

GeoPKI: Converting Spatial Trust into Certificate Trust

Tiffany Hyun-Jin Kim, Virgil Gligor, and Adrian Perrig

Carnegie Mellon University
{hyunjin1,virgil,adrian}@ece.cmu.edu

Abstract. The goal of GeoPKI is to enable secure certificate validation (without user interaction) for situations in which a user interacts with an online entity associated with the physical space where the user trusts (and usually is currently located). GeoPKI enables the owner of a space to associate a certificate with that space, and enables space-based certificate lookup to set up a secure channel to the online resource associated with the space. Such a system enables several secure applications, such as secure authentication of paywall certificates at an airport or hotel.

Keywords: Geographic Public-Key Infrastructure (PKI), certificate associated with geographic space, coordinate-based certificate lookup.

1 Introduction

Consider a world in which a physical space can be uniquely associated with a public key. Here, a user who is physically in a space can look up the associated public key for that space. Consequently, if the user trusts the space she is in, she can also trust the public key and thus set up a secure channel with any online resource that is associated with that space. Such an approach has several advantages:

- People understand physical spaces, and they associate trust with a physical location, as they can return to that same location in case of a dispute. In essence, the physical location creates accountability, as physical presence establishes a point of legal approachability (i.e., police can be sent to the physical location).
- A public key that is correctly associated with a physical space can create accountability for the online resource through the accountability of the physical space. We will elaborate on how “correct association” can be accomplished.
- By using the geographic coordinates to look up the certificate, certificate verification can be completely automated, and without any user intervention, we avoid several security issues: social engineering attacks (e.g., `my-secure-bank.com`), homophone attacks (e.g., `goggle.com`, `bankofthevest.com`), etc.

Given a system that maps a public key to a space, we can address several challenging security problems: (1) Consider the authentication of a certificate of a paywall¹ at an airport or hotel login page to access the wireless network. How can

¹ A paywall is a server which authenticates users or collects payments on airport or hotel networks.

a mobile device verify that the certificate offered by the login page is indeed from the local entity and not from an evil twin access point? Association of a public key with a space will address this issue, as the unique certificate that is associated with a space can be used to establish a secure connection. (2) Consider accessing the web page associated with the bank the user is currently at via SSL/TLS. Using the public key associated with the local space can assist establishment of a secure connection. (3) Consider a wireless payment protocol at a shop, where a shopper’s cell phone implements a digital wallet. In this case, the certificate associated with the local space can be used to set up the secure connection with the payment infrastructure.

In this paper, we design and analyze GeoPKI, a system that associates public keys with physical spaces. GeoPKI is a geography-based Public-Key Infrastructure extension that (1) creates a hierarchy of digital certificates that maps public keys to *both* entities and geographic locations, (2) securely stores these certificates in append-only public repositories, and (3) revokes them if needed. Note that our public repositories offer a service that can prevent equivocation,² in the same vein as recent proposals for public log servers [1, 5, 8].

It may appear that current PKIs could achieve similar properties by adding 3D coordinates to certificates. However, such an approach cannot provide *geographic coordinate-based lookups* and thus fails to efficiently detect situations in which the same physical space is claimed by multiple entities. Furthermore, Certificate Authorities (CAs) lack global name-space resolution, even for current domain names. For name resolution, DNS-based Authentication of Named Entities (DANE) is a promising approach that provides additional information about cryptographic credentials associated with the domain [3]. However, we still need a 3D data structure to map a physical space to a key. In this paper, we focus on designing such a 3D data structure that is efficient not only for certificate storage and retrieval, but also for certificate validation.

The contribution of this paper is to propose and study an environment where users can perform a certificate lookup based on geographic coordinates and obtain a hierarchy of certificates that are associated with that space. We explore the security and trust establishment implications, and find that several current, challenging problems are addressed by our GeoPKI framework.

2 Problem Definition

In this paper, we explore how to associate a public key with a physical space, and use this association to help users validate online resources. This approach assumes a service for certificate lookups based on geographic coordinates.

² In equivocation, an entity provides different answers to different queries; preventing equivocation thus means that the same query always results in the same response in a globally consistent manner.

2.1 Desired Properties

1. **Correct location-to-certificate association.** The data structure of GeoPKI should contain correct information in order to prevent an entity who does not own a space from associating a certificate with that space.
2. **Response integrity.** We seek globally consistent query replies, such that two different queries for the same coordinates yield the same correct response (assuming that the database has not been updated between the queries), thus preventing equivocation.²
3. **Availability.** Querying and retrieving digital certificates associated with a 3D space should be simple and efficient. The service should remain available even in the case of a DDoS attack.
4. **Efficient validation.** The data structure of GeoPKI should enable efficient validation of query results, including results for non-existing database entries.

2.2 Assumptions

We assume that users know their correct geographic location, both indoor and outdoor. Several efforts are currently under way to provide accurate indoor localization, such as Qualcomm’s system, which combines several sensor modalities to achieve accurate (on the order of 1 meter) indoor localization.³ We also assume that GPS and location spoofing attacks can be addressed by combining localization methods with three sources of information: (1) the position of known landmarks that are detectable and that users frequently pass by (e.g., location of a home, office, restaurant, or grocery store), (2) the use of inertial navigation, and (3) the use of path constraints such as road maps or walls [7]. If these approaches are used in concert, location spoofing would need a sophisticated local attacker to divert a user. Moreover, methods for secure localization are under active investigation [11]. Therefore, we assume that it is possible for future systems to become resilient to location spoofing.

We assume the presence of a publicly accessible append-only log service that keeps track of changes to GeoPKI certificates and makes all changes publicly visible, similar to Certificate Transparency [1] and Sovereign Keys [5]. This public log reduces the level of trust we need to place in the operator of the GeoPKI service, without requiring trusting the log service except for providing availability of the information. As the log can be publicly validated, any divergence from correct operation can be publicly proven. We assume that current map operators or large-scale Internet businesses would host the GeoPKI service. We also assume that a client has the correct public key of the GeoPKI service, and that the private keys of entities and CAs are secure.

2.3 Attacker Model

We consider an adversary whose main goal is to associate his own public key with a space that is occupied and owned by someone else. An adversary has the following properties:

³ <http://www.qualcomm.com/about/research/projects/indoor-positioning>

- **Control of network communication:** An adversary may control all network communication, including dropping, eavesdropping, and manipulating packets. However, we do assume that the network provides availability for communication (i.e., we consider DoS attacks as outside the scope of this paper).
- **Physical presence:** An adversary may be physically present at a victim’s location.

3 GeoPKI Overview

Before discussing the details, we briefly provide an overview of GeoPKI. More specifically, we explain (1) how an entity can register a certificate containing geographic information, (2) how a user can search for a certificate at a given location, and (3) how the retrieved certificate can be used for validation.

Registration. Alice owns some physical space and generates her own public-private key pair. She then acquires a self-signed or CA-signed certificate for GeoPKI, which includes the coordinates of her physical space. In essence, a certificate for GeoPKI not only associates her public key with her name (i.e., URL of the associated online entity), but also binds her public key to her physical space. Then Alice contacts a public append-only Log Server (LS) [1, 5] to record her certificate for the claimed physical space. Note that the LS’s public append-only feature reduces trust in the GeoPKI database, since any change to an entry is logged; as a result, entities can query past certificates and verify the history of key changes, as presented in Certificate Transparency [1] and Sovereign Keys [5].

After logging Alice’s request, the LS signs Alice’s certificate as a proof that her claimed space has been recorded. With LS’s signature, Alice contacts the public GeoPKI database, which maintains a Merkle hash tree for integrity. Alice requests to register her certificate. The GeoPKI database then (1) verifies LS’s signature on Alice’s certificate, (2) records Alice’s certificate, and (3) updates its Merkle hash tree.

Note that the GeoPKI database assigns a cooling period (e.g., 1 week) to prevent an attacker from filing an incorrect registration. As long as no other entity attempts to revoke Alice’s registration request, her certificate becomes publicly accessible after the cooling period.

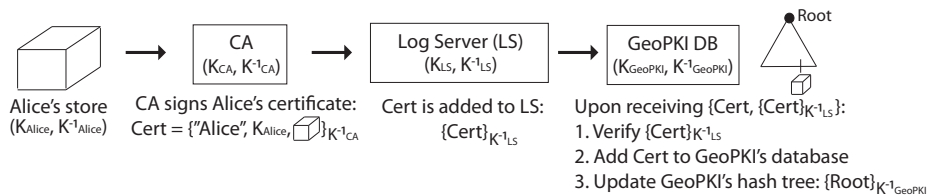


Fig. 1. Registration of Alice’s certificate to the GeoPKI database. (K_E, K_E^{-1}) represents a public-private key pair of an entity E.

Certificate Lookup. When Bob wants to access the resource R that is associated with Alice’s physical space, Bob’s browser queries the public GeoPKI database which is hosted by current map providers or a large-scale Internet business. If an entry exists for the queried physical space, the database returns the GeoCert, which contains Alice’s name, public key, and self-signed or CA-signed certificate, along with the signed root hash value of the updated Merkle hash tree ($\{Root\}_{K_{GeoPKI}^{-1}}$) and the intermediate hash values that are needed to verify Alice’s certificate.

The GeoPKI database prevents equivocation as follows: The database maintains a Merkle hash tree over all the entities, including Alice’s. As a result, changes in stored certificates result in changes in the root value, and the browser can detect equivocation with inconsistent root values. The signed hash tree root value also enables public defamation in situations where inconsistent root values were signed during a time period when only a single tree was valid.

Use of Certificates. Upon retrieving the GeoCert for Alice’s physical space, Bob’s browser can verify the name (URL), geographic location, and Alice’s public key (by comparing the root hash value with the hash computation over Alice’s certificate and the provided intermediate hash values). Bob can now trust that he obtained the correct certificate associated with Alice’s location.

4 GeoPKI Public Database Structure

In this section, we describe the 3D data structure that can map physical space to a cryptographic credential.

We describe how 3D-geographic spaces can be expressed for efficient querying and retrieval. We base our data structure on a combination of

- k -d trees ($k = 3$) for simplified querying of 3D spaces (desired property 1),
- Merkle hash trees for integrity validation (desired property 2).

Although other data structures exist for geographic space queries [2, 6, 12], we propose the 3D data structure for GeoPKI for efficient integrity validation using Merkle hash trees.

We represent the globe by a unit cube where the globe radius is normalized to one unit. Hence, the scale between the physical object (on the globe) and the geocentric Cartesian coordinate system is dictated by the normalized globe radius.⁴ Rather than using a standard geodetic system such as WGS 84, we apply this normalization to simplify the data structure. Figure 2 illustrates the Cartesian coordinates of the globe’s unit cube. Note that the globe is repositioned so that the equator lies along the horizontal line of $(-1,0,0)$ through $(1,0,0)$. We assume a publicly available application that translates a set of GPS coordinates into our geocentric Cartesian coordinates and vice versa.

Table 1 shows the fields of an intermediate node (N_i) in the GeoPKI’s k -d tree.

⁴ <http://resources.esri.com/help/9.3/arcgisengine/dotnet/b0e91ce8-c180-47dc-8323-06cac5d77064.htm>

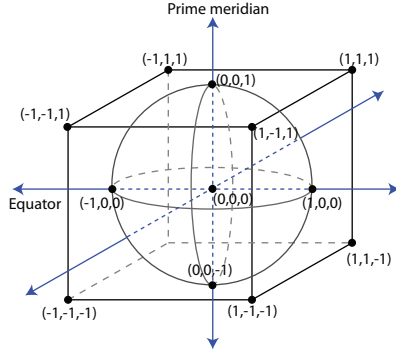


Fig. 2. Cartesian coordinate system for the globe.

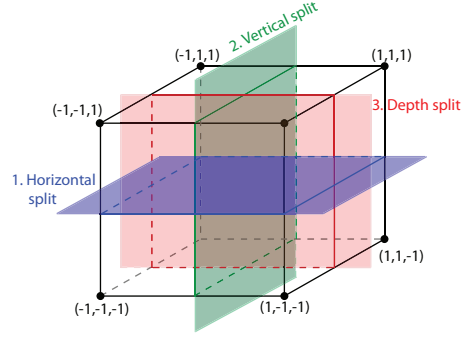


Fig. 3. GeoPKI's 3D data structure.

Table 1. Fields of an intermediate node in GeoPKI's 3D data structure (k -d tree). Each node represents a cuboid in the GeoPKI data structure. This table describes the fields that node N_i contains ($i \in \{0, 1\}^*$).

Field	Description
ptr_i^l	pointer pointing to N_i 's left-child cuboid N_{i0}
ptr_i^r	pointer pointing to N_i 's right-child cuboid N_{i1}
S_i	$\{(x_i, y_i, z_i), l_i, w_i, h_i\}$: set of numbers that uniquely identifies the geographic boundaries of N_i
(x_i, y_i, z_i)	a <i>base coordinate</i> of the cube, which we define to be the left-lowest-forefront vertex of the cuboid
l_i, w_i, h_i	length, width, height of the cuboid N_i
H_i	Merkle hash value of node N_i , namely $H_i = H(H_{i0} H_{i1})$

Initialization. Initially, the k -d tree of the globe (T) contains only 1 node N : $[ptr_\emptyset^l, ptr_\emptyset^r, S_\emptyset, H_\emptyset]$ where

- ptr_\emptyset^l and ptr_\emptyset^r are set to be \emptyset ,
- S_\emptyset , which is $\{(x_\emptyset, y_\emptyset, z_\emptyset), l_\emptyset, w_\emptyset, h_\emptyset\}$, is set to be $\{(-1, -1, -1), 2, 2, 2\}$, and
- H_\emptyset is set to \emptyset .

Consider entity (E_j) located at some physical space (PS_j) that requests a GeoPKI certificate (or GeoCert) at that location. During the search, subdivision into successively smaller cubes is performed until the subdivision results in a cube that is small enough to accommodate *only* E_j 's physical space. Here are three iterative steps that describe the order of the subdivision:

Step $3n - 2$ ($n > 0$). The current cube (N_i) is subdivided along the horizontal split as shown in Figure 3, generating two equal-sized rectangular cuboids: one below the horizontal split (N_{i0}) and one above the horizontal split (N_{i1}). At this point, the k -d tree gets updated with two additional children nodes: a left child (N_{i0}) and a right child (N_{i1}) of N_i . Also, N_i 's fields are updated. The values for N_{i0} , N_{i1} , and the updated N_i are shown in column 3 of Table 2. For Step 1 (when $n = 1$), $S_0 = \{(-1, -1, -1), 2, 2, 1\}$ and $S_1 = \{(-1, -1, 0), 2, 2, 1\}$.

Table 2. Values of GeoPKI data structure nodes when the k -d tree splits. This table shows the values of N_i 's left child N_{i0} (rows 2–8) and right child N_{i1} (rows 9–15) when N_i splits during Step $3n - 2$ (column 3), Step $3n - 1$ (column 4), and Step $3n$ (column 5), where $n > 0$. This table also shows how N_i 's fields are updated after the split.

Node	Field	Step $3n - 2$	Step $3n - 1$	Step $3n$	
N_{i0}	ptr_{i0}^l	\emptyset	\emptyset	\emptyset	
	ptr_{i0}^r	\emptyset	\emptyset	\emptyset	
	S_{i0}	(x_{i0}, y_{i0}, z_{i0})	(x_i, y_i, z_i)	(x_i, y_i, z_i)	(x_i, y_i, z_i)
		l_{i0}	l_i	$\frac{l_i}{2}$	l_i
		w_{i0}	w_i	w_i	$\frac{w_i}{2}$
		h_{i0}	$\frac{h_i}{2}$	h_i	h_i
	H_{i0}	\emptyset	\emptyset	\emptyset	
N_{i1}	ptr_{i1}^l	\emptyset	\emptyset	\emptyset	
	ptr_{i1}^r	\emptyset	\emptyset	\emptyset	
	S_{i1}	(x_{i1}, y_{i1}, z_{i1})	$(x_i, y_i, z_i + \frac{h_i}{2})$	$(x_i + \frac{l_i}{2}, y_i, z_i)$	$(x_i, y_i + \frac{w_i}{2}, z_i)$
		l_{i1}	l_i	$\frac{l_i}{2}$	l_i
		w_{i1}	w_i	w_i	$\frac{w_i}{2}$
		h_{i1}	$\frac{h_i}{2}$	h_i	h_i
	H_{i1}	\emptyset	\emptyset	\emptyset	
N_i	ptr_i^l	N_{i0}	N_{i0}	N_{i0}	
	ptr_i^r	N_{i1}	N_{i1}	N_{i1}	
	S_i	S_{i1}	S_{i1}	S_{i1}	

Step $3n - 1$ ($n > 0$). When the subdivided cuboid is still bigger than the physical space (PS_j) of the entity E_j that is requesting a GeoCert, the sub-cuboid N_i , where PS_j belongs, gets equally divided along the vertical split as shown in Figure 3. Table 2's column 4 shows the field values of two children nodes (N_{i0} and N_{i1}) and the values of the updated N_i when N_i is divided along the vertical split. For Step 2 (when $n = 1$), $S_{00} = \{(-1, -1, -1), 1, 2, 1\}$ and $S_{01} = \{(0, -1, -1), 1, 2, 1\}$.

Step $3n$ ($n > 0$). When the subdivided cuboid is still bigger than the physical space (PS_j) of the entity E_j that is requesting a GeoCert, the sub-cuboid, where PS_j belongs, gets equally divided along the depth split as shown in Figure 3. Table 2's column 5 illustrates the updated field values for N_{i0} , N_{i1} , and N_i when N_i is divided along the depth split. For Step 3 (when $n = 1$), $S_{000} = \{(-1, -1, -1), 1, 1, 1\}$ and $S_{001} = \{(-1, 0, -1), 1, 1, 1\}$.

Assignment. The three steps repeat until the subdivided cuboid can only fit E_j 's physical space PS_j . In this case, the following values are assigned to the leaf node (N_j) of the k -d tree:

- URL_j is the URL/name of E_j .
- K_j is the public key of E_j .
- $Cert_j$ is E_j 's CA- or self-signed certificate, which contains K_j and PS_j 's spatial coordinates. (Section 5.1 describes various certificate types with different security levels for $Cert_j$.)


- $ptr_{T'}$ is the pointer to the new k -d tree (T'), in case E_j sublets PS_j to other entities. For example, E_j can be the Mall of America which contains over 520 stores. In this case, E_j constructs a separate k -d tree for the stores inside the mall, where N of the Initialization Step becomes N_j and continues the subdivision steps until each individual store acquires (at least) 1 leaf cuboid.

Upon creating a leaf node, the hash values of all intermediate nodes of the k -d tree need to be updated. Hash value construction starts from the leaf nodes (i.e., bottom of the k -d tree) and moves up until the root of the k -d tree (e.g., N) gets a new hash value assigned to H_\emptyset . For example, assuming that $j = 110$ (i.e., after Step 3, E_j occupies N 's left-child cube), here are the hash values that are updated for the k -d tree:

$$H_{111} = H(H_j \| H_{111}) \rightarrow H_1 = H(H_{10} \| H_{11}) \rightarrow H_\emptyset = H(H_0 \| H_1)$$

After assigning a new hash value to H_\emptyset , the GeoPKI database can construct a GeoCert for E_j which contains $URL_j, K_j, Cert_j$, the signed root hash value of the updated Merkle hash tree, and the appropriate intermediate hash values that are needed to verify H_j .

Hierarchical ownership. Alice, who runs a business, may own or rent a subspace of someone else's space, which can also be a subspace of someone else's space, etc. For example, Alice's Verizon store could be located in Mall of America (MOA), in the City of Bloomington, Minnesota, U.S.: Alice's Verizon store \in MOA \in Bloomington \in U.S. This example indicates that a space can be associated with multiple owners, resulting in hierarchical ownership. GeoPKI can represent the hierarchy of certificates using the hierarchical k -d tree structure as described above: Alice's business is a leaf node of MOA's k -d tree, which is also a leaf node of the City of Bloomington's k -d tree, which is a leaf node of Minnesota, etc. As a result, such a parent-child relationship of the hierarchical k -d tree structure supports the representation of hierarchical ownership.

Single entity spanning multiple cubes. It is possible that PS_j spans multiple cubes. For example, the mall has  shape, and the upper right corner is acquired by another entity (with a different key). As a result, the mall spans three sub-cubes. Rather than acquiring three CA-signed certificates, one for each cube, GeoPKI allows E_j to only acquire a single certificate for PS_j such that three sub-cubes contain the same public key K_j and the same hash values H_j . The only different value among these three cubes would be S_j , which represents the boundaries of each cube.

5 GeoCert: GeoPKI Certificates

5.1 Certificate Strength based on Trust Hierarchy

A certificate's security level depends on the signer's security level. For example, we envision that self-signed certificates would be a good starting point for early adopters of GeoPKI. However, for enhanced security and trust, we suggest that the following trust hierarchy is applied to indicate the certificate's security level:

1. **Level 1: CA-EV-signed.** The strongest and most trustworthy certificate is one that is signed with extended validation by a reputable Certificate Authority (CA) (e.g., Verisign). To acquire a Level 1 certificate, not only must an entity owner prove the association between a physical location and the entity (along with its public key), but the CA also needs to physically validate the association, for example by visiting the actual location of the entity. With a successful validation, the CA generates a *CA-EV-signed* certificate.
2. **Level 2: CA-LV-signed.** The second strongest and most trustworthy certificate is one that is signed by a reputable CA with location validation. To acquire a Level 2 certificate, an entity owner must prove the association between a physical location and the entity (along with its public key). For example, a CA may require an entity to (physically) present a legal document, such as a property lease contract or a tax return, which proves the ownership of the space. In case of issuing an online certificate, the CA may validate the location by requesting the entity to provide the security code that is sent by a postal mail. After validating such an association, the CA generates a *CA-LV-signed* certificate.
3. **Level 3: CA-signed.** The third trustworthy certificate is one that is signed by a CA without proof of space ownership. Unlike Level 2 certificates, CAs only require an entity to submit its identity, public key, and its physical address via *email* to acquire a *CA-signed* certificate.
4. **Level 4: self-signed.** A self-signed certificate is created by the entity that owns the corresponding public key.

After acquiring one (or more) of the certificates in this hierarchy and obtaining a public Log Server's signature on the certificate(s), an entity can contact the GeoPKI database to acquire a GeoCert.

5.2 GeoCert Format

In general, when an entity E_i acquires a cuboid with the geographic boundaries of S_i in GeoPKI's database, the GeoPKI database provides a GeoCert for E_i ($GeoCert_i$), which is composed of :

- Name of the entity URL_i , its public key K_i , and S_i ,
- Self-, CA-, CA-LV-, and/or CA-EV-signed certificates $Cert_i$,
- Signed root hash value of the GeoPKI's Merkle hash tree ($\{Root\}_{K_{GeoPKI}^{-1}}$),
and
- Intermediate hash values that are required to verify H_i .

However, there may be cases where $GeoCert_i$ includes additional values, such as proprietor(s)' signature(s). For example, E_j can be a mall and E_i can be a store inside the mall. In this case, $GeoCert_i$ contains E_j 's signature on $Cert_i$.

Location Validation. If E_i acquires Level 2 and/or Level 3 certificate(s), CAs must be able to verify the valid geographic location of E_i , without physically visiting E_i 's business. Postal services can be utilized for location validation as follows: A CA mails out a security code to E_i 's claimed physical location, and E_i must present the security code to the CA for the proper certificate issuance.

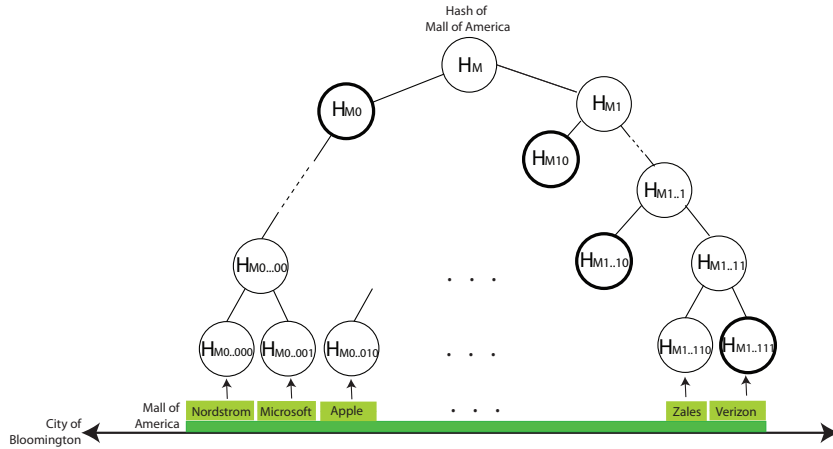


Fig. 4. Merkle hash tree of a mall. To validate the legitimacy of an ATM inside a mall, the certificate database provides a set of minimum hash values (shown in dark circles).

5.3 Certificate Lifecycle

We now describe how the GeoPKI database is modified to accommodate changes.

Insertion. To register a new certificate for the physical space PS_x to the existing k -d tree T , PS_x 's area boundary values (in GPS coordinates) are first translated to the geocentric Cartesian coordinates S_x . Then, S_x is compared with S_0 and follows ptr_x^l if $S_x \in N_{x0}$, or follows ptr_x^r if $S_x \in N_{x1}$. The search continues down the tree until the leaf node containing S_x is reached. If this leaf node's location boundary requires further split to fit S_x , this leaf node continues subdividing until the divided cube is small enough to fit S_x only. Then, that leaf node's values are updated with the new GeoPKI certificate information as described in *Assignment* in Section 4.

Validation. Inserting a certificate into the k -d tree does not guarantee the validity of the item since an attacker can claim a victim's business space and insert its own self-signed certificate. To mitigate such an issue, we suggest applying a *cooling period*, during which newly-posted certificates are suspended for some time period (e.g., 1 week) until they become publicly available. During this cooling period, any valid entity that discovers bogus certificates for its own address space can request their revocation (described below). For example, after x inserts its certificate for PS_x , its certificate can be grayed out (or made invisible) in the database; only when no other entity has provided another certificate does the certificate database (e.g., Google Maps) publicize x 's certificate.

Update. If the values of a leaf node change, as would occur with a new public key, then the node value must be updated. Similar to insertion, updates require searching down the k -d tree until the leaf node to be updated is reached, and the values of the leaf node are updated.

Lookup and verification. Alice wants to verify that a diamond store inside a mall, such as Zales, is indeed legitimate. Assuming that Alice’s mobile phone can acquire the GPS coordinates of her location (using IPS) and access the application to transfer the GPS coordinates to the geocentric Cartesian coordinates, Alice’s browser can search down the k -d tree T to reach a node for the mall. This node will contain ptr_M for the mall’s own k -d tree, where Zales’s physical space is specified. Upon successfully reaching the leaf node of M for the shop, the browser can acquire Zales’s certificate.

To verify the public key of this shop, M provides $GeoCert_{Zales}$, which contains a signed root hash value and the intermediate hash values to verify H_{Zales} . More specifically, to verify H_{Zales} , which is $H_{M1..110}$ in Figure 4, k -d tree M provides the following hash values: $H_{M1..111}$, $H_{M1..10}$, \dots , H_{M10} , H_{M0} , and H_M). Then, Alice verifies if

$$H(H_{M0} \parallel (H_{M10} \parallel (\dots \parallel (H_{M1..10} \parallel (H_{Zales} \parallel H_{M1..111}) \parallel H_{M1}) \dots))) = H_M \quad (1)$$

If Alice knows the mall’s public key, she can successfully authenticate that Zales’s entry in k -d tree M is legitimate should the verification succeed. If she does not know the mall’s public key in advance, she can verify with additional hash values from T along with the signed root hash value, since her browser already knows the public key of the GeoPKI database.

As an alternative, Zales can regularly contact the GeoPKI database (e.g., once a day) to acquire the minimal intermediate hash values along with the signed root hash value, and provide them along with its certificate while establishing a SSL/TLS session. In this case, the browser can validate $Cert_{Zales}$ without directly contacting the GeoPKI database.

Revocation. In some situations, certificates will need to be revoked, for example when PS_x is incorrectly acquired by an entity (i.e., attacker) besides x . In that case, x can reacquire PS_x by obtaining a stronger certificate from the trust hierarchy as mentioned in Section 5.1. For example, if the current certificate for PS_x is self-signed (Level 4), x can revoke it by obtaining a CA-signed (Level 3), CA-LV-signed (Level 2), or CA-EV-signed certificate (Level 1). If the current certificate for PS_x is a CA-LV-signed certificate (Level 2), this indicates that either the attacker successfully crafted an illegal document that binds PS_x to the attacker, or the CA was compromised. In this case, x can revoke the malicious certificate by acquiring a Level 1 certificate. If the current certificate is Level 1, this indicates that the issuing CA must have made a validation error, and may face legal actions. In this case, x can revoke the incorrect certificate by acquiring *multiple* Level 1 certificates from *reputable* CAs.

6 Security Analysis

In this section, we describe how GeoPKI achieves the desired properties as mentioned in Section 2.1.

Correct location-to-certificate association. GeoPKI’s goal is that only the genuine owner of the space can associate her certificate with that space. Consider

a scenario in which an attacker, Mallory, claims that some geographic location L_A corresponds to its own fraudulent resource R_M , when indeed L_A is the address for victim Alice’s resource R_A . For example, Mallory can attack Alice’s authentic Bank of the West by creating a mimicking website with the URL `https://www.bankofthevest.com` (instead of `https://www.bankofthewest.com` [4]) and claim that its location is a real Bank of the West branch. Mallory was able to obtain a certificate $GeoCert_M$ that binds her key K_M and the geographic location of the real Bank of the West branch L_A to Mallory’s resource R_M . When Alice realizes that her address has been acquired by someone else, she can take one of the following actions, depending on the type of certificate that Mallory obtained:

- If $GeoCert_M$ contains a *self-signed* certificate (Level 4), Alice can re-claim L_A by getting at least a Level 3 certificate (i.e., CA-signed, CA-LV-signed, or CA-EV-signed). Since Alice provides a stronger certificate, GeoPKI revokes $GeoCert_M$ and accepts $GeoCert_A$, validating that L_A is R_A ’s physical location.
- If $GeoCert_M$ contains an *CA-signed* certificate, this indicates that Mallory emailed the CA requesting that L_A be associated with her key K_M . In such a case, Alice can acquire a CA-LV-signed or CA-EV-signed certificate $GeoCert_A$, and GeoPKI will accept $GeoCert_A$ over $GeoCert_M$ based on the level of the certificate strength.
- If $GeoCert_M$ contains a *CA-LV-signed* certificate, this indicates that Mallory successfully demonstrated some legal document to the CA. In such a case, either Mallory was able to obtain an illegally crafted document, or the CA was compromised. To resolve the conflict, Alice can acquire a *CA-EV-signed* certificate, which will revoke $GeoCert_M$.
- If $GeoCert_M$ contains a *CA-EV-signed* certificate, Alice can acquire CA-EV-signed certificates from *multiple, uncompromised* CAs. Since multiple certificates indicate that L_A is for Alice’s resource, GeoPKI will revoke $GeoCert_M$ and accept Alice’s certificates. However, the *CA-EV-signed* certificate requires a significant amount of validation and documentation, and the adversary most likely will leave behind significant evidence which Alice can use to take legal action. Moreover, the issuing CA must have made a validation error, and may face legal action.

As a result, GeoPKI eventually provides the correct certificate for the location L_A that users can trust.

Response integrity. GeoPKI provides globally consistent query replies by using the Merkle hash tree. Whenever Bob queries the GeoPKI database, he receives not only the requested certificate, but also the root of the hash tree and the intermediate hash values for an integrity check. As a result, two different queries for the same set of coordinates will always yield the same correct response, assuming that the database has not been updated between the queries.

Available querying of 3D space. GeoPKI utilizes the k -d trees for the efficient query and retrieval of digital certificates associated with 3D spaces. By utilizing

current map operators or large-scale Internet businesses to host the GeoPKI service, GeoPKI can ensure availability while mitigating DDoS attacks.

Efficient validation. By combining k -d trees with Merkle hash trees, GeoPKI enables users to efficiently validate GeoCerts. Furthermore, the spatial ordering within k -d trees enables the efficient validation of non-existing entries as follows: the GeoPKI database only needs to search down a branch where the queried coordinate resides, and when it does not correspond to either of the two leaf nodes, the database confirms that the query is for a non-existing entry.

7 GeoPKI Overhead Analysis

In this section we analyze the overhead of GeoPKI. Because of the availability of information, we consider the businesses or physical entities within the U.S. However, the analysis can be readily extended to other countries given the required data. We evaluate GeoPKI overhead on three base parameters: size of the GeoPKI tree structure, certificate size, and verification overhead.

Size of the GeoPKI structure. To evaluate the size of GeoPKI’s 3D data structure to index U.S. businesses, we use the estimated number of physical resources that are within the U.S. As of March 2012, 20 million businesses have been registered in the U.S.⁵

To compute the approximate depth of the tree, we create a cube around the whole globe with the length of the side equal to the diameter of the earth (12756km). We use the approach discussed in Section ?? to generate the 3D data structure to store all the entities’ certificates. We consider a representative business size as $12m \times 12m \times 3m$. Since a cube with width of approximately $12m$ is given by $\log_2(12756000/12) = 20$, the approximate tree depth given the three dimensions is $3 \cdot 20 = 60$.

With depth $d = 60$, a binary tree would contain 2^{60} leaf nodes. However, most nodes would be empty. For computing an upper bound on the number of non-empty hash tree nodes, we consider 20 million randomly distributed businesses, and with $d = 60$, we use the formula deduced in prior work [10] to calculate the expected number of nodes (N) in the k -d tree:

$$N = \sum_{i=1}^d 2^i \pi(2^{d-i}, b, 2^d) (1 - \pi(2^d - i, b, 2^d - 2^{d-i}))$$

where b is the number of leaf nodes, d is the depth of the tree, and

$$\pi(a, r, n) = \prod_{i=0}^{r-1} \frac{n - a - i}{n - i}$$

where $\pi(a, r, n)$ is the probability that a number of leaves are empty after randomly choosing r leaves among n leaves ($\pi(a, r, n) = 0$ if $n - r < a$).

⁵ http://www.uscompanydatabase.com/database_us.aspx

Based on these estimates, the expected number of nodes the k -d tree will have is approximately $557 \cdot 10^6$.

Approximate size of the GeoPKI database. Each leaf node in the GeoPKI database contains a name, a public key, coordinates of the cube, a certificate, and a hash of the certificate.

The minimum length of an RSA public key recommended for use today is 2048 bits. Thus, the public key and a signature amount to $2 \cdot 2048 = 4096$ bits.

If we represent each dimension of the coordinates of the cube and distance measurements from all the three surfaces of the cube by 30 bits each (to achieve about .1 meter of resolution), then it requires 90 bits to store the coordinates of the cuboid and 30 bits for the cube dimensions.

Assuming 256 bits for a hash value and 128 bytes for the name, time stamps, etc., the approximate size of a leaf node is approximately $4096/8 + 120/8 + 256/8 + 128 \approx .7$ Kbytes. Consequently, the 20 million businesses' nodes will only require about 14 Gbytes to store, without considering additional opportunities for compression.

Verification overhead. We measure the number of hashes as well as the number of digital signatures that have to be computed by the client to verify a GeoCert. Since the GeoPKI's k -d tree has approximately 60 levels, the total verification overhead requires 60 hash function computations and one digital signature verification. Conservatively, considering a hash function computation to require $1\mu s$ and a 2048-bit RSA digital signature verification to require about $1ms$, the total verification time is still close to $2ms$, as the hash function computation is dominated by the verification of the CA signature and the GeoPKI signature on the Merkle hash tree root.

8 GeoPKI Applications

In this section, we describe how GeoPKI can assist people to verify an online resource based on trust in a physical space. More specifically, a user trusts that the physical space where she is currently standing is indeed the correct physical space of an institution. Consider the case of an airport, a university, or a bank. It would be exceedingly challenging for an adversary to construct a fake airport, for example. In cases where the user can trust the physical space, she can also trust that the owner of the space has registered its space with GeoPKI. Consequently, trusting the space and the corresponding GeoCert enables the user to establish trust in online resources. We demonstrate this with two applications, for which trust establishment without GeoPKI would be challenging.

Evil twin attack. Fake wireless access points (AP) mounted by hackers have been found at airports in Los Angeles, Atlanta, New York, and Chicago.⁶ An attacker can set up a fake AP with the same Service Set Identifier (SSID) as a legitimate AP at some nearby free hotspot (e.g., airport, cafe, hotel, library, etc.).

⁶ <http://www.metrowestdailynews.com/x282695013>

The attacker can position the evil-twin AP to be physically close to users, thus trumping the legitimate AP with a stronger signal. As a result, users' machines likely associate with the evil-twin AP.

GeoPKI can help prevent such attacks. When 802.11x is used with EAP-TLS, for example, the authentication server's public key can be authenticated using GeoPKI, where either the authentication server itself has a GeoCert that is associated with the space, or a hierarchy of certificates is associated with the space, as described earlier in the paper. In both cases, the AP needs to pass all certificate validation information to the client, including hash tree values, the signed root value, and potentially other GeoCerts along with their required verification information.

In other authentication environments, the user's browser is redirected to a web page hosted by an authentication or paywall server, where the user needs to enter authentication credentials. In this case, an adversary controlling an evil-twin AP can present a fraudulent site to steal the user's credit card or login credentials. The paywall can again obtain a GeoCert, which the client browser can then validate. Another option is for the larger space (such as the university, corporation, or airport) to obtain a GeoCert and digitally sign the certificates of a paywall or authentication server. Again in this case, the server needs to pass all required information to the client to validate the certificate, and in some cases, a hierarchy of GeoCerts and hash tree nodes may also be required for validation.

Impersonation attack. An attacker can impersonate a website to trick users into entering their personal information. For example, a new student who attempts to establish a SSL session with his university authentication server on campus can avoid logging into the impersonated webpage with GeoPKI as follows: the authentication server provides the university's GeoCert that signs the server's certificate. As a result, the browser accesses the GeoPKI database, acquires intermediate hash values, verifies the legitimacy of the university's GeoCert, and validates the server's certificate using the university's public key.

9 Related Work

We are not aware of other work that proposes a geographic PKI, which uniquely associates certificates to 3D spaces. We will thus discuss the most closely related work of which we are aware.

Pala et al. propose to embed coordinates in a temporary proxy certificate issued by a user's mobile device [9], which can be used by a server for controlling user access. In GeoPKI, a spatial database is used to associate a public key with a space, which is a different model achieving different security properties.

To address the issuance of bogus X.509 certificates, Sovereign Keys [5] and Certificate Transparency [1] have been developed to make all registered certificates publicly visible, thus creating accountability for CAs. These proposals, however, do not advocate the use of spatial certificate lookup.

In integrity regions [13], an access point continuously sends out the local certificate on a specially encoded jamming-resilient channel, whereby devices can

learn the local key. GeoPKI is a more general approach, providing more flexibility without requiring a special communication primitive, and also enabling spatial hierarchies and providing accurate delineation of spaces. Finally, GeoPKI could also be used remotely, assuming that the user has sufficient evidence about the space, for example by a past visit to that space.

10 Conclusion and Future Work

Our core goal in this paper is to provide automated verifiability of online resources connected to physical spaces. Our key insight is that trusted physical space can provide accountability for online resources, assuming the presence of the mechanisms described in this paper.

GeoPKI leverages a global certificate integrity service, in line with proposed systems such as Sovereign Keys [5] and CA Transparency [1]. In addition to the append-only and global visibility of all actions of these systems, GeoPKI also requires efficient coordinate-based certificate lookup services.

Once the utility of these ideas are established, much future work is needed to transition them into practice. First, details on how to encode GeoPKI information in a certificate and applicability with existing systems needs to be studied. An important area of further study is a better space verification mechanism, which prevents registration of a space that nobody owns – quite possibly hierarchical space allocation would help in this case, such that space registrations need to be approved by city, which itself needs to be approved by state, country, etc. Once deployed, additional applications could benefit from GeoPKI, and possibly new unexpected uses will emerge.

We anticipate that this paper will help researchers perceive how a GeoPKI-like infrastructure can enhance usability for establishing trust in online resources associated with physical spaces.

11 Acknowledgments

We would like to thank Payas Gupta for his help in finding applications for GeoPKI and for his help with the evaluation. We also thank the anonymous reviewers for their helpful suggestions.

This research was supported by CyLab at Carnegie Mellon under grants DAAD19-02-1-0389, and W911NF-09-1-0273 from the Army Research Office, by support from NSF under awards CCF-0424422 and CNS-1040801. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ARO, CMU, NSF or the U.S. Government or any of its agencies.

References

1. Certificate authority transparency and auditability. <http://www.certificate-transparency.org>, 2011.

2. R. Bayer and V. Markl. The UB-tree: Performance of multidimensional range queries. Technical report, Institut für Informatik, TU München, 1998.
3. M. M. Correia and M. Tok. DNS-based Authentication of Named Entities (DANE). Technical report, Universidade do Porto, 2011–2012.
4. R. Dhamija, J. D. Tygar, and M. Hearst. Why phishing works. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI)*, 2006.
5. P. Eckersley. Sovereign Key Cryptography for Internet Domains. <https://www.eff.org/sovereign-keys>.
6. A. Guttman. R-trees: A Dynamic Index Structure for Spatial Searching. In *International Conference on Management of Data*, 1984.
7. J. Han, E. Owusu, L. T. Nguyen, A. Perrig, and J. Zhang. ACComplix: Location inference using accelerometers on smartphones. In *Proceedings of International Conference on Communication Systems and Networks (COMSNETS)*, Jan. 2012.
8. B. Hill. CA T&A proofs as cert extensions. <http://www.ietf.org/mail-archive/web/pkix/current/msg30146.html>, 2011.
9. M. Pala, S. Sinclair, and S. W. Smith. PorKI: Portable PKI credentials via proxy certificates. In *Proceedings of European Workshop on Public Key Services, Applications and Infrastructures (EuroPKI)*, Sept. 2010.
10. A. Perrig. The biba one-time signature and broadcast authentication protocol. In *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, Nov. 2001.
11. R. Poovendran, C. Wang, and S. Roy. *Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks*. Springer Verlag, 2007.
12. H. Samet. Octree approximation and compression methods. In *In Proc. of the 1st Intl. Symp. on 3D Data Processing Visualization and Transmission*, 2002.
13. S. Čapkun, M. Čagalj, G. Karame, and N. O. Tippenhaur. Integrity regions: Authentication through presence in wireless networks. In *IEEE Transactions on Mobile Computing*, 2010.