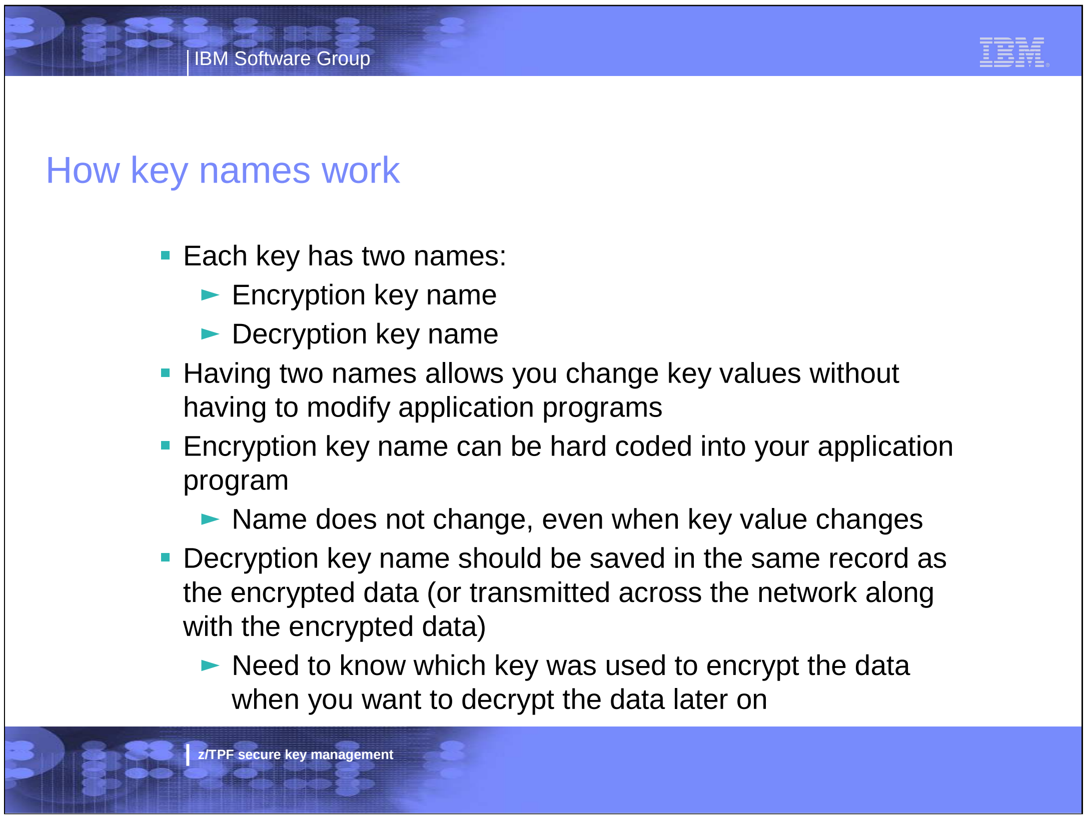
Keystore entry contents - displayable fields

- Encryption key name
  - Name applications use to encrypt data using this key
  - Input to the *tpf_encrypt_data* API
  - Multiple entries can have the same encryption key name
- Decryption key name
  - Name applications use to decrypt data that was encrypted by this key
  - Output of the *tpf_encrypt_data* API
  - Input to the *tpf_decrypt_data* API
  - Name is unique for all keystore entries
- Encryption key status
  - Active or not active
- Cipher
  - DES, TDES, AES128, DES-CBC, TDES-CBC, AES128-CBC
- Date when key was activated
- Statistics on key usage



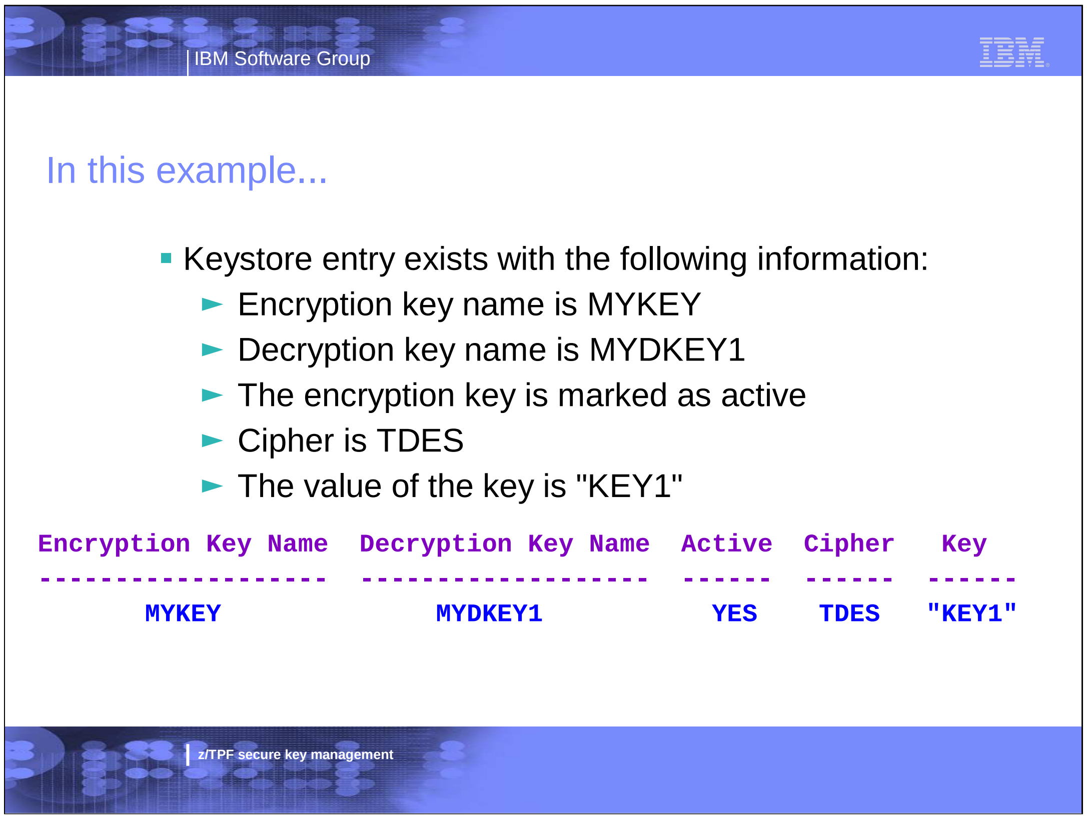
Keystore entry contents - other fields

- Key
  - The (encrypted) value of the key used to encrypt/decrypt data
- Validity values
  - Used to make sure the contents of an entry have not been corrupted
- Hash chain pointers
  - Exist in memory keystore only
  - Used to enable quick look up of a key



How key names work

- Each key has two names:
  - Encryption key name
  - Decryption key name
- Having two names allows you change key values without having to modify application programs
- Encryption key name can be hard coded into your application program
  - Name does not change, even when key value changes
- Decryption key name should be saved in the same record as the encrypted data (or transmitted across the network along with the encrypted data)
  - Need to know which key was used to encrypt the data when you want to decrypt the data later on

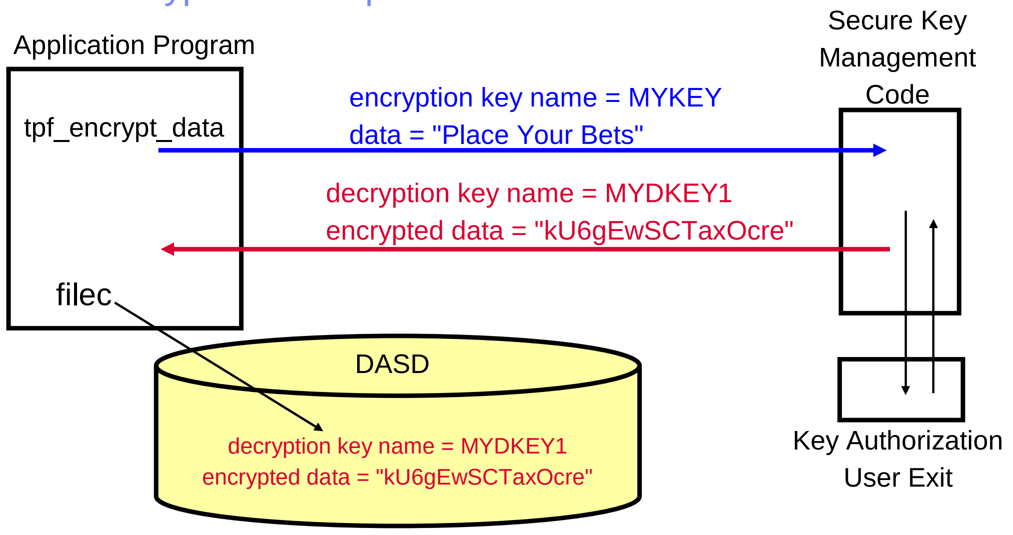


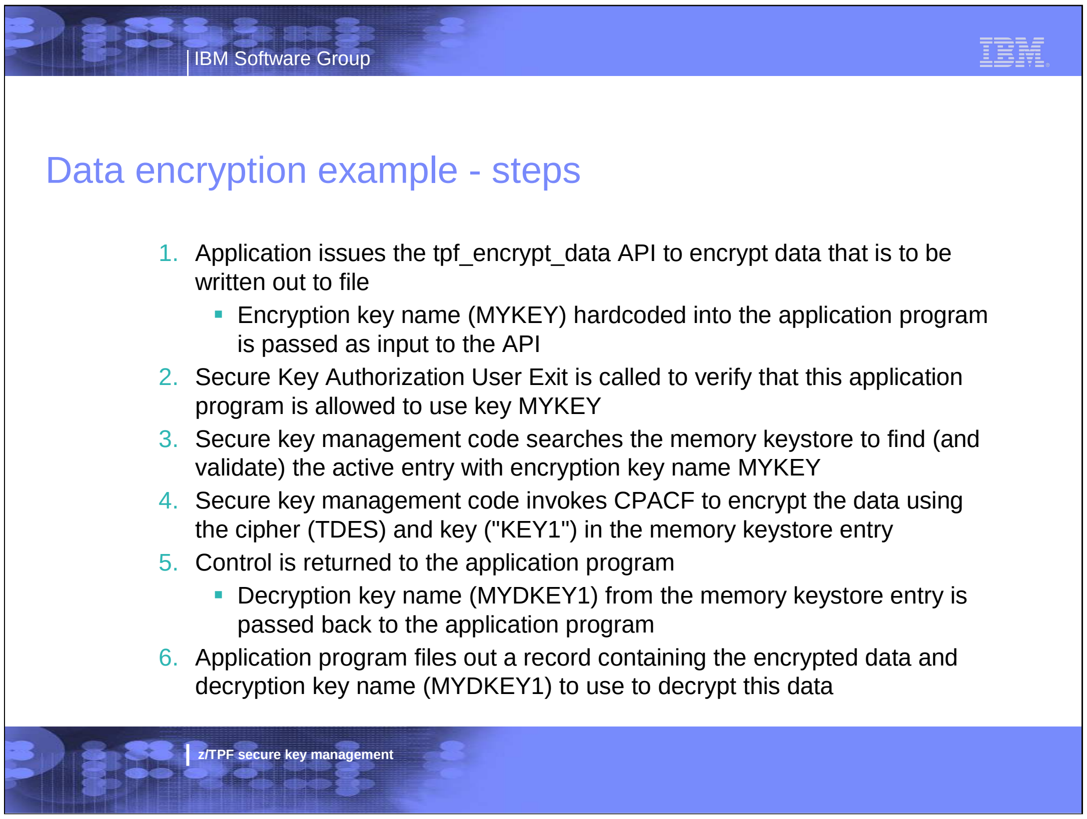
In this example...

- Keystore entry exists with the following information:
  - Encryption key name is MYKEY
  - Decryption key name is MYDKEY1
  - The encryption key is marked as active
  - Cipher is TDES
  - The value of the key is "KEY1"

```
Encryption Key Name  Decryption Key Name  Active  Cipher   Key
-------------------  -------------------  ------  ------  ------
       MYKEY               MYDKEY1          YES    TDES   "KEY1"
```

Data encryption example

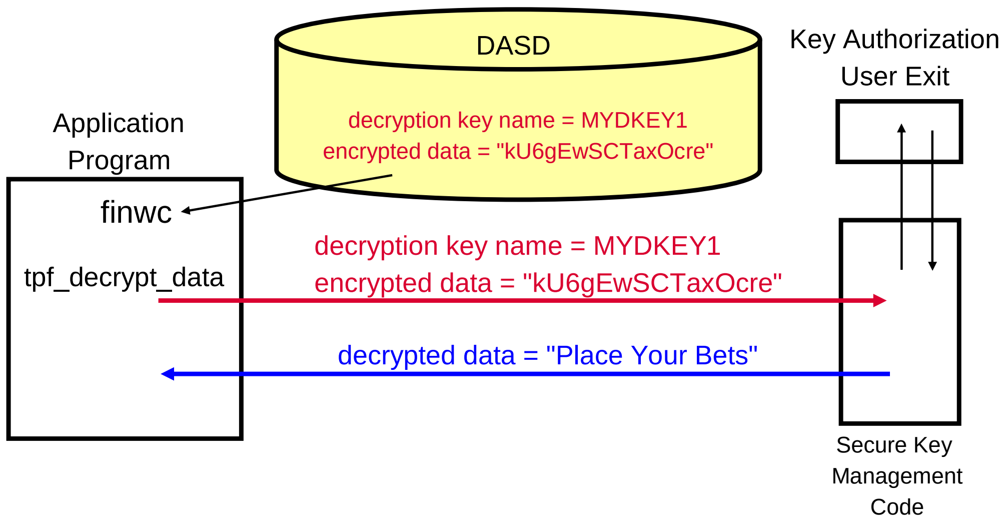

Application Program
tpf_encrypt_data
filec
Secure Key Management Code
encryption key name = MYKEY
data = "Place Your Bets"
decryption key name = MYDKEY1
encrypted data = "kU6gEwSCTaxOcre"
DASD
decryption key name = MYDKEY1
encrypted data = "kU6gEwSCTaxOcre"
Key Authorization User Exit

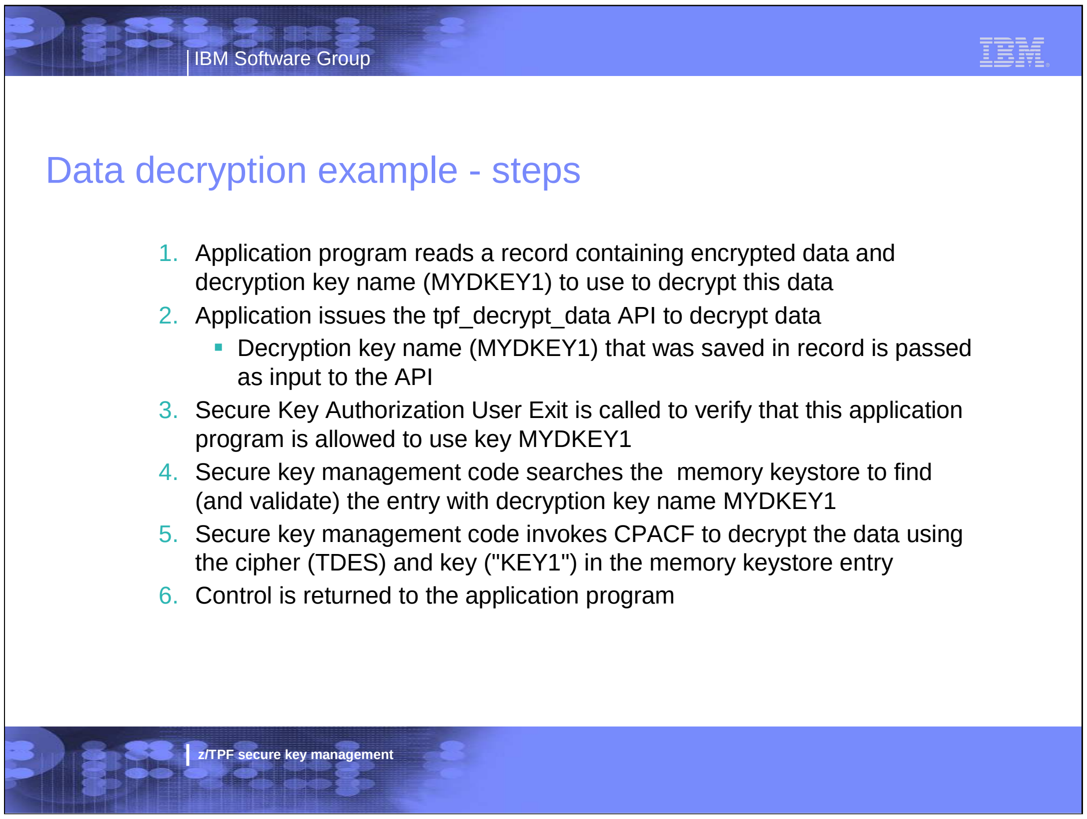


Data encryption example - steps

1. Application issues the tpf_encrypt_data API to encrypt data that is to be written out to file
   - Encryption key name (MYKEY) hardcoded into the application program is passed as input to the API
2. Secure Key Authorization User Exit is called to verify that this application program is allowed to use key MYKEY
3. Secure key management code searches the memory keystore to find (and validate) the active entry with encryption key name MYKEY
4. Secure key management code invokes CPACF to encrypt the data using the cipher (TDES) and key ("KEY1") in the memory keystore entry
5. Control is returned to the application program
   - Decryption key name (MYDKEY1) from the memory keystore entry is passed back to the application program
6. Application program files out a record containing the encrypted data and decryption key name (MYDKEY1) to use to decrypt this data

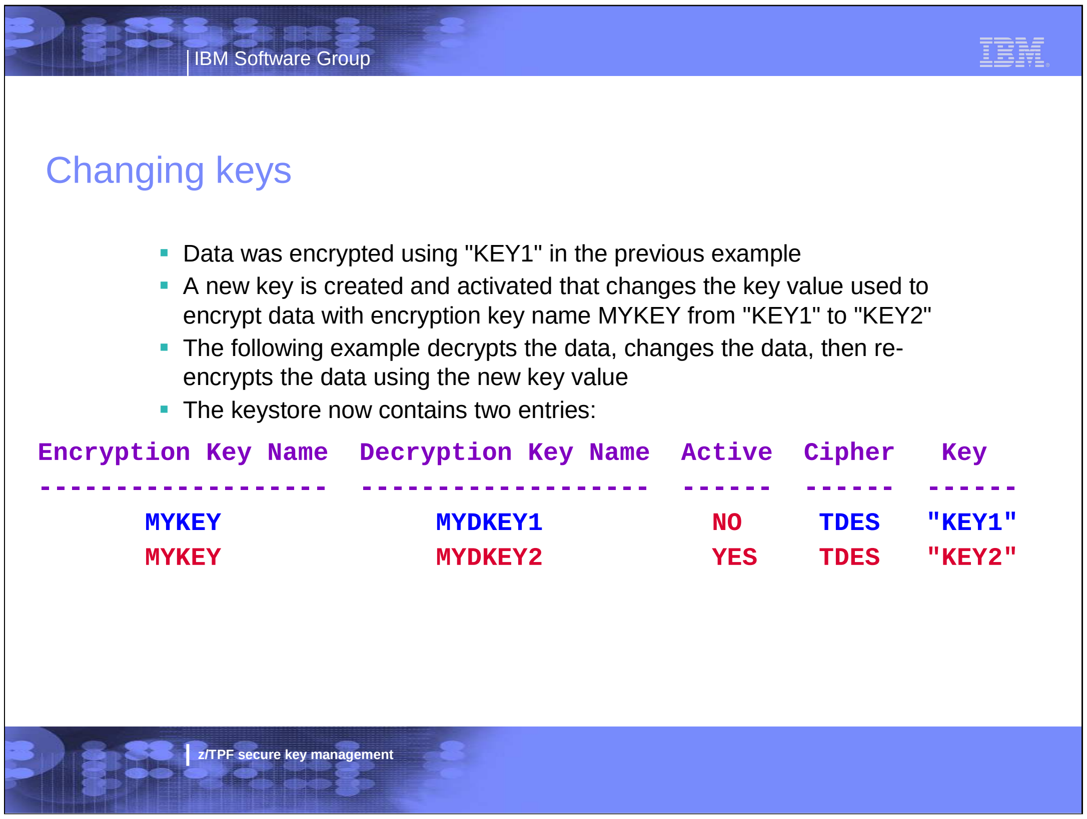
Data decryption example


DASD
decryption key name = MYDKEY1
encrypted data = "kU6gEwSCTaxOcre"
Key Authorization User Exit
Application Program
finwc
tpf_decrypt_data
decryption key name = MYDKEY1
encrypted data = "kU6gEwSCTaxOcre"
decrypted data = "Place Your Bets"
Secure Key Management Code



Data decryption example - steps

1. Application program reads a record containing encrypted data and decryption key name (MYDKEY1) to use to decrypt this data
2. Application issues the tpf_decrypt_data API to decrypt data
   - Decryption key name (MYDKEY1) that was saved in record is passed as input to the API
3. Secure Key Authorization User Exit is called to verify that this application program is allowed to use key MYDKEY1
4. Secure key management code searches the memory keystore to find (and validate) the entry with decryption key name MYDKEY1
5. Secure key management code invokes CPACF to decrypt the data using the cipher (TDES) and key ("KEY1") in the memory keystore entry
6. Control is returned to the application program

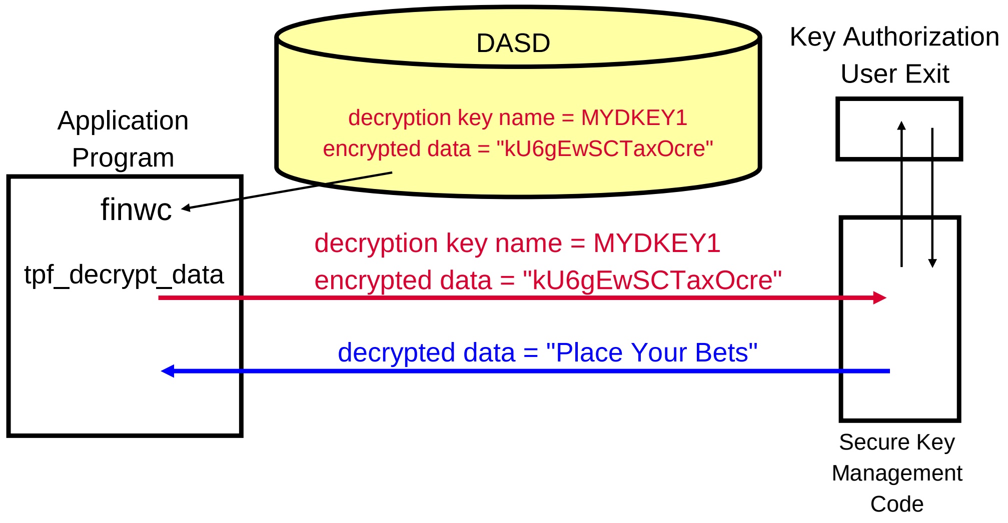


Changing keys

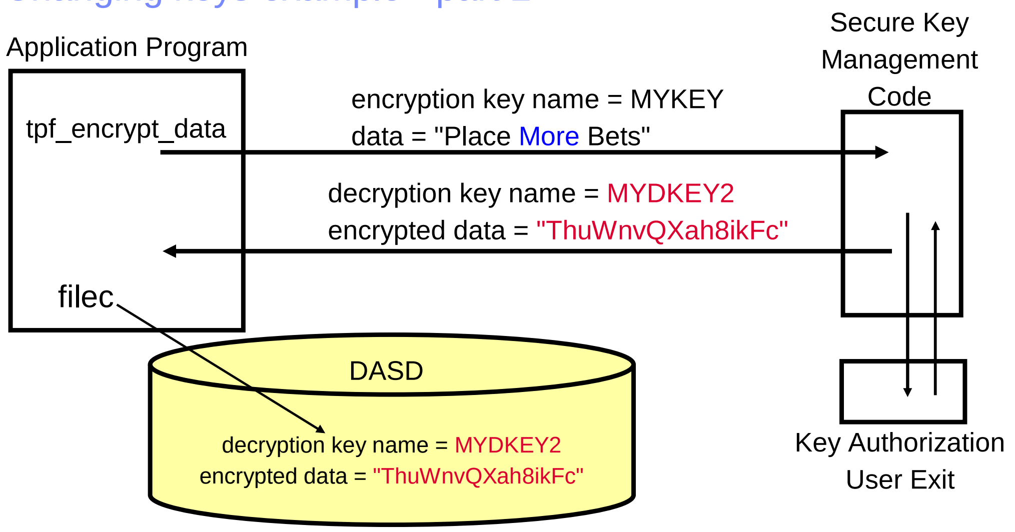
- Data was encrypted using "KEY1" in the previous example
- A new key is created and activated that changes the key value used to encrypt data with encryption key name MYKEY from "KEY1" to "KEY2"
- The following example decrypts the data, changes the data, then re-encrypts the data using the new key value
- The keystore now contains two entries:

```
Encryption Key Name  Decryption Key Name  Active  Cipher   Key
-------------------  -------------------  ------  ------  ------
       MYKEY               MYDKEY1          NO     TDES   "KEY1"
       MYKEY               MYDKEY2          YES    TDES   "KEY2"
```

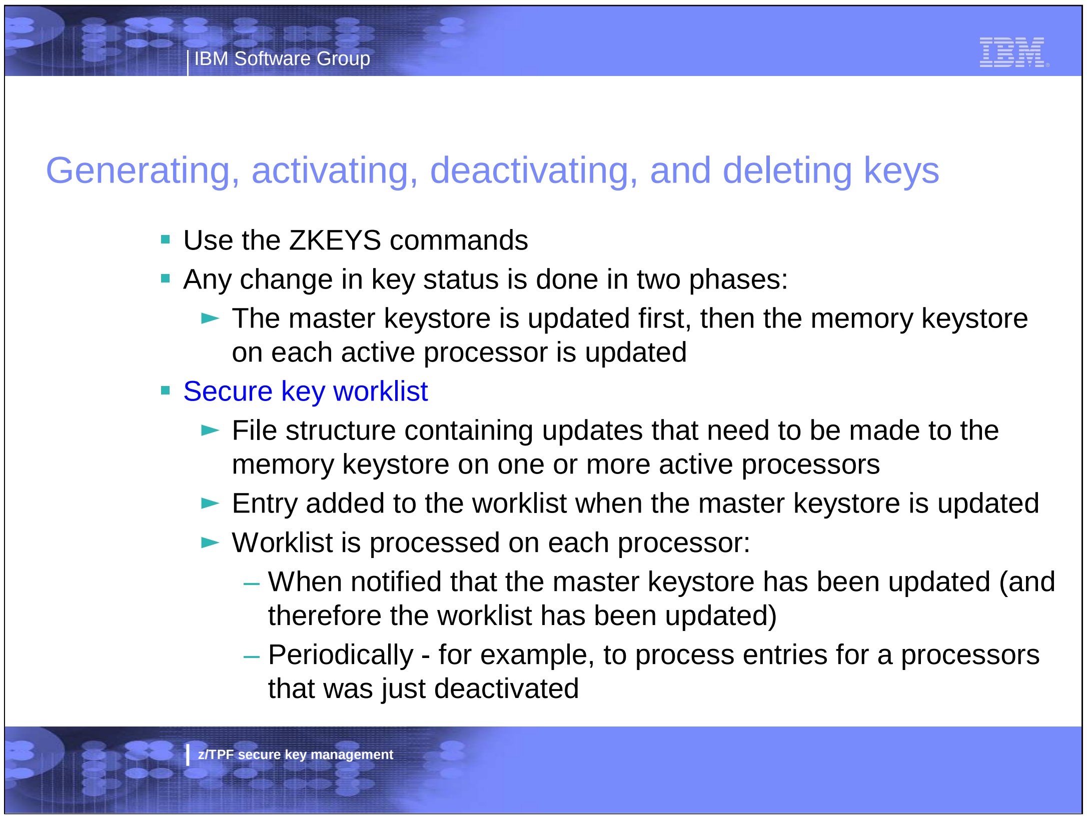
Changing keys example - part 1


DASD
decryption key name = MYDKEY1
encrypted data = "kU6gEwSCTaxOcre"
Key Authorization User Exit
Application Program
finwc
tpf_decrypt_data
decryption key name = MYDKEY1
encrypted data = "kU6gEwSCTaxOcre"
decrypted data = "Place Your Bets"
Secure Key Management Code

Changing keys example - part 2


Application Program
tpf_encrypt_data
filec
Secure Key Management Code
encryption key name = MYKEY
data = "Place More Bets"
decryption key name = MYDKEY2
encrypted data = "ThuWnvQXah8ikFc"
DASD
decryption key name = MYDKEY2
encrypted data = "ThuWnvQXah8ikFc"
Key Authorization User Exit

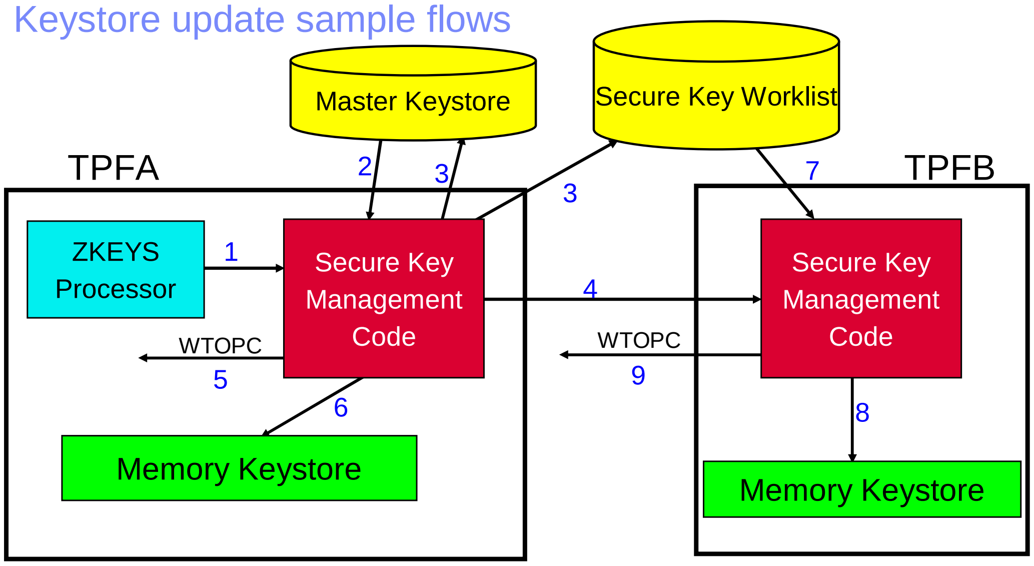
How keystore updates are made

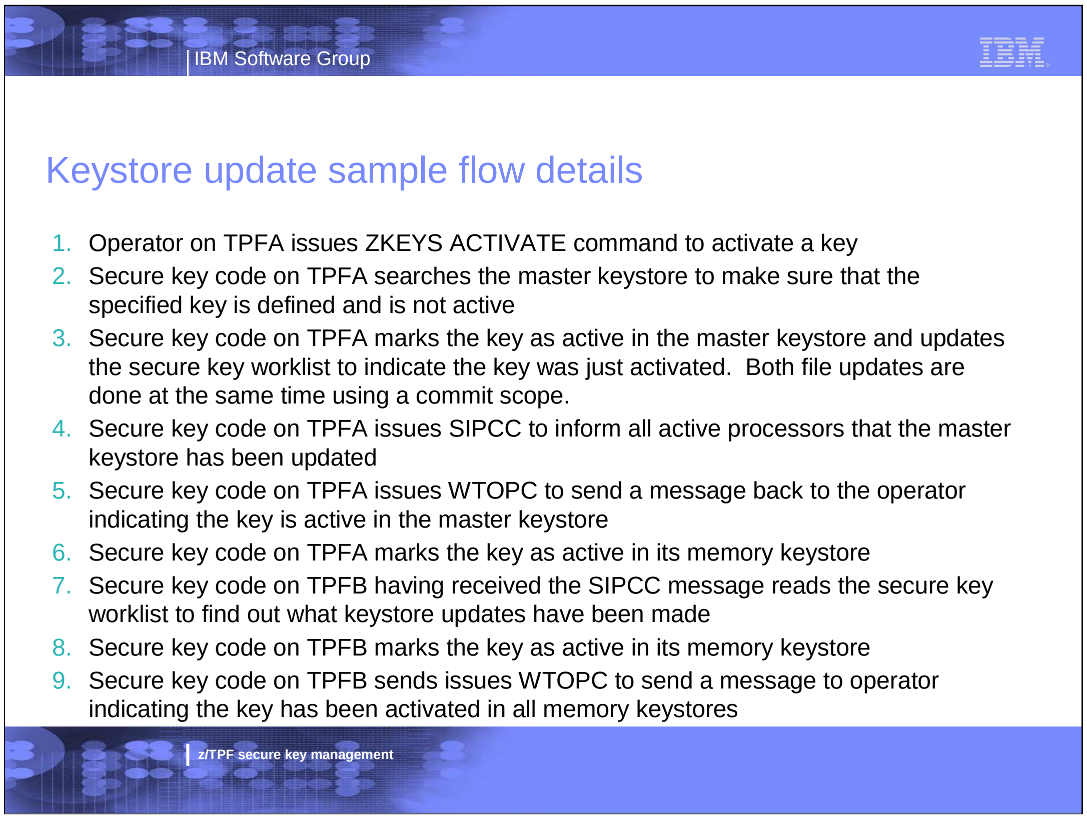


Generating, activating, deactivating, and deleting keys

- Use the ZKEYS commands
- Any change in key status is done in two phases:
  - The master keystore is updated first, then the memory keystore on each active processor is updated
- Secure key worklist
  - File structure containing updates that need to be made to the memory keystore on one or more active processors
  - Entry added to the worklist when the master keystore is updated
  - Worklist is processed on each processor:
    - When notified that the master keystore has been updated (and therefore the worklist has been updated)
    - Periodically - for example, to process entries for a processors that was just deactivated

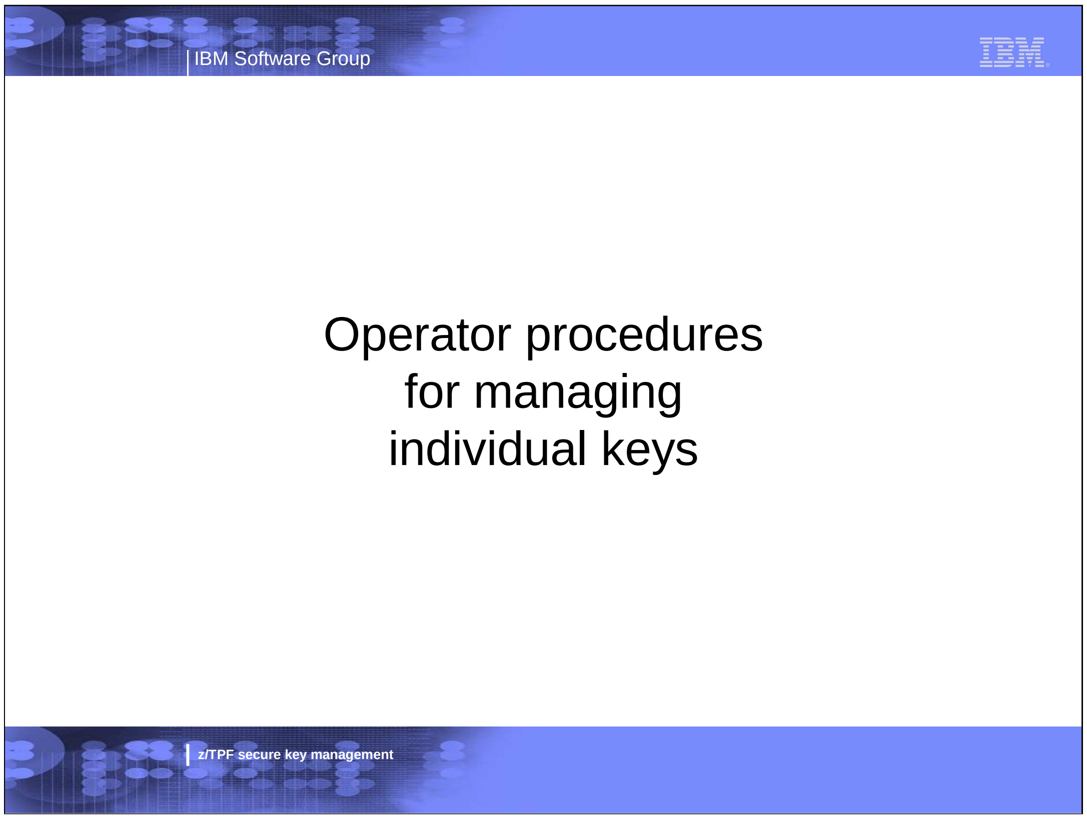
Keystore update sample flows



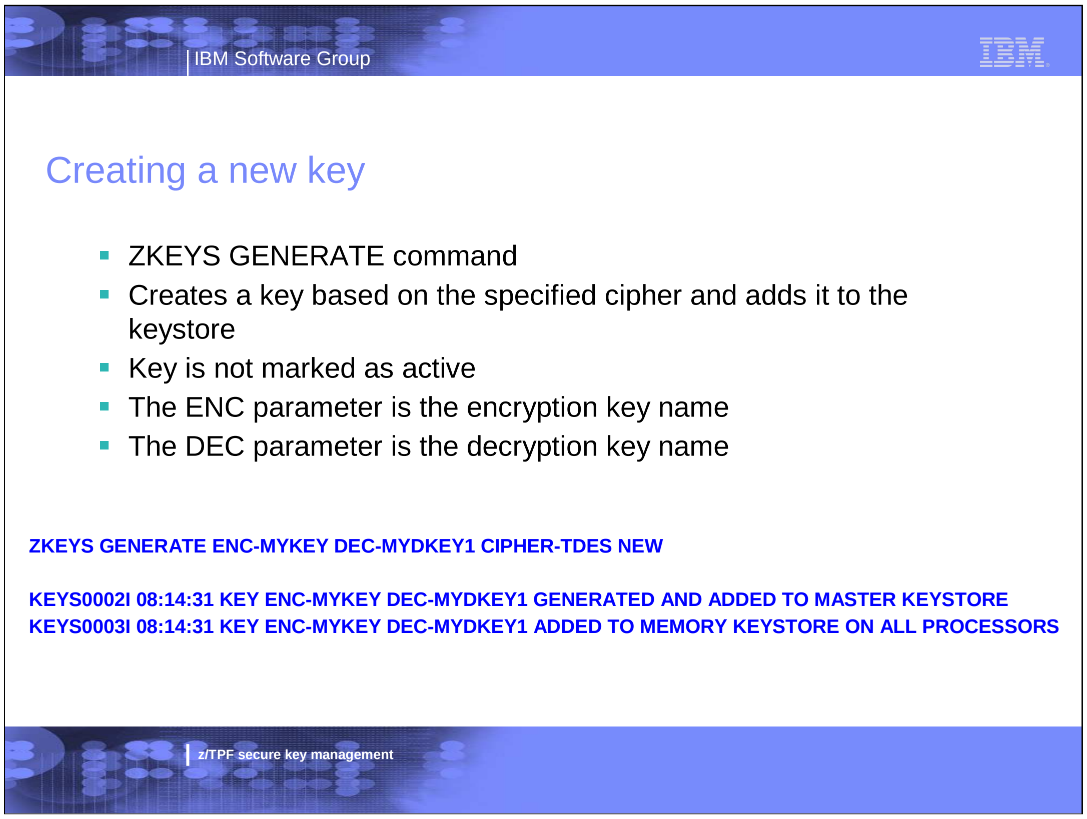


Keystore update sample flow details

1. Operator on TPFA issues ZKEYS ACTIVATE command to activate a key
2. Secure key code on TPFA searches the master keystore to make sure that the specified key is defined and is not active
3. Secure key code on TPFA marks the key as active in the master keystore and updates the secure key worklist to indicate the key was just activated. Both file updates are done at the same time using a commit scope.
4. Secure key code on TPFA issues SIPCC to inform all active processors that the master keystore has been updated
5. Secure key code on TPFA issues WTOPC to send a message back to the operator indicating the key is active in the master keystore
6. Secure key code on TPFA marks the key as active in its memory keystore
7. Secure key code on TPFB having received the SIPCC message reads the secure key worklist to find out what keystore updates have been made
8. Secure key code on TPFB marks the key as active in its memory keystore
9. Secure key code on TPFB sends issues WTOPC to send a message to operator indicating the key has been activated in all memory keystores



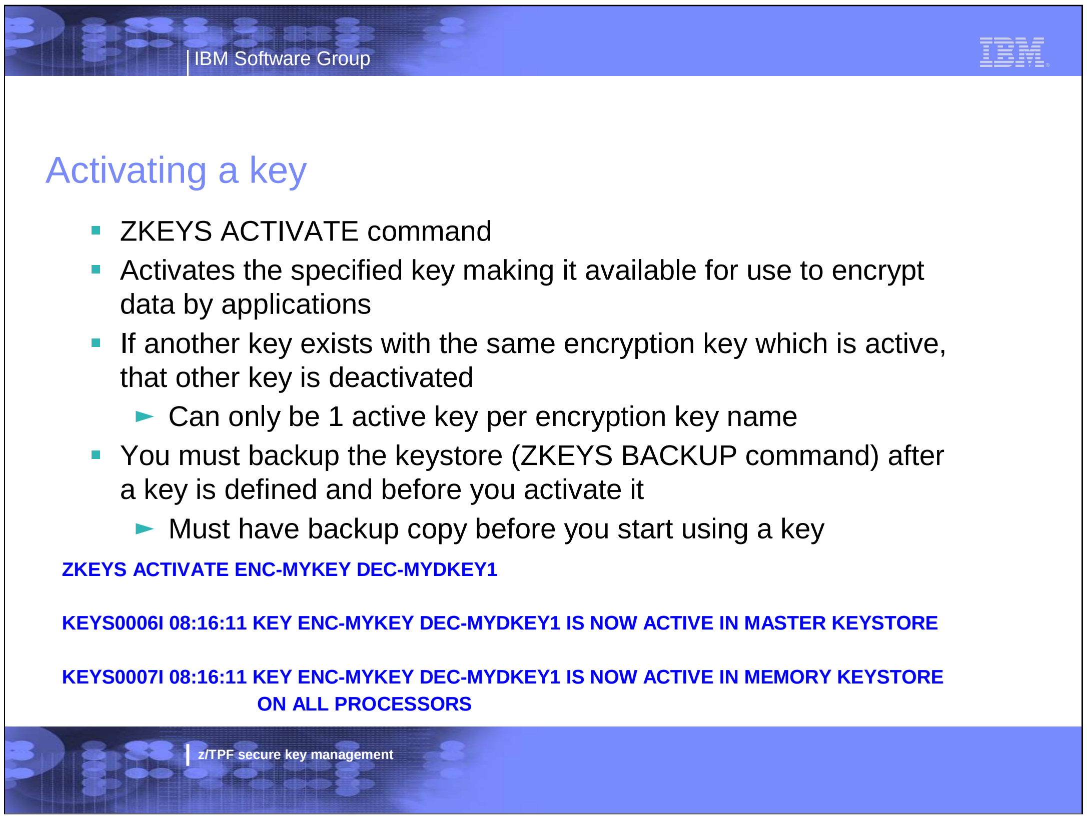
Operator procedures for managing individual keys



Creating a new key

- ZKEYS GENERATE command
- Creates a key based on the specified cipher and adds it to the keystore
- Key is not marked as active
- The ENC parameter is the encryption key name
- The DEC parameter is the decryption key name

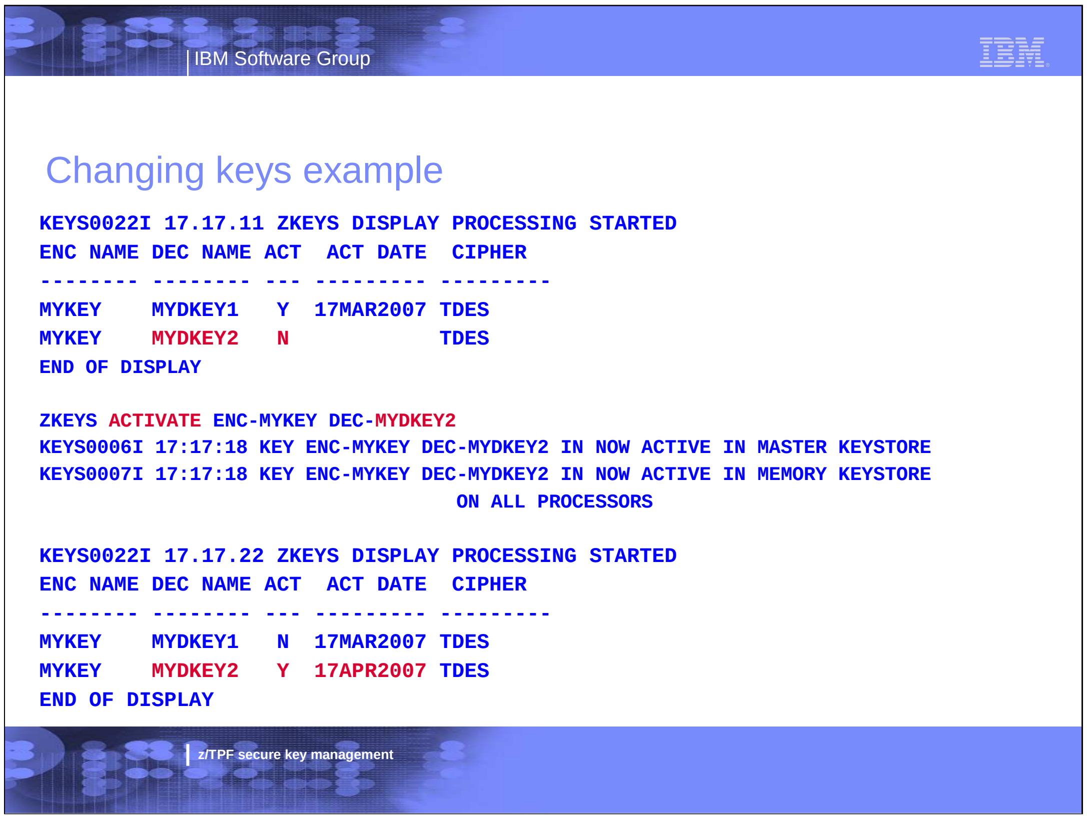
```
ZKEYS GENERATE ENC-MYKEY DEC-MYDKEY1 CIPHER-TDES NEW

KEYS0002I 08:14:31 KEY ENC-MYKEY DEC-MYDKEY1 GENERATED AND ADDED TO MASTER KEYSTORE
KEYS0003I 08:14:31 KEY ENC-MYKEY DEC-MYDKEY1 ADDED TO MEMORY KEYSTORE ON ALL PROCESSORS
```



Activating a key

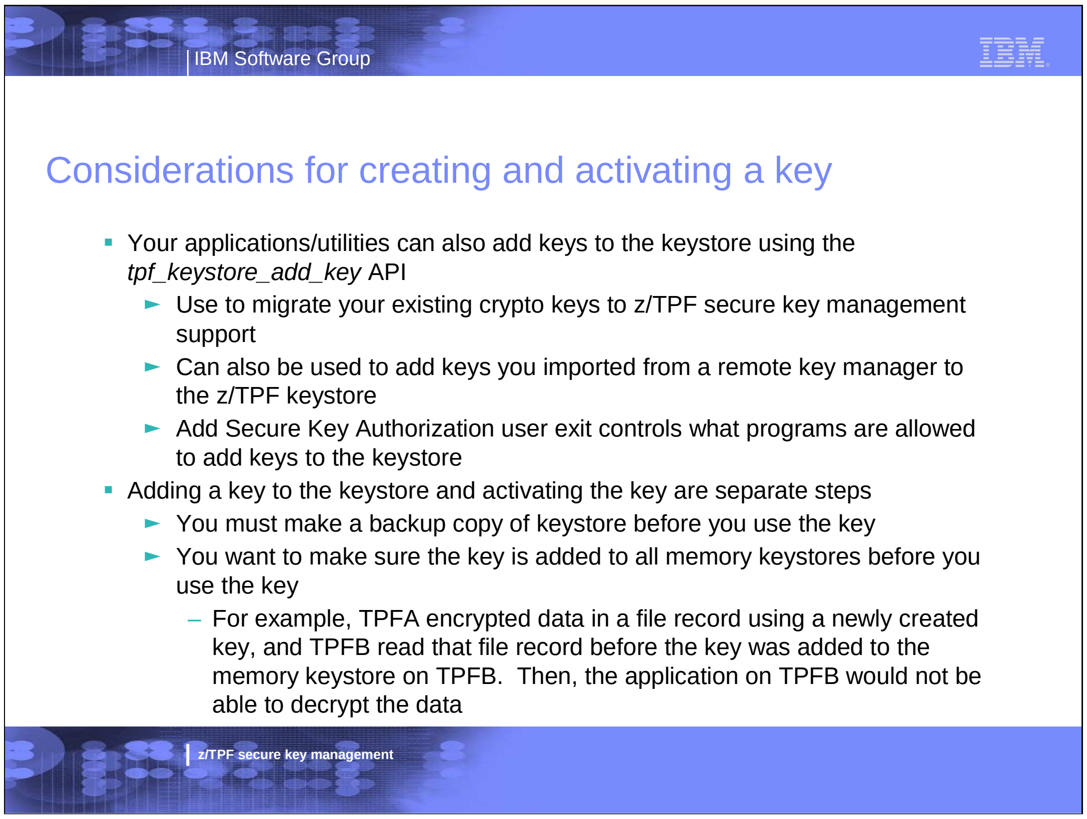
- ZKEYS ACTIVATE command
- Activates the specified key making it available for use to encrypt data by applications
- If another key exists with the same encryption key which is active, that other key is deactivated
  - Can only be 1 active key per encryption key name
- You must backup the keystore (ZKEYS BACKUP command) after a key is defined and before you activate it
  - Must have backup copy before you start using a key

```
ZKEYS ACTIVATE ENC-MYKEY DEC-MYDKEY1

KEYS0006I 08:16:11 KEY ENC-MYKEY DEC-MYDKEY1 IS NOW ACTIVE IN MASTER KEYSTORE

KEYS0007I 08:16:11 KEY ENC-MYKEY DEC-MYDKEY1 IS NOW ACTIVE IN MEMORY KEYSTORE
                   ON ALL PROCESSORS
```



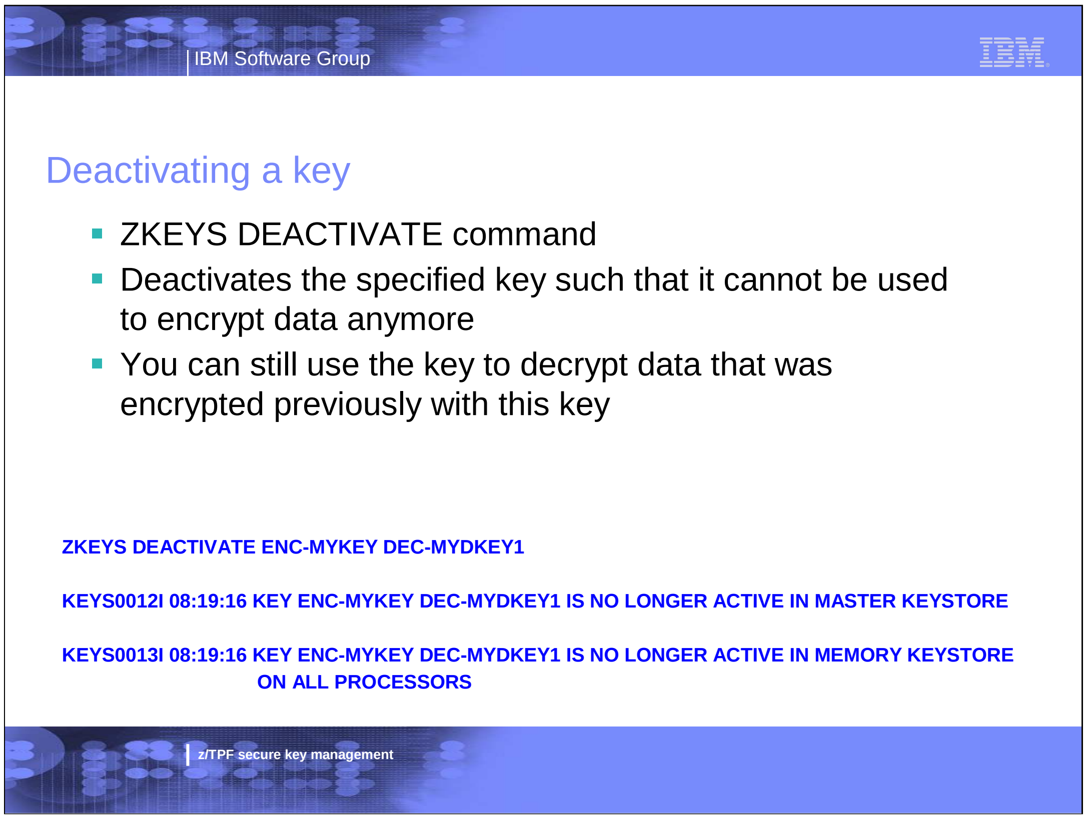
Changing keys example

```
KEYS0022I 17.17.11 ZKEYS DISPLAY PROCESSING STARTED
ENC NAME DEC NAME ACT  ACT DATE  CIPHER
-------- -------- --- --------- ---------
MYKEY    MYDKEY1   Y  17MAR2007 TDES
MYKEY    MYDKEY2   N            TDES
END OF DISPLAY

ZKEYS ACTIVATE ENC-MYKEY DEC-MYDKEY2
KEYS0006I 17:17:18 KEY ENC-MYKEY DEC-MYDKEY2 IN NOW ACTIVE IN MASTER KEYSTORE
KEYS0007I 17:17:18 KEY ENC-MYKEY DEC-MYDKEY2 IN NOW ACTIVE IN MEMORY KEYSTORE
                                    ON ALL PROCESSORS

KEYS0022I 17.17.22 ZKEYS DISPLAY PROCESSING STARTED
ENC NAME DEC NAME ACT  ACT DATE  CIPHER
-------- -------- --- --------- ---------
MYKEY    MYDKEY1   N  17MAR2007 TDES
MYKEY    MYDKEY2   Y  17APR2007 TDES
END OF DISPLAY
```



Considerations for creating and activating a key

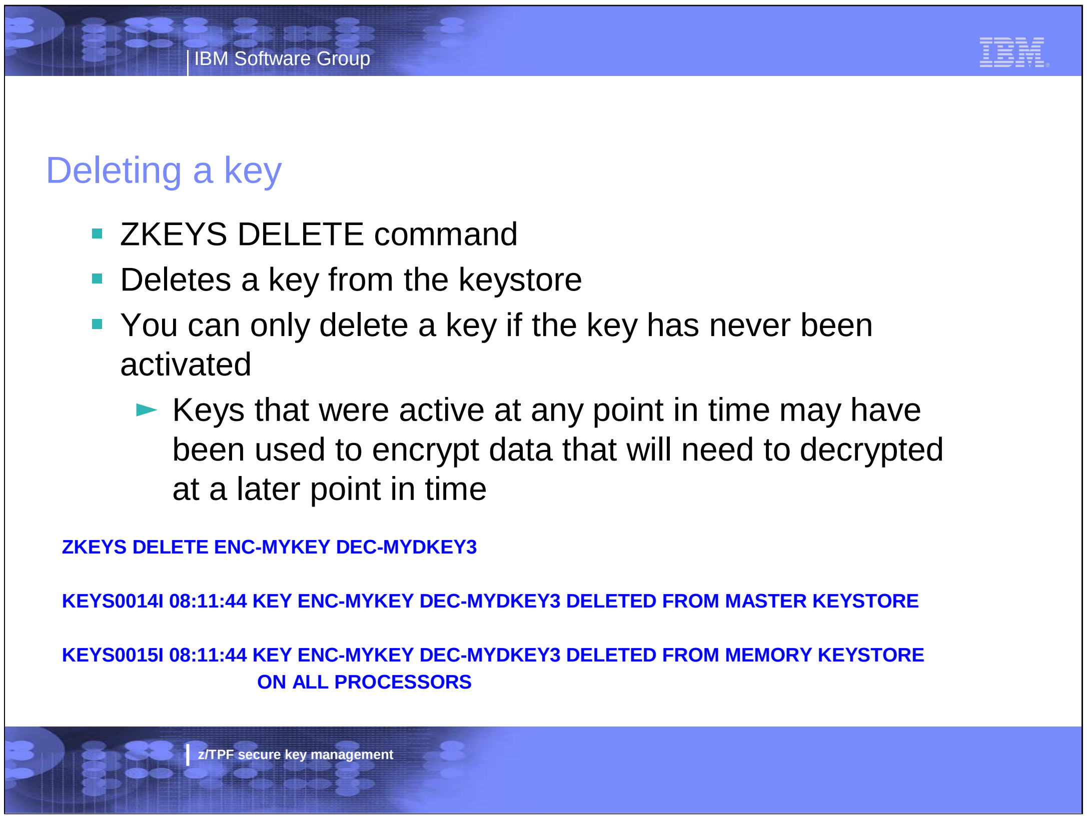
- Your applications/utilities can also add keys to the keystore using the *tpf_keystore_add_key* API
  - Use to migrate your existing crypto keys to z/TPF secure key management support
  - Can also be used to add keys you imported from a remote key manager to the z/TPF keystore
  - Add Secure Key Authorization user exit controls what programs are allowed to add keys to the keystore
- Adding a key to the keystore and activating the key are separate steps
  - You must make a backup copy of keystore before you use the key
  - You want to make sure the key is added to all memory keystores before you use the key
    - For example, TPFA encrypted data in a file record using a newly created key, and TPFB read that file record before the key was added to the memory keystore on TPFB. Then, the application on TPFB would not be able to decrypt the data



Deactivating a key

- ZKEYS DEACTIVATE command
- Deactivates the specified key such that it cannot be used to encrypt data anymore
- You can still use the key to decrypt data that was encrypted previously with this key

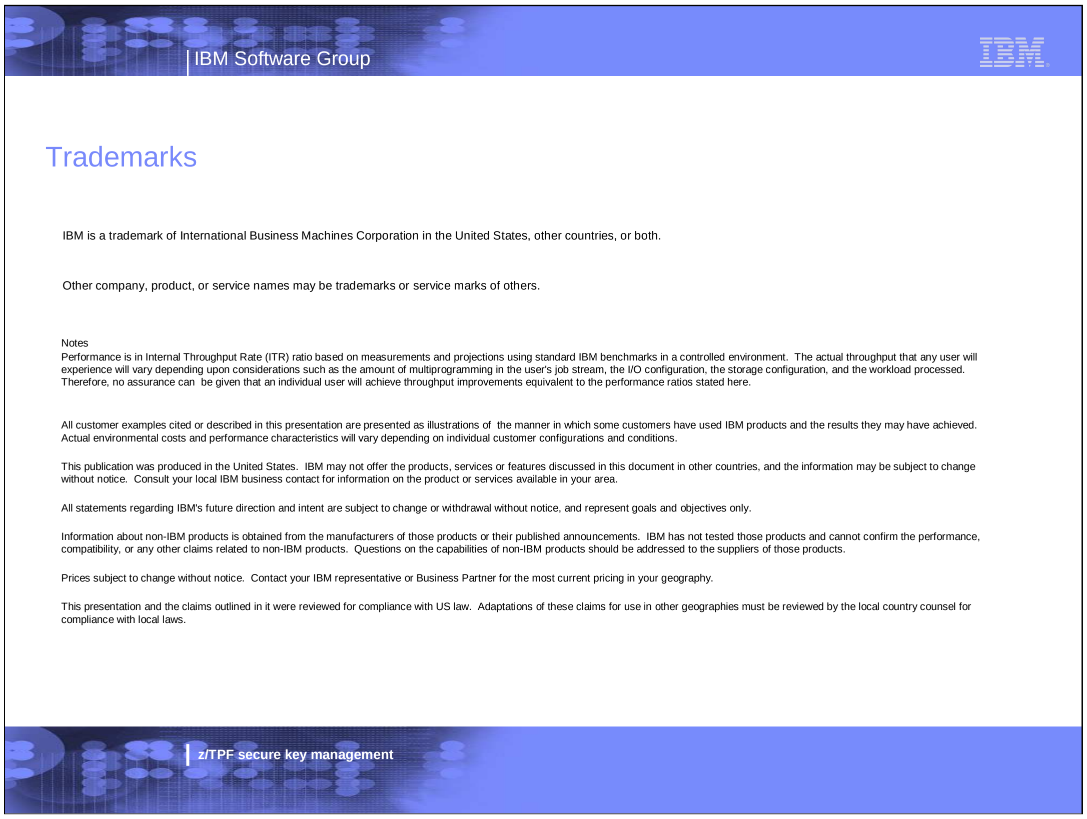
```
ZKEYS DEACTIVATE ENC-MYKEY DEC-MYDKEY1

KEYS0012I 08:19:16 KEY ENC-MYKEY DEC-MYDKEY1 IS NO LONGER ACTIVE IN MASTER KEYSTORE

KEYS0013I 08:19:16 KEY ENC-MYKEY DEC-MYDKEY1 IS NO LONGER ACTIVE IN MEMORY KEYSTORE
                   ON ALL PROCESSORS
```



Deleting a key

- ZKEYS DELETE command
- Deletes a key from the keystore
- You can only delete a key if the key has never been activated
  - Keys that were active at any point in time may have been used to encrypt data that will need to decrypted at a later point in time

```
ZKEYS DELETE ENC-MYKEY DEC-MYDKEY3

KEYS0014I 08:11:44 KEY ENC-MYKEY DEC-MYDKEY3 DELETED FROM MASTER KEYSTORE

KEYS0015I 08:11:44 KEY ENC-MYKEY DEC-MYDKEY3 DELETED FROM MEMORY KEYSTORE
                   ON ALL PROCESSORS
```



Trademarks