BIOMETRICALLY ENHANCED SOFTWARE-DEFINED RADIOS*

Joseph P. Campbell, William M. Campbell, Douglas A. Jones, Scott M. Lewandowski,
Douglas A. Reynolds, Clifford J. Weinstein
MIT Lincoln Laboratory
Lexington, Massachusetts 02420-9185 USA
{jpc, wcampbell, daj, scl, dar, cjw}@ll.mit.edu

ABSTRACT

Software-defined radios and cognitive radios offer tremendous promise, while having great need for user authentication. Authenticating users is essential to ensuring authorized access and actions in private and secure communications networks. User authentication for software-defined radios and cognitive radios is our focus here. We present various means of authenticating users to their radios and networks, authentication architectures, and the complementary combination of authenticators and architectures. Although devices can be strongly authenticated (e.g., cryptographically), reliably authenticating users is a challenge. To meet this challenge, we capitalize on new forms of user authentication combined with new authentication architectures to support features such as continuous user authentication and varying levels of trust-based authentication. We generalize biometrics to include recognizing user behaviors and use them in concert with knowledge- and token-based authenticators. An integrated approach to user authentication and user authentication architectures is presented here to enhance trusted radio communications networks.

1. INTRODUCTION

Software-defined radios (SDR) and cognitive radios (CR) [1] are expected to provide powerful new capabilities. To realize this promise, these radios and their networks will need user authentication. Authenticating users ensures that only authorized personnel have access to their radios and networks. Furthermore, sensitive radio-operations and access to resources will be limited to authorized personnel.

Users can be authenticated based on something they *know, have, do* and/or *are*. Recognition based upon something you are is conventionally known as biometrics—*automatically recognizing a person using distinguishing traits*. We generalize biometrics to include recognizing user behaviors, which can be used with conventional knowledge- and token-based authenticators. Some of these authenticators naturally operate continuously and transparently to the user.

The authentication architecture supports user authentication by communicating authentication information between various processes and by orchestrating the overall authentication processes. We generalize conventional authentication architectures to support varying levels of authentication or trust and continuous user authentication.

In the following sections, we develop means for authenticating users, a compatible architecture, and we discuss their combination in SDR and CR.

2. USER AUTHENTICATION

We present various means of user authentication, introduce generalized biometrics, and illustrate continuous and confidence-based user authentication. As shown in Figure 1, the four pillars of user authentication are: *knowledge* (e.g., PIN or password), *tokens* (e.g., key or badge), *behaviors* (e.g., usage patterns or outcomes), and *traits* (e.g., voice or fingerprint). The combination of all four pillars provides the strongest user authentication. Biometrics authenticates users, as opposed to something they know (which can be forgotten or compromised) or possess (which can be lost or stolen). Unlike knowledge- and token-based authenticators, however, the inability of users to transfer biometrics can

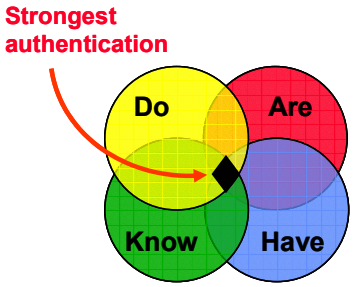

Figure 1: The four pillars of user authentication.

*This work was sponsored by the Defense Advanced Research Projects Agency under Air Force contract F19628-00-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

lead to difficulties, e.g., emergency transfer of operation of a radio with biometric access control to an unenrolled user. Knowledge- and token-based authenticators can be used to authenticate users in these situations to solve this difficulty.

Popular biometrics include voice, face, fingerprint, and iris (see http://www.biometrics.org/ for others). Voice and face biometrics (possibly in combination) are well suited to radios that already incorporate microphones and cameras. Some biometrics lend themselves to continuous user authentication (e.g., to guard against lost or captured radios) and varying levels of trust. For example, voice verification can be used to continuously authenticate a user while they are talking (this can be useful if the voice quality makes it difficult for the interlocutor to determine a change in operators). Figure 2 shows an example of an authentication process over time with varying levels of trust [2]. This example begins in a state of provisional trust and, over time, proceeds in continued states of provisional trust and then to a trusted or untrusted state. While in a state of provisional trust, benign operations can be performed, whereas sensitive operations would require a trusted state.

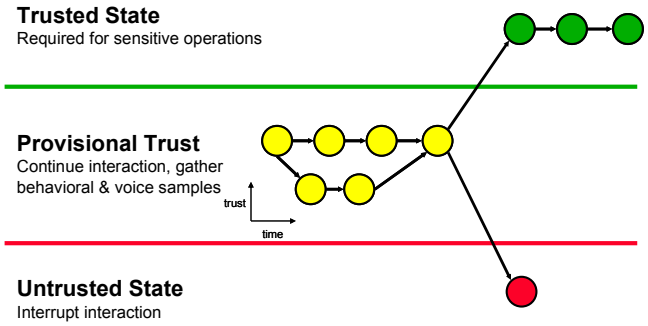

Figure 2: Continuous user authentication and trust.

Biometrics can provide user conveniences, such as recalling preferences, biometric logins, and screen locks, which can also guard against compromised equipment losses (e.g., disable a radio that is left behind). We generalize conventional biometrics by learning the users and recognizing their distinctive behaviors.

Behavior-based user authentication recognizes users via their actions, interests, tendencies, preferences, and other patterns. Examples of distinctive behaviors include 1) *how* a user does something (e.g., speed and pattern of typing, stylus angle and intensity, use of menus vs. keyboard shortcuts), 2) *what* a user does (e.g., patterns of applications use, program features used, patterns of collaboration), and 3) what a user *causes* to happen (e.g., sequences of system calls, patterns of resource access). These behaviors not only include a user's local actions, but also network interactions and outcomes. Behavior-based user authentication, like voice verification, has minimal adverse impact on mission. The authentication is inherent and transparent; there is continuous mode operation and modest resource utilization; and user acceptance is likely to be high. Monitoring these behaviors can be combined with situational awareness to fuse multiple factors into the authentication process.

A cognitive approach allows for many interesting possibilities. First, the threshold to reach the trusted state of user authentication can be adapted based upon situational, environmental, and mission awareness and the risk of the requested operation (e.g., benign volume adjustment to sensitive security operations). Second, authentication can be performed over time combining available information—voice communication, mouse/stylus movement, dialogue structure, etc.

Some issues and questions in biometric deployments are 1) remote vs. distributed vs. network enrollment and verification, 2) where are user models created and stored, 3) how are models maintained and updated, 4) how is enrollment conducted, 5) how are models bound to users, 6) what is the tolerable verification time or rate, 7) how are models of new users distributed and their integrity assured, 8) are there accuracy or policy requirements, and 9) what is the architecture to support the biometrics.

The integration of the user authentication pillars with an authentication architecture for software-defined radios and networks is explored and discussed next.

3. ARCHITECTURE

Reconfigurable software-defined radio (SDR) implementations must be based on an architecture that supports many different functions in addition to the core functionality of providing adaptive and dynamic communication services. Some of these functions include:

- **Discovery, negotiation and adaptation.** It must be possible to locate radios and services, and to determine which users are controlling each radio. Clients also need to be able to negotiate for access to services and limited resources and to specify the quality of service (QoS) required to adequately support the current task. The platform as a whole must adapt to status changes and faults, both malicious and benign.
- **Management.** System administrators must be able to manage terminals, services, applications, users and their profiles, and security information (such as credentials and privileges) from a central location. Users and system administrators may need to manage some of these entities locally as well.
- **Upgrades and configuration management.** It must be possible to reconfigure entire networks and individual radios by changing settings and by adding/modifying functionality using downloadable software modules. Configuration changes must be performed in a controlled and secure manner.

- **Security.** Radios and networks need to mutually authenticate, and users need to authenticate to radios. Radios must also "authenticate" to users – users must have confidence that the radio they are using is theirs, and that is has not been physically compromised or tampered with; however, we do not address the authentication of radios to users since the necessary assurances are generally provided through non-cyber means, such as tamper resistant hardware. Since all of the authentication relationships are transitive, these basic forms of authentication implicitly authenticate users to networks and the services that reside on those networks, and provide radio to radio and user to user authentication. In addition to authentication, resource access must be managed, and data confidentiality and integrity must be ensured.

In a real-world SDR implementation, these functions will not necessarily be implemented as discrete components that have a well-defined location in the instantiation of the SDR architecture. Rather, components responsible for implementing these functions can be distributed throughout the network and across the radios relying on the network; in fact, the implementation of these functions can be partitioned among multiple components that cooperate from distributed locations to provide the required functionality. A key difference between SDR architectures and the standard architectures used by wired computing hosts is that an SDR network has true functionality and can contribute to all of these functions, whereas a traditional wired network impacts only discovery and negotiation.[1]

While all of these functions are important and impact the overall SDR architecture, here we focus on the authentication-related aspects of the architecture. Note that many of the functions listed impact authentication in one way or another. For example, devices and networks must negotiate to learn requirements and supported protocols, users and their tokens must be managed, and new security functionality may be dynamically added to an in-use SDR system.

3.1. Network Security Architecture

Whereas the network architecture is of paramount importance for enabling general SDR functionality, the network itself plays a relatively small role in the realization of secure authentication. However, it is likely that the network would play a major role in implementing related functionality, such as user and privilege management and providing data confidentiality.

The network architecture directly supports the function of authenticating radios and networks. Radios and networks must mutually authenticate in order to form a trusted-base on which other levels of authentication can be layered and to ensure that the most basic network services (such as resource discovery) are accessed only by parties that are trusted at least nominally. It is especially important that the user not be part of the lowest level of authentication since user authentication relies on services that must be afforded at least a basic level of protection; since the user is not yet authenticated when these services are used, trust must come from the authentication of other entities.

For networks with closed user communities, such as those found in certain military environments, this requirement can be met implicitly through the use of shared symmetric keys. Using this approach, each authorized device is seeded with a common key and the network also has knowledge of this key. Senders encrypt the data they send using their key, and receivers attempt to decrypt it using their key. If the decryption yields a well-formed message, then the data must have been encrypted and decrypted using the same key, which implies that the data was transmitted by an authorized party. While this approach has limitations (for example, it creates the opportunity for an adversary to use certain styles of denial-of-service attacks, and it does not provide per-client data confidentiality), it is a simple and efficient solution to authenticating closed user communities. A significant problem with this approach is that it is impossible to securely remove a client from the group of trusted clients using in-band communication. Removing a client requires distributing a new key to the other clients (that will remain trusted), which cannot be done in-band without the key being accessible to the client that is no longer trusted; using the shared symmetric key approach, rekeying requires an out-of-band channel (such as physical access) to each machine that is to remain trusted – a daunting proposition in large communities. Although it is not possible to remove an individual client from the network with this single-community approach, a client can be designated as untrusted. However, that does not prevent the client from intercepting all traffic on the network, nor does it prevent the untrusted host from masquerading as a trusted host.

On open networks with diverse user communities, it is not feasible to use a shared symmetric key approach. In these environments, approaches relying on public-key encryption can be used. With such schemes, the network has a public/private key pair, and each client has a unique public/private key pair. When the network wants to send data to a client, it encodes it using the client's public key, and the client can then decode it with its private key. Communication from a client to the network occurs in a similar manner. Client-to-client communication occurs

[1] Some networks contribute to security through the use of hardware link encryptors, but these architectures are generally used only for networks with special security requirements since the deployment and management costs of software-based or per-host security (e.g., through encryption) outweigh the incremental benefits gained from integrating security with the network infrastructure.

through the network; that is, the sender encrypts the data using the network's public key sends the encrypted data to the network along with the necessary addressing information. The network then decrypts the data, encrypts it using the receiver's public key, and forwards the encrypted data to the receiver, which can decrypt it using its private key. Note that this approach assumes that the network can be trusted to route messages appropriately, since there is no direct sender-to-receiver authentication. A disadvantage of this approach is that secure broadcast is not possible, since clients do not share any common secrets. This requires broadcast to be implemented as a series of point-to-point communications, which wastes bandwidth and power, since multiple physical sends are required when one logical send would suffice.[2] The key advantages of this approach are that clients do not need to trust each other and that a single client can easily be removed from the pool of trusted hosts, simply by invalidating its public key (a task the network is well-positioned to handle). The public-key encryption approach can also be used for closed communities.

Although beyond the scope of this discussion, note that these encryption-based authentication schemes also provide a degree of data confidentiality and integrity that may be adequate for many applications.

3.2. Radio Security Architecture

The radio plays a greater role than the network in the realization of end-to-end system security. This is largely due to the fact that the radio is responsible for authentication on both the network and user sides. A notional radio security architecture is depicted in Figure 3. Note that this architecture is not meant for direct instantiation; rather, it shows the logical structure of a system built using our cognitive authentication approach.

The high-level security-related components of the SDR security architecture we propose are: a secure communication interface; a biometric subsystem, comprised of one or more biometric sensor and biometric processor pairs; an authentication API; an authentication user interface; a security manager; and a security API. These components collaborate to provide end-to-end security between all of the participating entities in the overall system, including the network, radio, applications and services, and users.

The sections that follow provide details pertinent to each component of the security architecture, as well as to the applications, which are a consumer of the services offered by these components.

[2] If the network provides a broadcast service instead of requiring clients to generate the multiple point-to-point messages, bandwidth is still wasted but the power-consumption issue is diminished in severity since the network infrastructure itself it usually not power-constrained.

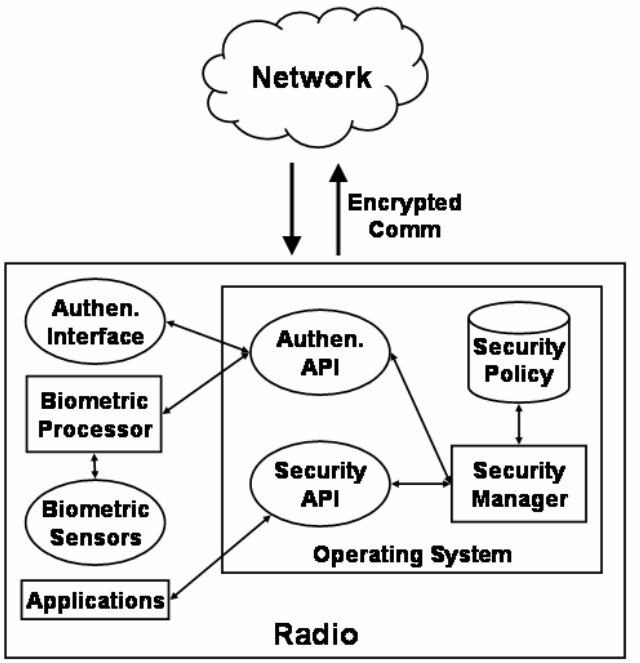

Figure 3: Notional radio security architecture.

3.2.1. Secure Communication Interface

The overall security of the SDR platform is predicated on the security of the data flowing to and from the radio. As such, protecting information in transit is critical to meeting the system security needs. Furthermore, as discussed earlier, secure communication can be used to perform the authentication that is required between the radio and the network. The implementation of this secure communication channel is beyond the scope of this paper since the details of its implementation are not pertinent to user-level authentication, so long as the expected functionality is provided.

3.2.2. Biometric Sensors

Section 2 discussed a number of biometric approaches to user authentication. These approaches all require some form of hardware input device to gather the required information about the user to be authenticated. For example, fingerprint recognition requires a fingerprint scanner, user voice recognition requires a microphone, and user behavior monitoring requires a traditional user input device (e.g., a keyboard or mouse). These hardware devices must be an integral part of the SDR platform and must communicate with their software counterparts over a secure channel. Many of these devices are high-bandwidth (although the utilization is frequently very bursty) so the channel connecting the hardware and software must be capable of supporting the data transfer requirements without an undue performance impact on the device's core functionality.

3.2.3. Biometric Processors

Once data has been gathered from a biometric sensor, it must be processed to determine user identity (or other user characteristics that the sensor has been designed to assess). Such processing can occur either in specialized hardware or in software. Although the use of specialized hardware provides the advantages of increased tamper-proofing and higher performance, the complexity of managing updates and modifications to the functionality of the hardware often outweighs these benefits. Given that the overall security of the radios and the services provided by radios and the network is built on other software components, implementing the biometric processor in software does not fundamentally diminish the overall security of the system. Although not shown in Figure 3, most biometric processors require access to a database that contains information required to authenticate users, such as biometric models or templates, profiles, or logs of past behavior.

3.2.4. Authentication API

Once the biometric processor has assessed the identity of the user, it needs to inform the operating system so that the appropriate resource use policy can be enforced by the operating system; this is done using an authorization API. Whereas this interface is quite simple in most operating systems, a system supporting the robust functionality we desire to exploit in our cognitive system must extend traditional authentication APIs by making authentication continuous and by providing a confidence measure.

Modern operating systems perform authentication in a discrete and binary manner. A user authenticates once, at the beginning of his session, and remains authenticated until the session is terminated, or an intervening event occurs, such as the activation of a screensaver. In addition, user authentication is binary; there are only two outcomes to a user authentication request – either authentication succeeds and the user receives the full rights and privileges associated with his identity, or the authentication fails and the user receives no privileges. Our cognitive approach to authentication, in contrast, authenticates a user throughout his session, either on a periodic basis, or in direct response to a security-critical user request or an external stimulus that indicates a possible compromise of security. Further, each time a user is authenticated, a confidence metric is assigned to the outcome authentication event. This measure of confidence can be used by an application to vary the functionality afforded to a user depending on the confidence of the authentication. For example, if the voice authentication system is only 70% sure that a user is who he claims to be, the system may restrict that user from using the most sensitive functions on the radio until a more solid authentication can be performed, perhaps using a second authentication factor.

3.2.5. Authentication User Interface

Throughout the authentication process, it may be necessary to prompt the user for addition information or to take an additional action. In addition, in some environments, the user should be kept informed of his authentication status (note that there are risks with disclosing information about the authentication system; for example, an adversary could learn a user's identity by "shoulder surfing" or an adversary who has gained control of a device could use the displayed information to determine how the authentication system operates in order to exploit it). This interface could be integrated with the standard interface of the computing platform, or it could be a standalone interface tailored for authentication status reporting.

3.2.6. Security Manager

Once the user has been authenticated, the authentication API reports the outcome to the security manager. The security manager is responsible for maintaining a mapping between operating system objects (e.g., files, processes, sockets, etc.) and the privileges that each user can exercise on those objects. One of our goals is to build atop COTS operating systems; thus, we do not specify how the mapping between objects and user privileges should be made or how checks on these privileges should be performed before a user is allowed to access a resource.

However, both continuous authentication and confidence-based authentication require modification to the standard security subsystems found on common operating systems. These modifications can be integrated with the existing security subsystem, or can be added as a translation layer that mediates between clients and the security system (this could be done using software wrappers, for example). Confidence-based authentication requires that privileges be mapped to objects considering not only the authenticated principal (i.e., user) but also the confidence of the authentication. Thus, instead of maintaining *{object, principal, privileges}* tuples, the operating system must maintain *{object, principal, confidence, privileges}* tuples. When the security system needs to determine the privileges a principal has for a given object, it finds the tuple with the highest *confidence* value below the user's currently assigned confidence level, which can vary throughout the session. While this may seem to complicate privilege management unduly, we do not believe this to be the case; appropriate assignment of fine-grained privileges to certain high-level objects (e.g., key data stores and applications) implicitly impacts the privileges that a principal has for dependent resources.

The mechanics of implementing continuous authentication are not terribly difficult, although it raises a number of semantic questions we have not yet completely explored. When a user's authentication status changes, the security manager is notified, and future resource access requests will be made based on that information. For

example, a user could open a file for editing. When he does so, the security subsystem verifies that the client has permission to open, read, and write to the file, and, assuming he does, it allows the operation to proceed – the file is opened, the current contents are displayed, and the user is allowed to edit the file. Suppose that while editing the file, the user's permission to write to the file is revoked. Now, the user will be unable to save his work, which is a suboptimal scenario and does not meet the user's expectations (he should not have been allowed to edit the file if he cannot save his changes). This simple scenario is illustrative of the challenges associated with dynamic privileges. These difficulties could be dealt with gracefully by applications that were written with consideration of the possibility of dynamic privilege updates, but we want to run existing applications on our SDR platform. Note that similar situations can arise on systems that do not use continuous authentication, since the privileges granted to a principal for a given resource can be changed at any time, and most systems respect the current privileges when making an allow/deny decision on an action (instead of relying on the privileges in effect when the object was opened for access). However, the issues associated with dynamically changing permissions are much more important when considered in the context of our cognitive SDR platform because changes in privilege are expected to be common, whereas they are very uncommon in traditional operating environments.

When a resource access fails due to inadequate confidence in the authentication (as opposed to failing because a principal is not authorized to access a resource), the application should not be immediately notified of the failure. Instead, the security manager should work to establish the required credentials. Using the authentication API, it can request stronger authentication of the user; this may trigger an explicit request to the user via the authentication interface. If the proper level of authentication can be established, the resource access can proceed as expected, transparent to the application. If the appropriate confidence cannot be established, then the operation can be failed (hopefully the application will handle the failure gracefully, but this is not always the case, especially when the application pre-asserts that certain privileges exist before exercising them), or the system can attempt to establish the required confidence after a suitable delay, perhaps once external conditions change such that they are more conducive to authentication.

3.2.7. Security API

Applications designed for the cognitive SDR platform can be written to leverage the advanced functionality provided by our continuous, confidence-based authentication. In addition, they can improve overall system performance by communicating their authentication needs so that the platform does not expend resources providing a degree of authentication not required and so that the platform can work to ensure that the required authentication has been performed by the time it is needed. For example, a device may be configured to provide 100% confidence in authentication using two factors, such as voice and typing pattern. However, if no application being used demands the extra confidence afforded by the voice authentication system, then the microphone can be disabled and speech data does not need to be processed, freeing memory and CPU cycles for other tasks, as well as extending battery life. The security API enables applications to express such information to the security manager. The security API also needs to interact with the secure communication subsystem so that it can inform the security manager when the trust relationship with the network has been broken.

3.2.8. Applications

Although our platform has been designed to accommodate existing applications without modification, new applications can be designed to leverage the capabilities exposed by the security API to improve the user experience and to improve overall system performance. We have not yet explored the full set of these possibilities, but designing applications to leverage this architecture is critical to its success.

Legacy applications will automatically benefit from continuous authentication, but will be completely unaware of confidence-based authentication. Therefore, the platform will need to define a confidence level at which the user is considered authenticated, and any level of confidence below that will cause the user to be considered completely unauthenticated by legacy applications. Applications that are aware of confidence-aware authentication, in contrast, can enable functionality or access to data based on the confidence in the user's identity.

4. CONCLUSIONS AND IMPLICATIONS FOR CR

We presented an integrated approach to user authentication and architecture to enhance trusted radio communications networks. User authentication, via generalized biometrics, can be combined with other authenticators to provide continuous, flexible, and strong user authentication. This biometrically enhanced authentication system approach can be extended to become part of a cognitive radio system which learns about users, situations, and surroundings and takes appropriate proactive or reactive actions. The area of learning emphasized here has been generalized biometric authentication, where the users' distinctive behaviors and traits are learned and recognized. An advanced cognitive radio will also learn about and take action based upon user preferences, availability of network resources, and other elements of the situation and surroundings.

5. REFERENCES

[1] J. Mitola & G. Maguire, "Cognitive Radios: Making Software Radios More Personal," *IEEE Personal Communications magazine*, Aug 99.

[2] T.J. Hazen, et al., "Integration of Speaker Recognition into Conversational Spoken Dialogue Systems," *Proc. Eurospeech*, 2003.