



JavaOneSM
Sun's 2003 Worldwide Java Developer Conference™

Implementing Security using JAAS and Java GSS API

Charlie Lai, Seema Malkani
Java Security Engineers
Sun Microsystems, Inc.

Overall Presentation Goal

Learn how to implement security using
JAAS and Java GSS API

Speaker's Qualifications

- Speakers are members of the J2SE Security Engineering Team at Sun Microsystems, Inc.

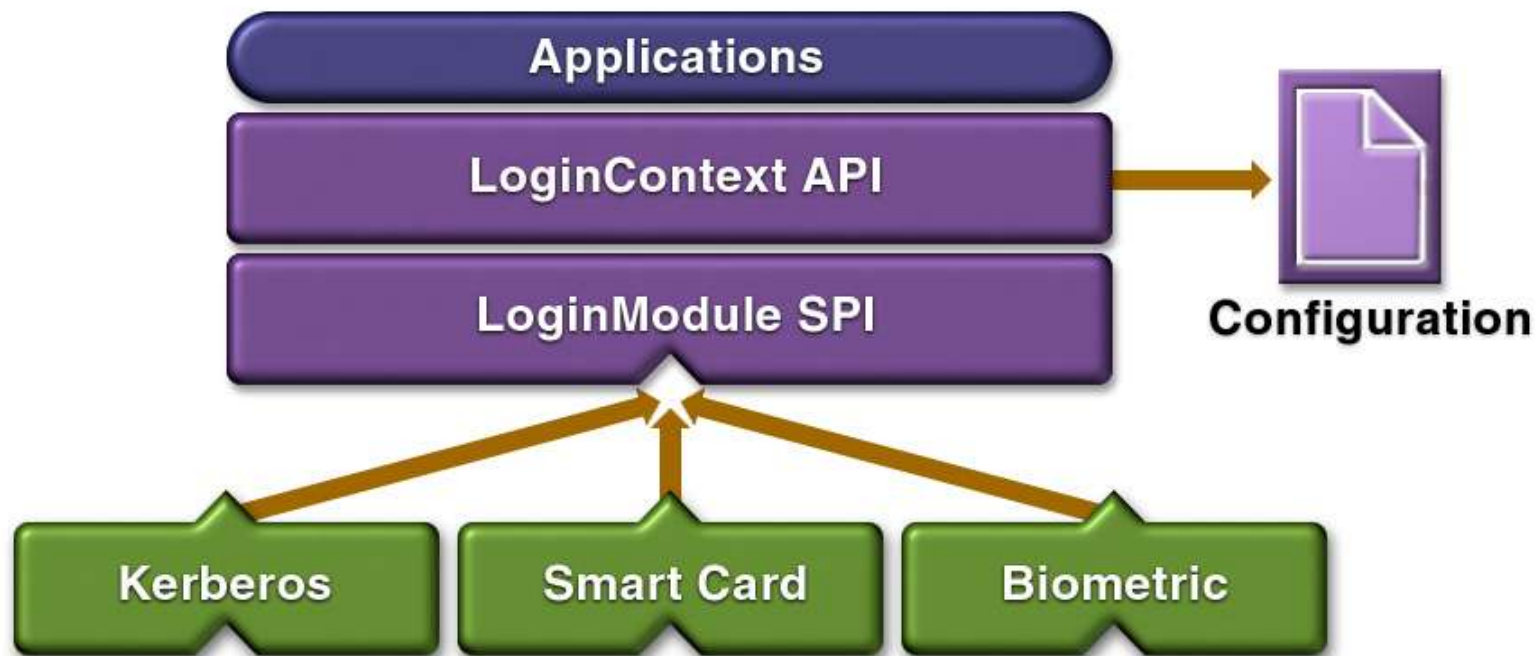
Presentation Agenda

- JAAS Authentication
- Kerberos Authentication via JAAS
- Secure Communication using Java GSS API
- Single Sign-On using JAAS, Java GSS, and Kerberos
- Future Directions

JAAS Authentication



JAAS Pluggable Authentication



Key Classes

- `javax.security.auth.login` Package
 - `LoginContext`
 - `Configuration`
- `javax.security.auth.spi` Package
 - `LoginModule`
- `javax.security.auth` Package
 - `Subject`

Sample Code

```
// create LoginContext and authenticate Subject  
LoginContext context = new LoginContext(name, ..);  
context.login();
```

```
// perform action as authenticated Subject  
Subject s = context.getSubject();  
Subject.doAs(s, action);
```

```
// Logout  
context.logout();
```

Kerberos Authentication via JAAS



Kerberos Overview

- Trusted third party authentication system
- Based on shared keys
- Defines a protocol for authentication and secure communication (RFC 1510)
- Uses concept of tickets
- Available on Solaris since Solaris 2.6
- Main authentication system for Windows 2000, Windows XP, Windows .NET

Kerberos Authentication via JAAS

- Application specifies “My_Name”

```
// create LoginContext and perform authentication
LoginContext context = new LoginContext("My_Name", ...);
context.login();
```

- Krb5LoginModule Configuration

```
My_Name {
    com.sun.security.auth.module.Krb5LoginModule required;
};
```

Krb5LoginModule Configuration Example

- Get ClientUser's Ticket Granting Ticket from the native credentials cache

```
Client {  
    com.sun.security.auth.module.Krb5LoginModule required  
        useTicketCache=true  
        ticketCache=/tmp/client_cache  
        principal="ClientUser" };
```

- Get ServerUser's key from the native keytab

```
Server {  
    com.sun.security.auth.module.Krb5LoginModule required  
        useKeyTab=true  
        keyTab=/etc/keytab  
        principal="ServerUser" };
```

Java GSS API



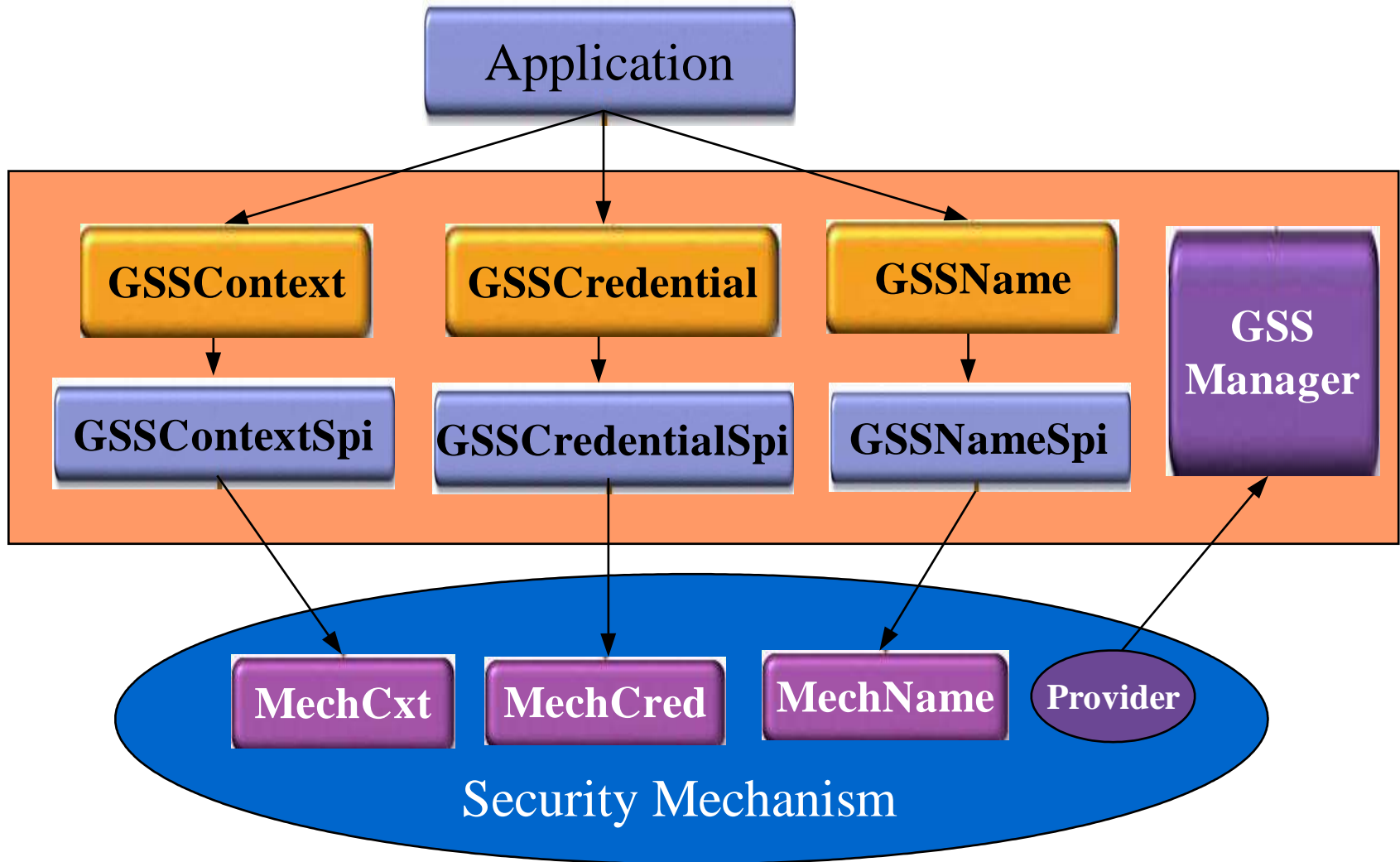
What is GSS-API ?

- Provides a generic authentication and secure messaging interface (RFC 2743)
- Different security mechanisms can be plugged-in : public key-based or secret-key
- Security context is established between peers
- Integrity and privacy per-message basis
- Exchange of GSS-API tokens
 - Transport is the responsibility of application
- Applications that use GSS-API e.g. NFS

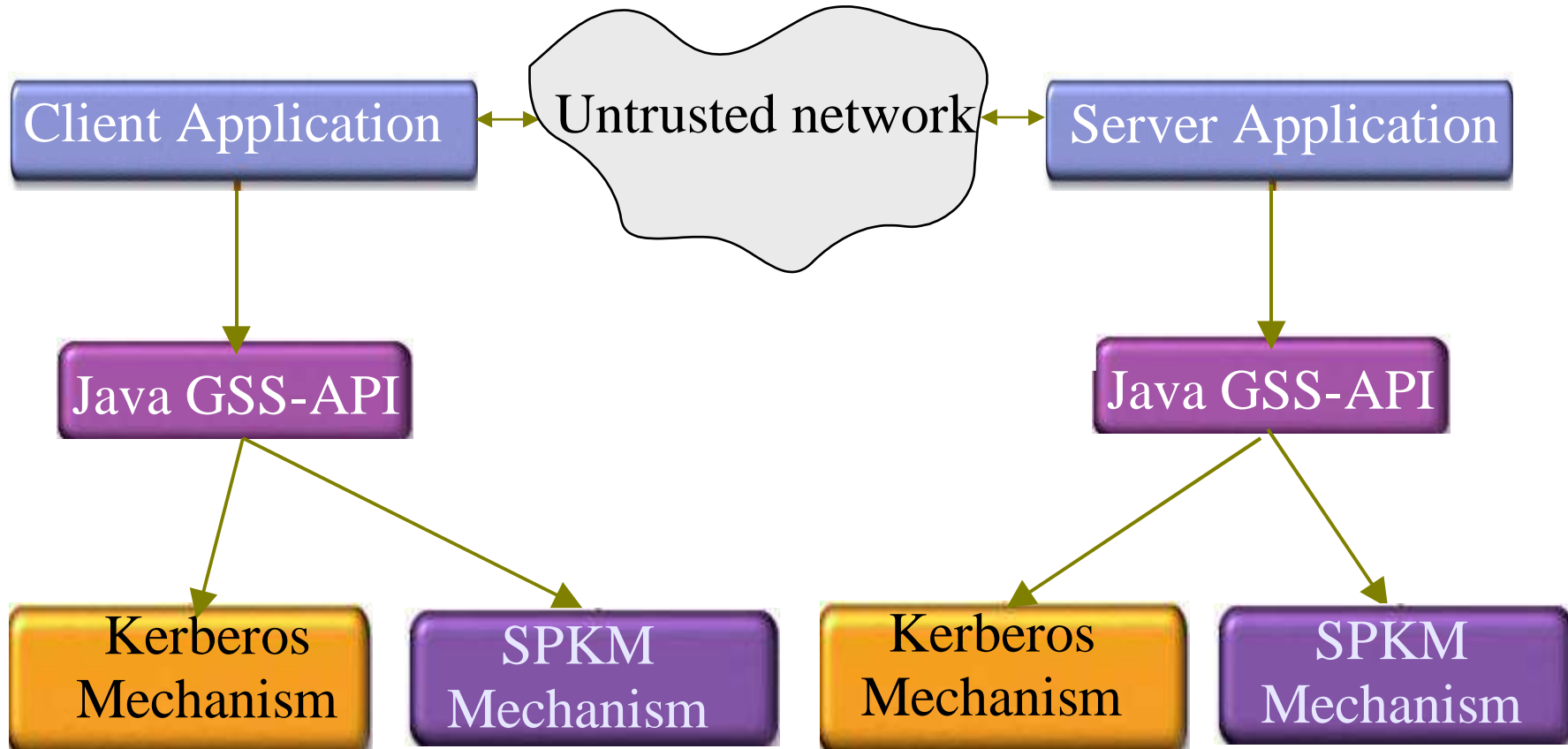
Java GSS Overview

- Java bindings for GSS-API
 - Standards : RFC 2853, JSR 072
- Java GSS-API framework (org.ietf.jgss)
 - *GSSManager factory*
 - *GSSName interface*
 - *GSSCredential interface*
 - *GSSContext interface*
- Mutual authentication, credential delegation
- Default security mechanism is Kerberos V5
- Single sign-on using JAAS

Java GSS Framework



Java GSS-API Network Authentication



Secure Communication using Java GSS API: Sample Client

```
class ClientAction implements PrivilegedAction {
    public Object run() {
        // initiate GSS security context to the peer
        GSSManager manager = GSSManager.getInstance();
        GSSName serverName = manager.createName(server, null);
        GSSContext secContext = manager.createContext(serverName,
                                                    mechOid, ...);

        // set desired optional features on the context
        secContext.requestConf(true);
        byte[] inToken = new byte[0];
        while (!secContext.isEstablished()) {
            outToken = secContext.initSecContext(inToken, ...);
            sendToken(server, outToken);
            if (!secContext.isEstablished()) {
                inToken = readToken(server);
            }
        }

        // call wrap to encrypt data
        outToken = secContext.wrap(data, ...);
    }
}
```

Secure Communication using Java GSS API: Sample Server

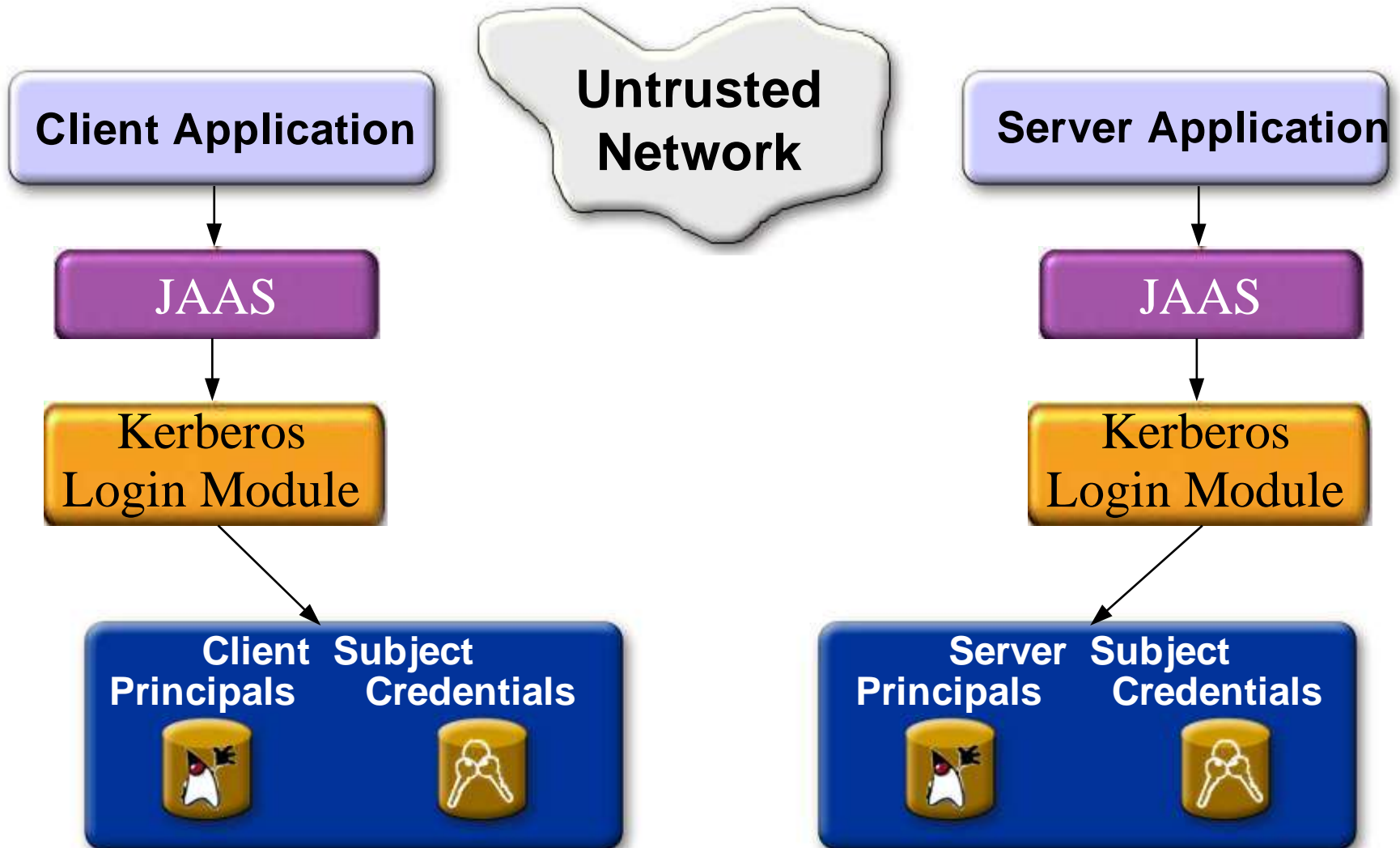
```
class ServerAction implements PrivilegedAction {
    public Object run() {
        GSSManager manager = GSSManager.getInstance();
        GSSContext secContext = manager.createContext(...);

        while (!secContext.isEstablished()) {
            inToken = readToken(client);
            outToken = secContext.acceptSecContext(inToken...);
            sendToken(client, outToken);
        }
        inToken = readToken(client);
        // call unwrap to decrypt data
        byte [] bytes = secContext.unwrap(inToken, ...);
        ...
    }
}
```

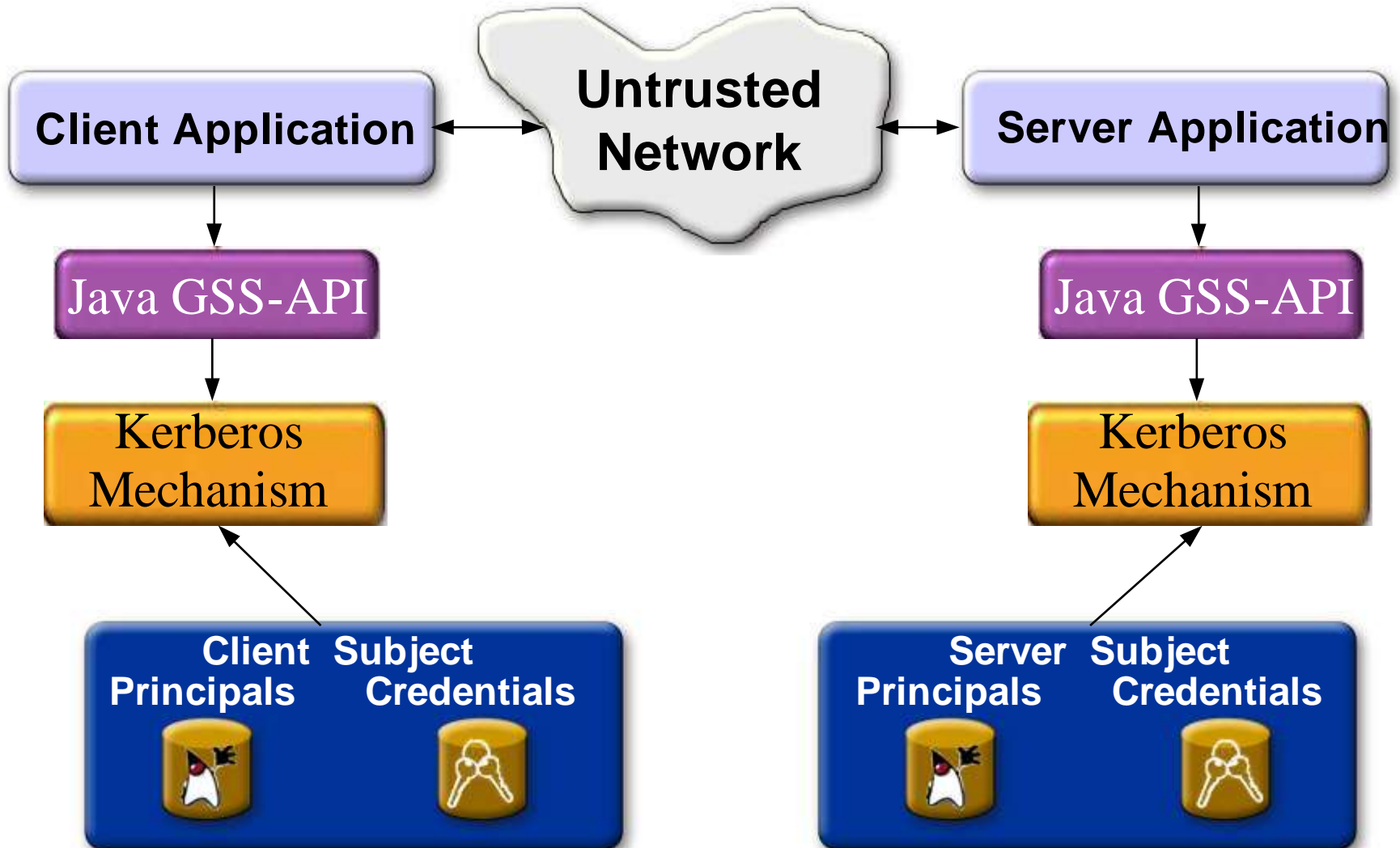
Single Sign-On using JAAS, Java GSS, and Kerberos



Kerberos Authentication via JAAS



Single Sign-On Using Kerberos



What's new in Java GSS ? (J2SE 1.4.2)

- Configurable Kerberos Settings
 - *RefreshKrb5Config* boolean flag in Krb5LoginModule configuration
- Support for Slave Kerberos Key Distribution Center (KDC)
- Support TCP for Kerberos Key Distribution Center Transport
- Kerberos Service Ticket in the Subject's Private Credentials

Beyond J2SE 1.5

- JAAS
 - JSR 196 (Authentication Service Provider Interface for J2EE Containers)
- Java GSS-API
 - Pluggability (Public Service Provider Interface)
 - SPNego and PKInit mechanisms
 - SPKM and LipKey mechanisms

Summary

- Use of JAAS for Authentication
- Kerberos Authentication using JAAS
- Use of Java GSS-API to secure communication
- Single Sign-On using JAAS, Java GSS, and Kerberos
- Future Directions

More Information

- Web sites
 - java.sun.com/security
 - java.sun.com/j2se/1.4.2/docs/guide/security
- Contacts
 - java-security@sun.com for feedback
 - www.sun.com/developer/support for support

Q&A

JAVA™



JavaOneSM

Sun's 2003 Worldwide Java Developer Conference[®]

JAVATM

java.sun.com/javaone/sf