

# **A security model supporting the legacy UserID:Passphrase** *the authentication model that won't go away!*

Eric Nielsen, Sylanro Systems  
Stu Jacobs CISSP, Verizon

## **Introduction**

Despite many technically superior choices, UserID:Passphrase pairs remain the most used means for assuring a user's communications are authentic. It has worked well enough. What has been a major security concern is how this data is entered and stored. Highly random passwords remain problematic for people too. Systems that persistently store these passwords can be corrupted, and the overuse of the password incrementally weakens the security it intends to assure.

This paper has a security model to co-ordinates the use of UserID:Passphrase pairs used at an application level with strong public-private key pairs at the device level. It is a model derived from VoIP security requirements published by this author.

## **Players in the model**

This model is described in terms of the players, human and machine, that are involved in the model. This is to clarify the identities involved, what is authenticated to what, and who retains what information.

The secure relationship starts out between a User and a Service Provider. There is one asymmetric element in this model that separates a User from a Service Provider. Otherwise they can interact as peers. The difference is that the "User" initiates the secure relationship from an "Endpoint" that does not store the User's passphrase, and the "Service Provider's" "Server" can automatically authenticate a message against a stored copy of the User's passphrase.

**User:** Typically a human or process acting as a proxy for an individual or organization.

**Service Provider:** The organization that administratively agrees to provide Users some service.

**Contract:** some agreement between a User and a Service Provider. In this case it is to provide an application layer service.

**User ID:** The identity associated with a user of the service.

**Passphrase:** The character string/biometric/other information that has been secretly shared between the User and the Service Provider prior to any network interaction.

**Service:** The network interactions that the user is authorized to access.

**Server:** The device (or logical group of devices) that provides a logical whole the service for the User. The server is assumed secure enough to have persistent access to registrar that can authenticate the User's passphrase.

**Endpoint:** A device used by a user to access the Service Provider's service. The endpoint is not assumed secure enough to store the persistent passphrase.

**Persistent Proxy Authenticator (PPA):** The PPA is an identity that is generated on a device. It has no rights except as a means to provide a cryptographically secure relationship for applications that run on that device. This device identity consists only of a public/private key pair that by itself has no rights. For the PPA to act as a proxy, a User must authentication in the context of the PPA. This gives the PPA the transient right to authenticate as if it had the rights of the User, and is the basis for creating session keys for network traffic authentication. Until the User withdraws this right, or policy causes it to expire, the PPA will persist as a proxy for the User without storing the User's authentication key/passphrase.

## **The Model**

The model consists of an identity to be authenticated, as well as authentication at the application, device, and network levels. It also adds a PPA in as a mechanism to map between these layers and sustain these connections.

*Application Level Identity:* How does the Server know whom it is interacting with? Initially the User and the Service Provider make an out-of-band contract. The contract, like the process of acquiring a phone number from a carrier today, is the basis for trust. This does not mean that the User has represented his/her/its identity absolutely. Instead, independent of the personal information provided, a UserID is chosen to identify the user in context of contracts with this service provider.

*Application Level Authentication:* The User enters the UserID and Passphrase pair as the basis for authenticating communications between the Endpoint device and the Service Provider's Server. Both ends of the communication must prove they know the shared Passphrase associated with the User ID. This must be done without any disclosure. Further, the material used to mutually authenticate the User <-> Service Provider connection is decoupled from the continuing authentication of communication between the Service Provider's Server and the User's Endpoint.

*Device Level Authentication:* Application level authentication (User <-> Service Provider) is the basis for creating a machine-to-machine layer relationship. The Server and the Endpoint must both have a public/private key pair. These key pairs do not require third party assurance of the identity nor require that any identity be bound to the key pairs as in a certificate. The devices exchange public key in the context of the application level authentication.

In other words, the authentication of the UserID:Passphrase pair enables the meaningful exchange of public keys between devices. Now these public keys can be used on behalf of the User and the Service Provider. In this model this creates a Persistent Proxy Authenticator (or PPA). Each Endpoint device can now act as the PPA for the User, and the Server acts as a PPA for the Service Provider.

*Network Level Authentication:* The public/private key pairs of the two PPAs are the basis to exchange session keys. These session keys may expire based on policy (time, kbytes sent, IP

address change, reboot) set by either end of the communications. Upon expiration of a session key a new key will be created based on the device level authentication. It is possible that the session key is generated by the Server and sent encrypted to the Endpoint device.

*PPA Persistence:* PPAs are intended to persist past a single use, over Endpoint and Server reboots, etc. The PPA key is expected to be stored securely on the device, inaccessible to any user. PPAs do not store the Passphrase of the User, and uses only its public/private key pair to establish session keys. If either the Endpoint or the Server determine that the PPA requires revalidation, the PPA can act as a proxy for a user no longer.

Thus, PPAs can reestablish a session key on behalf of the UserID without knowing the User's Passphrase. Local policies determine when a new session key is necessary. Either end of communications can decide to challenge the validity of the other end's PPA. To revalidate the PPA the UserID and Passphrase must be re-entered by the User. It is valid to have PPAs that do not expire. There is no negotiation in this process, either end can deny the other's validity anytime.

In this model the scope of the identity being authenticated is local. If the Service Provider happens to be a telephone company, for example, then it is administratively responsible for assuring the mapping between the UserID (an E.164 phone number) and the User. The mechanisms provided in this model are adequate to map the UserID to a device or devices. Local authentication by a recognized service provider can have real network-wide meaning.

## **Implementing this Legacy Authentication**

### **TLS**

This model can be mapped into TLS reasonably well. TLS is directly built into applications using an API. Maintaining these identities and authentication within a single application process is straightforward. It is important that authentication be maintained when a session is redirected to alternate Servers. One of the requirements associated with large-scale authentication is the need to reduce the burden associated with recovering huge numbers of sessions simultaneously after some disaster. Session key persistence across multiple hosts is possible in this model.

### **IPsec**

There is more than one way that IPsec might support this model, although it will not work "out of the box" as it stands today. It seems logical to use IPsec at the network level, and possibly IKEv2 (???) at the device level. There is a gap on how the network level security maps to security at the Application Level.

To implement this model with IPsec, session key establishment must occur in two different situations:

- a) when a pair of PPAs are currently associated with a UserID.
- b) where PPAs do not yet exist, and need to be associated with a UserID.

And there must be one final link. The application must know which UserID is the basis for authenticating each application level message.

If an IPsec Security Association expires, both sides must be able to renew that session key automatically based on the public key pairs. IKE can create a Security Association based on certificates. For case a) the exchange is initiated by public/private key pairs that have localized, temporal associations with UserIDs. It is a stretch to certificate semantics to include this kind of temporal association. A better semantic would be to have a self-signed certificate where the only identifier is the common name (CN), where the CN=<public key>. It is up to the application to determine whether a UserID is still associated with a particular Public Key. In other words, a PKI is unnecessary. To support this IKE must be flexible enough to use keying material in this way.

For case b) IKE would have to accept public keys from remote Endpoints/Servers based on the application level authentication. To accept the public key, IKE must allow a link to the application to validate the UserID:Passphrase pair, and thus authenticate the exchange of public keys for use.

Subsequent messages sent to/from an application should be associated with a UserID. Security Associations must use the IPsec SPI to make each connection unique. It is possible that multiple applications may share the IPaddress:port source/destination combination, thus the SPI is the differentiator. IPsec could maintain the UserID to public key relationship, or the application could maintain it. The main issue is that the application knows the UserID the message was authenticated for, and that the application can choose to reestablish a new session key or PPA relationship at any time.

An advantage of this model is that public/private key pair can be generated privately and dynamically at the Endpoint (even if it requires many minutes of real time to generate). It can take some time to acquire enough randomness to generate a new key, it would happen infrequently. This approach allows the private key of the PPA to be more secure. The key is never disclosed to any other device. The only way to extract the PPA's private key is by gaining "root" access on the device. There can be multiple concurrent PPAs on a device.