The highly secure document & data platform for capture, extraction, and storage.

1177 Branham Lane #345
San Jose, CA 95118
Web: p3idtech.com
Email: security@p3idtech.com
Main: 408-785-2005

Zero Trust and PKI: A new cryptosystem

A white paper by Bill Edwards

Public Key Infrastructure (PKI) is a fairly complex set of technologies, people, policies, and procedures, which are used together to request, create, manage, store, distribute, and (ultimately) revoke digital certificates – binding public keys with an identity (such as any MilDep organization, physical address, personal device, or email). These certificates, e.g., X.509, are very complicated and have potential vulnerabilities. For example, despite advances in security engineering, authentication in applications such as email and the Web still primarily relies on the X.509 public key infrastructure introduced in 1988.

DoD PKE has been tailored to enable secrecy, obfuscation and identity verification but it does require a large amount of trust be vested in one or more trust anchors; from the DISA CA and the DISA PKE office to internal Certificate Management Systems and the Certificate Revocation Lists themselves; all governed by DISA. Zero Trust can be used to secure a PKI infrastructure and/or enhance CRLs by automating Certificate Revocation.

Zero Trust does not require trust authorities and facilitates automated verification. The signatures are devoid of any secret data and can be used to mathematically verify the integrity of the data, providing non-repudiation, while also protecting against backdating.

PKI vendors are developing CA suites to address the scalability and portability challenges associated with automated certificate management for large-scale (such as IoT and M2M) identity management (IAM). This

1177 Branham Lane #345
San Jose, CA 95118
Web: p3idtech.com
Email: security@p3idtech.com
Main: 408-785-2005

new Zero Trust IaaS/PaaS platform can assist these PKI platform vendors to ensure the coherence and real-time resilience of their platforms, as well as strongly backstop the authenticity of identities on their ever-growing networks in a cost-effective, scalable, and compliance-related manner.

This PKI has many issues but is nearly impossible to replace. Leveraging recent progress in verifiable computation, we propose a novel use of existing X.509 certificates and infrastructure. Instead of receiving & validating chains of certificates, Navy applications receive & verify proofs of their knowledge, their validity, and their compliance with application policies. This yields smaller messages (by omitting certificates), stronger privacy (by hiding certificate contents), and stronger integrity (by embedding additional checks, e.g. for revocation).

X.509 certificate validation is famously complex and error prone, as it involves parsing ASN.1 data structures and interpreting them against diverse application policies. To manage this diversity, we propose a new format for writing application policies by composing X.509 templates, and our approach, using Zero Trust, provides a template compiler that generates C code for validating certificates within a given policy. We then use the Geppetto cryptographic compiler to produce a zero-knowledge verifiable computation scheme for that policy. To optimize the resulting scheme, our platform develops new C libraries for RSA-PKCS#1 signatures and ASN.1 parsing, carefully tailored for cryptographic verifiability. From a privacy standpoint, because modern verifiable computation protocols support zero knowledge properties for the prover's inputs, our Zero Trust validation process enables the selective disclosure of information embedded in standard X.509 certificates. Instead of revealing these

1177 Branham Lane #345
San Jose, CA 95118
Web: p3idtech.com
Email: security@p3idtech.com
Main: 408-785-2005

certificates in the clear, the prover can convey only the attributes needed for a particular application. For example, the outsourced computation can validate the prover's certificate chain, and then check that the issuer is on an approved list, that the prover's age is above some threshold, or that he is not on a blacklist.

An essential limitation of systems based on succinct proofs of outsourced computation is their reliance on a trusted party to generate the cryptographic keys. An honest key generator uses a randomly selected value to create the scheme's keys and then deletes the random value. A rogue validation key provider, however, could save the random value and use it as a backdoor to forge proofs for the associated policy. Dangerously, and in contrast to the use of rogue certificates, such proofs would be indistinguishable from honest proofs and thus undetectable.

Besides careful key management and local checks to reduce the scope of policies, a partial remedy is to generate keys using a multi-party protocol, which only requires that one of the parties involved is honest. A trust model.

Our Zero Trust model must meet the following criteria:

- *Completeness:* If the prover is providing the right password, the verifier will be convinced that it's actually the right password.
- *Soundness:* The verifier will be convinced, if and only if the prover is entering the right password.
- *Zero-Knowledgeness:* The verifier must not learn the password.

Our zero-knowledge system operates on the concept that the system has no knowledge about the content of data provided by users. Therefore, in an

1177 Branham Lane #345
San Jose, CA 95118
Web: p3idtech.com
Email: security@p3idtech.com
Main: 408-785-2005

implementation of zero- knowledge encryption, a private key, known only to the user, is used to encrypt a given set of data before it is copied to the server, which then manages the encrypted files. In conjunction with other security measures, the use of a key restricts the ability to decrypt the data to the user who originally stored it and creates social sustainability. This paper analyzes the comparative advantages of zero knowledge and traditional security schemes in cloud data storage.

Zero knowledge encryption is the joined application of the security concepts of zero knowledge proofs and encryption. Zero knowledge proofs are a means of proving one's possession of given knowledge without revealing any information to a verifier that can be used to reconstruct this knowledge. A zero knowledge proof is defined by the principles of completeness, that the pieces of revealed information can be independently verified by the verifier, soundness, that as the quantity of verified information approaches the complexity of the original problem, the probability of the solution being valid approaches one hundred percent, and perfectness or Zero Knowledgeness, that the verifier cannot reconstruct the original knowledge from the revealed information.

Encryption is a means of obfuscating information that makes use of the discrete logarithm problem. These are calculations that are easy to compute but nearly impossible to solve in the reverse direction without knowing the factor, known as a key, used to originally compute them.

Our proposal employs a fast and lightweight zero knowledge proof algorithm, which provides security of the conventional PKI. It involves client-side hashing of the userid/password to transparently generate a key-pair

1177 Branham Lane #345
San Jose, CA 95118
Web: p3idtech.com
Email: security@p3idtech.com
Main: 408-785-2005

and register the public key with the remote service provider. The server then generates aperiodic challenges and a lightweight JavaScript based client application computes the responses to each server-side challenge. Just to be clear, zero knowledge proofs aim to prove (probabilistically) a statement without revealing any information. RSA encrypts data with a public key and decrypts it with a private key. ZKP aims to take a logical proof and show that its result is correct though PKI. Some new approaches to solving various aspects we mention is a large number of such protocols are known for languages based on discrete logarithm problems, such as Schnorr's protocol and many of its variants, e.g., for proving that two discrete logs are equal. This last variant is useful, for instance, in threshold RSA protocols, where a set of servers hold shares of a private RSA key, and clients can request them to apply the private key to a given input. This model includes a trusted functionality for setting up a private/public key pair individually for each player (in fact, we only need this for the verifiers). Hence, unlike, the key setup is not tied to a particular prover/verifier pair: it can be implemented, for instance, by having the verifier send her public key to a trusted "certification authority" who will sign the key, once the verifier proves knowledge of her private key. Now, any prover who trusts the authority to only certify a key after ensuring that the verifier knows her private key can safely (i.e., in zero-knowledge) give non-interactive proofs to the verifier. Our technique requires homomorphic public-key encryption, and it preserves the communication complexity of the original protocol up to a constant factor.

For example, there are many different RSA encryption mechanisms in the literature. The oldest mechanisms use RSA to directly encrypt a user's message; this requires careful padding and scrambling of the message.

1177 Branham Lane #345
San Jose, CA 95118
Web: p3idtech.com
Email: security@p3idtech.com
Main: 408-785-2005

Newer mechanisms generate a secret key (for example, an AES key), use the secret key to encrypt and authenticate the user's message, and use RSA to encrypt the secret key; this allows simpler padding, since the secret key is already randomized. The newest mechanisms such as Shoup's "RSA-KEM" simply use RSA to encrypt log n bits of random data, hash the random data to obtain a secret key, and use the secret key to encrypt and authenticate the user's message; this does not require any padding.

Generating large amounts of truly random data is expensive. Fortunately, truly random data can be simulated by pseudorandom data produced by a stream cipher from a much smaller key. (Even better, slight deficiencies in the randomness of the cipher key do not compromise security.)

An immediate consequence of our results is non-interactive threshold RSA and discrete-log based cryptosystems without random oracles, and assuming only that each client has a registered key pair. In the context of threshold cryptography where keys must be set up initially anyway, this does not seem like a demanding assumption. Our protocols are as efficient as the best-known previous solutions (that required random oracles) up to a constant factor. This associate Zero Trust system is very functional and can always be implemented using a more standard PKI with the DoD CA, and generic zero-knowledge techniques. The verifier sends her public key to the DoD CA and proves in zero-knowledge that she knows a set of randomness that, using the given key-generation algorithm, leads to the public key she sent. One can see that all that is needed is knowledge of the challenge value e and of the RSA modulus n, plus assurance that e lies in the proper interval and that n is well formed. (Knowledge of the factorization of n, in particular, is not required.)

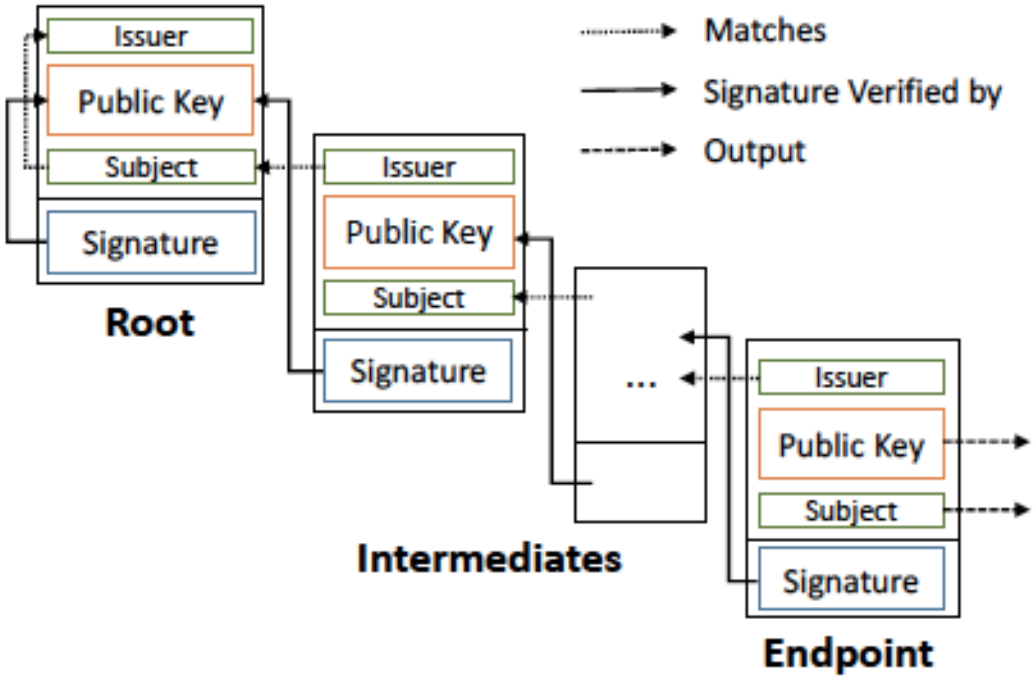
1177 Branham Lane #345
San Jose, CA 95118
Web: p3idtech.com
Email: security@p3idtech.com
Main: 408-785-2005

The X.509 PKI is slow to change; it is not uncommon for the DISA certification authority to use the same root and intermediate certificates for years, without upgrading the cryptographic primitives used for signing. For instance, the MD5 hashing algorithm remained widely used for several years after Wang et al. demonstrated that it was vulnerable to collision attacks. The ongoing migration from SHA1 to SHA2 has also been delayed by over a year, due to pressure to maintain legacy support. Similarly, a number of certification authorities have allowed the use of short RSA public keys, or keys generated with low entropy. As a moot point, the validation Zero Trust validator, outsources to the prover all of the checks that the verifier would have done on those certificates, and the verifier's job is simplified to checking only that the outsourced validation was performed correctly. This may partially mitigate these issues by allowing certificate owners to hide their public keys and certificate hashes, and hence to prevent some offline attacks; arguably, it also makes such issues harder to measure.

Mistakes and inconsistencies in implementations of X.509 have led to dozens of attacks. Famously, the first version of X.509 did not include a clear distinction between the DISA CA and endpoint certificates used by various Navy enterprise servers; e.g., Purebred key server.

Similarly, many implementations of DER are incorrect, leading to universal forgery attacks against PKCS#1 signatures; again, variations of this attack have reappeared every so often. In contrast, our new Zero Trust validation server does not trust X.509 parsers; instead, it verifies the correctness of untrusted parsing by re-serializing and hashing.

1177 Branham Lane #345
San Jose, CA 95118
Web: p3idtech.com
Email: security@p3idtech.com
Main: 408-785-2005



X.509 defines the syntax and semantics of public key certificates and their issuance hierarchy. The purpose of a certificate is to bind a public key to the identity of the owner of the matching private key (the subject), and to identify the entity that vouches for this binding (the issuer). Certificates also contain lifetime information, extensions for revocation checking, and extensions to restrict the certificate's use. The PKI's main high-level API is certificate-chain validation, which works as follows: given a list of certificates (representing a chain) and a validation context (which includes the current time and information on the intended use), it checks that:

1) the certificates are all syntactically well formed;
2) none of them is expired or revoked;
3) the issuer of each certificate matches the subject of the next in the chain;

1177 Branham Lane #345
San Jose, CA 95118
Web: p3idtech.com
Email: security@p3idtech.com
Main: 408-785-2005

4) the signature on the contents of each certificate can be verified using the public key of the next in the chain;
5) the last, root certificate is trusted by the caller; and
6) the chain is valid with respect to some context-dependent application policy (e.g. "valid for signing emails"). If all these checks succeed, chain validation returns a parsed representation of the identity and the associated public key in the first certificate (the endpoint).

As a concrete running example, consider a client who wishes to sign her email using the S/MIME protocol. She holds a certificate issued by a well-known CA for her public key, and she uses her corresponding private key to sign a hash of her message. With the current DoD PKI, she attaches her certificate and signature to the message. The recipient of the message extracts the sender's email address (from), parses and checks the sender's certificate, and verifies, in particular, that the sender's certificate forms a valid chain together with a local, trusted copy of the CA certificate; that its subject matches the sender's address (from); and that it has not expired. Finally, he verifies the signature on a hash of the message using the public key from the sender's certificate. These checks may be performed by a C function, declared as void validate (SHA2 hash, char from, time now, CHAIN certs, SIG sig) ; and for simplicity, assume that all S/MIME senders and receivers agree on this code for email signatures, with a fixed root CA.

With our new concept of Zero Trust validation process, we compile validate into cryptographic keys for S/MIME, i.e., an evaluation key and a verification key (xII-A). Navy Outlook email signature validation then proceeds as follows. The sender signs the hash of her message as usual, using the

1177 Branham Lane #345
San Jose, CA 95118
Web: p3idtech.com
Email: security@p3idtech.com
Main: 408-785-2005

private X.509 key associated with her certificate. Instead of attaching her certificate and signature, however, she attaches a new proof, meaning that a legitimate prover can always produce a proof that satisfies Verify; Zero Knowledge, meaning that the verifier learns nothing about the prover's input w; and sound, meaning that a cheating prover will be caught with overwhelming probability. This system offers strong asymptotic and concrete performance: cryptographic work for key and proof generation scales linearly in the size of the computation (measured roughly as the number of multiplication gates in the arithmetic circuit representation of the computation), and verification scales linearly with the verifier's IO (e.g., |u| + |y|), regardless of the computation, with typical examples requiring approximately 10 ms. The proofs are constant size (288 bytes).

To generate this proof, she calls the Zero Trust validation server with the S/MIME evaluation key, her message hash, email address, time, certificate, and signature. The Zero Trust server runs validate on these arguments and returns a proof that it ran correctly. Instead of calling validate, the recipient calls the Zero Trust validation server with the S/MIME verification key and the received message's hash, its from field, and it's proof. Although the certificate and signature never leave the sender's machine, the recipient is guaranteed that validate succeeded, and hence that the message was not tampered with.

These certificates are the staple solution today for verifying that a public key belongs to an individual. These technologies are used in conjunction with everything from web-based browser authentication to e-commerce and identity management for access to government and commercial e-services.

1177 Branham Lane #345
San Jose, CA 95118
Web: p3idtech.com
Email: security@p3idtech.com
Main: 408-785-2005

The policy may incorporate some application logic or even algorithms not implemented by the certificate verifier. The policy may be deployed, e.g., as a key in the configuration of the client banking application, or by re-using existing public key management mechanisms, such as key pinning. We expect such ad hoc, application-specific deployment of policies to help break the circular dependency of servers only wanting to use a policy once supported by almost all clients. Our concept of the Zero Trust validation server policies also enables some emancipation from traditional certificate issuers, notably root CAs, who can currently impersonate any of their customers. As an example, a DoD S/MIME policy may require that a class of official mail be signed by two certificates, issued by two independent CAs (hence the DoD CA and the Navy VM CA), or that the sender certificate be endorsed by some independent organization yet to be named. Similarly, just pushing a new key to the browser or the client software can deploy such policies.

For example, instead of installing a root certificate key to access some exotic service, installing a Zero Trust key for that service is more specific and more versatile, inasmuch as the client, or some trusted third party, can review the precise policy associated with the key.

Empirically, many past vulnerabilities have been due to bugs in X.509 certificate parsing and validation code using the DoD CAC accessing certain web portals using ill advised browsers, for example in their handling of ill-formed certificates, or their lax interpretation of certificate-signing flags, and each of those bugs required a full software patch. Our technical answer to this class of problem is to mostly generate the parsing and validation

1177 Branham Lane #345
San Jose, CA 95118
Web: p3idtech.com
Email: security@p3idtech.com
Main: 408-785-2005

routines from high-level specifications; however, we note that this approach can be applied to the native validation code, and an interesting research direction is to certify the compilation process. In addition, we argue any (potential) bug in a Zero Trust validation key policy or its implementation would be easier to patch by updating the policy key, regardless of the variety of application implementations. Furthermore, after Zero Trust's validation server key generation phase, if the generation is done honestly, there is no longer any secret associated with the Zero Trust keys, so they cannot be dynamically compromised.

The fixed code used by the Zero Trust verifier itself constitutes a smaller trusted computing base, used in a uniform manner, independent of the actual certificate contents. However, it also means that implementation-specific checks that depend on non-public values of the certificate are no longer possible.

The Zero Trust validation server supports the RSA PKCS#1v1.1 signature verification algorithm on keys of up to 2048 bits, coupled with the SHA1 and SHA256 hash functions. This combination of algorithms is sufficient to validate over 95% of recently issued certificate chains on the Web, according to recent PKI measurement studies. We assume all RSA certificates use the public exponent e = 65537, the only choice in practice. Our Zero Trust Certificate System supports several different cipher suites with the RSA key exchange: AES and SHA-256 Message Authentication. Advanced Encryption Standard (AES) ciphers have a fixed block size of 128-bits, and the keys can be either 128-bit or 256-bit. ... These cipher suites are FIPS-compliant. For example, If you want more security, RSA does not scale well—you have to increase the RSA modulus size far faster

1177 Branham Lane #345
San Jose, CA 95118
Web: p3idtech.com
Email: security@p3idtech.com
Main: 408-785-2005

than the ECDSA curve size. 1024 bit RSA keys are obsolete, 2048 are the current standard size. If you need to go farther, you'd stuck. First, if CA does not provide 4096 bit RSA keychain, signing your own 4096 bit RSA key with a 2048 RSA intermediary doesn't make sense. Second, note that every doubling of an RSA private key degrades TLS handshake performance approximately by 6–7 times. So, if you need more security, choose ECC. Our Zero Trust model employs forward secrecy. Forward secrecy means that if a private key is compromised, past messages, which are send cannot also be decrypted. Thus it is beneficial to have perfect forward secrecy for your security (PFS).

For ECC (elliptic curve cryptography), given two rational points on an elliptic curve, the line through them will almost always intersect the curve at one additional point, which will again have rational coordinates. It's easy to use two rational points to generate a third, but it's hard to do the reverse — to take one rational point and find two rational points that would generate it via the straight-line method. This is what makes elliptic curves so useful for cryptography: Operations that are easy to do but hard to undo are fundamental to cryptographic security.

The difference between ECDHE/DHE and ECDH is that for ECDH one key for the duration of the SSL session is used (which can be used for authentication) while with ECDHE/DHE a distinct key for every exchange is used. Since this key is not a certificate/public key, no authentication can be performed. An attacked can use their own key. Thus when using ECDHE/DHE, you should also implement client key validation on your server (2-way SSL) to provide authentication.

1177 Branham Lane #345
San Jose, CA 95118
Web: p3idtech.com
Email: security@p3idtech.com
Main: 408-785-2005

ECDHE and DHE give forward secrecy while ECDH does not. ECDHE is significantly faster than DHE. There are rumors that the NSA can break DHE keys and ECDHE keys are preferred. On other sites it is indicated DHE is more secure. The calculation used for the keys is also different. DHE is prime field Diffie Hellman. ECDHE is Elliptic Curve Diffie Hellman. ECDHE can be configured. ECDHE-ciphers must not support weak curves, e.g. less than 256 bits.

Capabilities of our Zero Trust server, client and certificate authority (required compatibility); one would choose a different cipher suite for an externally exposed website (which needs to be compatible with all major clients) than for internal security.

- Encryption/decryption performance
- Cryptographic strength; type and length of keys and hashes
- Required encryption features; such as prevention of replay attacks, forward secrecy
- Complexity of implementation; can developers and testers easily develop servers and clients supporting the cipher suite?

Our Zero Trust model secures and protects cryptographic keys in this manner:

1. Store cryptographic keys in a secure digital vault – Move keys into a digital hardware vault with multiple layers of security wrapped around it, enforce multi-factor authentication to all users who have access to the vault.
2. Introduce role segregation – Control individual access to stored keys, preventing even the most privileged administrators from getting to them unless explicit permissions have been granted.

1177 Branham Lane #345
San Jose, CA 95118
Web: p3idtech.com
Email: security@p3idtech.com
Main: 408-785-2005

3. Enable secure application access – Enable access to stored keys for authorized applications and verify that the applications are legitimate.
4. Audit and review access key activity – Audit all activity related to key access and implement trigger events to alert the necessary individuals of any key activity.
5. Enforce workflow approvals – Enforce workflow approvals for anything considered being highly sensitive and the same goes for accessing the keys.
6. Monitor cryptocurrency administrator activities – Facilitate connections – *similar to an automated secure proxy/jump host* – to target systems that are used to perform cryptocurrency administrator activities (e.g. the system hosting in hardware, e.g., embedded secure element (eSE) or wallet). The DoD probably won't adhere to cryptocurrency but the current trend is that cryptocurrencies use cryptography for three main purposes; to secure transactions, to control the creation of additional units, and to verify the transfer of assets.

For example, a bitcoin address is created from an ECDSA keypair. It is common to use a hashed version of the public key as the shared address, but the original bitcoin implementation also allowed for using the unaltered key directly. (which is revealed when the coins in the address is spent) The purpose of public key cryptography for our use case is that the owner can prove ownership of the address by a digital signature, which is required by the blockchain before accepting spending of the coins in an address. The ECDSA algorithm is not suited for encrypting messages. If an RSA keypair was used it would allow the sender of money to encrypt and convey some personal information to the receiver (e.g., by a public message server),

1177 Branham Lane #345
San Jose, CA 95118
Web: p3idtech.com
Email: security@p3idtech.com
Main: 408-785-2005

which obviously could be a useful feature. Is there any reason why an RSA keypair should or could not be used for cryptocurrency addresses? We propose use a Schnorr-style signature scheme, such as Ed25519, with size and computational cost comparable to the ECDSA over secp256k1 as used by Bitcoin, and then reuse the key pairs for ECDH/ECIES. XEdDSA enables use of a single key pair format for both elliptic curve Diffie-Hellman and signatures. In some situations it enables using the same key pair for both algorithms.

If quantum computers were built, they would pose concerns for public key cryptography, as we know it. Among other cryptographic techniques, they would jeopardize the use of PKI X.509 certificates (RSA, ECDSA) used today for authentication. Even though post-quantum signatures could work well for some use cases like software signing, there are concerns about the effect their size and processing cost would have on technologies using X.509 certificates. In this work, we investigated the viability of post-quantum signatures in X.509 certificates and protocols that use them (e.g. TLS, IKEv2). We will propose to the Navy a pilot to evaluate existing mechanisms built-in the protocols that deal with large records like record fragmentation, segmentation, caching, and compression. We think that it is not rare that large X.509 certificates and certificate chains need to be transferred over UDP as part of the IKEv2 peer authentication. Thus, fragmentation is already widely used on the Internet today in order to carry lengthy certificates that do not fit in the path MTU. Thus, it is straightforward to add support for new proposed post-quantum signature schemes in X.509 when necessary by defining new algorithm identifiers (that correspond to certain post-quantum signature scheme parameters and structures).

1177 Branham Lane #345
San Jose, CA 95118
Web: p3idtech.com
Email: security@p3idtech.com
Main: 408-785-2005

X.509 revocation checking can take one of two forms: revocation lists (CRL) must be downloaded out of band, while the online certificate status protocol (OCSP) can either be queried by the validator to obtain a signed proof of non-revocation; or this proof may be stapled to the certificate to prevent a high-latency query. The DISA CA requires delegation. Many practical Navy applications rely on some form of authentication delegation. In particular, many Navy servers delegate the delivery of their web content-to-content delivery networks (CDNs). Websites that use HTTPS with a CDN need to give their X.509 credentials to the CDN provider, which can cause serious attacks when CDNs improperly manage customer credentials. This proposes to reflect the authentication delegation of HTTPS content delivery networks as X.509 delegation. Unfortunately, this is impractical, because it requires an extension of X.509, which the DoD CA is unlikely to implement, as it is detrimental to their DoD CIO policy. Our Zero Trust validation approach allows a content-owner (Navy) to implement secure X.509 delegation to CDNs using short-lived pseudo-certificates, without the DoD CA's cooperation.

Several trust anchors are involved to authorize the binding of public keys with user identities. User identities are unique within each CA domain and third party Validation Authorities (VAs) can provide a validation service on behalf of the CA. Registration Authorities (RAs), Certificate Revocation Lists (CRLs) and Online Responders (Online Certificate Status) further complicate this picture, ensuring that the validity of signatures signed with a private key cannot be legally denied by the signer (non-repudiation) and valid – that the certificate is specifically bound to an individual in a way that is legally recognized, that each certificate's signature is valid, the current date and time are within the certificates validity period, and that the

1177 Branham Lane #345
San Jose, CA 95118
Web: p3idtech.com
Email: security@p3idtech.com
Main: 408-785-2005

certificate(s) have not been corrupted or malformed all the way up a certificate chain to the root certificate.

Further complicating this picture, conventional PKI solutions typically require manual interaction for the certification of a public key during an identity check. While this is not a substantial issue with e-mail encryption, (the participants are natural persons), this becomes problematic with machine-to-machine (M2M) based authentication where the embedded systems are machines, which require automatic processing of certification requests. How can any of these interactions be trusted without verification? Currently, as the only tool available for identity management, PKI as a scalable identity infrastructure has proven impractical to secure the billions of mobile devices (based on the Zero Trust Model), as well as the networks they utilize. There has been an explosion in the number of required certificates, as each device requires it's own unique certificate. Moreover, many of these M2M networks are distributed and decentralized, potentially having to utilize many disparate CAs. The framework breaks. This liability means it is imperative to securely automate certificate provisioning, renewal, and revocation processes.

A root CA is the important point of trust in an organization and subordinate CA are created to provide administrative benefits from the root and are set up practically to separate usage, organizational division, geographic divisions, load balancing, and backup and fault tolerance. These configurations and associated policies benefit from DoD ZERO TRUST integration.

Zero Trust can be implemented into these systems to secure and provide real-time continuous integrity monitoring of all critical CA management

1177 Branham Lane #345
San Jose, CA 95118
Web: p3idtech.com
Email: security@p3idtech.com
Main: 408-785-2005

applications such as newly formed NSA/NIST approved Certificate Services for a particular PKI deployment. These services include any CryptoAPIs (FIPS 140-2 compliance for LOA4) and Cryptographic Service Provider (CSP) dependencies underlying the DoD PKE system for cryptographic operations and private key management, as well as signing and verifying the Certificate stores themselves, which are responsible for storing and managing certificates in the Navy enterprise.

Moreover, with the increasing use of software-based DoD CSPs, private keys and cryptographic operations are not well isolated from the server they run on and the operating system, e.g., Purebred. With this vulnerability, application or DoD OS (any Navy OS) tampering are common exploitation approaches to expose keys and is in fact one of PKIs most glaring fundamental vulnerabilities. With ZERO TRUST and a new novel "trust platform", configuration and application baselines can be monitored in real-time to ensure that software-based DoD and Navy CSPs are secure with real-time tamper evidence of dependent Navy and DoD applications and the various OS (Office 365) components themselves in the event of attack.

For a Navy CA installation, ZERO TRUST is implemented to sign and provide real-time validation to Navy CA configuration files; security-related files responsible for permission management between DoD PKI subsystems (LRAs), and to ensure tamper detection of any certificate templates used by the Navy CA and infrastructure for DoD Certificate Services. These include all DoD Certificate Server services, Public Key Group policies, Issuer Statements, Certificate Database Logs and their associated configuration files (and dependencies) as well as common web enrollment applications associated with a new Navy PKI deployment.

1177 Branham Lane #345
San Jose, CA 95118
Web: p3idtech.com
Email: security@p3idtech.com
Main: 408-785-2005