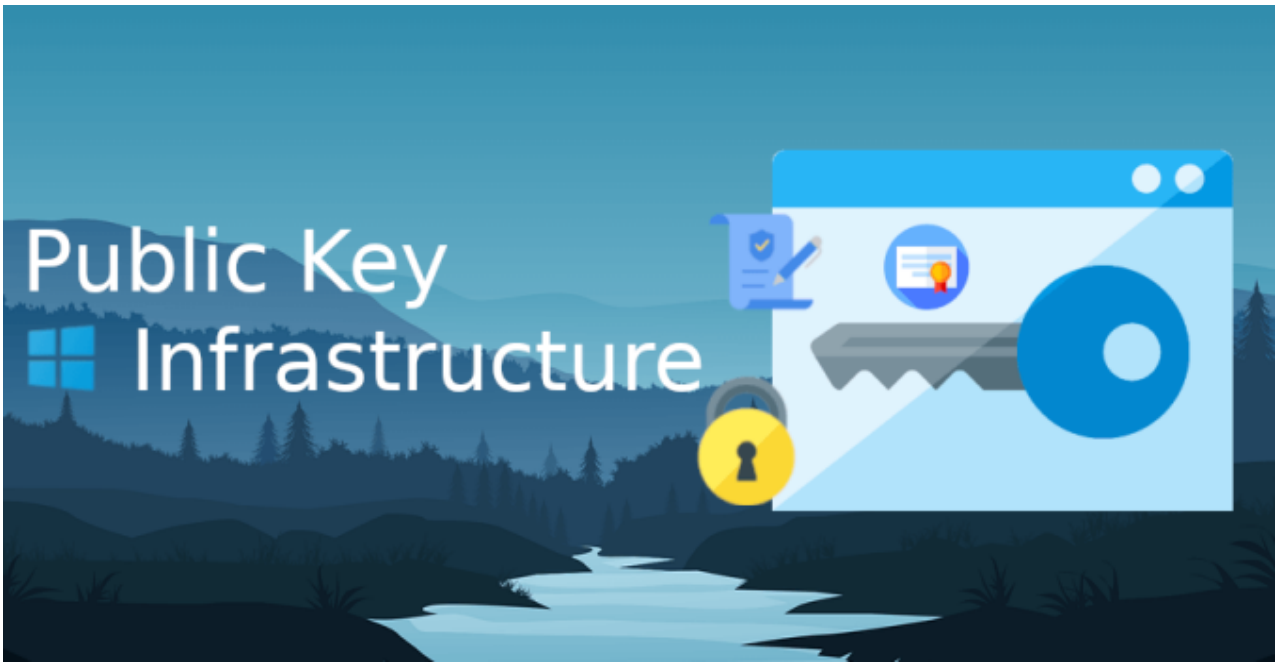
# PKI – Part 3: The role of hash functions in PKI

michaelwaterman.nl/2023/09/15/pki-part-3-the-role-of-hash-functions-in-pki

Michael Waterman September 15, 2023




With Public Key Infrastructure (PKI), cryptographic hash functions play a pivotal, yet often understated role, operating ceaselessly to secure the integrity and authenticity of digital data as it navigates through contemporary communication networks. To appreciate fully the indispensable role they occupy in PKI, it is essential first to delineate the fundamental principles of hash functions and to understand why they are frequently analogized as the digital fingerprints within the cybersphere.

Hash functions are specific types of algorithms that render an input — or 'message' — into a fixed string of bytes, generally producing a digest that is distinct for each set of unique inputs. Essentially, they create a digital fingerprint for a file, yielding a unique identifier that singularly represents the corresponding data. This distinctiveness ensures that even a minuscule modification to the input data, such as altering a single character, will generate a markedly different hash output. This attribute underpins the security apparatus deployed across a multitude of applications in PKI.

## Definition of hash functions

At its core, a **hash function** is a special kind of algorithm that takes an input (or 'message') and produces a fixed-size string of bytes, typically in the form of a digest that is distinct for every different set of inputs. The output, commonly referred to as the hash code, serves as a unique identifier for the data it represents. Through this mechanism, hash functions essentially create a 'digital fingerprint' of the input data.

### Essential properties

Hash functions are constructed to satisfy several critical properties that facilitate their broad spectrum of applications in digital security frameworks. These properties are:

1. **Deterministic:** Meaning that the same input will always produce the same hash output, ensuring a consistent and reliable representation of the data.
2. **Fast to Compute:** The function can calculate the hash code swiftly, allowing for efficient processing and verification of data.
3. **Preimage Resistance:** It is computationally infeasible to retrieve the original input from its hash output, thereby safeguarding the data against reverse-engineering attempts.
4. **Small Changes, Big Differences:** A minor alteration in the input should produce such a dramatically different hash that the new hash appears uncorrelated with the old hash, a property known as the avalanche effect.
5. **Collision Resistance:** It should be extremely unlikely, albeit not impossible, to find two different inputs that hash to the same output, preventing malicious actors from substituting malicious data for legitimate data.

By adhering to these properties, hash functions form a resilient defense against various cyber threats, including data tampering and unauthorized access.

## Understanding hashes: A beginner's guide

Hashes are like unique digital fingerprints for files, documents, or certificates. They help in verifying the integrity and authenticity of these digital assets. Let us break down how hashes are used step-by-step, in the simplest terms:

### Step 1: Creating a hash

1. **Imagine you have a document:** Think of a hash as a unique signature for your document.
2. **Applying a hash function:** A hash function is a special tool that takes the contents of your document and creates a unique hash (or signature) from it. This hash is a string of numbers and letters.

### Step 2: Transmitting the document

1. **Sending your document:** When you send your document to someone else, the hash goes along with it.
2. **Staying the same:** It is important that the document stays exactly the same during transmission, or the hash will change.

### Step 3: Verifying the document

1. **Receiving the document:** On the other side, the person receiving the document also gets the hash you sent.
2. **Creating a new hash:** The receiver creates a new hash of the document using the same hash function you used.

3. **Matching the hashes:** The receiver then compares the new hash they created with the hash you sent.

#### Step 4: Confirmation

1. **If the hashes match:** If the hashes match, it confirms that the document has not been altered in any way during transmission.
2. **If the hashes don't match:** If the hashes don't match, it signals that the document might have been tampered with, and it's not as it was originally.


#### Why is this important?

1. **Security:** Hashes add a level of security, ensuring that the files or documents you are receiving have maintained their integrity.
2. **Trust:** It builds trust in digital communications, reassuring that the exchanged data is authentic and untampered.

### Common hash algorithms

Over the years, several hash algorithms have been developed and deployed in different security systems, including:

- **MD5 (Message Digest Algorithm 5):** Once widely used, it has now been largely deprecated due to vulnerability to collision attacks.
- **SHA-1 (Secure Hash Algorithm 1):** Similarly to MD5, it is now considered insecure against well-funded attackers.
- **SHA-2:** A family of hash functions that are currently deemed secure and are widely used in various security applications and protocols.
- **SHA-3:** The latest addition to the Secure Hash Algorithm family, providing a higher level of security and performance efficiency.

## The role and importance of hash functions in PKI

Public Key Infrastructure (PKI) remains a fundamental framework in the secure communications space, underpinning numerous services from secure email to online banking. At the heart of this secure communications ecosystem lie hash functions, facilitating secure transactions and communication through a diverse range of applications. This chapter delves into the role and significance of hash functions in PKI, highlighting their indispensable function in maintaining the integrity and authenticity of digital data.

#### Ensuring data integrity

Hash functions are the gatekeepers of data integrity in PKI. Before any data is transmitted, it is first passed through a hash function to generate a unique hash value or ‘digital fingerprint’. This value is then used at the recipient’s end to verify the integrity of the received data, a process fundamental in detecting any form of data tamperings such as unauthorized alterations and interceptions.

### Authentication

By utilizing hash functions in tandem with digital signatures, PKI can authenticate the source of a message, ensuring it originated from a legitimate source. This is vitally important in secure communications as it facilitates trust and verifies the identity of the communicating parties, thereby preventing impersonation and fraud.

### Non-Repudiation

Hash functions facilitate non-repudiation in digital transactions. This ensures that a sender cannot deny the authenticity of the message they sent. By creating a unique hash of a message and encrypting it with a private key, the sender unequivocally associates themselves with the message, making disavowal impossible at a later stage.

### Digital certificates

Digital certificates, a vital component of PKI, rely heavily on hash functions. These certificates include a public key and the identity to which it is associated. Hash functions, in this regard, are used to create a fingerprint of the certificate, aiding in its verification and ensuring that the certificate remains untampered and valid.

### Secure password storage

In modern systems, passwords are seldom stored in plaintext format to prevent security breaches. Hash functions are utilized to store passwords securely. When a user creates or updates their password, it is hashed, and the hash value is stored. During login, the entered password is hashed again, and the hash value is compared with the stored hash value to authenticate the user.

## Understanding hash functions

In the previous chapters, I laid the groundwork for the fundamental roles and definitions pertaining to hash functions in the context of PKI. In this chapter, I’ll dive deeper to dissect the underlying principles of hash functions, offering a meticulous breakdown of the intricate processes and characteristics that define them. Let us embark on this exploratory journey to understand the mechanics of hash functions more profoundly.

### Properties of hash functions

To fully grasp the essence of hash functions, one must first become acquainted with their quintessential properties that govern their operation. These properties stand as the pillars supporting the reliable and secure functioning of hash functions in PKI:

1. **Deterministic:** Ensuring that identical inputs will consistently yield the same hash output, a property that ensures a steady and reliable representation of the data.
2. **Quick computation:** The function is designed to rapidly calculate the hash code, thereby promoting efficient data processing and validation.
3. **Preimage resistant:** It should be computationally challenging to deduce the original input from its hash output, a feature that protects the data from reverse-engineering endeavors.
4. **Avalanche effect:** A phenomenon where a minor alteration in the input orchestrates a substantially different hash, a characteristic essential in securing data.
5. **Collision resistance:** It should be a Herculean task to find two distinct inputs that hash to the identical output, a feature that guards against data substitution and other forms of attacks.

## Hash functions in PKI

After establishing a foundational understanding of hash functions in our preceding discussions, it is time to spotlight their critical role within the context of Public Key Infrastructure (PKI). Hash functions actively contribute to several functionalities in PKI, underpinning secure and efficient communications. This chapter seeks to detail the operative dynamics of hash functions in PKI, emphasizing their integration in systems where security is paramount.

### Digital signatures

In the realm of PKI, digital signatures stand as a testimony to the authenticity and integrity of a message or a document. Here, hash functions come into play by creating a unique hash of the message, which is then encrypted using the sender’s private key. This process ensures that the message is untampered and authentic, establishing a secure communication channel.

### Certificate Authorities (CA) and Digital Certificates

Certificate Authorities heavily rely on hash functions to secure digital certificates. A digital certificate’s hash facilitates its easy verification, ensuring the document remains unaltered and trustworthy. CAs utilize hash functions to maintain the integrity of these certificates, providing a bedrock of trust in the digital ecosystem.

### Secure Sockets Layer (SSL) and Transport Layer Security (TLS)

SSL and TLS protocols, which are vital in securing communications over a network, use hash functions during the handshake process. These functions authenticate the parties involved and establish encrypted sessions, ensuring a secure corridor for data transmission.

### Secure password storage

As delineated in previous chapters, hash functions provide a robust mechanism to securely store passwords, transforming them into unique hash values, which are then stored instead of the plaintext passwords, thereby augmenting security by several notches.

### File integrity checks

Hash functions are indispensable tools in file integrity checks, where they validate the integrity of files by comparing the hash value of the original and the received file. This process is integral in detecting any form of alterations or corruptions that might have occurred during transmission.

### Timestamping

Timestamping services leverage hash functions to record the exact time a document was created or altered. By hashing the document alongside the timestamp, these services provide a secure way to verify the authenticity and the exact time of creation or modification of a document.

## Digital signatures

As I go deeper into the nuances of Public Key Infrastructure (PKI), it is essential to shed light on a critical aspect that stands central to secure digital communications – digital signatures. Through this chapter, I will delve into the intricate workings of digital signatures, explaining how they leverage hash functions to offer secure, authenticated, and irrefutable transactions.

### Defining digital signatures

At their essence, digital signatures are cryptographic equivalents of handwritten signatures or stamped seals, but much more secure. They offer a high level of security and are used to verify the authenticity of digital messages or documents.

### The role of hash functions in digital signatures

Hash functions play a pivotal role in the creation and verification of digital signatures. Here's how:

1. **Creation:** When a message is to be sent, it is first hashed, creating a message digest. This digest is then encrypted with the sender's private key to create the digital signature. The signature, along with the message, is then sent to the recipient.
2. **Verification:** Upon receiving the message and its digital signature, the recipient decrypts the signature using the sender's public key, retrieving the message digest. Simultaneously, the recipient creates a new hash of the received message. If this new hash matches the decrypted message digest, it confirms the message's integrity and the sender's authenticity.

### Characteristics of digital signatures

Digital signatures offer a range of features that establish them as a secure means of authentication:

1. **Authentication:** They verify the origin of the message, confirming the identity of the sender.
2. **Integrity:** By identifying any alterations made to the message post-signing, digital signatures ensure data integrity.
3. **Non-repudiation:** Digital signatures make it impossible for the signer to deny the authenticity of the message they signed, ensuring non-repudiation.

### Applications of digital signatures

In the modern digital landscape, digital signatures find applications in various domains, ensuring secure transactions and communications in:

- **Legal documents:** Authenticating legal documents and ensuring they remain unaltered during transmission.
- **Emails:** Securing email communications by authenticating the sender and safeguarding the email content.
- **Software distribution:** Verifying the integrity of software files and establishing the authenticity of the distributor.

## How hash functions are used in digital signatures

In previous chapters, I have explored the pivotal role digital signatures play within the PKI landscape. Now, it is prudent to delve further to understand precisely how hash functions are utilized in the creation and verification of digital signatures, serving as the linchpin in the security protocol.

### Step-by-step process of using hash functions in digital signatures

Understanding the application of hash functions in digital signatures necessitates breaking down the process step by step, delineating the detailed workflow:

1. **Message digest creation:** The initiation of the process involves creating a message digest. The original message undergoes a hashing algorithm, resulting in a fixed-size string of characters, which is the message digest.
2. **Digital signature creation:** The message digest isn’t sent as is. It is encrypted using the sender’s private key, creating what is known as the digital signature.
3. **Transmission:** This digital signature, coupled with the original message, is then transmitted to the recipient.
4. **Verification process:** Upon receiving the message and its corresponding digital signature, the recipient embarks on the verification process. Using the sender’s public key, they decrypt the digital signature to retrieve the original message digest.
5. **Hashing the received message:** Simultaneously, the recipient hashes the received message to create a new message digest.

6. **Matching the hashes:** In the final step, the new message digest is compared with the original message digest retrieved from the digital signature. If they match, it authenticates the integrity of the message and the identity of the sender.

#### Ensuring security through hash functions

Hash functions ensure that the digital signature process is secure by:

- **Maintaining data integrity:** By creating a unique hash of the message, any alteration, however minor, during transmission, will result in a different hash, flagging potential security breaches.
- **Authentication:** The process authenticates the sender, as only they have the private key needed to create the digital signature.

#### Potential vulnerabilities and countermeasures

Though hash functions bolster the security of digital signatures significantly, they are not impermeable to attacks. It's crucial to be aware of potential vulnerabilities, such as collision attacks, and to leverage countermeasures like utilizing strong, cryptographically secure hash functions.

## Certificate creation

As I progress deeper into the intricacies of PKI, the next logical juncture leads us to the exploration of certificate creation. This process stands as a cardinal point in the PKI, serving as the genesis of a certificate's lifecycle, a digital passport that facilitates secure and authenticated communications in the digital ecosystem.

### How hash functions are utilized in creating digital certificates

understanding the role of hash functions in the creation of digital certificates becomes vital. Digital certificates, akin to digital IDs, authenticate the identity of the certificate holder and provide the public key to establish secure communications. In this chapter, I'll dissect the intricate role of hash functions in the certificate creation process, emphasizing their fundamental contribution to ensuring security and trust in digital environments.

#### The birth of a digital certificate

The creation of a digital certificate commences with a Certificate Signing Request (CSR). It is during this stage that hash functions begin to play a crucial role. Let's dive deeper into the steps involved and witness the pivotal role hash functions play:

1. **Generating a key pair:** The first step in certificate creation involves generating a pair of cryptographic keys: a public key and a private key.
2. **Creating a certificate signing request (CSR):** The public key, along with other identification information, is incorporated into a CSR, which is further signed using the private key. A hash function is applied to the CSR to maintain data integrity and to secure the information therein.

3. **Signature by Certificate Authority (CA):** Once the CSR is forwarded to a CA, it uses a hash function to create a message digest of the CSR and then signs it using its private key, creating a digital signature. This digital signature is embedded in the digital certificate, ensuring its authenticity and integrity.

### Anatomy of a digital certificate

A digital certificate contains several fields that hold critical information. Notably, it encompasses the public key of the certificate holder and the digital signature generated by the CA. Hash functions work diligently behind the scenes, orchestrating a secure and tamper-evident environment for these elements.

### Validating the digital certificate

Whenever a digital certificate is presented, its authenticity is verified by utilizing hash functions to corroborate the integrity of the certificate. The validation process involves hashing the certificate's contents and comparing it with the decrypted CA signature (which reveals the original hash) to ensure they match.

## The role of hash functions in certificate fingerprinting

a pertinent concept that emerges is certificate fingerprinting. A certificate fingerprint, also known as a thumbprint, is a unique identifier for a digital certificate, created through the process of hashing the certificate data. In this chapter, I elucidate the pivotal role hash functions perform in certificate fingerprinting, ensuring the unique identification and verification of digital certificates.

### What is certificate fingerprinting?

Certificate fingerprinting involves applying a hash function to a digital certificate to create a unique identifier or 'fingerprint' for the certificate. This fingerprint serves as a reliable and quick method to identify and manage certificates, providing a unique marker for each certificate in circulation.

### The mechanics of certificate fingerprinting

Delving into the mechanics, I notice the cardinal role played by hash functions at every juncture:

1. **Creation of the fingerprint:** After a digital certificate is created, a hash function (like SHA-256) is applied to the entire certificate file. This process yields a unique series of numbers and letters – the certificate fingerprint.
2. **Unique Identification:** The fingerprint, being unique, assists in pinpointing a specific certificate among a pool of certificates, facilitating easy identification and management.
3. **Verification of Integrity:** The fingerprint can be used to verify the integrity of a certificate. By comparing the fingerprints before and after transmission, one can ensure that the certificate has remained unaltered during transmission.

### Applications and utilities

Certificate fingerprints are not just theoretical constructs but find substantive applications in real-world scenarios, including:

- **SSL/TLS certificates:** In the context of SSL/TLS certificates, fingerprints are utilized to verify the authenticity and integrity of the certificates during the handshake process.
- **Quick certificate lookup:** Fingerprints offer a quick way to look up and manage certificates in large environments with numerous certificates, saving time and resources.
- **Security assurance:** Being a quick and reliable method to verify a certificate's authenticity, it augments security procedures by offering a way to ascertain the integrity of certificates.

## Transitioning to more secure hash algorithms

As the digital landscape evolves, so does the sophistication of potential threats. Within this dynamic environment, the cornerstone of security lies in the continual advancement and adaptation of cryptographic tools, such as hash functions, to bolster digital security. In this chapter, I am set to traverse through the critical pathways of transitioning to more secure hash algorithms, phasing out older vulnerable functions, and embracing the stalwarts of security like SHA-256 and SHA-3.

### Phasing out older, vulnerable hash functions

In the realm of cybersecurity, complacency can be a harbinger of vulnerability. Hash functions, once considered impervious, might over time exhibit vulnerabilities due to the advent of more potent computational resources and analytical techniques. This section sheds light on the necessity to phase out older hash functions that have become susceptible to attacks, such as collision attacks, which can potentially compromise the integrity of digital signatures and certificates. Here, I will unravel the inherent risks associated with the continued use of outdated hash algorithms and underscore the importance of a timely transition to more secure hash functions.

### Adoption of more secure hash algorithms like SHA-256 and SHA-3

Taking a stride towards fortified security demands the adoption of modern and robust hash algorithms. In this segment, I'll take a closer look at the championing algorithms of today, such as SHA-256 and SHA-3. These algorithms stand tall with enhanced resistance against known vulnerabilities, offering a much-needed shield in the present digital ecosystem. By diving into the features, strengths, and applications of SHA-256 and SHA-3, I facilitate a comprehensive understanding that underscores their role as the sentinels of present-day digital communications, paving the path for secure, authentic, and immutable digital transactions. Moreover, I'll touch upon the practical aspects of transitioning to these secure algorithms, spotlighting their role in fostering a resilient digital infrastructure.

## Hashing of certificates in a Microsoft PKI

Microsoft stands as a significant player, offering solutions that are intricately designed to bolster security in digital communications. Central to this endeavor are hash functions, facilitating the hashing of certificates to ensure a robust line of defense against potential threats. In this chapter, I will uncover the nuances of hashing certificates in a Microsoft PKI, emphasizing the strategies employed and the standards adhered to in ensuring optimal security.

Microsoft PKI supports a range of hashing algorithms, with a distinct focus on encouraging the use of more secure and contemporary algorithms such as SHA-256. In this segment, I'll outline the various algorithms supported and their respective roles and functionalities in ensuring the security of digital certificates.

### The process of certificate hashing in Microsoft PKI

Diving deep into the process, I'll elucidate the step-by-step procedure undertaken in Microsoft PKI for certificate hashing:

1. **Certificate creation:** During the certificate creation process, the information to be encapsulated in the certificate is defined, followed by the application of a hash function to create a unique fingerprint.
2. **Certificate signing:** Post creation, the certificate is signed by a Certificate Authority (CA) using a hashing algorithm to ensure its integrity and authenticity.
3. **Verification and authentication:** Upon receipt of a certificate, the receiving entity can utilize the public key of the CA to verify the certificate's authenticity, ensuring that it has remained untampered during transmission.

Just remember the following when choosing the hash during any operation:

## Hashing recommendations in a Microsoft PKI

In a Microsoft Public Key Infrastructure (PKI) featuring multiple tier levels it is pivotal to employ hashing algorithms judiciously to navigate the fine line between robust security and broad compatibility. Here I delineate the hashing algorithms that strike a harmonious balance at each tier:

### Root CA certificate

At the apex of the hierarchy, where the Root Certificate Authority (CA) resides, the utmost level of security is non-negotiable. For the Root CA:

- **Recommended hashing algorithm:** SHA-256 or higher (SHA-384, SHA-512)
- **Considerations:** Given that Root CAs have a long lifespan and are not frequently used in operations, prioritizing security over compatibility is recommended. Deploying a high-security hashing algorithm shields the infrastructure against potential vulnerabilities for a prolonged period.

- **Compatibility:** While prioritizing security, it's pertinent to ensure that the chosen hashing algorithm is compatible with the organizational infrastructure, including legacy systems, to prevent potential interoperability issues.

### Enterprise (Intermediate) CA

Occupying the middle tier, the Enterprise or Intermediate CA facilitates a cushion of security while connecting the Root and the Leaf tiers. For the Enterprise CA:

- **Recommended hashing algorithm:** SHA-256
- **Considerations:** Here, a balanced approach is ideal, marrying security with compatibility. SHA-256 stands as a recommended choice, offering high security while maintaining broad compatibility.
- **Compatibility:** Considering that Enterprise CAs interact more frequently with end-entity certificates, compatibility with a wide array of systems, including legacy systems, becomes crucial to avoid disruptions in daily operations.

### Leaf certificates (End-entity certificates)

At the bottom tier, where the end-entity or leaf certificates operate, the focus shifts significantly towards compatibility without compromising on security. For the Leaf Certificates:

- **Recommended hashing algorithm:** SHA-256 or SHA-1 (in constrained environments)
- **Considerations:** While SHA-256 remains a recommended choice, in environments constrained by older systems, SHA-1 might still be in use, albeit with a known lesser degree of security.
- **Compatibility:** Ensuring broad compatibility is pivotal at this tier to facilitate seamless operations. While SHA-1 offers broader compatibility, especially with legacy systems, it should be employed judiciously, acknowledging its vulnerabilities and phasing it out in favor of more secure alternatives as soon as practicable.

By aligning the hashing algorithm strategy with the operational dynamics at each tier, organizations can build a robust Microsoft PKI, characterized by a fine synergy of security and compatibility. It is a continuous endeavor, evolving with the advancements in cryptographic research, to foster an environment where security meets functionality seamlessly.

## Conclusion

In a Microsoft PKI hierarchy, selecting the appropriate hashing algorithm at each level is critical to ensuring a secure, yet compatible infrastructure. While the Root CA can afford to prioritize high-security algorithms, the lower tiers should harmonize security with broad compatibility to facilitate smooth operations. In essence, a judicious selection of hashing

algorithms, which acknowledges both the security requisites and the compatibility needs, stands central to building a resilient and efficient PKI, paving the path for secure and seamless digital communications.