# A Blockchain-based Security Management Framework for Cyber-Physical Systems

**ABSTRACT:**

 The Single sign-on (SSO) is a new authentication mechanism that enables a legal user with a single credential to be authenticated by multiple service providers in a distributed computer network. Recently, Chang and Lee proposed a new SSO scheme and claimed its security by providing well-organized security arguments. In this paper, however, we demonstrative that their scheme is actually insecure as it fails to meet credential privacy and soundness of authentication. Specifically, we present two impersonation attacks. The first attack allows a malicious service provider, who has successfully communicated with a legal user twice, to recover the user’s credential and then to impersonate the user to access resources and services offered by other service providers. In another attack, an outsider without any credential may be able to enjoy network services freely by impersonating any legal user or a nonexistent user. We identify the flaws in their security arguments to explain why attacks are possible against their SSO scheme. Our attacks also apply to another SSO scheme proposed by Hsu and Chuang, which inspired the design of the Chang–Lee scheme. Moreover, by employing an efficient verifiable encryption of RSA signatures proposed by Ateniese, we propose an improvement for repairing the Chang–Lee scheme. We promote the formal study of the soundness of authentication as one open problem.

**Objective:**

Our Aim SSO scheme should meet three basic security requirements, i.e., *unforgeability*, *credential privacy*, and *soundness*. Efficient verifiable encryption of RSA signatures proposed by Ateniese, we propose an improvement for repairing the Chang–Lee scheme. We promote the formal study of the soundness of authentication as one open problem.

**Problem Definition:**

1.Allows a malicious service provider, who has successfully communicated with a legal user twice, to recover the user’s credential and then to impersonate the user to access resources and services offered by other service providers.

2.insecure against both impersonation attacks and identity disclosure attacks

3.scheme suffers from Deniable of Service (DoS) attacks and presented a new scheme

4.one user to maintain distinct pairs of identity and password for different service providers, since this could increase the workload of both users and service providers as well as the communication overhead of networks

**Literature Survey**

1.Mitigating Distributed Denial of Service Attacks in Multiparty Applications in the Presence of Clock Drifts

There are several types of such attacks. An attacker can possibly launch a DoS attack by studying the flaws of network protocols or applications and then sending malformed packets which might cause the corresponding protocols or applications getting into a faulty state. In order to deplete the victim’s key resources (such as bandwidth and CPU time), the attacker has to aggregate a big volume of malicious traffic. Most of the time, the attacker collects many (could be millions) of zombie machines or bots to flood packets simultaneously, which forms a Distributed Denial of Service(DDoS) attack. the attacker may change its strategy and attack an application directly, especially when the application involves complex computations. It could be easier to exhaust its computational resources with small volume of messages. Therefore, the malicious traffic against an application has usually small volume and it is difficult to be detected The method does not induce any need for group synchronization which would have raised scalability issues, but instead employs a simple interface of the server with each client. The options for the adversary to launch a directed attack to the application’s ports after eavesdropping is minimal, since the port hopping period of the protocol is fixed. Here is that the adaptive method can work under timing uncertainty and specifically fixed clock drifts

2.An Efficient Password Authentication Scheme for Smart Card

This will reduce the server over head of maintaining large user data for authentication. Remote user authentication using smart cards is a good solution for many e-based applications. Smart card implementation ensures secure communications. Several schemes using timestamp for remote authentication have already been proposed. However these are vulnerable to certain types of forgery attack. To access resources at remote system, users should have proper access rights. One of the simplest and efficient mechanisms is the use of a password authentication scheme. To access the resources, each user should have an identity (ID) and a password (PW). In the existing traditional set up the ID and PW are maintained by the remote system in a verification table. If a user wants to login to a remote server, he has to submit his ID and password PW to the server. The remote server receives the login message and checks the authenticity of the user by referencing the verification table. We propose an efficient password authentication scheme using smart card. The proposed scheme restricts most of the well-known attacks with reasonable computational cost. The proposed scheme is based RSA. The server need not maintain password table, instead it maintains only registration time.

3.oPass: A User Authentication Protocol Resistant to Password Stealing and Password Reuse Attacks

People select their username and text passwords when registering accounts on a website. In order to log into the website successfully, users must recall the selected passwords. Generally, password-based user authentication can resist brute force and dictionary attacks if users select strong passwords to provide sufficient entropy. However, password-based user authentication has a major problem that humans are not experts in memorizing text strings. Thus, most users would choose easy-to-remember passwords (i.e., weak passwords) even if they know the passwords might be unsafe. Another crucial problem is that users tend to reuse passwords across various websites a user authentication protocol named oPass which leverages a user’s cellphone and short message service (SMS) to prevent password stealing and password reuse attacks. In our opinion, it is difficult to thwart password reuse attacks from any scheme where the users have to remember something. We also state that the main cause of stealing password attacks is when users type passwords to untrusted public computers. Therefore, the main concept of oPass is free users from having to remember or type any passwords into conventional computers for authentication. Unlike generic user authentication, oPass involves a new component, the cellphone, which is used to generate one-time passwords and a new communication channel, SMS, which is used to transmit authentication messages.

4. **An Improved Secure Anonymous Protocol for Distributed Computer Networks**

Computer networks connect hosts and users into a distributed computing environment which provides the advantages of increasing reliability, sharing information and computing power. Usually, the process of authentication involves the exchange of identities and authenticated key generation. It is increasingly important to protect systems and user privacy and provide security from malicious adversaries. In distributed computing environments, it may be advantageous to maintain user anonymity. That is, only the service provider can identify the user, while all other entities cannot. showed a new weakness in Wu and Hsu’s protocol where a service provider could obtain a valid user’s secret token after an exchange of messages. As such, Yang et al. proposed a protocol to overcomes the weakness of Wu and Hsu’s protocol to achieve user anonymity, user identification and key agreement. pointed out that protocol possessed a Denial-of Service (DoS) vulnerability. At the same time, Mangipudi and Katti proposed a secure identification and key agreement protocol with user anonymity Hsu and Chuang demonstrated an identity disclosure attack on the scheme to show the identity of the communicating user can be easily ascertained from the exchanged messages.

5.Zero Knowledge Proofs of Identity

An Identification scheme is a protocol which enables party ,4 to prove his identity polynomially many times to party B without enabling B to misrepresent himself as A to someone else. Identification schemes are related to the notion of digital signatures, but there are no messages judges and disputes: The proof of identity is either accepted or rejected in real time, and as a result the requested access or service is granted or withheld. This is one of the fundamental problems in cryptography, and it has numerous practical applications. The basic problem with most of the current identification techniques (ID cards, credit cards, computer passwords, PIN numbers, etc) is that A proves his identity by revealing a constant S in the form of a printed card or a memorized value. A sophisticated adversary who cooperates with a dishonest verifier B can use a Xerox copy of the card or a recording of the secret value to misrepresent himself successfully as A at a later stage. The obvious solution is to use zero knowledge proofs of knowledge, which convince B that A knows S without revealing even a single bit of information about A’s key. Our goal in the next two sections is to develop a truly practical scheme, which can be implemented in software in a fraction of a second even on the weak microprocessors embedded in smart cards. For reasonable choices of the parameters, our scheme is about two ordiers of magnitude faster than RSA-based identification sclhemes.

**EXISTING SYSTEM:**

The other side, it is usually not practical by asking one user to maintain distinct pairs of identity and password for different service providers, since this could increase the workload of both users and service providers as well as the communication overhead of networks. That, after obtaining a credential from a trusted authority for a short period each legal user’s authentication agent can use this single credential to complete authentication on behalf of the user and then access multiple service providers. Intuitively, an SSO scheme should meet at least three basic security requirements, enforceability, credential privacy, and soundness. Enforceability demands that, except the trusted authority, even a collusion of users and service providers are not able to forge a valid credential for a new user. Credential privacy guarantees that colluded dishonest service providers should not be able to fully recover a user’s credential and then impersonate the user to log in to other service providers. Soundness means that an unregistered user without a credential should not be able to access the services offered by service providers.

**DISADVANTAGES OF EXISTING SYSTEM:**

* Actually an SSO scheme, has two weaknesses an outsider can forge a valid credential by mounting a credential forging attack since the scheme employed naïve RSA signature without using any hash function to issue a credential for any random identity.
* Their scheme is suitable for mobile devices due to its high efficiency in computation and communication.

**PROPOSED SYSTEM**

The first attack, the “credential recovering attack” compromises the credential privacy in the scheme as a malicious service provider is able to recover the credential of a legal user. The other attack, an “impersonation attack without credentials,” demonstrates how an outside attacker may be able to freely make use of resources and services offered by service providers, since the attacker can successfully impersonate a legal user without holding a valid credential and thus violate the requirement of soundness for an SSO scheme. In real life, these attacks may put both users and service providers at high risk In fact; this is a traditional as well as prudential way to deal with trustworthiness, since we cannot simply assume that beside the trusted authority, all service providers are also trusted. The basic reason is that assuming the existence of a trusted party is the strongest supposition in cryptography but it is usually very costly to develop and maintain. In particular defined collusion impersonation attacks as a way to capture the scenarios in which malicious service providers may recover a user’s credential and then impersonate the user to login to other service providers. It is easy to see that the above credential recovery attack is simply a special case of collusion impersonation attack where a single malicious service provider can recover a user’s credential. It must be emphasized that impersonation attacks without valid credentials seriously violate the security of SSO schemes as it allows attackers to be successfully authenticated without first obtaining a valid credential from the trusted authority after registration.

**ADVANTAGES OF PROPOSED SYSTEM:**

* The authors claimed to be able to: “prove that and are able to authenticate each other using our protocol.” but they provided no argument to show why each party could not be impersonated by an attacker. Second, the authors did discuss informally why their scheme could withstand impersonation attacks.
* The authors did not give details to show how the BAN logic can be used to prove that their scheme guarantees mutual authentication.
* In other words, it means that in an SSO scheme suffering these attacks there are alternatives which enable passing through authentication without credentials.

**Hardware requirements:**

Processor : Any Processor above 500 MHz.

Ram : 512Mb.

Hard Disk : 40 GB.

Compact Disk : 650 Mb.

Input device : Standard Keyboard and Mouse.

Output device : VGA and High Resolution Monitor.

**Software requirements:**

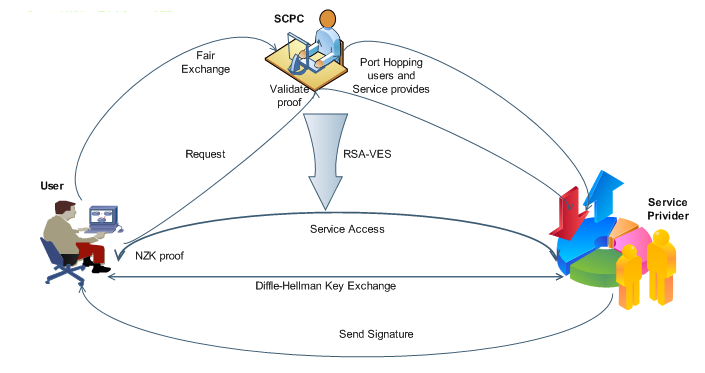
Operating System : Windows Family

Language : JAVA

Front End : Java Swings

Database : My Sql.

**System Architecture:**

****

Modules:

1. Port hopping for Users and Service Providers

2. Diffle - Hellman Model

3. Non-interactive zero – knowledge (NZK)

4. RSA - Verifiable Encryption Signature

5. Attacks and Authentication Against the Scheme

**1.Port hopping for Users and Service Providers**

Clients get the seed from SCPC for the pseudorandom function to compute the port sequence. The application data from Client to Server provider is sent out to the open ports of SCPC that changes every time units of Server clock, corresponding to client time units in Client’s clock.

Periodically, the sender and receiver can use new seeds of the pseudorandom function to generate different port number sequences, so that the port number sequence used for communication is changed periodically.

The clients and the server providers share a pseudorandom function to compute which port should be used in a certain time slot.

****

**2.Diffle-Hellman Model:**

For validating the proof, Bob can send his signature for the same message to Alice. For the purpose of fair exchange, Alice should send her signature in plaintext back to Bob after accepting Bob’s signature. If she refuses to do so, however, Bob can get her signature from the trusted party by providing Alice’s encrypted signature and his own signature, so that the trusted party can recover Alice’s signature and sends it to Bob, meanwhile, forwards Bob’s signature to Alice. Thus, fair exchange is achieved. SCPC randomly picks generator of , selects an ElGamal decryption key , and computes the corresponding public key . In addition, for completing the Diffie-Hellman key exchange SCPC chooses generator , where is another large prime number. SCPC also chooses a cryptographic hash function , where security parameter satisfies . Another security parameter is chosen to control the tightness of the ZK proof. Finally, SCPC publishes , and keeps secret.



**3. Non-interactive zero-knowledge(NZK)**

SCPC’s RSA signature on the square of the hashed user identity For user authentication, will encrypt his/her credential using ElGamal encryption of SCPC’s other public key by computing and , where of big order and is SCPC’s secret decryption key. In this improvement, SCPC also plays the role of the trust authority in VES. To convince a service provider that does encrypt his/her credential (i.e. SCPC’s RSA signature for ), must also provide an NZK proof to show that he or she knows a secret such that and .Such a proof , is called ‘proving the equality of two discrete logarithms in a group of unknown order will convince the service provider without leaking any useful information about ’s credential . For server authentication, service providers can simply issue signatures as the did, though the proposed changes give service providers the freedom to employ any secure signature scheme. The other procedures are the same as in the Chang-Lee scheme.



**4.RSA-Verifiable Encryption Signature**

In this phase, upon receiving a register request, SCPC gives fixed-length unique identity and issues credential . calculated as SCPC’s RSA signature on is an element of , which will be the main group we are calculating. each service provider with identity should maintain a pair of signing/verifying keys for a secure signature scheme (not necessarily RSA). denotes the signature on message signed by using signing key . denotes verifying of signature with public key , which outputs “1” or “0” to indicating if the signature is valid or invalid, Credential privacy or credential irrecoverableness requires that there be a negligible probability of an attacker recovering a valid credential from the interactions with a user. Again this property can be deduced from the signature hiding property of RSA-VES.. Signature hiding means that an attacker cannot extract a signature from VES without help from the user who encrypted the signature or the trusted authority who can decrypt a VES. So, if this improved SSO scheme fails to meet credential privacy, it implies that RSA-VES fails to satisfy signature hiding, which is contrary to the analysis. In fact, soundness and signature hiding are the two core security properties to guarantee the fairness of digital signature exchange using VES.



**5.Attacks and Authentication Against the Scheme**

The Chang–Lee scheme is actually not a secure SSO scheme because there are two potential effective and concrete impersonation attacks. The first attack, the “credential recovering attack” compromises the credential privacy in the Chang–Lee scheme as a malicious service provider is able to recover the credential of a legal user. The other attack, an “impersonation attack without credentials,” demonstrates how an outside attacker may be able to freely make use of resources and services offered by service providers, since the attacker can successfully impersonate a legal user without holding a valid credential and thus violate the requirement of soundness for an SSO scheme. In real life, these attacks may put both users and service providers at high risk.

. By agreeing with, when they said that “the Wu–Hsu’s modified version cold not protect the user’s token against a malicious service provider, the work also implicitly agrees that there is the potential for attacks from malicious service providers against SSO schemes. Moreover, if all service providers are assumed to be trusted, to identify him/her user can simply encrypt his/her credential under the RSA public key of service provider. Then, can easily decrypt this cipher text to get ’s credential and verify its validity by checking if it is a correct signature issued by . In fact, such a straightforward scheme with strong assumption is much simpler, more efficient and has better security, at least against this type of attack.



**Data flow Diagram:**

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

**0th level:**



**1st level:**

****

**2nd Level**

****

**OverAll:**



**Use case diagram:**

A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.



**Activity diagram:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organisational processes (i.e. workflows). Activity diagrams show the overall flow of control.

Activity diagrams are constructed from a limited number of shapes, connected with arrows. The most important shape types:

* rounded rectangles represent actions;
* diamonds represent decisions;
* bars represent the start (split) or end (join) of concurrent activities;
* a black circle represents the start (initial state) of the workflow;
* an encircled black circle represents the end (final state).

Arrows run from the start towards the end and represent the order in which activities happen.

****

Sequence Diagram:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart.

Sequence diagrams are sometimes called Event-trace diagrams, event scenarios, and timing diagrams.

A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.



Colloboration Diagram:

Collaboration diagrams belong to a group of UML diagrams called Interaction Diagrams. Collaboration diagrams, like Sequence Diagrams, show how objects interact over the course of time. However, instead of showing the sequence of events by the layout on the diagram, collaboration diagrams show the sequence by numbering the messages on the diagram. This makes it easier to show how the objects are linked together, but harder to see the sequence at a glance.



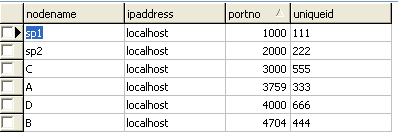
Class Diagram:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

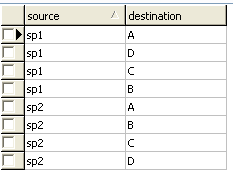
****

Dataset:

Node Register:



Service provider:



**ElGamal Algorithm:**

In cryptography, the ElGamal encryption system is an asymmetric key encryption algorithm for public-key cryptography which is based on the Diffie–Hellman key exchange. It was described by Taher Elgamal in ElGamal encryption is used in the free GNU Privacy Guard software, recent versions of PGP, and other cryptosystems. The Digital Signature Algorithm is a variant of the ElGamal signature scheme, which should not be confused with ElGamal encryption.

ElGamal encryption is probabilistic, meaning that a single plaintext can be encrypted to many possible ciphertexts, with the consequence that a general ElGamal encryption produces a expansion in size from plaintext to ciphertext. Encryption under ElGamal requires two exponentiations; however, these exponentiations are independent of the message and can be computed ahead of time if need be.

Diffie–Hellman Algorithm:

Diffie–Hellman key exchange (D-H) is a specific method of exchanging cryptographic keys. It is one of the earliest practical examples of key exchange implemented within the field of cryptography. The Diffie–Hellman key exchange method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.

The scheme was first published by Whitfield Diffie and Martin Hellman although it had been separately invented a few years earlier within GCHQ, the British signals intelligence agency, Hellman suggested the algorithm be called Diffie–Hellman–Merkle key exchange in recognition of Ralph Merkle's contribution to the invention of public-key cryptography.

**TESTING**

**PROCESS**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTS**

**UNIT TESTING**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**INTEGRATION TESTING**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**FUNCTIONAL TESTING**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**SYSTEM TESTING**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**WHITE BOX TESTING**

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

**BLACK BOX TESTING**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test Results**

All the test cases mentioned above passed successfully. No defects encountered.

TEST CASES

**TESTING OBJECTIVE**

Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.
* Features to be tested
* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

**Case1:** In Case1 we have test the performance of Network and Topology invention module, by giving various inputs and then compare with our expected output.

**Case2:** In Case2, we have test the performance of Spam zombies’ detection machine creation module, by giving various inputs and then compare with our expected output.

**Case3:** In Case3, we have test performance of spot detection and analysis module, by giving various inputs and then compare with our expected output.

**Case4:** In Case4, we have test the performance of Spam count based detection analysis module, by giving various inputs and then compare with our expected output.

**Case5:** In Case5, we have test the performance of Spam percentage based detection analysis module, by giving various inputs and then compare with our expected output.

**Case6:** In Case6, we have test the performance of Evaluation of dynamic IP address module, by giving various inputs and then compare with our expected output.

**15. ABOUT THE SOFTWARE**

## Java Technology

Java technology is both a programming language and a platform.

### The Java Programming Language

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

* + - Simple
    - Architecture neutral
    - Object oriented
    - Portable
    - Distributed
    - High performance
    - Interpreted
    - Multithreaded
    - Robust
    - Dynamic
    - Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Java byte codes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.



FIGURE 2- WORKING OF JAVA

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it’s a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make “write once, run anywhere” possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.

### The Java Platform

A platform is the hardware or software environment in which a program runs. We’ve already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and Mac OS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it’s a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

* The Java Virtual Machine (Java VM)
* The Java Application Programming Interface (Java API)

You’ve already been introduced to the Java VM. It’s the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages. The next section, What Can Java Technology Do?, highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that’s running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.



**FIGURE 3- THE JAVA PLATFORM**

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

## What Can Java Technology Do?

The most common types of programs written in the Java programming language are applets and applications. If you’ve surfed the Web, you’re probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a server serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a servlet. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs? It does so with packages of software components those provide a wide range of functionality. Every full implementation of the Java platform gives you the following features:

* **The essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
* **Applets**: The set of conventions used by applets.
* **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.
* **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
* **Security**: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
* **Software components**: Known as JavaBeans TM, can plug into existing component architectures.
* **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
* **Java Database Connectivity (JDBCTM)**: Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.



FIGURE 4 – JAVA 2 SDK

URL

The Web is a loose collection of higher-level protocols and file formats, all unified in a web browser. One of the most important aspects of the Web is that Tim Berners-Lee devised a saleable way to locate all of the resources of the Net. The Uniform Resource Locator (URL) is used to name anything and everything reliably.

The URL provides a reasonably intelligible form to uniquely identify or address information on the Internet. URLs are ubiquitous; every browser uses them to identify information on the Web. Within Java’s network class library, the URL class provides a simple, concise API to access information across the Internet using URLs.

Format

Two examples of URLs are http;//www.osborne.com/ and http:// www.osborne.com:80/index.htm.

A URL specification is based on four components. The first is the protocol to use, separated from the rest of the locator by a colon (:). Common protocols are http, ftp, gopher, and file, although these days almost everything is being done via HTTP. The second component is the host name or IP address of the host to use; this is delimited on the left by double slashes (/ /) and on the right by a slash (/) or optionally a colon (:) and on the right by a slash (/). The fourth part is the actual file path. Most HTTP servers will append a file named index.html or index.htm to URLs that refer directly to a directory resource.

Java’s URL class has several constructors, and each can throw a

MalformedURLException. One commonly used form specifies the URL with a string that is identical to what is displayed in a browser:

URL(String urlSpecifier)

The next two forms of the constructor breaks up the URL into its component parts:

URL(String protocolName, String hostName, int port, String path)

URL(String protocolName, String hostName, String path)

Another frequently used constructor uses an existing URL as a reference context and then create a new URL from that context.

URL(URL urlObj, String urlSpecifier)

The following method returns a URLConnection object associated with the invoking URL object. it may throw an IOException.

URLConnection openConnection( )-It returns a URLConnection object associated with the invoking URL object. it may throw an IOException.

### ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a *de facto* standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program, and each maintains a separate list of ODBC data sources.

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn’t change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn’t as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

**JDBC**

In an effort to set an independent database standard API for Java, Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of “plug-in” database connectivity modules, or *drivers*. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC’s framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

### JDBC Goals

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

1. **SQL Level API**

The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to “generate” JDBC code and to hide many of JDBC’s complexities from the end user.

1. **SQL Conformance**

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

1. ***JDBC must be implemental on top of common database interfaces***The JDBC SQL API must “sit” on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.
2. **Provide a Java interface that is consistent with the rest of the Java system**

Because of Java’s acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

1. **Keep it simple**

This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.

1. **Use strong, static typing wherever possible**

Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime.

1. **Keep the common cases simple**

Because more often than not, the usual SQL calls used by the programmer are simple SELECT’s, INSERT’s, DELETE’s and UPDATE’s, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible.

## Networking

### TCP/IP stack

The TCP/IP stack is shorter than the OSI one:



FIGURE 5 – TCP/IP STACK

TCP is a connection-oriented protocol; UDP (User Datagram Protocol) is a connectionless protocol.

### IP datagram’s

The IP layer provides a connectionless and unreliable delivery system. It considers each datagram independently of the others. Any association between datagram must be supplied by the higher layers. The IP layer supplies a checksum that includes its own header. The header includes the source and destination addresses. The IP layer handles routing through an Internet. It is also responsible for breaking up large datagram into smaller ones for transmission and reassembling them at the other end.

### TCP

TCP supplies logic to give a reliable connection-oriented protocol above IP. It provides a virtual circuit that two processes can use to communicate.

Internet addresses

In order to use a service, you must be able to find it. The Internet uses an address scheme for machines so that they can be located. The address is a 32 bit integer which gives the IP address. This encodes a network ID and more addressing. The network ID falls into various classes according to the size of the network address.

### Network address

Class A uses 8 bits for the network address with 24 bits left over for other addressing. Class B uses 16 bit network addressing. Class C uses 24 bit network addressing and class D uses all 32.

### Subnet address

Internally, the UNIX network is divided into sub networks. Building 11 is currently on one sub network and uses 10-bit addressing, allowing 1024 different hosts.

### Host address

8 bits are finally used for host addresses within our subnet. This places a limit of 256 machines that can be on the subnet.

### Total address



FIGURE 6 - IP ADDRESSING

The 32 bit address is usually written as 4 integers separated by dots.

### Port addresses

A service exists on a host, and is identified by its port. This is a 16 bit number. To send a message to a server, you send it to the port for that service of the host that it is running on. This is not location transparency! Certain of these ports are "well known".

### Sockets

A socket is a data structure maintained by the system to handle network connections. A socket is created using the call socket. It returns an integer that is like a file descriptor. In fact, under Windows, this handle can be used with Read File and Write File functions.

#include <sys/types’>

#include <sys/socket.h>

int socket(int family, int type, int protocol);

Here "family" will be AF\_INET for IP communications, protocol will be zero, and type will depend on whether TCP or UDP is used. Two processes wishing to communicate over a network create a socket each. These are similar to two ends of a pipe - but the actual pipe does not yet exist

**References:**

# 1. Z. Fu, M. Papatriantafilou, and P. Tsigas, “Mitigating Distributed Denial of Service Attacks in Multiparty Applications in the Presence of Clock Drifts,” Proc. IEEE Int’l Symp. Reliable Distributed Systems (SRDS), Oct. 2008.

2. T.-S.Wu and C.-L. Hsu, “Efficient user identification scheme with key distribution preserving anonymity for distributed computer networks,” *Comput. Security*, vol. 23, no. 2, pp. 120–125, 2004.

3. H.-M. Sun, Y.-H. Chen, and Y.-H. Lin, “oPass: A user authentication protocol resistant to password stealing and password reuse attacks,” *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 651–663, Apr. 2012.

4. L. Harn and J. Ren, “Generalized digital certificate for user authentication and key establishment for secure communications,” *IEEE Trans. Wireless Commun.*, vol. 10, no. 7, pp. 2372–2379, Jul. 2011

Coding:

package server;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.io.\*;

import java.util.\*;

import java.sql.\*;

import java.net.\*;

public class ServiceCreation extends JFrame

{

private JLabel jLbl\_Source;

private JLabel jLbl\_Destination;

private JLabel jLabel3;

private JComboBox jCmbox\_Spname;

private JComboBox jCmb\_nodename;

private JButton jBtn\_Create;

private JPanel contentPane;

Vector v=new Vector();

Vector v1=new Vector();

String nodeList="";

Connection con;

Statement st;

ResultSet rs;

ObjectOutputStream oos;

ObjectInputStream ois;

Socket soc;

ServerSocket ss;

public static void main(String[] args)

{

ServiceCreation frame = new ServiceCreation();

//EventQueue.invokeLater(new Runnable() {

// public void run() {

// try {

// ServiceCreation frame = new ServiceCreation();

// frame.setVisible(true);

//

// }

// catch(Exception ex)

// {

// ex.printStackTrace();

// }

// }

//});

}

public ServiceCreation()

{

super();

initializeComponent();

create();

this.setVisible(true);

}

private void initializeComponent()

{

jLbl\_Source = new JLabel();

jLbl\_Destination = new JLabel();

jLabel3 = new JLabel();

jCmbox\_Spname = new JComboBox(v1);

jCmb\_nodename = new JComboBox(v);

jBtn\_Create = new JButton();

contentPane = (JPanel)this.getContentPane();

jLbl\_Source.setText("Service Provider Name :");

jLbl\_Destination.setText("Node Name :");

jLabel3.setText("Security Analysis of Single Sign On Mechanism in Distributed Computer Networks");

jCmbox\_Spname.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e)

{

jCmbox\_Spname\_actionPerformed(e);

}

});

jCmb\_nodename.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e)

{

jCmb\_nodename\_actionPerformed(e);

}

});

jBtn\_Create.setText("Create Service ");

jBtn\_Create.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e)

{

jBtn\_Create\_actionPerformed(e);

}

});

contentPane.setLayout(null);

addComponent(contentPane, jLbl\_Source, 45,99,117,18);

addComponent(contentPane, jLbl\_Destination, 49,166,88,30);

addComponent(contentPane, jLabel3, 53,27,409,30);

addComponent(contentPane, jCmbox\_Spname, 183,104,124,25);

addComponent(contentPane, jCmb\_nodename, 184,165,122,26);

addComponent(contentPane, jBtn\_Create, 185,235,125,28);

this.setTitle("ServiceCreation");

this.setLocation(new Point(80, 27));

this.setSize(new Dimension(521, 380));

this.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

this.setResizable(false);

}

private void addComponent(Container container,Component c,int x,int y,int width,int height)

{

c.setBounds(x,y,width,height);

container.add(c);

}

private void jCmbox\_Spname\_actionPerformed(ActionEvent e)

{

System.out.println("\njCmbox\_Spname\_actionPerformed(ActionEvent e) called.");

Object o = jCmbox\_Spname.getSelectedItem();

System.out.println(">>" + ((o==null)? "null" : o.toString()) + " is selected.");

}

private void jCmb\_nodename\_actionPerformed(ActionEvent e)

{

System.out.println("\njCmb\_nodename\_actionPerformed(ActionEvent e) called.");

Object o = jCmb\_nodename.getSelectedItem();

System.out.println(">>" + ((o==null)? "null" : o.toString()) + " is selected.");

}

private void jBtn\_Create\_actionPerformed(ActionEvent e)

{

System.out.println("\njBtn\_Create\_actionPerformed(ActionEvent e) called.");

serviceCreate();

}

public void serviceCreate()

{

String source=jCmbox\_Spname.getSelectedItem().toString().trim();

String dest=jCmb\_nodename.getSelectedItem().toString().trim();

System.out.println(source);

System.out.println(dest);

try

{

DatabaseConnection.getConnection();

st=con.createStatement();

int i=st.executeUpdate("insert into serviceprovider values('"+source+"','"+dest+"')");

if(i!=0)

{

JOptionPane.showMessageDialog(this,"Service Created Sucessfully");

}

st.close();

}

catch (Exception e)

{

e.printStackTrace();

}

}

public void refresh()

{

try

{

DatabaseConnection.getConnection();

st=con.createStatement();

int j=st.executeUpdate("delete \* from serviceprovider");

}

catch (Exception sqle)

{

sqle.printStackTrace();

}

}

public Vector create()

{

Vector allelements=new Vector();

try

{

System.out.println("i am create method");

Class.forName("com.mysql.jdbc.Driver");

con=DriverManager.getConnection("jdbc:mysql://localhost:3306/ssomech","root","root123");

DatabaseConnection.getConnection();

st=con.createStatement();

rs=st.executeQuery("select nodename from noderegister");

v.add("-Select-");

while(rs.next())

{

nodeList=rs.getString(1);

if(nodeList.length()<2)

{

v.add(nodeList);

System.out.println("\*\*\*\*\*node list value \*\*\*\*"+nodeList);

}

else

{

v1.add(nodeList);

System.out.println("----------serviece list value--------- "+nodeList);

}

}

allelements.addAll(v);

allelements.addAll(v1);

System.out.println("the vector elements Are"+allelements);

}

catch (Exception e)

{

e.printStackTrace();

}

return allelements;

}

}

// JFrame.setDefaultLookAndFeelDecorated(true);

// JDialog.setDefaultLookAndFeelDecorated(true);

// try

// {

// UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");

// }

// catch (Exception ex)

// {

// System.out.println("Failed loading L&F: ");

// System.out.println(ex);

// }

// new ServiceCreation();

// }

//= End of Testing =

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* NodeRegister \*/

/\* \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

package server;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.io.\*;

import java.net.\*;

import java.util.\*;

import java.sql.\*;

public class NodeRegister extends JFrame

{

private JLabel jLabel1;

private JLabel jLabel2;

private JLabel jLabel3;

private JTextField jTextField1;

private JTextField jTextField2;

private JPasswordField jPasswordField1;

private JButton jButton1;

private JButton jButton2;

private JButton jButton3;

private JLabel jL\_Image;

private JLabel jLabel4;

private JTextField jTextField3;

private JPanel contentPane;

Connection con;

Statement st;

ResultSet rs;

public NodeRegister()

{

super();

initializeComponent();

this.setVisible(true);

}

private void initializeComponent()

{

jLabel1 = new JLabel();

jLabel2 = new JLabel();

jLabel3 = new JLabel();

jTextField1 = new JTextField();

jTextField2 = new JTextField();

jPasswordField1 = new JPasswordField();

jButton1 = new JButton();

jButton2 = new JButton();

jButton3 = new JButton();

jL\_Image = new JLabel();

jLabel4 = new JLabel();

jTextField3 = new JTextField();

contentPane = (JPanel)this.getContentPane();

jLabel1.setText("Node Name");

jLabel1.setFont(new Font("Courier New",Font.ITALIC,16));

jLabel1.setFont(new Font("Courier New",Font.BOLD,16));

jLabel2.setText("IP Address");

jLabel2.setFont(new Font("Courier New",Font.ITALIC,16));

jLabel2.setFont(new Font("Courier New",Font.BOLD,16));

jLabel3.setText("Port Number");

jLabel3.setFont(new Font("Courier New",Font.ITALIC,16));

jLabel3.setFont(new Font("Courier New",Font.BOLD,16));

jLabel4.setText("Unique ID");

jLabel4.setFont(new Font("Courier New",Font.ITALIC,16));

jLabel4.setFont(new Font("Courier New",Font.BOLD,16));

jTextField1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e)

{

jTextField1\_actionPerformed(e);

}

});

jTextField2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e)

{

jTextField2\_actionPerformed(e);

}

});

jPasswordField1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e)

{

jPasswordField1\_actionPerformed(e);

}

});

jTextField3.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e)

{

jTextField3\_actionPerformed(e);

}

});

jButton1.setText("Register");

jButton1.setFont(new Font("Courier New",Font.ITALIC,16));

jButton1.setFont(new Font("Courier New",Font.BOLD,16));

jButton1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e)

{

jButton1\_actionPerformed(e);

}

});

jButton2.setText("Service Creation");

jButton2.setFont(new Font("Courier New",Font.ITALIC,15));

jButton2.setFont(new Font("Courier New",Font.BOLD,15));

jButton2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e)

{

jButton2\_actionPerformed(e);

}

});

jButton3.setText("Cancel");

jButton3.setFont(new Font("Courier New",Font.ITALIC,16));

jButton3.setFont(new Font("Courier New",Font.BOLD,16));

jButton3.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e)

{

jButton3\_actionPerformed(e);

}

});

contentPane.setLayout(null);

contentPane.setBackground(new Color(199, 133, 148));

addComponent(contentPane, jLabel1, 136,75,120,18);

addComponent(contentPane, jLabel2, 138,161,120,18);

addComponent(contentPane, jLabel3, 135,260,120,18);

addComponent(contentPane, jLabel4, 140,300,120,18);

addComponent(contentPane, jTextField1, 269,72,120,27);

addComponent(contentPane, jTextField2, 267,157,120,27);

addComponent(contentPane, jPasswordField1, 269,253,120,27);

addComponent(contentPane, jTextField3, 269,300,120,27);

addComponent(contentPane, jButton1, 45,350,110,28);

addComponent(contentPane, jButton2, 160,350,225,28);

addComponent(contentPane, jButton3, 390,350,110,28);

addComponent(contentPane, jL\_Image, -5,-2,558,480);

jL\_Image.setIcon(new ImageIcon("Images\\wood-ipad-background.jpg"));

this.setTitle("NodeRegister ");

this.setLocation(new Point(150,150));

this.setSize(new Dimension(530, 480));

this.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

this.setResizable(false);

}

private void addComponent(Container container,Component c,int x,int y,int width,int height)

{

c.setBounds(x,y,width,height);

container.add(c);

}

private void jTextField1\_actionPerformed(ActionEvent e)

{

System.out.println("\njTextField1\_actionPerformed(ActionEvent e) called.");

}

private void jTextField2\_actionPerformed(ActionEvent e)

{

System.out.println("\njTextField2\_actionPerformed(ActionEvent e) called.");

}

private void jTextField3\_actionPerformed(ActionEvent e)

{

System.out.println("\njTextField2\_actionPerformed(ActionEvent e) called.");

}

private void jPasswordField1\_actionPerformed(ActionEvent e)

{

System.out.println("\njPasswordField1\_actionPerformed(ActionEvent e) called.");

}

private void jButton1\_actionPerformed(ActionEvent e)

{

register();

}

private void jButton2\_actionPerformed(ActionEvent e)

{

new ServiceCreation();

}

private void jButton3\_actionPerformed(ActionEvent e)

{

}

public void register()

{

try

{

String nodeName=jTextField1.getText();

String ipadd=jTextField2.getText();

String portno=jPasswordField1.getText();

String uniqueid=jTextField3.getText();

int port=Integer.parseInt(portno);

System.out.println(nodeName);

System.out.println(ipadd);

System.out.println(portno);

Socket soc=new Socket(ServerAddress.getIp(),8000);

ObjectOutputStream oos=new ObjectOutputStream(soc.getOutputStream());

ObjectInputStream ois=new ObjectInputStream(soc.getInputStream());

oos.writeObject("nodeRegister");

oos.writeObject(nodeName);

oos.writeObject(ipadd);

oos.writeObject(portno);

oos.writeObject(uniqueid);

String serRply=(String)ois.readObject();

System.out.println("Server is replied "+serRply);

if(serRply.equals("ok"))

{

JOptionPane.showMessageDialog(this, "Node Created Successfully","Msg", JOptionPane.INFORMATION\_MESSAGE);

}

else

{

JOptionPane.showMessageDialog(this,"Input Not Supported", "Error",JOptionPane.ERROR\_MESSAGE);

}

}

catch (Exception regex)

{

regex.printStackTrace();

}

}

//============================= Testing ================================//

//= =//

//= The following main method is just for testing this class you built.=//

//= After testing,you may simply delete it. =//

//======================================================================//

public static void main(String[] args)

{

JFrame.setDefaultLookAndFeelDecorated(true);

JDialog.setDefaultLookAndFeelDecorated(true);

try

{

//UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");

UIManager.setLookAndFeel(ch.randelshofer.quaqua.QuaquaManager.getLookAndFeel());

}

catch (Exception ex)

{

System.out.println("Failed loading L&F: ");

System.out.println(ex);

}

new NodeRegister();

}

//= End of Testing =

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Login \*/

/\* \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

package server;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.io.\*;

import java.util.\*;

import java.net.\*;

import java.sql.\*;

/\*\*

\* Summary description for Login

\*

\*/

public class Login extends JFrame

{

// Variables declaration

private JLabel jLblUsrname;

private JLabel jLblPass;

private JLabel jLblCfrmpass;

private JTextField jTF\_UsrName;

private JPasswordField jPass;

private JPasswordField jPassConfrm;

private JButton jBtnLogin;

private JButton jBtnCancel;

private JPanel contentPane;

// End of variables declaration

Vector v=new Vector();//

String username="";

String pass="";

int passip;

public Login()

{

super();

initializeComponent();

//

// TODO: Add any constructor code after initializeComponent call

//

this.setVisible(true);

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always regenerated

\* by the Windows Form Designer. Otherwise, retrieving design might not work properly.

\* Tip: If you must revise this method, please backup this GUI file for JFrameBuilder

\* to retrieve your design properly in future, before revising this method.

\*/

private void initializeComponent()

{

jLblUsrname = new JLabel();

jLblPass = new JLabel();

jLblCfrmpass = new JLabel();

jTF\_UsrName = new JTextField();//

jPass = new JPasswordField();

jPassConfrm = new JPasswordField();

jBtnLogin = new JButton();

jBtnCancel = new JButton();

contentPane = (JPanel)this.getContentPane();

//

// jLblUsrname

//

jLblUsrname.setText("User Name");

//

// jLblPass

//

jLblPass.setText("Password");

//

// jLblCfrmpass

//

jLblCfrmpass.setText("Confirm Password");

//

// jTF\_UsrName

//

jTF\_UsrName.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e)

{

jTF\_UsrName\_actionPerformed(e);

}

});

//

// jPass

//

jPass.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e)

{

jPass\_actionPerformed(e);

}

});

//

// jPassConfrm

//

jPassConfrm.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e)

{

jPassConfrm\_actionPerformed(e);

}

});

//

// jBtnLogin

//

jBtnLogin.setText("Login");

jBtnLogin.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e)

{

jBtnLogin\_actionPerformed(e);

}

});

//

// jBtnCancel

//

jBtnCancel.setText("Cancel");

jBtnCancel.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e)

{

jBtnCancel\_actionPerformed(e);

}

});

//

// contentPane

//

contentPane.setLayout(null);

addComponent(contentPane, jLblUsrname, 134,71,100,25);

addComponent(contentPane, jLblPass, 133,125,67,26);

addComponent(contentPane, jLblCfrmpass, 131,183,150,26);

addComponent(contentPane, jTF\_UsrName, 258,66,133,29);

addComponent(contentPane, jPass, 259,123,129,30);

addComponent(contentPane, jPassConfrm, 258,184,130,29);

addComponent(contentPane, jBtnLogin, 187,275,114,28);

addComponent(contentPane, jBtnCancel, 332,276,91,28);

//

// Login

//

this.setTitle("Login");

this.setLocation(new Point(0, 0));

this.setSize(new Dimension(570, 458));

this.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

this.setResizable(false);

}

/\*\* Add Component Without a Layout Manager (Absolute Positioning) \*/

private void addComponent(Container container,Component c,int x,int y,int width,int height)

{

c.setBounds(x,y,width,height);

container.add(c);

}

//

// TODO: Add any appropriate code in the following Event Handling Methods

//

private void jTF\_UsrName\_actionPerformed(ActionEvent e)

{

System.out.println("\njTF\_UsrName\_actionPerformed(ActionEvent e) called.");

// TODO: Add any handling code here

}

private void jPass\_actionPerformed(ActionEvent e)

{

System.out.println("\njPass\_actionPerformed(ActionEvent e) called.");

// TODO: Add any handling code here

}

private void jPassConfrm\_actionPerformed(ActionEvent e)

{

System.out.println("\njPassConfrm\_actionPerformed(ActionEvent e) called.");

// TODO: Add any handling code here

}

private void jBtnLogin\_actionPerformed(ActionEvent e)

{

System.out.println("\njBtnLogin\_actionPerformed(ActionEvent e) called.");

// TODO: Add any handling code here

login();

}

public void login()// jPass jPassConfrm

{

username=jTF\_UsrName.getText();

pass= jPass.getText();

System.out.println("my username "+username);

System.out.println("my password is "+pass);

passip=Integer.parseInt(pass);

try

{

Socket soc=new Socket("localhost",8000);

ObjectOutputStream oos=new ObjectOutputStream(soc.getOutputStream());

ObjectInputStream ois=new ObjectInputStream(soc.getInputStream());

oos.writeObject("login");

oos.writeObject(username);

oos.writeObject(pass);

// v.add(username);

// v.add(pass);

String login=(String)ois.readObject();

//Vector value=(Vector)ois.readObject();

if(login.equals("loginsuccess"))

{

JOptionPane.showMessageDialog(this,"Login Successfully","Message", JOptionPane.INFORMATION\_MESSAGE);

System.out.println("Welcome to the page");

System.out.println("login form user "+username);

System.out.println("login form user "+passip);

new Node1(username,passip);

}

else if(login.equals("loginSP"))

{

JOptionPane.showMessageDialog(this,"Login Successfully","Message", JOptionPane.INFORMATION\_MESSAGE);

System.out.println("login form service provider "+username);

System.out.println("login form service provider "+passip);

new Serviceproviderhome(username,passip);

}

else if(login.equals("loginfailed"))

{

JOptionPane.showMessageDialog(this,"Login Failed", "Error",JOptionPane.ERROR\_MESSAGE);

}

}

catch (Exception ex)

{

ex.printStackTrace();

}

}

private void jBtnCancel\_actionPerformed(ActionEvent e)

{

System.out.println("\njBtnCancel\_actionPerformed(ActionEvent e) called.");

// TODO: Add any handling code here

}

//

// TODO: Add any method code to meet your needs in the following area

//

