

Hauptseminar

Applet Firewall und Freigabe der Objekte

Nachweis von Sicherheitseigenschaften für JavaCard

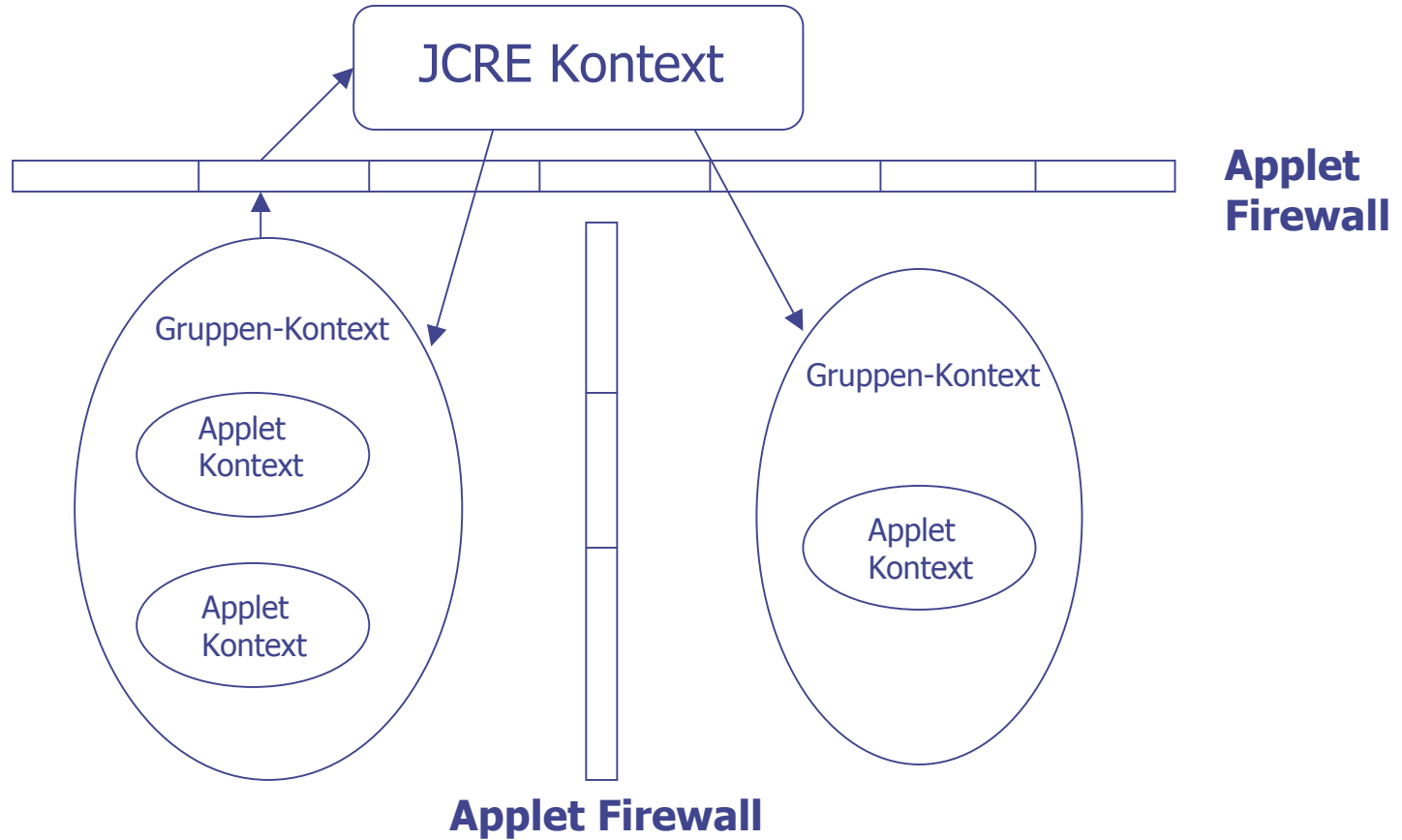
Jin Zhou

Hauptseminar Java Card
November 2001

Ein Überblick über diesen Vortrag

- Applet Firewall
- Kontext
- JCRE Entry Point Objekt
- Shareable Interface Objekt
- Beglaubigung der Freigabe

Kontext



Ausnahme für Kontextüberprüfung

- Statische Arrays
- Statische Felder und Methoden

Beispiel:

```
If (apdu_buffer[ISO07816.OFFSET_CLA] != EXPECTED_VALUE)  
    ISOException.throwIt(ISO07816.SW_CLA_NOT_SUPPORTED);
```

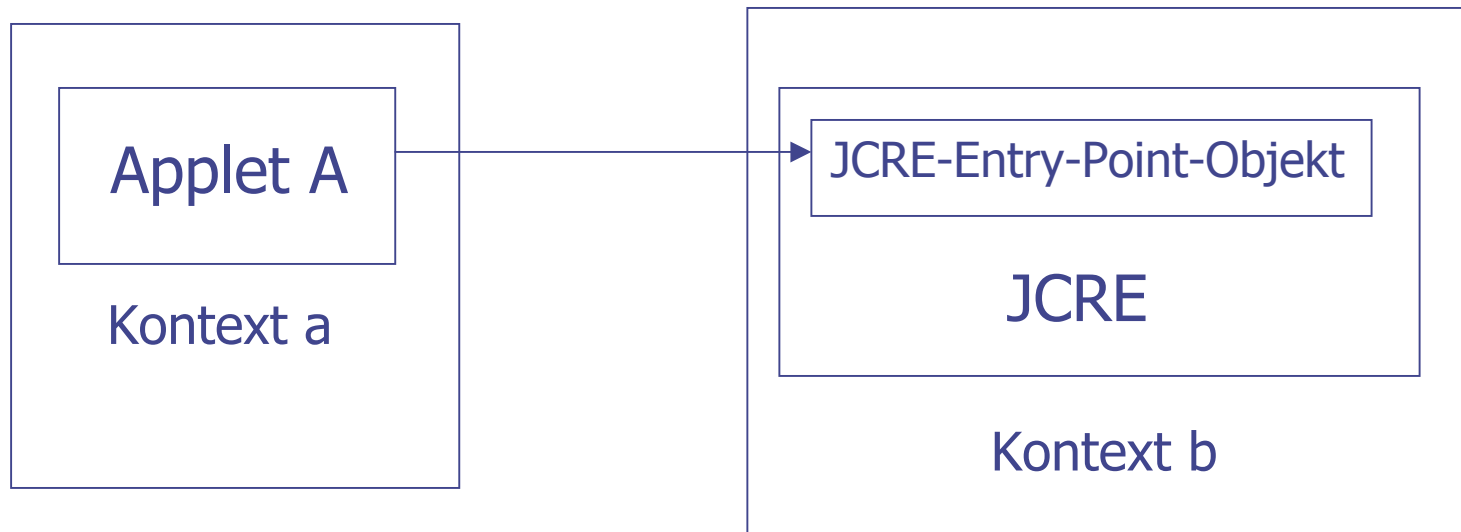
Objekt freigabe über den Kontexten

Das Freigabemechanismus der Java Card Technologie:

- JCRE Privilegien
- JCRE Entry Point Objekte
- Globale Arrays
- Shareable Interface

Kontextumwandlung

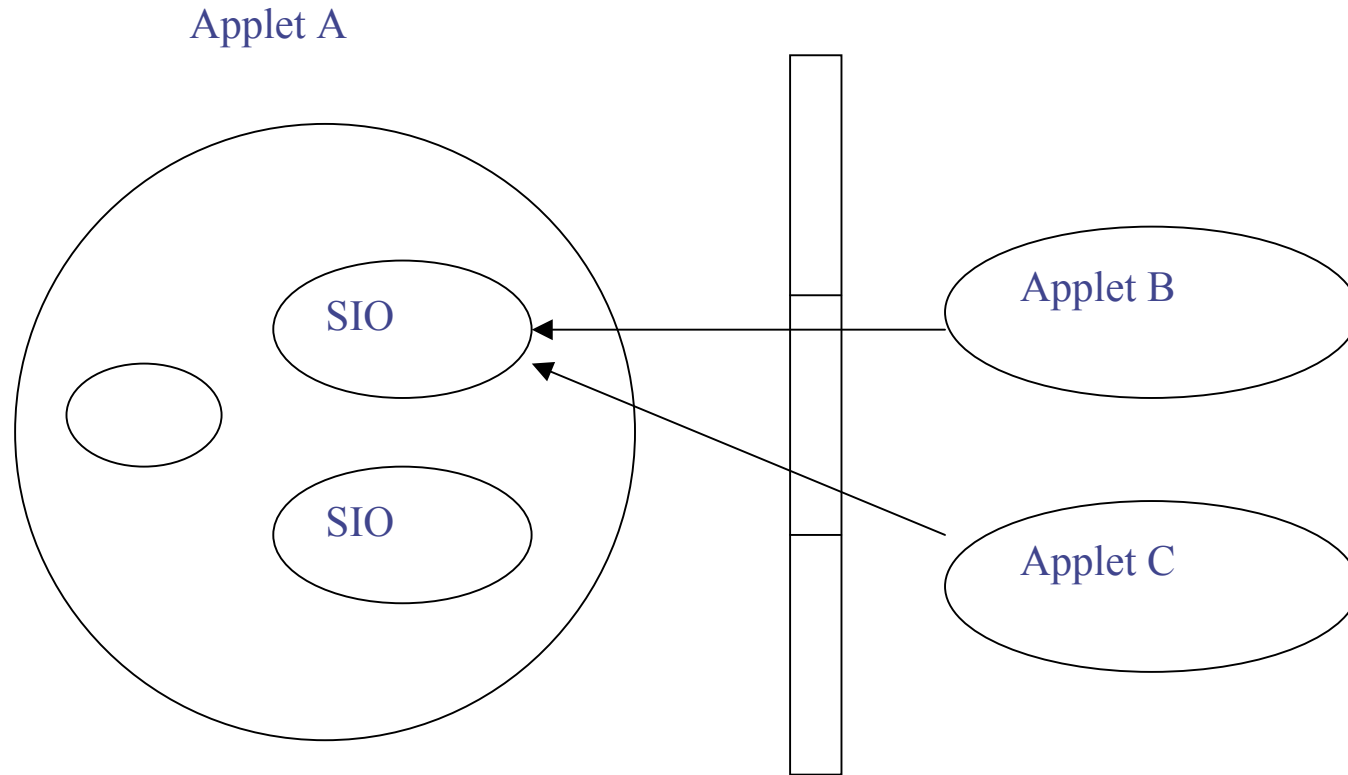
1. der aktuelle Kontext wird gespeichert
2. Kontextumwandlung
3. Am Ende der Methode der ursprüngliche Kontext als der aktuell aktive Kontext wiederhergestellt



Freigabemechanismus

- JCRE Privilegien
- JCRE Entry Point Objekte (zwei Kategorien)
 - Temporäre JCRE Entry Point Objekte : kann nicht speichern
z.B.: JCRE Exception-Objekte
 - Ständige JCRE Entry Point Objekte : kann speichern
z.B.: JCRE AID
- Globale Arrays
- Shareable Interface

Gedanke hinter dem SI Mechanismus



Objekt-Shareabel-Interface-Mechanismus

1. Shareable Interface:

Ist vererbt vom Interface *javacard.framework.Shareable*

Z.B.: `public interface LuftMeilenInterface extends Shareable`

2. Shareable Interface Objekt (SIO):

Objekt, das Shareable Interface implementiert.

Z.B.: `public class LuftMeilenApp extends Applet implements LuftMeilenInterface`

Ein Beispiel von Objektfreigabe zwischen Applets

1. Das Luft-Meilen-Applet erzeugt ein SIO.
2. Das Brieftasche-Applet fordert ein SIO von dem Luft-Meilen-Applet auf.
3. Das Brieftasche-Applet fordert auf, dass die Meilen addiert werden, indem es eine Service-Methode vom SIO aufruft.



Ein Shareable Interface erzeugen

Zur Erzeugung eines SIO muß das Server-Applet (das Luft-Meilen-Applet) erst ein Shareable Interface definieren, das vom *javacard.framework.Shareable* vererbt wird.

```
package com.fasttravel.luftmeilen;
import javacard.framework.Shareable;

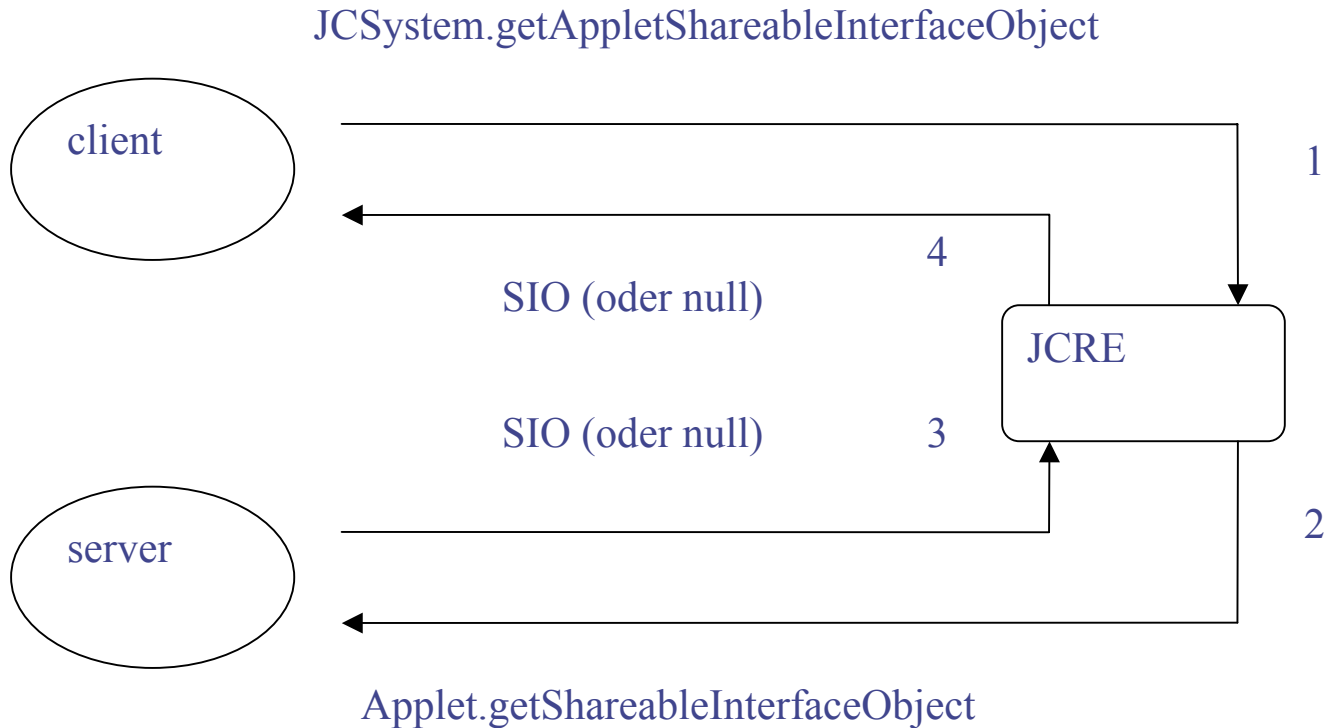
public interface LuftMeilenInterface extends Shareable{
    public void grantMeilen (short amount);
}
```

Ein Shareable Klasse erzeugen

Und dann erzeugt das Server-Applet eine Serviceversorgerklasse die das Shareable Interface implementiert.

```
package com.fasttravel.luftmeilen;
import javacard.framework.*;
public class LuftMeilenApp extends Applet implements
LuftMeilenInterface{
    private short meilen;
    public void grantMeilen(short amount){
        meilen = (short)(meilen + amount);
    }
}
```

Aufforderung an einen Shareable Objekt



Aufforderung an einen Shareable Objekt

3 wichtige Methoden:

```
public static lookupAID(byte[] buffer, short offset,  
                        byte length)
```

```
public static Shareable getAppletShareableInterfaceObject(  
    AID server_aid, byte parameter)
```

```
public Shareable getShareableInterfaceObject(  
    AID client_aid, byte parameter)
```

Aufforderung an einen Shareable Objekt

Das Server-Applet:

```
public class LuftMeilenApp extends Applet implements
LuftMeilenInterface{
    short meilen;
    public Shareable getShareableInterfaceObject(
        AID client_aid, byte parameter){
        return this; // das SIO zurückgeben
    }
    public void grantMeilen(short amount){
        meilen = (short)(meilen + amount);
    }
}
```

Anwendung mit einem SIO

```
package com.smartbank.brieftasche;
import javacard.framework.*;
import com.fasttravel.luftmeilen.LuftMeilenInterface;
public class BrieftascheApp extends Applet{
    private short balance;
    private byte[] luft_meilen_aid_bytes = SERVER_AID_BYTES;
    //Geld ausgeben
    private void debit(short amount){
        if(balance < amount) ISOException.throwIt(SW_EXCEED_BALANCE);
        balance = (short)(balance - amount);
        // das Meilenaddierungsservice auffordern
        requestMeilen(amount);
    }
}
```

```

private void requestmeilen(short amount){
    //ein Server AID-Objekt bekommen

    AID luft_meilen_aid = JCSystem.lookupAID(luft_meilen_aid_bytes,
    (short)0, (byte)luft_meilen_aid_bytes.length);

    if(luft_meilen_aid == null)
    ISOException.throwIt(SW_LUFT_MEILEN_APP_NOT_EXIST);

    //das SIO vom Server auffordern

    LuftMeilenInterface sio = (LuftMeilenInterface)
    (JCSystem.getAppletShareableInterfaceObjekt(luft_meilen_aid, SECRET));

    If(sio == null) ISOException.thowIt(SW_FAILED_TO_OBTAIN_SIO);

    // das Meilenaddirungsservice vom Server auffordern

    sio.grantMeilen(amount);
}
}

```

Beglaubigung eines Client-Applets

```
public class LuftMeilenApp extends Applet implements
    LuftMeilenInterface{

    public Shareable getShareableInterfaceObject(AID client_aid,
        byte parameter){

        if(client_aid.equals(brieftasche_app_aid_bytes, (short)0,
            (byte)(brieftasche_app_aid_bytes.length)) == false)
            return null;

        return (this);
    }
}
```

Beglaubigung eines Client-Applets

```
public void grantMeilen(short amount){
    //das AID des Aufrufers bekommen
    AID client_aid = JCSystem.getpreviousContextAID();

    //überprüfen, ob die Methode richtig vom Brieftasche-
    Applet aufgerufen wird
    if(client_aid.equals(brieftasche_app_aid_bytes, (short)0,
    (byte)( brieftasche_app_aid_bytes.length)) == false)
    ISOException.throwIt (SW_UNAUTHORIZED_CLIENT);

    meilen = (short)(meilen + amount);
}
```

Challenge-Response Schema

```
public class LuftMeilenApp extends Applet implements
LuftmeilenInterface{
    public void grantmeilen(AuthenticationInterface authObject,
byte[] buffer, short amount){
    // eine gelegentliche Prüfungsphrase im Buffer erzeugen
    generateChallenge(buffer);
    //prüft das Client-Applet
    //die Antwort ist in den buffer gespeichert
    authObject.challenge(buffer);
    //die Antwort prüfen
    if(checkResponse(buffer) == false) ISOException.throwIt
(SW_UNAUTHORIZED_CLIENT);
    meilen = (short)(meilen + amount);
    }
}
```

Challenge-Response Schema

```
public interface AuthenticationInterface extends Shareable{  
    public void challenge(byte[] buffer);  
}
```

```
public class BrieftascheApp extends Applet implements AuthenticationInterface{  
    private byte[] buffer;  
    public void challenge(byte[] buffer){  
        private short balance;  
        this.buffer = buffer;  
        //Antwort bekommen  
        //die Prüfung und die beantworteten Daten sind in den Buffer  
speichern  
        getResponse(buffer);  
    }  
}
```

```
//Geld ausgeben
private void debit(short amount){
    if(balance < amount)
        ISOException.throwIt( SW_EXECEED_BALANCE);
    balance = (short)(balance – amount);
    //dem Luft-Meilen-Applet eine Meilenaddierung anfordern
    requestMeilen( buffer, amount);}
private void requestMeilen(byte[] buffer, short amount){
    //das AID-Objekt bekommen
    AID luft_meilen_aid = JCSytem.lookupAID(luft_meilen_aid_bytes,
(short)0, (byte) luft_meilen_aid_bytes.length);
    //das SIO vom Luft-Meilen-Applet auffordern
    LuftMeilenInterface sio = (LuftMeilenInterface)
(JCSytem.getAppletShareableInterfaceObject(luft_meilen_aid, SECRET));
    //dem Luft-Meilen-Applet eine Meilenaddierung anfordern
    sio.grantMeilen(this, buffer, amount);}
}
```

Zusammenfassung 1

1. Wenn das Server-Applet A ein Objekt mit anderem Applet freigeben will, muß es zuerst ein Shareable Interface SI definieren. Ein Shareable Interface ist vom Interface `javacard.framework.Shareable` vererbt. Die Methoden, die im SI repräsentiert werden, sind diejenigen, die das Applet A den anderen Applets zugreifbar lassen will.
2. Das Applet A definiert dann einen Serviceanbieter, nämlich Klasse C, die das SI implementiert. C bietet eine tatsächliche Implementation von den in SI definierten Methoden. C kann auch andere Methoden und Felder definieren, aber die sind vom Applet Firewall geschützt. Nur die im SI definierten Methoden sind zugreifbar für die anderen Applets.
3. Das Applet A erzeugt ein Objektinstanz O von der Klasse C. C gehört zum Applet A und das Firewall erlaubt den Zugriff von A an allen Felder und Methoden von C.

Zusammenfassung 2

4. Wenn ein Client-Applet B das Objekt O vom Applet A zugreifen will, ruft es die `JCSystem.getAppletShareableInterface` Methode zur Aufforderung einen SIO vom Applet A auf.
5. Das JCRE sucht in seiner internen Applettabelle nach dem Applet A. Wenn es A gefunden hat, ruft es dann die `getShareableInterfaceObject` Methode auf.
6. Das Applet A bekommt die Aufforderung und prüft, ob A dem Applet B das Objekt freigeben will. Wenn A mit der Freigabe mit B einverstanden ist, antwortet A dann B mit einer Referenz zu O.

Zusammenfassung 3

7. Das Applet B bekommt die Objektreferenz vom Applet A, castete sie zum Typ SI, und dann speichert sie in den SIO. Obwohl das SIO wirklich das Objekt O vom A anspricht, ist es noch vom Typ SI. Nur die im SI definierten shareable Interface Methoden sind sichtbar von B. Das Firewall verhindert, dass B andere Felder und Methoden von O zugreifen kann. Das Applet B kann Service vom Applet A auffordern, indem B eine von den SI-Methoden vom SIO aufruft.

8. Vor der Ausführung des Services kann die SI-Methode das Client(B) autehtisieren, um festzustellen, ob der Service bewilligt wird.