WiKID Systems, Inc
Technology White Paper

Dual-Source Two-factor Authentication

The WiKID Strong Authentication System uses public key encryption on PCs and smart phones to authenticate users. It is less expensive, more extensible and easier to use than traditional hardware tokens. Token clients run on Windows, Mac, Linux, Android, iPhone, Blackberry, J2ME & Windows mobile. PC clients include mutual https authentication to thwart MITM attacks on SSL VPNs.

Version 2.0
October, 2012
http://www.wikidsystems.com

Introduction

The WiKID Strong Authentication System was created as a strong two-factor security technology to replace existing, high-cost, two-factor systems. The WiKID Strong Authentication System employs lightweight clients using asymmetric cryptography for receiving passcodes for authorization into intranets, VPNs, online banking sites, etc.

The WiKID Strong Authentication System aims for three *major* goals: 1) to be as secure or more secure than existing two-factor security systems, 2) to be significantly less costly to implement and maintain and 3) to provide the extensibility required in this connected, security conscious era. In this regard, WiKID engineers have designed the system to use asymmetric cryptographic elements for client registration and identification (through cryptographic key exchange) and for transport security (to prevent the interception of codes over untrusted wireless/wired networks).

Like existing two-factor authentication methods, the WiKID Strong Authentication System requires the passcodes to be derived and verified in two separate channels. Through verification of the validity of the device and triangulation of the outgoing and incoming codes, the passcodes are authenticated and matched against a named user. However, the WiKID Strong Authentication System is more effective than traditional two-factor systems in several key ways:

1) The intelligence of the passcode generation is not within the client device, preventing theft and reverse engineering;
2) The system is not 100% counter/time/algorithm-based (as are most competing systems), preventing the existence of N+1 and N-1 valid codes as the single-use devices age and lose synchronization;
3) The system is based on a Request-Response Architecture whereby it generates a code *only when requested*, not continuously when not needed, which would open the system to algorithm analysis or cracking;
4) The system employs no single-use devices, which eliminates the expenditure for and investment in short-life devices;
5) The system can support multiple security domains both on the client, to reduce the need for multiple single use devices, and on the server, to enforce flexible security policies;
6) The system uses asymmetric keys which enables a single client to be used across multiple enterprises without the need for a federated trust relationship between enterprises;
7) The PC clients can provide host/mutual validation by verifying the SSL certificate of the targeted website for the user and launching the default browser to the site, eliminating man-in-the-middle attacks such as the typical phishing attack; and
8) Because each domain is cryptographically distinct, WiKID is capable of securely handling session and transaction authentication with a single token client.

This document provides a technical overview of the system and the security employed, and includes a glossary of terms as used by WiKID for familiarization with the system.

I. Development and Security Philosophy

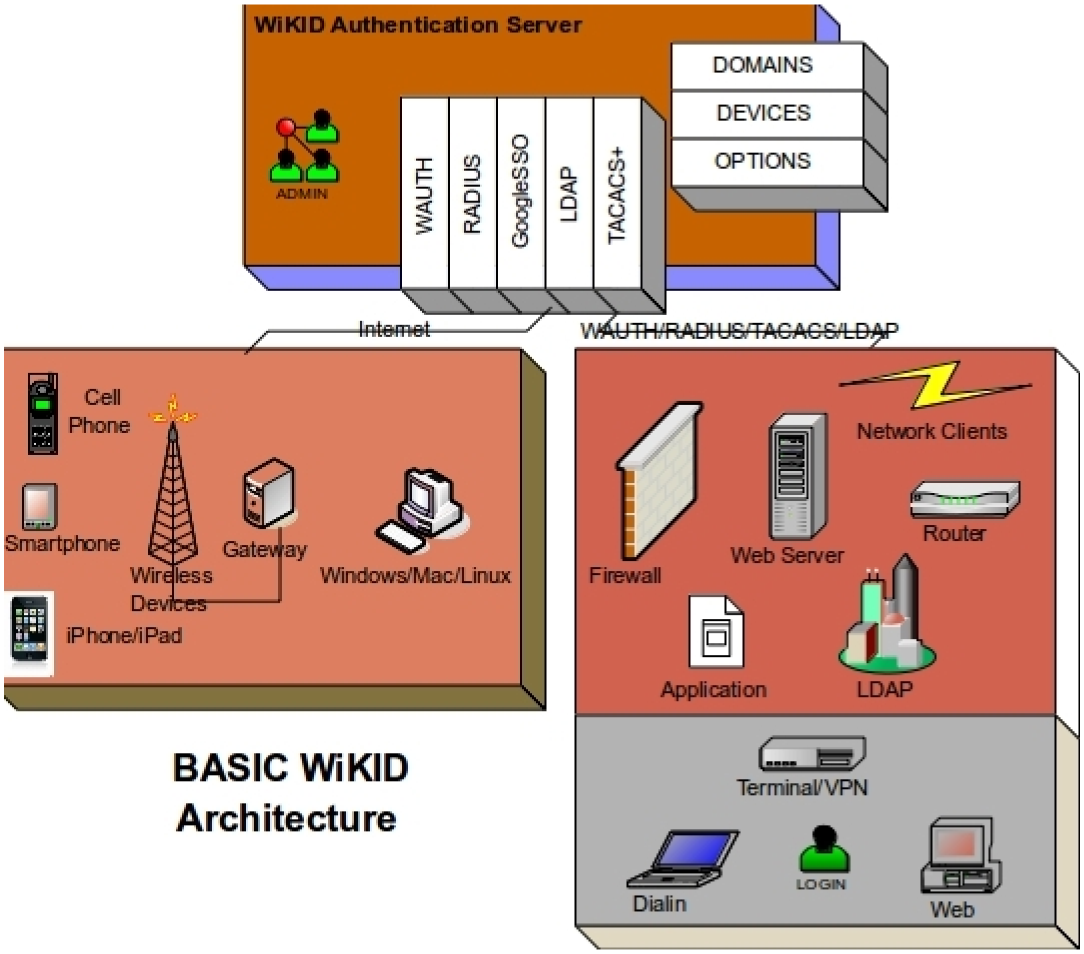
As a two-factor authentication system, the WiKID Strong Authentication System follows several guiding principles:

1. The system is as secure or more secure than existing two-factor security systems.
2. The system is easier to use, maintain and administer than existing two-factor security systems.
3. The system is less expensive to implement and maintain than existing two-factor security systems.
4. The system is highly extensible, addressing multiple authentication requirements as they arrive.
5. The system uses standard, published cryptographic methods in the belief that "security through obscurity" is a fallacy. Instead, security is maximized by the strength of the system design and of the underlying cryptographic methods.

The design and implementation of the WiKID system consistently implements these principles in all respects.

II. Architecture

A. Overview



1. WiKID Authentication server

a. **Server.** The ISO version of the WiKID Authentication Server (WAS) runs on a hardened version of Centos Linux (see *System Considerations* for more information on hardened Linux). The WAS uses mainly JAVA-based server components and application components. The underlying database is an embedded version of PostgreSQL that is completely self-contained and does not require database administration. (System administrators using non-ISO versions are expected to harden the server themselves to their specifications.)

b. **Components.** The WAS contains a database of domains, devices, users and Protocol Modules. Additionally, the WAS also offers a web-based administrative utility for the management of these components.

i. **Domains.** Each WAS can host multiple security domains. The security domain is intended to segregate users with respect to access and services. For example, VPN access for users may be provided with one domain, while access to critical infrastructure or PCI-compliant systems can use another. Separate security policies can be provided for each domain and access can be granted on a device/individual user basis. Unlike other two-factor systems, the client for each domain (the WiKID Client) is the same (see *WiKID Clients* below). Upon creation, each domain generates a key pair for payload encryption within the passcode request/passcode reception process. These keys are the domain private key and the domain public key and are exchanged in the registration process. If the domain is for an SSL protected Website, it can include a hash of the website's certificate for host validation.

ii. **Devices.** The cryptographic signature for each WiKID Client is stored within the WAS and associated with a domain and user. The cryptographic signature is a 2048 bit-equivalent client public key as generated in the registration process (see *Registration Process* later in this document). This strong, asymmetric encryption key is generated **on** the device and serves to identify a valid device within the security domain and to provide payload security during the reception of passcodes. The device also receives, stores and utilizes the public key of the WAS server which is provided by the server during the registration process.

iii. **Users.** The WAS stores named users and associates each user with a device and a security domain. This process allows for login within a network service, whether it is via a RADIUS-based VPN, secure website or any other service that is provided by a Network Client (See *Network Clients* below).

iv. **Protocol Modules.** For the WAS to communicate with a Network Client, the WAS must have the appropriate Protocol Module installed and configured. Currently, the WAS supports multiple Protocol Modules:

1) WiKID Authentication Protocol, or WAUTH
2) Remote Authentication Dial In User Service or RADIUS.
3) LDAP version 3
4) TACACS+ (From Cisco Systems)
5) SAML support for Google Apps for your Domain SSO

The Protocol Modules must be configured with standard properties – interface, port, etc. – and with protocol specific properties – shared secret, accounting toggle for RADIUS, Network Client certificate for WAUTH.

1) **WAUTH.** WAUTH is a WiKID-proprietary, encrypted protocol for the verification of passcodes from certain Network Clients. WAUTH is more secure than RADIUS due to client certificate authenticated SSL transport, but it requires use of a WiKID Application Component to be implemented within a Network Client. Typically, the component is a JAVA component, COM object, PHP, Python or Ruby module that can be integrated into a website, a web application or a client-server application. In addition, WiKID has created a number of

modules based on WAUTH to add two-factor authentication to various projects such as Plone and Cloudstack.

2) **RADIUS.** RADIUS is a standard TCP/IP-based service for authorization and access control. The RADIUS protocol is detailed within Internet Engineering Taskforce RFC 2865 with additional information provided by RFC's 2866 to 2869. The RADIUS protocol is, as noted, less secure than WAUTH since it utilizes a MAC encoding of the packets within the protocol exchange. WiKID recommends that RADIUS/TACACS+ be used only on trusted networks, *e.g.,* corporate Intranets, or to support standard VPN and dial-in clients. RADIUS is supported by Microsoft's RAS, Cisco's routing and firewall software as well as by most of the terminal and PPP device makers. The WAS fully supports RADIUS authentication and less fully RADIUS accounting and proxy features.
3) **LDAP.** WiKID supports the use of authenticated LDAP bind commands to validate passcodes.
4) **TACACS+**. TACACS+ is an older protocol provided by Cisco systems. It has generally been superseded by RADIUS and is provided for backward compatibility with existing customer environments.
5) Google SSO. Point your Google Apps for your domain at your WiKID server for SSO and two-factor authentication.

i. **Administration.** WiKID provides a fully web-enabled administration system to create, modify, enable and disable each of the components noted in this document (see *Administration Details* in this document for more details).

2. **WiKID Clients.** The chief differentiator between WiKID and other two-factor authentication providers is the use of asymmetric encryption. The WiKID Client is the platform for the identification of the individual user and the provision of the passcode. Currently, WiKID supports Windows, Mac, Linux, iPhone/iPads, RIM Blackberries, Android devices and Java-enabled telephones. Additionally, we have an experimental HTML5 tokens. The client device requires the installation of a small WiKID client application. This application manages several key processes including key generation, registration, passcode request, passcode reception, and offline passcode verification.

   a. **Key Generation.** When the application is started for the first time, the application automatically generates an asymmetric key pair. These keys are used for the decryption of the payload from the WAS server and identification of the device. The keys are asymmetric, and the strength of the key pair is equivalent to RSA 2048 bits. The time for the key generation process is typcially unnoticeable.

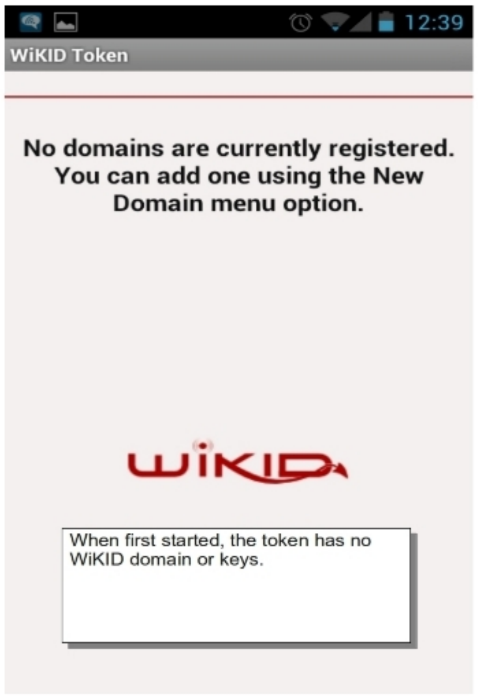
      For smart phone tokens, WiKID has chosen to use the NTRU algorithm for key generation and payload encryption. It is generally accepted that the encryption strength of the NTRU algorithm is the same (or superior) to the existing elliptical curve or RSA asymmetric algorithms, but, with the inferior computing power of the wireless devices, WiKID selected the NTRU algorithm because it is much, much faster when running on a small device. **It is important that the key generation be completed *on the device*, not on a PC or server and transferred to the device**. In this way the local client key never leaves the device and is not subject to interception, electronic copying or redistribution. Thus, the WiKID Client functions similarly to a Smart Card. But unlike a Smart Card, it does not require a wired reader, which greatly reduces the cost of implementation and greatly increases portability.

b. **Registration Process.** Before a WiKID client can authenticate with the WAS, it must first be registered within the WAS and the security domain (more specific technical information on registration can be found in the *Technical Considerations* section of this document) . Each supported security domain requires approximately 1200 bytes of storage on the client. There are two methods for registering a device:

    i. **Self-registration.** If the domain is configured for self-registration, the WiKID Client can request registration through the client application. First, the user must use the client application to request that the device be added to the WAS and security domain in question. The *Config* menu is used to add a domain. The end-user must enter a server code. The technical security staff should provide this server code. Once this 12-digit code has been entered, the user must establish a PIN for the domain connection. A separate PIN can be provided for each domain, and it is recommended that the user establish unique PINs for each domain. At this point in the process, the public key that was generated in the key generation process is provided to the WAS. The WAS will then record the cryptographic key and provide the domain's public key, a unique identifier for the instance of the device within the security domain and a large registration code. Additionally, the server will generate a second set of keys unique to that particular client device in the security domain for offline passcode verification (see below).

    The registration code is a one-use temporary element. It is not a passcode or password and cannot be used for access to the WiKID system. Instead, the registration code is used to associate the WiKID Client with a known user within a trusted system. When the user goes to the WiKID registration website (or other registration system), he or she is required to enter an existing user ID, identifying information and the registration code. This process associates the client device with the user, verifies the client device as valid within the security domain and activates the client device within the security domain. We provide sample scripts on the server that allow users to login with their Active Directory credentials and register their own token clients.

The Client View of the Registration Process Illustrated

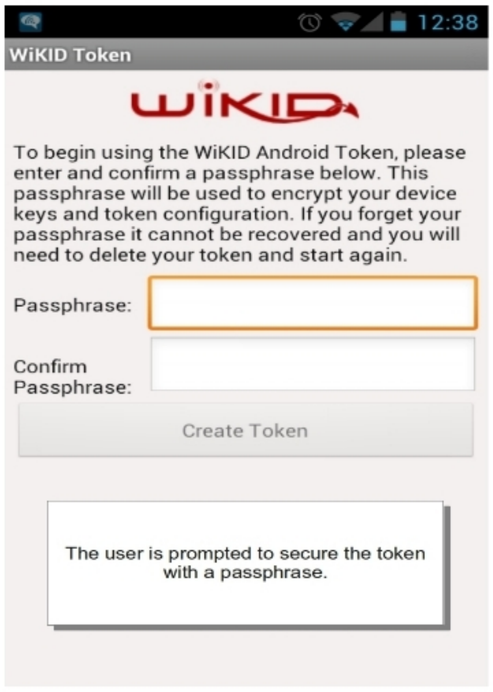


12:39

WiKID Token

No domains are currently registered. You can add one using the New Domain menu option.

When first started, the token has no WiKID domain or keys.



12:38

WiKID Token

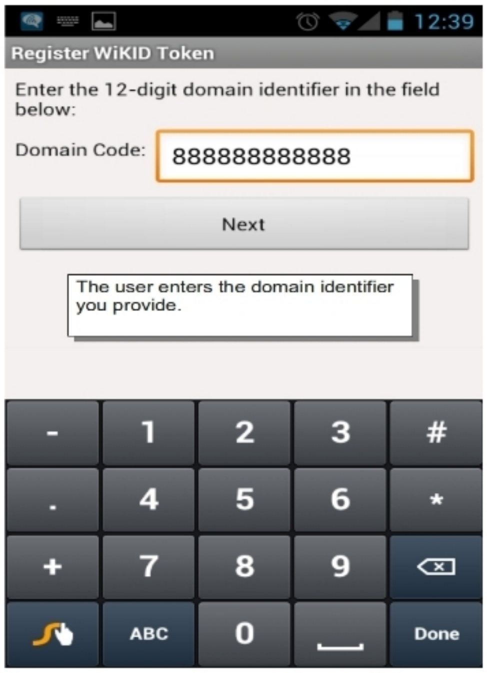
To begin using the WiKID Android Token, please enter and confirm a passphrase below. This passphrase will be used to encrypt your device keys and token configuration. If you forget your passphrase it cannot be recovered and you will need to delete your token and start again.

Passphrase:

Confirm Passphrase:

Create Token

The user is prompted to secure the token with a passphrase.



12:39

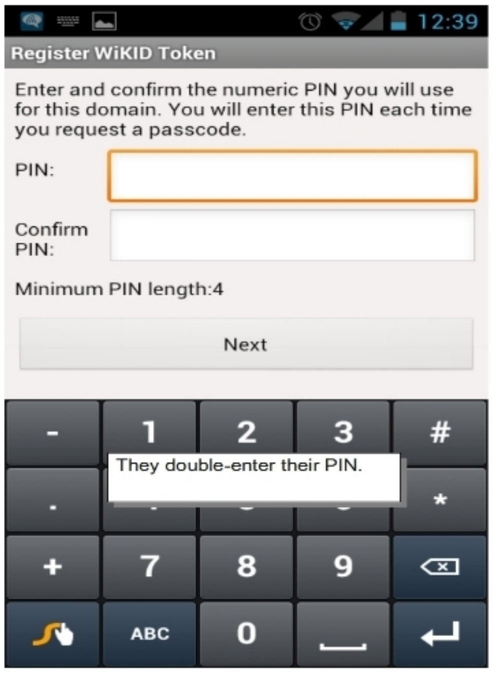
Register WiKID Token

Enter the 12-digit domain identifier in the field below:

Domain Code: 888888888888

Next

The user enters the domain identifier you provide.



12:39

Register WiKID Token

Enter and confirm the numeric PIN you will use for this domain. You will enter this PIN each time you request a passcode.

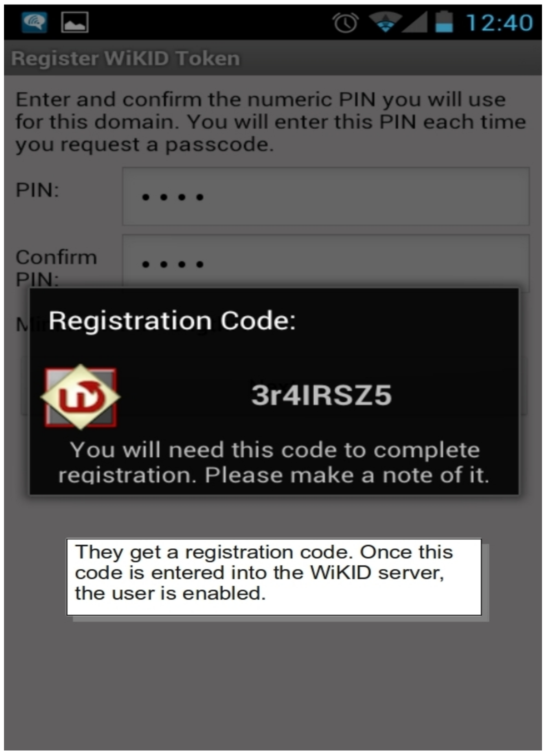
PIN:

Confirm PIN:

Minimum PIN length:4

Next

They double-enter their PIN.


12:40

Register WiKID Token

Enter and confirm the numeric PIN you will use for this domain. You will enter this PIN each time you request a passcode.

PIN: ••••

Confirm PIN: ••••

Registration Code:

3r4IRSZ5

You will need this code to complete registration. Please make a note of it.

They get a registration code. Once this code is entered into the WiKID server, the user is enabled.

ii. **Manual registration.** Of course, users can be manually validated by and administrator.. The key exchange is the same. The difference is in the final registration step. Instead of the user completing this step, the administrator of the WAS would associate the client device with the user ID and security domain and enable it. The manual process can be used when an existing user joins the WiKID system and continuity with the existing system is desired.

c. **Transport Encryption.** Without strong encryption, the WiKID solution would not be as secure as current two-factor systems. Simply put, the weakness of using an untrusted network channel, namely the Iinternet, is too great to overcome without strong cryptography. WiKID encrypts the payload of the passcode request and passcode reception as noted in the following sections.

d. **Passcode Request.** After the client device is registered with a particular domain, the client device is ready for passcode request. This process is the most important and will be repeated each time a user requires access to a particular network resource protected by the WiKID System. This process is as follows:

i. The user selects the domain for which he or she needs a passcode.

ii. The user provides the PIN for the domain, created in the registration process.
iii. The WiKID client application takes the following payload: [deviceID|PIN|serverID|random AES key] (separators are provided for readability purposes only) and encrypts this payload with the server public key provided in the registration process.
iv. The payload (password request) is sent encrypted to the server.
v. The server receives the request and decrypts the request with its private server key.
vi. The server looks up the appropriate requesting client device (via the deviceID for the security domain) and verifies the PIN.
vii. If the PIN is correct for the security domain and client device, the server creates the passcode payload by creating a passcode (length is administratively configurable) for the client device and encrypting this payload with the client public key. The payload is then transport encrypted with the random AES key provided by the client. If the domain supports host validation, the SSL certificate hash is also returned.
viii. The passcode is then time-stamped with an expiration window based on the parameters of the security domain (can range from seconds to days, depending on the sensitivity of asset).
ix. The WAS then returns the client request with the payload.

e. **Passcode Reception.** As the WAS payload is returned, the client device must follow the following process:
   i. The response must be decrypted with the random AES key sent in the request.
   ii. The payload must be decrypted with the client private key.
   iii. If the payload is verified, the passcode is displayed on the device and the user can use the passcode for access to the network service.
   iv. If the payload is not verified, an error is created and the process should be repeated.
   v. The WAS only allows passcodes to be valid for a specific time period and they may only be used one time.
   vi. If the domain supports host/mutual validation, the token client will fetch the SSL certificate from the targeted website directly, hash it and compare it to the hash retrieved from the WiKID Server. If the hashes match, the client presents the OTP and the clickable URL as validated. On supported platforms, the default browser is launched.

f. **Passcode Provision.** Once the passcode is received and decrypted, the user must then use the passcode. The user must connect to a Network Client of the WAS security domain (see *Network Clients* below) and provide the passcode and any other identifying information required by the Network Client. It is up to the Network Client to provide the credentials to the WAS for verification. Some examples of this action are as follows (there are many other uses of the WiKID Strong Authentication System; these examples are for illustration purposes only):

i. The user may enter the URL of a protected website, and provide a user ID and the passcode as the password.
ii. The user may provide a user ID and passcode as credentials for VPN access to a particular network.
iii. The user may provide a user ID and passcode as credentials for Google Apps.

g. **Offline Passcode Verification.** On wireless devices, a wireless network may not always be accessible, or may only provide limited access. Please note that with the prevalence for WiFI and support for WiFi in smart phones, in our experience, users rarely need this capability. When the Wireless Device can reach the Network Client but not the WAS server , the following process is followed (while seemingly complex, this process is mostly hidden from the user and similar to a typical CHAP challenge):
   i. Based on the user's action, the Network Client requests a *challenge code* from the WAS, rather than a passcode verification.
   ii. The WAS provides a large (usually 10-digit) code for the challenge code to the network device which, in turn, displays it to the user.
   iii. The user runs the WiKID Client Application on the Wireless Device in offline mode and enters the challenge code into the device.
   iv. The device assembles the following message: [client public key | PIN for domain | challenge code | offline secret | servercode] (again, separators are shown for readability) and creates a SHA1 hash which is alpha-decimal encoded. This offline secret is a 251 byte value is specific to each the WiKID client and the security domain.
   v. The user then returns to the process associated with the Network Client (such as web page login or terminal server login) and enters the resulting message as an answer to the challenge.
   vi. The challenge answer is provided by the Network Client to the WAS.
   vii. The WAS independently recreates the message for offline verification, and compares the results.

3. **Channels.** The passcode reception process and passcode provision process are conducted over two separate and distinct channels. One channel is the untrusted Internet network, while the other is an untrusted or trusted internal network between the Network Client and the WiKID server. In short, the passcode is received on one band and provided on another. The transfer between the bands is accomplished *manually by the user*. It is important to note that WiKID recommends strong encryption be utilized when transmission takes place on an untrusted network. Additionally, the WAUTH protocol utilizes both payload *and* transport encryption for client devices (as explained above).
4. **Network Clients.** Network Clients provide network services on the secured network channel. They can vary greatly in their implementation, depending on the requirements of the organization that deploys the WiKID Strong Authentication System. For example, a Network Client can be a firewall that provides VPN services to a partner extranet (via RADIUS/TACACS+) or a private website that provides sales support services (via WAUTH over SSL). The options are limitless as long as the Network Client implements either

RADIUS, TACACS+, LDAP (common for most network devices) or WAUTH (through a WiKID application component). Some scenarios are presented in the *Sample Implementation Architectures* section of this document. Whatever the requirements of the organization, Network Clients have certain aspects in common:

   a. **Registering Network Clients.** For a Network Client to become active within the WAS security domain, it must be registered. The registration of Network Clients is accomplished through the administration system. The following requirements are required for the registration of a Network Client:
      i. The Network Client must be related to a security domain.
      ii. The Network Client's properties (typically, name and IP address) must be provided.
      iii. The Network Client must be configured to communicate through a Protocol Module. Currently, this is either WAUTH, LDAP, TACACS+ or RADIUS.
      iv. In the case of SSL capable clients, a client SSL cert must be created by the WAS for each Network Client.

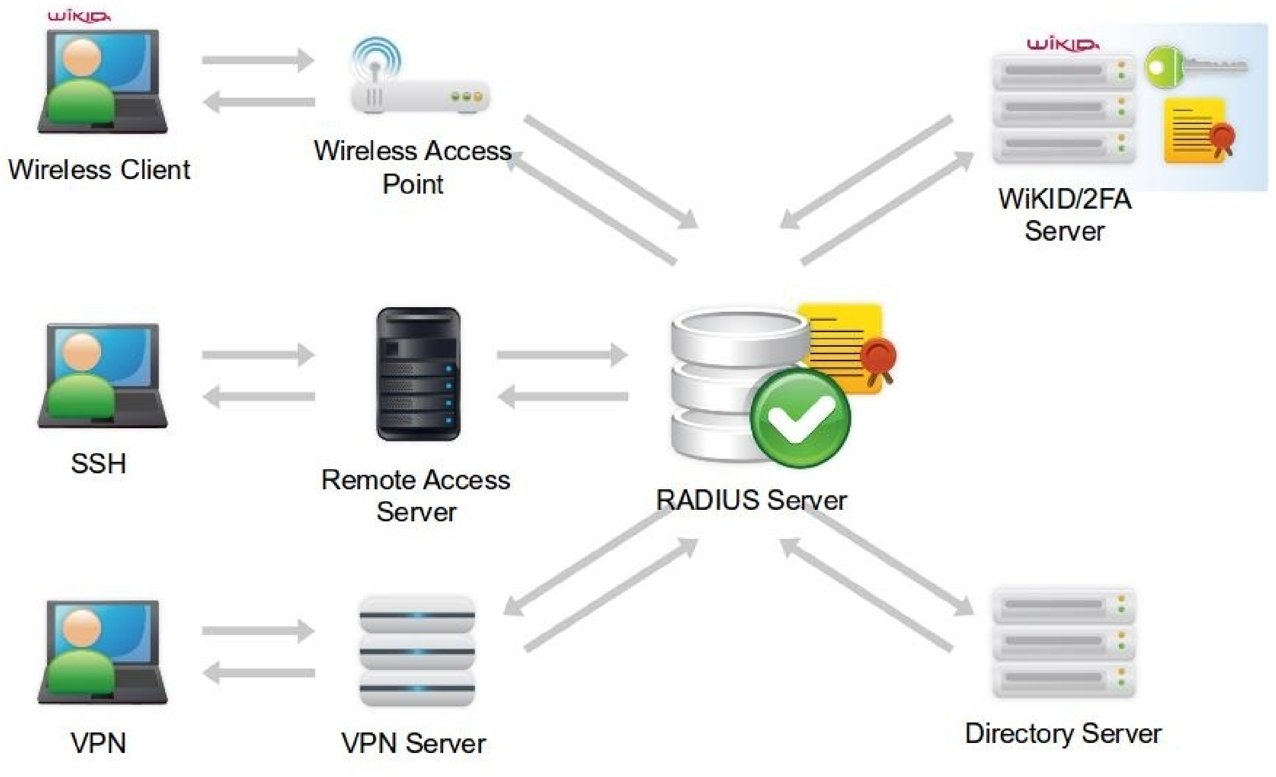
   b. **Verifying passcodes.** It is the responsibility of the Network Client to provide passcodes for verification by the WAS. The Network Client does not verify the code itself; instead it provides the code to the WAS through the chosen protocol. When the acceptance (or denial) of the code is returned from the WAS, the Network Client must act upon the acceptance (or denial). In the case of RADIUS, TACACS+, and LDAP protocols, the network devices are by design programmed to act on the acceptance or denial of the code. In the case of WAUTH Network Clients, the appropriate action should also be taken by the program through the WiKID application component.

   c. **Offline Verification Challenges.** As noted earlier in the *WiKID Clients* section of this document, the Network Client also plays a key role in the offline passcode verification process. The Network Client must be configured to provide (or display) challenges should the Wireless Device be offline. Additionally, when the user re-enters the response, the Network Client must be able to provide the challenge for verification by the WAS and act on the response accordingly. Like the Network Client's role in verifying passcodes, the offline verification challenge is similar and should be inherent in RADIUS/TACACS+ devices (via CHAP) and through the WiKID application component for WAUTH.

5. **Extended Terminal/VPN Services.** Network Clients may have clients of their own as is often the case with Radius servers. Please see the Typical Implementation section below.

B. Typical Implementation

In a typical installation, there will be a Radius server (NPS on Windows, Freeradius or a Cisco ASA, e.g.) sending authorization requests to a directory (LDAP or AD) and the authentication requests to WiKID.


Wireless Client
Wireless Access Point
WiKID/2FA Server
SSH
Remote Access Server
RADIUS Server
VPN
VPN Server
Directory Server

1.

III. Technical Considerations

A. System Considerations. The WAS runs on a hardened version of Linux. The operating system running on the WAS appliance is hardened in the following ways:

1. The WiKID engineering staff applies security kernel patches, system patches and application patches.

2. All processes run under an unprivileged user, including WiKID application processes, application server processes, protocol modules and database server processes.

3. All unnecessary services, including network services such as telnet, ftp, line printer, etc. are removed from the system, if possible, or disabled.

4. A firewall process is created and configured to remove access to unwanted and unneeded processes, applications and ports.

5. Access to any process - most importantly the terminal services, file transfer services and database administration services – must be conducted over an encrypted connection (SSH2) and negotiated through public key exchange. Additionally, inherently non-encrypted services (like database administration services) must be conducted over a SSH2-tunneled connection.

6. All internal WiKID services are conducted over an access controlled loopback service.

7. All file system, application and system services are set to deny access by default.

8. Buffer overruns, unchecked variables and other application weaknesses are protected.

9. Access to the cryptographic keys and database passwords is via a protected process. The keys and passwords never appear in plain text on the file system.

B. Network Considerations. In addition to the system considerations, certain network guidelines should be followed to insure safe operation of the WAS.

1. Generally, firewalls with NAT'ed public or semi-public addresses should be used to protect the WAS from port scanning and undesired probing.

2. RADIUS/TACACS+ should only be utilized on a private network. If required on a public network, the RADIUS protocol should be tunneled over SSL or similar encrypted transport.

3. Additionally, all other safe network computing practices should be followed.

IV. Administration Details

A. Administration Application. Most of the WAS administration is completed using the WiKID Administration Application. This entirely web-based system provides administration of WiKID Clients, Security Domains, Users, Protocol Modules, Network Clients and Preferences. In addition, the application provides access to logs, reports, statistics and help.

V. Cryptographic/Messaging Details

A. Overview. The following section provides detailed information concerning the encrypted messages that make up the registration and passcode request/response portions of the WiKID Strong Authentication System. This section is intended for security professionals and programmers who may wish to analyze the messages for strength or weaknesses against certain predetermined attacks.

B. Messages.

1. Registration

   a) When a security domain is created within the WAS two keys are generated for the domain: 1) the server private key {SK^1} and the server public key {SK^2}, these keys roughly relate to the security domain's public and private key respectively; however, terminology used by the NTRU algorithm does not match RSA's terminology precisely.

   b) At the initiation of the WiKID Client Applications the device creates a key pair: the client private key {CK^1} and the client public key {CK^2}.

   c) Communication is initiated by an unregistered device. The device communicates with the WAS based on the "server code" {SC} entered by the user. This code is either a zero-padded IP address representing the address on the intranet of a 12-digit alias within the wikidsystems.net namespace (for ASP services).

   d) After resolving the address {RA} of the target WAS, the device will generate a random AES key {RAK} and POST {CK^2}{RAK} to the WAS:

   e) The server will encrypt the following message:

```
CK2{[UTF encoded string][int][int][long][int][bytes]}
```

Corresponding to:

```
CK2{[domain name][minPIN][PIN TTL][deviceID({DID})][SK2 length][{SK2}]}
```

f) The typical length of the reply (after expansion) is approximately 3526 bytes depending on configuration and length of $\{SK^2\}$.

g) The reply is then AES encrypted with {RAK} provided in request.

h) The device should decrypt string with $\{CK^1\}$ and prompt for PIN, utilizing the minPIN provided in (e).

i) The PIN selection and new {RAK} is then encrypted with $\{SK^2\}$ and POSTed to the WAS.

j) f) The server decrypts with $\{SK^1\}$ and verifies. The server generates a 251 byte shared secret {SS}to be used in offline authentication. Then, the server replies with the following encrypted message:

   CK^2{[regcode {RC}{SS}]}

k) The reply is then AES encrypted with {RAK} provided in request.

l) The device should display the {RC} hashed with $\{SK^2\}$ and base-62 encoded.

m) The remainder of the registration must take place within the wired channel.

n) NOTES:

- The PIN and {RC} are *not* stored on the device to prevent theft.
- The client device is not enabled for authentication until the two-factor registration is complete on the second, wired channel.

2. PIN request/response

a) When a user requests a passcode for a security domain, the device prompts for the PIN for this domain.

b) The user enters the PIN for the security domain that was configured during registration for this domain.

c) The device encrypts the PIN with the $\{SK^2\}$ for the target domain and POSTs the ciphertext and new {RAK} to the WAS:

d) The PIN request should be decrypted with $\{SK^1\}$. The PIN is verified and if invalid the bad PIN counter is incremented. If the bad PIN counter exceeds the maximum for this domain, the device is disabled.

e) If the PIN is valid, the server will reply with the passcode encrypted with $\{CK^2\}$.

f) The reply is then AES encrypted with {RAK} provided in request.

g) The device should decrypt this message with {CK[1]} and display it.

h) NOTES:

- The passcode is not stored on the device.

VI. Conclusion

The WiKID Strong Authentication System is an alternative two-factor system that is as secure as existing two-factor authentication systems. The system is easy to administer and highly flexible. By employing asymmetric cryptography, the WiKID system eliminates the need for single-use devices, thus lowering the total cost of ownership of an advanced authentication system. Some of the major benefits of WiKID include:

- A low cost way to meet auditor's demands – the WiKID System ***starts at just $240 per year for 10 users.***
-
- A software-based solution that requires *no hardware* tokens
- Less expensive than a hosted solution and you maintain control over the keys to your kingdom
- Site or ***mutual authentication*** based on cryptography, not images or cookies thwarts network-based MiTM attacks
- Simple integration with all Enterprise VPNs and Directories through open standards
- ***Transaction authentication*** that is cryptographically distinct from session authentication
- Functionality that allows users to configure their own tokens securely, further reducing costs
- A system that won't require users to carry a "key chain of tokens"
- A *highly scalable, open-source* solution

In a world in which data security is critical, and an organization's information is its chief asset, WiKID effectively and economically addresses threats to data security. As the threats become more numerous and sophisticated the need for a new form of authentication system, one that employs newly available technologies is never more pressing. WiKID meets these needs with a unique, adaptive, cost-effective solution.

TWO-FACTOR AUTHENTICATION WITHOUT THE HASSLE FACTOR