Figure 8 presents a high level enterprise architecture showing how a USB-based TCP is meant to be used, and the components that are required to support it. There are three components involved in running an application based on the USB-based TCP. In a typical use scenario, a USB-based TCP is plugged into an unknown host computer. Depending on the application selected, the host computer may also be used to allow the user access to a variety of Input/Output interfaces such as a screen display and keyboard/mouse. Note that they are not a part of the USB-based TCP reference platform as shown in Figure 4. The Privacy Certifying Authority (Privacy CA) validates a USB-based TCP’s characteristics to a remote party by running a TCG attestation protocol. This means all devices’ characteristics are known to the Privacy CA. The application server hosts an application that USB-Based TCP uses to process any critical transaction. Based on the above enterprise architecture, we have developed a small demonstration application that exercises most of our USB-based TCP’s capabilities. Our demonstrator uses a familiar banking transaction scenario which allows a customer to view account balances and transfer money from one account to another after successful authentication took place. The layouts and technologies used in our demonstrator are shown in Figure 9.

Our USB-based TCP contains a mini Web server, developed in-house, that contains sufficient functionality to process SOAP formatted HTTP request/response messages. Similarly, we also have another mini Web server running on the Privacy CA component. These mini Web servers enable us to conduct driverless communication among the components in our enterprise architecture using standard Web languages such as HTML, JavaScript, and AJAX. The application server hosts our banking demonstration written in JSP. The Apache Tomcat Server and Apache Axis engine are also installed in our application server to support Web-based services. Java Objects are used in the application server to provide database functionality to allow us to store the banking information for each customer. We developed our own in-house Privacy CA as a standalone Java application.

When the device is plugged in to the host machine, it allocates a link local IP address to it, used for communication between the local host and the device. When a user types a URL to access services from the application server, the host machine becomes aware of the IP address of the application server. However, the USB device and application server are unaware of each other’s IP address as the current addresses are local. We solved this issue by designing a Web page with embedded JavaScript code that performs as an address proxy. We next describe the design and implementation of the Web page in the context of the attestation protocol, and include some code snippets to illustrate the important points.

Figure 10 illustrates the overall Web page components. A JavaScript-based proxy is embedded in the main page. The proxy is basically a listener that watches for any incoming or outgoing messages. The main page itself contains two iframe tags. Each iframe uploads HTML files that are located on the portable device and the Privacy CA. These files are loaded when the main page is rendered by the client browser. Once loaded, the JavaScript proxy code associated with the main page starts executing, allowing communication by both the device and Privacy CA to the application server.

The mobile and trusted platform operates within this scenario as follows.

1) A customer plugs in the TCP device into the USB port on an untrusted host PC. When the device is plugged in, it draws power from the host PC and boots its operating system. It then loads libraries and applications as shown in Figure 7.

2) The customer uses a known, secured browser to access the bank’s application through a URL. The main page contains two iframes as follows.

```html
<iframe src="%=pcaURL%" /privacyCA.htm id="caFrame" />
<iframe src="%=tedURL%" /ted.htm id="tedFrame" />
```

The first iframe is designed for privacy CA and the second for the TCP device.

3) When the main page is rendered, two HTML pages, `privacyCA.htm` and `ted.htm`, are uploaded connecting the application server to the device and privacy CA. The user then performs a simple name/password