International Journal of Information & Network Security (IJINS)
Vol.2, No.1, February 2013, pp. 1~16
ISSN: 2089-3299

Securing Wireless Grids: Architecture Designs for Secure WiGLET-to-WiGLET Interfaces

Tyson Brooks*, Lee McKnight*
* School of Information Studies (iSchool), Syracuse University

Article Info

Article history:

Received Sept 20th, 2012
Accepted Oct 26th, 2012

Keyword:

Wireless Grid
Security Architecture
WiGLET
Cloud Computing
Service-Oriented Architecture

ABSTRACT

Wireless grids are ad-hoc dynamic sharing of physical and virtual resources among heterogeneous devices. In order for wireless grids to be secure, the main communication device called a WiGLET must have precise control over the distribution of information exchanges amongst the devices with the ability to accommodate new devices and resources spanning different levels of trust. A lack of trust in the WiGLET-to-WiGLET interface hinders its ability to protect information and to collaborate with unanticipated devices on unanticipated events at the time that such collaboration is needed. Not only do the WiGLET's have to agree on the level of sensitivity of data, but they need to be able to determine if they all understand that sensitivity in the same manner. This article identifies three different types of wireless grid architecture's and specific security considerations for each type around the WiGLET.

Corresponding Author:

Tyson Brooks, PMP
School of Information Studies (iSchool)
Syracuse University
Email: ttbrooks@syr.edu

1. INTRODUCTION

There are two significant events occurring at this point in time that are affecting communications throughout the world. First, there has been increased activity to improve security of network and computers systems on a global scale, both to ensure reliability as the internet becomes more open due to social networking sites and to ensure the confidentiality of these data exchanges [1]. Second, there is movement to adopt wireless technologies, especially wireless grids, due to its agility in using existing devices and infrastructure [47]. Wireless grids are starting to emerge as new type of resource-sharing network which will eventually connect sensors, mobile phones, and other edge devices with each other and with wired grids [31].

The wireless grid extends grid resources to wireless devices of varying sizes and capabilities such as sensors, mobile phones, laptops, special instruments, and edge devices [25, 32]. These devices might be statically located, mobile, or nomadic, shifting across institutional boundaries and connected to the grid via nearby devices such as desktops [3, 33]. [3] identifies theses grids as an augmentation of a wired grid that facilitates the exchange of information and the interaction between heterogeneous wireless devices. These grids will enable shared resources among dynamic groups or social networks of computing and communication devices and are composed of objects and resources with individual profiles that are assigned a specific status relative to similar objects and resources [30, 31]. Wireless grids can take ubiquitous computing to the next level by providing seamless wireless extensions to the wired grid. The security architecture for these types of grids are of importance because they can be deployed to provide autonomous nodes that communicate with each other in a decentralized manner and how each node of the network connects others with wireless links, and acts as both a host and a router in sending and receiving data.

Under emerging wireless grid architecture, however, such network-based security models are far from adequate. These new systems will be about net-centric information sharing and collaborating business

functionality which will become service-enabled and exposed to external wireless clients via standard web services type protocols. These wireless clients, which themselves may be applications, will dynamically discover services and make real time use of their code and data. Their services will be inherently location independent, not necessarily bound to a physical location which can change over time as services are relocated or for fail-over reasons. Since wireless grid clients and service providers may belong to different physical networks or even different service providers, these networks and/or organizations may be governed by entirely different security policies. This article identifies three different types of wireless grid architecture's and specific security considerations for each type around the WiGLET.

2. SECURING THE WIGLET-TO-WIGLET ARCHITECTURE

The paradigm shift towards wireless collaboration and composition also brings fundamental changes to security architecture. A security architecture encompasses IT security and data privacy requirements as well as solutions within the context of an over-arching enterprise architecture [5]. Most existing security solutions today are based on the assumption that both clients and servers are located on the same physical (e.g. local area network [LAN]) or logical (e.g. virtual private network [VPN]) network. They generally rely heavily on perimeter-based security such as demilitarized zone (DMZ), firewalls, and intrusion detection to thwart security threats [44]. Similarly, security policies behind those solutions are also to a large extent perimeter-based. For example, obtaining access to an application usually requires creation of a new user account on the machine or network where the application is installed, plus granting the user physical access to the facility where the machine or network is located. By contrast, application/system level security is usually regarded not quite as critical as network security and oftentimes enforced simply by a username/password. However, for a WiGLET within a wireless grid, this may not be the case.

A WiGLET (see Figure 1) is a wireless semantics-aware unit of workload division, computation off-load and data related to the processing task which interacts directly with the wireless grid user interface (UI), the application program interface (API), the connection, messaging, permissions and metadata (CORE operations) employing resource management and accounting of devices directly through the interface layers of a wireless grid architecture [46, 47].

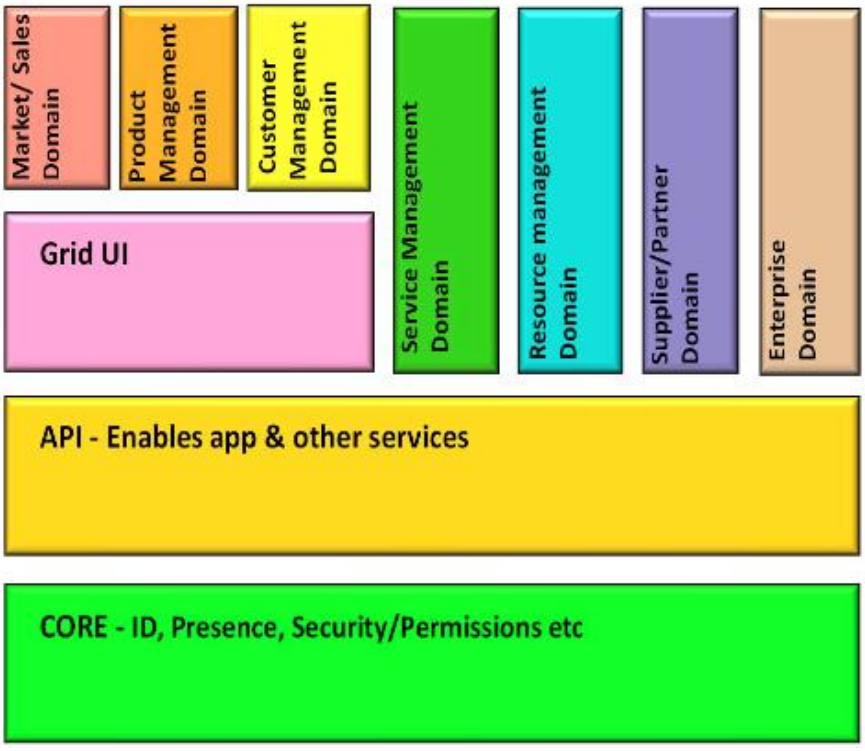

Figure 1. WiGLET
(Source: Wireless Grids Corporation/WiGiT)

The vertical boxes in Figure 1 represent 'Edgeware' applications that reside on a user interface, which in turn reside on an API, and may represent dozens or hundreds of different sorts of mini-programs that enable different kinds of resource sharing and functionality [34]. Wireless grids "edgeware" technology

sits at the outermost limits of existing networks, allowing all facets of a user's environment: printers, mp3, documents, photos, cell phone, PC and new plasma TV, etc. to be interoperated and shared easily [29, 47]. Other network hardware, software, services, and content may be controlled and shared through the wireless grid 'edgeware'; however, if those 'edge' resources are in a relationship with other hardware, software, and services which are part of the wireless grid, they may function as if they were fully cognitive [34]. The Core components are embedded in certain devices or sensors depending on their capability, makes every device a node on the wireless grid, is extremely 'light', easy to embed on a wide range of different kinds of equipment and users are allowed to share and manage the digital resources at their fingertips through applications of the architecture's eight core components [34].

[34] discusses that the wireless grid architecture core components handle four primary functions: management of identification (ID) and presence, permissions management, data transfer ability, and API/interfacing. These are the elements that make the grid-enabled ecosystem possible and the layers above the core are comprised of the API which enables connections with other applications and services, the User Interface (which may or may not be necessary depending on the device upon which it sits), and finally the edgeware applications. Once a grid is established then resources can be published or accessed across the grid, enabling the infinite functional possibilities of the Grid technology.

Securing a WiGLET to WiGLET interface will rest on the principles of both protecting data and communication devices in the Wireless Grid. Such interoperability amongst WiGLET's is envisioned as shared devices with controlled and changing environments spanning trust levels within the grid. WiGLET's need to share not only data but also interactive exchanges of secure devices. A major issue with this problem space is the issue of revocable sharing, addressing the need to revoke access to data and/or resources formerly available to grid users who have either voluntarily withdrawn from the wireless grid or whose access has been involuntarily severed. WiGLET's will also need to work across multiple trust levels. The first step in securing this interface is to identify the functionality of the security token and the security standards/policies (authorization, delegation, authentication, message protection and message format [21, 51] as displayed in Figure 2, which will need to be enforced.

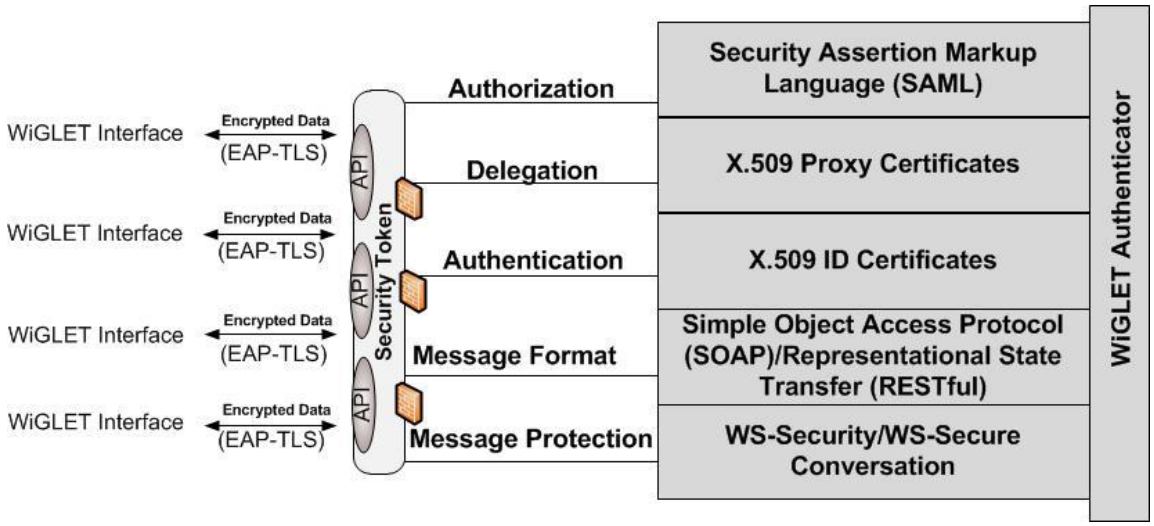

Figure 2. WiGLET Security Layer

Identified, authenticated and authorized WiGLET's should be able to access and process any service and data required to meet its specified purpose. Identification consists of the process that an information system uses to recognize an entity. Access control is accomplished by the resource owner (the provider) authenticating the entity seeking access (the requestor) to the resource, and then determining whether that entity is authorized and authenticated to access a WIGLET. As WiGLET's transmit encrypted data to other WiGLET's through defined application programming interfaces (API), it must first pass through a security token.

Authentication for WiGLET's must involve strong encryption to prevent eavesdropping, and must involve mutual authentication to ensure that sensitive information is transmitted only over legitimate wireless grid networks. WiGLET's must also have strong authentication standards to meet the challenges for strict encryption and authentication requirements. Strict encryption and authentication requirements are met by the use of the extensible authentication protocol (EAP) directly over a link layer protocol [24]. EAP is essentially a transport protocol that can be used by a variety of different authentication types, known as EAP

methods. EAP was standardized by the Internet Engineering Task Force (IETF) in March 1999 [6]. Among the EAP methods developed specifically for wireless networks are a family of methods based on public key certificates and the transport layer security (TLS) protocol for message protection. These are extensible authentication protocol transport layer security (EAP-TLS) and the extensible authentication protocol tunneled transport layer security (EAP-TTLS).

EAP-TLS uses the TLS public key certificate authentication mechanism within EAP to provide mutual authentication of client to server and server to client [20]. With EAP-TLS, both the client and the server must be assigned a digital certificate signed by a certificate authority (CA) that they both trust. The EAP-TLS key system uses dynamic WEP or temporal key integrity protocol (TKIP) keys. WEP is the security standard for all 802.11 standards with a goal to secure wireless local area networks (WLAN) at the same level as wired networks [41]. EAP-TLS uses a TLS handshake as the basis for mutual authentication [20]. EAP-TTLS has been implemented in some Remote Authentication Dial-In User Service (RADIUS) server and supplicant software designed for use in 802.11 WLAN networks. In EAP-TLS, a TLS handshake is used to mutually authenticate a client and server [2005]. EAP-TTLS extends this authentication negotiation by using the secure connection established by the TLS handshake to exchange additional information between client and server. In EAP-TTLS, the TLS handshake may be mutual, or it may be one-way, in which only the server is authenticated to the client. The secure connection established by the handshake may then be used to allow the server to authenticate the client using existing, widely-deployed authentication infrastructures such as RADIUS [2005].

The security token (or policy enforcement point [PEP]) install on the hardware device that the WiGLET carriers to authorize access to a wireless grid. The security token is embedded inside the WiGLET software and provides the first level of security assurance before authentication and authorization. A WiGLET will make a request to another WiGLET's security token and will form a request based on the WiGLET's attributes, the resource in question, the action, and other information pertaining to the request. The security token will then send this request to a policy decision point (PDP), which will look at the request and some policy that applies to the request and come up with an answer about whether data access should be authorized. That answer is returned to the PDP, which can then allow or deny access to the requesting WiGLET. The sending WiGLET has an identification number (ID#), which authorizes the WiGLET as the owner of that particular device. The device then displays a number which uniquely indentifies the WiGLET to its specified service, allowing the WiGLET access to exchange encrypted data.

Before the WiGLET authenticator, which enforces authentication before access to services, allows access to the WiGLET, certain security parameters and standards should be met. For authorization, Security Assertion Markup Language (SAML) is a standard from Organization for the Advancement of Structured Information Standards [OASIS] [38]. SAML defines a framework for exchanging security information in extensible markup language (XML) format, with the main design goal of supporting Single Sign-On (SSO). The authentication facts are represented in XML constructs called "assertions", which are managed by SAML Authorities [38, 8]. Interested WiGLET's could ask a SAML Authority whether an entity possesses a certain assertion, thus eliminating the need of authenticating the entity each and every time. SAML also maps seamlessly to the Simple Object Access Protocol (SOAP) and in many areas complements the Web Services Security [WS-Security] specification [27, 38].

In supporting delegation and authentication, X.509 proxy and identification (ID) certifications are the fundamental public key infrastructure-based technology critical for establishing identities and secure, trusted communications between the components [52]. In many cases, X.509 certificates may be used in place of SAML assertions to provide initiator identity (Lock and Sommerville, 2002). For the actual message format, Simple Object Access Protocol (SOAP), which performs the low-level XML communications for transmitting web service calls across the wire may provide a secure solution [50, 51]. Also, representational state transfer (RESTful) based web services often describe any simple interface which transmits domain-specific data over hypertext transfer protocol (HTTP) without an additional messaging layer such as SOAP [19, 13, 49]. Both SOAP/RESTful potentially provides a means of XML-based messaging between a WiGLET provider and a WiGLET consumer.

Other forms of message protection are needed to protect data from unauthorized (accidental or intention) modification, disclosure or destruction in protected messaging is needed. WS-Security is a standard jointly proposed by IBM, Microsoft, and Verisign and ratified by OASIS [35]. It serves as the foundation to address SOAP-level security issues, with three major propositions: (1) use of security tokens in SOAP headers for user identity and authentication, (2) use of XML-Signature standard for message integrity, and (3) use of XML-Encryption for message confidentiality [35, 36, 53]. There are of course many other security requirements that are not addressed by WS-Security, such as trust relationships and secure transactions. WS-Secure Conversation [WSSC], jointly proposed by IBM, Microsoft, and Verisign, that also specifies the syntax and rules with which security tokens can be exchanged in SOAP [27, 36, 55]. These

protocols and in particular their messaging constructs may potentially become another vehicle to carry the returned SAML authentication assertion for the WiGLET.

It is critical that only identified, authenticated, and authorized WiGLET's have access to other WiGLET's within the wireless grid. Identifying data from an incoming WiGLET to a receiving WiGLET may rely also upon a wireless identification and sensing platform (WISP) using a wireless access point (WAP) type of architecture. A WISP is a class of battery-free wireless sensors that scavenge power from ambient radio-frequency-identification readers to communicate sensed data using a usage model to enable a variety of distributed sensing applications [42]. This type of platform uses Radio Frequency Identification (RFID) technology for sensing and computing power to wireless devices [43]. A WAP point facilitates wireless connection between devices (e.g. sensors) and other wireless modules (e.g. routers, etc.) [10, 16]. The WISP also provides the interface for the WiGLET system resources for timing, power management, memory, central processing unit (CPU) and program and debug services.

Using WISP/WAP construct, identification and authentication data would be provided remotely and authorization information would have to be embedded in a WiGLET security certificate. The WiGLET security certificate requires a service that presents the certificate as a proxy for the WiGLET to other WiGLET's as needed, as well as data that verifies the validity of the security token and manages WiGLET authorizations. Securing these WISP's and WAP's are obtainable through various wireless traffic encryption methods such as IEEE 802.1x/ Remote Authentication Dial In User Service (RADIUS) [14] and wired equivalent privacy (WEP) [45]. Additionally, the WiGLET embedded security kernel (as displayed in Figure 3) plays a vital role in the WiGLET to WiGLET security architecture.

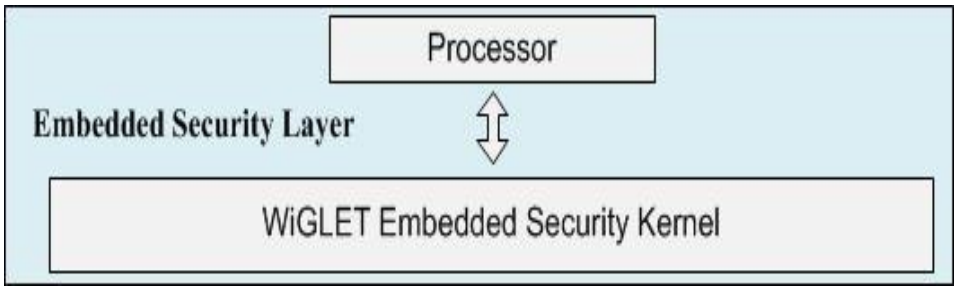

Figure 3. WiGLET Embedded Security Kernel

A kernel can be defined as the code executing in the hardware's privileged mode in an embedded system [23]. The WiGLET embedded security kernel is used to protect the WiGLET from corrupt API interfaces, decrease the size of the core trusted code base, and to put a clear, inviolable interface between the trusted code base and less trusted code within the WiGLET. The WiGLET embedded security kernel is responsible for enforcing the security mechanisms of the entire WiGLET system. The WiGLET embedded security kernel represents a small trusted system, and can be implemented by hardware, software, or a combination of the two with a direct interaction with the processor. It also protects the WiGLET's hardware initialization, device control, application scheduling, and application partitioning.

Even with this limited set of services and security features, the security kernel can run the services of a WiGLET while maintaining a high level of security. By enforcing a process (or application) separation policy, the secure kernel should guarantee that two independently running processes within the WiGLET will not be able to affect each other. This isolation process ensures that malicious code inserted into one isolation segment of the WiGLET kernel cannot access or steal resources, corrupt or read files, or otherwise harm the processes and data in another isolation segment. The WiGLET embedded security kernel must work directly with the other WiGLET's components to directly support the processing of secure data.

Access control mechanisms for the WiGLET must ensure that data is accessed and processed only by other authorized WiGLET's, including precise control over distribution (how many and when). A significant gap in access control that may affect this architecture is the lack of research, policies, standards and mechanisms for this type of environment. For interoperability to exist, the WiGLET will need to be engaged in collaboration with multiple security domains at the same time. In order for such collaboration to be secure, WiGLET's in one domain must be secure and not able to access unauthorized information that resides in other domains. The shared data must be secured and protected during the identification and access control (authorization/authentication) process and internally using an embedded security kernel against threats, attacks, disclosure, corruption and loss of availability (e.g. denial of service).

3. SECURING A WIGLET-TO-CLOUD COMPUTING ARCHITECTURE

Since the emergence of Web Services technologies, there has been a major paradigm shift in distributed computing: from distributed object architectures to service-oriented architecture (SOA) to cloud computing environments [4, 17]. There has been a growing need for increased integration and collaboration across organizational boundaries. A wireless grid WiGLET to cloud integration (Wireless Cloud) is well positioned to become the key technology enabler for such business drivers due to its decentralized, loosely coupled, and highly interoperable architecture. Securing the wireless cloud, however, faces new challenges that may not be fully addressed by existing information technology (IT) or wireless security solutions.

Cloud computing has emerged as a new computing paradigm that arrays masssive numbers of computers in centralized data centers to deliver web-based applications, application platforms, and services via a utility model [4, 48]. The primary difference between cloud computing and previous software as a service models (e.g., outsourcing or data center consolidation) is scalability. The premise is that as the scale of the cloud infrastructure increases, the incremental time and cost of application delivery trends toward zero. The term cloud computing refers to the practice of leveraging third-party computing resources such as network grids and server farms to extend IT capabilities and reduce cost of ownership [15]. Most of the cloud providers today are data centers that have quickly adopted the technology to offer cloud computing services. Figure 4 illustrates the distinction in a cloud computing architecture.

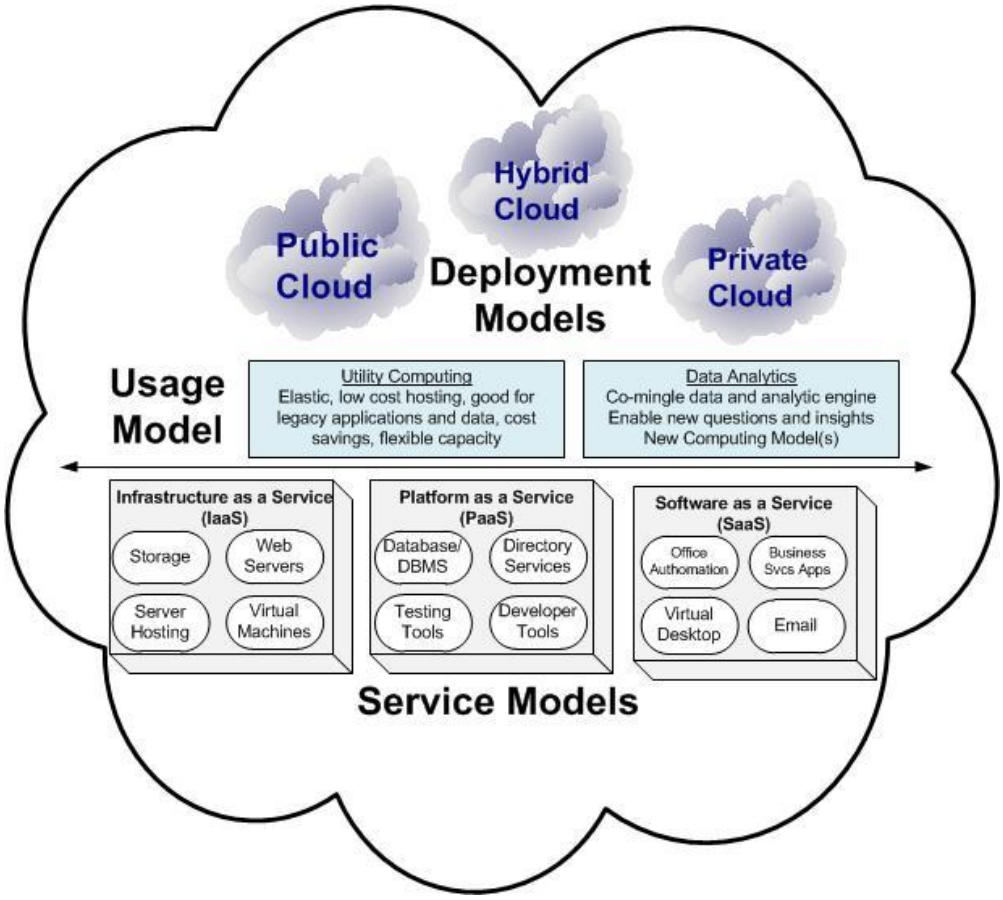

Figure 4. Cloud Computing

Therefore, in a secure wireless cloud environment, organizations will need to shift their focus from perimeter-based security models to a service-level view of security, with emphasis not so much on ownership and control, but on network identities, trust, and authorization of both users and applications. The architectural construct of a secure wireless cloud computing environment is identified in Figure 5. [26] identifies the wireless cloud as a natural extension of the wireless grid, which provides seamless access to the internet, networked devices and computing capabilities. The wireless cloud is a kind of next-generation wireless grid, an emerging technology and the publication on it is limited [26].

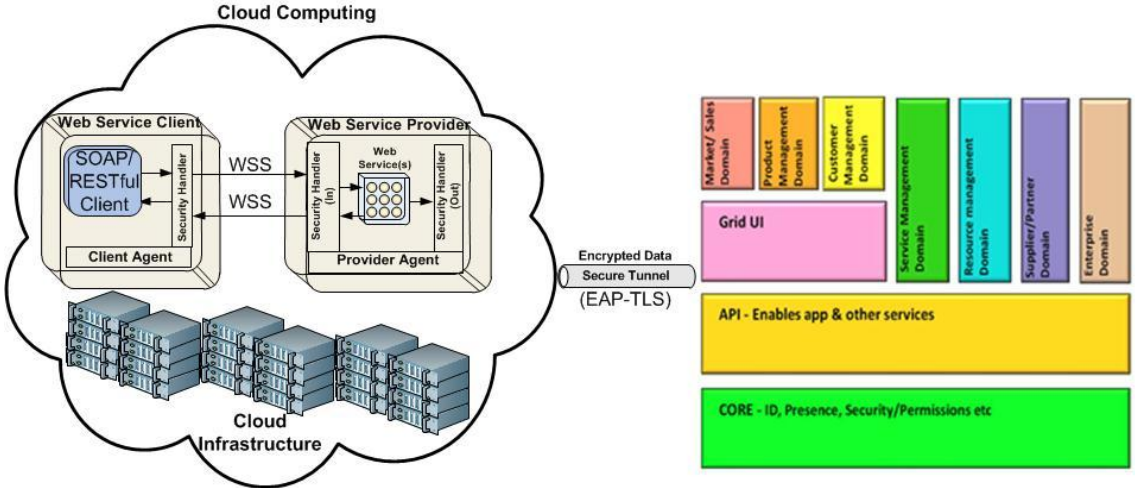

Figure 5. A Future Wireless Cloud Security Architecture

The wireless cloud security architecture is purposely kept technology-agnostic: there is no mention of any specific wire protocols, interfaces, or data models, except that everything will be web services based since the WiGLET interfaces directly with the cloud. Over time, the security standards will evolve and new standards will continue to emerge and the conceptual architecture will not be inherently stable and may not remain the same. For this architecture, data exchanges will be transmitted through the use of encrypted data via a secure tunnel EAP-TLS/TTLS [9, 20]. This secure tunnel provides a secure interaction amongst the WiGLET and the Cloud software. A security token-based authentication (e.g. x.509 certificates) model could be used before invoking any secure web services. A cloud client needs to explicitly authenticate itself to a security token service which is one of the services offered by the WiGLET. After successful authentication, the client will be given a one-time security token, which is then passed in the regular web service request message to invoke a target service. The token essentially establishes a session whose duration is the lifespan of the token. During the session, the client can use the same token to invoke any services (subjecting to the local access control policy of course) and doesn't have to be authenticated again, thereby achieving service-level Single Sign-On (SSO). Compared with the direct authentication model specified in the WS-Security [WSS] standard [40], this approach improves performance by eliminating per-message authentication overhead, which is quite significant when asymmetric key exchange is involved.

In this hypothetical environment, the cloud client application issues a web service request in the form of a regular SOAP/RESTful message to the security handler. The security handler in the cloud client agent intercepts the SOAP/RESTful message. In the absence of a valid security token, the client agent presents WiGLET credentials to the security token service and obtains a security token upon successful authentication. The cloud client agent adds the security token in the SOAP/RESTful message header and sends the request to the target service provider. The cloud provider agent for the target service intercepts the SOAP/RESTful request and extracts the security token. It then checks with the policy decision service (PDS), which fetches needed resource attributes for the target web service from a service registry which acts as the resource attribute authority [54], in the WiGLET on whether this request can be authorized. The latter validates the token and checks the identity against the WiGLET access control policies and responds with a SAML authorization assertion which either permits or denies access. If access is granted, the provider agent forwards on the SOAP/RESTful request to the target web service for processing. The invocation results are returned back to the client in a SOAP/RESTful response.

The WiGLET embedded security kernel provides transparent security enforcement using client/provider agents software components installed in the client/provider application hosting environments in the cloud. At runtime these agents reside in the same process space as the processor, analogous to linked code libraries, with the difference being these agents securing any exposed API interfaces. Using SOAP/RESTful message interception mechanisms (such as Java API for XML-Based RPC [JAX-RPC] [11] or Java API for XML Web Services [JAX-WS] [12] for message handler chains, these agents (among other things) transparently handle message-level security aspects such as authentication, token passing, and policy enforcement – all without the knowledge of the client or service applications. Moreover, third-party web services already developed without being security-aware can easily become so by plugging in to the WIGLET security architecture with little or no modification.

The underlying security functionality such as security token management and policy management are themselves wrapped as web services in securing data to the WiGLET. When a web service invokes another service hosted in a different service provider on the client's behalf, the provider agent can transparently pass on the client's security token, thereby enabling the second provider to authenticate and authorize the client in the same manner. This is especially necessary to support dynamic service composition and orchestration, one of the most appealing features promised by the cloud. To ensure interoperability and API stability, fully standard compliant interfaces between architecture components may need to use SAML for authentication assertion /security token exchange between client agent and WiGLET, and for exchanging authorization decision assertions between provider agent and WiGLET. Also, WS-Security for attaching security tokens to SOAP/RESTful requests to target service providers, extensible access control markup language (XACML) for exchanging access control policies [39] and XML key management specification (XKMS) for key management [22] and PKI integration [2] may also have to be considered.

In a future wireless cloud environment with potentially millions of users and millions or more resources, having a central authorization authority simply won't scale. With this model, multi-level trust domains, policy management on the resources may have to be delegated down to the level where the resource is located and owned, from communities to organizations or even to individuals, effectively limiting the management span at each level. Information exchanged from the Cloud to the WiGLET needs to be secured because the overall security is only as strong as the weakest link in the system. There may be a need for new mechanisms to secure this type of architecture. These mechanisms not only ensure data integrity and confidentiality, but also facilitate mutual authentication and non-repudiation.

4. SECURING A SERVICE-ORIENTED WIGLET ARCHITECTURE (SOWA)

SOA is a way of reorganizing a portfolio of previously siloed software applications and support infrastructure into an interconnected set of services, each accessible through standard interfaces and messaging protocols [18]. Under SOA, a set of network-accessible operations and associated resources are abstracted as a "service", which is described in a standard fashion, made available by publishing the service description to service registries, and found by the service consumers via querying the registry (Figure 6). Four basic standards, SOAP, RESTful, Web Service Definition Language (WSDL) and Universal Discovery, Description, and Integration (UDDI), serve as the foundation of the web services protocol "stack" [7]:

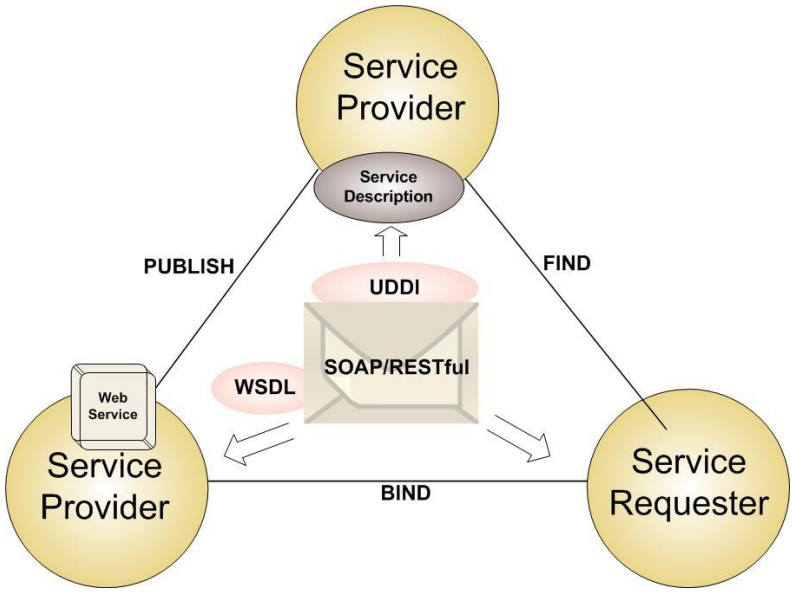

Figure 6. Service-Oriented Architecture (Source: Brooks, 2009)

SOAP performs the low-level XML communications for transmitting web service calls across the wire [18]. It provides a means of XML-based messaging between a service provider and a service consumer. RESTful based web services often describe any simple interface which transmits domain-specific data over HTTP without an additional messaging layer such as SOAP [13]. RESTful style is an abstraction of the architectural elements within a distributed hypermedia system and ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements [19]. WSDL is the

language that defines the functional interfaces for a web service [18]. In other words, a WSDL document, which is itself in XML, represents the official "contract" between the service consumers and providers. These interfaces are described first in abstract message structures, and then bound to a concrete transport protocol and a communication "endpoint". UDDI is the emerging standard for organizing and accessing the service registry [18]. A service registry serves as the "yellow" page of a collection of web services, providing mechanisms for a service provider to publish its capabilities and for a service user to find matching services.

These protocols provide loose coupling of web services standards focus on defining the functional interfaces that represent the minimal understanding between service requestor and provider. Knowledge of the service provider is discovered dynamically from a service registry rather than statically coded in the client program. Simply put, web service calls are essentially XML messages over HTTP (or potentially other standard Internet protocols), which represents the "least common denominator" of network protocol stacks and makes it easier to overcome firewall and infrastructure constraints [7]. When it comes to information sharing among different autonomous networks, web services is likely to be the only viable option. The foundation of a SOWA architecture is one that must be fully developed on a Trust Domain and Trust Model (as displayed in Figure 7), which has been one of importance in distributed computing environments [37]. It may receive more attention (compared with other models such as virtual organizations) in the web services architectures, largely due to the growing acceptance of SAML and RESTful architecture.

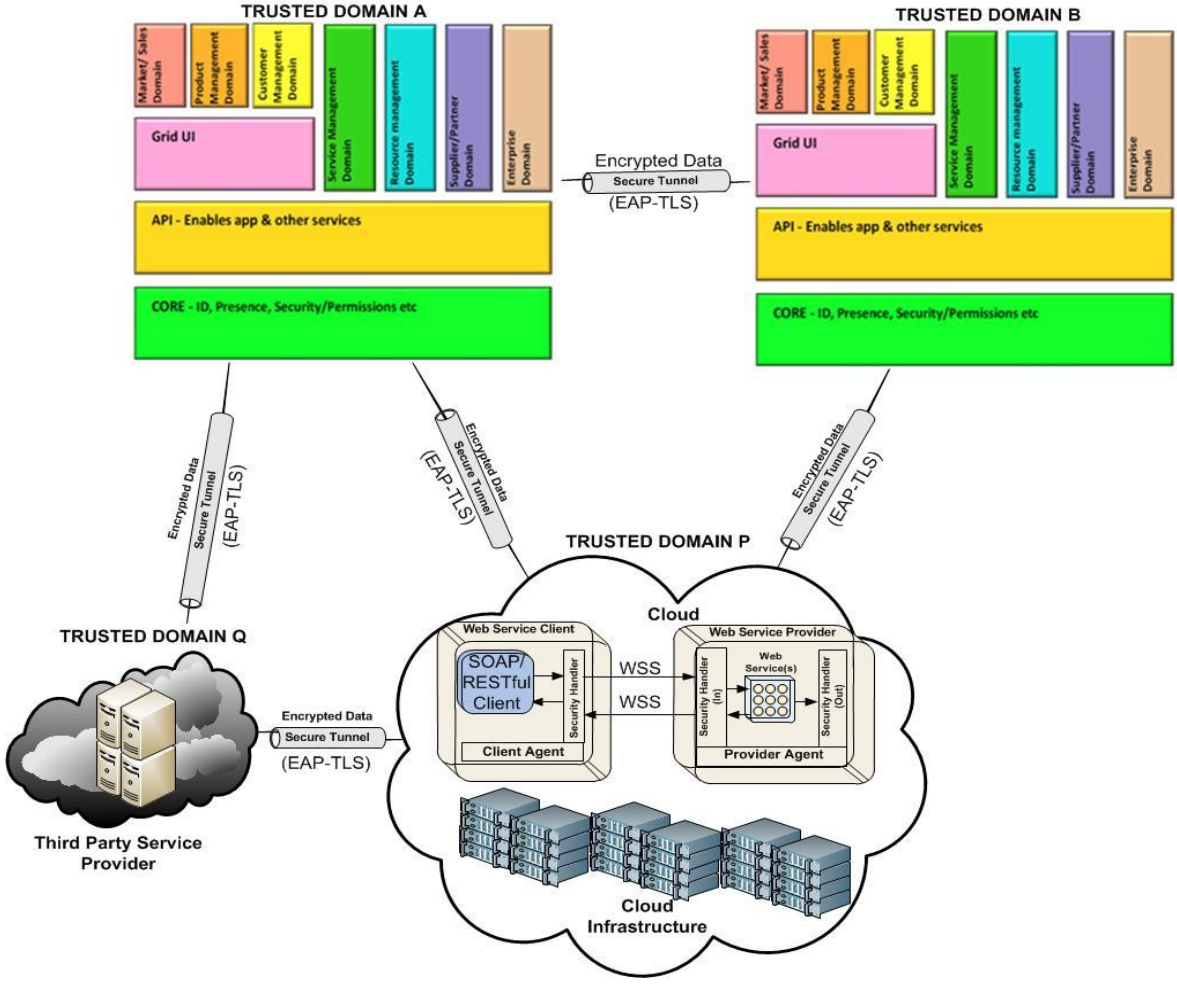

Figure 7. Service-Oriented Architecture (Source: Brooks, 2009)

In the context of SOWA, trust occurs at two levels: First, at the provider level (i.e. the trust domain level). Before information exchange or resource sharing can happen between two trust domains, a certain level of trust agreement must be established. The agreement can take many forms and is often domain specific. For example, in exchanging data amongst WiGLET's, the cloud and third party service providers,

there may be quite a number of legal constraints (e.g. in the form of US laws and government regulations) that dictate what information may nor may not be shared among different parties. Other agreements may be more technical in nature, such as common attribute definitions and required certifications. Secondly, at the identity level: after the trust relationship has been put in place, interoperability can happen between WiGLET's or other system entities between the two trust domains

First, we define what WiGLET trust domain means in technical terms. For example, when WiGLET domain A is said to be trusted with WiGLET domain B, the two domains are able to identify each other, able to locate and communicate with each other and trust the transport on which secure data information is exchanged between each other. WiGLET domain B trust A's process and mechanisms for authenticating its own identities. In other words, B is willing to acknowledge and accept a valid security token issued by A's as the proof of identity. Domain B understands and trusts the attribute assertions issued by A for its identities. This also implies that the two domains share a common set of attribute definitions for identities. Given a valid security token and associated attribute assertions, domain B is willing to grant access to its resources based on its local security policy. The last point is especially worth noting because it stresses that trust domain is a necessary but not sufficient condition for a WiGLET in one domain to access resources in the other; the WiGLET will still be fully subject to the local access control policy in the target WiGLET domain.

Next, we formally define the trust model as the circle of trust consists of multiple trust domains at multiple levels. A WiGLET domain may explicitly enter into a trust relationship with another by mutual agreement as defined in the terms above. The two domains are then called "integrated". A WiGELT domain may also join a "parent" trust domain through the same process. All peer domains within the same parent domain are implicitly considered integrated. Alternatively, the parent WiGLET domain may also instruct the child WiGLET domain to integration with only some but not all of the other child WiGLET domains, based on its policies. Base on its discretion, a parent WiGLET domain can selectively pass its integration relationships down to its child WiGLET domains. For example, if WiGLET domain A is not integrated with another WiGLET domain Q while its parent WiGLET domain P does, and P decides to pass this relationship to A, then A is considered integrated with Q. Again, the integrated by no means grant unrestricted access; it is only a prerequisite.

Within a WiGLET trust domain, the local security policy governs access to local resources for both local identities and remote identities from other integrated domains. Trust is not permanent; it may be terminated by the parent WiGLET domain or either party of the peer WiGLET domains. Trust can be bilateral or unilateral, depending on the policies of the participating domains. In the unilateral case, for instance, WiGLET domain A's users can access resources in WiGLET domain B, but WiGLET domain B's users cannot access resources in A. The trust model should support both scenarios. Inter-domain trust relationships can be established in two ways, either explicitly between two domains or implicitly via joining a parent domain.

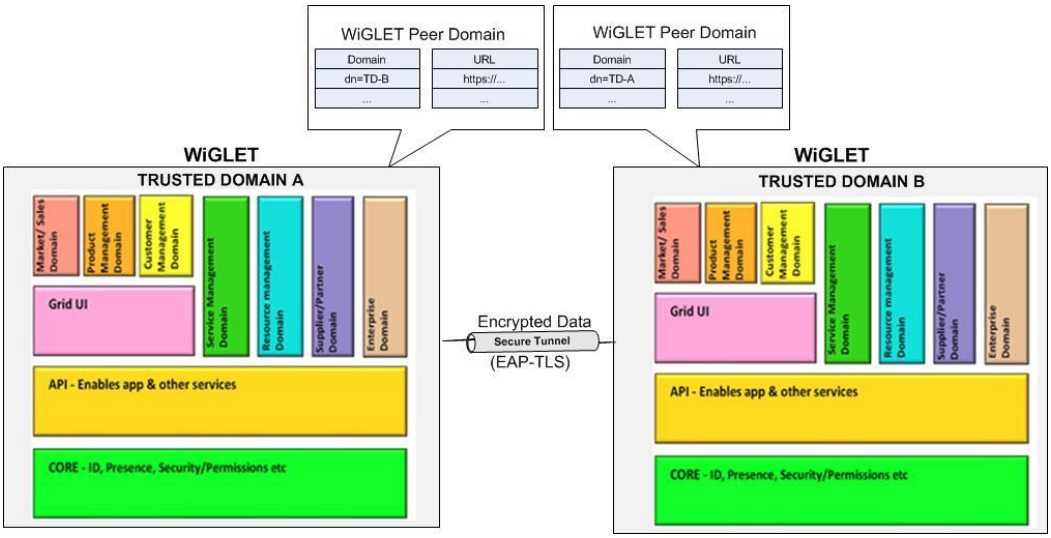

Figure 8. Explicit Integration of WiGLET Peer Domains

A WiGLET trust domain integrates with another peer WiGLET domain when the two domains have established mutual agreement on the trust relationship through a secure connection. Each domain simply needs to obtain the other party's identifier (which in this architecture is the domain name (DN) suffix of the

domain) out-of-band and put in an internal "WiGLET Peer Domains" table. The DN needs to resolve to an actual uniform resource locator (URL) of the other domain's WiGLET; this is achieved through the security token process as is illustrated in Figure 8. To ensure authenticity, both domains may need to acquire and validate the other domain's digital certificate (e.g. the server), facilitated by the underlying security infrastructure.

Furthermore, WiGLET's may need to integrate with additional WiGLET's within the trust domain. Integrating with a 'parent' domain is similar in the sense that the parent and child domains also need to exchange domain DNs and certificates. Additionally, the new child domain may retrieve the list of other child domains from the parent; these domains will implicitly become peer domains of the new child. The other child domains will also learn from the parent that a new child has joined, enabling them to integrate with the new WiGLET.

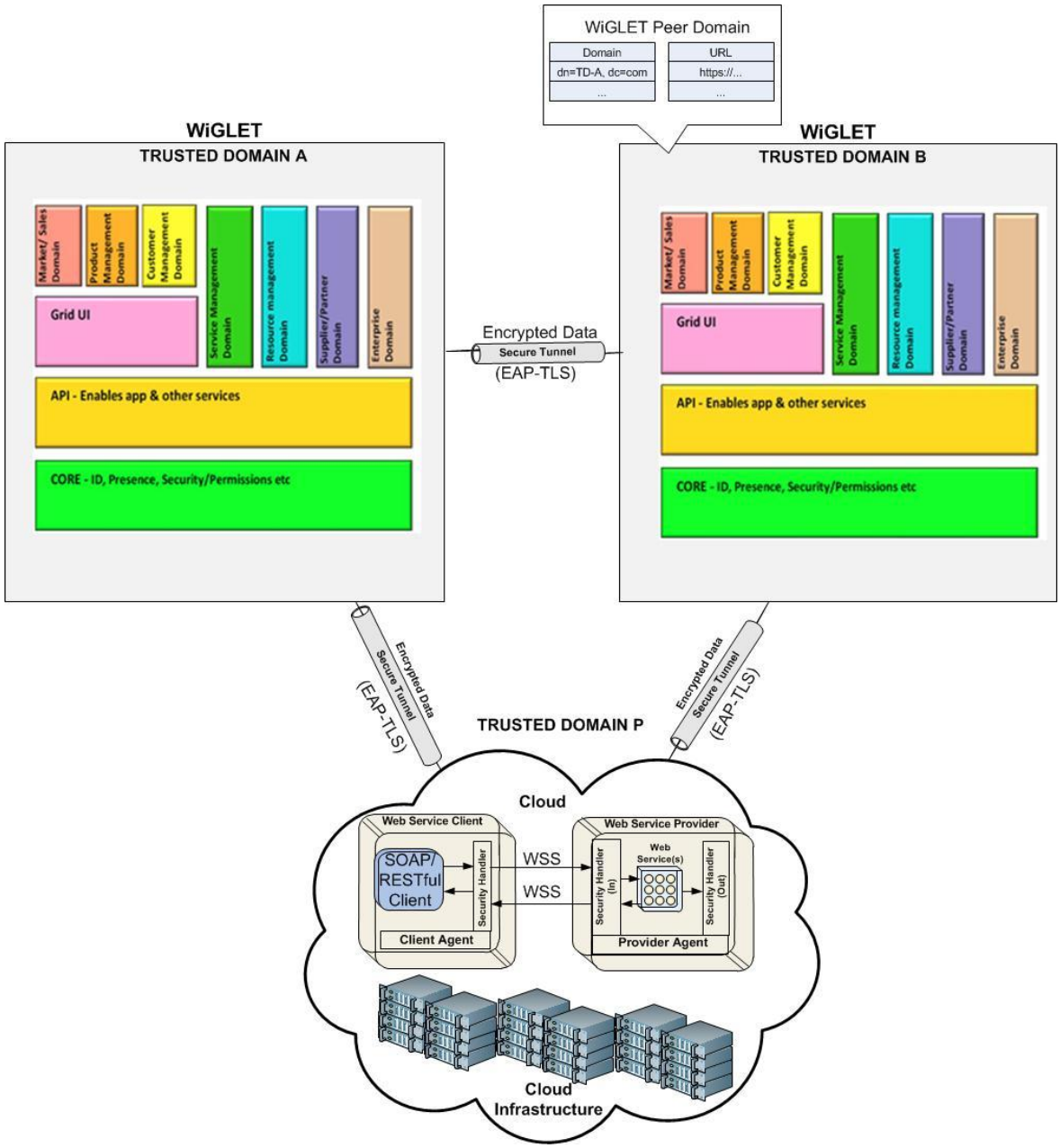

Figure 9. Invoking a Service in a Peer Domain

Invoking a web service through another trust domain (e.g. cloud) assumes that the communication path to the remote domain, most likely SOAP/RESTful/HTTP based, can be established and the target web service has been discovered. The client agent authenticates with the local WiGLET domain's security token

service, then passes the token in the outbound SOAP/RESTful request to the target service. This is the more typical scenario in which trust domain A would like to invoke a service in the cloud while connected to trust domain B and even though trust domain B doesn't have identity in that domain. Presumably the security token in domain B has already learned about this intent beforehand and has assigned the appropriate permissions in the local security policy. The invocation process is shown in Figure 9 above:

When the WiGLET for the target service receives the request from the cloud or another WiGLET, it extracts the security token and queries the local PDS for a SAML authorization decision assertion, making no distinction where this request comes from. When the PDS receives the SAML query, and recognizes by the DN suffix of the identity that the token is not issued by the local domain, it attempts to look up the WiGLET's domains list. The local PDS may also need to ask the remote domain to verify the security token is valid, if the token is not signed. When an entry can be found, it sends a SAML attribute query to the remote domain's attribute management service (AMS), which requests a list of attributes it needs to make access control decisions. Example attributes may include "What is the WiGLET's request?", "How was the WiGLET authenticated?", "What data cloud is the cloud requesting?" and so on. If the query is successful, it feeds those attributes to the evaluation process against the local policy sets and either grants or denies the request.

When processing data from third party service provides, the provider agent protecting a web service can pass the client's identity and security token on to another service, supporting service level "workflow" and composition. This feature is naturally extended the inter-domain architecture, where security token forwarding across domain boundaries is governed by trust as displayed in Figure 10.

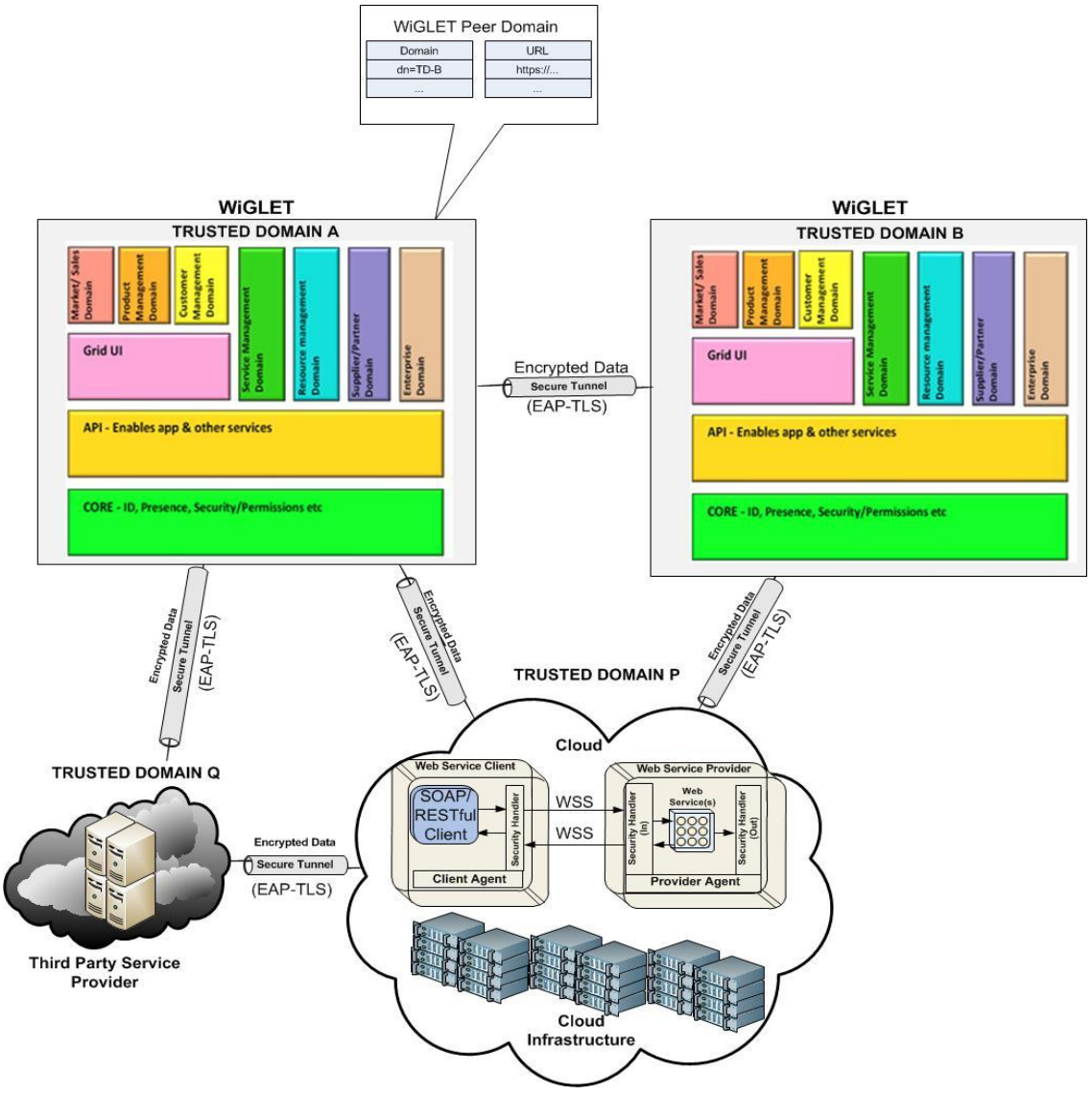

Figure 10. Inter-Domain Services

In this architecture, we assume a WiGLET trust domain A is integrated with WiGLET domain B. WiGLET domain B in requesting information from a third party service provider through a cloud interface however, B is not directly integrated with the third party service provider. Now a client from domain A successfully invokes a service in domain B and the cloud, which in turn needs to access another service located in domain Q from the third party provider. The provider agent for the service transparently passes the security token from domain A to the target service in domain b. When B's PDS examines the foreign token, it should be able to find a secure domain from which attribute assertions can be retrieved, forcing it to accept the security token and provide its access to the target service.

In securing these types of new WiGLET SOWAs, it is almost certain that core set of security services have to be developed in order to define service level interfaces, and domain-specific security services. First, it must includes a security token service (STS) which defines the interfaces to access security tokens based on common authentication mechanisms, such as X.509 certificates [28]. Major operations could potentially include obtaining a new security token (based on the user credentials) and the issuance of a new one-time security token. The token may take different forms, such as a SAML authentication assertion. This may be the most interoperable, but requires good PKI support for assertion signing, renewing and revoking a security token. As to protocols, because the service is only accessed within the trust domain, it can be implemented in a way that is more suitable to the existing tooling and user preferences. Options may include a regular web service interface, a SAML query interface, or a WS-Secure Conversation based interface.

Another key component of securing this WiGLET SOWA is the inclusion of a PDS. The PDS serves as the policy decision point (PDP) of the trust domain. It offers a SAML-compliant interface, accepting authorization queries and returns authorization decision assertions [54]. The heart of the service is basically a policy evaluation engine. The relevant policy rules are then evaluated against the attributes and a permit/deny decision is returned. Although hypothetical, this type of service-oriented architecture could potentially provide new business models for organizations using wireless grids.

5. CONCLUSION

All told, security is based on trust and the design of a solution of trust for the WiGLET is no simple task. A secure WiGLET transaction (i.e. communication) will rely on trust and WiGLET's engaged in a secure transaction must mutually trust each other's identities. This article identified three different types of wireless grid architecture's and specific security considerations for each type around the WiGLET. Due to the inherent nature of the wireless grid connection, the potential unreliability of the end-devices and the power constraints of the mobile device, securing this initial architecture represents the first major challenge in securing the wireless grid environment.

ACKNOWLEDGEMENTS

The development of the Wireless Grid Innovation Testbed (WiGiT) is primarily funded by the support of the National Science Foundation (NSF) Partnership for Innovation (PFI) program grants NSF #0227879 (2002-2006) and continued under NSF # 0917973 (2009-2011).

REFERENCES

[1] R. Abdulrahman, *et al*., "Multi Agent System for Historical Information Retrieval from Online Social Networks," In J. O'Shea, N. Nguyen, K. Crockett, R. Howlett and L. Jain (Eds.), *Agent and Multi-Agent Systems: Technologies and Applications*, pp.54-63, 2011.

[2] C. Adams and S. Lloyd, "Understanding Public-Key Infrastructure, 2nd edition," *Addison Wesley Professional*, 2003.

[3] A. Agarwal, *et al*., "Wireless Grids: Approaches, Architectures and Technical Challenges," MIT Sloan Working Paper No. 4459-04, pp. 1-10, 2004, [online] available: http://ssrn.com/abstract=489782 [Accessed: February 5, 2012].

[4] M. Armbrust, *et al*., "Above the Clouds: A Berkeley View of Cloud Computing," University of California at Berkeley Technical Report No. UCB/EECS-2009-28, pp. 1-25, 2009 [online], available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf [Accessed: February 5, 2012].

[5] S. Bernard and S. Ho, "Enterprise Architecture as Context and Method for Designing and Implementing Information Security and Data Privacy Controls in Government Agencies," In Saha, P. (Ed.), *Advances in Government Enterprise Architecture*, pp. 340-370, 2008.

[6] S. Bradner, "The Internet Engineering Task Force," In DiBona, C., Ockman, S. and Stone, M. (Eds.), *Open Sources: Voices from the Open Source Revolution*, pp. 47-52, 1999.

[7] T. Brooks, "Service-Oriented Enterprise Architecture (SOEA) Conceptual Design Through Data Architecture," Journal of Enterprise Architecture, vol. 5 (4), pp. 16-26, 2009.

[8] S. Cantor, *et al*., "Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS SSTC Document ID sstc-saml-metadata-2.0, pp. 1-43, 2005 [online], available: http://www.oasis-open.org/committees/security/ [Accessed: June 5, 2012].

[9] J. Chen and Y. Wang, "Extensible Authentication Protocol (EAP) and IEEE 802.1x: Tutorial and Empirical Experience," *IEEE Communications Magazine*, vol. 43 (12), pp. 26 – 32, 2005.

[10] Y. Chen, *et al*., "Integrated Wireless Access Point Architecture for Wireless Sensor Networks," in proceedings of the 11th International Conference on Advanced Communication Technology (ICACT 2009), vol.1, 2009, pp.713-718.

[11] R. Chinnici, "Java API for XML-Based RPC (JAX-RPC) Specification 1.1," *Sun Microsystems, Inc*., 2003 [online], available: http://java.sun.com/webservices/jaxrpc/ [Accessed: May 1, 2012].

[12] R. Chinnici, *et al*., "The Java API for XML Web Services (JAX-WS) 2.0 Java Specification Request 224," Sun Microsystems Inc., 2005 [online], available: http://java.net/projects/jax-ws [Accessed: May 1, 2012].

[13] R. Costello, "Building Web Services the REST Way", 2011 [online], available: http://www.xfront.com/REST-Web-Services.html [Accessed: June 5, 2012].

[14] P. Congdon, *et al*., "IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines, RFC 3580, 2003 [online], available: https://tools.ietf.org/html/rfc3580 [Accessed: May 1, 2012].

[15] M. Dikaiakos, *et al*., "Cloud Computing: Distributed Internet Computing for IT and Scientific Research," *IEEE Internet Computing*, vol. 13(5), pp. 10–13, 2009.

[16] X. Dong, *et al*., "Improving the Aggregate Throughput of Access Points in IEEE 802.11 Wireless LANs," in proceedings of the *28th Annual IEEE International Conference on Local Computer Networks (LCN '03)*, 2003, pp. 682- 690.

[17] T. Erl, "Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services," *Prentice-Hall*, 2004.

[18] T. Erl, "Service-Oriented Architecture (SOA): Concepts, Technology, and Design," *Prentice-Hall*, 2005.

[19] R. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," *Ph.D. Dissertation*, University of California, Irvine, 2000.

[20] P. Funk and S. Blake-Wilson, "Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0) RFC 5281, 2008 [online], *Internet Engineering Task Force*, available: http://www.ietf.org/ [Accessed: February 5, 2012].

[21] I. Foster, *et al*., "A Security Architecture for Computational Grids," in proceedings of the *5th ACM Conference on Computer and Communications Security*, 1998, pp. 83-92.

[22] P. Hallam-Baker and S. Mysore, "XML Key Management Specification (XKMS 2.0)," *W3C*, 2005 [online], available: http://www.w3.org/TR/xkms2/ [Accessed: June 15, 2012].

[23] G. Heiser, "Virtualization for Embedded Systems", *Open Kernel Labs, Inc*., 2007, pp. 1-30 [online], available: http://www.scribd.com/doc/27250904/Technology-White-Paper [Accessed: May 31, 2012].

[24] Institute of Electrical and Electronics Engineers, Inc. (IEEE), "Std 802.11 - Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," *IEEE*, 2007, pp. 1-1232 [online], available: http://www.ieee.org/index.html.

[25] H. Kurdi, *et al*., "A Classification of Emerging and Traditional Grid Systems," *IEEE Distributed Systems Online*, vol. 9 (3), 2008, pp. 1-13.

[26] G. Li, *et al*., "A survey on wireless grids and clouds," in proceedings of the *8th International Conference on Grid and Cooperative Computing*, 2009, pp. 261- 267.

[27] H. Liu, *et al*., "Performance of Web Services Security," in proceedings of the *13th Annual Mardi Gras Conference*, 2005, pp. 1-8.

[28] R. Lock and I. Sommerville, "Grid Security and its use of X.509 Certificates", *Lancaster University*, 2002, pp. 1-14 [online], available: http://www.comp.lancs.ac.uk/computing/research/cseg/projects/dirc/papers/gridpaper.pdf [Accessed: June 2, 2011].

[29] J. Marsden, "Determining the Role of Geospatial Technologies for Stigmergic Coordination in Situation Management: Implications of the Wireless Grid," in proceedings of the *1st IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA 2011)*, 2011, pp. 131-135.

[30] L. McKnight, *et al*., "Wireless Grids, Distributed Resource Sharing by Mobile, Nomadic and Fixed Devices," *IEEE Internet Computing*, 8(4), pp. 24–31, 2004.

[31] L. McKnight, *et al*., "Wireless Grids: Assessing a New Technology for a User Perspective," *Designing Ubiquitous Information Environments: Socio-Technical Issues and Challenges*, Springer, 2005, pp. 169–181.

[32] L. McKnight, "The Future of the Internet is not the Internet: Open Communications Policy and the Future Wireless Grid(s)", *NSF/OECD Workshop on Social & Economic Factors Shaping the Future of the Internet*, 2007, pp. 1-4 [online], available: http://www.oecd.org/dataoecd/18/42/38057172.pdf [accessed: June 1, 2011].

[33] L. McKnight, *et al*., "Wireless Grids or Personal Infrastructure: Policy Implications of an Emergent Open Standard," in proceedings of the *TPRC 38th Research Conference on Communication, Information and Internet Policy*, 2010, pp. 1-23.

[34] L. McKnight, *et al*., "Open Specifications for Wireless Grids Technical Requirements, Version 0.1," *WiGiT*, Syracuse University, 2012, pp. 1-22.

[35] A. Nadalin, *et al*. "OASIS Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)," *OASIS Standard 200401*, 2004, pp. 1-56 [online], available: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf [Accessed: June 1, 2011].

[36] M. Naedele, "Standards for XML and Web Services Security," *Computer*, vol. 36(4), pp. 96-98, 2003.

[37] N. Nagaratnam, *et al*. "The Security Architecture for Open Grid Services," *Open Grid Service Architecture Security Working Group (OGSA-SEC-WG)*, 2002, pp. 1-31 [online], available: ftp://ftp.cigs.unimo.it/pub/OGSA-SecArch-v1-07192002.pdf [Accessed: June 1, 2011].

[38] OASIS Open Standards, "Security Assertion Markup Language (SAML) v2.0," *OASIS*, 2005 [online], available: http://www.oasis-open.org/standards#samlv2.0 [Accessed: June 15, 2011].

[39] OASIS Open Standards, "eXtensible Access Control Markup Language v2.0(XACML)," *OASIS*, 2005 [online], available: http://www.oasis-open.org/standards#xacmlv2.0 [Accessed: June 15, 2011].

[40] OASIS Open Standards, "Web Services Security (WSS) 1.1," *OASIS*, 2005 [online] available: http://www.oasis-open.org/standards#samlv2.0 [Accessed: June 15, 2011].

[41] J. Park and D. Dicoi, "WLAN Security: Current and Future," *IEEE Internet Computing*, vol.7 (5), pp. 60- 65, 2003.

[42] M. Philipose, *et al*., "Battery-Free Wireless Identification and Sensing," IEEE Pervasive Computing, vol. 4(1), pp. 37- 45, 2005.

[43] A. Sample and J. Smith, "Experimental Results with Two Wireless Power Transfer Systems," in proceedings of the *IEEE Radio and Wireless Symposium (RWS '09)*, 2009, pp. 16-18.

[44] J. Sherwood, *et al*., "Enterprise Security Architecture: A Business-Driven Approach, *CMP Books*, 2005.

[45] W. Stallings, "Cryptography and Network Security," *Prentice Hall*, 1999.

[46] J. Treglia, *et al*. "Collaboration in a Wireless Grid Innovation Testbed by Virtual Consortium," *Networks for Grid Applications, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, vol. 25 (6), pp. 139-146, 2010.

[47] J. Treglia, *et al*., "Interoperability by 'Edgeware': Wireless Grids for Emergency Response," in proceedings of the *44th Hawaii International Conference on System Sciences (HICSS '11)*, 2011, pp. 1-10.

[48] L. Vaquero, *et al*., "A Break in the Clouds: Towards a Cloud Definition," *SIGCOMM Computer Communications Review*, vol. 39 (1), pp. 50-55.

[49] S. Vinoski, "RESTful Web Services Development Checklist," *IEEE Internet Computing*, vol. 12 (6), pgs. 95-96.

[50] W3C, "Simple Object Access Protocol (SOAP) 1.2," *W3C*, 2007 [online], available: http://www.w3.org/TR/soap/ [Accessed: June 3, 2011].

[51] V. Welch, *et al*., "Security for Grid Services," in proceedings of the *12th International Symposium on High Performance Distributed Computing (HPDC-12)*, 2003, pp. 48-57.

[52] V. Welch, *et al*., "X.509 Proxy Certificates for Dynamic Delegation," in procedings of the *3rd Annual PKI RandD Workshop*, 2004, pp. 1-17.

[53] S. Weerawarana, *et al*., "Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-PEL, WS-Reliable Messaging and More," *Prentice Hall*, 2005.

[54] E. Yuan and J. Tong, "Attribute Based Access Control (ABAC) for Web Services," in proceedings of the *3rd International Conference on Web Services (ICWS 2005)*, 2005, pp. 561-569.

[55] H. Zhuang, "A Secure Framework for Web Service Interaction," in proceedings of the *2011 International Conference on Electric Information and Control Engineering (ICEICE)*, 2011, pp. 95-98.

BIOGRAPHY OF AUTHORS

Tyson Brooks, PMP, works for the U.S. Department of Defense (DoD) and has more than 17 years of professional experience in the design, development and production of a broad range of information systems/technology products and services for the federal government. His expertise includes work in the areas of enterprise architecture, cybersecurity, business process modeling, project management, systems analysis and design, systems engineering and requirements analysis.

Mr. Brooks is pursuing his Doctorate in Information Management from Syracuse University and holds a master's degree in Information and Telecommunications Systems from Johns Hopkins University; a master's degree in Business Administration from Thomas More College; and a bachelor's degree in Business Administration/Management from Kentucky State University. Mr. Brooks also received his Certificate of Advanced Study (CAS) in Information Security Management (ISM) from Syracuse University; a Certification in Enterprise Architecture (CEA) from Carnegie Mellon University's Institute for Software Research International (ISRI); and a Project Management Professional (PMP) certification from the Project Management Institute (PMI).

Lee W.McKnight is Kauffman Professor of Entrepreneurship and Innovation and an Associate Professor in the iSchool (The School of Information Studies), Syracuse University; Founder and Member of the Board of Directors of Wireless Grids Corporation; as well as a Founding Member of the Board of Directors of Summerhill Biomass Systems. Lee is Principal Investigator of the National Science Foundation Partnerships for Innovation Wireless Grids Innovation Testbed (WiGiT) project, is recipient of the 2011 TACNY Award for Technology Project of Year, and is the inventor of edgeware, a new class of software for creating ad hoc overlay network applications, known as wiglets. Lee's research focuses on virtual markets and wireless grids, the global information economy, national and international technology policy, and Internet governance and policy. He was an Associate Professor and Director of the Edward R. Murrow Center at the Fletcher School of Law and Diplomacy, Tufts University; Principal Research Associate and Lecturer at MIT, and Founder of the Internet Telephony Consortium, also at MIT.

McKnight received a Ph.D. in 1989 from MIT; an M.A. from the School of Advanced International Studies, Johns Hopkins University in 1981; and a B.A. magna cum laude from Tufts University in 1978.