PKI: Ten Years Later

Carlisle Adams
University of Ottawa
Ottawa, Canada
cadams@site.uottawa.ca

Mike Just
Treasury Board of Canada, Secretariat
Ottawa, Canada
just.mike@tbs-sct.gc.ca

Abstract

In this paper, we examine the history and evolution of so-called Public Key Infrastructure (PKI). We compare the original definition of PKI with a broader and more flexible definition that better reflects the variety of implementation philosophies available today. This current definition shows how the understanding of this technology has matured (although its essential characteristics have remained unchanged) and is derived, at least in part, from an evaluation and comparison of several quite different forms of PKI as well as a consideration of PKI criticisms over the years. The original definition of PKI may be dead or dying, but PKI technology continues to thrive as an extremely useful (and, in some cases, necessary) authentication solution.

1 Introduction

The technology known as "PKI" has been simultaneously maligned and praised. PKI praise can come in two flavours. The first results from a dislike for other security technologies. For example, a dislike for password-based authentication may result in a stronger preference for PKI solutions. Secondly, public-key technology offers some important benefits that are not similarly offered by other technologies, such as digital signatures. However, PKI is equally, if not more often, criticized. Difficulties around issues such as application integration, interoperability, and trust often lead critics to predict the end of PKI. While these shortcomings are very real, other issues have often been raised that either are orthogonal to PKI, or similarly impact non-public-key-based technology. In this paper, we try to identify such issues so that their true impact on PKI can be understood.

We attempt to provide some clarity to the status of PKI by discussing how it is understood today, compared with how it was initially defined. We examine the ways in which this updated definition encompasses ten years of PKI evolution, while remaining true to the original, essential characteristics of this technology. In order to do this, we compare several different PKI implementation models and see what lessons can be learned from some previous criticisms of PKI.

In Section 2, review and highlight several concepts related to public key technology and introduce six components that will contribute to our definition of a PKI. Section 3 reviews four PKI examples relative to these PKI components. In Section 4, we review and critique some well-known criticisms of PKI. Section 5 builds upon the previous PKI components, examples and critique to provide a modern definition for a PKI, while Section 6 recognizes those areas that still require development in order to support more successful PKI deployment.

2 Public Key Technology

In this section, we briefly examine some of the cryptographic properties of a PKI, and proceed to discuss how these properties may be used in practice.

2.1 Public Key Cryptography

Public key (a.k.a. "two key" or "asymmetric") cryptography was invented by Diffie and Hellman in 1976 [DH76]. Unlike secret key (a.k.a. "symmetric") cryptography, in which the same key K is shared between parties A and B, pairs of corresponding private and public keys for each user allow the unique realization of some operations. Specifically, let the respective private and public keys, $priv_A$ and pub_A, belong to party A. By operating on data with $priv_A$, A can *digitally sign* data that is verifiable by party B (or any other party) operating on the signed data with pub_A. Equivalently, party B (or any other party) can encrypt data using pub_A, where the encrypted data can only be decrypted with $priv_A$.

The true power of public key cryptography lies in the possession of a private key, *uniquely*, by each party. The "demonstration of knowledge" of the private key by operating on data with said key, provides a powerful tool that distinguishes asymmetric cryptography from its secret key counterpart.

2.2 Public Key Cryptography in Practice

Most secure transfers of data involve an exchange between identifiable parties. On its own, public key cryptography only supports asymmetric, mathematical operations on data; it does not by itself provide a connection to applications or environments such as e-commerce, e-mail, or the Web.

To provide such a connection, several additional pieces are necessary. These additional pieces form the definition of a PKI – an "infrastructure" that makes public key technology available to the applications and environments that wish to use it. In subsection 2.3 below, we identify several components that are integral to an infrastructure for supporting public key cryptography (these components are used to capture the evolving definition of a PKI in Section 5).

Identification is a property that is particularly critical to a PKI, and (at least historically) a strong differentiator between some different PKIs. Specifically, public key cryptography is made considerably more useful if the public key is *bound* to a so-called *identifier*. As distinguished below, this *identifier* may or may not provide direct information regarding an actual *identity*. This identifier may be an

- *Anonym*: "No name"; a single-use identifier providing no information as to the identity of the key owner.
- *Pseudonym*: "False name"; providing a "pretend" identity that can be used over a period of time to protect the real identity of the key owner.
- *Veronym*: "True name"; providing the identity of the key owner.

The identifier is typically meaningful only within a specific context or environment; it may or may not need to be globally unique, depending upon the applications for which it will be used.

Parties that use public key cryptography for encrypting data, or for verifying digitally signed data, will rely on the binding of an identifier to the public key (whether this binding is preserved in a certificate or database, for example) in order to associate that key with an entity with which they may have past or future interactions. This also supports repeated use of the same public key, whether or not the key is directly associated with an actual identity.

2.3 Public Key Infrastructure

Approximately ten years ago, the 1993 version of the ISO/IEC CCITT/ITU-T International Standard X.509 began to be disseminated, recognized, and implemented in small-scale environments. Late 1993 and early 1994 was effectively the beginning of PKI (although that acronym had yet to be coined) because that version of the X.509 standard – more than the 1988 version – fleshed out some of the important details of certificates, certification authorities, and related concepts.[1]

In those early days, a "PKI" was defined fairly rigidly, although with hindsight we can identify six major components to the definition that are still critical today, three to do with the *validity* of bindings (authority[2] functions), and three to do with the *use* of bindings (client functions). With respect to the validity of the binding between a public key and an identifier, what is needed is

1. an *authority* whose responsibility it is to create and destroy these bindings, as required, or aid in related authoritative actions,
2. an *issuance process* for expressing these bindings in a way that can be understood by other parties (i.e., in an agreed syntax) and for making this information available to parties that wish to know it, and
3. a *termination process* for breaking bindings when necessary and making this information available to parties that need to know it.

With respect to the use of such bindings, what is needed is

4. an *anchor management process* for augmenting or diminishing the set of authority public keys that will serve as roots or trust anchors for the client[3],
5. a *private key management process* for ensuring that a client private key can be used for its desired purpose (this can include key pair generation and update, registering and binding an identifier to the corresponding public key, proper protection of the private key while it is valid, and backup & recovery of the private key in case of loss), and
6. a *binding validation process* for determining when the client should trust that a given public key (retrieved or acquired from some external entity) is authentically associated with a given identifier.

In Section 5, we develop a more detailed definition of a PKI, based on these components, that reflects the decade-long evolution of a PKI. The degree to which these components are implemented is commonly a risk management decision. The PKI examples in the next section can differ based upon such choices.

In the original Diffie and Hellman model [DH76], public keys would be retrieved from a secured repository. The security of this repository served to bind the public key to other attributes of the key owner. In support of offline binding production and distribution, Kohnfelder introduced the notion of a certificate [Kohn78], whereby a public key and an identifier (e.g., a name) were placed in a data structure signed by a Certification Authority (CA) and made available in an unsecured repository.

Various PKI systems can be distinguished and compared based upon the above six PKI characteristics. In Section 3, we categorize several examples of PKI systems with respect to these characteristics. It is particularly interesting to note that one of these examples makes use of

[1] Though the first version of Pretty Good Privacy (PGP) appeared in 1991, it wasn't until later that features consistent with a PKI were provided (see Section 3.2).
[2] An "authority" may be any specially designated entity, though an end-entity client may also act authoritatively.
[3] These roots form the axiomatic elements of direct trust for a client. Trust in other public keys is derived from these roots.

Diffie and Hellman's original concept of a secured repository (AADS; see Section 3.3), while the remaining examples use "certificates" with varying syntax.

3 PKI Examples

Over the past 10-15 years, there have been several examples of public-key technology solutions. Below, we focus on those solutions that offer the best contrasts within the PKI components identified above. The representative example names (X.509, PGP, AADS/X9.59, SPKI) are quite overloaded with varying descriptions, as they may refer to several standards or even several varying implementations of those standards. In our review below, we have tried to focus on those features that are independent of specific product implementations yet representative of distinctive features for each PKI example.

On a related note, it is also recognized that over such a time period, the solutions have each grown and matured greatly. Though we attempt to identify this growth, our main purpose is to identify the philosophical differences between the solutions, so that not all features of each PKI solution may be acknowledged.

3.1 X.509

The X.509 standard [X509-00] and its Internet profile [RFC3280] do well to represent the PKI components identified in the previous section. In most cases, implementations differ based upon the rigour with which they implement the suite of appropriate standards (e.g., see the exhaustive list of Internet standards for X.509 [PKIX-WG]). Below, we examine relevant components of an X.509 PKI.

- *Authority*. A Certification Authority (CA) issues X.509 certificates that bind the public key to a Distinguished Name (DN) identifier (although other name forms are also allowed), in addition to other information contained in the certificate. An Attribute Authority (AA) is similarly defined, and binds more general attributes to one another in an attribute certificate, and provides an optional link to a corresponding public key certificate.
- *Issuance process*. Typically, though not necessarily, certificate issuance involves a Registration Authority (RA) responsible for registering the user (including their identification, if performed). Traditionally, the DN and alternative name forms would be veronymous. However, neither the ASN.1 syntax nor the standard restricts this so that anonymous or pseudonymous name forms are fully supported.[4] Once issued by a CA, certificates require no further integrity protection and may be distributed amongst parties or made available in a repository. This repository is commonly an X.500 or LDAP directory, though various other repositories are typically supported now, including Web servers. Retrieval is predicated upon knowing the identifier of the certificate holder (typically the DN, although an email address contained in the certificate can also be used).
- *Termination process*. Certificates contain an expiry date that acts as a default termination date for the certificate. Certificates may also be "revoked" prior to their expiry, in which case the revocation information must be disseminated. There are traditionally two ways for this to be achieved: (i) by posting information regarding the revocation in a Certificate Revocation List (CRL), or (ii) by making the revocation information available through an Online Certificate Status Protocol (OCSP) responder [RFC2560]. The location of revocation information is typically included within the certificate that is being verified (e.g., as a URL for the CRL).

[4] See [Just03] for an example of an X.509 PKI with a pseudonymous certificate identifier.

- *Anchor management*. The standards describe protocols for retrieving trust anchors as part of the registration (and key update) process(es). Depending upon the implementation, a client may be able to trust a number of trust anchors simultaneously (as part of a certificate trust list). Traditionally, there are two forms of trust for X.509 certificates. In the first, the application software holds the public key of a root CA. All certificates that may be trusted by this client are issued, either directly or indirectly (e.g., within a hierarchy), by this CA. In the second form of trust, a party holds the public key of the CA that issued their own certificate.
- *Private key management*. The standards support protocols for renewing key material prior to the expiry of the corresponding certificate. They also support the backup of decryption keys (and, more importantly, their recovery). The standards also allow a separate lifetime for the private key itself, primarily in support of preventing the creation of signatures too close to the time of certificate expiry, though this lifetime value is also helpful to trigger a timely key update in support of uninterrupted client operation.
- *Binding validation*. Clients use their trust anchors and possibly chain building to establish certificate trust. Trust in certificates issued by other CAs may be obtained through cross-certification between the CAs, or possibly by the party importing or retrieving the certificates of the other CAs as necessary. There are numerous variations to these two simple trust models. Traditionally, clients would be required to retrieve and validate the entire chain of certificates, though recent standards have been developed to support the off-loading of some of these operations [RFC3379].

In the often-cited "browser-based PKI", it is important to recognize that certificates are issued to servers, while clients use those certificates to authenticate a server and establish a confidential communication channel.[5] Clients retrieve the server's certificate a part of the SSL protocol [Resc01]. The termination process supports expiry, though automated revocation support is minimal and inconsistent. Client anchor management is essentially static, and established by the version of Web browser being used, though users can manually update their trust anchors, if they so desire.

3.2 PGP

Though the first version of Pretty Good Privacy (PGP) appeared in 1991, its primary focus was in the support of public key cryptographic operations, not the provision of a PKI. Later versions supported notions such as key servers (see, for example, [PGPks]) thereby supporting an "infrastructure" for the management of public key information. In more recent times, PGP has highlighted its ability to also support features similar to X.509 [PGP99]. Traditionally, however, PGP has been distinguished by its distributed approach to key management. PGP certificates are formed of one or more "certifications", which bind keys to user information with a digital signature. There is great flexibility in the key and user information that can be conveyed in a single certificate.

- *Authority*. Traditionally, a PGP PKI avoided the need for any form of authority. As discussed below, trust relies upon a "web" of users. However, the syntax does not preclude "authoritative users" that might act in a similar fashion to a Certification Authority (CA). For many communities, the lack of an authority greatly eases initial deployment as small communities need only rely upon the bilateral sharing of certificates among users who wish to communicate securely.
- *Issuance process*. Certificates are created and populated by the key owner. They can be distributed via email, Web pages, key servers, and/or other means. The identifier is typically an email address, though there is nothing to preclude other identifiers.

[5] Though client authentication with certificates is supported by the standards, it is not often implemented.

- *Termination process*. Certificates can contain an expiry date (though no expiry date need be set) that acts as a default termination date for the certificate. The distribution of certifications is not managed, so that manual revocation would have to be performed, though revocation information can be made available in a similar fashion to publication of the certificate.
- *Anchor management*. Trust must be anchored in the *direct trust* of other users' certificates. Various mechanisms can be used to establish this direct trust base. For example, two users can exchange key information by email, but verify the authenticity of the exchange by exchanging message digests for the key information by phone. Such key information serves the role of PGP "roots" or "trust anchors."
- *Private key management*. Lacking a 3rd-party authority, private key management is the responsibility of the key owner. For example, key owners can backup private keys on a separate disk. It would be a simple task for software to remind users of the need to update their key material. Similar ease would allow an update of key material, since no communication with an authority is required, though updated key distribution would still be performed by the key owner.
- *Binding validation*. Based upon some initial, direct trust, there are a couple of options for indirectly extending trust to others. With hierarchical trust ("chain of trust"), you trust others that are trusted by people you trust. With cumulative trust ("web of trust"), you trust others only when they are trusted by a number of people you trust.

3.3 AADS/ ANSI X9.59

ANSI X9.59[6] is a financial industry standard for secure financial payments, based on AADS (Account Authority Digital Signature) [AADS99]. For our purposes, we use it as an example of a non-certificate-based public key solution.
A public key is stored and managed as part of a key owner's financial account, along with other identity or authorization information. So, the issuer is an authority that already has a relationship with the user, and thus should be able to easily identify said user.

- *Authority*. The authority, or maintainer of the binding, is the user account manager. The public key serves, quite simply, as an additional attribute to a user's financial account record.
- *Issuance process*. Users may be identified by their account manager, based upon shared financial secrets, or other account information. The public key is retained by the manager. For AADS, the public key need only be accessed by an authentic request and response with the account manager to retrieve the public key for signature verification. For other applications, this could be easily adapted to allow for public-key encryption. Note that by not relying on a certificate, the AADS solution is more similar to the ideas of Diffie and Hellman than those of Khonfelder (see Section 2.3), except that with AADS the repository of public keys is built, held, and used by the relying party alone, whereas with the original Diffie-Hellman proposal this repository was to be created for the use of the whole world.
- *Termination process*. There are no "certificates" and use of any public key is always initiated by a request to the account manager. Expiry or revocation occurs by removing or replacing the public key for a user's account. Therefore, a type of immediate, online validation is supported as part of the public key retrieval.
- *Anchor management*. Relying parties require a method by which they can trust the account manager. A method similar to server-authenticated SSL would suffice. An initial trust anchor could be retrieved when the user's key pair is generated, for example.
- *Private key management*. Updates to private key material may be managed and initiated by the account manager. In the case of AADS, since only digital signature operations are

[6] See http://www.ansi.org/

performed, there is no need to backup private key material. If encryption operations were supported with the user public keys, there may be a need for key backup support.

- *Binding validation*. Trust is isolated to the domain of a single account manager. As mentioned above, online trust validation is implicit with the retrieval of the public key.

Similar to SSL representing a client-server PKI implementation using X.509 certificates, SSH [SSH03] is representative of a certificate-less client-server PKI implementation. As part of the SSH transport protocol, clients establish trust in the server based on their trust in the server host key. Though there are options for how this trust might be established, including the client maintaining a store of trusted server host keys, or certification by a CA, SSH presents an interesting compromise whereby "the server name - host key association is not checked when connecting to the host for the first time" [SSH03]. Though introducing the potential for a middle-person attack, this novel variation offers great improvement for SSH bootstrapping. Once a server-authenticated, confidential channel is established, the client may authenticate to the server; this is often performed using password authentication.

3.4 SPKI

Simple Public Key Infrastructure [RFC2692, RFC2693] was developed in response to several criticisms of X.509. The major philosophical objection to X.509 surrounds its relation to X.500 naming. SPKI, more correctly an authorization infrastructure, relies upon the uniqueness of the combination of a pseudonym and a public key.

- *Authority*. SPKI focuses on the issuance of authorization information within certificates. Thus, an SPKI authority might be referred to as an authorization authority. With regard to an issuance authority, SPKI theory indicates that certificates may be generated "by any keyholder empowered to grant or delegate the authorization in questions." [RFC2692]
- *Issuance process*. In support of the authorization information, the SPKI certificate syntax uses an S-Expression, which is a LISP-like expression using parentheses. *Authorization certificates* bind authorization information to a key, while *attribute certificates* bind an authorization to a name. The use of names differs from the initial use of global names for an X.500 directory, as part of X.509, and was inspired by the use of SDSI's [SDSI96] local names. Combined with the (globally unique) hash of a public key, such a name can become globally unique.
- *Termination process*. Certificate lifetime is parameterized by a validity period so that certificates can be set to expire. Several options for certificate revocation are supported, including Certificate Revocation Lists (though they are not the preferred choice). Options for online revocation status checking are also supported. Preference is given to "positive statements" on certificate validity, so that a protocol returning an indication that a certificate is currently valid is favourable to one that returns a notice of invalidity.
- *Anchor management*. Details regarding anchor management are left open for developers so that, for example, protocols similar to those previously described could be used. For validation of authorization information, however, the relying party maintains an access control list (ACL).
- *Private key management*. The management of private keys depends upon the certificate issuer regarding issues of key backup; however, when used only for authorization purposes, the need for key backup is limited. Support for key updates does not appear to be standardized, so would be dependent upon the specifics of a particular implementation.
- *Binding validation*. The main difference with traditional X.509 is the use of the pseudonym for SPKI. Processing decisions for SPKI certificates are defined through "tuple reduction." [RFC2693].

3.5 Summary of Examples

The following table compares the solutions based upon the *validity of bindings* (see Section 2.3).

PKI Solution	Authority	Issuance Process	Termination Process
X.509	Certification Authority (CA) Attribute Authority (AA). The CA is the owner / definer of the namespace for the identifier.	ASN.1 syntax Traditionally available from X.500 or LDAP directories.	Certificate contains an expiry date. Revocations posted through revocation lists, or made available through an OCSP responder.
PGP	No external authority required. Key pair and certificate are self-generated. The user (end entity) is the owner / definer of the namespace for his/her identifier.	Made available to others by key owner (e.g. via Web page, email signature, or key server).	Certificates can expire. Termination performed by key owner. Dissemination of termination notice by key owner as with certificate publication.
AADS/ X9.59	User account manager. The relying party (the account manager) is the owner / definer of the namespace for the identifier (the acc't. #).	Public keys available in secured repository from account manager.	Public keys removed from repository when binding is terminated.
SPKI	No explicit authority is required as the authorization granter or delegator may issue certificates. The relying party is the owner / definer of the namespace for the identifier.	Issue authorizations based on pseudonymous identifier or SDSI names.	Similar to X.509, though "positive statements" through online validation are preferred.

The following table compares the solutions based upon the *use of bindings* (see Section 2.3).

PKI Solution	Anchor Management Process	Private Key Management Process	Binding Validation Process
X.509	Single or multiple roots. Standardized protocols support changes.	Standardized protocols support update, backup and recovery.	Client search of Directory for cross certificates. Delegated path discovery and validation services are being standardized.
PGP	Direct trust of other user certificates. Trust anchor is user's own key(s).	Manual update, backup, and recovery performed by user.	Chain of trust, or web of trust.
AADS/ X9.59	Trust in account manager is required.	Depends upon expiry policy. Backup and recovery not a concern when only digital signatures are used.	Only direct validation through trusted key retrieval.
SPKI	Open to developer. Trust anchor is the ACL at the relying party.	Open to developer. Fewer backup and recovery requirements when certificates used only for authentication or authorization.	Tuple reduction.

These fundamental PKI examples contribute to a greater understanding of the different options available for PKIs within what is mistakenly viewed as a "rigid" structure. In the following section, we further examine criticism of PKI, identifying those issues that are specific to the components of this infrastructure. In Section 5, we use the examples and the lessons learned from the criticism to capture the evolutionary definition of PKI.

4 Criticism of PKI

Over the past ten years, PKI has been the subject of criticism from various quarters. Some of this criticism has been beneficial, driving the evolution of this technology and leading to a deeper understanding and broader application of PKI. However, much of the criticism has been misdirected, aimed at PKI when the actual problem or challenge is either independent of this technology, or common to many technologies.

In this section we review some popular PKI criticisms to see which can fairly be applied to the current state of the art. While it is certainly not the only collection of criticisms, arguably the best known collection can be found in the paper by Ellison and Schneier [ElSc00]. We therefore use that paper as the basis for our examination of PKI, circa 2004.

"Ten Risks of PKI: What You're not Being Told about Public Key Infrastructure" aims to explore some basic questions around PKI ("What good are certificates anyway? Are they secure? For what?") so that potential users of this technology can be made aware of some of the risks involved with its use. This is unquestionably a worthy goal and will serve the industry well, but only if the highlighted risks are accurate and fair (i.e., legitimate criticisms of PKI technology). Let us examine some of the risks discussed in that paper.

Risk #1 ("Who do we trust, and for what?") warns that the certificates issued by a CA should not be automatically trusted for a plethora of application-level purposes, such as making a micropayment or signing a million-dollar purchase order ("Who gave the CA the authority to grant such authorizations? Who made it trusted?"). Unfortunately, this criticism highlights only the misuse of PKI by some implementers; it is not a valid criticism of PKI itself. PKI is an authentication technology; authorization is an independent matter and may or may not be linked to authentication in any way. The authors suggest that "Many CAs sidestep the question of having no authority to delegate authorizations by issuing ID certificates." However, the issuance of ID certificates is the primary function of a CA (not a "sidestep"). In some environments, it may be natural for information other than an identifier to be linked to the public key by the CA; for such situations a variety of authorities for such information may be used in conjunction with the CA (these are discussed as part of the evolving PKI definition in Section 5). On a related note, while certificate policies appear to contain some notion of authorization, they are more properly viewed as statements regarding the "quality" of the key. For example, given the specific process used to generate the key pair, the rigour with which identification of the key holder was done, the care with which the private key will be safeguarded, and so on, the CA declares (by including a policy to this effect in the certificate) that the public key can be used for signing million-dollar purchase orders. But this is not a granting of authority. In a properly-implemented system, the signer must still prove that s/he is authorized to sign such a purchase order (and this authorization will typically come from some entity in the environment that is not the CA). Certificate policy may be viewed as a "fit for purpose" declaration: if the signer is allowed to sign such a transaction, then this key pair can be used to create and to verify that signature.

Risk #2 ("Who is using my key?") warns that the private key stored on your computer may not be secure (without physical access controls, TEMPEST shielding, air-gap network security, video surveillance, and so on). Clearly this is true, but is equally true of all technologies that store data on a computer. In order to address this, PKI has evolved to support both "soft token" solutions (in which the user retains the private key) and roaming solutions (in which the private key may be stored at a server). As always, there are security / convenience trade-offs for each. When stored at the user's computer, the user can authenticate with his/her private key to a server that has an authentic copy of the public key. This is arguably more secure than solutions that store either a clear-text or hashed version of a password at a server in support of password-based authentication (the latter is susceptible to brute-force attack). The discussion about PKI vendors "lobbying for laws to the effect that if someone uses your private signing key, then you are not allowed to repudiate the signature" does not reflect any of the myriad debates we have heard and read on this topic. (On the contrary, if there is any reasonable evidence that someone else has used your private key, this is precisely when you can repudiate a digital signature.) In any case, in recognition of the vulnerabilities associated with typical computing platforms, PKI has come to strongly support alternative devices, such as hardware tokens and smart cards, for storing private keys and trust anchors.

Risk #3 ("How secure is the verifying computer?") examines the insecure computer question (i.e., Risk #2) again, but this time from the side of the verifying machine. As above, this risk is shared by all technologies that use computers and is not specific to PKI. Again, alternative storage devices can be helpful here.

Risk #4 ("Which John Robinson is he?") warns that the name in a certificate may not be as valuable as it appears to be ("You may know only one John Robinson personally, but how many does the CA know? … How many John Robinsons are in the New York City phone book, much less in a hypothetical phone book for the global Internet?") Additionally, the authors ask, "How do you find out if the particular John Robinson certificate you received is your friend's certificate?" But one could equally ask how you find out if the e-mail you received is from your friend John Robinson or from some other John Robinson, or how you find out if the person on the other end of the telephone is your friend John Robinson, or how you find out if the postcard you received in your mailbox is from your friend John Robinson. Real life requires us to be able to resolve the potential ambiguity with regard to a name, and we do this all the time, but this is not a problem that is either created, or purported to by solved, by PKI. Users in the electronic world, as in the physical world, need to be able to do the mapping between name and identity whether or not PKI was ever invented. This is true of all authentication technologies. A PKI binds an identifier to a public key. Associating that identifier with an identity, or with entitlements within the context of the application being used, is outside of the scope of PKI; it always has been. Applications that rely upon PKI for authentication need to recognize that this issue is not solved by PKI. (More discussion of this topic can be found in Chapter 14 of [AL03]).

Risk #5 ("Is the CA an authority?") warns that the CA may not be an authority on what the certificate contains (e.g., the corporate name of the keyholder and the DNS name of the server in an SSL server certificate). There are authorities on DNS name assignments, and authorities on corporate names, but the CA is likely to be neither of these. This is quite true but, as stated above, it is not necessarily the job of the CA to create names, or even assign names to entities; its primary function (and its authority) is to bind an identifier to a public key. As time has passed, it has become more generally recognized that a CA may make use of other authorities in order to do this task. In particular, it will often collaborate with other naming authorities to ensure that the information in a certificate is as accurate as possible.

Risk #6 ("Is the user part of the security design?") warns that users will often make security decisions (such as whether to shop at a given SSL-protected Web page) without even seeing the certificate involved or knowing whether it has any relation to what is displayed. This is certainly true, but is equally true of many security technologies. If a security infrastructure provides a technical means by which application security decisions can be automated and enforced, and then these means are not used, this is not the fault of the infrastructure and is not a risk of the infrastructure. More accurately, the "risk" is that security software is often not implemented correctly or used properly, but this is true of all security software, everywhere, and has nothing specific to do with PKI. However, in general, PKI implementations do need to provide for more useful interaction with the user [WhTy99].

Risk #7 ("Was it one CA or a CA plus a Registration Authority?") warns that "the RA+CA model allows some entity (the CA) that is not an authority on the contents to forge a certificate with that contents." This is true, but authorities in any system can always abuse their power. This is not a risk specific to PKI, but is true for all systems, everywhere. Furthermore, even if an RA+CA combination can be less secure than a single CA, there are certainly environments in which placing all power into the hands of a single authority can also be a highly risky thing to do. As in any system, it is important to choose the authorities carefully or the system will not work as intended. Over the years, explicit statements of CA practices and policies (see, for example, the framework specified in [RFC3647]) have come to be used throughout the PKI community so that external auditors and inspectors can check whether a given CA is abusing its power in some way.

Risk #8 ("How did the CA identify the certificate holder?") warns that the CA may not use good information to check the identity of the entity applying for the certificate, or may not ensure that this entity really controls the private key corresponding to the public key being certified. This is true, but again is not a risk specific to PKI. If authorities in any system do not do their jobs diligently and with integrity, the system cannot be expected to work. This is not a failing of the system itself. Authorities in a PKI (in particular, the CAs) need to be chosen carefully and trained well, but this is no different from any other system. As in Risk #7 above, auditable statements of CA practices and policies can be helpful to avoid problems in this area.

Risk #9 (“How secure are the certificate practices?”) warns, in a nutshell, that “Certificates must be used properly if you want security.” It is hard to imagine that anyone would argue with such a statement. But passwords must be used properly if you want security; biometrics must be used properly if you want security; smart cards must be used properly if you want security; and so on. Once again, this is not a risk specific to PKI; this basic statement holds true for all security technologies.

Risk #10 (“Why are we using the CA process, anyway?”) warns that PKI does not solve all security problems, even though it is sometimes marketed and sold under that premise. This is in some ways a fair criticism, as some over-zealous marketing executives have sought to increase profits by stretching the truth in several directions. However, this is not a risk of PKI. All this highlights is a need to get accurate information out regarding what PKI actually is, and what it actually does. Things have improved significantly in this area in the past few years, but more can certainly be done.

In summary, we find that of the popular PKI criticisms voiced in the literature and at various conferences, many do not apply to PKI at all, and most of the rest apply equally to all security technologies. (As a measure of items related to implementing and deploying a PKI, however, they do highlight some specific concerns. And, as with the other security technologies to which they pertain, solutions – often outside the scope of a PKI – can be applied.) For the remaining criticisms that are accurate and valid, the evolution of this technology has come to understand and address these comments so that the current (at least theoretical) view of PKI no longer appears to be deficient in these ways. Such criticisms have therefore been very beneficial to the industry as a whole.

5 PKI Evolution and a Current Definition

The comparison of approaches in Section 3 makes it clear that a PKI can be instantiated today in many different ways. But ten years ago, several of the above instantiations would not have fit into the “accepted vision” of what a PKI was. Clearly, something has changed, but is it the essence of the definition, or the implementation details? Guided by the general definition of Section 2.3, we see that the essence remains intact; only our understanding of each of the components of that definition has evolved over time.

Definition 1994. In 1994, the six components of the general definition given in Section 2.3 were restricted in the following ways.

1. *Authority*. The authority was always and only a Certification Authority (CA). There was no place in the PKI for any other kind of authority.
2. *Issuance process*. The syntax was always and only an X.509 public key certificate which binds a public key to a Distinguished Name (DN) for the user. The certificate was made available to other parties through the use of an X.500 Directory.
3. *Termination process*. The termination process was always and only a Certificate Revocation List (CRL) which could be made available to other parties through the use of an X.500 Directory (perhaps pointed at using a CRL Distribution Point in the certificate).
4. *Anchor management process.* A user may trust the CA that is “closest” to him/her (i.e., the CA that actually issued the user’s certificate) or may trust the root of a hierarchy of CAs

which includes the CA that actually issued the user's certificate. In either case, however, the client code was pre-installed with a single trust anchor and no changes were possible.

5. *Private key management process.* Very little of this was specified, although it was generally assumed that key generation occurred at the CA, registration occurred via an out-of-band, in-person process, and private keys were "safe" in the user's local environment (perhaps protected by a password).
6. *Binding validation process*. Client machines had to be configured with a large, special-purpose software toolkit that could understand all the details of certificate processing and could make validated public keys available to application code.

We now propose an updated definition of PKI.

Definition 2004. By 2004, after ten years of evolution that has resulted from extensive discussion, research, and implementation by various interested parties around the world, we find that each of the above six components of the definition has broadened considerably. However, interestingly, the same six components comprise the core of the definition. That is, the essential characteristics of the definition remain unchanged, even if the thinking about how to realize these characteristics has deepened and matured over time.

1. *Authority*. The notion of an "authority" has broadened from the CA that is "closest" to a user, to a CA that may be "farther away" (e.g., at the root of the user's hierarchy), to a CA that may be even farther away (e.g., at the root of a different hierarchy in another domain), to a CA that may be entirely independent of the user (e.g., one offered as a public service). In addition, the authoritative role of a CA might be performed by an end entity. Furthermore, it is now recognized that a CA may make use of other entities prior to issuing a binding. For example, an *Identification entity* (perhaps a Registration Authority, or some other entity altogether, such as a Naming Authority) may be used to properly determine the correctness of an identifier (on behalf of, or at the request of, a CA) before the identifier is bound to the public key. As well, PKI now recognizes the utility and value of other authorities in the environment that are not CAs, such as OCSP Responders, certificate path validation authorities, Attribute Authorities, and so on.
2. *Issuance process*. A number of different syntax proposals have been discussed and implemented over the years, and it is now well recognized that some environments will be more suited to a particular syntax than others. There is therefore a need for various ways of encoding the binding expressed by an authority. Similarly, options for the type of identifier (see Section 2.2), and the actual location of the trustworthy binding, have evolved as different choices. Certificate formats such as PGP, SPKI, SAML, XKMS Responses (see Section 6), and so on, all have a place in this broader definition. Furthermore, it is recognized that X.500 Directories are but one possible mechanism for making these bindings available to other entities, and many other technologies are now commonly in use to achieve this.
3. *Termination process*. Breaking the binding between a public key and an identifier can now use many more mechanisms than the traditional CRL. Online certificate status checkers (e.g., OCSP) were an early step in this direction, but even broader online services, such as delegated path validation [RFC3379] and XKMS servers, have also been envisioned and implemented. The use of on-line checking includes the option of online certificate retrieval, where only if the certificate is available, is it considered valid at the time of retrieval. The PKI community has come to realize that the information regarding a revoked binding needs to take different forms and use different delivery mechanisms for different environments.
4. *Anchor management process*. In support of the broader definition of authority, mechanisms for establishing how different parties accept the bindings issued by an authority have been defined and used, including trust root installation, cross-certification, trust lists, and so on. It

has become recognized that the trust anchors for a user will typically be a set of size greater than one, and that the members of this set will need to change over time.

5. *Private key management process*. Thinking in this area has broadened significantly over the past ten years as the need for flexibility in different environments became clear. To list a few examples, key generation may take place at the CA, at the client, or at a third party and protocols have been developed to handle any of the three options securely (along with protocols to allow secure backup and recovery of key material). Registration might need to be an online process (rather than an offline process) for efficiency and user convenience. As well, private keys might be stored in software, in hardware tokens, in smart cards, or in other formats, and might be stored with the user or at some 3rd-party server; protocols and interfaces have been developed over the years to handle all of these options.
6. *Binding validation process*. With respect to software, there was a growing concern that large, special-purpose toolkits were not the best alternative for some environments (for a number of reasons, including cost, size requirements, and complexity of upgrades). Interest in this area shifted to so-called "thin clients": toolkits that were very small for fast and easy download (e.g., in Java applet form). But there was also a growing realization that, in some environments at least, the ideal situation would be native applications (e.g., off-the-shelf browsers) that could properly understand all the details of certificate processing.

The current view of PKI, as expressed in "Definition 2004", is a reflection of the evolution that has occurred in this community over the past ten years. It benefits from the innovative thinking and fruitful technical discussion of researchers the world over, and has been steered greatly in more practical and useful directions by constructive criticism and numerous implementation efforts (see Sections 3 and 4 above). This definition, we believe, represents the "state of the art" in the understanding of PKI.

6 Moving From Theory to Practice

Reflecting – in an updated definition – the evolution (and the occasional revolution!) that has occurred over the years may be a useful step, but it is not sufficient. Clearly, this deeper understanding of PKI needs to be embraced in a real way in real implementations. This is not to suggest that a given PKI implementation should strive to be all things to all people. If we as a community have learned anything over the past decade, it is that the many options available for each component of the definition preclude any "one size fits all" PKI. However, even for a given set of choices, most (perhaps all) implementations can improve in both correct operation and suitability to a given environment. Many common implementation bugs and challenges have been summarized well by Gutmann [Gut, Gut02]. Specifically, Gutmann identifies issues regarding hierarchical naming, revocation, and certificate chain building. Current and prospective implementers of PKI technology would do well to look through some of this material.

One important realization of Gutmann is that original (and even some current) PKI implementations would "constrain the real world to match the PKI", as opposed to "adapt[ing] the PKI design to the real world." [Gut02]. It is hoped that we similarly capture this concern in our discussions above. In considering further issues that may remain regarding the deployment of PKI within some environments, a survey was recently performed by the OASIS PKI Technical Committee [OAS03]. The main impediments cited were the cost and lack of PKI support within client applications. Noteworthy in this regard are more recent PKI-related efforts that were motivated to address this specific concern. In particular, XML Key Management Services [XKMS03] have primarily been designed in order to abstract away some of the technical PKI detail in order to work with relatively simple clients. The generic protocol description for XKMS should allow it to support any of the PKI examples discussed in Section 3. Key management issues are part of the key registration service component (X-KRSS), while key validation issues are part of the key information service component (X-KISS). A common Web services interface may go some way to aid the otherwise difficult process of integrating these components with existing software.

An area that is yet to be widely embraced in real implementations concerns the nature of the identifier used in a certificate. There are times when there is a legitimate need for this identifier to be veronymous, other times when a pseudonym would be preferable, and still other times when an anonym should be used (even within a single environment). Yet existing CAs are typically built to use only a single type of identifier (perhaps, if the CA is very flexible, in a range of formats). Standards, in their language and in their syntax, do not generally preclude the use of different identifier types, but history and tradition have made rigid interpretations and resulted in PKI deployments that are almost exclusively one type or another. More flexibility in this area (i.e., CAs that can bind keys to any of the three types, as required) would make PKIs more suited to many real-world requirements.

The goal of this paper has been to demonstrate that the PKI community has significantly broadened its understanding of this technology over the past ten years. The challenge now is to translate that understanding to real PKI implementations that solve authentication challenges in real, heterogeneous environments.

Acknowledgements

The authors wish to thank the reviewers for their numerous comments. They greatly improved the content and readability of this paper.

7 References

[AADS99] L. Wheeler, "Account Authority Digital Signature and X9.59 Payment Standard", slides presented at the *3rd CACR Information Security Workshop*, June 1999. (http://www.cacr.math.uwaterloo.ca/conferences/1999/isw-june/wheeler.ppt)

[AL03] C. Adams, S. Lloyd, *Understanding PKI: Concepts, Standards, and Deployment Considerations, Second Edition*, Addison-Wesley, 2003.

[DH76] W. Diffie, M. Hellman, "New Directions in Cryptography", *IEEE Transactions on Information Theory,* Vol. 22, No. 6, November 1976, pp. 644.

[ElSc00] C. Ellison, B. Schneier, "Ten Risks of PKI: What You're not Being Told about Public Key Infrastructure", Computer Security Journal, vol.XVI, no.1, 2000.

[Gut] P. Gutmann, "Everything you Never Wanted to Know about PKI but were Forced to Find Out"; see http://www.cs.auckland.ac.nz/~pgut001/pubs/pkitutorial.pdf

[Gut02] P. Gutmann, "PKI: It's Not Dead, Just Resting", *IEEE Computer*, August 2002; see http://www.cs.auckland.ac.nz/~pgut001/pubs/notdead.pdf

[Just03] M. Just, "An Overview of Public Key Certificate Support for Canada's Government On-Line (GOL) Initiative", to appear in *Proceedings of 2nd Annual PKI Research Workshop*, April 2003.

[Kohn78] L. Kohnfelder, "Towards a Practical Public-key Cryptosystem", *MIT Thesis*, May. 1978.

[OAS03] P. Doyle, S. Hanna, "Analysis of June 2003 Survey on Obstacles to PKI Deployment and Usage", report of the *OASIS PKI Technical Committee*, v1.0, 8 August 2003. Available at http://www.oasis-open.org/committees/pki/pkiobstaclesjune2003surveyreport.pdf.

[PGP99] J. Callas, "The OpenPGP Standard", slides presented at the *3rd CACR Information Security Workshop*, June 1999. (http://www.cacr.math.uwaterloo.ca/conferences/1999/isw-june/callas.ppt)

[PGPks] The MIT PGP Key Server; see http://pgp.mit.edu/

[PKIX-WG] IETF Public-Key Infrastructure X.509 (PKIX) Working Group; see http://www.ietf.org/html.charters/pkix-charter.html

[Resc01] E. Rescorla, *SSL and TLS: Designing and Building Secure Systems*, Addison-Wesley, 2001.

[RFC2560] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, "X.509 Internet Public Key Infrastructure: Online Certificate Status Protocol – OCSP", Internet Request for Comments 2560, June 1999.

[RFC2692] C. Ellison, "SPKI Requirements", *Internet Engineering Task Force (IETF) Request for Comments (RFC) 2692*, September 1999.

[RFC2693] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylönen, "SPKI Certificate Theory", *Internet Engineering Task Force (IETF) Request for Comments (RFC) 2693*, September 1999.

[RFC3280] R. Housley, W. Polk, W. Ford, D. Solo, "Internet X.509 Public Key Infrastructure: Certificate and Certificate Revocation List (CRL) Profile, Internet Request for Comments 3280, April 2002.

[RFC3379] D. Pinkas, R. Housley, "Delegated Path Validation and Delegated Path Discovery Protocol Requirements", Internet Request for Comments 3379, September 2002.

[RFC3647] S. Chokhani, W. Ford, R. Sabett, C. Merrill, S. Wu, "Internet X.509 Public Key Infrastructure: Certificate Policy and Certification Practices Framework", Internet Request for Comments 3647, November 2003.

[SDSI96] R. Rivest, B. Lampson, "SDSI – A Simple Distributed Security Infrastructure", 17 September 1996, http://theory.lcs.mit.edu/~rivest/sdsi10.html

[SSH03] T. Ylonen, D. Moffat, "SSH Protocol Architecture", *Internet Draft*, October 2003. Available at http://www.ietf.org/internet-drafts/draft-ietf-secsh-architecture-15.txt.

[WhTy99] A. Whitten, J.D. Tygar, "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0.", in *Proceedings of the 9th USENIX Security Symposium*, August 1999.

[X509-00] ITU-T Recommendation X.509. *Information Technology – Open Systems Interconnection – The Directory : Public Key and Attribute Certificate Frameworks*. March 2000 (equivalent to ISO/IEC 9594-8:2001).

[XKMS03] P. Hallam-Baker, "XML Key Management Specification", W3C Working Draft, v2.0, 18 April 2003. Available at http://www.w3c.org/2001/XKMS/.