The Failings of the Public Key Infrastructure
An Argument to Replace Modern Trust Mechanisms

By: Jacob P. Mohrbutter
University of Nebraska at Omaha
College of Information Science and Technology

The Failings of the Public Key Infrastructure
An Argument to Replace Modern Trust Mechanisms

Jacob P. Mohrbutter[1]

[1]University of Nebraska at Omaha

May 10, 2019

Authors Note

Jacob P. Mohrbutter: University of Nebraska at Omaha
College of Information Science and Technology.

Executive Summery

Our lives have changed drastically due to the influx of new technologies; The Internet allows an almost instantaneous flow of data on an intercontinental level. Undoubtedly this has benefited humanity on an unprecedented level; however, the risks one faces in the modern technological world has evolved along side the technological boom. These risks are not always clear to an end user. One assumes an invisible risk blindly when connecting to apparently secured servers; collectively humanity has embraced the idea that "HTTPS" automatically means safe, and "HTTP" means not safe. The little green padlock in the corner of a users browser ever present to reassure them of their apparent security. But, when the public key infrastructure, the mechanism that individuals rely on to generate this trust, is inherently broken on a fundamental level; is it a disservice to train the user base to trust that mechanism?, and, if the answer is no, what is the plan of recourse; does one continue to trust this mechanism blindly.

The public key infrastructure is implicitly flawed. The flaws that are discussed are inherent. There are multiple instances in which malware survived in the wild for long periods of time due to an attackers utilization of fraudulent but valid certificates: Stuxnet, Duqu, Flame, Shadowhammer. When is enough enough, is it not a travesty that such events could be allowed to occur for months or years at a time due to valid certificates signed by Trusted third parties, valid members of the public key infrastructure. The problem is in the ability for an entity to conduct audits of the public key infrastructure. Even with certificate transparency, a solution that was instated by Google, one must assume that a technical adversary is not manipulating the view one has of the log. Fragmented logs and the introduction of unnecessary entities has led to a system that is still vulnerable to attack.

The flaws that are inherent to the public key infrastructure may be irredeemably broken, but, a solution does exist. the public key infrastructure needs a mechanism that can be audited with ease. This document argues for a blockchain based solution, which utilizes a smart contract to manipulate a mutable contract. The ability for an attacker to game this mechanism would be exceedingly difficult, and due to the inherent nature of block-chain, auditing and monitoring the logs would become a simple task.

Contents

List of Figures

List of Tables

Introduction

The public key infrastructure (PKI) is the mechanism by which entities interacting on the internet derive trust. This document shall discuss the cryptographic security mechanisms underpinning that trust mechanism in such instances as asymmetric cryptography, hashing, signing, and certificates. Furthermore, it shall be shown that there exists circumstances in which an attacker can subvert these underlying mechanisms such as instances where malware was signed by valid code signing certificates, or certificate authorities were compromised, allowing certificates to be issued to unauthorized users. It will be shown that these failings are inherent to the underpinning mechanisms being utilized and thus cannot be fixed in any efficient manner. This shall be used as the basis for this document's proposed argument; an argument to replace the current PKI with a new mechanism which shall address the flaws that exist in the current trust mechanism. The solution this document shall primarily focus on is that of a decentralized public ledger; it shall be shown that such a solution would eliminate the inherent underpinning flaws while maintaining the cryptographic securities the current system provides.

Background

Cryptography

Cryptography is the application of mathematical transformations enacted upon data to ensure that the data cannot be recovered without some secret that is not known to an attacker. Furthermore, the secret (or "Key" as it is commonly referred to) should be computationally infeasible for an attacking party to produce. For the scope of this document, it is important for readers to be familiar with the cryptographic principles discussed in this subsection.

Cryptography can, in general, be reduced to the transformation seen in equation (1).

$$T : \{P, K\} \rightarrow C \tag{1}$$

T in this instance refers to a transformation that requires two inputs: plaintext denoted as P, and keying material, denoted as K. The transformation maps these inputs to cyphertext denoted C. The cyphertext can be considered secure as long as the keying material remains in the possession of the authorized party (Barker, 2016, p.1-4).

Asymmetric cryptography refers to a form of cryptography by which any party with access to the public key can apply the transformation that encrypts data:

$$T : \{P, Pub_K\} \rightarrow C \tag{2}$$

This is due to a "Trapdoor" function that allows the transformation seen in equation (2) however, the inverse transformation is not true.

$$T^{-1} : \{C, Pub_K\} \not\rightarrow P \tag{3}$$

This allows the public key to be dispersed to all entities, while ensuring no entity with the public key may apply an inverse transformation as seen in Equation (3). Furthermore, there should be no function F such that a malicious entity could derive the private key as seen in equation (4).

$$\nexists F : \{Pub_K\} \rightarrow Private_K \tag{4}$$

The party in possession of the private key would be able to retrieve the plaintext by applying the transformation *T'* as seen in Equation (5), thus ensuring the security of any correspondence from a client to a server as long as the client can verify the owner of the public/private key pair (Barker, 2016, p.23-27).

$$T' : \{C, Private_K\} \rightarrow P \tag{5}$$

Asymetric key cryptography is used by a server and client to establish a secure connection by exchanging keys to be used in Symmetric key cryptography. The public key infrastructure is the entity responsible for ensuring the validity of public keys. Without the public key infrastructure it would be trivial for a malicious entity to impersonate any other entity by producing their own private/public key pair and supplying a victim with a malicious key.

Hashing is a cryptographic transformation that must be understood as one moves forward throughout this document. Not only does Google's certificate transparency (CT) mechanism use hashing to construct signed Merkle Hash Trees, but so, too, does the proposed solution of utilizing Blockchain in the Model. As such it is vitally important for readers to be familiar with this concept.

A cryptographic hash is a transformation applied to map an arbitrary set of data $\{D\}_{L'}$ onto a finite sub-space.

$$T : \{\{D\}_{L'}\} \rightarrow \{F_C\} \tag{6}$$

For a hashing algorithm to be secure it must hold distinct properties; for instance, small changes in the input value must map to distinctly different values within the finite space. Furthermore, any hashing transformation must be resistant to collisions. In terms of a cryptographically secure hashing algorithm, a collision is an event by which two distinctly different data sets $\{D\}_{L',1}$ and $\{D\}_{L',2}$ undergo the same transformation as seen in equation (6) map to the same data set on some finite sub-space

$\{F_C\}$. Ideally, the probability of a collision event occurring should be $\approx 1 : 2^C - 1$, where C is the length of the hash in bits. As expressed in equation (7), there must exist no inverse transformation that takes a value from $\{F_C\}$ sub-space and maps it to an arbitrary data set. This is important since the existence of such a transformation would allow the discovery of collisions.

$$\nexists T^{-1} : \{F_C\} \to \{\{D\}_{L'}\} \tag{7}$$

Hashing is the only cryptographic transformation that requires no keying material. Other cryptographic transformations use keying material to ensure that only authorized parties can apply the inverse transformation. However, hashing is a one way transformation. There must be no transformation that recovers the originating data (Barker, 2016, p.19-20).

Signing refers to the cryptographic mechanism that ensures non-repudiation. This is essential to the functionality of the public key infrastructure (PKI). The transformation is asymmetric in nature. One may verify the integrity of a message M by applying the inverse transformation T^{-1}. This transformation can only be applied if the verifying party has the public key, denoted Pub_K, the signature, denoted Sig, and the message. This transformation can be seen in equation (8).

$$T^{-1} : \{Pub_K, Sig\} \to M'\textbf{; if } \{M = M'\}\textbf{then the signature is valid.} \tag{8}$$

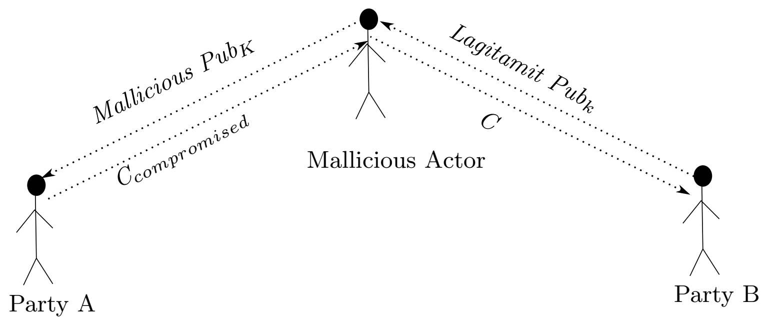
One cannot apply the inverse transformation without first having access to the private key. $T : \{Pub_K, M\} \not\to Sig$; furthermore, there must not exist a function such that one may acquire the private key given the public key $\nexists F : \{Pub_K\} \to Private_K$. An entity with access to the private keying material could apply the transformation in equation (9) and sign a message. This offers all entities with access to the public key a manner by which the origins of a message could be verified.

$$T : \{Private_K, M\} \to Sig \tag{9}$$

The Public Key Infrastructure

The public key infrastructure exists to prevent a form of attack known as an active man-in-the-middle attack. Such an attack breaks down the integrity of asymmetric cryptography as seen in Figure 1.
A malicious entity would intercept the public key. Instead of allowing the public key to continue to the intended victim, the attacker would send their own public key. At this point, they would become the "man in the middle," and all communications between the client and the server would be intercepted by the malicious entity who completed the key agreements (Ali Davanian, 2016).

Figure 1: Asymmetric Cryptography



This is an example man-in-the-middle attack. In this example, a malicious actor would be able to intercept the communications between Party A and Party B.

Certificates bind an entity's identity to their public key; assuming the private keys are securely stored by the entity for whom the certificate was issued, (presumably in a cryptographic manner), the mechanisms for authentication and secure transmission will remain cryptographically secure. This assumes the signing keys of the issuing certificate authority also remain secure. [**NOTE: this document will discuss instances where authorities failed to keep their signing keys secure**]. Certificates will also contain metadata often used to describe the scope of the certificates applicability and the issuance and expiration dates of the certificates. Certificates must also store metadata about the issuing certificate authority (Afshar, 2015, p.8-15).

Root Certificate Authorities are trusted third parties (TTPs) with public keys that come pre-installed by vendors of the operating systems and/or browsers. These TTPs act as the initial point of trust. Without these entities there would be no way for one to establish the initial trust on the Internet. As such, root certificate authorities are an integral member of the public key infrastructure (PKI). Because of the integral role that individual root certificate authorities hold, once they are installed in the root it often becomes nearly impossible to remove their keys from the list of trusted entities without removing a large part of the cryptographic trust mechanism that exist to secure client server communications on the internet (Symantec, n.d, p.2).

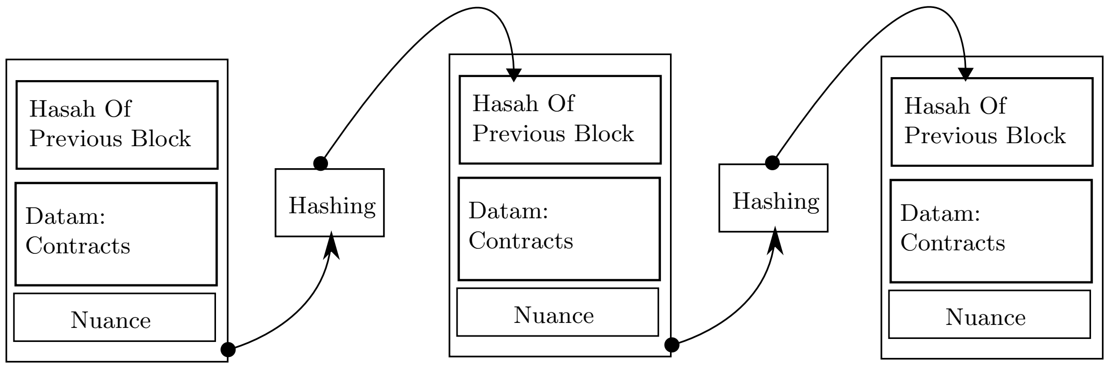
Subsidiary Certificate Authorities refer to those TTPs that are not installed in the root by a vendor. Instead, this entity would have a key signed by a valid TTP which would establish the trust mechanism. For example, envision a set of TTPs denotedE_n. If $\forall n, n \neq 0$; there exists an Entity E_{n-1} that has signed a public key for E_n enabling said entity to sign certificates. Then all entities are entrusted with the same powers and obligations as the root certificate authority E_0. There are no limits as to the number of subsidiary authorities a given certificate authority can sign.

This makes it difficult for one to determine the sheer magnitude of the public key infrastructure (Richard D. Kuhn, 2001, p.17).

Blockchain Mechanics

This document shall pose an argument that utilizes the fundamental properties of blockchain to address the serious security flaws that are inherent to the mechanism currently being utilized by the public key infrastructure; as such, one should take the time to become familiar with the mechanisms that enable blockchain to function in a secure and trusted manner. The main advantage that blockchain offers is a log that is both tamper resistant and tamper evident. The chain itself consists of blocks that are cryptographically linked together in a manner that protects the blocks that have already been added to the chain. Consider Figure 2, where each block is treated as an independent datum structure. This structure can in general be broken down into three fields: the cryptographic hash of the preceding block, the transaction or data that shall be added to the chain (in the case of the public key infrastructure, the datum fields are autonomous code segments known as contracts), and a cryptographic nuance (a random seed value that effects the hash value in an arbitrary manner. In PoW mechanisms, the nuance causes a number of bits at the beginning of the hash to be 0. It is computationally hard to brute force this nuance, yet for one with the nuance, calculating the resulting hash becomes trivial).

Figure 2: blockchain



Another advantage of blockchain is in the publicly visible nature of this technology, any party may access the log without compromising the implicit security mechanism that prevents a malicious entity from compromising the validity of that ledger; this allows for a time series ledger that can easily be audited by any individual at any point in time. These properties are critical to the functionality of the public key infrastructure, but there are additional properties that must be considered. These properties will be discussed in the section entitled **The Public Key Infrastructure and blockchain Solution**.

Consensus Mechanism refers to the mechanisms by which entries are added to the ledger; Blockchain is decentralized, meaning, there is no controlling entity/entities to manipulate or add entries to the ledger. Instead multiple peering parties would agree on the next block to be added to the public ledger. there are many types of consensus mechanisms, but for the scope of this document, the discussion will be limited to the three distinct mechanisms seen in Table 1. Consensus mechanisms ensure that peering parties gossip with each-other, and this ensures that all peers in the network share the same view of the log, and false logs become easier to detect.

Table 1: Consensus Mechanisms

Consensus Model	Description	Ledger Forks	resource intensive
PBFT	Practical Byzantine Fault Tolerance: This is a consensus mechanism based that remains cryptographically secure as long as 2/3 of the nodes connected to the mining network are controlled by honest nodes, but this also means that an entity could connect several false nodes to the network and manipulate the consensus mechanism (Dylan Yaga & Scarfone, 2018, p.22)	No	Minimal.
PoW	Proof of Work is a consensus mechanism that does not rely on individual nodes in the way that PBFT does; instead miners compete to complete a computationaly intensive cryptographic task. The task, once completed is trivial to verify. The essential principle behind PoW is one CPU one vote (Dylan Yaga & Scarfone, 2018, p.19)	Yes	Very Computationally Intensive.
Hybrid Mechanisms	Combine Consensus mechanisms such as PBFT and POW in an attempt to create a mechanism that is not suseptable to ledger forks while at the same time being computationally intense making it tougher to manipulate the PBFT protocol.	No	Very Computationally intensive

Mathematically provable Continuity of blockchain: Each block of data in a blockchain is mathematically dependent on the previous block in such a way that modifying the chain is infeasible. There are several mechanisms that may be utilized to accomplish this task, and the mechanism is dependent on the consensus mechanism being utilized. The properties unique to our use case would dictate the need for a Hybrid Consensus Mechanism. This, in turn, would require both a PBFT consensus mechanism to maintain a ledger that is free of forks while still maintaining the ”one CPU one vote” property inherent to the POW mechanism.

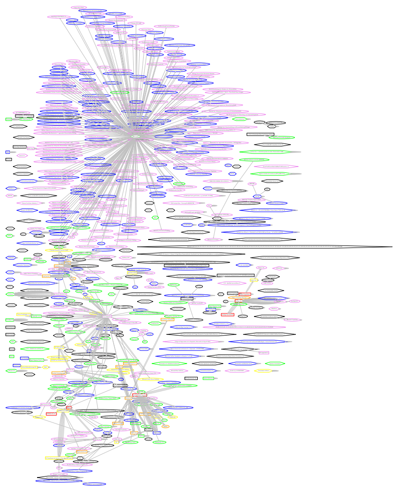
Contracts represent segments of code that are added to the blockchain in the datum field, and there contents are executed when a certain condition is met. Once the contract is added to the blockchain, it cannot be modified. Once the appropriate condition is met, the code will be executed. As we shall see through the explanation of the certledger protocol, this code execution provides a mechanism for certificate expiration, revocation, and renewal. These mechanisms will remain secure and a record of all events will be shared in real time across all members of the public key infrastructure (Dylan Yaga & Scarfone, 2018, p.32-p.33).

Problems With The Current Public Key Infrastructure

There are a number of security flaws that must be addressed in the modern Public Key Infrastructure. Some of the flaws are inherent to the public key infrastructure and others are obscured by the complexity of the system itself. This document shall address both inherent flaws to the system itself, and the flaws that arise due to the overwhelming complexity of the public key infrastructure. This section shall focus on three specific points: malware samples signed by valid code signing certificates, certificate authorities whom had bad security practices or otherwise were compromised by an attacker, and corrupt entities who issued malicious certificates and abused their power.

Compromised Certificates Found in Malware Samples

Figure 3: Color Map Of CAs



This is the color_map_of_cas.pdf that was create by the Electronic Frontier Foundation, (EFF, 2011) this raster vector file can be scaled to any magnitude.

Malware samples represent a large problem to modern society. A primary heuristic detection mechanism for an anti-virus is to check for a valid code signing certificate. For an adversary capable of signing a Malware sample, bypassing heuristic detection mechanisms becomes a trivial task. The only solution is to conduct thorough audits against issued certificates; here lies the inherent flaw of the public key infrastructure. Consider Figure 3. This map was generated by the Electronic Frontier Foundation (EFF) in 2010-2011. They discovered a total of 124 root certificate authorities and 651 entities that were capable of signing certificates. The system has only increased in complexity since that time; how is the public key infrastructure audited at this point in time? What mechanism is there to quickly detect and respond to such threats? The answer is that one assumes the inherent risk blindly. The mechanisms that enable the public key infrastructures functionality are the very mechanisms that obfuscate the existence of malicious certificates. This section shall address instances in which sophisticated adversaries were able to introduce signed malware samples into the wild. One should not consider the existence of malware signed by certificates a failing of the mechanism. Instead, one should consider the length of a signed malware samples life-cycle as the inherent flaw; this flaw arises due to the lack of visibility within the PKI (Peter Eckersley, 2010b, p.18).

Stuxnet represents one of the most complex malware samples ever written. The sample discovered in 2011 had unique properties. For instance, Stuxnet was searching for a specific set of Programmable Logic Controllers (PLCs) operating inside of a specifically targeted industrial facility. This attack represented the first instance of malware being used in a surgical attack; Stuxnet was an extremely evasive sample, partially due to the fact that it was signed as being a valid Realtec device driver. This digital certificate was valid from January 25, 2010 to July 22, 2010 (Nicholas Falliere, 2011).

Flame was a Malware sample that, with the aid of a valid certificate, could conduct an active man-in-the-middle attack against Microsoft's Windows updates. When a victim made a request to the update server they would be infected with Worm.win32.Flame.a; this sample was signed December 28th, 2010 and went undetected until June 4th, 2012 (Bharadwaj, 2012).

Duqu arose around the time of stuxnet. It shared multiple similarities, and its complex nature led researchers to conclude that the authors of Duqu were the same as the authors of Stuxnet. The major differences between Stuxnet and Duqu were the payloads. Duqu and Stuxnet both targeted very specific entities with surgical precision. However, where Stuxnet was hunting for a specific target to deliver a malicious payload to cause physical destruction, Win32.Duqu was a reconnaissance tool that gathered information from specific targets. This malware sample had a valid certificate until its revocation in October 12, 2011(Symantec, 2011).

Shadowhammer is a supply chain attack that affected the Asus live update services. This attack went undetected until January 2019. This is yet another example of a malware sample that surgically targeted victims from an unknown pool of users. The Shadowhammer incident is yet another example of malware being signed by a valid entity and thus going undetected for an abnormally large period of time. This lasted from June 2018 to November of 2018 (CERT-EU, 2019).

Compromised or Insecure Certificate Authorities

Another major concern in the public key infrastructure is the threat of compromised and insecure Certificate Authorities. If a compromised certificate leads to signed malware samples, what risk does one undertake when an entity is able to issue their own valid certificates through a TTP(Trusted Third Party)? The risk is increased greatly. This is especially true when there are efforts to cover up breaches. In a system that is difficult to audit, the risk of successfully covering up a breach for a long period of time is increased drastically. This again is a concern with the visibility within the Public Key Infrastructure, because this means the ability for one to audit this network is exceedingly difficult. This section shall address instances in which a certificate authority either did not practice security or was otherwise compromised by some means.

Comodo is a certificate authority that was compromised in August of 2011. Malicious certificates signed by Comodo were discovered in the wild. The hacker, who went by the name "ComodoHacker," claimed responsibility for the incident, for the breach, as well as, for a number of other attacks(Symantec, 2012, p.23).

Diginotar is yet another example of a certificate authority that was compromised. July 10, 2011 marked the beginning of the end for the Dutch certificate authority formally known as Diginotar. Intruders were able to issue a series of invalid certificates. It went undetected until the 19th of July. Though the Dutch-based company was able to revoke most of the certificates, there were a few key certificates that were not revoked during the emergency. Without any public announcement, Diginotar assumed the posture of business as usual. It wasn't until August 28th, 2011 that the true impact of the events would begin to unravel as a rogue certificate that was not revoked was found being used to actively attack a gmail user. That was a glimpse of what was to come. 531 certificates were issued for various domains. On September 2, 2011, Diginotar was removed from being a trusted root authority, as a result, the company filed for bankruptcy 18 days later (der Meulen, 2013).

Digicert Sdn . Bhd. is not related to the digicert root certificate authority, This subsidiary certificate authority did not get compromised, and no invalid certificates were issued. This was a certificate authority that was removed as an authority due to their blatant lack of good security practices. Their issuance of weak certificate keys was a large factor in their revocation.

Corruption

The entities that are entitled with providing trust to this mechanism are the very entities that should not be trusted. This is exactly what occurred in 2009 when the Emirates Telecommunication group, Etisalat, pushed a network update to blackberry devices in 2009. At the time, this was a trusted subordinate certificate authority.

The malware compromised approximately 100,000 customers (Peter Eckersley, 2010a). The problem stems once again from a lack of visibility within the public key infrastructure. Without the proper mechanisms to conduct audits against the Public Key infrastructure, this sample likely would have went largely unnoticed. The spyware was noticed due to implementation flaws in the sample itself, which led to noticeable performance differences (Mayanak Aggarwal, 2010, p.23).

Problems with Current Model

The problems of the system are obvious, the main one being a lack of visibility within the public key infrastructure. One only needs to consider the events that have unfolded to ascertain this problem's existence. Attempts, up until this point, to create visibility in the public key infrastructure have failed. Malware samples are still spreading in the wild with valid code signing certificates, as is the case with the recent Shadowhammer malware sample discussed earlier in this section. Throughout the remainder of this document we shall discuss the pitfalls that exist in Google's currently implemented solution known as Certificate Transparency.

Aftermath, A Failed System

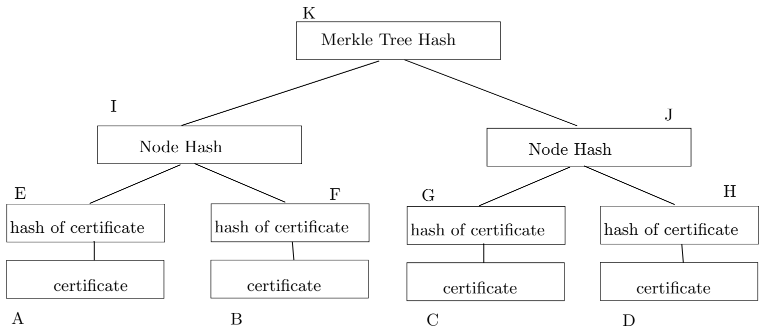
There were a series of events that took place after the incidents that happened between 2009 and 2011. The reactions addressed a multitude of problems and attempted to bring much needed transparency to the public key infrastructure, however, not all of the problems have been solved. The most recent event, Shadowhammer, occurred in 2019. This section will address the certificate transparency system implemented by Google, as well as the short-comings of that solution.

Certificate Transparency

Certificate Transparency is an attempt to alleviate some of the failings of the public key infrastructure by creating a publicly visible log of issued certificates that are easier to audit. It solves multiple fundamental problems and removes a large amount of the burdens that CAs are entrusted with when maintaining a Certificate Revocation List (CRL). Certificate Transparency adds three entities to the public key infrastructure: Certificate Logs, Monitors, and Auditors (”what is Certificate Transparancy”, N.D).

Certificate Logs are an internet service that collects records individually. Each record is a Cartographic proof containing multiple certificates. Merkle Hash Trees (MHT) leads to an append only record, and the head of each tree is signed by the Certificate Log. Any certificate can be validated against a log proof given the entity validating the log has access to the Certificate Log.

Figure 4: Merkle Hash Tree



Let's say one wishes to validate that certificate B is a member of the log. First one would compute the hash $H : \{B\}$ to find F, then compute the hash $H : \{E\&F\}$ to find I. Once one computes the hash $H : \{I\&J\}$, they would check that the resulting value equates to the hash value signed by the server running the certificate log(”what is Certificate Transparancy”, N.D).

Monitors are servers that periodically post requests to certificate log servers to collect and validate logs. These monitors look for abnormalities within certificates. Monitors aid in the certificate transparency mechanism by notifying the certificates issuer of the new certificate or looking for other signs of a fraudulent certificate that has been issued to the network. This is one of two mechanisms that help to add visibility to the public key infrastructure (”what is Certificate Transparancy”, N.D).

Auditors operate as a lightweight application. These applications are suited to be run on the client side, but may also be ran by certificate authorities and other servers. These applications are designed to preform two distinct functions. First an auditor can validate that a particular log is behaving correctly. This is achieved by submitting signed certificate time-stamps (SCT's) to a log. The auditor then may query the log further to determine if the certificate and SCT has been added to the log. Certificates that have not been added to the log may be deemed suspect and an SSL client may refuse to connect to that service (”what is Certificate Transparancy”, N.D).

Split World Attack

The security mechanisms that were added to the public key infrastructure may have aided in preventing malicious certificates from being long-lived, but CT has its own unique security flaws that should be considered thoroughly. The threat landscape is changing, and one should consider the dangers of sophisticated technical adversaries, entities capable of creating targeted malware samples such as Stuxnet, and Duqu. Certificate revocation has always been the intrinsic flaw unique to the public key infrastructure and it is revocation that presents a problem to the Certificate Transparency mechanism as well.

Consider an instance where a technical adversary acquires a fraudulent certificate from a well meaning certificate authority, the certificate is valid but the entity controlling the certificate is malicious; at this point the certificate is submitted to a CT log by either the malicious entity, or the certificate authority. The malicious entity then acquires an SCT for that certificate. The malicious entity now has everything necessary to conduct a split world attack. When the victim tries to connect to the CT log to validate the certificate the malicious entity would intercept that request and returns the fraudulent certificate and the invalid SCT. The client would validate even if the certificate was revoked at a later point in time. This is due to the fact that the log still maintains the Merkle proof for that specific SCT, the client has no way to infer that they are retrieving an invalid view of that log (Murat Yasin Kubilay & Mantar, 2018, p.2-p.3).

The dangers of compromised certificates are obvious, Shadowhammer has shown that long lived malware samples are still prevalent on the internet; the split-world attack represents a fundamental flaw in Certificate Transparancy, and in itself, CT was an attempt to fix fundamental issues that underpin the public key infrastructure.

The Public Key Infrastructure and Blockchain Solution

The public key infrastructure has serious flaws, however there is a number of flaws that can be solved by transitioning from the current public key infrastructure to a mechanism that utilizes blockchain. It shall be demonstrated through the course of this subsection that blockchain is indeed a suitable candidate for use in the public key infrastructure. This will be accomplished by utilizing a decision based flow chart used by the Department of Homeland Security(DHS) to determine if blockchain is a suitable solution to a particular problem. The flow chart breaks down into six key questions that should be considered thoroughly when deciding if a blockchain solution is appropriate in a particular use case. These questions will be discussed thoroughly throughout the remainder of this subsection; a direct emphasis will be directed at how these properties uniquely address the problems that are inherent to the public key infrastructure (Dylan Yaga & Scarfone, 2018, p.42).

”Do you need a shared consistent data store?”

Prior to Googles Certificate Transparency (CT) network there was not a straight forward method to audit logs. This is what led to the massive failings that this document has explored up until this point; furthermore, it has been demonstrated that an attacker may manipulate the view a victim has of a log by means of a split world attack. In such instances, the security mechanisms of CT disintegrate rapidly. Fragmented views of the data store have led to a system that cannot be maintained in a trivial fashion. These failings are inherent to the security mechanisms that arbitrate trust on the internet, and such failings cannot be accepted. A shared consistent data store is the very property that makes blockchain such an appealing candidate to replace the current trust mechanism.

"Do more then one entity have to contribute data?"

The public key infrastructure is massive, consider once more the findings of the Electronic Frontier Foundation (EFF), In 2010-2011 there existed 124 root certificate authorities, furthermore there were 651 organizations capable of signing certificates. The internet has grown rapidly since that point in time, and deriving a number to describe the magnitude of organizations trusted to issue certificates is not a trivial task, if it is even possible, furthermore, the techniques used to map this network out are obsolete by this point in time. All of these entities would be valid contributors to the data set, but the problem we are addressing in this document is one of visibility, and multiple entities are needed to provide integrity to the trust mechanisms being relied on.

Blockchain works best in large user pools. One user may post fake but valid certificates to the log, but due to the multitude of honest nodes it is more likely that the majority consensus will outweigh the malicious node, and this is a key factor to consider. The public key infrastructure is a conglomeration of trusted third parties that cannot always be trusted. This document has discussed multiple instances in which the trusted third party was either Compromised, like Comodo and Diginotar, or Corrupt like Etisalat; the intrinsic properties of blockchain require more then one entity; Blockchain is a technology that works under the assumption that malicious entities are already trying to manipulate the ledger. Decentralization of contributors adds resiliency to the blockchain mechanism (Murat Yasin Kubilay & Mantar, 2018, p.6).

"Data records once written, are never updated or deleted?"

Blockchain creates a ledger that is immutable, and this is due to the mechanisms discussed previously. Recall that a block in blockchain has three main components: The hash of the previous block, the datum field, and the cryptographic nuaunce. Denote the bitmap of the entire block as a set B_n and let the hash of the previous block be denoted as h_{n-1}; one may deduce trivially that $h_{n-1} \subset B_n$. The result is a chain of blocks, where each block is mathematically dependent on the previous block.

$$\forall k, 0 \leq k < n, H : \{B_{n-(k+1)}\} \subset \{B_{n-k}\} \tag{10}$$

Each block contains a cryptographic nuance which makes it computationally trivial to validate a block once constructed, however calculating a cryptographic nuance for a block is computationally expensive. This property ensures that the modification of prior records is computationally infeasible, thus, preventing an attacker from arbitrarily generating false entries within the chain. However, it also prevents that entry from being deleted and/or updated once added to the ledger. This subsection shall show that the ability for one to update and/or delete records from the public ledger would not be of any consequence to the security mechanisms that this document is addressing. The revocation and/or updating of Certificates can be accomplished by submitting future transactions to the ledger, furthermore, one may conclude trivially that an immutable ledger would aid in the auditing of TTPs.

The solution this document shall primarily focus on is that of CertLedger. CertLedger utilizes a mutable data section that is managed by an immutable smart contract. The contract contains code that assists clients in validating the certificate at that blockchain address. The smart contract also contains code to manage the state of a certificate. When a specific condition is met the smart contract would automatically update the mutable component of the contract appropriately(Murat Yasin Kubilay & Mantar, 2018, p.7).

”Sensitive identifiers will not be written to the data store?”

The Department of Homeland Security specifies that sensitive identifiers should not be written to a data store; this is a vital consideration. Once a record is written to a data store, it becomes a permanent member of the data store. Being a publicly visible ledger, sensitive identifiers in the ledger would represent a large concern in terms of both security and privacy (Dylan Yaga & Scarfone, 2018, p.42).

In terms of CertLedger, no sensitive identifiers need to be written to the data store. The primary component to be transacted to the data store is the smart contract code. The smart contract acts upon a mutable document to validate that the certificate is still valid, and/or to update a certificates revocation status. This information is already public knowledge, utilizing this form of public ledger enables auditing and revocation to be seamless and efficient.

”Are the Entities with write access having a hard time deciding who should be in control of the data store?”

Beyond the ability for one to control what entities may contribute to a permissioned blockchain, there is no centralized authoritarian figure to write to the ledger. Currently the public key infrastructure has multiple entities whom produce fragmented logs; blockchain would solve this; all entities would be forced to collaborate and agree on the same view of the log (Dylan Yaga & Scarfone, 2018, p.42).

”Do you want a tamper proof log of all writes to the data store?”

Blockchain creates a ledger that is secured by cryptographic mechanisms. The mechanisms ensure that any modification of the log is computationally expensive, thus any attempt to modify the log would result in entries that are not computationally valid members of the log, furthermore all entries written to the ledger would become permanent members of that ledger, and any attempt by an attacker to modify and or present a false log would be detected by trivial means. Though situations exist in which a sophisticated adversary may manipulate the ledger by means of a ”51% attack,” the attack itself is beyond the computational(or financial) viability of most adversaries. A sophisticated attacker may more readily execute a Split world attack on the current CT model. In terms of concern, the permissioned PBFT+PoW model one relies on in the proposed CertLedger protocol would make such attempts to manipulate the ledger extremely apparent; since attempts to manipulate the ledger are evident within a permissioned system, it would be a trivial task to remove the offending node from the network (Dylan Yaga & Scarfone, 2018, p.34).

The public key infrastructure would benefit greatly by constructing a single tamper evident/tamper proof log of all writes to the data store. The auditing of such a network becomes a trivial task. any entity that is acting in a manner that compromises the integrity and or security of individuals acting on the internet would quickly be detected. A tamper proof log of all writes to the data store is desired for the public key infrastructure.

Conclusion

This document has discussed the fundamental mechanisms which enable trust on the public key infrastructure, as this document progressed the multitude of failings that are inherent to the public key infrastructure have been thoroughly demonstrated. It has been demonstrated that, due to the fragmentation of lofs, the public key infrastructure represents a network that cannot be easily audited or maintained. This intrinsic flaw was the primary reason that malware samples were discovered months or years after being signed by valid code signing certificates. This is a very real problem that persists up until this point in time, one only need to look at the lifespan of the shadowhammer incident to conclude that the problem is persistent even after the initiation of the Certificate Transparency (CT) solution designed by Google. This document has described an attack method that negates the security mechanisms of CT and the primary flaw being fragmentation of logs. It has been demonstrated that, by use of blockchain technology, one may mitigate these attack vectors and construct a ledger that is far easier to maintain, and audit then existing solutions.

References

Afshar, R. (2015). Digital certificates(public key infastructure).

Ali Davanian, J. H. W., Amit Kumar Gupta. (2016). Man in the middle attacks.

Barker, E. (2016). Guideline for using cryptographic s tandards in the federal government: Cryptographic mechanisms. *NIST: National Institute of Standards and Technology*.

Bharadwaj, P. (2012). *Flame malware exploits microsoft's digital certificate.* Retrieved from `https://www.symantec.com/`

CERT-EU. (2019). Operation shadowhammer - compromised asus computers. *CERT-EU*.

der Meulen, N. V. (2013). Diginotar: Dissecting the first dutch digital disaster.

Dylan Yaga, N. R., Peter Mell, & Scarfone, K. (2018). Blockchain technology overview. *NIST: National Institute of Standards and Technology*.

EFF. (2011). *Color map of ca's.* Retrieved from `https://www.eff.org/`

Mayanak Aggarwal, K. Y. (2010). *Blackberry proof-of-concept: Malicious application.*

Murat Yasin Kubilay, M. S. K., & Mantar, H. A. (2018). Certledger: A new pki model with certificate transparency based on blockchain.

Nicholas Falliere, E. C., Liam O Murchu. (2011). W32.stuxnet dossier. *Symantec*.

Peter Eckersley, J. B. (2010a). *Defcon 18: Observatory for the ssliverse.*

Peter Eckersley, J. B. (2010b). *An observatory for the ssliverse.*

Richard D. Kuhn, T. W. P. S.-J. C., Vincent C. Hu. (2001). Introduction to public key technology and the federal pki infastructure. *NIST, National Institute of Standards and Technology*.

Symantec. (2011). W32.duqu the precursor to the next stuxnet. *Symantec Security Response*.

Symantec. (2012). 2011 trends. *INTERNET SECURITY THREAT REPORT*.

Symantec. (n.d). *Certificate pinning.* Retrieved from `https://www.symantec.com/`

"what is Certificate Transparancy". (N.D). *What is certifificate transparancy.* Retrieved from `t`