Public keys: 53 years of evolution survey

Andrei Titu

January 9, 2024

Abstract

Embraced by both small enterprises and large corporations alike, public key infrastructure (PKI) serves as a cybersecurity technique for verifying, validating, and safeguarding digital information. Authentication, validation, and authorization of identities play a pivotal role in the realm of cybersecurity for all types of organizations. Originally stemming from the British intelligence community in the early 1970s, employing PKI for authentication and encryption has been in practical use within commercial contexts for more than two decades. However, choosing or designing the suitable PKI remains an unsolved problem as there simply isn't a one-fits-all solution. One must inquire whether there exists a one-fits-most solution nevertheless.

1 Introduction

The implications of public key cryptography extend far beyond individual transactions or secure email exchanges. Its integration into the fabric of digital infrastructure has ushered in transformative shifts in the realms of e-commerce, secure communication protocols, and the very structure of our digital identities. It underpins the security and authenticity of websites we visit daily, safeguarding our personal information from prying eyes. Public key infrastructure (PKI), which manages the creation, distribution, and revocation of digital certificates, has emerged as a crucial component of digital trust. Moreover, public key cryptography plays a pivotal role in the development of blockchain technology, enabling the creation of decentralized, tamper-proof ledgers underpinning cryptocurrencies and smart contracts.

The literature survey presented in this essay will offer a panoramic view of the academic, technological, and practical landscape surrounding public key cryptography and its integration into digital infrastructure. By synthesizing key insights, trends, and debates, this paper aims to provide a comprehensive understanding of the current state of knowledge in this dynamic field. Additionally, this reading will delve into the challenges and open questions, as well as the latest developments and future prospects, highlighting the ever-evolving nature of public key cryptography in a world where secure communication and data protection have become indispensable.

2 Background

Before diving deeper into security solutions it is important to understand the actual problems which need solving and define a lens through which alternatives will be looked at. This lens is an evaluation model which will start from the requirements which need to be fulfilled, will look into the extent to which these requirements are fulfilled and the incurred costs for achieving these levels. Most readers will be interested to optimise the balance between increasing the level of guarantees for certain requirements while keeping the cost within some boundaries. In order to systematically introduce this analytic view, the current section will present a very brief history of the space and introduce the required security concepts.

2.1 History

The history of cryptography is a long and fascinating journey, starting well before the Enigma machine and continuing to the present day. Here's a chronological overview of some key developments in the history of cryptography:

World War I: Both sides in World War I used various cryptographic techniques to protect their communications. Notably, the Germans used the ADFGVX cipher, which was broken by the French.

Enigma Machine (20th century): Developed by the Germans during World War II, the Enigma machine was a complex electromechanical cipher machine. Allied codebreakers, including British mathematician Alan Turing, successfully cracked the Enigma code, a significant turning point in the war.

Public Key Cryptography (1970s): It was publicly thought that Merkle, Hellman and Diffie were the first people to develop public key cryptography until 1997, when the British Government declassified work done in the early 1970s by James Ellis, Clifford Cox and Malcolm Williamson. Ellis, Cox and Williamson had come up with the first public-key encryption scheme between 1969 and 1973, but their work was classified for two decades. It was conducted under the Government Communication Headquarters (GCHQ), a UK intelligence agency. Their discovery was actually the RSA algorithm, so Diffie, Hellman and Merkle were still the first to develop the Diffie-Hellman key exchange, but no longer the first inventors of public-key cryptography. This laid the foundation for modern encryption methods like ECC.

Data Encryption Standard (DES): In the 1970s, the U.S. National Institute of Standards and Technology (NIST) introduced the Data Encryption Standard, a widely used symmetric-key encryption algorithm.

Rise of Internet Cryptography: With the growth of the internet, encryption became crucial for securing online communications. Protocols like SSL/TLS and cryptographic algorithms like RSA and AES were developed to ensure data security online.

Advanced Encryption Standard (AES): AES became the successor to DES and is widely used for encrypting data today.

Elliptic Curve Cryptography (ECC): ECC is a popular asymmetric encryption technique used in modern cryptographic systems. It offers strong security with smaller key sizes.

Quantum Cryptography: As quantum computing technology advances, quantum-resistant cryptography is becoming an area of active research to protect against potential threats posed by quantum computers to existing encryption methods.

Blockchain and Cryptocurrencies: Technologies like blockchain and cryptocurrencies such as Bitcoin rely heavily on cryptographic principles to provide security and enable decentralized transactions.

Post-Quantum Cryptography: Researchers are actively working on post-quantum cryptographic algorithms that can withstand the potential threat of quantum computers. NIST is leading standardization efforts in this area.

2.2 Public Key Cryptography (PKC)

Public key cryptography, also known as asymmetric cryptography, is a cryptographic system that uses a pair of keys: a public key and a private key, to secure digital communication and data. Each key in the pair has a specific role:

Public Key: This key is intended to be shared openly and is used for encryption. Anyone can use the public key to encrypt a message or data, but only the holder of the corresponding private key can decrypt it. Public keys are used for confidentiality and data protection.

Private Key: The private key is kept secret and known only to the owner. It is used for decryption and digital signature generation. When someone receives an encrypted message or digital signature created with the public key, they use their private key to decrypt it or verify the signature's authenticity.

Real world problems which are currently solved with PKC: symmetric key exchange, secure communication, SSL/TLS connections, S/MIME encrypted email, secret management, access control, secure data storage, code signing, document sharing, enforcing regulations and compliance, secure remote access (via SSH keys), Bitcoin etc.

2.3 Security requirements

Nowadays, in the case of most projects the security requirements which need to be fulfilled stem out of five root concepts: authentication, integrity, confidentiality, non-repudiation and, in some cases, authorization.

Confidentiality: Confidentiality ensures that information remains private between the parties for which its exposure was intended. Systems leverage both public (asymmetric) and secret (symmetric) cryptography for confidentiality. While public key cryptography is less efficient for large data, it is suitable for encrypting small data objects, such as symmetric encryption keys. Secret key cryptography is often used in PKIs for bulk data encryption, providing actual confidentiality.

Integrity: This concept ensures that data cannot be corrupted or modified, and the integrity of transactions remains intact. For this task, PKIs use public key cryptography along with hashing algorithms (e.g., SHA-1 or MD5). For example, a Message Authentication Code (MAC) can be generated using secret key cryptography in a PKI environment. However, using symmetric cryptographic systems for integrity in a PKI may not scale well, so public key cryptography, combined with hashing, is typically more efficient.

Authentication: Authentication involves verifying the identities of entities using public key certificates and digital signatures. As a result, plain PKC can't guarantee the authenticity of a sender after a certain extent, but PKIs excel at it.

Non-Repudiation: This concept ensures that data cannot be denied or transactions disavowed. It is achieved through digital signatures in public key cryptography. Non-repudiation is a crucial security service in e-commerce, legal, and contractual negotiations. It is a by-product of using public key cryptography. When data is cryptographically signed with a private key, anyone with the corresponding public key can verify that only the key's owner could have signed the data. This underscores the importance of securely protecting private keys used for digital signatures.

Public key cryptography ensures all of the above to some degree, but by itself is vulnerable to a range of attacks, particularly with regard to authenticity. The hard question is: how can someone know for sure that the public key they are encrypting their precious data with really belongs to the intended receiver? For gaining insights into what a secure system needs to watch out for, some studied adversarial models will be reviewed.

2.4 Adversarial models

Equally notable is the existence and perpetual development of attack models looking to exploit insufficient guarantees in one of the 5 concepts from above.

Confidentiality

Eavesdropping: Attackers intercept and monitor data transmission, attempting to decrypt or gain access to sensitive information.

Brute Force Attacks: Attackers attempt to break encryption by trying all possible decryption keys, particularly with symmetric encryption.

Integrity

Data Tampering: Attackers modify data during transmission, potentially altering the content of messages, documents, or transactions.

Replay Attacks: Attackers capture legitimate data and replay it, causing actions to be performed multiple times, potentially leading to unauthorized operations.

Authentication:

Man-in-the-Middle (MitM) Attacks: Attackers intercept communication between two parties, potentially altering or eavesdropping on the messages, while both parties believe they are securely communicating with each other. This is usually the most encountered and feared attack when it comes to large ecosystems with a lot of communicating actors.

To guarantee the authenticity of a public key, the traditional PKC uses a certificate that is a digitally signed statement binding an entity and his public key. Since the amount of keys to manage and operations to perform seem to multiply, here is where the need for some designated architecture come in. This architecture is what's been refferred to as public key infrastructure.

Identity Theft: Attackers impersonate a legitimate entity by stealing or compromising private keys, allowing them to masquerade as the entity.

Non-repudiation:

Key Compromise: If a private key is compromised, an attacker may falsely sign data, leading to non-repudiation failures.

Forgery: Attackers may create counterfeit digital signatures or manipulate digital signatures, leading to false non-repudiation claims.

2.5 Required Infrastructure

A few suplimentary measures can go a long way in raising security guarantees in each of the aforementioned security services. A few of these measures are: digital certificates, multi-factor authentication (MFA), role-based access control, auditing, security information and event management (SIEM), zero trust security model, intrusion detection and prevention systems (IDPS), vulnerability scanning etc. All these various endeavours will require infrastructure so they can exist. On the other hand, the clear need for public key infrastructure supporting certificates is considered the main difficulty in the deployment and management of traditional PKC.

However, it is important to understand that a such infrastructure is not by itself an authentication, authorization, auditing, privacy, or integrity mechanism. Rather, a it is mearly an enabling factor that supports these various business and technical needs. For example, a PKI does not infer trust by itself, but requires the establishment of a trust base, on which the PKI can rely. This requirement means that the basis of trust must be established on a personal, business, or other level, before it can be accepted by the PKI. The problem of firstly authenticating a entity will remain a hard one, and must be tackled in isolation. This problem will not make the subject of this paper.

2.6 Adoption bottlenecks

In addition to the problems defined above, several other requirements are generally involved in deciding whether a solution is worth using. These extra requirements can be considered the reasons why no standard solution exists, and why mass adoption is halted.

Usability:

Secret management:

3 Traditional PKI

A PKI enables the establishment of a trust hierarchy. This is one of the primary principles of a PKI. In Internet-based e-commerce, formal trust mechanisms must exist to provide risk management controls. The concept of trust, relative to a PKI, can be explained by the role of the CA. In the Internet environment, entities unknown to each other do not have sufficient trust established between them to perform business, contractual, legal, or other types of transactions. The implementation of a PKI using a CA provides this trust.

A Public Key Infrastructure (PKI) is a comprehensive system of hardware, software, policies, standards, and practices that work together to provide a framework for secure communications and authentication. It is used to manage digital keys and certificates. PKIs are commonly used for tasks like securing email communications, establishing secure connections over the internet (e.g., SSL/TLS), and for digital signatures. Here are the key components of a PKI:

Certificate Authority (CA):

Root CAs are the highest-level CAs in the hierarchy. It issues and signs intermediate CAs' certificates. Intermediate CA: These CAs are subordinate to the root CA and issue certificates to end entities. They can also sign other intermediate CA certificates. End Entities: These are the users, devices, or servers that require certificates issued by the PKI to authenticate themselves or secure communications.

Digital Certificates: Certificates bind a public key to the entity's identity. They include information such as the public key, the entity's name, the digital signature of the issuing CA, and the certificate's expiration date.

Public and Private Key Pairs: End entities generate and keep private keys secure. Public keys are shared widely and included in certificates.

Registration Authority (RA): The RA verifies the identity of entities before they are issued a certificate. It acts as an interface between the end entity and the CA.

Certificate Repository: A secure location for storing issued certificates, making them available for validation and lookup.

Certificate Revocation Lists (CRLs): Lists of certificates that have been revoked by the CA before their expiration date. Clients and applications can use CRLs to check the status of certificates.

Certificate Policy and Practice Statements (CP/CPS): These documents outline the PKI's operational and security practices, including how certificates are issued, managed, and revoked.

Key Management System: A system for securely generating, storing, and managing cryptographic keys. It should ensure the security of private keys.

Security Protocols and Standards: The PKI should adhere to industry-standard security protocols and standards, such as X.509 for certificate formats, TLS for secure communications, and

OCSP for real-time certificate status checks.

Secure Hardware: The CA's private key should be stored securely, often in hardware security modules (HSMs), to protect against theft or tampering.

Auditing and Monitoring: Regular monitoring of the PKI infrastructure to detect and respond to security incidents or anomalies.

Backup and Recovery: Procedures for backup and recovery of CA keys and data in case of hardware failure or disaster.

Cross-Certification: When operating in a distributed environment, PKIs may need to establish trust with external PKIs through cross-certification.

Compliance and Legal Requirements: Compliance with relevant laws and regulations, including data protection and privacy laws, may be required. Additionally, adherence to any industry-specific standards or requirements is essential.

User Education and Training: Training and education programs for users to understand how to use certificates and secure communication channels.

Scalability and Redundancy: The PKI should be designed to scale as the organization grows, and it should incorporate redundancy for high availability.

Lifecycle Management: This involves the management of certificate lifecycles, including issuance, renewal, and revocation.

Secure Communication: All communication within the PKI should be secure, including the transmission of certificates, CRLs, and certificate revocation information.

Policy Enforcement: Ensure that policies regarding certificate issuance, usage, and revocation are consistently enforced.

Diffie-Hellman, RSA or elliptic curves algorithms such as Ed25519 are generally used for generating public key pairs, and X.509 is the de-facto standard for building certificates.

How does a clerk in Denmark determine if a driver's license, temporary or otherwise, is legitimate if it was issued in Japan? How do they determine if they should trust the credentials presented? What mechanism do they use to make that determination? How did the original authority, which issues the credentials, determine the identity of the requestor? Do you trust the original authority to perform its identification tests properly? These are all fundamental issues that a PKI must contend with.

4 Decentralized PKI

The issues and limitations of centralized Public Key Infrastructures (PKIs), such as those relying on Certificate Authorities (CAs), stem from their dependence on a central trusted entity. Within this framework, individuals do not have the autonomy to choose their online identity; instead, it is determined by trusted third parties, including CAs, which can be either private entities or governmental bodies.

This vulnerability poses a significant problem as it creates opportunities for attackers to execute Man-in-the-Middle (MITM) attacks. Currently, there are approximately 3,675 trusted CAs globally, making them attractive targets for cybercriminals. Each of these entities possesses the capability to establish alternative identities on behalf of users.

Various forms of MITM attacks, such as ARP spoofing, IP spoofing, DNS spoofing, HTTPS spoofing, and Man in the Browser (MITB), have been identified. Incidents have demonstrated that excessive reliance on CAs increases the risk of MITM attacks.

In practical terms, attackers can deceive CAs into believing they are someone else or compromise the CA to obtain a rogue certificate. The 2011 DigiNotar incident is an example where fraudulent certificates were issued due to an attack on the Dutch certificate authority company.

Another incident in 2017 involved hackers gaining control of a Brazilian bank's DNS server and manipulating a CA into issuing a valid certificate to them.

The Internet Engineering Task Force (IETF), responsible for Web PKI, has acknowledged the existing problems in a memo. Additionally, a group of researchers, including Vitalik Buterin, associated with Rebooting the Web of Trust, has outlined weaknesses in the current Web PKI implementation. Both highlight the need for addressing the challenges posed by the outdated PKI design.

The outdated nature of centralized PKI systems introduces significant security risks, as a single point of failure could compromise any encrypted online communication. These systems struggle to adapt to the dynamic digital landscape, emphasizing the urgent need for a more robust and decentralized approach to PKIs in the modern world.

4.1 Web of Trust

The Web of Trust (WoT) serves as an alternative to the centralized Certificate Authority (CA)

model. The concept of WoT was first introduced together with Pretty Good Privacy (PGP), an encryption program developed by Phil Zimmermann, which is a decentralized trust system that was created when blockchain didn't exist. Unlike the CA model, WoT operates without designated certificate authorities. In WoT, any system user has the ability to sign the public key certificates of other users, and the design encourages multiple signatures on keys. In the event that a signer is compromised and their key is revoked, the impact on the trust network remains constrained. PGP, as an example of a WoT model, relies on trustworthy users who mutually sign each other, managing private and public key rings.

Each user has the option to select their preferred set of trusted users. Rather than relying on a central authority universally trusted by everyone, users validate and endorse each other's keys, forming a network of individual public keys where each connection is represented by a signature. The collection of all keys known to a key owner is commonly known as a Key Ring.

Let's consider a scenario where Alice is friends with Bob. In this trust network, Alice signs Bob's public key. If someone receives Bob's certificate, they can see that Alice vouches for its authenticity. Say Carol wants to send a message to Bob but doesn't know whether Bob's public key that they received is truly Bob's. However, Carol sees that Alice signed Bob's key, so now the question becomes: can Carol know for sure Alice's key is truly Alice's. Recursively this process is repeated until Carol finds someone who they actually trust. This is nice because suddenly, each 2 parties can implement their own way of initial authentication and requirements for establishing trust. Unfortunately, there is another dimension to this whole process which doesn't simplify the trust inference: how good is Alice at determining Bob's authenticity? What if Alice didn't actually perform any checks before signing Bob's key and now they endanger everyone who trusts them? This can be solved by associating degrees of trust to each party by each party, representing how likely a certain node is to vouch truly. These approaches have generally worked in practice but guaranteeing real correctness and preserving privacy while doing so remains a limitation which modern block-chain approaches promise to solve through algorithmic consensus and smart contracts.

4.2 Self-sovereign identity

Self-sovereign identity (SSI) is a concept in identity management and digital identity that empowers individuals with control over their own identity information. In that sense it can be viewed as an extension of WoT. The fundamental idea is to shift the control and ownership of identity from governments or corporations, to the individuals themselves. To put it more specifically SSI is concerned with:

User Control: Individuals have complete control over their personal information, including identity attributes, credentials, and other relevant data.

Decentralization: SSI systems often leverage decentralized technologies, such as distributed ledger technology. This helps enhance security, privacy, and resilience compared to using CAs.

Portability: SSI allows users to carry their digital identity with them across various contexts and services. Users can choose when and with whom they share specific pieces of information.

Interoperability: SSI systems aim for interoperability, allowing different platforms and services to recognize and accept the same set of credentials or identity information.

Verifiability: The information stored in a self-sovereign identity system is cryptographically secure and can be independently verified by third parties without revealing the actual data.

Consent-based Sharing: Users must provide explicit consent before sharing any part of their identity information. This aligns with the principle of user consent and privacy.

The goal of self-sovereign identity is to empower individuals, enhance privacy, and provide a more user-centric, secure, and flexible approach to identity management in the digital realm. Various SSI projects and standards, including W3C's Verifiable Credentials and Decentralized Identifiers (DIDs), aim to establish a framework for implementing self-sovereign identity solutions. The following section will look at a couple of implementations of SSI.

4.3 Blockchain-enabled DPKI

Decentralized PKI approaches employ block-chain technology to disperse a CA's responsibility across a network. Block-chain steps into the shoes of the traditional trusted third parties or better said, in a DPKI system leveraging blockchain technology, third parties take on roles such as miners or validators. Trust is established and preserved through consensus protocols such as Proof-of-Work or Proof-of-stake. What is essentially happening is that instead of relying on certificates emitted by a CA to map identities to public keys, this mapping is kept up-to-date on a distributed ledger, which is globally accessible and transparent. No identity to key mapping would appear on

the ledger unless it passed the checks defined by an algorithmic consensus mechanism. Revocation or other key management operations are ensured by smart contracts.

For example, EBSI (European Blockchain Services Infrastructure) is used for verifying institutional accreditaions by storing the public keys of issuers and verifiers, as well as the ones of individual looking to get accredited, using data structures called DIDs. Verifiable credentials (VCs) are issued by issuers, who sign the claims and send them to the individual to be stored in an encrpyted wallet. The individual sends the document to a verifier who checks that the claims were signed by the issuer and grants the individual with the requested resource. All identity checks are done via the globally trusted EBSI block-chain. The World Wide Web Consortium (W3C) has developed standards such as the Decentralized Identifier (DID) specification and the Verifiable Credentials Data Model (VC Data Model) that provide a common framework for implementing DIDs and verifiable credentials.

5 NoPKI

Many would argue that the deployment of any sort of additional resources or infrastructure constitutes the primary adoption bottleneck. Several algorithms have been designed with the goal of minimizing the amount of involved parties, or the required software and hardware.

5.1 Identity-based PKC

Identity-Based Encryption (IBE) is a type of public-key cryptography in which a user's identity information, such as an email address, username or UID, is used as a public key. In traditional public-key cryptography, users obtain public keys from a centralized authority or a public key infrastructure, whereas with IBE the sender itself can derive the public key of the receiver using said receiver's unique identifier.

Identity information is used to generate the public key, and the corresponding private key is generated by a trusted third party (TTP) known as the "Private Key Generator" (PKG). The PKG is responsible for generating private keys corresponding to the public keys, as well as distributing them through a secure channel. In the context of IBE the PKG TTP is a necessary component, as the security guarantees of the generated private keys are stemming from a master key (private key) owned only by the PKG. Anyone can encrypt a message using the recipient's identity-based public key. The encrypted message can only be decrypted by the intended recipient, who possesses the corresponding private key generated by the PKG. When a user needs their private key, they must contact the PKG, which verifies the user's identity and then provides the private key. Retrieving the private key usually only has to happen once, unless key rotation is required.

Some advantages of IBE are:

Simplified Key Management: Since public keys are derived from user identities, there's no need for a separate PKI to manage and distribute public keys.

Flexibility: Users can use their existing identities (such as email addresses) as public keys, making it convenient for identity-based access control.

Rotation: If a user's private key is compromised or needs to be revoked, the PKG can recompute a new private key associated with the user's identity.

However, there are also challenges and concerns with IBE:

The inherent Key Escrow problem: the reliance on a trusted central authority (the PKG) that knows everyone's private keys causes the entire system to have one single point of failure.

and potential security risks associated with key extraction processes. A secure communication channel is still necessary for distributing private keys.

Key rotation can also be challenging due to the inherent nature of the system and its reliance on users' identities as public keys. In traditional public key cryptography, key revocation is relatively straightforward – if a user's private key is compromised or needs to be revoked, the corresponding public key certificate can be added to a Certificate Revocation List (CRL), and relying parties can check this list to determine if a key is still valid. In ID-PKC, there is no direct mapping from a user's identity to a fixed public key. The public key is dynamically generated based on the user's identity and some system parameters. This lack of a fixed public key makes it challenging to maintain a centralized revocation list that associates specific public keys with users.

ID-PKC systems may be susceptible to offline attacks, where an attacker attempts to derive a user's private key from previously collected information, even after the key has been revoked. This is particularly challenging when users' identities are tied to long-lived attributes (e.g., email addresses).

5.2 Certificate-less PKC

Certificate-less Public Key Cryptography (CL-PKC) is a cryptographic paradigm that aims to combine the simplicity of identity-based cryptography with the security benefits of traditional public key cryptography to resolve with inherent key escrow problem. In CL-PKC, users' public keys are derived from their identities, but unlike identity-based cryptography, the generation of private keys involves collaboration between the user and a trusted third party called the Key Generation Center (KGC).

Users contribute a secret to the process of generating their full private key. This user-specific secret, combined with the partial private key from the KGC, is used to compute the full private key, while it's also being used to derive the public key in combination to some user identifying information. In this way, the public keys still need to be retrieved by a sender, similarly to how it's done when using traditional PKI, but certificates are no longer required to bind the public key to the user.

Similarly to the PKG, the KGC generates and holds a master key that allows it to compute partial private keys for users. However, this master key alone is insufficient to compute the full private key. This is how this solution addresses the key escrow concern. Also, if the partial private keys or, alternatively, user's secrets get leaked, this is still not sufficient information to derive the full private key with ease. CL-PKC removes the extra work for establishing secure communication channels between end users and the TTP.

However, reference has proved that a certificate-less cryptographic system exists iff an identity-based cryptographic system exists, deeming the two approaches equivalent, in terms of security guarantees, under the assumption that the CL-PKC implementation is based on Boneh-Franklin identity-based encryption.

A disadvantage that certificate-less cryptography doesn't overcome is the hardship in revoking and rotating a public key pair to the users.

5.3 Certificate-based PKC

The concept of certificate-based Public Key Cryptography (CB-PKC) was introduced to tackle the challenge of public key revocation. In this model, a certificate serves a dual purpose, functioning both as a partial private key and a traditional public key certificate. When Bob wishes to decrypt a ciphertext sent by Alice, he requires both his private key and an updated certificate from the Certificate Authority (CA). As a result, participants in a certificate-based PKC system do not need to acquire real-time information about the status of certificates. This approach simplifies the issue of public key revocation, eliminating the necessity for infrastructures such as Certificate Revocation Lists (CRL) [13] and Online Certificate Status Protocol (OCSP) [14]. In a certificate-based PKC system, challenges like private key escrow are nonexistent because the CA remains unaware of users' private keys. Furthermore, there is no problem of secret key distribution since the CA's certificate does not require confidentiality.

In certificate-based encryption, the TTP component is represented by a CA who owns the generation of the master key and the deterministic algorithm that takes uses as input a user's id, a time period t, the user's public key and the master key, to return the user id's certificate for the time period t. A certificate is given to a user id by the CA not necessarily through a secure channel, since the certificate doesn't contain any sensitive information. As the CA only deals with partial private keys and the user is responsible for generating their own public-key pair, the key escrow of the user's private key is not inherent.

Again, this method was proven equivalent with the other 2 presented models, and it's still classified as an identity-based method.

6 Password Authentication

Passwords have been the most adopted authentication method due to usability and ease of understanding how knowledge of a secret value can prove ones identity. However classic password authentication has several drawbacks including susceptibility to knowledge leaking through phishing, or insecure communication channels. The fact that a user instead of an algorithm is responsible for coming up with the authentication token, makes it insecure to find out through brute-force attacks on dictionary attacks. Also, managing passwords is usually left to the user's latitude and bad habits such as having a multi-purpose password or storing it insecurely with the purpose of not forgetting it are more spread out than using password manager solutions. This section will look at improved password-based authentication methods that aim to conserve usability while elevating security guarantees.

6.1 PAKE protocols

Password-Authenticated Key Exchange (PAKE) protocols are cryptographic protocols designed to allow two parties to establish a shared secret

(a cryptographic key) over an insecure network based on the knowledge of a password. The primary goal of PAKE protocols is to provide secure key exchange even if an attacker is eavesdropping on the communication. PAKE protocols have a range of applications, including secure authentication and key establishment for secure communication. The parties involved (usually a client and a server) share a common password, which serves as the basis for authentication. Unlike traditional password-based authentication, PAKE protocols ensure that the actual password is not transmitted over the network. PAKE protocols often incorporate zero-knowledge proofs. Zero-knowledge proofs allow one party to prove knowledge of a secret (the password) to another party without revealing the actual secret. This ensures that even if the communication is intercepted, the password remains confidential. Once the parties have completed the authentication process, they derive a shared secret from the knowledge of the password. This shared secret can then be used as a basis for deriving cryptographic keys for secure communication. PAKE protocols involve secure computations that enable both parties to contribute to the generation of a shared secret without directly revealing sensitive information. Common cryptographic primitives such as commitment schemes and hash functions are used to achieve secure computation, depending on the actual implementation (SPEKE, SRP, OPAQUE, J-PAKE etc). These protocols are designed to resist offline dictionary attacks, where an attacker tries to guess the password by repeatedly attempting authentication. Even if an attacker intercepts the communication between the parties, they should not be able to derive the user's password or the shared secret with ease.

7 Passkeys

Passkeys are believed to represent the future of safeguarding account security and protecting our sensitive data, and their presence is undeniable. Google is currently working on Android 14 and APIs that will enable people to create and use passkeys inside Chrome and any other app that has added passkey support. Passkeys make it easier for everyone to use passwordless authentication across all of their devices. Perhaps more importantly, they're backed by influential technology companies including Apple, Google, Microsoft. By championing passkeys together, this group can raise awareness and, by extension, overall adoption around the world.

7.1 Passwordless and FIDO Alliance

The FIDO Alliance, or Fast Identity Online Alliance, is a global organization focused on developing open standards for strong, passwordless authentication. The alliance was founded in 2012 by several major technology companies with the aim of addressing the weaknesses and vulnerabilities associated with traditional password-based authentication methods. FIDO is now formed of some of the largest technology companies in the world including Apple, Google, and Microsoft.

The primary goal of the FIDO Alliance is to create a more secure and user-friendly authentication framework that reduces reliance on passwords and provides a simpler, yet stronger, means of verifying user identities. FIDO standards enable interoperability between various devices and technologies, promoting widespread adoption.

Numerous authenticator specifications have been defined by FIDO, including U2F, UAF, and CTAP. Universal 2nd Factor (U2F) stands out as one of the early specifications for WebAuthn authenticators, known for its straightforward implementation. However, development has shifted towards the Client To Authenticator Protocol (CTAP2), with the "2" in CTAP2 denoting its version, positioning it as a successor to U2F. Passkeys leverage an API called WebAuthn, or Web Authentication. The API was jointly developed by the FIDO Alliance, and the World Wide Web Consortium (W3C), a community that works together to develop new standards and guidelines for the web.

7.2 WebAuthn

WebAuthn, short for Web Authentication, is an API empowering website developers to facilitate a passwordless login experience on their websites and applications. This crucial software serves as the link between these platforms and the user's selected authenticator, and stays at the core of the "passkey" buzzword. It is developed by the World Wide Web Consortium (W3C) and the FIDO Alliance. The primary goal of WebAuthn is to provide a standardized way for websites to support strong authentication mechanisms.

The specification defines 2 participants: authenticators and relying parties.

Authenticators come in two primary forms:
Roaming authenticators: These are independent devices designed for portability, such as hardware security keys that can be easily carried.
Platform authenticators: These are integrated

into existing devices, like PCs or phones, streamlining the authentication process for users.

Whereas a relying party can be any back-end application which is responsible for keeping track of users' identities.

A WebAuthn Authenticator generates and stores a public key credential upon the request of a WebAuthn Relying Party, contingent on user consent. Subsequently, access to the public key credential is restricted to origins associated with that specific Relying Party. This limitation is enforced collaboratively by compliant User Agents and authenticators. Furthermore, privacy is upheld across Relying Parties, preventing them from detecting any properties or the existence of credentials scoped to other Relying Parties.

Relying Parties utilize the Web Authentication API in two distinct yet interrelated procedures involving a user. The first is Registration, where a public key credential is established on an authenticator and scoped to a Relying Party associated with the user's current account (which may already exist or be created at this time). The second is Authentication, during which the Relying Party receives an Authentication Assertion confirming the user's presence and consent for the previously registered public key credential. Functionally, the Web Authentication API incorporates a PublicKeyCredential, extending the Credential Management API [CREDENTIAL-MANAGEMENT-1], along with infrastructure enabling the use of these credentials through navigator.credentials.create() for Registration and navigator.credentials.get() for Authentication.

In a broader context, compliant authenticators safeguard public key credentials and collaborate with user agents to implement the Web Authentication API. Such authenticators can be implemented in software running on (a) a general-purpose computing device, (b) an on-device Secure Execution Environment, Trusted Platform Module (TPM), or a Secure Element (SE), or (c) off-device. Authenticators implemented on-device are referred to as platform authenticators, while those implemented off-device (roaming authenticators) can be accessed through transports like Universal Serial Bus (USB), Bluetooth Low Energy (BLE), or Near Field Communications (NFC).

Key features and concepts of WebAuthn include:

Replay attack resistance: WebAuthn is designed to be resistant to replay attacks. Even if a user unknowingly authenticates on a malicious website, the attacker won't be able to use the captured credentials elsewhere. This means that one user will have a separate set of passkeys for each relying party it communicates with.

Public Key Cryptography: WebAuthn relies on public-key cryptography for authentication. Instead of sharing secret information (like a password), the user's device generates a public-private key pair. The private key remains on the device, and the public key is registered with the online service.

Cross-Browser Compatibility: WebAuthn is designed to work across different web browsers and platforms, providing a consistent and interoperable authentication experience.

Privacy Considerations: WebAuthn is designed with privacy in mind. It minimizes the amount of information exchanged during authentication and allows users to control what information is shared.

This is what the standard entails: Imagine that a user visits a website that supports passkeys. First, the user creates an account and opts to secure it with a passkey instead of a traditional password. The website's server shares some information about the website and asks the user to confirm their authenticator. This could be the user's phone, tablet, PC, or a password manager that adheres to the WebAuthn standard. A passkey, comprising a public and private key pair, is then generated for that specific website. This process occurs locally, on the user's device. The public key is sent to the website's server for storage where it's mapped to this user's unique id, while the private key remains securely stored in the user's authenticator. The user never even gets to see or interact with their private key, as the authenticator abstracts all the heavy lifting away. All the user needs to do is to provide their fingerprint (or any form of biometric authentication the device supports) or device password, when prompted for it.

The next time the user signs in, the website will create a "challenge," similar to a puzzle. The user's authenticator will "sign" the challenge using the private key, then send the completed "signature" to the website. Finally, the website uses its copy of the user's public key to verify the signature's authenticity. And that's it! The user has signed in using their unique passkey.

WebAuthn has a history dating back to 2016, with the publication of the WebAuthn Level 1 standard as a W3C recommendation occurring three years later. Many web browsers, including Chrome, and various hardware security keys (roaming authenticators) already support the API.

Despite this, the standard has not yet reached widespread adoption. The majority of individuals still rely on traditional usernames and passwords

Authentication Solution	Can Johhny encrypt	Single point of failure	Self sovereign	Deployable	Rotation
(Traditional) PKI	X	X	X	X	V
WoT (PGP)	X	V	V	V	X
Blockchain-based	X	V	V	X	V
Identity-based	X	X	X	V	X
PAKE	V	V	X	V	V
Passkeys	V	V	X	V	V

Table 1: Solution evaluation table.

for their online accounts, and only a few websites currently offer a passwordless login experience.

To address this, major technology companies are collaborating on a solution known as passkeys, utilizing the WebAuthn standard. Passkeys provide a seamless and secure sign-in experience using existing devices (platform authenticators). While WebAuthn is already in use, passkeys have the potential to significantly increase its exposure and adoption due to their convenience, user-friendliness, and enhanced security features.

7.3 Benefits

8 Evaluation

9 Conclusion

References