Establishing Trusted Identities in Disconnected Edge Environments

Sebastián Echeverría, Dan Klinedinst, Keegan Williams, Grace A. Lewis
Carnegie Mellon Software Engineering Institute
Pittsburgh, PA USA
{secheverria, djklinedinst, kmwilliams, glewis}@sei.cmu.edu

***Abstract*—When establishing communication between two nodes, identification, authentication, and authorization provide the information and assurances necessary for the nodes to trust each other. A common solution for establishing trust between two nodes is to create and share credentials in advance, and then use a third-party, online trusted authority to validate the credentials of the nodes. However, the characteristics of tactical environments — such as those in which first responders, search and rescue teams, and military personnel operate — do not consistently provide access to that third-party authority or certificate repository because they are DIL environments (disconnected, intermittent, limited). The goal of this paper is to present a solution for establishing trusted identities in disconnected environments based on secure key generation and exchange in the field. For the implementation and evaluation of the solution we use our open source implementation of a tactical cloudlets system that is targeted at supporting disconnected operations.**

I. Introduction

First responders, search and rescue teams, military personnel, and others operating in crisis environments increasingly make use of handheld devices to help with tasks such as face recognition, language translation, decision support, and mission planning and execution. Due to the computation-intensive — and often data-intensive — nature of these tasks, mobile systems can make use of cyber-foraging to leverage proximate resource-rich surrogates to augment the capabilities of resource-limited mobile devices through computation offload and data staging [1]. In these tactical environments, often characterized as DIL environments (disconnected, intermittent, limited), surrogates are pre-provisioned with all the computation and data needed for a mission so that they do not have to rely on reach back to the enterprise.

To support mobile computing at the edge we developed **tactical cloudlets**. These are forward-deployed, discoverable, virtual-machine-based servers that can be hosted on vehicles or other platforms to provide infrastructure to offload computation, provide forward data-staging for a mission, perform data filtering to remove unnecessary data from streams intended for users, and serve as collection points for data heading for enterprise repositories. The forward-deployed, single-hop proximity to mobile devices promotes energy efficiency as well as lower latency (faster response times) [25]. Tactical cloudlets are intended in many cases to work completely disconnected from the enterprise [26]. Prior to a deployment, cloudlets are pre-provisioned with the capabilities and data that will be needed for a particular mission. Once in the field, mobile devices discover proximate cloudlets, query for services, and then start services on demand. The initial version of our tactical cloudlets implementation had no security or trust embedded into the system other than at the network level, meaning that a user could connect to a cloudlet if it had network accessibility to it.

When establishing communication between two nodes — such as between a mobile device and a tactical cloudlet in the field — identification, authentication, and authorization provide the information and assurances necessary for the nodes to trust each other (i.e., mutual trust). A common solution for establishing trust between two nodes is to create and share credentials in advance, and then use a third-party, online trusted authority to validate the credentials of the nodes. However, the characteristics of tactical environments do not consistently provide access to that third-party authority or certificate repository.

In the context of tactical cloudlets we need to develop a trusted identity solution that meets four major requirements:

1) The solution cannot require network connectivity to a third party such as the Internet, an enterprise or wide-area network (WAN), or a Certificate Authority (CA). In a DIL environment, these connections may be unreliable, non-existent, or even undesirable. Therefore the solution cannot use technologies such as a central authentication service or Internet-based identity management.
2) The solution cannot place any specific security requirements on hardware, such as a Trusted Platform Module (TPM) processor (Section II-A). Multi-organization groups often come together to support missions and need to be able to join the group without specially-provisioned hardware.
3) The solution cannot require pre-provisioning of credentials on the mobile devices. Although cloudlets themselves can be pre-provisioned for a specific mission or deployment, end devices must be able to join during the mission, in a contested environment.
4) The solution must address the threats of a tactical environment (Section III-A). The main difference with other threat models is that there is likely to be an adversary in physical proximity to the system. Therefore, the solution must consider loss or theft of the mobile devices, proximity to short-range radios, and the ability of an adversary to control or contest any network connection

978-1-5090-3322-5/16 $31.00 © 2016 IEEE
DOI 10.1109/SEC.2016.27

to the Internet or enterprise network.

The goal of this paper is to present a solution for establishing trusted identities in disconnected environments based on secure key generation and exchange in the field that meets the above requirements. For the implementation and evaluation of the solution we use our open source tactical cloudlets system that is targeted at supporting disconnected operations.

Section II presents related work in the area of trusted identities. Section III describes our trusted identity solution, including the development process and rationale. Section IV presents the implementation of the solution in the tactical cloudlets system. Section V presents the evaluation of the solution. Finally, Section VI summarizes and concludes the paper.

II. Related Work

Establishing trust in disconnected environments requires decentralized security solutions and infrastructure that are challenging to implement due to basic security concerns such as how to exchange keys, how to manage keys, how to integrate with existing applications, and how to configure security policies [2]. This section presents potential solutions for decentralized security with discussion related to their applicability to disconnected environments, in particular those involving mobile clients interacting with servers deployed in the field.

A. Hardware-Based Solutions

Hardware-based solutions require the presence of an onboard secure hardware component that stores security credentials. Because credentials are embedded in hardware, these solutions are typically harder to break than software-based solutions. In addition to cost, a problem with hardware-based credentials is that these need to be delivered to the field should they need to be changed, which could be problematic in disconnected environments. Examples of hardware-based solutions include Trusted Platform Module (TPM) [3], ARM TrustZone [4], and SmartCards [5].

B. Software-Based Solutions

Software-based solutions rely on credentials stored in software components of a system, such as certificate stores, configuration files, and databases. Examples of software-based solutions that could be applicable to disconnected environments include:

- Identity-Based Cryptography (IBC): In IBC, a public key is derived from an arbitrary data string, and the corresponding private key is created by binding this string with a system master secret owned by a trusted authority called a public key generator (PKG) or key generation center (KGC) [6]. IBC is ideal for disconnected environments because (1) it does not require users to pre-compute key pairs and obtain certificates for their public keys and (2) nodes contact KGCs only once to obtain their private key. The main disadvantage of IBC is the property of key escrow because the KGC knows the user's private key.
- Secure Key Agreement without a Trusted Third Party (TTP): Work in this area leverages out-of-band channels for securely pairing two devices (computational units) without previous exchange of a secret key, or needing to have each other's public key. The challenge is to do so without relying on a trusted third-party to create and distribute this secret key. Examples of solutions in this space include SafeSlinger which leverages physical proximity and visual confirmation to provide secure communication between members of a group [7]; MVSec which leverages various out-of-band channels readily available in commercial vehicles and mobile devices, such as humans, light, sound, and vibration, to secure communication between an individual's smartphone and his/her vehicle [8]; and SPATE which relies on visual channels and physical interactions to establish trust in small groups [2]. The advantage of these solutions is the ability to generate credentials in the field. The disadvantages are related to the lack of centralized control, which makes it difficult to add a node to a group of trusted nodes once credentials have been exchanged and validated, or to remove a node.
- Distributed Trust Models: These solutions are common in ad-hoc networks, in which there are mechanisms that allow a node to evaluate the trustworthiness of other nodes based on, for example, trust chains, trust tables or reputation scores [9][10]. The advantage of these solutions is that they leverage peers for trust verification. The disadvantage is that they rely on nodes that are connected or aware of other nodes, which is not necessarily the case of mobile devices that leverage field-deployed servers in disconnected environments.

C. Hybrid Solutions

Hybrid solutions have a software and a hardware component. As an example, layered trust models have a software layer built on top of a hardware layer, such as using smart cards as secure containers for digital certificates and a software PKI-based trust model built on top [11]. Hybrid solutions inherit both advantages and disadvantages of software-based and hardware-based solutions.

D. Human-Centric Solutions

Human-centric solutions, as the name indicates, involve humans for establishing trust. Examples of human-centric solutions that could be applicable to disconnected environments include:

- Social Networks: In these solutions the trust relationships between users in their real social networks are automatically translated to trust relationships between their devices [12]. The advantage is that there is no need to exchange keys or certificates. However, the disadvantage is that the relationships in the social world have to be trusted.

- Biometrics and Behaviometrics: These solutions use biometrics (e.g., fingerprints, face recognition, voice recognition, retinal scans) and/or behaviometrics (e.g., keystroke analysis, handwriting, gestures) as identities [13][14][15][16]. The advantage of these solutions is identity strength. The main disadvantage is that they need to compare against a saved or network-accessible template that may not be available in disconnected environments.

III. Development of the Trusted Identity Solution

This section presents the approach by which our trusted identity solution components were selected and our trusted identity solution was developed. We first identified a threat model for disconnected environments, then validated the solutions from Section II against the threat model, and finally developed an identity solution based on components that best addressed the threats in the threat model, and the requirements from Section I.

A. Threat Model for Disconnected Environments

The context for the threat model is a client/server type of system in which the client is a mobile device and the server is providing capabilities to mobile devices. The server is fully disconnected from the network and provides capabilities to proximate mobile devices connected via WiFi. The threat model was developed using Microsoft's SDL Threat Modeling Tool [17] which generated 60 potential threats. These threats were examined by a threat modeling expert on our team, evaluated for their applicability to trust in disconnected environments, and consolidated into the 14 relevant threats shown in Table I. Assigning priorities to those threats based on impact and probability of occurrence in an operational setting is also part of the threat model.

B. Evaluation of Existing Solutions Against Threat Model

Table II maps threats against identity solutions. The number of plus signs (⊕) in a cell represents the potential of the identity solution (or elements of the identity solution) to mitigate the threat.

⊕⊕⊕⊕	Solution fully addresses the threat
⊕⊕⊕	Solution mostly addresses the threat but needs to be combined with other solution(s) to fully address the threat
⊕⊕	Solution has some elements that could be used to address the threat but need to be needs to be combined with other solution(s) to fully address the threat
⊕	Solution has minimal support to address the threat
None	Solution has no support to address the threat

Based on an analysis of Table II by threat we can note that (1) most solutions will address threats that involve mobile device and server identity/authentication, (2) threats related to code identity will require mechanisms for code signing and validation, and (3) threats in which mobile devices and servers are compromised will require mechanisms to limit connection time and identity expiration.

The analysis of Table II by solution, in conjunction with an analysis of the advantages and disadvantages of each, and the solution requirements presented in Section I, shows that:

- Traditional PKI addresses most of the threats and there is a lot of out-of-the box support and easy integration with HTTP and TLS. However there are several major drawbacks to Traditional PKI systems. One is that it requires regular, frequent network connectivity to some form of central hub for one or more of the following functions: (1) authentication against a central server (e.g., Kerberos), (2) the ability to receive Certificate Revocation Lists (CRLs), and (3) the ability to revoke an entire CA if it is compromised. This violates Requirement 1, which is to not require network connectivity to a third party. Furthermore, existing PKI solutions use very long public keys by the standards of DIL environments — even the MD5 signature of a public key certificate is 32 hexadecimal characters. Because credentials cannot be pre-provisioned per Requirement 3, keys need to be bootstrapped on to devices, sometimes over very low bandwidth channels (e.g., voice, visual). This is further limited by Requirement 4, the ability to remain secure in an adversarial, tactical environment.
- Hardware-based solutions are the strongest, but the reliance on special servers and mobile devices with hardware trust components make it a challenge for disconnected environments. Our use cases envision teams from multiple services, countries, and agencies being able to form ad hoc networks with their own equipment (Requirement 2).
- IBC is a decentralized solution that maps well to disconnected environments. There are several algorithms and implementations that could enable the server to act as the PKG.
- Secure Key Agreement without a Trusted Third Party is also a decentralized solution that maps well to disconnected environments, specifically exploiting initial physical proximity between servers and mobile devices for secure key exchange, and also as part of two-factor initial authentication (bootstrapping).
- Distributed trust models do not address many of the threats by themselves, but elements of this solution could be employed as part of a two-factor authentication solution.
- Layered trust models combine the advantages and disadvantages of hardware- and software-based solutions. This type of solution could work well in a homogeneous hardware environment.
- Social network solutions do not address many of the threats. Even though elements of a social network solution could be employed as part of a two-factor authen-

TABLE I
THREAT MODEL FOR DISCONNECTED ENVIRONMENTS

#	Name	Description	Priority
1	Impersonating a device	Unauthorized device attempts to gain access to the server environment	H
2	Finding an active client	Authorized phone is lost with an established connection	H
3	Finding a device	Authorized phone is lost without a connection currently operating	H
4	Altered software	Software on an approved device is changed due to downloaded malicious code, tampering, unintended changes, or some other means	M
5	Daisy chaining	External device is able to connect to the authorized device and exploit its approved access	M
6	Lost credentials	Authorization information is obtained by a malicious person who then tries to spoof the device	H
7	Sniffing wireless	WiFi signal is monitored by an external party providing visibility of traffic stream	H
8	Site intrusion	Physical access to server is obtained providing hands-on access to the equipment	H
9	On the net	Network access to the service infrastructure is obtained	H
10	On the box	Access to server OS is obtained	H
11	Super-user compromise	System admin access is compromised and software and data can be stolen or changed impacting services and integrity	H
12	Application compromise	Application controls are compromised	L
13	Seeing everything	Data management controls are compromised	L
14	Server impostor	Impersonating a trusted server environment and enticing devices to connect	H

TABLE II
EVALUATION OF SOLUTIONS AGAINST THREAT MODEL FOR DISCONNECTED ENVIRONMENTS

Threat	Traditional PKI	Hardware-Based	Software-Based			Hybrid	Human-Centric	
			IBC	Key Agreement w/o TTP	Distributed	Layered	Social Networks	Bio- and Behaviometrics
1. Impersonating a device	⊕ ⊕ ⊕⊕	⊕ ⊕ ⊕⊕	⊕ ⊕ ⊕⊕	⊕ ⊕ ⊕⊕	⊕⊕	⊕ ⊕ ⊕⊕	⊕⊕	⊕⊕
2. Finding an active client	⊕		⊕		⊕	⊕⊕	⊕	⊕⊕
3. Finding a device	⊕ ⊕ ⊕	⊕ ⊕ ⊕	⊕ ⊕ ⊕	⊕ ⊕ ⊕	⊕⊕	⊕ ⊕ ⊕	⊕⊕	⊕ ⊕ ⊕⊕
4. Altered software	⊕	⊕ ⊕ ⊕	⊕	⊕	⊕	⊕⊕		⊕
5. Daisy chaining	⊕	⊕	⊕	⊕		⊕		⊕
6. Lost credentials	⊕	⊕ ⊕ ⊕⊕	⊕	⊕	⊕	⊕ ⊕ ⊕	⊕	⊕ ⊕ ⊕
7. Sniffing wireless	⊕ ⊕ ⊕	⊕ ⊕ ⊕	⊕ ⊕ ⊕	⊕ ⊕ ⊕	⊕⊕	⊕ ⊕ ⊕	⊕⊕	⊕ ⊕ ⊕
8. Site intrusion	⊕	⊕⊕	⊕	⊕	⊕	⊕⊕	⊕	⊕
9. On the net	⊕⊕	⊕⊕	⊕⊕	⊕⊕	⊕⊕	⊕ ⊕ ⊕	⊕⊕	⊕ ⊕ ⊕
10. On the box	⊕	⊕	⊕	⊕	⊕	⊕		⊕
11. Super-user compromise	⊕ ⊕ ⊕⊕	⊕ ⊕ ⊕⊕	⊕ ⊕ ⊕⊕	⊕ ⊕ ⊕⊕	⊕ ⊕ ⊕⊕	⊕ ⊕ ⊕⊕	⊕	⊕ ⊕ ⊕⊕
12. Application compromise	⊕ ⊕ ⊕⊕	⊕ ⊕ ⊕⊕	⊕ ⊕ ⊕⊕	⊕ ⊕ ⊕⊕	⊕	⊕ ⊕ ⊕⊕	⊕	⊕ ⊕ ⊕⊕
13. Seeing everything	⊕ ⊕ ⊕⊕	⊕ ⊕ ⊕⊕	⊕ ⊕ ⊕⊕	⊕ ⊕ ⊕⊕	⊕	⊕ ⊕ ⊕⊕	⊕	⊕ ⊕ ⊕⊕
14. Server impostor	⊕ ⊕ ⊕⊕	⊕ ⊕ ⊕⊕	⊕ ⊕ ⊕⊕	⊕ ⊕ ⊕⊕	⊕⊕	⊕ ⊕ ⊕⊕	⊕⊕	⊕⊕
SCORE	**34**	**39**	**34**	**33**	**21**	**40**	**16**	**35**

tication solution, there is no equivalent of software-level social relationships in disconnected environments, other than for example being part of the same team or squad, which even then would require that relationship to be represented at the software-level.

- Biometrics and behaviometrics provide very strong identities but are not a good match for disconnected environments because access to templates for comparison would need to be available, violating Requirement 1.

Overall, from a quantitative perspective (last row of Table II), 5 of the 7 alternative solutions to Traditional PKI provide similar threat mitigation potential (in the 33-40 range). Hardware-based and hybrid solutions provide greater threat mitigation because they provide hardware elements that can be used for device/server identities as well as user identities. However, from a qualitative perspective, IBC and Secure Key Agreement without a Third Party provide very similar threat mitigation potential to Traditional PKI but also address some of its limitations. A combination of IBC and Secure Key

Agreement would therefore address most of the threats but would need to be combined with code signing capabilities, and external server-, OS-, network-, and application-level controls to address all of the threats.

C. Description of Developed Solution

The structure of the tactical cloudlet implementation that includes our trusted identity solution is presented in Figure 1 and is based on the design presented in [26]. In this implementation, a tactical cloudlet is composed of a **Cloudlet Host** computer that acts as a server, which is connected via Ethernet to a **WiFi Access Point** that provides wireless access to clients to the cloudlet's services. These two components define a cloudlet unit. A **Cloudlet Client** is a mobile device with WiFi capabilities. Both the Cloudlet Host and the Cloudlet Client also have Bluetooth and/or USB capabilities (which will be used for pairing).

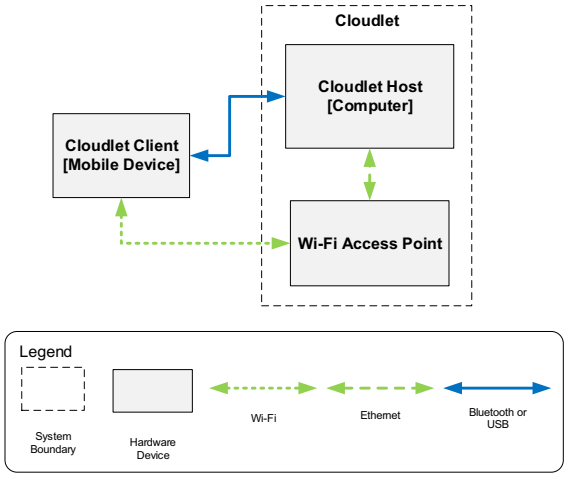

Fig. 1. Tactical Cloudlet Physical Components

The goal is to establish trust between a device (Cloudlet Client) and the cloudlet. There are two human users involved in this process: the Device User, who uses the Cloudlet Client, and the Cloudlet Admin, who manages the cloudlet as needed. Initially, the device does not know or trust the cloudlet, and is not able to connect to it in any way. Likewise, the cloudlet does not know or trust the device. When in close proximity, however, the Cloudlet Admin can recognize and trust the Device User. The end result of the process we will describe is that the device and cloudlet trust each other; more specifically, that an authorized device is allowed to connect and be authenticated to the cloudlet's WiFi Access Point, and that it is also allowed to securely request services from the Cloudlet Host through the network.

The developed solution follows the Identity Based Cryptography (IBC) methodology described in [18] while using Secure Key Agreement [19] to facilitate logistical requirements of disconnected environments. Our implementation depends on the Stanford Identity Based Encryption (IBE) Library, which uses the Boneh Franklin scheme as a Key Encapsulation Mechanism (KEM) and off-the-shelf (OpenSSL) ciphers and HMACs for the actual encryption [20]. Identity-Based Short Signatures [21] are used for the WiFi Authentication process.

The selected Secure Key Agreement without a Trusted Third Party ceremony takes advantage of deployments in disconnected environments; specifically, the presupposition of physical proximity. The proposed solution requires a participant's physical proximity to the PKG (i.e., server) for the initial identification and authentication. Because the Cloudlet Admin trusts the Device User, it can also trust a device carried by the Device User. Physical proximity can then be used to establish trust between the device and the cloudlet with the help of the users.

We define the following cryptographic elements for the processes:

1) **Server** (Server Private Key, Server Public Key): The Server is a program running on the Cloudlet Host that provides cloudlet services to devices. The Server Public Key, called IBE params in IBE, is generated by the IBE library with `IBE_setup()`. The Server Private Key, called Master Key in IBE, is a non-RSA key also generated with `IBE_setup()`.
2) **RADIUS Server** (RADIUS Server Certificate, RADIUS Server Private Key): We use RADIUS (Remote Authentication Dial-In User Service) as the client/server networking protocol for WiFi authentication because it is supported natively by most WPA Enterprise access points. [22]. The RADIUS server runs on the Cloudlet Host and it is configured to authenticate users trying to connect to the WiFi Access Point. Both the self-signed X.509 RADIUS Server Certificate and the corresponding RSA RADIUS Server Private Key are generated using OpenSSL.
3) **Device** (Device Private Key, Device Public Key, Device BLS Certificate): The Device Public Key is the Device ID and acts as the unique identifier required by IBE. The Device Private Key is a non-RSA key generated by the IBE Library with `IBE_extract()` using the Device Public Key and the Server's Public and Private Keys. The BLS Certificate is the Device Public Key signed by the Server's Public and Private keys using `IBE_certify()` from the IBE Library.

Our solution to establish trust consists of four subprocesses: Bootstrapping, Pairing, WiFi Authentication, and API Requests. The first two processes perform the actual trust establishment; the other two are used to authenticate a paired device requesting access at the WiFi and network level respectively. In addition, there are two ways to revoke device credentials: Automatic and Manual.

1) Bootstrapping Process: The Bootstrapping process establishes the encryption and identity parameters on the server. Every server deployment starts from a clean state, which is why the first step of the process is to delete any existing server credentials. To function as an IBC PKG, the server must first generate its own Server Private Key and Server

Public Key. This is the Setup phase as described in [18]. These keys are analogous to a private/public keypair in traditional public key cryptography. It also generates the RADIUS Server RSA private / public keypair and X.509 certificate in order to perform WiFi Authentication using WPA2-Enterprise [23] against the RADIUS Server running on the server. The generated RADIUS Server Certificate and Private Key are used by the RADIUS Server in order to use the EAP-TTLS protocol [24]. Finally, to limit connection time and enforce identity expiration as required by the threat model, a deployment duration is set as the last step of the bootstrapping process.

2) Pairing Process: Pairing is the process of identifying a client device to the server, authorizing it to access that server, and transferring the required credentials. A Mobile Device User would begin by presenting their device to the Cloudlet Admin. The Cloudlet Admin would decide if the user is authorized to use the server based on physical credentials and characteristics, e.g., photo ID, uniforms and insignia, personal knowledge, or delegation of trust. If approved, the Cloudlet Admin would login to the Server and the Mobile User would connect the candidate mobile device to the server via USB or Bluetooth. The server asks the device to send its Device ID (which functions as the Device Public Key) and uses it to generate the Device Private Key. This is the Extract phase in [18]. The advantage is that there is no need for the server to have a pre-configured list of good devices or public keys. If the Cloudlet Admin trusts the possessor of the device, he/she can grant that device access. We use the Android Device ID as the Device ID. The Android Device ID is a unique 64-bit number (represented as a hex string) obtained with the command

```
Settings.Secure.getString(context.
  getContentResolver(),
  Settings.Secure.ANDROID_ID).
```

This ID is randomly generated when the user first sets up the device and should remain unique for the lifetime of the device.

This process addresses one shortcoming of basic IBC, which is the need for key escrow. Under the original Boneh Franklin scheme, the PKG knows and can recompute the private key for any given public key. The IBE library has a method to split that ability across multiple servers, so that no one server has that information. However, in a disconnected environment, the server can be considered trustworthy for its stated class of information. This trust is verified by the user of a device before pairing and is validated in the same way that the user was validated — photo ID or other physical credentials. Different servers could offer different applications or data, and each could independently choose which users and devices to trust.

Once the private key has been extracted, the server also creates a Device BLS Certificate for that device that is used in the following phases. The Device Public Key (Device ID) and Device BLS Certificate are stored as a key/pair value in a list of paired devices. The server registers the Device Public Key and Device BLS Certificate with the RADIUS Server. The server also sends the Mobile Device four pieces of information: Device Private Key, Device BLS Certificate, Server Public Key, and RADIUS Server Certificate. The Device Public Key and the Device BLS Certificate are set as the user credentials in the WPA2-Enterprise WiFi profile created on the device. Finally, the Device Private Key is deleted on the server in order to address elements of the threat model.

3) WiFi Authentication Process: The servers provide all of their services over WiFi using standard 802.1X authentication. Clients connect to a server's WiFi Access Point (AP) and request access to the network. The AP provides the RADIUS Server Certificate and the client verifies its own copy of the RADIUS Server Certificate, which it obtained via pairing. If the certificate is valid, the client sends its Device Public Key (Device ID) and Device BLS Certificate as the PAP username:password tuple over a TLS-encrypted tunnel (EAP-TTLS). The RADIUS Server checks the Device BLS Certificate against those in the RADIUS database. For added security, the server can recalculate the certificate using the Device ID to ensure, cryptographically, that the device has previously paired. At this point, WPA2 Enterprise authentication has succeeded and the client is authorized to use the server's WiFi network.

4) API Request Process: Services on a cloudlet are provided by a simple HTTP Request/Response protocol. To secure the requests, the client can decide to use any of the cryptographic elements generated in the previous phases, or can generate and share a new secret key. If a new secret key is generated it has to be replaced as the password for that device in the list of paired devices on the server. Each request/response pair is encrypted and decrypted using the agreed-upon secret key/password. To address elements of the threat model, before processing a request, the server first checks if the Device Public Key (Device ID) included in the request is in the list of paired devices. If so, it then checks if the deployment duration set in the bootstrapping process has not expired (Section III-C1). If it has not expired, it retrieves the password for that device and uses it to decrypt the request. The request is processed and the response is encrypted using the same password.

5) Automatic and Manual Device Credential Revocation: To address elements of the threat model, the solution includes two ways of revoking device credentials, therefore terminating the connection between the server and the mobile device.

- Automatic due to deployment timeout: As mentioned in Section III-C1, the last step of the bootstrapping process is to set a deployment duration. The server will automatically disable all device credentials at the end of the configured duration. This means that server will no longer accept pairing or API requests from any device until a new deployment is configured.
- Manual due to known loss or compromise: To address elements of the threat model, the solution requires a way to manually revoke device credentials if a device is known to have been lost or stolen. Once credentials are revoked, the device is removed from the list of paired devices, and the server will no longer accept API requests from that device.

IV. Implementation

A. Overall Architecture and Design

The architecture of the tactical cloudlet implementation is presented in Figure 2 and is based on the architecture presented in [26]. As shown in Figure 2, Cloudlet Clients have **Cloudlet-Ready Apps**, which are applications that can access cloudlet services through an HTTP API over WiFi, the **Pycloud API**, provided by the cloudlet. The **Cloudlet Client App** allows a mobile device to setup Cloudet-Ready Apps and monitor the status of a cloudlet. A Cloudlet Host has a web-based local management interface, called **Pycloud Cloudlet Manager**, that allows the Cloudlet Admin to manage and configure the cloudlet and its services. The **Avahi Daemon** component on the Cloudlet Host allows discovery of the Pycloud API though Zeroconf [27] once a Client has connected to the cloudlet's network.

Each capability that is made available to apps is considered a **Service**. Each service has associated metadata (Service Metadata), the actual capabilities packaged as VM disk and memory images (VM Images), and one or more Cloudlet-Ready Apps that can use the capability. Services are stored in a **Service Repository** inside the Cloudlet Host.

B. Security Components

The following are the system changes and new components that implement our trusted identity solution:

1) **Bootstrapping and Pairing Processes**: A new interface was added to the Cloudlet Manager that enables the execution of the Bootstrapping and Pairing processes needed to set up a mission/deployment, and to pair devices securely to the cloudlet. The Cloudlet Client App was also updated to handle the device side of the Pairing process.
2) **FreeRADIUS Server**: FreeRADIUS is an open-source implementation of a RADIUS server [28] that was added to the Cloudlet Host to handle WiFi authentication. The Access Point for the cloudlet is configured with WPA2-Enterprise in order to authenticate devices connecting to the network through this FreeRADIUS server (Section IV-C).
3) **Secure-Key Agreement (SKA) Package**: This package was added to Pycloud to handle the key and credential exchange during the Pairing process. The SKA package implements a communication protocol that can work over Bluetooth or USB to securely pair a device to a cloudlet for the duration of a mission/deployment (Section IV-C). The Cloudlet Client App was updated to implement this protocol. The Cloudlet Client App also handles the creation of the WiFi profile that will allow authentication with FreeRADIUS.
4) **Security Package**: This package was added to Pycloud to handle the credentials for the server and its paired devices. The package creates the appropriate server credentials and device credentials during the Bootstrapping and Pairing processes, and manages the repositories where the credentials are stored. It manages the list of devices that have been paired with the system and can revoke authorization if requested. The Security Package is also responsible for properly configuring the FreeRADIUS server each time a new device is paired or unpaired with the cloudlet. In addition, the package is responsible for encrypting and decrypting communication going through the Pycloud API (Section IV-C). The client side also has a simpler, similar package that stores and retrieves the credentials stored on the device, which can then be used by the Cloudlet Client App or any other Cloudlet-Ready App to encrypt the communication with the Pycloud API.

C. Communication Protocols

There are four different communication protocols that were modified or added to our system to implement our trusted identities solution.

1) **Secure Key Agreement (SKA) Protocol** (red arrow in Figure 2): This protocol is used during the pairing process. Although the protocol itself is independent of the method used to transfer its messages, it is intended to be used when the two devices are in proximity. The two implementations included in the system are for Bluetooth and serial USB connection message transfer. The protocol follows a client-server model in which the Pycloud Cloudlet Manager acts as the client, and a mobile device with the Cloudlet Client App acts as the server. All messages and their replies are structured as JSON text objects. The protocol consists of three messages and their replies: **Get Data** (from the mobile device), **Send Data** (to the mobile device) and **Send File** (to the mobile device).
   The Bluetooth implementation uses service discovery and RFCOMM [30] to connect to a device and communicate with it. The Cloudlet Client App provides a Bluetooth RFCOMM server that is started during the pairing process to receive and respond to requests.
   The USB implementation uses Android's ADB [31] for communication. The ADB Daemon running on the mobile device is used to receive commands and route them to the appropriate handlers in the Cloudlet Client App that push or pull JSON or binary files.
   During the pairing process, the Get Data command is used to obtain the Device ID. The Send File command is used to send credential information (certificates and keys); and the Send Data command is used to send a command to create the WiFi profile, along with the required information for the profile.
2) **EAP over 801.X and RADIUS Protocol** (light green arrows in Figure 2): Used for WPA2-Enterprise WiFi Authentication. The WiFi profile on the device and the FreeRADIUS configuration are set by our code, but the actual authentication occurs outside of our components, between Android's wpa_supplicant, the Access Point, and FreeRADIUS. The authentication method used is

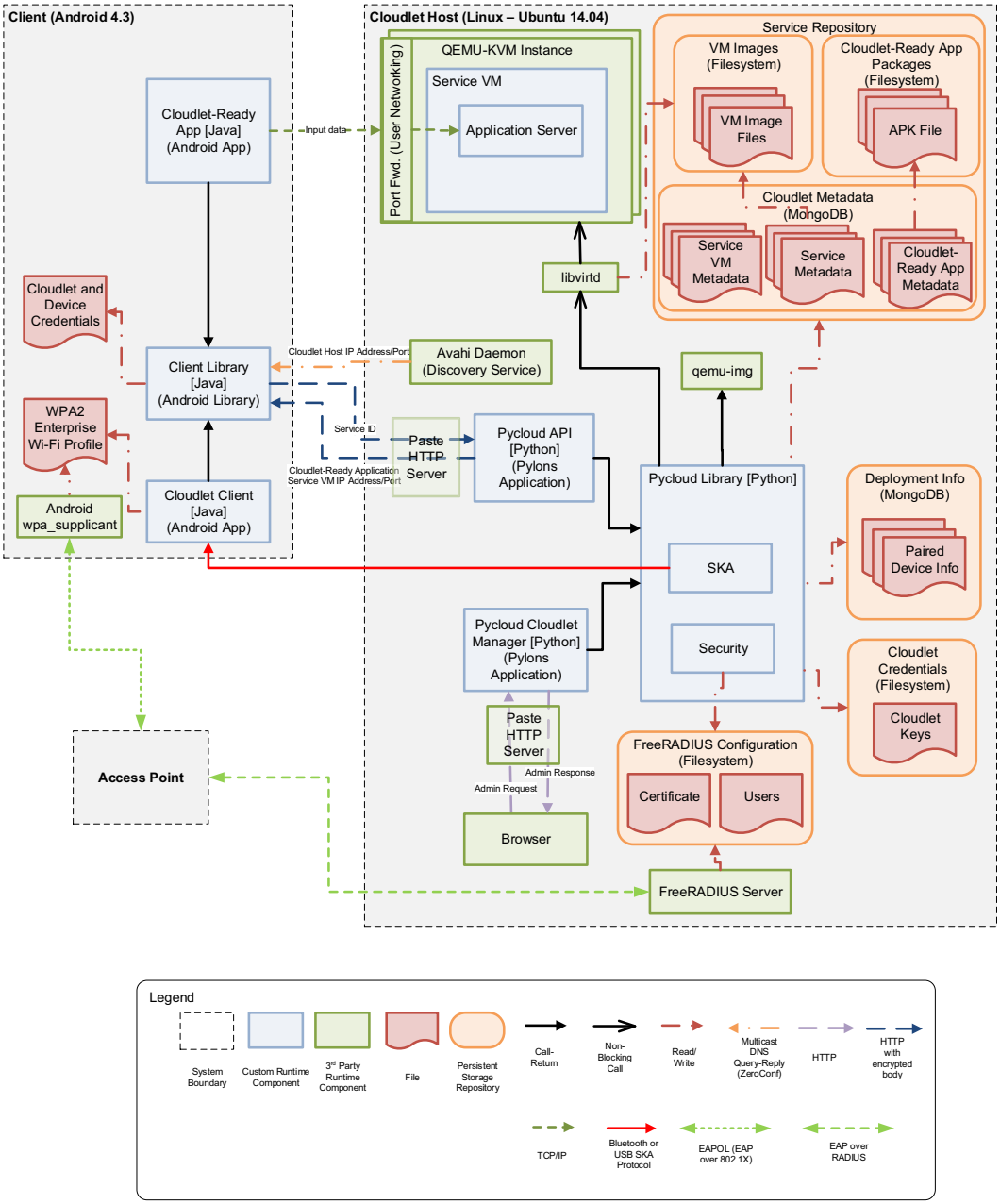

Fig. 2. Tactical Cloudlet Architecture

EAP-TTLS PAP, with the credentials being the Device ID and the Device BLS Certificate set up during the pairing process.

3) **Pycloud API Protocol** (blue arrows in Figure 2): These are REST HTTP requests to the Pycloud API. The replies for most of these messages is data in JSON format. The current API includes commands to get information about the capabilities of the cloudlet, a list of services and specific service information, a list of apps and APKs for specific apps, and commands to start and stop Service VMs for specific services.

Because all of these messages are self-contained requests, there is no session between the mobile device and the cloudlet. Access to this API is implemented in the *Client Library*, which is used by the Cloudlet Client App and any Cloudlet-Ready App.

If encryption is enabled, the API works differently. In this case there is only one possible message, which

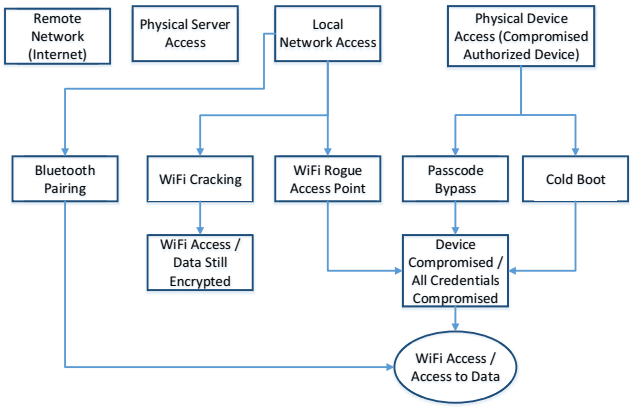

Fig. 3. Attack Tree.

includes the Device ID as an HTTP header, and only one HTTP parameter, *command*, that contains the actual command to be executed. The value of this parameter is a URL string corresponding to one of the unencrypted API messages mentioned above. This string is sent encrypted with a symmetric 256-bit AES key associated to the device, so that only the device and the Pycloud API that the device is paired to are able to decrypt it. The password used to generate the key is the SHA-256 of the Device Private Key created during the pairing process with the device. Replies are also encrypted with this symmetric key.

V. Evaluation

A. Evaluation Against Threat Model

The first form of evaluation that we conducted was against the threat model defined in Section III-A. How each threat was fully addressed, partially addressed, addressed outside the solution, or not addressed is described in Table III.

In summary, the solution fully addresses five of the 14 threats and partially addresses one of the threats due to an explicit tradeoff. Six of the 14 threats are addressed outside of the solution, and will be further addressed in the Ceremony Analysis (Section V-C). Two threats are not addressed by the solution but we provide potential mitigations that we did not implement.

B. Vulnerability Analysis

We additionally evaluated the trusted identities solution by performing architectural and technical analysis of possible vulnerabilities based on the threat model (Section III-A). Given this set of threats, we first created a simple attack tree [32] to determine potential attack vectors, as shown in Figure 3. As an attacker, our overall goal would be to gain access to the data being shared between clients and the server. We identified four possible paths to access that data, of which only two are relevant when considering the trusted identity solution.

1) Remote Network (Internet) Access: This is not relevant because the clients and server are expected to be operating in a disconnected environment.
2) Physical Server Access (including via a USB port): We exclude this threat because physical security controls in a disconnected environment are outside the scope of the identity solution. That is, physical attacks on the server would bypass the identity solution credentials.
3) Local Network Access: This includes Bluetooth or WiFi access from an unauthorized device.
4) Physical Device Access (Compromised Authorized Device): We include physical security of the mobile device because it is much more prone to loss or theft than the server, and could lead to compromise of the identity solution.

The following sections address the relevant paths of the attack tree shown in Figure 3.

1) Local Network Access: Regarding Bluetooth Pairing, Bluetooth is known to be vulnerable to attack in certain configurations [33]. Although Bluetooth is a short range protocol, it can be received within ˜100 meters. In urban or disaster scenarios, the pairing process may happen in crowded environments where this is a risk. We configured the Bluetooth pairing process to use Numeric Comparison pairing, which has been proven to be secure [34].

In our solution, the WiFi credentials consist of the Device ID as the username and a SHA-256 hash of the Device BLS Certificate as the password. SHA-256 is considered secure against current attacks by the U.S. National Institute of Standards and Technology (NIST) [35]. Furthermore, the SSID is user-selected, making it impossible to create rainbow tables even if a user intentionally weakened the password [36].

The other threat to WiFi is that of the user connecting to a WiFi Rogue Access Point and inadvertently sharing its credentials. The RADIUS Server Certificate created in the bootstrapping process (Section III-C1) prevents this. The device acquires the certificate at the same time as it acquires its credentials, and the SSID and certificate are automatically configured to use those credentials. Android will not allow the user to connect to another access point with that SSID unless it can present the RADIUS Server Public Key matching the stored RADIUS Server Certificate. A user could possibly connect to an SSID with a confusingly similar name, but the stored credentials would not be sent to that SSID. The only way an attacker could potentially spoof a legitimate access point would be with the RADIUS Server Private Key, which is never transferred to the device.

Our final mitigation against WiFi attacks is the fact that all API requests are encrypted with the full Device BLS Certificate using AES [37]. Even if an attacker gained access to the WiFi network, they would only see encrypted data. The WiFi password is a SHA-256 hash of the Device BLS Certificate, which prevents attackers who obtain the WiFi password from discovering the certificate itself.

2) Physical Device Access (Compromised Authorized Device): The device contains all of the information needed

TABLE III
EVALUATION OF TRUSTED IDENTITY SOLUTION AGAINST THREAT MODEL

Threats Fully Addressed	
Threat	**Mitigation**
(1) Impersonating a device	During the pairing process (Section III-C2), the cloudlet sends the RADIUS Server the Device Public Key and Device BLS Certificate Hash. The RADIUS Server stores these credentials in a list of authorized devices. During the WiFi authentication process (Section III-C3) the device authenticates with the RADIUS Server by sending its stored Device Public Key and Device BLS Certificate Hash. The communication continues only if these match. Finally, the last step of the bootstrapping process (Section III-C1) is that the device private key is deleted on the server, which means that the only place in which it resides is the device.
(2) Finding an active client	This threat is mitigated through several mechanisms: (1) the last step of the bootstrapping process (Section III-C1) is to set a deployment duration, (2) device credential revocation mechanisms (Section III-C5) are implemented so that all API Requests are rejected after deployment expiration time and a device can be manually deleted in the Cloudlet Manager if lost or compromised, and (3) each API Request is validated against the list of valid devices.
(3) Finding a device	The pairing process (Section III-C2) requires physical proximity to the cloudlet to be able to connect via Bluetooth or USB, plus visual confirmation from the cloudlet admin.
(7) Sniffing wireless	The system performs transport- and message-level encryption. WPA2-Enterprise (802.1X) CCMP (AES) encryption with a 128-bit key based on a 256-bit password is used at the transport level. The body of each HTTP message is encrypted with AES (CBC) with a 256-bit key and a random IV at the message level.
(14) Server impostor	The bootstrapping process (Section III-C1) creates a RADIUS Server Certificate that is validated by the device in the WiFi authentication process (Section III-C3). During the pairing process (Section III-C2), the certificate is sent to the mobile device. During the WiFi authentication process, the device asks the RADIUS Server for its certificate, which has to be sent before that device can trust the network. If the certificates do not match, the device will not connect to the corresponding WiFi network.
Threats Partially Addressed	
Threat	**Mitigation**
(6) Lost credentials	The main mitigations are the manual and automatic device credential revocation processes described in Section III-C5. All API Requests include the Device Public Key (Device ID) which is validated against the list of paired devices before responding to a request. Anything stronger would require TPM and/or encrypting the Device Credentials (Device Public Key and Device BLS Certificate), which affects usability because a password would have to be entered for every interaction. A possibility would be for the password to be cached on the device. If an attacker obtains the Device Credentials and RADIUS Server Certificate, it would not be able to use them without knowing the password, which is only cached on the device itself.
Threats Addressed Outside the Implementation	
Threat	**Mitigation**
(8) Site intrusion	In addition to requiring strong passwords for the root user and the cloudlet admin, the server would have to reside in a safe, protected site.
(9) On the net	The server is disconnected from the network. The cloudlet only accepts connections that are authorized by the RADIUS Server.
(10) On the box	Strong passwords for the root user and the cloudlet admin are required.
(11) Super-user compromise	Cloudlets only have two users: root and cloudlet admin. Strong passwords for the root user and the cloudlet admin are required. Cloudlet admin does not know the root password. Cloudlet admin account does not have root privileges.
(12) Application compromise	There are settings in place so that the Cloudlet Manager can only be run locally.
(13) Seeing everything	Strong passwords for the root user and the cloudlet admin are required. Service VMs are responsible for encrypting data residing within the VM.
Threats not Addressed	
Threat	**Potential Mitigation**
(4) Altered software	Mitigation would require integration with TPM or code signing.
(5) Daisy chaining	Mitigation would require device controls that do not allow connections to a mobile device.

to connect to the server and decrypt data. Loss or theft of the device potentially compromises that information. We primarily mitigate this threat by establishing the two methods of revoking the credentials of a lost or stolen device (Section III-C5): automatic due to deployment timeout and manual due to known device loss or compromise. This leaves a potential window of vulnerability between an adversary obtaining a device and the credentials being revoked. Although Android has been susceptible to passcode bypasses [38], a properly patched and encrypted device should be sufficiently resistant to attack for the amount of time required to notice the loss and rescind the credentials. Android drive encryption has also been shown to be susceptible to Cold Boot attacks [39], but these attacks take time due to the need to chill the device. The use of one-time passwords would alleviate these vulnerabilities, but at a cost of making the device much more cumbersome to personnel in crisis environments.

The device is assumed to be fully patched, encrypted and protected with a passcode. Further, it is assumed that no unauthorized or vulnerable software has been installed or side-

loaded, and that it does not contain a SIM card for cellular data connection. Addition of any of these functions would expand the attack surface [40] and therefore the threat model would have to be adjusted to account for the greater attack surface.

After navigating the attack tree (Figure 3) and addressing the vulnerabilities, we modeled all possible states of the encryption entities to ensure there were no unexpected states that produced vulnerabilities. Through the life cycle of a given device, we enumerated the following state changes and the related movement of encryption entities.

1) State 0: The device contains its public key (Device ID); the server contains no entities.
2) State 1 (Bootstrapping): Server creates the Server Public Key, Server Private Key, RADIUS Server Public Key, RADIUS Server Private Key, RADIUS Server Certificate.
3) State 2 (Pairing - Phase 1): Server acquires Device Public Key.
4) State 3 (Pairing - Phase 2): Device acquires Server Public Key, RADIUS Server Certificate, Device Private Key, Device BLS Certificate
5) State 4 (WiFi Authentication): Device re-acquires RADIUS Server Certificate which contains the RADIUS Server Public Key.
6) State 5 (Deployment Timeout): Server removes all paired Device Credentials and deletes all encryption entities.

Based on these state changes, we determine that at no point does the device gain access to an encryption entity that could be used to exceed its level of authorization. Specifically, the Server Private Key or RADIUS Server Private Key would be needed to violate the confidentiality or integrity of the protected communications.

We do not address availability attacks because Denial of Service (DoS) attacks would likely be analog or kinetic rather than digital in a disconnected environment.

C. Ceremony Analysis

The concept of ceremony extends the concept of network protocol by including human beings as nodes in the network [41]. Ceremonies include all protocols, applications with a user interface, and security provisioning workflows. In essence, there is nothing out of band in a ceremony. Security analysis of a node in a ceremony is the same for both human and computer nodes: With what probability will an attacker be able to fool the node into making an incorrect decision?

As Table III shows, there are multiple threats in the threat model that are addressed outside of the implementation. In essence, these are assumptions that are made by the solution and left open for arbitrary people to satisfy in arbitrary ways [41]. In addition, the pairing process requires an out-of-band channel (physical proximity plus visual confirmation). These are all elements that benefit from a ceremony analysis in order to provide guidance on how to validate or enforce all elements of the end-to-end solution.

Ceremony nodes in the exchange of credentials between a cloudlet and a mobile device are shown in Figure 4. The human nodes are:

- Mobile User: Owner and user of the mobile device
- Cloudlet Admin: User responsible for operating a cloudlet in the field
- Cloudlet Provider: User responsible for setting up a cloudlet for use in the field. The Cloudlet Provider physically hands over a cloudlet to a Cloudlet Admin.

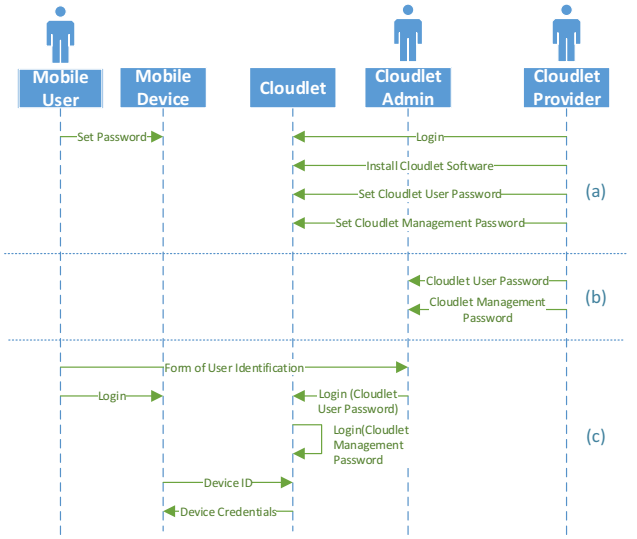

Fig. 4. Ceremony between a Mobile Device and a Cloudlet

With the exception of the *Device ID* and *Device Credentials* exchange shown in Figure 4 all other exchanges between nodes in the ceremony are considered out-of-band for the trusted identities solution. The ceremony analysis therefore demonstrates the need to provide assurances for all other exchanges.

(a) Cloudlet Setup
  - Password used by the *Cloudlet Provider* to login to set up a cloudlet (i.e., root or another user with admin privileges) must follow rules for strong passwords.
  - Installing cloudlet software by the *Cloudlet Provider* must be an automated process that does not install extra software on the cloudlet that could potentially compromise the cloudlet.
  - Passwords created for the *Cloudlet User* (to log into the physical machine) and *Cloudlet Management* (to log into the Cloudlet Manager application) must follow rules for strong passwords.
  - Password set by the *Mobile User* on his/her *Mobile Device* must follow rules for strong passwords. This is harder to enforce on personal mobile devices (i.e., BYOD).

(b) Cloudlet Delivery: The *Cloudlet User Password* and the *Cloudlet Management Password* have to be delivered in

a secure way to the *Cloudlet Admin*. An alternative is for the *Cloudlet Admin* to set up his/her own set of passwords following the same rules for strong passwords.

(c) Device Credential Exchange: Valid forms of *User Identification* presented by a *Mobile User*, and validation mechanisms for the *User Identification*, need to be defined such that the *Cloudlet Admin* only starts the pairing process after confirmation of a valid user.

VI. Summary and Conclusions

The paper presented a trusted identity solution for disconnected environments that combines Identity-Based Cryptography (IBC) with mechanisms for Secure Key Exchange without a Trusted Third Party. The solution was developed based on a threat model for disconnected environments and implemented in our open-source tactical cloudlets project targeted at deployment in these types of environments. Evaluation of the implementation was done against the threat model and using vulnerability analysis. The results show that it is a resilient solution that addresses most of the threats and characteristics of disconnected environments if combined with proper application-, OS-, network- and site-level controls. An additional ceremony analysis was conducted to provide guidance on threats that are addressed outside of the trusted identity solution. Even though the solution was implemented in the the context of a client/server tactical cloudlets systems, we believe the solution could be applied to any form of trusted communication between two or more computing nodes. Future work will focus on expanding the threat model to a connected environment, and identity federation across multiple computing nodes.

The implementation of the tactical cloudlets system that includes the developed trusted identity solution is available at https://github.com/SEI-AMS/pycloud/wiki.

Acknowledgment

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. [Distribution Statement A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution (DM-0003391).

References

[1] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *Pervasive Computing, IEEE*, vol. 8, no. 4, pp. 14–23, 2009.

[2] Y.-H. Lin, A. Studer, Y.-H. Chen, H.-C. Hsiao, L.-H. Kuo, J. Lee, J. M. McCune, K.-H. Wang, M. Krohn, P.-L. Lin, A. Perrig, H.-M. Sun, and B.-Y. Yang, "SPATE: Small-group PKI-less authenticated trust establishment," *IEEE Transactions on Mobile Computing*, vol. 9, pp. 1666–1681, 2010.

[3] *ISO/IEC 11889-1:2009 - Information Technology - Trusted Platform Module - Part 1: Overview*, International Organization for Standardization Std., 2009.

[4] C. Marforio, N. Karapanos, C. Soriente, K. Kostiainen, and S. Capkun, "Secure enrollment and practical migration for mobile trusted execution environments," in *Proceedings of the Third ACM workshop on Security and privacy in smartphones & mobile devices*. ACM, 2013, pp. 93–98.

[5] B. Parno, J. M. McCune, and A. Perrig, "Bootstrapping trust in commodity computers," in *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 2010, pp. 414–429.

[6] L. Chen, "An interpretation of identity-based cryptography," in *Foundations of security analysis and design IV*. Springer, 2007, pp. 183–208.

[7] M. Farb, Y.-H. Lin, T. H.-J. Kim, J. M. McCune, and A. Perrig, "Safeslinger: Easy-to-use and secure public-key exchange," in *Proceedings of ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2013.

[8] J. Han, Y.-H. Lin, A. Perrig, and F. Bai, "MVSec: Secure and easy-to-use pairing of mobile devices with vehicles," Carnegie Mellon University, Tech. Rep. CMU-CyLab-14-006, May 2014.

[9] J. Sen, P. R. Chowdhury, and I. Sengupta, "A distributed trust establishment scheme for mobile ad hoc networks," in *Computing: Theory and Applications, 2007. ICCTA'07. International Conference on*. IEEE, 2007, pp. 51–58.

[10] D. Kidston, L. Li, H. Tang, and P. Mason, "Mitigating security threats in tactical networks," DTIC Document, Tech. Rep., 2010.

[11] J. Undercoffer, F. Perich, A. Cedilnik, L. Kagal, and A. Joshi, "A secure infrastructure for service discovery and access in pervasive computing," *Mobile Networks and Applications*, vol. 8, no. 2, pp. 113–125, 2003.

[12] M. Conti, S. K. Das, C. Bisdikian, M. Kumar, L. M. Ni, A. Passarella, G. Roussos, G. Tröster, G. Tsudik, and F. Zambonelli, "Looking ahead in pervasive computing: Challenges and opportunities in the era of cyber–physical convergence," *Pervasive and Mobile Computing*, vol. 8, no. 1, pp. 2–21, 2012.

[13] H. Aronowitz, M. Li, O. Toledo-Ronen, S. Harary, A. Geva, S. Ben-David, A. Rendel, R. Hoory, N. Ratha, S. Pankanti, and D. Nahamoo, "Multi-modal biometrics for mobile authentication," in *Biometrics (IJCB), 2014 IEEE International Joint Conference on*, Sept 2014, pp. 1–8.

[14] T.-Y. Chang, C.-J. Tsai, and J.-H. Lin, "A graphical-based password keystroke dynamic authentication system for touch screen handheld mobile devices," *Journal of Systems and Software*, vol. 85, no. 5, pp. 1157 – 1165, 2012.

[15] G. D. Clark and J. Lindqvist, "Engineering gesture-based authentication systems," *Pervasive Computing, IEEE*, vol. 14, no. 1, pp. 18–25, Jan 2015.

[16] T. Feng, Z. Liu, B. Carbunar, D. Boumber, and W. Shi, "Continuous remote mobile identity management using biometric integrated touch-display," in *Microarchitecture Workshops (MICROW), 2012 45th Annual IEEE/ACM International Symposium on*. IEEE, 2012, pp. 55–62.

[17] Microsoft Corporation, "SDL Threat Modeling Tool," 2015. [Online]. Available: https://www.microsoft.com/en-us/sdl/adopt/threatmodeling.aspx

[18] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," *SIAM Journal on Computing*, vol. 32, no. 3, pp. 586–615, 2003.

[19] B. Maheshwari, "Secure key agreement and authentication protocols," *International Journal of Computer Science and Engineering Survey*, vol. 3, no. 1, p. 113, 2012.

[20] B. Lynn, "Authenticated identity-based encryption," *IACR Cryptology ePrint Archive*, vol. 2002, p. 72, 2002.

[21] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," *Journal of Cryptology*, vol. 17, no. 4, pp. 297–319, 2004.

[22] *RFC 2865: Remote Authentication Dial In User Service (RADIUS)*, Internet Engineering Task Force Std. [Online]. Available: https://tools.ietf.org/html/rfc2865

[23] *IEEE 802.11-2012: Standard for wireless local area networks (WLANS)*, IEEE Standards Association Std., 2012. [Online]. Available: https://standards.ieee.org/findstds/standard/802.11-2012.html

[24] *RFC 5281: Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)*, Internet Engineering Task Force Std. [Online]. Available: https://tools.ietf.org/html/rfc5281

[25] G. Lewis, S. Echeverria, S. Simanta, B. Bradshaw, and J. Root, "Tactical cloudlets: Moving cloud computing to the edge," in *Military Communications Conference (MILCOM), 2014 IEEE*, Oct 2014, pp. 1440–1446.

[26] S. Echeverria, G. Lewis, J. Root, and B. Bradshaw, "Cyber-foraging for improving survivability of mobile systems," in *Military Communications Conference (MILCOM), 2015 IEEE*, Oct 2015, pp. 1454–1459.
[27] "Zero Configuration Networking (Zeroconf)." [Online]. Available: http://www.zeroconf.org/
[28] The FreeRADIUS Server Project and Contributors, "The FreeRADIUS Project," 2015. [Online]. Available: http://http://freeradius.org/
[29] The Avahi Team, "Avahi," 2014. [Online]. Available: http://avahi.org
[30] Bluetooth Developer Portal, "RFCOMM with TS 07.10," 2015. [Online]. Available: https://developer.bluetooth.org/TechnologyOverview/Pages/RFCOMM.aspx
[31] Android, "Android debug bridge," 2015. [Online]. Available: http://developer.android.com/tools/help/adb.html
[32] B. Schneier, "Attack trees," *Dr. Dobb's Journal*, vol. 24, no. 12, pp. 21–29, 1999.
[33] A. Y. Lindell, "Attacks on the pairing protocol of bluetooth v2. 1," *Black Hat USA, Las Vegas, Nevada*, 2008.
[34] ——, "Comparison-based key exchange and the security of the numeric comparison mode in bluetooth v2. 1," in *Topics in Cryptology–CT-RSA 2009*. Springer, 2009, pp. 66–83.
[35] *FIPS PUB 180-4: Secure Hash Standard (SHS)*, National Institute of Standards and Technology Std., 2015. [Online]. Available: http://dx.doi.org/10.6028/NIST.FIPS.180-4
[36] *RFC2898: PKCS #5: Password-Based Cryptography Specification Version 2.0*, Internet Engineering Task Force Std., 2000. [Online]. Available: https://tools.ietf.org/html/rfc2898
[37] *FIPS PUB 197: Advanced Encryption Standard (AES)*, National Institute of Standards and Technology Std., 2001. [Online]. Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
[38] J. Gordon, "Android 5.x Lockscreen Bypass (CVE-2015-3860)," 2015. [Online]. Available: http://sites.utexas.edu/iso/2015/09/15/android-5-lockscreen-bypass/
[39] T. Müller and M. Spreitzenbarth, "Frost," in *Applied Cryptography and Network Security*. Springer, 2013, pp. 373–388.
[40] P. K. Manadhata and J. M. Wing, "An attack surface metric," *Software Engineering, IEEE Transactions on*, vol. 37, no. 3, pp. 371–386, 2011.
[41] C. M. Ellison, "Ceremony design and analysis." *IACR Cryptology ePrint Archive*, vol. 2007, p. 399, 2007.