Developing Secure Web Applications for Constrained Devices

Dr. Vipul Gupta

Sr. Staff Engineer, Sun Labs

Sun Microsystems, Inc.

11th Int'l World Wide Web Conference, Hawaii
May 7-11, 2002
Developers Day

Sun microsystems
We make the net work.

What is Security?

"Preventing adverse business consequences from the illegitimate actions of human beings."

Whitfield Diffie,
Chief Security Officer, Sun Microsystems

- Availability, reliability, robustness
- Integrity, authenticity, access control
- Secrecy, confidentiality, privacy

Impact of Device Constraints

- Possible constraints: CPU, memory, power, screen size, network bandwidth and latency
- Influence choice of algorithms and protocols, feature richness, user interaction (can impact security!)
- Workarounds: offload computation, new algorithms, feature profiling, caching/reuse, other clever implementation optimizations

J2ME™ Market Penetration

- 10 million Java enabled (J2ME) phones in Japan. Nextel sold 1.3 million Java phones in North America. Nokia expects to sell 50 million Java phones in 2002.
- J2ME will be leveraged by 70% of the smart phones PDAs by 2004 (Gartner group, Nov. 2000)
- 680 million Java enabled mobile devices worldwide by 2004 (ARC Group, June 2000)

Java™ 2 Platform Overview

Java 2 Enterprise Edition (J2EE™)

Java 2 Standard Edition (J2SE™)

Java 2 Micro Edition (J2ME™)

Java Card

Java Platform

HotSpot™ | CVM | KVM | Card VM

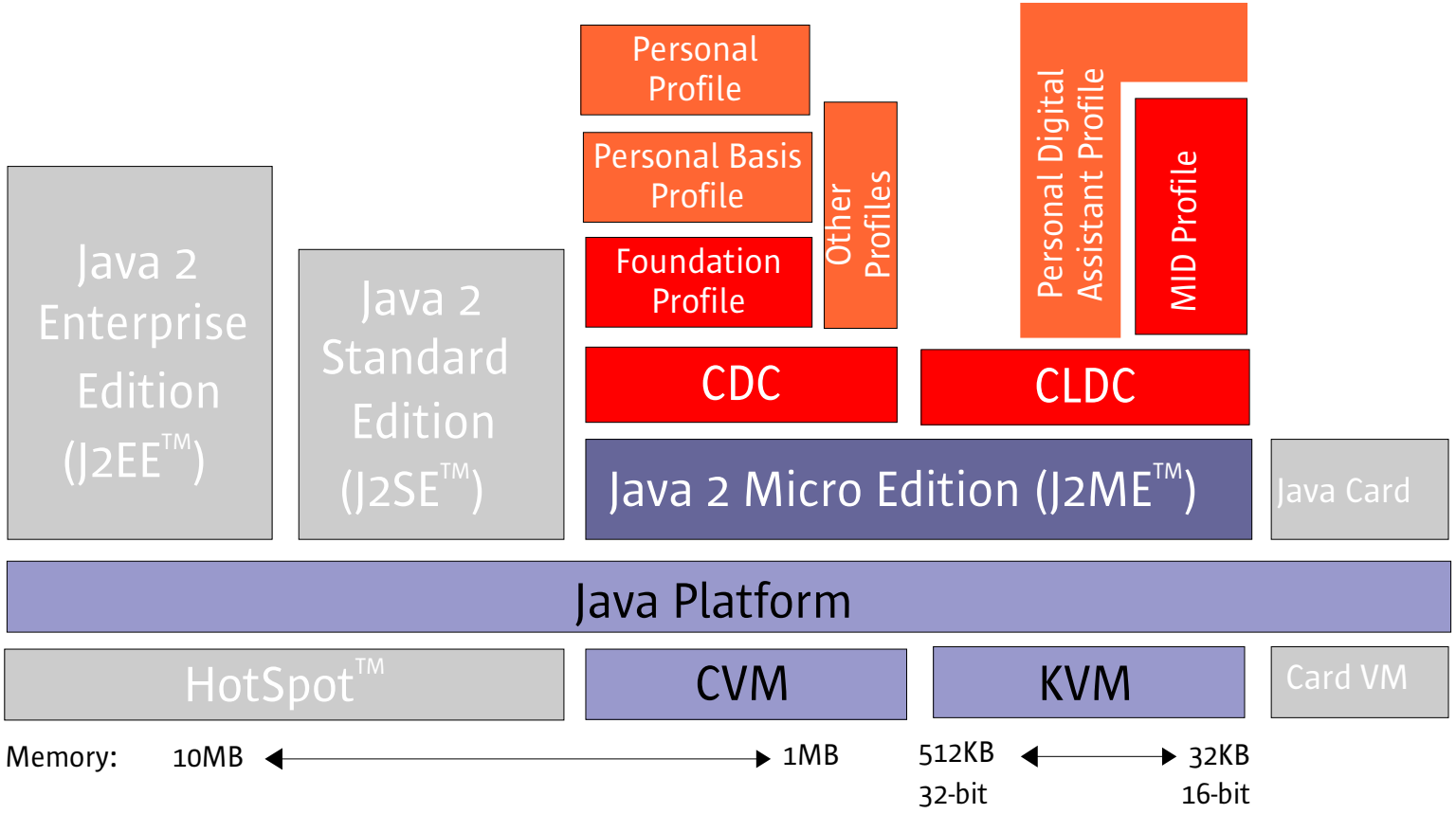
Memory: 10MB ◄——► 1MB | 512KB ◄——► 32KB

64-bit | 32-bit | 16-bit | 8-bit

Java™ 2 Platform Micro Edition

- *The* standard for development on a wide range of consumer electronics and embedded devices
- Consists of VMs and core APIs ("Configurations") plus application- and market-specific APIs ("Profiles") defined through Java Community Process (JCP℠)

J2ME™ Technology Overview



CLDC: Connected Limited Device Configuration
MID: Mobile Information Device

CDC: Connected Device Configuration

J2ME™ Availability

Technology	Type	Spec	Stage
CLDC 1.0	J2ME Config.	JSR-30	Final
MIDP 1.0	CLDC Profile	JSR-37	Final
CDC 1.0	J2ME Config.	JSR-36	Final
Foundation	CDC Profile	JSR-46	Final
RMI	CDC Profile	JSR-66	Proposed final
CLDC 1.1	J2ME Config.	JSR-139	Public review
MIDP 2.0	CLCD Profile	JSR-118	Public review
Personal Basis	CDC Profile	JSR-129	Public review
Personal	CDC Profile	JSR-62	Public review
PDAP	CLDC Profile	JSR-75	Community review

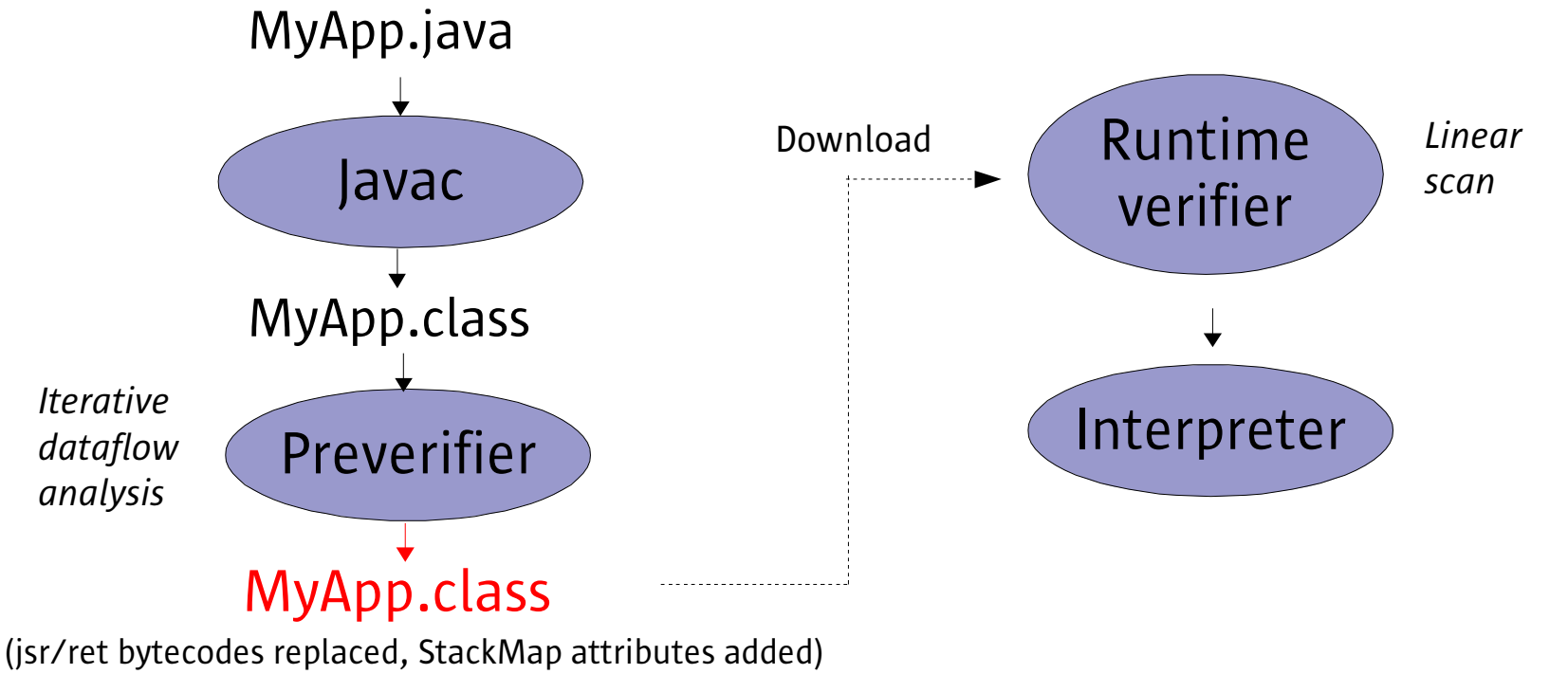
CLDC Security

- Low-level virtual machine security
  - Ensures applications cannot harm device
  - Classfile verification rejects invalid classfiles
- Application-level security
  - Controls access to external resources, e.g. files
  - Simple sandbox model
- End-to-end security mechanisms are out of scope (left for Profiles)

CLDC Classfile Verification

Development Workstation

Target device



(jsr/ret bytecodes replaced, StackMap attributes added)

CLDC Sandbox Model

- Requires classfiles properly verified and guaranteed to be valid Java applications.
- Programmer cannot override, modify or add system classes or modify/bypass VM's standard class loading mechanism.
- Only a limited, predefined set of APIs available to application programmer.
- Set of native functions is closed (no JNI).

CLDC Generic Connection

- Framework provides coherent, flexible, extensible support of different networking and I/O protocols. Follows URI syntax (RFC2396)
  - Connector.open("<protocol>:<address>;<parameters>");
- Profiles determine supported protocols:
  - Connector.open(file:///CFCard/newfile.txt");
  - Connector.open("http://www.sun.com");
  - Connector.open("socket://128.226.3.29:8080");
  - Connector.open("comm:COM1;baudrate=9600");

MIDPv1.0 Security Features

- No additional low-level or application-level security features beyond CLDC.
- No end-to-end security mechanism mandated -- but MIDP Reference Implementation (RI) 1.0.3 from Sun provides https support using KSSL*.

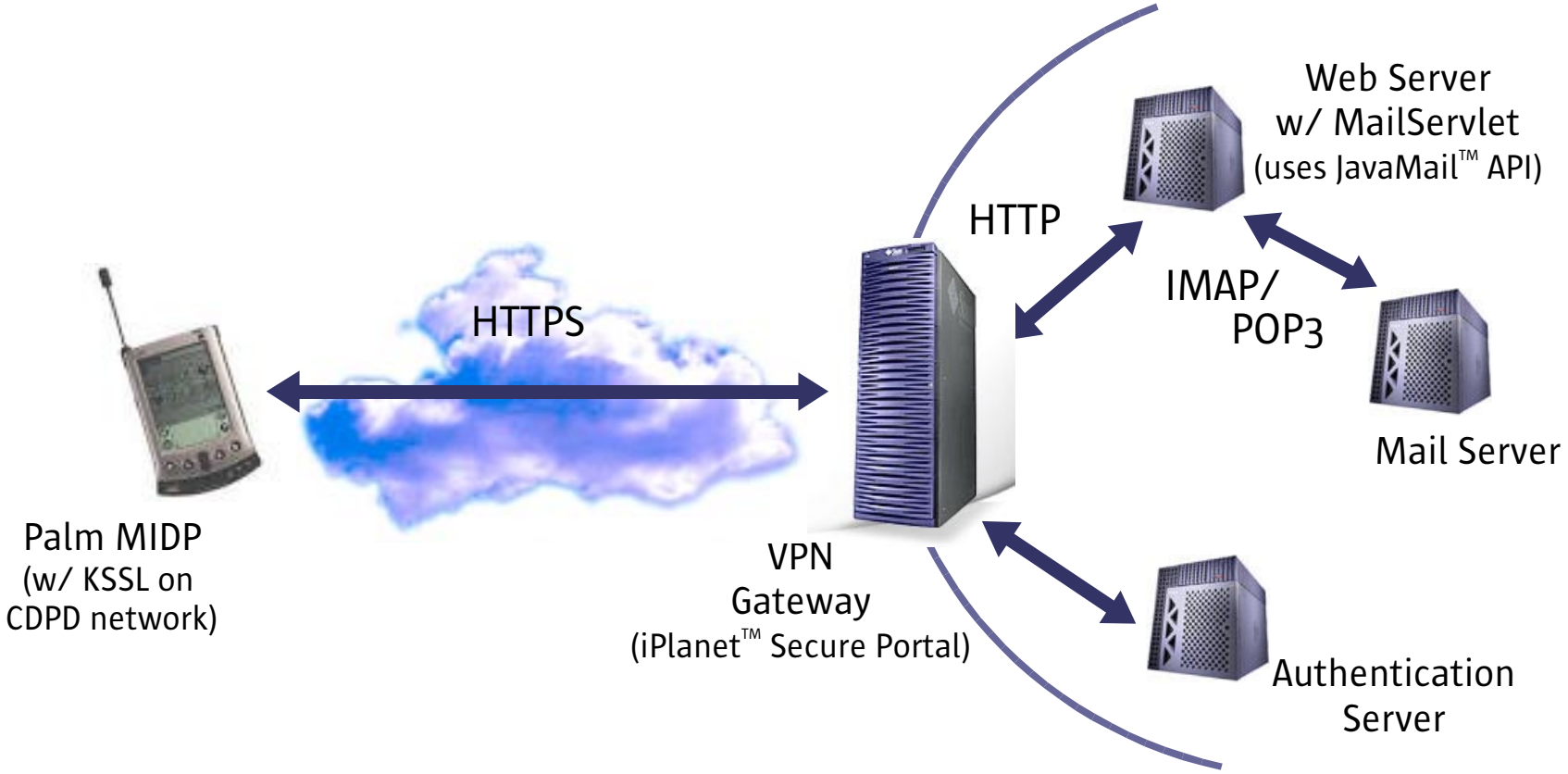
*Sun Labs research project, see http://playground.sun.com/~vgupta/KSSL

MEKeyTool

- Manipulates the set of trusted root certificates in MIDP-RI from Sun
- Based on J2SE™ keytool and described in tools/com/sun/midp/mekeytool/package.html
- Example usage: list trusted root certificates

```
$ java –classpath bin/MEKeyTool.jar
 com.sun.midp.mekeytool.MEKeyTool
–list –MEkeystore appdb/_main.ks
```

Example Application

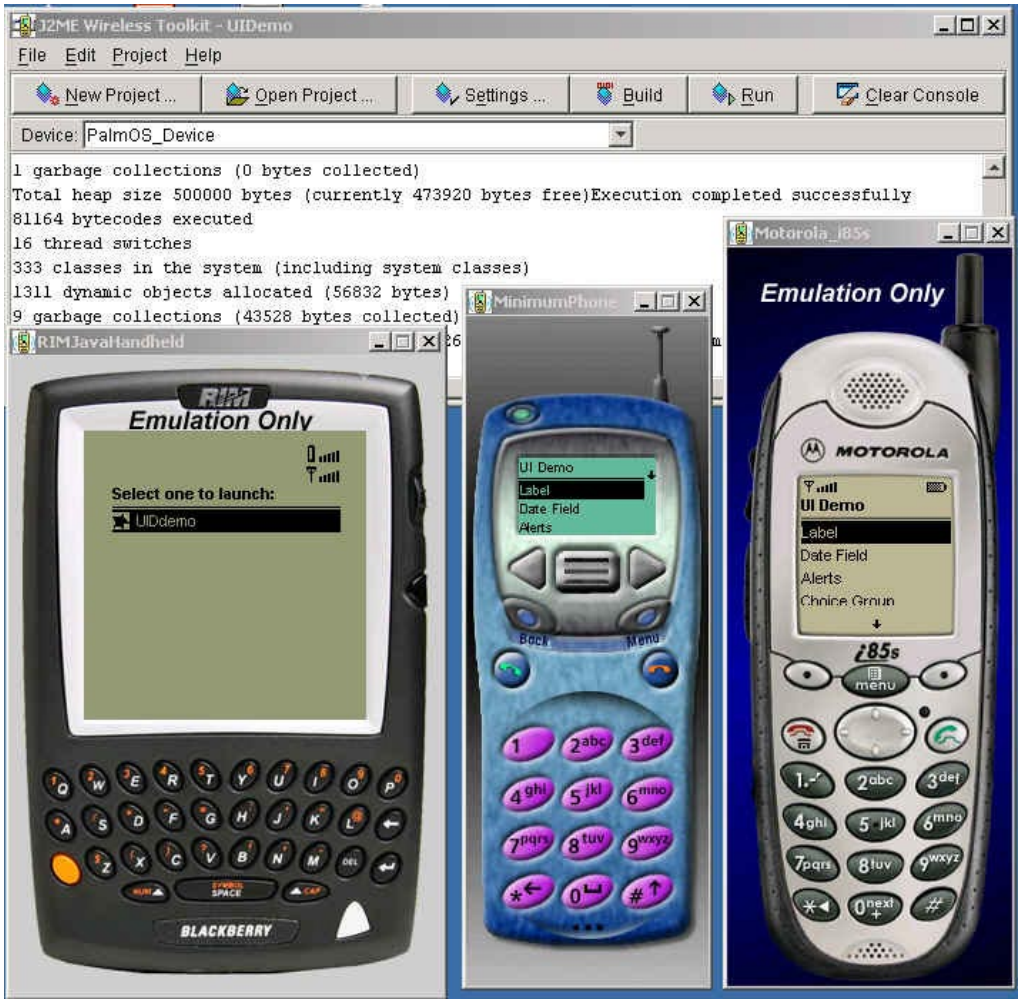


- Secure access to enterprise email.

MIDlet Development Steps

- Write your J2ME application
- Compile it
- Preverify it
- Package it into a JAR file
- Create the application descriptor (JAD)
- Deploy and run on emulator or device

J2ME™ Wireless Toolkit



- Comprehensive development environment -- standalone or with Forte™ for Java.
- Free download!
- Supports multiple device skins
- Integrable with third party IDEs

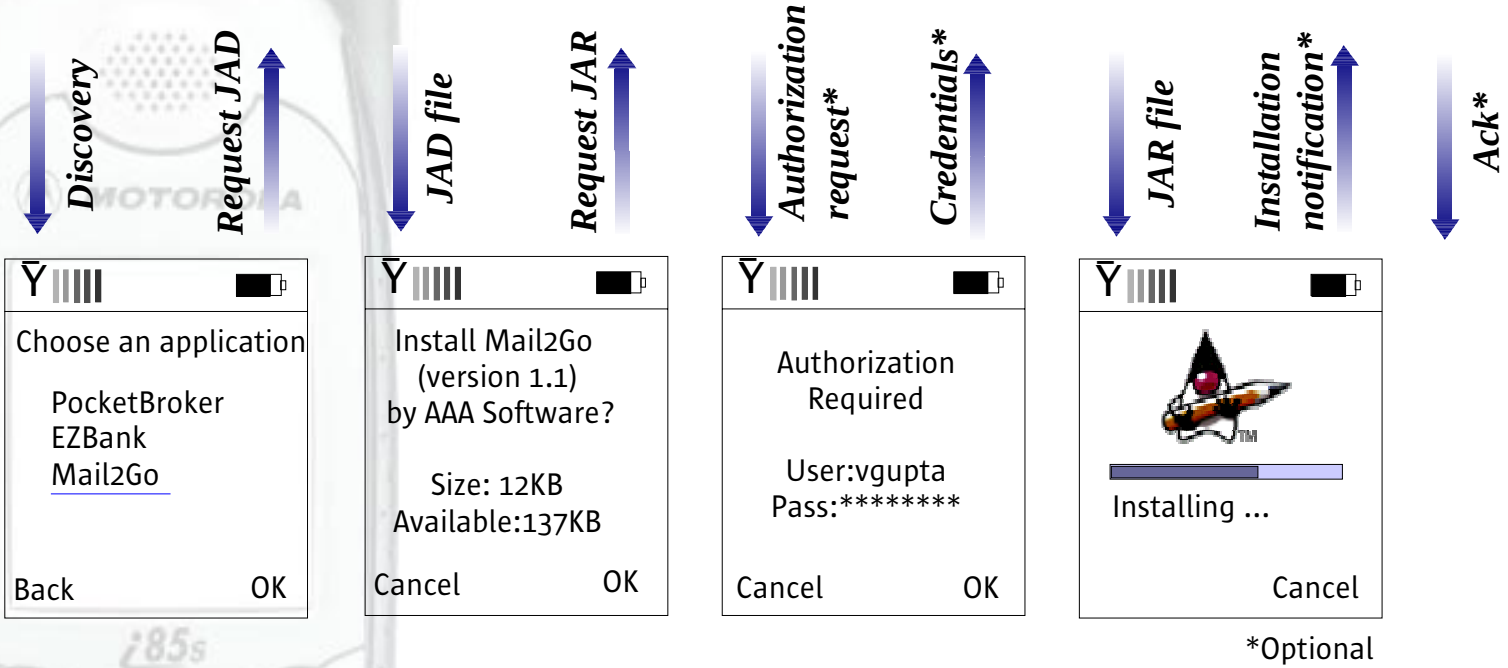
Sample JAD file

MIDlet-Name*: SunNet
MIDlet-Version*: 1.4
MIDlet-Vendor*: Sun Labs
MIDlet-Description: Mobile access to enterprise applications
MicroEdition-Profile: MIDP-1.0
MicroEdition-Configuration: CLDC-1.0
MIDlet-1: Configure,,net.sun.Configure
MIDlet-2: Login,,net.sun.Login
MIDlet-3: Nametool,,net.sun.Nametool
MIDlet-4: Caltool,,net.sun.Caltool
MIDlet-5: Mailtool,,net.sun.Mailtool
MIDlet-6: Logout,,net.sun.Logout
MIDlet-Jar-URL*: http:// ... /sunnet.jar
MIDlet-Jar-Size*: 38149
MIDlet-Data-Size: 1024

**Required*

- Used by App. Mgmt Software (AMS)
- Same format as JAR manifest.
- MIME type:
  text/vnd.sun.j2me.app-descriptor

Over The Air Provisioning



*Optional

- Application Management Software (AMS) uses HTTP 1.1 functionality for network interaction. Session management addresses privacy concerns.
- MIDlet suite is the unit of installation or deletion. Notification supported for both operations. Installation or update is “atomic”.

MIDPv2.0 Security Features

- Extends Sandbox model: trusted MIDlets can access restricted APIs
- Provides mechanisms for secure network communication -- ensuring source authentication, integrity and confidentiality of data

Signed MIDlets

- New attributes for JAD file, both use Base64 encoded values:
  - MIDlet-Certificate-*n*-*m*: *certificate*
  - MIDlet-Jar-RSA-SHA1: *signature*
- Device verifies signer certificate using configured root certificates and validates signature on JAR file.

Signature	*Verification*	*Action*
Absent	–	May install as untrusted
Present	Fails	Do not install
Present	Succeeds	Install as trusted

MIDlet Access Control

- Untrusted MIDlet suites
  - No access to protected functions or through explicit user permission
- Trusted MIDlet suites
  - Request named permissions with:
    - MIDlet-Permissions (critical)
    - MIDlet-Permissions-Opt (non-critical)
  - Device enforces access control in accordance with policy

Permissions

- Named by restricted APIs, names are hierarchical, share package name with API
- Granted with *User* input or *Allowed* unconditionally

Permission	*Protocol*
javax.microedition.io.Connector.http	Http
javax.microedition.io.Connector.https	Https
javax.microedition.io.Connector.datagram	Datagram
javax.microedition.io.Connector.datagramreceiver	Datagram server
javax.microedition.io.Connector.socket	Socket
javax.microedition.io.Connector.serversocket	Server socket
javax.microedition.io.Connector.comm	Serial port
javax.microedition.io.PushRegistry.datagram	Datagram
javax.microedition.io.PushRegistry.socket	Socket

User Permissions

- Requires explicit user approval
- Grant types
  - Blanket: valid until revoked or MIDlet deletion
  - Session: valid until MIDlet suite terminates
  - Oneshot: valid for single invocation of restricted method

Example Policy

domain: operator
grant: javax.microedition.io.Connector.socket
user: javax.microedition.io.Connector.comm

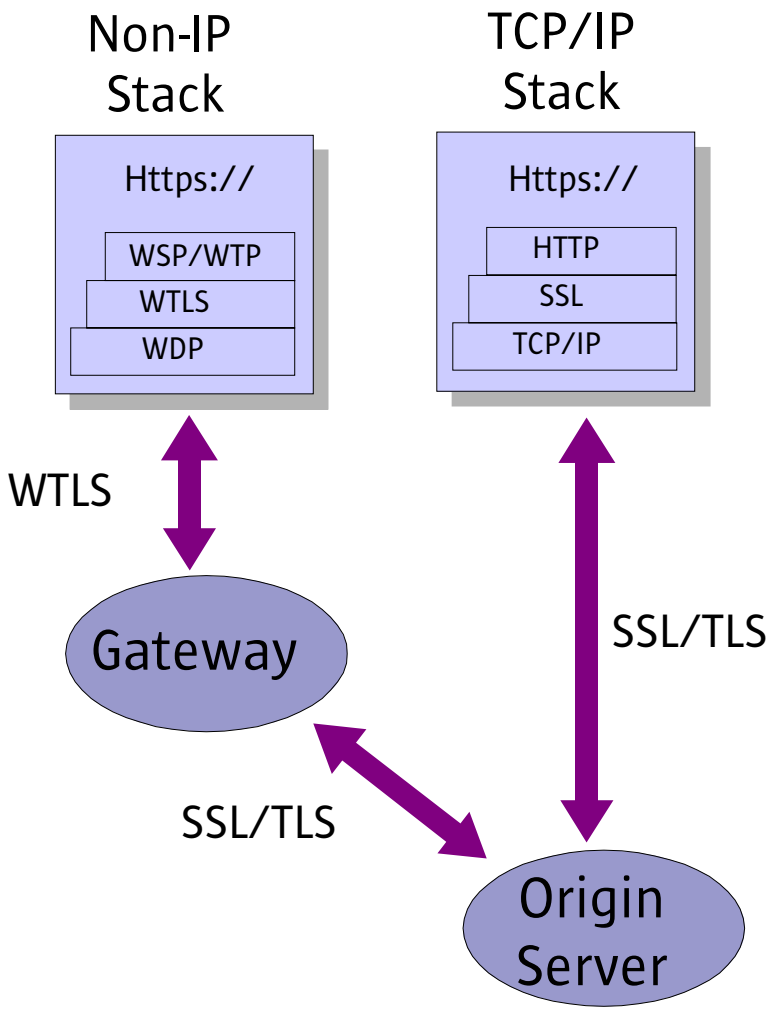
alias: lots_of_stuff
permissions: java.microedition.io.Connector.socket,
javax.microedition.io.PushRegistry

domain: manufacturer
grant: lots_of_stuff

Secure Networking

- HTTPSConnection
  - Secure HTTP over SSLv3.0 or TLSv1.0 or WTLS
- SecureConnection*
  - Secure socket connection using SSLv3.0 or TLSv1.0
- SecurityInfo
  - Access to server certificate and other secure connection parameters, e.g. Cipher suite, protocol name and version
- Certificate
  - Access to server identity and other certificate parameters
- CertificateException
  - Exceptions for certificates and connection setup errors

HTTPS & Secure Connection



- Like HTTP, HTTPS may or may not use IP-based transport.
- SecureConnection is truly end-to-end.
- Other application protocols (e.g. IMAP) can be layered atop SecureConnection

Security and Trust Services API for J2ME™

- Brand new JSR-177 for integration of a “security element”, e.g. smart cards
  - Secure storage for private keys, root certificates, personal information etc.
  - Cryptographic operations
  - Support services such as user authentication, banking, payment, digital rights management
  - Expected standardization: first half of 2003

Developer Resources

- Wireless Developer Homepage
  - http://wireless.java.sun.com/
- J2ME™
  - Homepage (http://java.sun.com/j2me)
  - Wireless Toolkit (http://java.sun.com/products/j2mewtoolkit)
  - Bill Day's J2ME Archive (http://www.billday.com/j2me)
- Java Community Process (JCP℠) Homepage
  - http://www.jcp.org/

The Takeaway

J2ME™ technology offers a comprehensive framework for developing and deploying secure applications on constrained devices.

Vipul Gupta

vipul.gupta@sun.com

http://java.sun.com/

Sun microsystems

We make the net work.

J2EE™ Client Provisioning

- The J2EE Client Provisioning Specification (JSR-124) will provide a standard for partitioning server applications that provision client applications.
- The expert group is studying schemes used to provision content built for: J2ME MIDP, NTT DoCoMo iAppli, etc

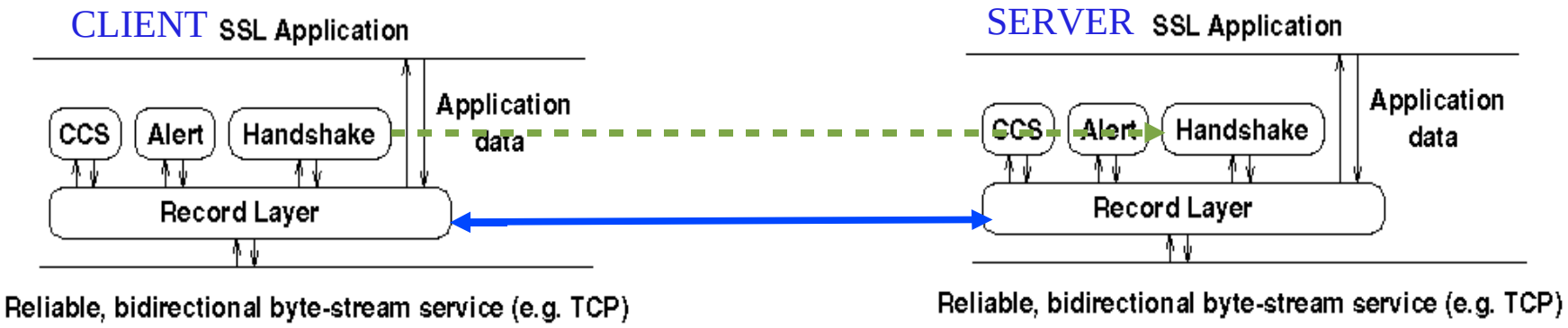
Crypto Building Blocks

- Shared Secret (aka symmetric-key) Cryptography: One key, two operations inverses of each other (RC4, 3DES)
  - Fast, Requires pre-determined shared secret key
- Public Key (aka asymmetric-key) Cryptography: One operation, two mathematically inverse keys (RSA, DH, DSA)
  - Only private key needs to be kept secret, public key only needs to be authenticated
  - Expensive, Often used to distribute secret keys

Crypto Building Blocks – cont'd

- Message Digest/Hash function: one-way, collision resistant function that produces a unique fingerprint of input, e.g. MD5, SHA. Typically very fast.
- Keyed Hash: hash function with a secret key as additional input produces Message Authentication Code (MAC). MAC provides source authentication and bulk-data integrity.
- Certificate: signed document attesting subject's ownership of public key, popular format X.509
  - Certificate Authority (CA): verifies claimed identity & proof of possession of private key, issues certificates

SSL Overview



- Client/Server negotiate ciphersuite (Key exchange, MAC, and Symmetric Key algorithm). Eg: RSA_WITH_RC4_128_MD5
- Handshake Protocol uses (expensive) Public Key Crypto Algorithms to establish a shared secret
- Record Layer uses shared secret to perform MAC and (cheaper) Symmetric Key encryption of bulk data

SSL Handshake (RSA)